1992

A Reconfigurable Multicomputer System: Implementation And Performance

Paul Anthony Smeulders

A Reconfigurable Multicomputer System: Implementation and Performance

Volume I

by

Paul Anthony Smeulders

Faculty of Engineering Science
Department of Electrical Engineering

Submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy

Faculty of Graduate Studies
The University of Western Ontario
London, Ontario
July 1992

ISBN 0-315-75380-3

Abstract

Architecture of the computer has always been dictated by the attribute considered the most important during its design. The classical von Neumann computer system was the consequence of a quest for reliability; the modern multiple-processor architectures have resulted from a search for performance. While the modern system's existence is based on its relative reliability, its efficiency depends upon the match between the problem's inherent structure and the system's architecture. The architecture of a typical multiple-processor machine is fixed: essentially it either supports simultaneous, or sequential task solutions with efficiency. Since most computer-tractable problems are neither purely parallel nor serial, the machine whose architecture can be configured to match the problem would offer higher performance in a broader range of applications than its fixed-architecture counterpart. Clearly, a real-time reconfigurable machine would then provide an optimal problem-architecture match.

This thesis introduces a research-oriented, high performance, Reconfigurable Multicomputer System (RMCS), which combines versatility, connectivity, and incremental expandability. The prototype system that has been designed, implemented, and characterized, comprises an elemental cell of four autonomous processors (slaves), an Interprocessor Communication Network (ICN) that supports the reconfigurability, and a network controller module. A supervisory processing unit (the master), provides the slaves with instructions and data for a task, and synchronizes their activity. The ICN features a number of Programmable Signal Routers (PSR) which were designed and fabricated at the silicon chip level to implement the unique architecture. The ICN provides unidirectional and exclusive data communications among the processors. The system is expandable by replicating the elemental cell; a single master is employed for systems of any size. The chosen architecture warrants eventual VLSI implementation of large multi-celled systems.

A prototype of the reconfigurable multicomputer has been built and the characteristics critical to its performance and future optimization determined. The PSR electrical behaviour, interprocessor data communications, synchronization overhead, and computational performance have been tested. The performance tests, chosen to encompass typical applications, include matrix operations, Fast Fourier Transform computations, frequency domain filtering, and alternating series calculations. The tests utilize various computational and control strategies, which exploit the system's reconfigurability and demonstrate its efficacy. The performance advantage of the RMCS architecture is compared to that of a congruent uniprocessor, executing the same task, yielding thus a speed-up factor as the main measure of the performance increase. The experiments evince the effect of the ICN and control strategies on overall performance. Results may be scaled for systems of different size and sophistication.

The RMCS performance speed-up is task, problem-size, and control-strategy sensitive. Nearly ideal speed-up factors were obtained for parallel matrix multiplication, as well as for alternating series computations using parallel/pipelined mode. Parallel FFT tests yielded speed-up factors from 2.7 to 4.2 for 256-point complex series. The frequency domain filtering experiments took greater advantage of the system's reconfigurability, resulting in improved performance. The experiments provided sufficient data to develop an accurate performance model.

The study established the viability of the reconfigurable multicomputer architecture and demonstrated its advantages in scientifically-oriented computations.

Acknowledgements

The author would like to thank his Chief Advisor, Professor Z. Kucerovsky, for his guidance, enthusiasm, and extraordinary patience throughout the period of study. Professors I.I. Inculet and W.D. Greason contributed significantly by serving on the advisory committee, and for this, the author extends gratitude. Much appreciation is due to the graduate students of Physics and Astronomy Room 109, who not only contributed to the success of the project, but enhanced the student-life experience by being good friends throughout. They are: Kenneth Chum, Adrian Li, Jonathon Reis, Dave Stewart, Frederick Tang, and Anthony Wu. Dave Sutton's contribution cannot be overstated, and the author is forever in his debt. Finally, the author would like to thank his family and friends for their support and patience.

The author appreciates the financial support provided by NSERC throughout the period of research.

Table of Contents

List of Figures

Other Appendices (A-F) contain non-annotated diagrams (see table of contents).

List of Tables

Nomenclature

Units:

s	seconds
ms	milliseconds (10^{-3} seconds)
µs	microseconds (10^{-6} seconds)
ns	nanoseconds (10^{-9} seconds)
V	Volts
VDC	Volts Direct Current
MHz	Megahertz (10^{6} cycles/second)
bit	Binary Digit
byte	8 bits
word	2 bytes; 16 bits
longword	2 words; 4 bytes; 32 bits
kbyte	kilobyte; 1024 bytes
Mbyte	Megabyte; 1024 kilobytes

Acronyms:

ACC*	Data Accept signal, active low
ACIA	Asynchronous Communications Interface Adapter
APC	Autonomous Processor Cell
ASCII	American Standard Code for Information Interchange
CCW	Configuration Control Word
CIF	Caltech Intermediate Format
CMC	Canadian Microelectronics Corporation
CMOS	Complementary Metal-Oxide Silicon
CPU	Central Processing Unit
DAV*	Data Available signal, active low
DFT	Discrete Fourier Transform
DMA	Direct Memory Access
ESC	Escape Character (ASCII code $1B)
ESD	Electrostatic Discharge
ESM	Expected Service request Mask
FCFS	First-Come, First-Served

FFT	Fast Fourier Transform
FLOPS	Floating-point Operations Per Second
I/O	Input/Output
ICN	Interprocessor Communication Network
IEEE	Institute of Electrical and Electronics Engineers
ILLEGAL	Illegal instruction, MC68008
LSI	Large Scale Integration
MIMD	Multiple Instruction stream, Multiple Data stream
MIPS	Millions Instructions Per Second
MPU	Microprocessing Unit
MSB; LSB	Most Signficant Bit (Byte); Least Significant Bit (Byte)
NCC	Network Controller Card
NMI	Non-Maskable Interrupt
OE	Output Enable
PE	Processing Element
PGA	Pin Grid Array
PI/T	Parallel Interface/Timer
PSR	Programmable Signal Router
RAM	Random Access Memory
RFD*	Ready for Data signal, active low
RISC	Reduced Instruction Set Computer
RMCS	Reconfigurable Multicomputer System
ROM	Read Only Memory
RTE	Return from exception instruction, MC68008
RTS	Return from subroutine instruction, MC68008; or Request to Send, RS-232C standard
SIMD	Single Instruction stream, Multiple Data stream
SRQ; SRQbus	Service Request; Service Request bus
SRQACK	Service Request Acknowledge
TRAP	Trap instruction, MC68008
TTL	Transistor-Transistor Logic
VCC, VDD	Power supply voltage, +5 VDC
VLSI	Very Large Scale Integration
VSS	Power supply voltage, 0 VDC
WSI	Wafer Scale Integration

Conventions and Operators

$	Precedes a quantity which is specified in the Hexadecimal (Base 16) number system. Valid digits are 0-9 and A-F inclusive.
Asserted	Logically active state.
Negated	Logically inactive state.
Low	A low TTL Voltage, <0.8 V.
High	A high TTL Voltage, >2.4 V.
SIGNAL	Indicates that signal denoted as SIGNAL is a logically active high signal.
SIGNAL*	Indicates that signal denoted as SIGNAL* is a logically active low signal.
(SIGNAL)'	Indicates that signal denoted as SIGNAL is logically complemented.
A+B	A OR B (Logical operator OR).
A•B	A AND B (Logical operator AND).
O(expr)	$f(n) = O(g(n))$ if $f(n) \leq Kg(n)$ for some fixed number K and n sufficiently large.
A mod *B*	the modulo function; returns the remainder of A/B.
⌊*expr*⌋	floor operator: returns greatest integer less than or equal to *expr*.

1 Introduction

This chapter describes the motives that underlie the presented study, and briefly discusses modern computer architectures. The last part of the chapter defines the goals, objectives and scope of the thesis.

1.1 General Introduction

Since scientists first established that nature could successfully be described using mathematics, they have sought machines with which to diminish the time and labour required to complete the characteristically complex computations. Pascal built a mechanical device for addition and subtraction in 1642; Leibniz (1646-1716) developed a system which could also multiply and divide. Charles Babbage (1792-1871) is perhaps the best-known pioneer of the mechanical calculator for his *difference* and *analytical engines*. By the early twentieth century, electromechanical calculators were common (Wulforst 1, 82) (Tanenbaum 1, 90).

The vanguards of electronic computing were scientists and mathematicians dissatisfied with the performance of mechanical devices in solving problems of meteorology, spectroscopy, geophysics, and geomagnetics. J.V. Atanasoff developed an arithmetic unit in 1939, to aid in solving partial differential equations. The super-secret *Colossus*, dedicated to cryptanalysis, was used by the British to decipher German Enigma-encrypted communications as early as 1943. The team of J.P. Eckert and J. Mauchly are widely recognized for building the first general-purpose electronic computer, the ENIAC. It was to be used for calculating weapons firing tables during World War II, but the war ended prior to the machine's completion. ENIAC was configured to perform various functions by physical rewiring. Von Neumann proposed the first stored program machines, wherein a sequence of *orders* was stored in the same *memory organ* as the data upon which they operated. The principles he put forth have survived, and to this day, we refer to a certain type of machine using his name (Shurkin, 84) (Hodges, 83) (Wulforst 2, 82) (Von Neumann, 45) (Burks, 46).

Since the invention of the transistor, integrated circuit, and microprocessor (1948, 1959, 1971, respectively), electronic computing has become dominant in information processing. In a competitive marketplace, the original 4-bit microprocessor used mostly in calculators, has grown to 32-bits, with performance and capabilities rivalling that of earlier mainframe systems. Today, not only scientists and mathematicians, but people in business, bankers, and publishers make use of the technology; a *personal computer* or video display terminal is a familiar fixture on any productive desktop. Household appliances and automobiles use computers which are effectively hidden from the user, yet they are considered to be necessary. Sporting, recreational, and medical machines are used to directly enhance human health, well-being and performance. It is hard to find a human endeavour that would not be irrevocably altered by the computer.

The computer not only is a tool for solving complex scientific problems, it has also spawned new disciplines such as graphics and image processing, simulation, language design, fractal geometry, chaos theory, neural networks, artificial intelligence, and knowledge-based systems. The new tool has provided humans with the ability to explore new concepts at an accelerated rate. The computing machine has truly transgressed its original purpose, manipulating not only numbers, but also letters, symbols, and ideas.

As the power and performance of computing machines increases, the scope of their utility expands. Man's capacity for knowledge exceeds the rate of technological advances; our capability for proposing increasingly complex problems exceeds the ability of current systems to solve them within reasonable time. For this reason, the search for faster machines is one of the most challenging quests of our time.

While mostly adhering to the classical computer architecture, a wide variety of strategies aimed at increasing their performance are being studied, proposed, and implemented. Reduced feature-size in integrated circuits, new materials, higher clock frequencies and hierarchical memory structures are used extensively in state-of-the-art

uniprocessor systems. Phenomena such as wave reflection and signal propagation speed place limits on the ultimate performance of these machines, and optical rather than electrical systems are being researched (Stone, 91) (Sawchuck, 87).

Alternative, non-classical approaches to the problem of increasing computer performance often utilize several, cooperating processing units. A typical *multiple processor system* is organized into identical or dissimilar cells possessing a varying degree of complexity and autonomy, that allows the system to execute the given task in less time than with a single cell. The task allocation, interconnection and synchronization of the various units are important factors affecting the performance of such systems. A problem which is partitionable into identical sub-problems, each with independent data, may be solved by processing elements simultaneously, in *parallel*. A problem characterized by sequences of dependent operations may be performed efficiently by a *pipeline* of specialized units. Today, it is rare to find uniprocessor designs which do not incorporate some degree of pipelining or parallelism in their architectures.

To some extent, all machines, including uniprocessor *general purpose* systems, are designs which strike a compromise among a number of factors, such as arithmetic performance, memory addressing, looping efficiency and subroutine linkage. The interaction of these factors ultimately determines the scope of problems for which the machine is most adept. Likewise, for multiple processor systems with fixed interprocessor connections, the range of applicability is sacrificed to favour optimum performance in a particular class of problem.

The presented thesis proposes and studies a *reconfigurable computer architecture*, that enables a broader class of problems to be solved efficiently by a single machine. The architecture can be altered while the problem is being solved, while the principles of pipelining and parallel processing are exploited to various degrees, according to the attributes inherent in the application. The dynamic nature of the system architecture not only allows

an optimal topology to be used in solving a large problem, but its sub-problems may themselves be executed in distinctly different configurations. The configurations used for a given problem may be entirely predetermined by the programmer according to inherent problem structure, or they may be dynamically determined according to intermediate conditions. That is, subsequent optimum processor configurations may be dependent upon the results of a previous computation. The thesis experiments focus upon the former case. The structure of the machine is proposed with future VLSI implementation in mind; a cellular design is stressed. Expanded systems may be realized through replication of the basic cell.

The remainder of this chapter discusses classical and non-classical computer architectures, problem classification, and the match between the problem and machine. The reconfigurable cell concept is described in further detail, the prototype system architecture is presented, and the objectives and original contributions of the thesis are stated.

Chapter 2 addresses issues of computer performance measurement and characterization. The algorithms used to examine the system's performance are described in general terms, and maximum theoretical performance increases are calculated for each. Descriptions of the actual performance test programs are deferred to Chapter 4, since they are easier to appreciate after the system's hardware and control software are reviewed.

Chapter 3 describes hardware and software implementation details of the prototype Reconfigurable Multi-Computer System (RMCS), encompassing the autonomous processor cells, programmable signal router devices, interprocessor communication network, and system control methodologies.

Chapter 4 presents a detailed description of the various tests used to characterize system performance, as well as the results. Chapter 5 presents analyses of the performance tests and conclusions of the presented study, with emphasis on the performance advantages offered by the RMCS over similar, uniprocessor applications. The thesis concludes with recommendations for future research.

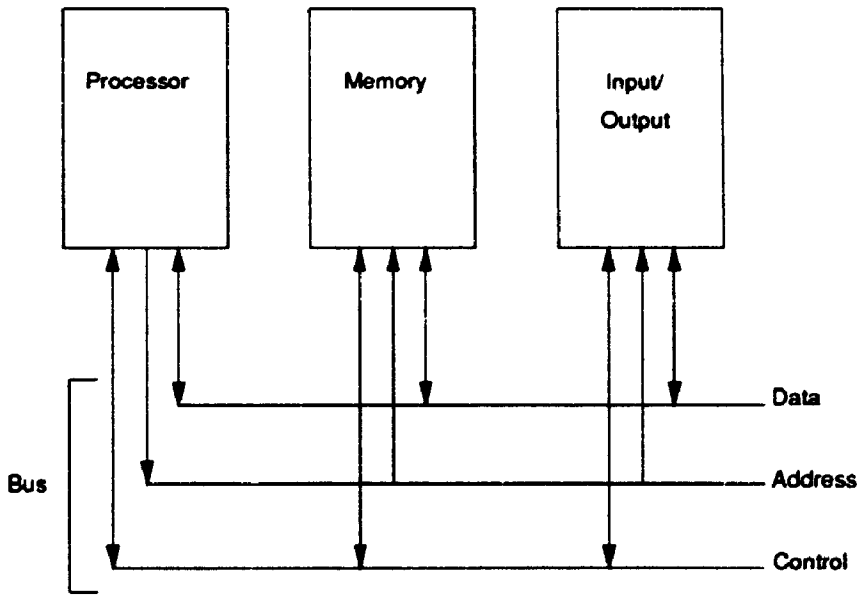
1.2 Von Neumann Class Machine

The most common computing systems have their foundations in an architecture proposed by János (John) von Neumann in the mid 1940's. Fundamentally, he proposed storage of a computation's instruction sequence in the same memory unit as the data upon which they operate. Instructions and data were indistinguishable, and were accessed identically by a control unit. The control discerned instructions from data according to its current state. The system did not require rewiring for each new application, since re-programming was facilitated by writing a different sequence of instructions into the memory. Computations consisted of arithmetic operations carried out in the arithmetic unit and accumulator, and modifications to memory contents. Memory cells of 40 binary digits were individually addressable. The control encoded the binary address on a set of signals linking the memory, where they were decoded to select the cell whose contents were to be read or over-written. Address information for instructions was integrated with the operation codes. The repertoire of basic operations included addition, subtraction, multiplication, division, data movement between memory and control registers, absolute value, negation, bit shifting, and both conditional and unconditional branching of control (Von Neumann, 45) (Burks, 46).

Interaction between the computer and its operator utilized an Input/Output (I/O) system consisting of a modified teletypewriter, magnetic wire storage units, and a set of viewing tubes. Von Neumann linked the control and I/O via the machine's accumulator. He required additional, dedicated instructions to perform I/O operations, and observed that computation

and I/O could not be concurrent. Von Neumann's early papers on computers discuss arithmetic and memory access in both bit-serial and bit-parallel forms, hierarchical memory structures, instruction tracing mechanisms, and fault tolerance through redundancy. What is commonly known today as a "von Neumann" class machine is characterized by a single central processing unit (CPU), and a memory system linked by a *bus* of address, data, and control signals. The I/O system is connected to the CPU in a similar manner, often utilizing the same bus. The organization is inherently sequential; only a single transaction may occur between bus-attached units at any one time. Figure 1.2.1 illustrates the structure of typical systems.

Figure 1.2.1 Organization of von Neumann class machine.



1.2.1 Technological Feasibility

Von Neumann's designs were technologically feasible at the time of their proposal, despite their complexity. Bit-parallel memories were made possible by development of the Selectron tube. Since then, magnetic core and semiconductor memories have had considerable influence on both the power and economics of computing devices.

Other technological advances have led to increased performance of the basic von Neumann class machine. The processor unit has been integrated in silicon, with *cache* memory incorporated. Instructions still encode address information, but more flexible means with which to specify them have evolved (addressing modes). A number of modern processors have instructions which operate on multiple operands. Multiple bus masters, such as direct memory access controllers, are often used to improve the I/O and memory system transaction rates. The most common multiple processor systems available are extensions of the classical concept; a number of processors may access a memory or other resources via a bus interconnection, with arbitrated access to shared resources. Despite the fact that many of the improvements described encroach upon the fundamentals of the original concept, the majority of modern machines can still be classified as von Neumannian, and the term has become synonymous with bus-based organizations.

1.2.2 Advantages and Limitations

The von Neumann architecture's advantage lies in its overall simplicity with respect to hardware and programming. Many of its advantages stem from the technological advances made since von Neumann, the related economics, and the evolution of programming principles. Faster, smaller, and more energy efficient devices have made systems available with extremely high performance and sophistication. The shortcomings of the older

implementations are recognized and alleviated, new principles incorporated, but much of the original concept is left intact[1]. The basic sequential properties are perpetuated in today's most powerful microprocessors.

Although the classical machine owes much of its early feasibility and success to its simple, serial nature, that very attribute is at the root of its limitations. Often, a computation arises where a number of operations may be performed concurrently, since their respective operands exhibit varying degrees of independence. Higher performance may be achieved by systems which permit concurrent operations. Such systems generally comprise a collection of processing units of varying sophistication and interconnection topologies. Depending on the problem complexity and data interdependence, system configurations can be specified for optimal performance. High-performance system organizations and the match between the problem class and the architecture will be discussed in the following section.

1.3 High Performance Computers

The performance of computer systems may be enhanced using a variety of means. Most methods exploit some degree of parallelism, whether specific to algorithms for which the system is intended, or the lowest-level system operations. The strategies require not only additional resources, but also interconnection and control schemes which ensure that available resources are utilized efficiently. What follows is a brief discussion of high performance, multiple processor systems, and the methods employed to increase performance. A number of taxonomical classifications for multiple processor systems are

---

1 For examples, witness the evolution of microprocessor "families" from Intel and Motorola.

proposed in the literature. Terminology published elsewhere is used here to clarify characteristics of several machine types, without concern to arguments about the merits of the various author's classifications. (Flynn, 72) (Gajski, 85) (Skillicorn, 88) (Duncan, 90)

1.3.1 Problem Characteristics

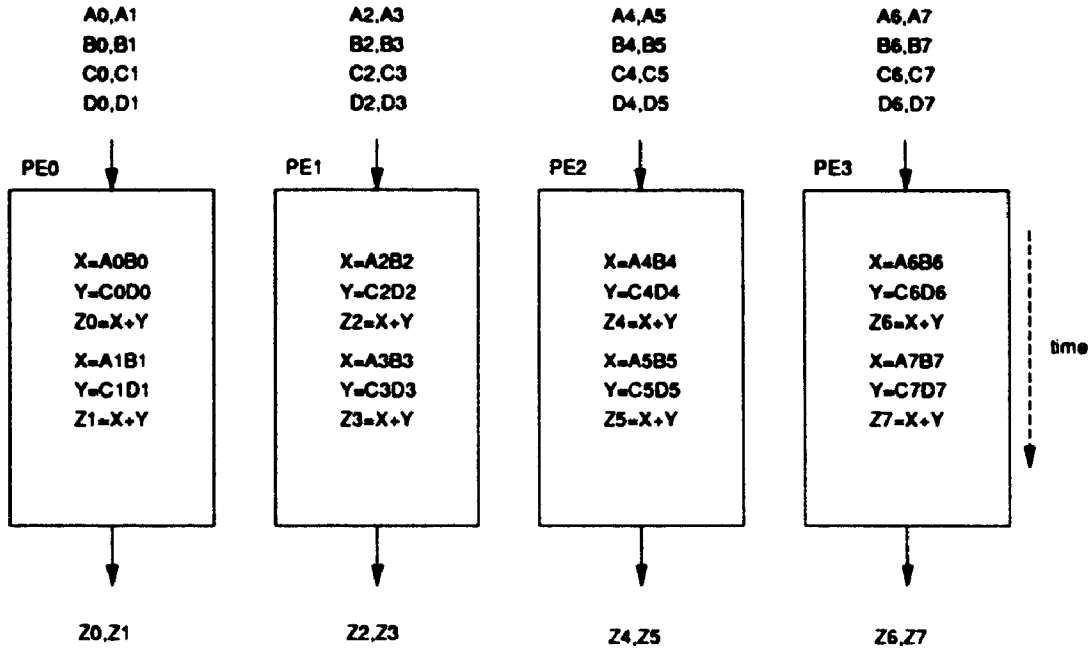
A fundamental concern in achieving high performance in a computer system is the method for distributing data among the functional units. The von Neumann machine accesses operands individually through a single channel, retaining intermediate results within the processing unit. Operations are sequential, utilizing one functional element at any given time; the data are routed to the appropriate element according to the current operation code. Given additional functional units and a system structure which fosters availability of operands to them, simultaneous operations may be conducted. However, most algorithms have intrinsic properties and data dependencies which make it necessary to perform computations in a certain order. Ultimately, algorithm-specific properties, functional element capabilities, and system control strategy are combined to determine the interconnections among elements, and the optimum system organization. The effect of the former two concepts may be illustrated by the vector calculation of equation 1.3.1.1.

$$Z_i = A_i \times B_i + C_i \times D_i \qquad i = 0, 1 \ldots N \tag{1.3.1.1}$$

The calculation of both product A_iB_i and C_iD_i must be completed prior to the addition step to form Z_i, however each Z_i is independent of any other input term and other Z_i. The calculation for each Z_i may be individually assigned to a *processing element* with both multiplication and addition abilities. If N is greater than the number of available processors, p, a number of calculations of Z_i must be performed sequentially in each. The elements require no direct interconnection nor communications with the other processors throughout the computation. Figure 1.3.1.1 shows a system of four elements, and the operations required to compute Z_i for i=0,1...7. The computation strategy is essentially

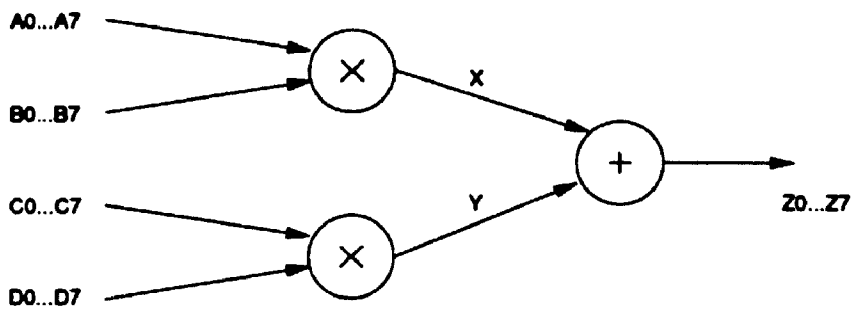
parallel.

Figure 1.3.1.1 Calculation of Z_i using unconnected, sequential adder/multiplier elements.



The computation may alternatively be accomplished by a system consisting of single-function processing elements. Each calculation of Z_i involves two multipliers and an adder. The algorithm structure and processor capability directiy stipulate the necessary interprocessor connections, since the output of each multiplier must connect to an input of the adder. Note also that while a multiplication is taking place, the adder is idle, and vice-versa. The multiplication of $A_{i+1}B_{i+1}$ and $C_{i+1}D_{i+1}$ may be initiated at the same time as the addition of A_iB_i and C_iD_i. Figure 1.3.1.2 shows computation of Z_i using two parallel multipliers in series with an adder. Computation is essentially *serial*, with operations occurring simultaneously on distinct problems to utilize the available resources efficiently.

Figure 1.3.1.2 Calculation of Z_i using single-function elements.



A hybrid implementation of the two organizations yields higher performance for the given problem for cases with $N>p$. A parallel system similar to that of figure 1.3.1.1 may be employed, where each processing element consists of a serial unit as in figure 1.3.1.2.

For the described problem, a number of system organizations were possible. Algorithms exhibit varying degrees of adaptability to different architectures. The simple example was selected to illustrate how the relationship between the processors, the interconnections among them, and the problem specification must be considered to optimize the calculation procedure.

Problem character also influences system control strategies, and the example is used to illustrate. In the parallel implementation, the instructions executed in each element are identical in every respect, and occur in the same order. The elements may be designed to respond to a common instruction stream, broadcast from some control unit. Alternatively, a separate instruction stream may exist for each element, permitting asynchronous operation. The common instruction stream permits high performance and simplified system control for this algorithm, however, flexibility of the system is compromised for algorithms wherein elemental instruction sequences are dissimilar. Since the system awaits completion of the slowest element at each step, maximized performance requires that the processors be

matched, and data-dependence of instruction execution time not exist.

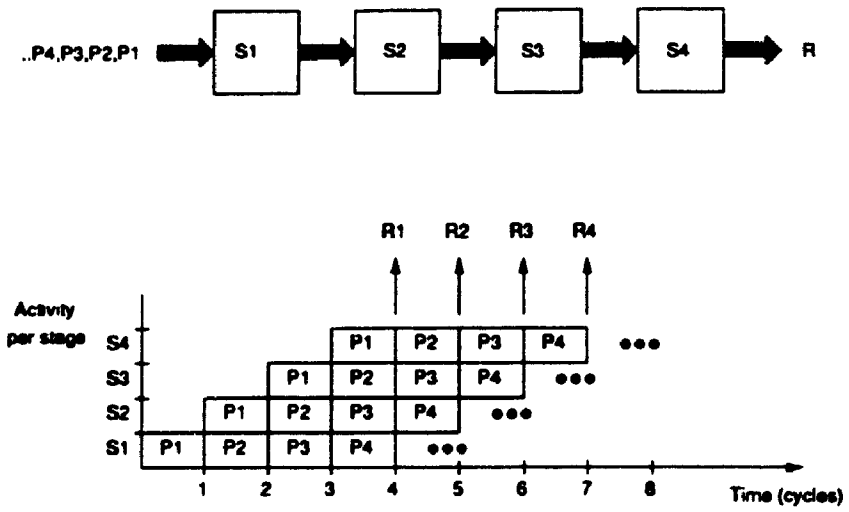
Algorithms may be classified in a number of ways, and terminology is largely context specific. Stone considers whether they are full information algorithms or not (Stone 1, 90); Hoshino distinguishes physical computations as being either continuum or particle models (Hoshino, 86). Locally and globally recursive are distinctions made by Kung (Kung, 1, 88) in the context of array processors. For multiprocessors and multicomputers, algorithms are categorized as synchronous, asynchronous, and macropipelined by Hwang (Hwang 1, 84), whereas Quinn adopts the terminology partitioned, relaxed, and pipelined for corresponding concepts (Quinn 1, 87).

Further clarification of the link between algorithms and architectures (interconnections and control strategies) is facilitated by discussion of various types of multiple processor systems in the literature.

1.3.2 Pipelined Systems

Pipelined systems utilize overlapped parallelism to achieve high performance. In its most general form, a sequence of commonly used operations is subdivided into its elementary operations, each performed by a separate functional unit. Data flows serially through the elements. The terminal elements have access to the external system, such as the memory. Since at any time a particular problem occupies only a single element, the remaining units are available to process subsequent problems, analogous to an assembly line. It is common for pipelined systems to operate synchronously, with a clock signal timing operations and transitions of data from one stage to the next. Optimal performance is achieved when stages complete their function in equal time; otherwise, performance is limited by the slowest element. Figure 1.3.2.1 shows a 4-stage linear pipeline, and a timing diagram to illustrate pipeline activity.

Figure 1.3.2.1 A 4-stage linear pipeline and its timing characteristics.



Throughput of a system is defined as the number of results completed per unit time. A linear synchronous pipeline will produce a result for every clock cycle once the pipeline has been filled; the time required to fill it is the product of its length and the clock period. The *speed-up factor* (or simply *speed-up*) for a system is defined as the ratio of the best serial-processor speed to the multiple processor system speed. If a serial processor requires a clock period, τ, for each elementary operation, then a sequence of k operations executed n times requires $nk\tau$ time to complete. The speed-up of a pipeline of length k is given by equation 1.3.2.1. It is clear that speed-up approaches k as n increases.

$$Speed\text{-}up_{pipeline} = \frac{nk\tau}{[k + (n-1)]\tau} \qquad (1.3.2.1)$$

Various types of pipeline exist, and they are generally classified according to the degree to which operations are subdivided. *Arithmetic* pipelines divide operations into elementary arithmetic steps: a floating point multiplication consists of exponent addition, fraction

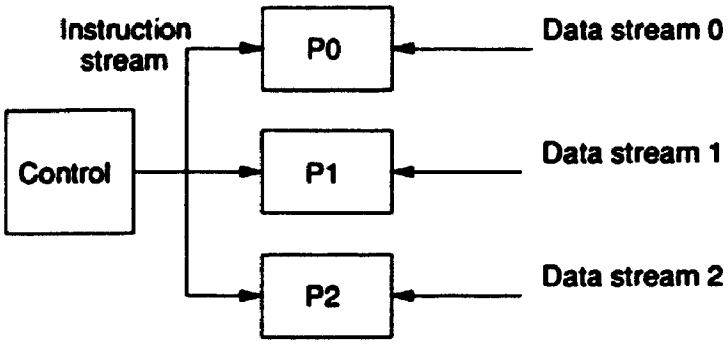
multiplication, normalization, and round-off stages. Often, the functional units in a pipeline may be utilized by a number of operations. Timing and control of *multifunction* pipelines is a significant problem, since optimal performance depends on efficient utilization of the available elements without conflicts between instructions (Stone 2, 90) (Hwang 2, 84).

Elementary operations common to all instructions may be pipelined as well (op-code fetch, decode, operand fetch, and execution). This form of *instruction* pipelining is common in modern microprocessors (M68020, 87) (i80386, 89). Efficient operation depends on the pipeline remaining full, and branch instructions cause temporary stalls. RISC architectures (Reduced Instruction Set Computers) implement a simplified instruction set with emphasis on internal register operations and limited memory referencing modes to permit increased efficiency in their instruction pipelines (A29000, 90) (Patterson, 82) (M88100, 88). Finally, *macropipelining* represents the opposite extreme, where many complex operations of a larger task are assigned to individual processors. While flexibility for problem sub-division is high, maintaining equal execution times among the processors in the pipeline can be challenging. Communication time may be significant, and thus each task-division strategy must strike a balance between minimized communications and equalized processor execution times.

1.3.3 Synchronous Parallel Systems

Various system organizations may be broadly classified as synchronous parallel. Flynn's taxonomy categorizes most of these as SIMD parallel systems (Single Instruction stream, Multiple Data stream). A number of processing elements respond to a single instruction stream broadcast by a control unit. The processors perform their operations on different data, which may be stored in local memories, global memory, or provided by the other processing elements via an interconnection network. Figure 1.3.3.1 shows a generalized SIMD machine organization.

Figure 1.3.3.1 Generalized SIMD parallel machine organization.



In its most simplified form, a parallel machine with k processors may conceivably provide a speed-up of k; however, interprocessor communications and algorithm mapping are important limiting factors. It is unusual for a computation to be entirely parallel, and some degree of serial operation is necessary to complete most problems. The serial operations are executed by the control unit, or a single processing element. The serial operations prevent indefinite increases in speed-up with increases in the number of processing elements. Amdahl's law (eq. 1.3.3.1) (Quinn 2, 87) expresses the phenomenon for a k-processor parallel system executing an algorithm with a fraction, f, of the operations necessarily sequential.

$$Speed\text{-}up \leq \frac{1}{f + (1-f)/k} \tag{1.3.3.1}$$

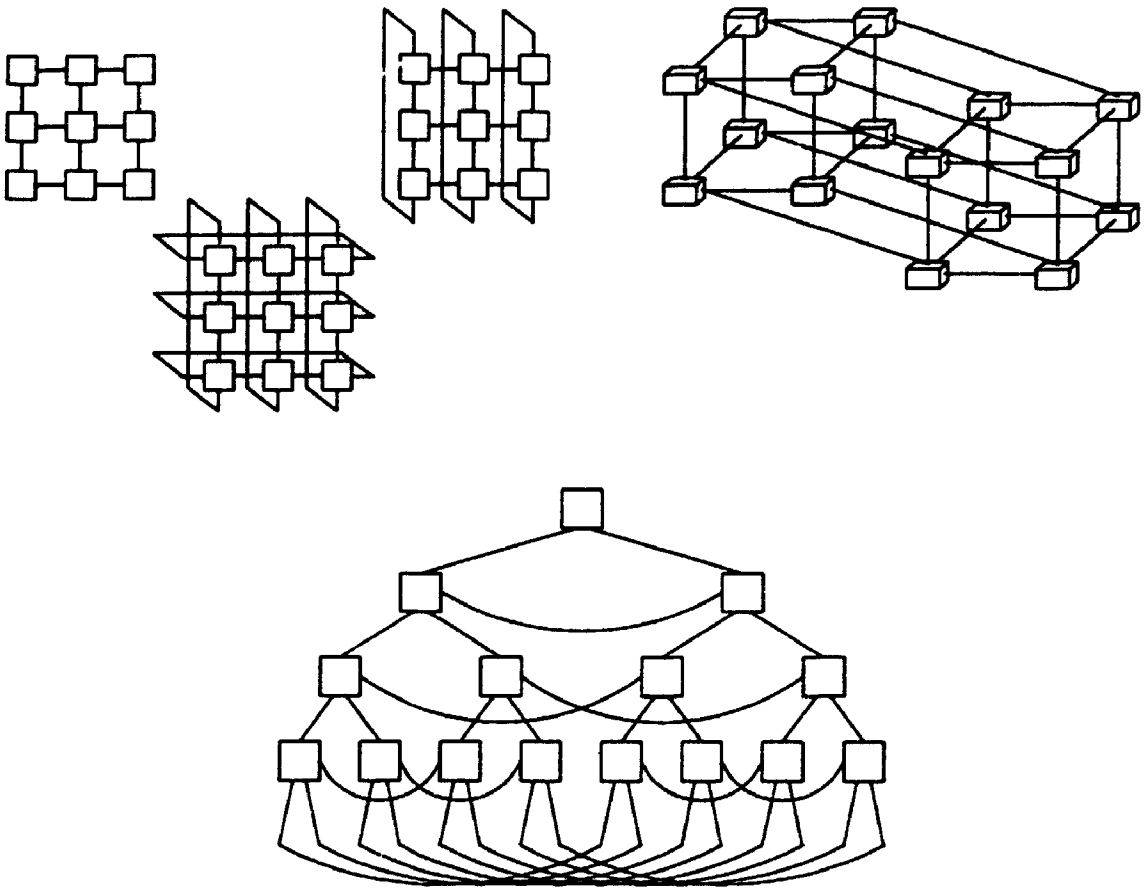
Since by definition, a uniprocessor algorithm requires no interprocessor communications, speed-up provided by a multiple processor system is diminished by communications time. Complete interconnection networks which link elements directly and independently to all others are optimum, since results produced by one element are readily accessed by all others. However, the number of links in a complete interconnection network grows as the number of processors squared, and the network cost quickly becomes

prohibitive. Numerous incomplete interconnection networks have been proposed and a number of systems implemented. Data produced by one element which is required by an indirectly linked element must be passed through intervening units. The advantage of matching an algorithm to the interconnection topology is evident: it serves to decrease communication time.

Interconnection networks may be either static, (unchangeable) or dynamic, where interconnections may be reconfigured in response to a control methodology. Section 1.3.5 discusses switched and reconfigurable systems in greater depth. A number of interconnection topologies are cited in the literature, such as mesh array, multidimensional cubes, shuffle-exchange, and switching network arrangements. Many of the interconnection networks are also utilized in asynchronous parallel systems (section 1.3.4). Figure 1.3.3.2 shows common mesh, 4-d hypercube, and hypertree interconnections for array processing.

Synchronous parallel systems are known by many names, which generally refer to the types of problem for which they are suited, or distinctive aspects of their operation. *Array* processors are most widely used where operations on array and vector-data predominate. Array and vector processors often exploit pipeline and parallel processing techniques, where vector operations are performed either by multiple pipelines operating simultaneously (Charlesworth, 86), or by utilizing a pipeline to process elements within a vector (Hwang 3, 84) (Stone 3, 90). *Systolic* array architectures consist of many identical processing elements interconnected in a regular fashion (linear, rectangular, hexagonal, octagonal meshes), with the edge elements providing the only connection to the external system. Data flow and computation is rhythmic and regular throughout the network (Kung, 82). *Associative memory* processors generally belong to the synchronous class, as well (Lea, 91).

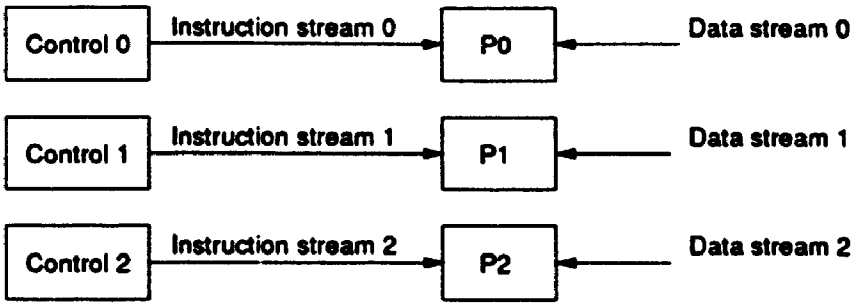
Figure 1.3.3.2 Mesh, Cube and Tree interconnections.



1.3.4 Asynchronous Parallel Systems

The set of asynchronous parallel systems encompasses those categorized by Flynn as MIMD (Multiple Instruction stream, Multiple Data stream), as well as data-flow and reduction architectures, and wavefront arrays (Duncan, 90). A generalized MIMD parallel system is depicted in figure 1.3.4.1. The figure indicates that centralized control is non-existent; system control operations may be distributed, or assigned to a designated unit. In practice, the control and processor elements shown in the diagram are integrated.

Figure 1.3.4.1 Generalized MIMD parallel machine organization.

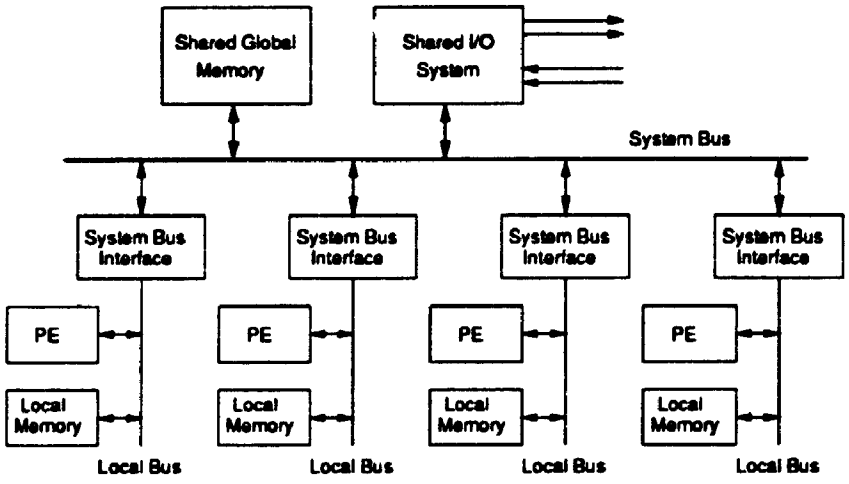


The processing elements in MIMD systems execute instruction sequences (which have varying degrees of disparity) which are usually stored in the local memory of each element. Processes or tasks are assigned to the various elements, and data may be exchanged between processes using both static interconnection networks similar to those described in the previous section, as well as dynamic networks, discussed in section 1.3.5.

Multicomputer systems utilize message passing strategies to transfer data between interconnected nodes, each of which consists of a processor, local memory, and a number of communications channels (Reed, 87). See Seitz and Althas for discussions on multicomputer hypercube systems, and Agrawal and Reed for performance issues (Seitz, 85) (Athas, 88) (Agrawal, 86) (Reed, 86). The term *multiprocessor* is used to describe MIMD organizations with shared memory. Multiprocessor systems with bus-oriented connections to the shared memory and other shared resources are common. The von Neumann interconnection philosophy is extended to permit multiple bus masters, and numerous multiprocessing bus standards exist (VMEbus, Microchannel, Multibus, Nubus, etc.) (Cornejo, 86) (Del Corso, 86) (Dexter, 86). Multiprocessors of this type are economical, since processing power can be enhanced inexpensively. Utilization of peripherals is small for computationally intensive tasks, and may be efficiently shared by the various processes. In multi-user systems, distinct user tasks are often assigned to separate processors, enabling

a larger user load to be accommodated. The strategy requires little data exchange between processors, and therefore less bus traffic. Figure 1.3.4.2 shows a multiprocessor system with a bus interconnection to shared resources. Multiprocessor systems may also utilize multi-port memories for data exchange in order to reduce individual bus activity and to alleviate bus arbitration overhead and complexity (Jagadish, 89).

Figure 1.3.4.2 Typical bus-connected multiprocessor system.



Since the processors execute separate instruction streams, some synchronization between communicating processes is necessary. In multicomputers, transmitting processors await the intended destination processor(s) to become ready to receive a message. In multiprocessors, access to shared memory space is regulated by *semaphore* strategies (Stone 4, 90). The *grain-size* of a parallel computation refers to the amount of computation which occurs between synchronizations (Tanenbaum 2, 90). Multiprocessors with high-bandwidth communications paths between elements (*tightly-coupled*) are suitable for fine-grained tasks. *Loosely coupled* systems are more suited to coarse-grained tasks, since communications are more time-consuming. Under the given definition, the SIMD systems of the previous section have very fine-grained parallelism.

Data-flow and *reduction* architectures are asynchronous parallel systems of a different nature; the concept of a program counter does not exist. In a data-flow machine, operations are enabled for execution upon the availability of operand data. Concurrency can be very high, and execution closely follows operations specified in a data flow graph (figure 1.3.2.1) (Watson, 82) (Srini, 86). Reduction architectures are seemingly opposite to data-flow architectures, since execution of an operation is enabled upon demand of its results by previously enabled operations. In primitive terms, demand tokens are propagated through a network of functional units until they reach terminal units. Operations are initiated, and results propagate back toward the demand source in a manner akin to that in the data-flow strategy (Dunca, 90).

Wavefront arrays are similar in structure to systolic arrays, however, activity and communication among elements is asynchronous. Operations are therefore enabled by the availability of data, and for this reason, wavefront arrays represent a combination of both systolic and data-flow principles (Kung 2, 88).

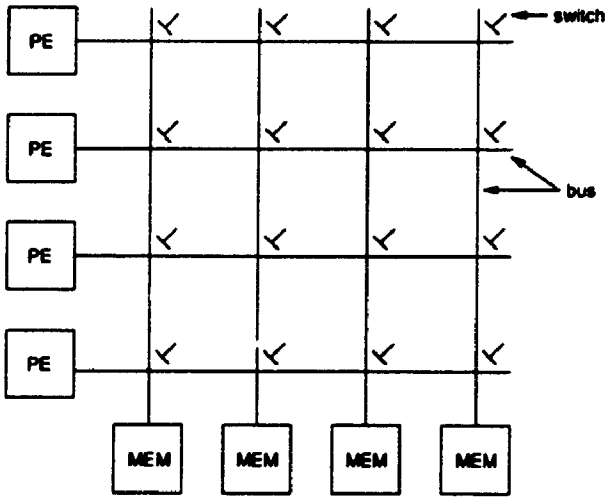
1.3.5 Switching Networks and Reconfigurable Systems

Since a fixed interconnection network can limit the scope of applications for which a machine is suited, networks that can be reconfigured in some manner are attractive. Issues of network control magnify the complexity and cost of such systems, however, their increased flexibility and performance can make them cost-effective. Many of the networks discussed in this section may interconnect processors operating under SIMD or MIMD paradigms.

A commonly referenced dynamic interconnection system is the crossbar-switch network. Processors are connected to memories via buses logically arranged in a rectangular grid, with switches straddling the crossing points. Switches connect processors to memories upon request. Figure 1.3.5.1 illustrates a crossbar connected, shared-memory

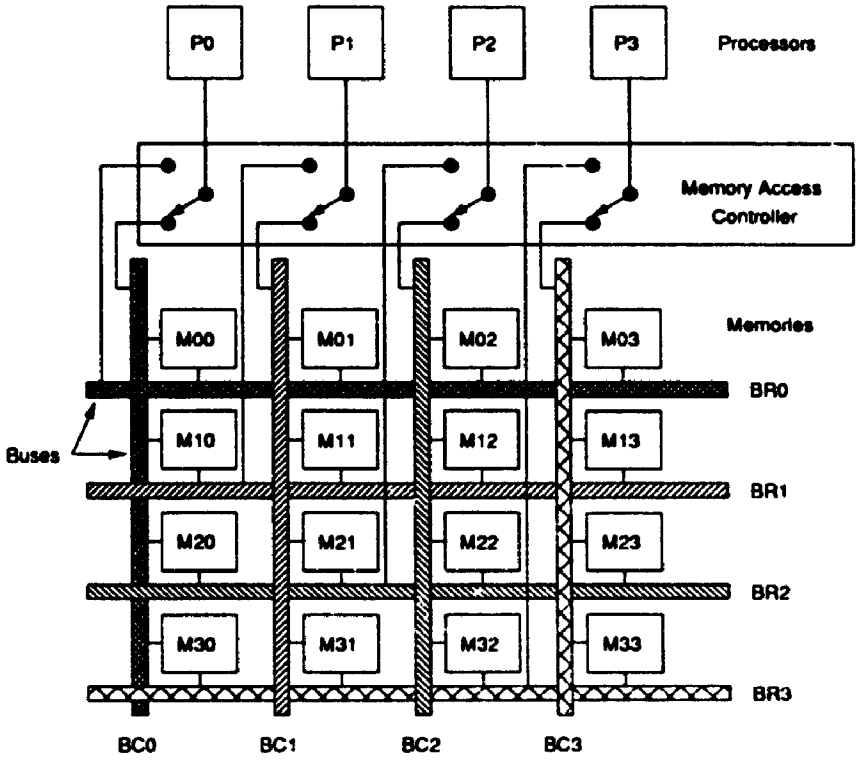
multiprocessor. As in the single-bus standard, simultaneous access to a memory is forbidden, but the effects of the von Neumann single-bus bottleneck are diminished by the presence of multiple buses. Concurrent memory requests must be arbitrated, and problem-specific data distribution techniques are applied to minimize conflicts. The crossbar network has complexity which increases as the square of the number of processors, and network cost is a prohibitive factor for large systems (Van de Goor, 89). An *orthogonal multiprocessor system* which switches access to the rows and columns of a memory array is a related alternative. It combines an efficient data-sharing methodology, simple control strategy, and high communications bandwidth to yield performance comparable to more complex networks (Hwang 1, 89).

Figure 1.3.5.1 Crossbar connected, shared-memory multiprocessor.



A number of interrelated switching networks with more modest growth in complexity have been developed. They are known as multistage switching, shuffle-exchange, Omega, generalized cube, Benes, and Banyan networks (Lawrie, 75) [Tanenbaum 3, 90) (Haynes, 82) (Padmanabhan, 83). Each provides interconnections to system resources utilizing diverse inter-stage connections, and variety of crossbar switches (fan-in, fan-out,

Figure 1.3.5.2 Orthogonal multiprocessor system.



and broadcast capabilities vary). Figures 1.3.5.3 shows an Omega network connecting eight processors to eight memories. Figure 1.3.5.4 shows a generalized cube network of eight processing elements. Both networks employ 2×2 crossbar switches.

Figure 1.3.5.3 Omega network with N=8.

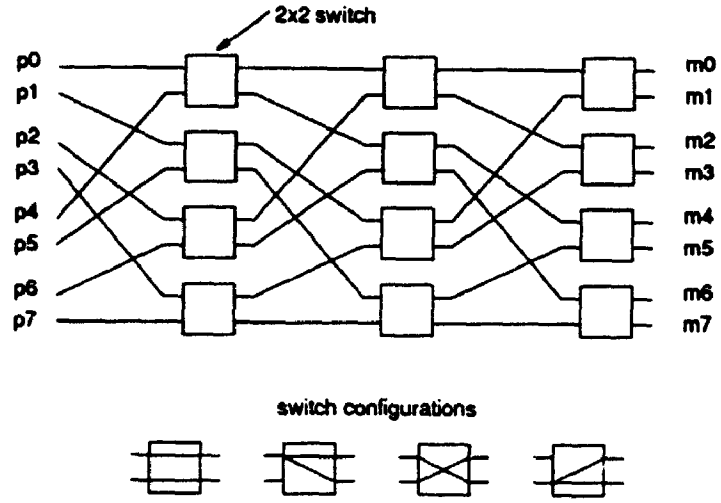
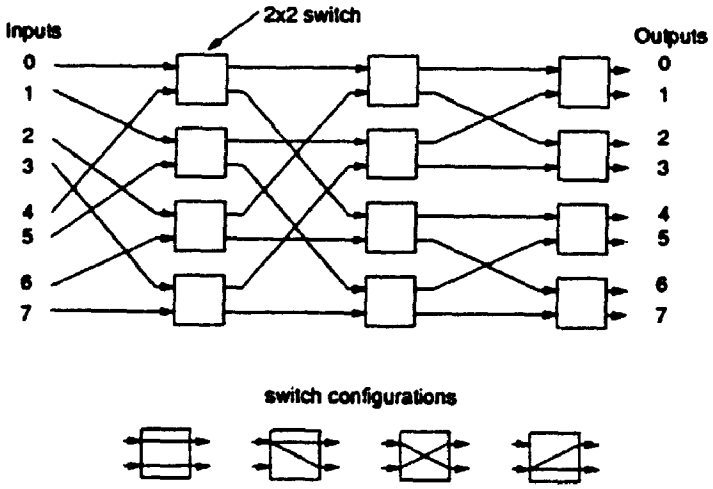


Figure 1.3.5.4 Generalized Cube network with N=8.



Routing decision and message enqueuing circuits are sometimes integrated with each switch. Control and routing methods are discussed in the references. Implementations may utilize either packet or circuit switching. In packet-switched systems, a path through a

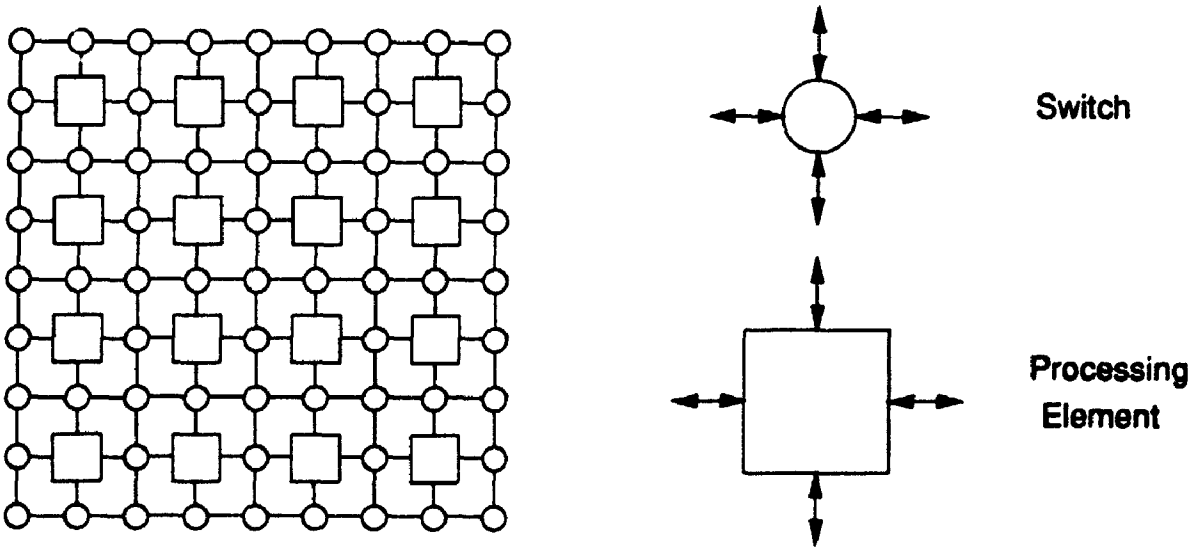
switch or stage is locally enabled according to the eventual destination of the current packet of data, without reference to the state of other switches in the system. Circuit-switched systems establish complete paths from source to destination throughout a transaction.

The generalized cube network is particularly interesting for its partitionability, wherein an N sized cube can be configured into independent cubes of smaller dimension (Siegel 1, 81) (Siegel 2, 81). The multistage switching networks share $O(\log N)$ and $O(N\log N)$ growth rates in number of stages and switches, respectively. The networks provide at least one path between any pair of elements in the systems, and unless sufficient redundancy exists, blocking of paths will occur. Many applications have been shown to execute efficiently using these networks, in particular, the FFT, matrix computations, and numerous sorting algorithms. Siegel discusses various multistage interconnections networks, their interrelationships, and fault-tolerant enhancements [Siegel 1, 90]. See also (Quinn 3, 87), (Cvetanovic 1, 87), (Stone, 71), (Lang, 76), (Chen, 81), (Siegel, 92). Popular and successful examples of this class of reconfigurable architecture are the Connection machine (Tanenbaum 4, 90), (Tucker, 88), and the IBM GF-11, which comprises 576 processors operating in SIMD mode, using 24×24 crossbar switches arranged in three stages, with shuffle interconnections between stages. (Hwang 2, 89).

The multistage interconnection networks share another attribute: they are not incrementally expandable. As the number of processors increases, the interconnection networks must be re-specified and extensively rewired. Even some static networks, such as a fixed-cube (Seitz, 85) may not be expanded by replication of elementary cells, since ports must be added to all processing elements in the system (Goodman, 81). Implementation of components for the systems in LSI or VLSI technology is necessary due to their complexity. However, VLSI implementation is most attractive and most efficiently accomplished when elemental cells may be identically replicated to form larger, more powerful systems; static systolic and wavefront arrays are more suitable.

Another approach to reconfigurability was introduced by Snyder which directly addresses issues of VLSI augmentation (Snyder, 82). The proposed system consists of a regular structure of processing elements attached to a programmable switch lattice at regular intervals. Each processor comprises four or eight bidirectional communications ports to link them to the switch lattice. Each switch incorporates a configuration memory, where configuration codes are loaded prior to program execution, via some *skeleton* network. The skeleton network also permits a controller to access processing element local memory, for program and data loading. Switch settings change in response to the controller's signals. The system may be reconfigured throughout a program to permit maximum efficiency. Snyder also determined issues of corridor width (the number of switches between rows and columns of processors), and related it to overall degree of reconfigurability which the system could achieve. The possibility for fan-out on the switches was discussed (broadcast settings), however, most subsequent studies have focussed on 2×2, bidirectional switches without broadcast abilities. Figure 1.3.5.5 is an example of the organization suggested by Snyder.

Figure 1.3.5.5 Rectangular arrangement of processors for Snyder's reconfigurable system.



Implementation required high-yield *wafer scale integration* to support the complexity of processor envisioned (even today, most studies yield simulated results). The ideas have found widespread acceptance in the areas of fault tolerance and WSI yield improvement. Reconfiguration is most commonly utilized for routing signals such that they do not link faulty processing elements and switches (which can also fail). Spare elements may be linked in, resulting in logically regular, fixed processing arrays. After fabrication and identification of faulty cells, the switches may be permanently configured by custom metallization, laser or electrical surgery, or non-volatile programming techniques (Kung 3, 88) (Boubekeur, 92). Alternatively, reconfiguration may occur as a result of a run-time fault, in which case, the switch matrix must maintain programmability. Reconfiguration methodologies and algorithms concentrate on casting a network into its original logical topology in the presence of a fault. See for example (Chean, 90), (Belkhale, 92), (Balasubramanian, 87), (Sami, 86) (Choi, 91) (Li, 91). Hwang exploits reconfiguration to achieve increased performance using *pipeline nets*, which is a technique related to pipeline-chaining in supercomputers (Hwang, 88).

1.3.6 The UWO Reconfigurable Multicomputer System

The presented thesis focusses on a novel reconfigurable multicomputer architecture. A prototype system is specified and implemented, and its performance characterized experimentally. The system was designed for high flexibility and connectivity, making it suitable for computations belonging to a wide range of problem classes. Many features of the design are inspired by incremental expandability considerations, warranting eventual VLSI implementation of systems with similar architecture. Since the main goal was to test architectural concepts, a complexity limit was imposed on the processing cells and network elements. The chosen restrictions encumber the system performance, however, the penalty is slight, due to efficiency in the system control and its flexibility.

The system consists of *autonomous processor cells*, interconnected with an array of 2×2 *programmable signal router* (PSR) devices. The PSR devices bear resemblance to crossbar switches, but are not identical, hence distinguishing terminology is adopted. The PSR array is called the Interprocessor Communication Network or ICN. The system is controlled by another autonomous cell designated the system *master*. The remaining *slave* cells are identical in both hardware and ROM resident software. All processor cells comprise a commercial, high performance microprocessor, local memory, and two data communications ports; one permanently dedicated to input transactions, the other to output. The ports provide the only data communications link between a cell and the rest of the system. The master cell fulfills the network configuration, slave synchronization, and task supervision functions, as well as user and host interface operations. It is also responsible for loading the slave memories with their program code and initial input data, which it accomplishes via the ICN. Slave task synchronization is facilitated using a *service request bus* (SRQbus), which indicates slave status to the master (not to other slaves), and blocks slave processes pending acknowledgement by the master. The master cell may also be employed for computations if its essential obligations are met. System operation is decidedly MIMD, however quasi-MIMD may be employed (Hoshino, 86), and even SIMD may be emulated. Data communications throughout the system are asynchronous, and require sender-receiver interlocked handshakes, which provides a complementary process synchronization method.

The four-slave system (and associated ICN) is itself designed as an elementary cell, and system expansion is accomplished through replication. Cells may be appended in a number of locations, and only connections on the network periphery are modified to accommodate new units. The prototype system is fully connected and has high redundancy. Selection of expansion-cell interface sites influences configurations available in larger systems. In all system expansions, only a single master is necessary, which always

communicates with slaves utilizing the ICN. Slave SRQ request and acknowledge signals tap into the SRQbus, following the usual shared-parallel bus paradigm. Figure 1.3.6.1 is a block diagram showing the architecture of the reconfigurable multicomputer, figure 1.3.6.2 presents the interprocessor interconnection network topology, and figure 1 3.6.3 shows valid PSR configurations.

Figure 1.3.6.1 Reconfigurable Multicomputer System Architecture

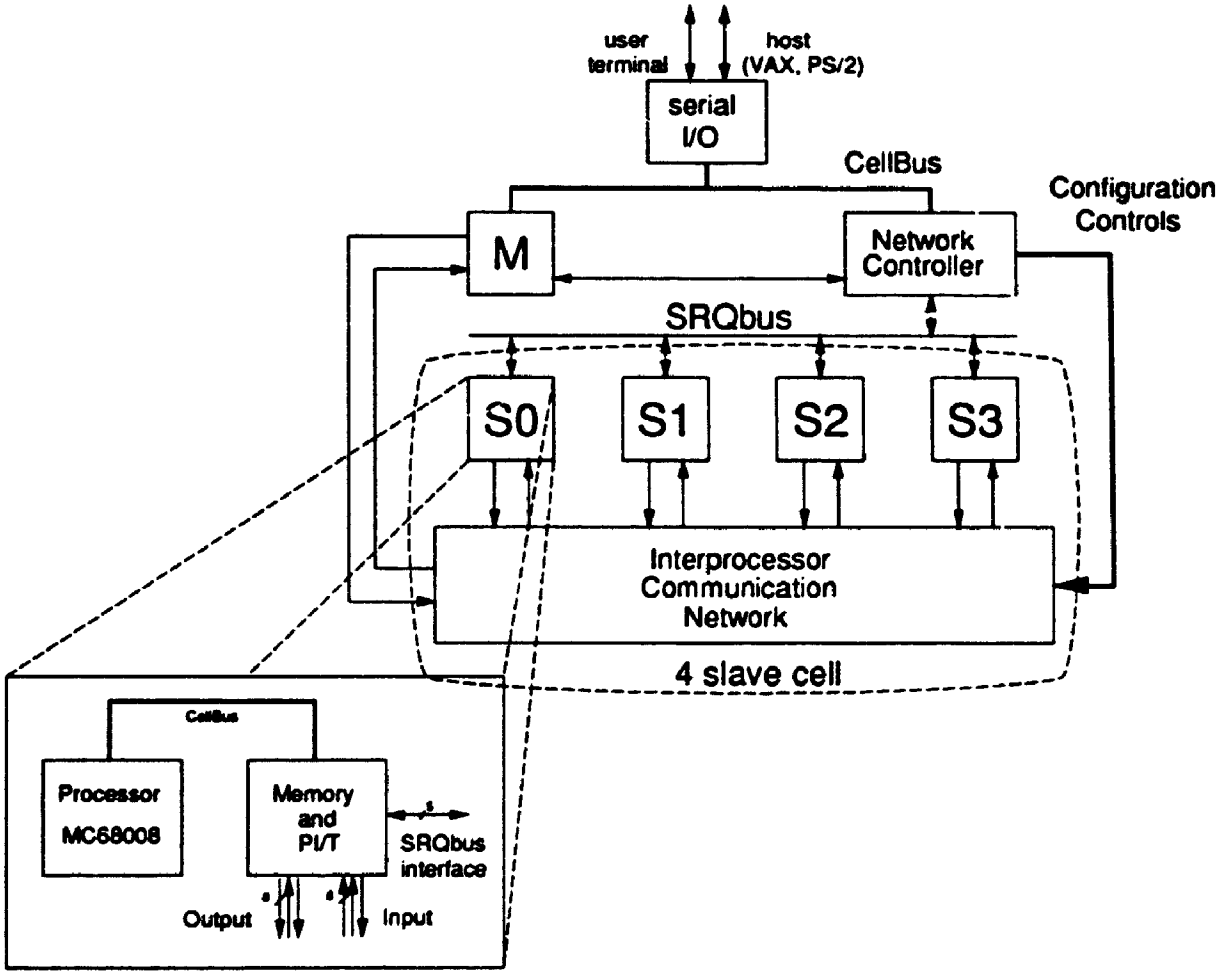


Figure 1.3.6.2 Interprocessor Communication Network

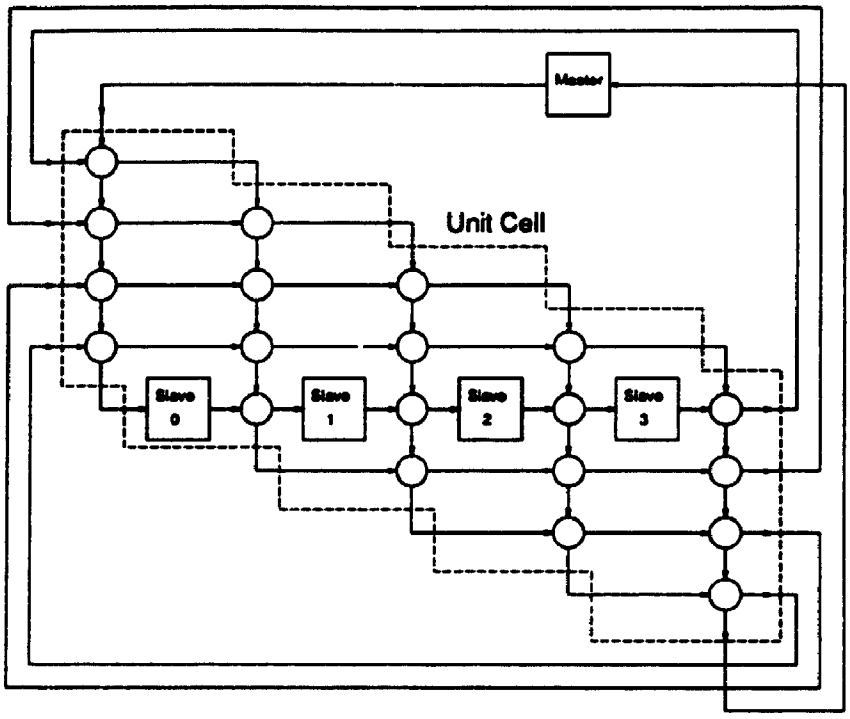
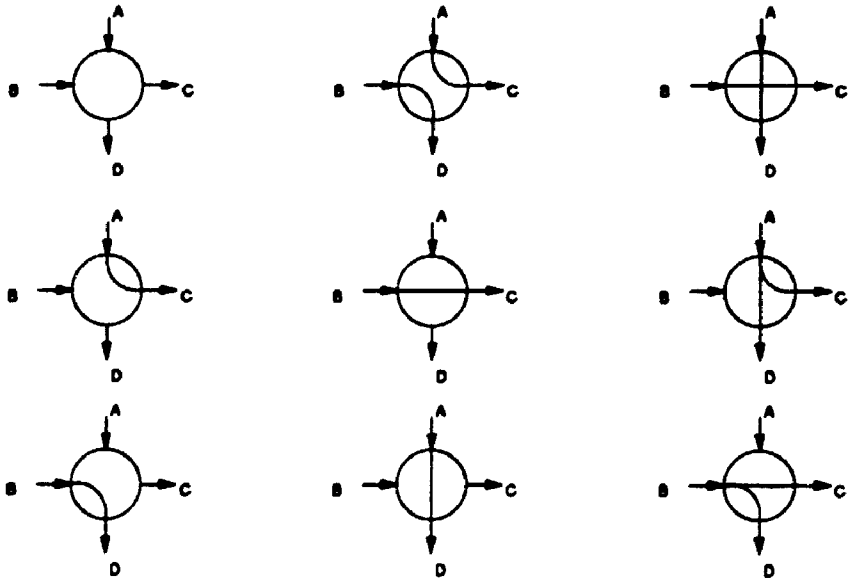


Figure 1.3.6.3 PSR Configurations



1.3.7 Objectives of the Study

The thesis has several objectives. A four-slave, single-cell prototype system is designed and constructed using commercially available devices, wherever possible. The PSR devices are custom-designed and fabricated using a 2-metal layer, 3-micron CMOS VLSI process. The prototype system addresses and resolves all issues of hardware system control and operation, as well as the necessary system-level control software, user and host interface code, and debugging facilities. The resulting system is therefore a complete, operating and reliable multicomputer system with unique flexibility and adaptability, that provides high-performance in a wide range of applications.

The experiments focus on the performance increase provided by the architecture, and determine the operational overhead incurred due to inherent features of the design. Since the practice of performance-benchmarking computers of different classes is generally problematic, the multicomputer performance is compared to that of a uniprocessor system. Physically, the uniprocessor is identical to the constituent slave processors in the system; in fact, it is used as the master cell itself. The performance tests are carefully defined, and comparisons are made using clearly-specified frames of reference. Problems which utilize different topologies are of interest, since the intention was to show that efficiency was high for numerous configurations. Of special interest are problems which benefit from dynamic system reconfiguration throughout the computation of a problem. Operational overhead costs for data communication (including signal propagation delay), and slave process synchronization are investigated. The significance of the incurred overhead is analyzed with respect to overall task time. The goal of the work is to determine whether or not the system organization offers significant speed-up factor and efficiency in a broad range of tasks and configurations. The study shall also indicate aspects of the system design which are most critical to performance, and will suggest improvements which could be made to enhance performance of subsequent designs. Finally, the results of the study are scalable,

so that performance of systems comprising more or less sophisticated processing elements may be predicted based on the knowledge accumulated by the presented thesis. Processor, communication and reconfiguration speeds have implications to task granularity, applicable topologies, and control strategies. The thesis seeks to determine the performance-relevant factors in reconfigurable systems, and to show the relation between the system architecture and performance.

The original contributions of the thesis include the cellular reconfigurable architecture of the multicomputer system, the PSR design and its implementation, the slave synchronization strategy via the SRQbus, and the overall system control strategy which has been adopted (which incorporates the control software component of the system). The thesis investigates the efficacy of the selected architecture in solving a range of problems from different classes of scientific computations, and determines the effect on performance of various control strategies for a given problem. The thesis also demonstrates the importance of the match between the algorithm and the architecture in achieving maximum computational performance, and thus the desirability of a dynamically reconfigurable multicomputer system for scientific research-oriented applications.

2 Theory and Background

This chapter focusses on issues of computer performance and its measurement. The algorithms used to characterize the reconfigurable multicomputer's performance are outlined, as well as their uniprocessor counterparts. Theoretical limits of performance increase are determined for each application. System overheads, inseparable from the multicomputer operation, are also described.

2.1 Computer Performance

As the efficiency with which a computer performs a task depends on the attributes of the given task, the term *performance* is difficult to define; an absolute unit of performance may be impossible to formulate. It follows that the interpretation of performance test results is not unequivocal. The performance rating is task sensitive.

Advertisements, press releases, and trade journals which introduce new machines regularly use terms such as MIPS (millions of instructions per second) and FLOPS (floating-point operations per second) as units with which to specify a computer's capabilities. The MIPS rating has largely been discredited as a valid unit, since execution time of all instructions in a processor's repertoire is rarely equal. Current RISC architectures which boast single cycle execution for all instructions depend heavily on continuous instruction flow through their pipelines to achieve such performance. Additionally, the work potential of the individual instructions must be regarded. A raw MIPS comparison, without some indication of the overall task performed, may be of limited usefulness. The FLOPS unit partially addresses the task-sensitivity issue by categorizing instructions into a particular class. It is somewhat useful as a comparative measure for applications where floating point operations predominate. As with the MIPS rating, the matter of instruction execution time equivalence cannot be overlooked. It is essential that computer performance is specified with respect to the task or overall work performed.

In an effort to quantify system performance, several "benchmark tests" have been developed. BYTE magazine used programs specified in the C language to compare systems (BYTE, 87). The validity of the tests was immediately questioned, since not only was system performance being compared, but also compiler efficiency (Grehan, 88). The magazine has since revised its benchmarks, and now includes tests which use commercial software packages such as AUTOCAD, LOTUS 1-2-3, etc., to solve well-defined problems. The microprocessor's performance affects results, but so does disc speed, operating system overhead, and graphics sub-systems. The authors achieve their stated goals of "demonstrating product improvements" and to "help us decide which products to buy" (BYTE, 87). The use of specific application software limits comparisons to machines of the same genre: IBM PC and compatible machines, or Apple Macintosh-series machines. The authors recognize the problem in comparing the two breeds of machine fairly, and their procedure prevents comparisons of these machines to common mainframes or minicomputers.

Electronics Design News (EDN) magazine adopted a set of benchmark programs originally presented by S.H. Fuller *et al*, of Carnegie-Mellon University. The tests exercise processor abilities in tasks which stress I/O device interrupt handling, character searching, bit operations, sorting, and matrix transpositions. The test algorithms are coded in assembly language, often by competing device manufacturers. Microprocessor companies frequently use the EDN benchmarks in their literature (Motorola 1, 86) (Motorola 2, 86) (Intel, 85), but despite apparent standardization, disparate conclusions are drawn. Results are often quoted for input data, data structure, or data size which are selected to obscure a device's shortcomings. With the advent of extremely high performance microprocessors, incorporating cache memory and basic operating system paradigms, benchmarking has become more complex. Competing systems are compared using similar operating systems (i.e., all systems running a UNIX implementation for the given processor), and "newest

available" optimizing compilers (Motorola, 88). Although the method may be a practical way to compare dissimilar systems, the effects of compiler and operating system efficiency, as well as peripheral performance, degrade the tests as indicators of processor performance. The inherent difficulties of UNIX benchmarking are described in further detail, in Hinnant (Hinnant, 88), and the many loopholes which may be used to obfuscate results are outlined; Weick and Price provide similar insight (Weick, 91) (Price, 89).

The benchmarking process presents a fundamental problem, due to the flexibility of the encoding method. A programmer may opt for faster, memory-wasteful, in-line code as an alternative to slower, but memory-efficient looping, or structured code. Compromises must be struck between absolute maximum-performance coding, memory usage, program and data structure, subroutine linkage, and programming ease. Considerably different instruction sequences may be specified to realize similar functions, and dissimilar performance among them will be observed. Incompatibility of test results is a consequence of the lack of unification in test standards among competing manufacturers.

For multiple processor systems, benchmarking difficulties are compounded. Since systems are often designed for specific problem classes, general benchmark tests may not be applicable. The elements which complicate uniprocessor benchmarks remain, and the necessity to assign tasks to the various processing elements provides an additional level of complexity. Algorithms may have to be radically modified to convert them into parallel forms, and some component of the performance rating is attributable to algorithmic efficiency. The programs, in general, are not directly transportable between systems of dissimilar processor organization and interconnection topology. Worlton attempts to form a taxonomy for performance metrics for multiple-processor systems, in which it is undoubtedly proven that the undertaking is indeed complex (Worlton, 91).

When comparing uniprocessor performance to that of a multiple processor system, tests and measurement intervals must be clearly defined. It is not always practical to define a problem's starting time as the instant input data is available to all elements, nor to define the end as the instant each element has produced results. In many multiprocessors, input data must first be made available to the processing elements, and this may require a number of operations and time to achieve. Similarly, local memory-resident results may not be readily accessible to the user, or subsequent subroutines; some further operations may be necessary to establish the desired final state. Performance tests must be clearly defined, and measurement intervals unequivocally stated so that fair, practical, and useful conclusions may be drawn.

2.2 Performance Tests for the Reconfigurable System

The RMCS performance tests recognize the intrinsic difficulties with the benchmarking process described preceding; comparison with other architectures is not undertaken. Absolute performance of a system is implementation specific, dependent upon element performance and sophistication, control strategy, and system topology. A fair comparison between systems requires a degree of standardization in both hardware and program implementations which is impractical and perhaps impossible. Comparison of the actual performance of the RMCS with theoretical performance of other architectures would likewise be illusory. The goal of the work is to determine the performance advantage, if any, that the RMCS offers over uniprocessor performance, where the uniprocessor is identical to an RMCS constituent processing element. Using this procedure, the interconnection network, synchronization, and system control overheads may be observed. The results may be scaled to estimate performance of systems comprising processing elements of higher sophistication or greater number. Algorithms must be modified to be executed on the uniprocessor or multicomputer to facilitate efficient utilization of available resources. In the performance test programs, a degree of similarity between uniprocessor

and multiprocessor algorithms was purposely maintained wherever possible, practical or informative. Although some programs could have been more heavily optimized using in-line coding etc., judicious use of available library functions (for data communication and processor synchronization) was made, instead. The method used more closely reflects typical system application. The tests are specified such that initial and final states of the system are similar for related tests, to mitigate ambiguity in test definitions and measurement intervals. Intermediate phases of execution are also measured to permit comparisons under various problem definitions. The following sections briefly describe the tests used to characterize the RMCS's performance, and concentrate on theoretical maximum performance increases offered by the multicomputer applications, based on arithmetic operation tally. Details of actual program implementations and test variations are deferred until chapter 4, since the programs should not be evaluated in isolation from the system operation, which is described in chapter 3.

2.2.1 Matrix Multiplication

Matrix multiplication is frequently required in the area of numerical analysis, and it is often time consuming to accomplish for matrices with large dimension. Equation 2.2.1.1 summarizes the multiplication of two $n \times n$ matrices $[C]=[A][B]$.

$$C(i,j) = \sum_{k=0}^{n-1} A(i,k) \times B(k,j) \qquad i,j = 0,1,\ldots,n-1 \tag{2.2.1.1}$$

To calculate the result requires n^3 multiplications and $(n\text{-}1)n^2$ additions.

The computation of the product is readily partitioned for concurrent processing, and numerous algorithms have been devised for systems with diverse organizations (Quinn 4, 87). The selected parallel algorithm assigns each processor evaluation of specific rows of the resultant matrix. The individual processors require access to the entire matrix $[B]$,

and the entries of the rows in $[A]$ corresponding to the rows in $[C]$ for which they are responsible. For a system with r processors where n/r is a positive integer, each processor performs n^3/r multiplications and $(n\text{-}1)n^2/r$ additions, yielding a theoretical maximum speed-up factor of r.

2.2.2 Fourier Transforms

Calculation of the Fourier transform is frequently required in signal and image processing, vibration analysis, and applied geophysics. The discrete Fourier transform (DFT) and the fast Fourier transform (FFT) are numerical methods for computing the Fourier transform. The arithmetic operations required are numerous, and consume a considerable amount of processing time. For decreasing the overall FFT calculation time, special multiple processor system architectures and algorithms have been proposed, developed and studied (Cvetanovic 1, 87) (Cvetanovic 2, 87) (Pease, 68) (Kirk, 87) (Bi, 89) (Singleton, 67) (Viswanath, 87) (Huang, 91) (Averbuch, 90). This section discusses calculation of the one-dimensional FFT using a uniprocessor system, and an algorithm suitable for execution on a globally parallel, multiple processor system.

The continuous Fourier transform is defined by:

$$\mathcal{F}\{x(t)\} = X(f) = \int_{-\infty}^{\infty} x(t)e^{-j2\pi ft}dt \qquad (2.2.2.1)$$

t = time, and

f = frequency

The discrete Fourier transform is similarly defined by:

$$\mathcal{F}\{x(k\Delta t)\} = X(n\Delta f) = \sum_{k=0}^{N-1} x(k\Delta t)e^{-j2\pi n\Delta f k\Delta t}$$

where:

Δf is frequency interval,

Δt is time interval,

n,k are integers = 0,1,2,..N-1, and

N is the number of sample points.

With $\Delta t = 1/(N\Delta f)$, the DFT equation is more commonly expressed as:

$$\mathcal{F}\{x(k)\} = X(n) = \sum_{k=0}^{N-1} x(k)e^{-j2\pi nk/N} \qquad (2.2.2.2)$$

Equation 2.2.2.2 can be written in matrix form for N=4 and using the substitution:

$$W = e^{-j2\pi/N} \qquad (2.2.2.3)$$

the following matrix formulation results:

$$\begin{pmatrix} X & (0) \\ X & (1) \\ X & (2) \\ X & (3) \end{pmatrix} = \begin{pmatrix} W^0 & W^0 & W^0 & W^0 \\ W^0 & W^1 & W^2 & W^3 \\ W^0 & W^2 & W^4 & W^6 \\ W^0 & W^3 & W^6 & W^9 \end{pmatrix} \begin{pmatrix} x_0 & (0) \\ x_0 & (1) \\ x_0 & (2) \\ x_0 & (3) \end{pmatrix} \qquad (2.2.2.4)$$

The system of equation 2.2.2.4 requires N^2 complex multiplications and $N(N\text{-}1)$ complex additions to calculate a complete solution. The radix-2, Cooley-Tukey FFT algorithm for N=2^γ will be presented, with the N=4 case used in the examples.

Indices n and k are rewritten in their binary form:

$$n = n_{\gamma-1}n_{\gamma-2}\ldots n_1 n_0 \qquad n_m = 0 \quad or \quad 1 \qquad m = 0,1,\ldots,\gamma-1$$

$$k = k_{\gamma-1}k_{\gamma-2}\ldots k_1 k_0 \qquad k_m = 0 \quad or \quad 1 \qquad m = 0,1,\ldots,\gamma-1$$

giving:

$$n = 2^{\gamma-1}n_{\gamma-1} + 2^{\gamma-2}n_{\gamma-2} + \ldots + n_0$$
$$k = 2^{\gamma-1}k_{\gamma-1} + 2^{\gamma-2}k_{\gamma-2} + \ldots + k_0$$

The DFT equation can be written using these substitutions as:

$$X(n_{\gamma-1}n_{\gamma-2}\ldots n_0) = \sum_{k_0=0}^{1}\sum_{k_1=0}^{1}\cdots\sum_{k_{\gamma-1}=0}^{1} x(k_{\gamma-1}k_{\gamma-2}\ldots k_0)W^p \qquad (2.2.2.5)$$

$$p = nk = (2^{\gamma-1}n_{\gamma-1} + 2^{\gamma-2}n_{\gamma-2} + \ldots + n_0)(2^{\gamma-1}k_{\gamma-1} + 2^{\gamma-1}k_{\gamma-2} + \ldots + k_0) \qquad (2.2.2.6)$$

Since:

$$W^{a+b} = W^a \times W^b$$

W^p may be expressed as:

$$W^p = W^{\left(2^{\gamma-1}n_{\gamma-1}+2^{\gamma-2}n_{\gamma-2}+\ldots+n_0\right)\left(2^{\gamma-1}k_{\gamma-1}\right)}W^{\left(2^{\gamma-1}n_{\gamma-1}+2^{\gamma-2}n_{\gamma-2}+\ldots+n_0\right)\left(2^{\gamma-2}k_{\gamma-2}\right)}\ldots \qquad (2.2.2.7)$$
$$\times W^{\left(2^{\gamma-1}n_{\gamma-1}+2^{\gamma-2}n_{\gamma-2}+\ldots+n_0\right)k_0}$$

Equation 2.2.2.7 may be factored using the relations:

$$W^{2^\gamma} = 1 \tag{2.2.2.8}$$

$$W^{nk} = W^{nk \bmod(N)} \tag{2.2.2.9}$$

to give:

$$W^P = W^{2^{\gamma-1} n_0 k_{\gamma-1}} W^{(2n_1 + n_0)2^{\gamma-2} k_{\gamma-2}} \cdots \times W^{\left(2^{\gamma-1} n_{\gamma-1} + 2^{\gamma-2} n_{\gamma-2} + \cdots + n_0\right) k_0} \tag{2.2.2.10}$$

Taking the sums individually, and successively factoring the W^P terms yields the following set of equations:

$$x_1(n_0 k_{\gamma-2} \ldots k_0) = \sum_{k_{\gamma-1}=0}^{1} \left[x_0(k_{\gamma-1} k_{\gamma-2} \ldots k_0) \times W^{2^{\gamma-1}(n_0 k_{\gamma-1})} \right] \tag{2.2.2.11}$$

$$x_2(n_0 n_1 k_{\gamma-3} \ldots k_0) = \sum_{k_{\gamma-2}=0}^{1} \left[x_1(n_0 k_{\gamma-2} \ldots k_0) \times W^{(2n_1 + n_0)2^{\gamma-2} k_{\gamma-2}} \right] \tag{2.2.2.12}$$

$$\vdots$$

$$x_\gamma(n_0 n_1 \ldots n_{\gamma-1}) = \sum_{k_0=0}^{1} \left[x_{\gamma-1}(n_0 n_1 \ldots k_0) \times W^{\left(2^{\gamma-1} n_{\gamma-1} + 2^{\gamma-2} n_{\gamma-2} + \ldots + n_0\right) k_0} \right] \tag{2.2.2.13}$$

Using N=4, the above equations may be expressed in matrix form:

$$\begin{pmatrix} x_1 & (0,0) \\ x_1 & (0,1) \\ x_1 & (1,0) \\ x_1 & (1,1) \end{pmatrix} = \begin{pmatrix} 1 & 0 & W^0 & 0 \\ 0 & 1 & 0 & W^0 \\ 1 & 0 & W^2 & 0 \\ 0 & 1 & 0 & W^2 \end{pmatrix} \begin{pmatrix} x_0 & (0,0) \\ x_0 & (0,1) \\ x_0 & (1,0) \\ x_0 & (1,1) \end{pmatrix} \tag{2.2.2.14}$$

$$\begin{pmatrix} x_2 & (0,0) \\ x_2 & (0,1) \\ x_2 & (1,0) \\ x_2 & (1,1) \end{pmatrix} = \begin{pmatrix} 1 & W^0 & 0 & 0 \\ 1 & W^2 & 0 & 0 \\ 0 & 0 & 1 & W^1 \\ 0 & 0 & 1 & W^3 \end{pmatrix} \begin{pmatrix} x_1 & (0,0) \\ x_1 & (0,1) \\ x_1 & (1,0) \\ x_1 & (1,1) \end{pmatrix} \tag{2.2.2.15}$$

Recalling equation 2.2.2.9, equation 2.2.2.15 is the same as equation 2.2.2.4 with:

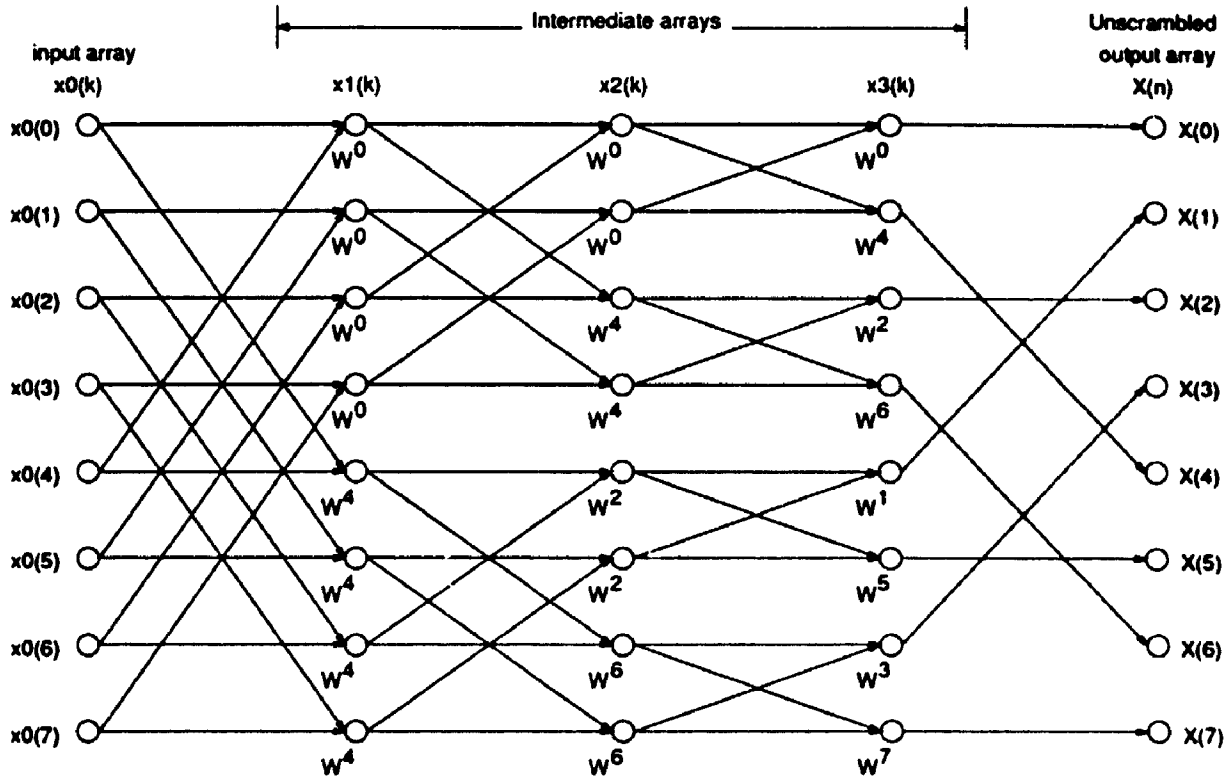
$$X(n_1, n_0) = x_2(n_0, n_1) \tag{2.2.2.16}$$

In general,

$$X(n_{\gamma-1} n_{\gamma-2} \ldots n_0) = x_\gamma(n_0 n_1 \ldots n_{\gamma-1}) \tag{2.2.2.17}$$

The FFT algorithm is essentially a matrix factorization process, often described using a signal flow graph, as in figure 2.2.2.1. The graph is interpreted as follows: an array is represented by a vertical column of nodes; the arrows transmit quantities from a previous node, multiplied by the factor adjacent to the arrow. Products entering a node are added to form an entry in the next array (Brigham, 74).

Figure 2.2.2.1 Signal flow graph of an 8-point, radix-2 Cooley-Tukey FFT calculation.



The symmetries of the signal flow graph may be exploited to yield additional computational efficiency. Calculation of two node values requires a single complex multiplication, since in every array there are two nodes whose arrows stem from the same pair of nodes in the previous array, and their multipliers differ only in sign. Dual node pairs are related by:

$$x_l(k) = x_{l-1}(k) + W^p x_{l-1}(k + N/2^l) \tag{2.2.2.18}$$

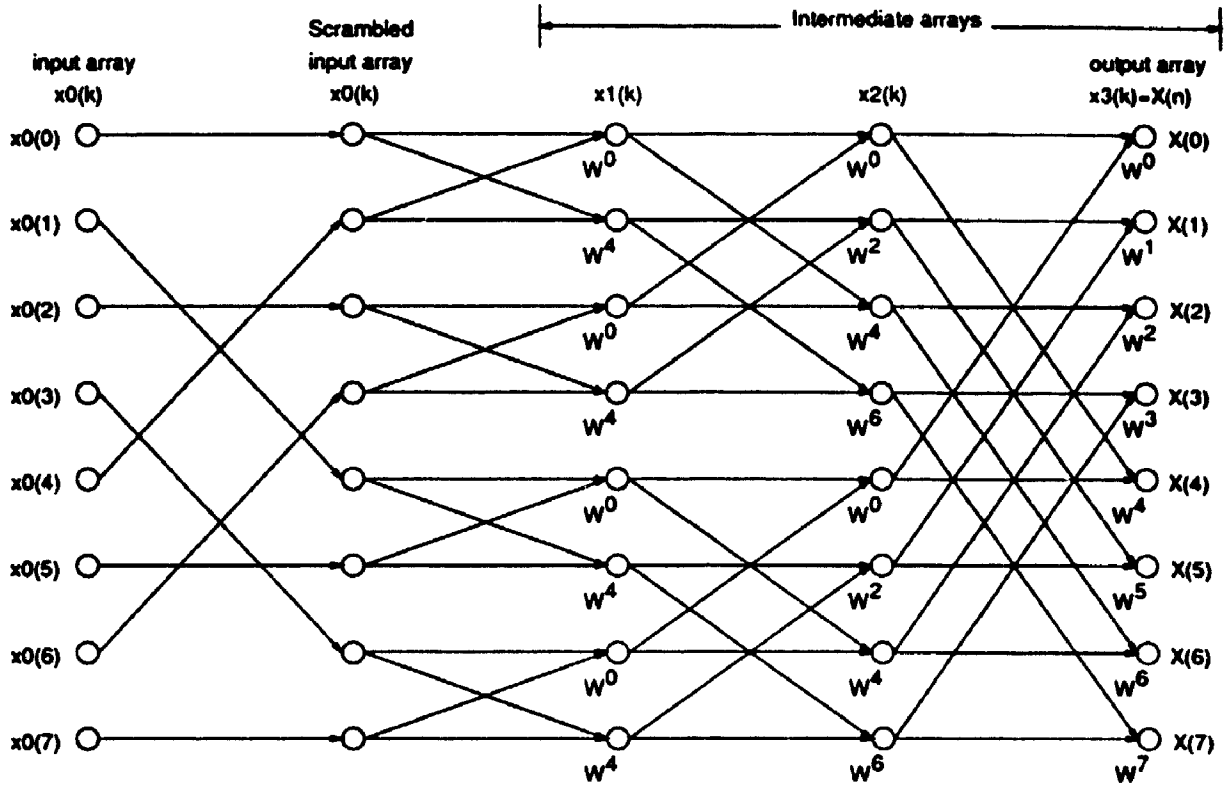
$$x_l(k + N/2^l) = x_{l-1}(k) - W^p x_{l-1}(k + N/2^l) \tag{2.2.2.19}$$

The exponent of W is determined by employing a binary shift, bit-reversal procedure. Unscrambling the final result is also a bit-reversal operation on the array indices. The FFT

algorithm for $N=2^\gamma$ points will calculate the DFT using only $\gamma 2^{\gamma-1}$ complex multiplications and $\gamma 2^\gamma$ complex additions. This compares to $2^{2\gamma}$ and $2^\gamma(2^\gamma - 1)$ multiplications and additions, respectively, required to evaluate the DFT defining equation directly (Eq. 2.2.2.2). Not only does the FFT significantly decrease the number of operations, it also reduces memory requirements. The calculation is performed "in place", since computation of each dual node pair is independent of other nodes. A number of FFT algorithms which use different radices are in widespread use, as well as special-case algorithms, which claim some advantage over the general radix-2 algorithm in terms of efficiency or memory requirements. For this study, a modified form of the radix-2, Cooley-Tukey algorithm was adopted which requires fewer bit-reversal operations. Calculation of the exponent of W requires bit reversal, and the result is used as an index to a table of sine and cosine values. The table could be arranged in bit-reversed order, eliminating the need for the bit-reversal operations altogether. Alternatively, a canonic form of the FFT may be used in which the input array is shuffled into bit-reversed order, the exponents of W occur in natural order, and the output array is automatically naturally ordered. Since ordering the tables naturally is more usual, convenient, and intuitively pleasing, the canonic form was selected for the tests. A signal flow graph of the algorithm is shown in figure 2.2.2.2. A scaling factor of $1/N$ is used throughout the study for forward transforms, as well.

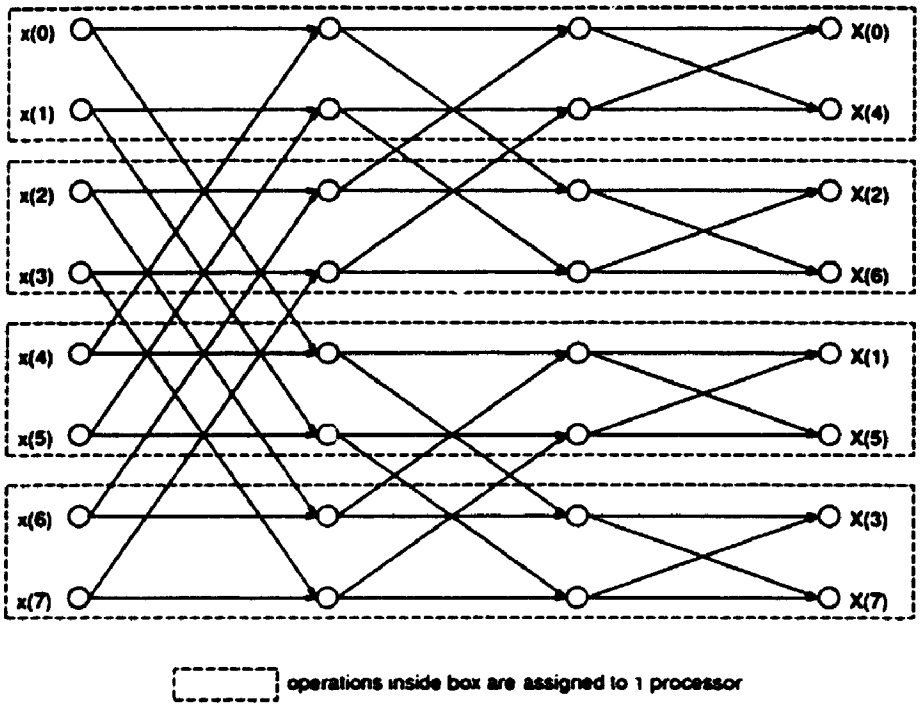
Calculating the FFT on a multiple processor system is more complex. Methods exist for performing the function on shared memory multiprocessors (Cvetanovic 1, 87), and many variations of multistage switching networks (Cvetanovic 2, 87), (Stone 5, 90). Some implementations are based on dividing the tasks among processors in an obvious manner, such as that shown in figure 2.2.2.3.

Figure 2.2.2.2 Cooley-Tukey 8-point, radix-2 algorithm, data in bit-reversed order.



It is evident that the arrangement of figure 2.2.2.3 requires a high degree of interprocessor communication, and some authors use intermediate shuffles and "twiddle factors" in an attempt to decrease the number of transactions. An algorithm for a globally parallel system, proposed by Bergland and Wilson (Bergland, 69), requires no interprocessor communication during the main computation period. The method requires supplemental arithmetic operations to achieve the parallelism, but the potential performance increase remains substantial. In the experiments, the multiprocessor FFT calculation tests adopt their procedure, which is summarized below.

Figure 2.2.2.3 Task assignment among four processors for 8-point FFT calculation.



Starting from equation 2.2.2.2, the DFT defining equation, it is necessary that N is factorable, as $N=r_1r_2$. The procedure requires each processing unit to have enough memory and computational ability to perform an r_2-point FFT. The following notational substitution is made:

$$W_s = e^{-j2\pi/s}$$

Indices n and k are reformulated as:

$$n = n_1 r_1 + n_0$$

$$k = k_1 r_2 + k_0$$

where:

$$n_1, k_0 = 0, 1, \ldots, r_2 - 1$$

$$n_0, k_1 = 0, 1, \ldots, r_1 - 1$$

The DFT equation can then be expressed as:

$$X(n_1, n_0) = \sum_{k_0=0}^{r_2-1} W_{r_2}^{n_1 k_0} W_{r_1 r_2}^{n_0 k_0} \left(\sum_{k_1=0}^{r_1-1} x(k_1, k_0) W_{r_1}^{n_0 k_1} \right) \qquad (2.2.2.20)$$

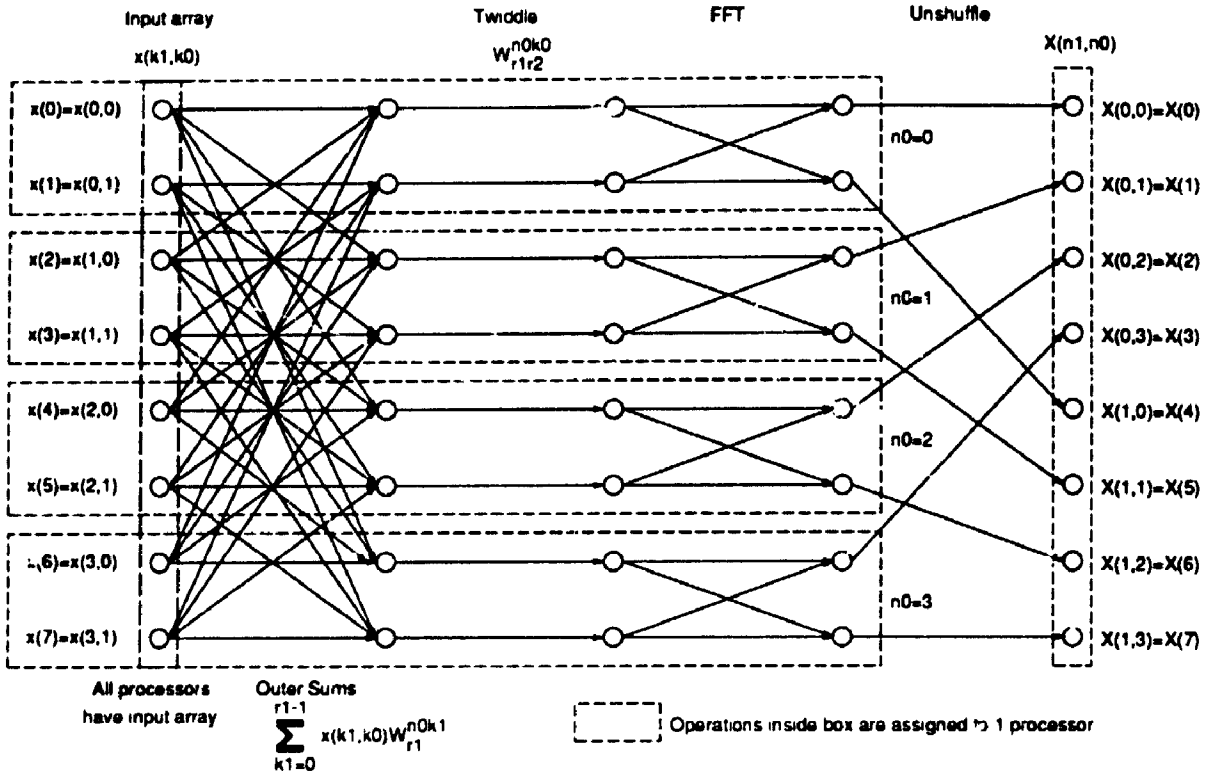
Equation 2.2.2.20 suggests the following computation strategy: Using r_1 processors, the memory of each is loaded with the entire input data array (the FFT is a *full-information* function, section 1.3.1.). The inner summation is performed simultaneously by each processor, requiring a number of multiplications. Each processor then performs multiplication of the *twiddle factors*, $W_{r_1 r_2}^{n_0 k_0}$, on the resulting r_2 points. Finally, the outer summation is an r_2-point Fourier transform of the twiddled data, which may be evaluated using any fast algorithm. The results appear in scrambled order, distributed among the r_1 processors, and a complete r_1-shuffle of N elements must be performed in order restore them to natural order[1]. A signal flow graph of the algorithm is shown in figure 2.2.2.4 for a system with four processors.

---

1 $S_{qc}(i)$,the q-shuffle function for $N=qc$ objects, is defined by:

$$S_{qc}(i) = qi \mod(qc) + \left\lfloor \frac{i}{c} \right\rfloor$$

Figure 2.2.2.4 Signal flow graph of four processor, parallel FFT algorithm (N=8)



Analysis of the number of complex arithmetic operations follows.

Let $r_2 = 2^p$ to permit radix-2 FFT's for each r_2-point FFT per processor. The number of multiplications and additions required in the r_2-point transforms is $p2^{p-1}$ and $p2^p$, respectively, as in the uniprocessor version, assuming the same FFT algorithm is used. The outermost summation requires $r_1 - 1$ additions and r_1 multiplications per input datum by each processor, for a total of $2^p(r_1 - 1)$ additions and $2^p r_1$ multiplications. The twiddle factor multiplication phase requires 2^p additional multiplications per processor. The number of operations for the parallel and uniprocessor algorithms may be compared when:

$$N = 2^\gamma = r_1 r_2 = r_1 2^p \qquad (2.2.2.21)$$

which necessitates:

$$r_1 = 2^q, \qquad q = \gamma - p \tag{2.2.2.22}$$

The total number of complex multiplications is:

$$p2^{p-1} + 2^p + 2^p 2^q \tag{2.2.2.23}$$

and the number of additions is:

$$p2^p + 2^p(2^q - 1) \tag{2.2.2.24}$$

If N=256, r_1=4, and r_2=64, a parallel system requires 512 multiplications per processor, compared to 1024 on the uniprocessor, a speed-up factor of only two. However, the summation phase of the algorithm yields exceptional cases when q=1 or q=2, since:

$$W_{r_1}^{n_0 k_1} = \pm 1 \qquad r_1 = 2 \tag{2.2.2.25}$$

$$W_{r_1}^{n_0 k_1} = \pm 1, \pm j \qquad r_1 = 4 \tag{2.2.2.26}$$

The multiplications degenerate to negations and real-imaginary transpositions, eliminating $2^p 2^q$ multiplications. Total operation counts are as shown in table 2.2.2.1. If multiplications are the significant limiting operations, a maximum speed-up factor of four may be achieved.

Table 2.2.2.1 Operation counts for 256-point FFT

Operation (complex)	Uniprocessor count	Parallel Processor count 4 processors
Multiplication	1024	256
Addition	2048	576

2.2.3 Frequency Domain Digital Filtering

Frequency domain filtering of sampled signals is a common application for which the Fourier transform is used. Filtering allows isolation or suppression of selected frequency components in a signal in order to perform some function, or to clarify information otherwise obscured. For example, an object present in an image may have its edges emphasized by high-pass filtering. Low-pass filtering may be employed to reduce the effects of high-frequency noise, enabling meaningful components of the signal to become evident. Filtering permits frequencies of interest to be amplified or attenuated (Gonzalez 1, 87).

Filtering may be accomplished by operating on the signal entirely in its native domain by the convolution method. Alternatively, the signal may be transformed into another domain, permitting subsequent operations to be performed using simpler, algebraic methods. The new signal must eventually be inverse transformed to its original domain (Dudgeon 1, 84). The convolution method, although conceptually simpler, requires a large number of arithmetic operations for large data sets. Since the Fourier transform can be computed using significantly fewer operations, the transform method of filtering is preferred. The term "frequency domain filtering" refers to the transform method of filtering time-domain signals, but it is applied to spatial signals as well (Gonzalez 1, 87). Equations 2.2.3.1 through 2.2.3.5 show the operations necessary to filter a time domain signal, $x(t)$, with a filter described in the frequency domain by $G(f)$, to yield a filtered, time domain signal $h(t)$. The sequence of operations is depicted in figure 2.2.3.1.

$$X(f) = \mathcal{F}\{x(t)\} \tag{2.2.3.1}$$

$$H(f) = X(f) \times G(f) \tag{2.2.3.2}$$

$$h(t) = \mathcal{F}^{-1}\{H(f)\} \tag{2.2.3.3}$$

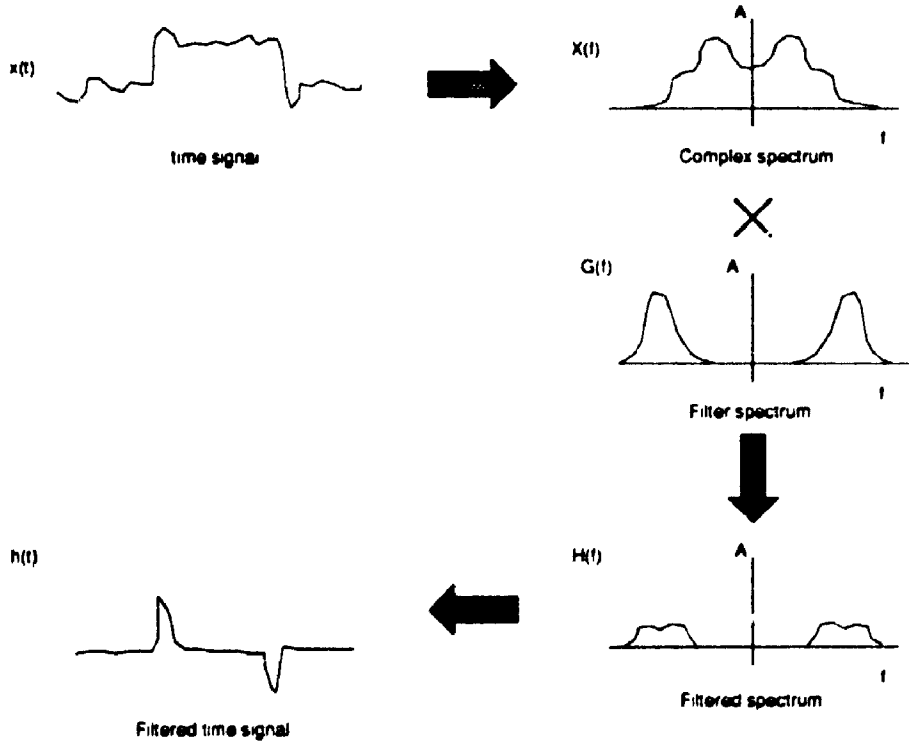
where:

$$\mathcal{F}^{-1}\{H(f)\} = h(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} H(f)e^{j2\pi ft}df \tag{2.2.3.4}$$

or:

$$\mathcal{F}^{-1}\{H(n)\} = h(k) = \frac{1}{N}\sum_{n=0}^{N-1} H(n)e^{j2\pi nk/N} \tag{2.2.3.5}$$

Figure 2.2.3.1 Frequency Domain Filtering



Frequency domain filtering may be accomplished using a parallel configuration of processors. The Fourier transform of the input signal is calculated using the algorithm described in section 2.2.2. The filter's spectral coefficients are pre-loaded into the various processors' local memories, and are distributed such that they reside in the memory of the processor which will calculate corresponding frequency components of the input signal. Filtering is performed as a series of complex multiplications. Given N points and r_1 processors, the number of multiplications per processor is N/r_1. New frequency-domain data result, and an inverse transform must be computed to represent the filtered signal in its original domain.

Before the filtered signal can be inverse-transformed using the same parallel algorithm as the forward transformation, each processing element requires the entire filtered-data array. The redistribution of data may be accomplished in a number of ways. The first strategy requires that each processor report filtered spectral data (N/r_1 points) to a designated processor. That processor rebroadcasts the accumulated spectrum back to the remaining processors (following any necessary data reordering), and the inverse process commences. A second strategy recognizes that the first incurs unnecessary processor idle time while they await an open channel to the designated processor. In the alternate method, a transmitting processor broadcasts its data to all of the other processors. Eventually, each unit has a local copy of the entire data array. Potential time savings may be significant, but the strategy requires an interconnection network with complete broadcast modes, and each processing unit must perform the data-reordering procedure. The inverse transformation proceeds as in the forward transform, with the usual sign-change in the exponential term. Finally, the filtered-signal data is extracted from the processing units by the designated processor, and reordered. The number of arithmetic operations required by the inverse transform is identical

to that of the forward transform. If the number of input data is N=256, r_1=4, and r_2=64 as in the previous examples (section 2.2.2), the theoretical speed-up factor (based on number of multiplications) is four.

2.2.4 Alternating Series Calculations

Calculation of infinite alternating series is rarely performed in application-oriented computing, since many series converge to known solutions, which are more efficiently computed using other means. However, alternating series exhibit properties which may be exploited for efficient calculation on the RMCS using a different configuration than in the previously described tests. The experiment is included to demonstrate and investigate the system's performance in such a configuration.

The series in equation 2.2.4.1 converges to some finite limit L if the conditions stated in equations 2.2.4.2 and 2.2.4.3 are satisfied.

$$a_0 - a_1 + a_2 - a_3 + \ldots = \sum_{n=0}^{\infty} (-1)^n a_n \tag{2.2.4.1}$$

$$a_0 \geq a_1 \geq a_2 \geq a_3 \geq \cdots \tag{2.2.4.2}$$

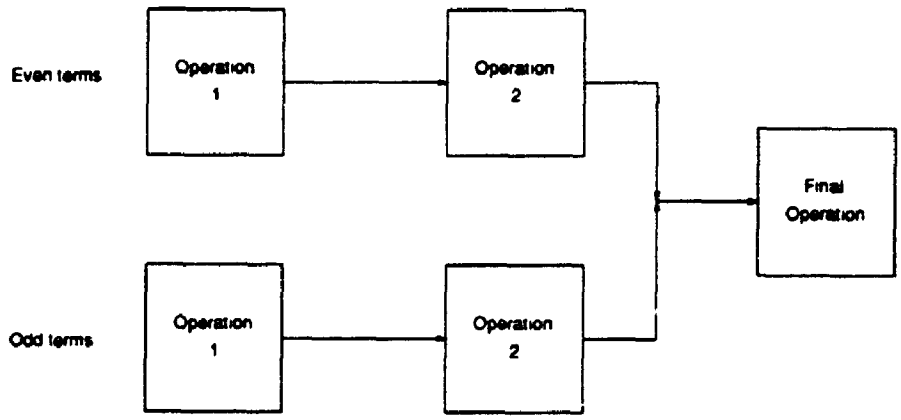
$$\lim_{n \to \infty} a_n = 0 \tag{2.2.4.3}$$

The limit L is known to be between s_n and s_{n+1} for all n, where $\{s_n\}$ is the sequence of partial sums (Shockley, 82). The rate at which alternating series converge depends on how quickly the progression in equation 2.2.4.2 converges. Equation 2.2.4.1 is separable, and therefore alternating series can be calculated as the difference of two sums, as shown in equation 2.2.4.4.

$$\sum_{n=0}^{\infty} (-1)^n a_n = \sum_{n=0}^{\infty} a_{2n} - \sum_{n=0}^{\infty} a_{2n+1} \tag{2.2.4.4}$$

The series summation may be computed by a system of processors configured in a parallel-pipelined fashion. Two parallel paths are established, each one consisting of a number of processors connected in series. The two sums of equation 2.2.4.5 may be calculated simultaneously, one in each parallel branch. The first processor in each branch launches new terms, or some intermediate form of them, into the pipeline, with subsequent processors calculating each a_n term in the series. The results finally arrive at the last processor in the branch, which adds the incoming data to form a cumulative sum. After a predefined number of terms have been added, a designated processor collects the two partial sums, and performs the final subtraction. Figure 2.2.4.1 shows a system of five processors configured with two parallel paths, and two processors in series per path. The fifth processor is assigned to perform the final subtraction.

Figure 2.2.4.1 System configuration for alternating series calculation.



Maximum theoretical performance assumes that each stage in the pipeline executes its task in equal time, τ. The pipelines each consist of p processors, and the total number of terms to be included in the computation is N, where N is even. Time for each pipeline to generate its partial sum is given by equation 2.2.4.5.

$$time_{pipe} = \left(p - 1 + \frac{N}{2}\right)\tau \tag{2.2.4.5}$$

The final subtraction is also assumed to require τ time, and thus overall time is given by equation 2.2.4.6.

$$time_{alt\ series} = \left(p - 1 + \frac{N}{2}\right)\tau + \tau = \left(p + \frac{N}{2}\right)\tau \tag{2.2.4.6}$$

The uniprocessor calculation time may be expressed in terms of τ as well, if each arithmetic operation is also assumed to take τ time. The uniprocessor must perform p operations per series term, for each of N terms, save one, yielding an execution time $(Np - 1)\tau$. Equation 2.2.4.7 expresses the speed-up factor as a function of N and p. Equation 2.2.4.8 shows that the maximum speed-up factor attainable is equal to twice the number of processors in the individual pipelines.

$$speed\text{-}up = \frac{Np - 1}{p + \frac{N}{2}} \tag{2.2.4.7}$$

$$\lim_{N \to \infty} \frac{Np - 1}{p + \frac{N}{2}} = 2p \tag{2.2.4.8}$$

2.3 Operational Overhead

The previous sections discuss relative performance issues based solely on arithmetic operation totals. It is a prevalent technique in estimating maximum relative performance increase between systems, or in comparing different algorithms for a single system. It assumes that the operations counted are the most time-consuming. Invariably, supplemental operations are necessary; simpler arithmetic functions, address calculations, loop counter modifications, and conditional-branch tests are common. The ancillary operations are

considered either to require insignificant time, or they are performed in constant proportion to the number of counted operations. When the assumptions are thus justified, predictions based on operation count are reasonably accurate.

When multiple processor systems are considered, additional conditions must be satisfied to utilize the same technique accurately. Multiple processor applications generally require data communication among processing elements, which often requires extra (*overhead*) time due to memory arbitration, semaphore maintenance, communications bandwidth, message framing, and processing element synchronization, all of which are implementation or architecture dependent. If the overheads are insignificant compared to the counted operations, then the operation count-based theoretical speed-up may closely match the observed speed-up. It is rare that the overheads can be ignored entirely, however.

There are three major sources of operational overhead in the RMCS: reconfiguration time; processor synchronization time; and data communication time. All are hardware-related functions, however, they are controlled by software resident in each cell's ROM, and are accessed using various software traps (section 3.7.4). Hence, some degree of operating system overhead is associated with the hardware factors. The following sections briefly describe the sources and nature of the RMCS's operational overheads.

2.3.1 Reconfiguration Overhead

Every application executed on the RMCS involves some network reconfiguration overhead. Chapter 3 describes how reconfigurability is achieved and controlled. All data communications between processing elements in the system utilize the reconfigurable interconnection network, whose topology is controlled by a single processor, designated the *master*; the remaining processors are referred to as *slaves*. The master configures the network by writing *configuration control words* to a set of registers residing in its memory map. Reconfiguration of the network may be overlapped with slave execution time, since

communication paths may sometimes be altered, without interfering with slave computation; otherwise, reconfiguration occurs sequentially. Reconfiguration overhead time is equal to the execution time of the instructions which modify the registers (assuming minimal PSR device reconfiguration delay, see section 3.3.1 and 4.1.1.3), and do not execute concurrently with slave processes. The programmer may modify the registers using individual memory modification instructions, or a monitor service routine may be called which overwrites all of the network configuration control registers on each invocation (section 3.7.4.8).

2.3.2 Processor Synchronization Overhead

The processing elements in the RMCS are autonomous. Since the master must not reconfigure the network at mid-transaction, some independent means to monitor and control slave processing is necessary. The *service request bus* (SRQbus) and its protocol (section 3.6) fulfills the function. The slaves signal the master at critical points in the program, such as prior to, or following, a data transfer cycle. When the master acknowledges the request, the slave continues processing; otherwise the slave is compelled to wait. The SRQbus does not allow slaves to monitor nor signal other slaves. The SRQbus strategy empowers the master with absolute control over program execution and data communications throughout the system.

Programs for the RMCS necessarily include code for execution by the master, as well as the slave processors. A slave requests service of the master using a TRAP call, which, upon entry, asserts a request, and returns after acknowledgment has been received (the *SRQASRT* trap, section 3.7.4.3). The master program likewise uses a TRAP call to service the various requests using any desired priority (the *SRQACK* trap, section 3.7.4.9). In both cases, the processor synchronization overhead includes the TRAP exception vector processing time, and the time to execute the instructions which implement prioritization. It is difficult to precisely measure the processor synchronization overhead; the time between call and return from the *SRQACK* routine may be measured at the master, however, if an

SRQ from a particular slave is expected, and it is not immediately forthcoming (the slave may be busy completing another task), the observed time is longer than the minimum time. A test program that does not call the *SRQACK* routine until the expected SRQs are known to be pending may be used; the execution time observed is for "guaranteed-pending" requests. However, typical behaviour is like that of the first case; the master calls the *SRQACK* routine concurrently with slave execution time, and it may spend much of its time awaiting requests, since it is normally not utilized for computations. The acknowledge time is slightly dependent on the number of requests to be serviced in each TRAP exception call, and it may fluctuate due to the asynchronous nature of the system. In the experiments, the guaranteed-pending request procedure was used to obtain approximate results.

2.3.3 Data Communication Overhead

Data communication overhead is theoretically the most significant overhead factor in the RMCS's performance. A message (consisting of a variable number of bytes) exchanged between processing elements traverses a number of PSR devices. Each byte transferred is asynchronously handshaken, and thus a transmitter cannot transmit a message until the receiver signals its readiness. Data transmit and receive programs for block oriented data exist in each processor's I/O subroutine repertoires. The data transfer overhead includes the time required to call the routines, the transmitter/receiver initial byte synchronization period, and the actual data transfer time, which is expected to be linearly related to data block-length for a fixed signal-path length. The path length affects transfer times since propagation delays of signals must be considered. The possibility exists that signal propagation time is significantly smaller than the time required by a processor to perform an iteration in the block transfer program; a byte could be completely transferred within the iteration time. In such cases, the limiting factor is not the network bandwidth, but the processor and I/O port performance. The effect of signal propagation delay is

inconsequential, and transfer time will be independent of path-length below some limit. The data transfer overhead may be expressed as a linear equation of the form shown in equation 2.3.3.1.

$$t_{data-transfer} = (byte\ \ count) \times m + b \qquad (2.3.3.1)$$

where b represents the fixed overhead cost incurred per block transfer, and m is the time per byte transferred. Analogous to the processor synchronization overhead, the value of b is difficult to measure accurately. A number of tests may be conducted using various data-block length transfers, and regression analysis used to determine the fixed penalty. The accuracy of the measurement is affected by receiver and transmitter synchronization. In the experiments, data transfer times and overheads are measured in cases where receivers are always ready prior to transmitters calling their data-output routines. When measurements are made at a receiver, a similar provision is made. The procedure standardizes the tests, and minimizes the synchronization time. The fixed synchronization cost is nonetheless expected to be small compared to transfer times of even moderate block lengths (10's of bytes), so some small amount of error may be inconsequential to calculating predictive estimates of system performance.

3 Prototype System

This chapter focusses on the hardware and software of the reconfigurable multicomputer system developed for the thesis. The major hardware components are described in detail, and overview is given of the multicomputer system topology and operating principles. The system-level programs developed to control the multicomputer are presented.

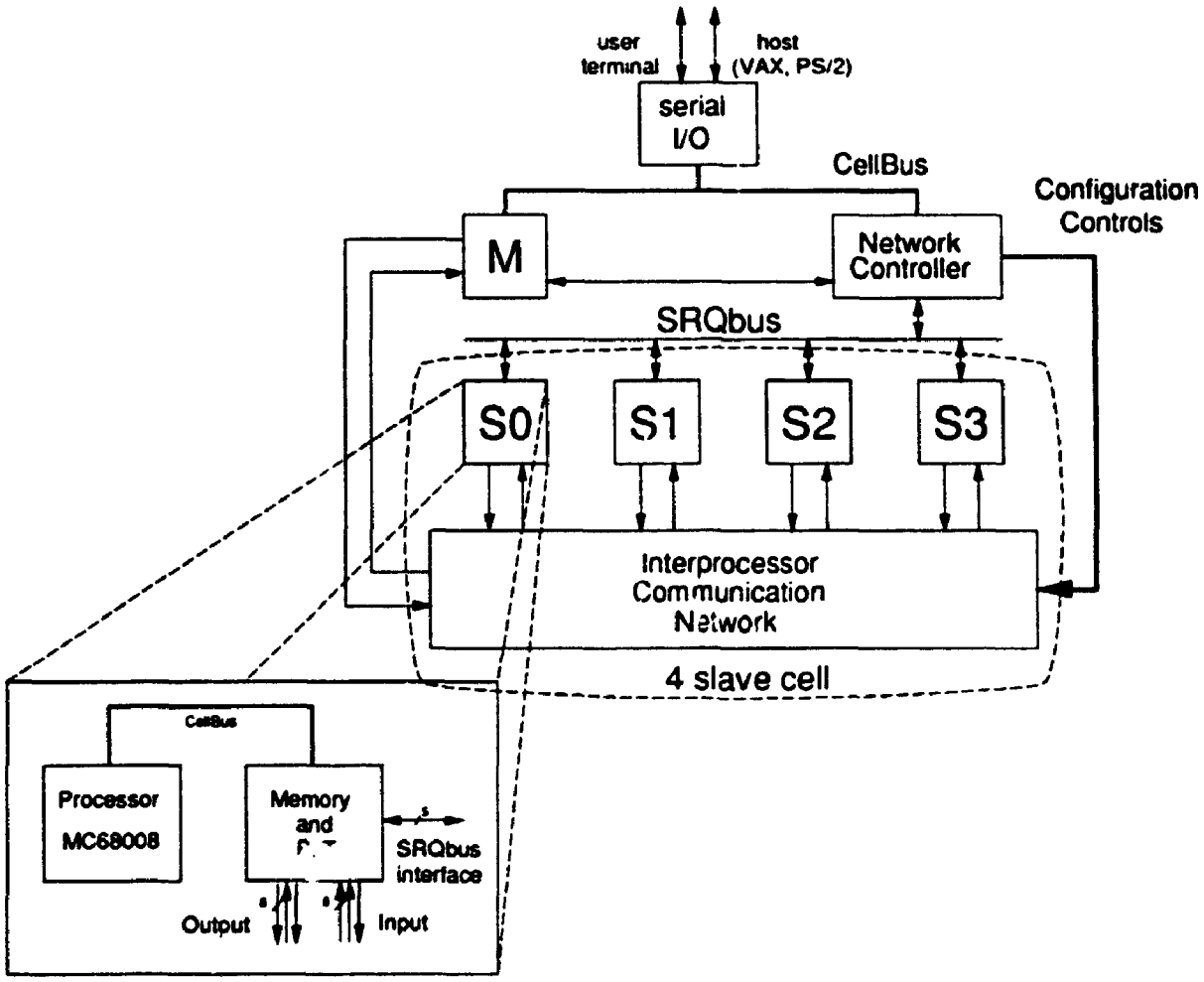
3.1 General System Description

The prototype reconfigurable multicomputer system (RMCS) consists of five processing elements or cells, four of which are designated as *slave* cells, the remaining one, the *master*. The slave cells are comprised of identical hardware and system software, and they can perform equivalent tasks. The master cell consists of the same core hardware as the slaves, with additional modules to provide host computer and user terminal interfaces, and network control. The master cell incorporates system software similar to that of the slaves, with supplementary routines to facilitate supervision of slave processing. All processor cells exchange data via the interprocessor communication network (ICN), which embodies an array of custom-designed *programmable signal router* (PSR) devices. The ICN and *network controller card* (NCC), make the system reconfigurable. Under the control of the master, communication channels among processor cells may be altered to optimize the interconnection topology for the current program. Reconfiguration may occur within the framework of a program, to optimize performance of sub-tasks within more complex, larger tasks. Data transfers are synchronized by the participating processing elements using an asynchronous, 2-wire handshake protocol; data transfers do not require master supervision. There is no shared memory in the RMCS, nor a common clock signal; all interprocessor transactions are asynchronous.

Each slave processor accesses a *service request bus* (SRQbus) via a parallel I/O port, and an interface circuit. The master processor monitors the processing state of the various

slaves using the SRQbus. Data is not exchanged using the SRQbus; it implements a "ready/continue" signalling strategy only. The SRQbus enables the master to maintain slave processor synchronization throughout execution of a program. The bus protocol does not include facilities for slaves to exchange status information with other slaves. The strategy strengthens the role of the master processor as system overseer, and provides system control (and therefore programming) uniformity. Figure 3.1.1 shows a block diagram of the prototype system.

Figure 3.1.1 Block Diagram of Prototype System.



The four-slave system by itself constitutes a unit-cell. Expansion of the system to include additional processors is accomplished by replication of the four-slave cell and ICN. Larger systems consisting of many cells are themselves reconfigurable, and intercellular connections determine the scope of available topologies. The individual system components are described in detail in the following sections.

3.2 Autonomous Processor Cell

The RMCS constituent processors are autonomous, high-performance microcomputer systems. Each *autonomous processor cell* (APC) comprises an 8 MHz MC68008 microprocessor, an MC68230 parallel interface/timer (PI/T) providing three parallel I/O channels, 64 kbytes of RAM and 8 kbytes of ROM. The MC68008 was chosen since the 68000 architecture and instruction set is well accepted as superior to many others, the internal 32-bit register architecture is suitable to perform complex arithmetic calculations quickly and efficiently, programming tools are widely available, and the instruction set allows the processor to perform well in register, memory and I/O intensive applications. The 8-bit external data bus matches the ICN data-path width; it also reduces the hardware cost of the prototype system, while providing a performance level comparable to that of processors with 16-bit buses (Motorola 1, 86).

Physically, each cell consists of a CPU board, a memory and parallel I/O board, and a backplane bus. The bus (hereinafter referred to as Cellbus) is an asynchronous, expanded local bus of the microprocessor, with seven interrupt priority levels, external system reset and halt signals, address translation, synchronous bus cycle capability, bus cycle re-run, and three-wire daisy-chain bus arbitration to accommodate alternate bus masters. The CPU board design and bus structure permit system expansion boards to consist solely of memory and peripherals, and necessary interface logic to implement either asynchronous or synchronous processor accesses, and interrupts.

The APC hardware is similar to that described in Smeulders (Smeulders, 88) with the addition of an MC68230 PI/T device, and its Cellbus interface. Block diagrams of the APC CPU card and Memory & PI/T card are shown in figure 3.2.1 and figure 3.2.2, respectively. Schematic diagrams for the subsystems are presented in Appendix B and Appendix C. Details of the APC hardware (bus signal descriptions, bus protocol, and system performance), are the subject of Smeulders (Smeulders, 88). Due to the MC68230's timing characteristics and Cellbus interface, read operations by the processor occur in four system clock periods, while write operations require five.

Slave processor cells do not require a host/terminal interface; only the master processor incorporates a serial I/O card for those purposes. The master is responsible for providing the slave processors with the instructions and data for a task. Programs stored on a host-computer system are downloaded to the master; the programs include instructions for distributing slave programs and data, as well as the master's system control code for the application. A single terminal interface exists as well, enabling the user to communicate directly with the master. The serial I/O module is completely described in Smeulders (Smeulders, 88). Figure 3.2.3 is a block diagram of the master's serial I/O module; schematic diagrams are presented in Appendix D.

Since a number of identical system units are used repeatedly to implement the RMCS, two-layer printed circuit boards for the CPU, Memory and PI/T, bus backplane, and serial I/O modules were designed and fabricated for the study. All printed circuit board designs, device placement, bus and connector pinout specifications are shown in Appendix E.

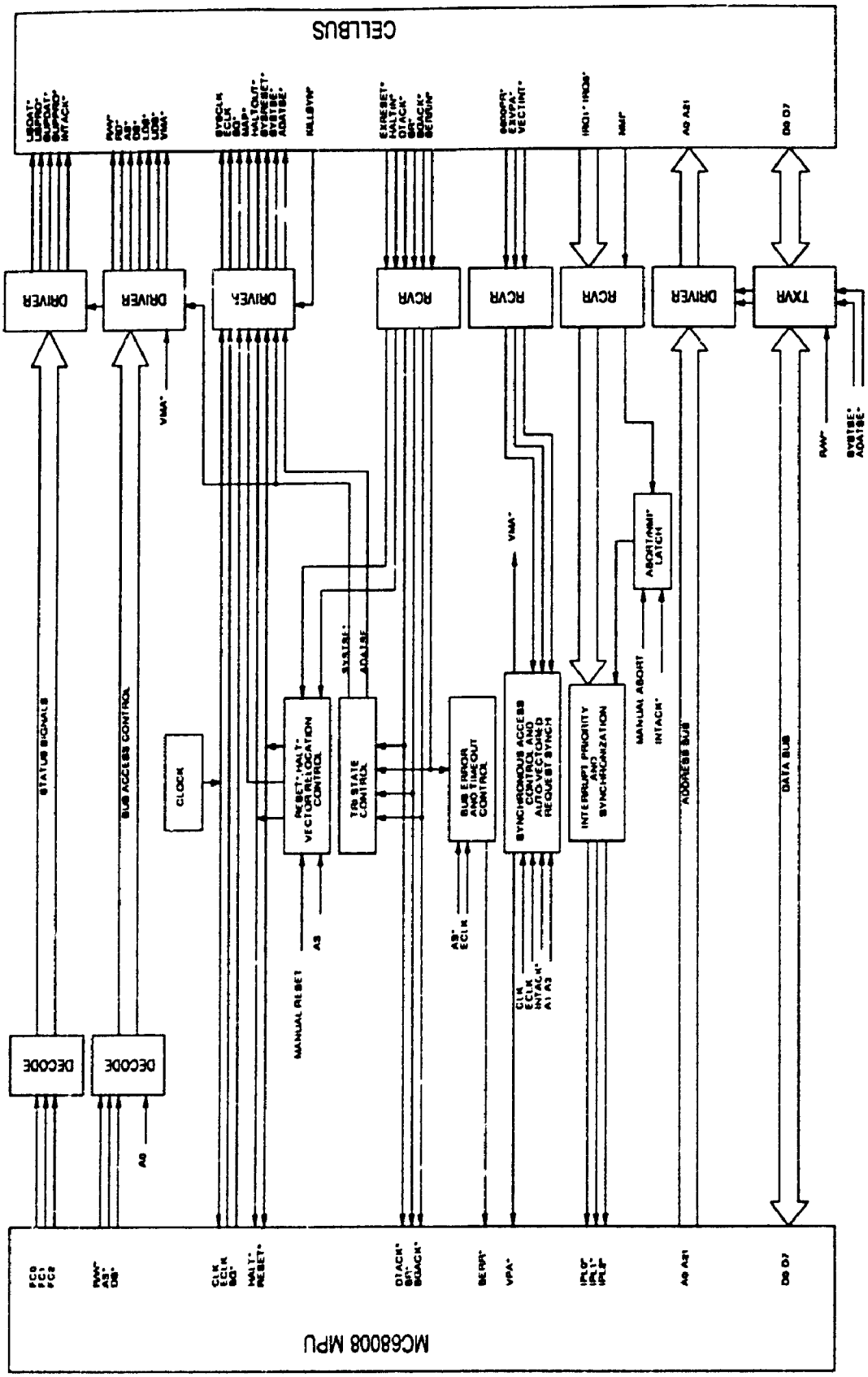


Figure 3.2.1 Block diagram of APC CPU module.

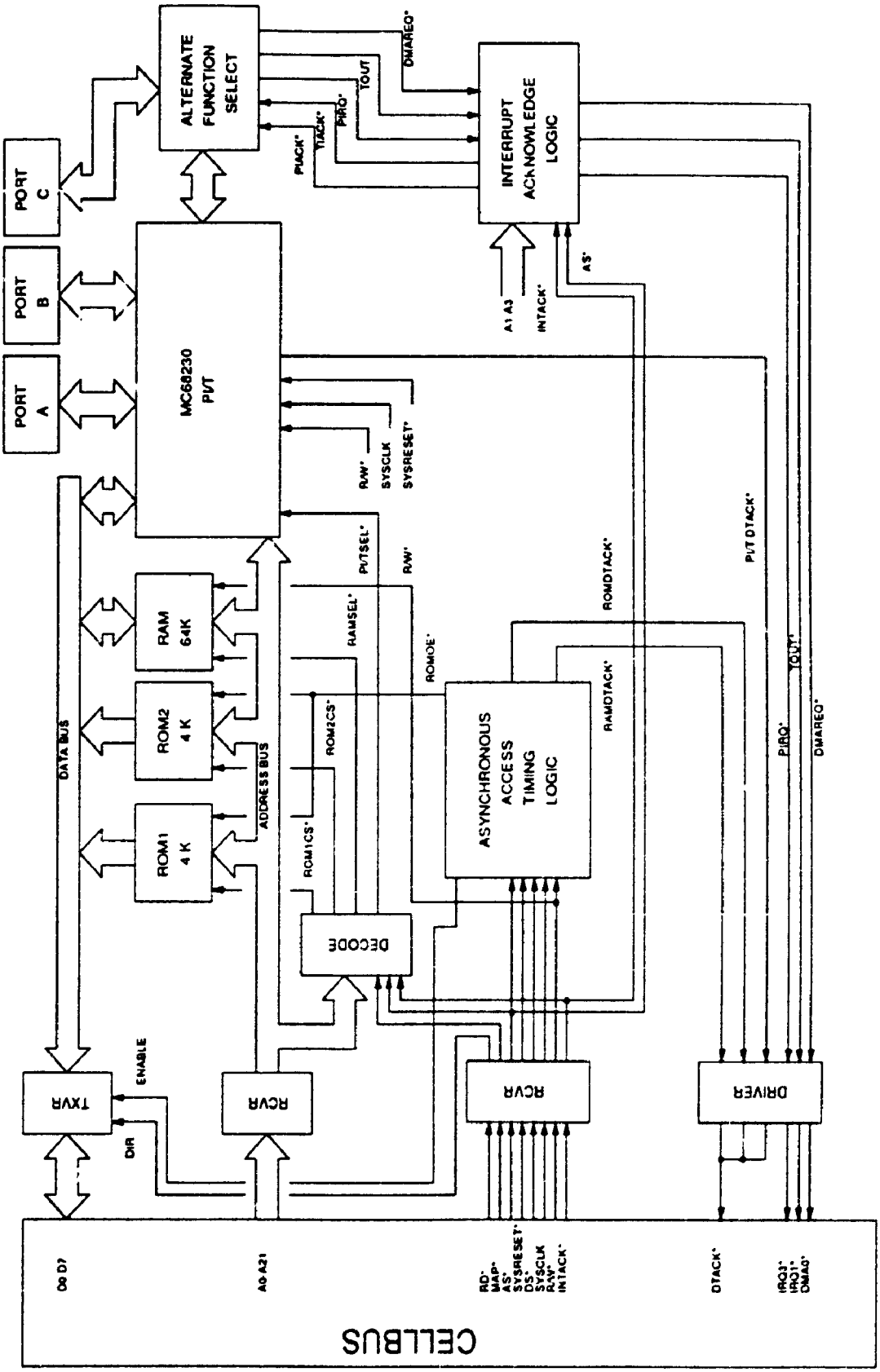


Figure 3.2.2 Block diagram of APC Memory and PI/T module.

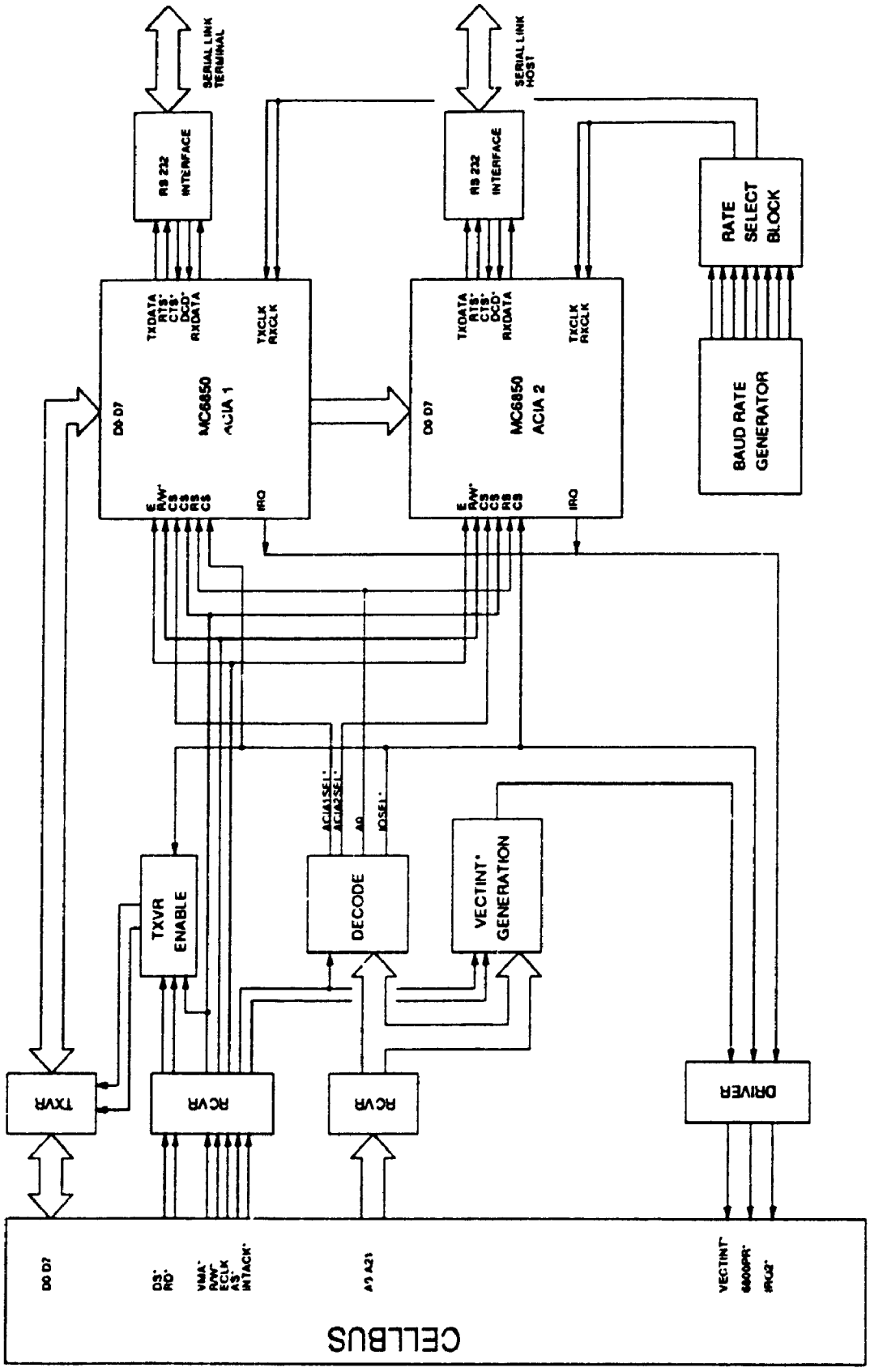


Figure 3.2.3 Block diagram of Master Processor's Serial I/O module.

3.3 Programmable Signal Router

The reconfigurability of the RMCS is achieved via an array of *programmable signal router* (PSR) devices designed and fabricated specifically for the project. This section presents the functional description, operational principles, and implementation details for the device. A brief section is devoted to applications and interfacing strategies for the PSR. It emphasizes the generality of the device design, and its versatility.

3.3.1 Functional Description

The PSR is similar in many ways to the *crossbar switch* commonly referred to in the literature describing multistage interconnection networks [section 1.3.5], but it incorporates fundamental differences which make it unique, and highly flexible.

The device comprises two 9-bit input ports: eight bits for data, one for asynchronous data transfer handshaking, designated as *data available* (DAV*), and two, 9-bit output ports. The device may be configured to transmit data (and DAV*) from an input to either or both output ports. Another signal, *data accepted*, (ACC*) is controlled by the receiving system to signal the transmitter that a transfer has been completed, or may commence (in the latter case, it may be referred to as *ready for data*). The design allows single and multistage interconnection networks to be implemented with reduced circuit board area and hardware. The PSR does not provide a bidirectional processor-to-memory interface, nor does it arbitrate multiple access requests or enqueue messages, as some multistage networks require; it is unidirectional. The RMCS architecture does not call for bidirectionality. Multiple PSR's and additional hardware may be used to implement the specialized crossbars.

The configuration of data paths within the device is controlled by a 4-bit *configuration control word* (CCW). Two bits determine internal data path routing, the remainder control enabling of the outputs, which allows isolation of selected ports. An external unit controls configuration of the data paths among processors, and the data transfer synchronization is

provided by the sending and receiving units. The external control strategy introduces sufficient flexibility to permit alternate configuration control methodologies to be utilized (supervisory processor control, address dependant routing, etc.). Figure 3.3.1.1 shows a block diagram of the PSR. A summary of the pin designations and their functions is given in table 3.3.1.1.

Figure 3.3.1.1 Programmable Signal Router block diagram.

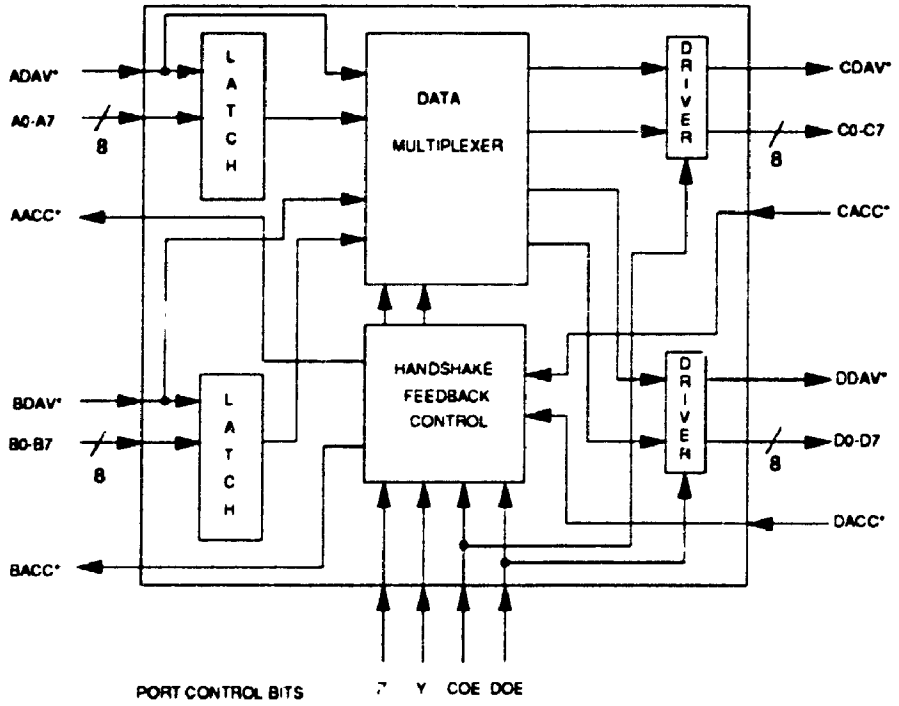


Table 3.3.1.1 Programmable Signal Router pin designations and descriptions.

Pin #	Designation	Description
3-10	A0-A7	Port A data input
57-64	B0-B7	Port B data input
48-41	C0-C7	Port C data output
31-24	D0-D7	Port D data output
11 65	A_{DAV}* B_{DAV}*	Port A (B) data available input. Signals port A (B) data available and latches data currently on A0-A7 (B0-B7) to the internal data latches.
40 23	C_{DAV}* D_{DAV}*	Port C (D) data available outputs. Reflects state of A_{DAV}* or B_{DAV}* depending on the current data path configuration.
39 22	C_{ACC}* D_{ACC}*	Port C (D) data accepted. Asserted by a receiver at Port C (D) to signal that data has been accepted, or receiver is ready for new data.
12 66	A_{ACC}* B_{ACC}*	Port A (B) data accepted. Reflects the state of C_{ACC}* or D_{ACC}* or both, depending on the current data path configuration.
13-16	PC0-PC3 (Z,Y,C_{OE}, D_{OE})	Port configuration control bits 0-3. The state of the PC bits determines the current data-path configuration.
21,56	VDD	+5 V power supply
55,32	VSS	0 V power supply (ground)

3.3.2 Device Operation

The PSR may be configured to provide all possible connections between the two input ports and two output ports. The active configuration is determined by the state of four port configuration control bits (PC0-PC3) comprising the *configuration control word* (CCW), defined in figure 3.3.2.1.

The configurations corresponding to valid control words are shown in figure 3.3.2.2.

Figure 3.3.2.1 PSR Configuration Control Word bit assignments.

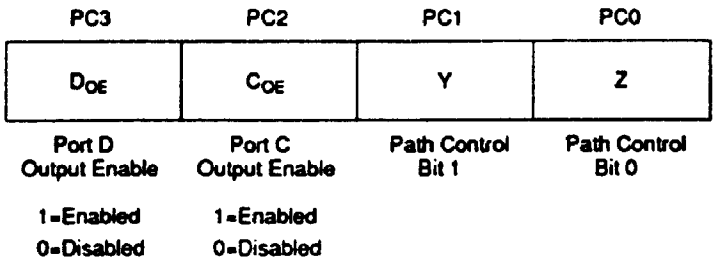
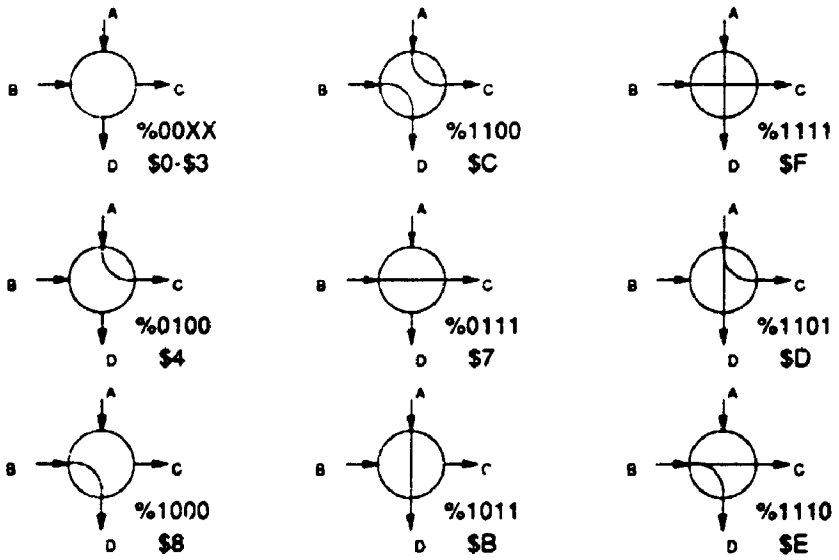


Figure 3.3.2.2 PSR Configurations and correspondir Control Words (X=don't care).



The remaining seven control words result in repeated or illogical port configurations. Invalid words presented to the device cause output port data and DAV* signals to be driven to the high state, as well as the input side ACC* signal. The same states exist when an output port is disabled.

A data transfer cycle between ports is summarized in the following paragraphs, assuming a transmitter connected to port A, a receiver at port C, and a configuration which provides a path from port A to port C.

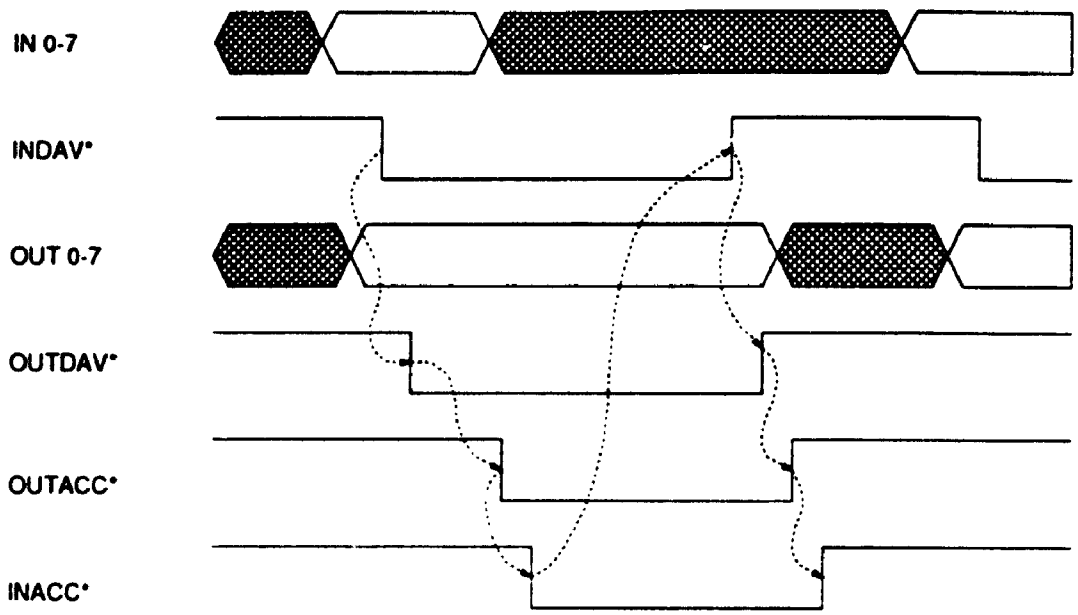
The transmitter applies data to pins A0-A7. The data is transparently reflected to pins C0-C7 with a small delay. When data is stable, the transmitter asserts A_{DAV}* low, latching data at the device, hence locking the data valid at C0-C7. The transmitter may allow its data to become invalid at any time thereafter, but it must keep A_{DAV}* asserted to maintain valid data at C0-C7. The A_{DAV}* signal is reflected to C_{DAV}*, which the receiver may use to latch the currently available data. Upon acceptance, the receiver asserts C_{ACC}* low. This signal is reflected to A_{ACC}* and thus to the transmitter, signalling that data was accepted, and A_{DAV}* may be negated. Negation of A_{DAV}* is reflected to C_{DAV}* as usual, and the receiver may negate C_{ACC}*. A_{ACC}* is negated at the transmitter side, finally indicating to the transmitter that the transfer cycle is complete, and another may commence. Correct operation of a series of PSR devices along a given data communication path requires that the delay imposed by the device on data signals is equal to or less than that imposed on the corresponding DAV* signal.

All transfers follow the described process, including broadcast transfers, in which case both C_{ACC}* and D_{ACC}* must be asserted low before the input port A_{ACC}* or B_{ACC}* signal is asserted low.

The data transfer cycle is asynchronous; timing is controlled by the DAV* and ACC* protocol, and no maximum transfer time is imposed. Figure 3.3.2.3 shows the data transfer cycle timing described above. Tests were performed to determine critical timing characteristics of the device. Results are presented in section 4.1.1.5.

The flexible design of the PSR permits variations of the protocol described above, depending on the peripheral used for data transfer between system units. In particular, the

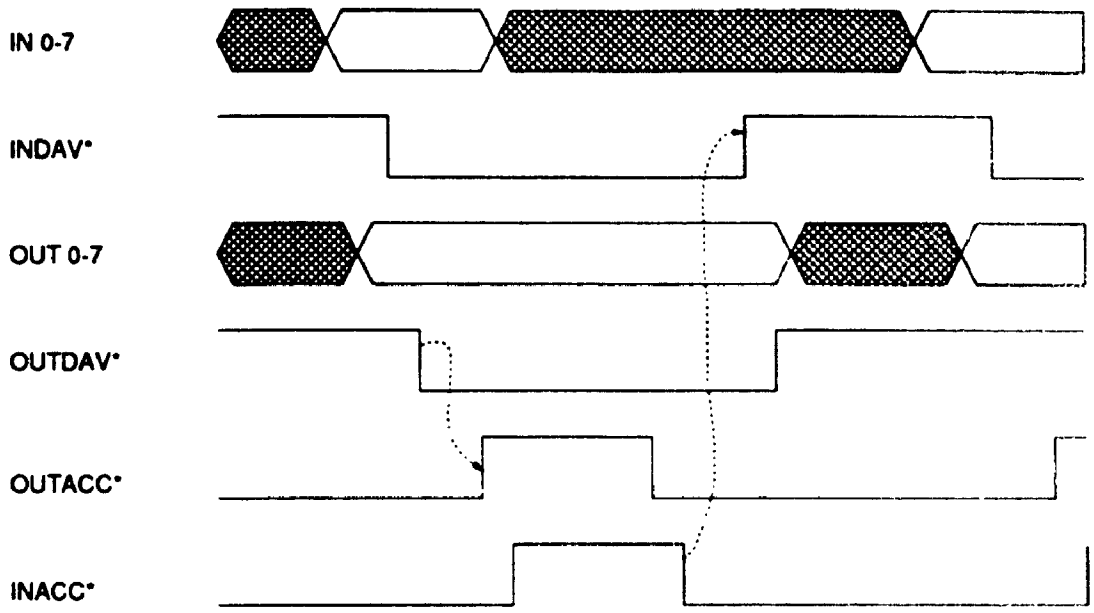
Figure 3.3.2.3 Data Transfer Cycle timing.



MC68230 PI/T lends itself better to a protocol where the A_{ACC}*/C_{ACC}* pair indicates *ready to accept* data. The negative edge of C_{DAV}* negates the receiver's C_{ACC}* immediately, and it is reasserted after the receiving processor has read the newly available data from its PI/T data register. The negative edge of A_{ACC}* at the transmitter immediately negates the A_{DAV}*/C_{DAV}* signal, and the next transfer may commence. The protocol is shown in figure 3.3.2.4. The necessary connection between two MC68230 PI/T's is shown in figure 3.3.5.2.1, as part of a later section on data-port interfacing.

Some implementations may take advantage of input data latch transparent operation when A_{DAV}* or B_{DAV}* signals are negated. The PSR may be used in systems without asynchronous handshaking if some other method of data synchronization between communicating devices is employed, such as a common clock. Since the internal data latches are transparent, a synchronous data transfer pipeline cannot be implemented without additional sequencing hardware and signals. Such pipelining was specifically not desired

Figure 3.3.2.4 Data Transfer Cycle timing, MC68230 I/O ports (interlocked handshake modes).



in the system, since it imposes potentially unnecessary delays and increased complexity. Transmitted data propagates directly to the receiver(s); the latches are provided only to permit transmitter invalidation of data signals following assertion of DAV*, which may be the case with certain peripheral devices. The transmitter may assert its output data signals for the next byte to be transmitted, and control its subsequent propagation with DAV*.

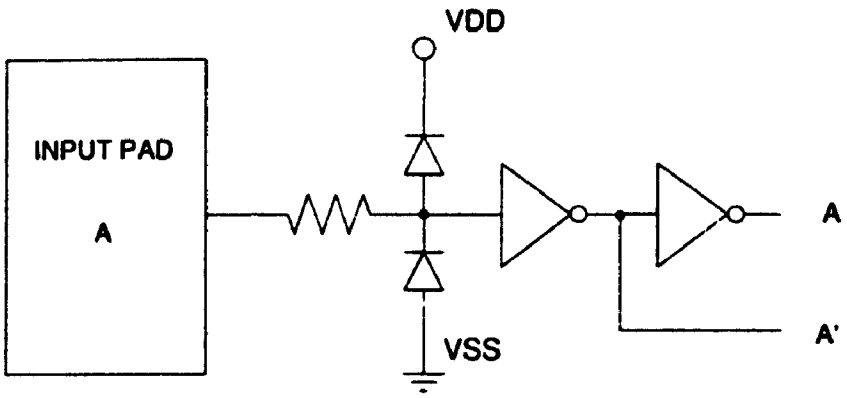
3.3.3 Internal Design of the PSR

This section discusses elements of the PSR's internal design.

3.3.3.1 Input Bonding Pads and ESD Protection Network

All input signals of the PSR device are high impedance loads. Each input circuit incorporates an electrostatic discharge (ESD) protection network comprising two diodes and a diffused resistor. The pad circuitry provides the input signal to the internal circuits in inverted and non-inverted conditions.

Figure 3.3.3.1.1 Input pad circuit diagram.

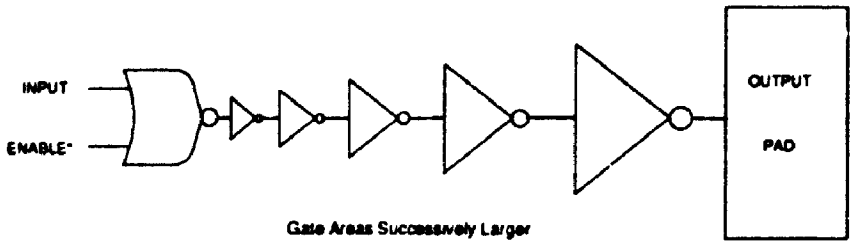


3.3.3.2 Output Circuits and Bonding Pads

All output signals of the PSR are buffered by 2-state output pad circuits. Each consists of four series-connected inverters with successively larger gate areas, providing increased output current with minimized propagation delay.

The signals driving C0-C7, D0-D7, C_{DAV}* and D_{DAV}* are logically ORed with the inverted C_{OE} and D_{OE} port configuration control bits. When a port is enabled, the logical state of the transmitting input appears on the corresponding output, otherwise, the output signals are driven high.

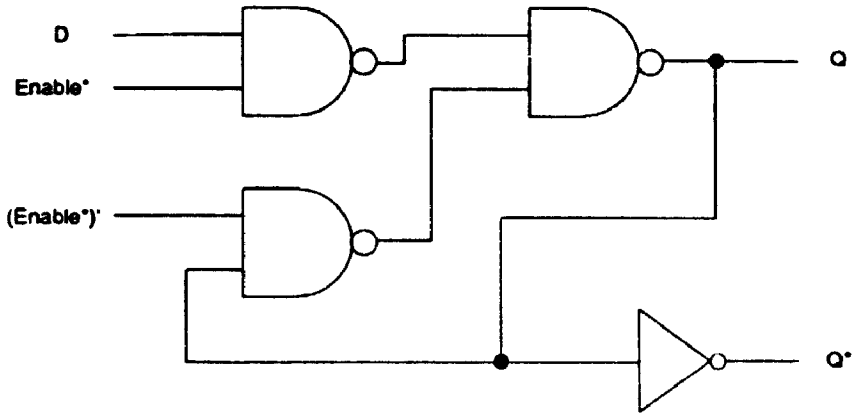
Figure 3.3.3.2.1 Output pad circuit diagram.



3.3.3.3 Data Latches

The data latches used to temporari' . tore input data for the duration of a transfer cycle are transparent latches with active-low enable signals. The circuit diagram for a one-bit latch is shown in figure 3.3.3.3.1.

Figure 3.3.3.3.1 Transparent Data Latch with active-low enable.



A set of eight latches is used for each input-port data latch (figure 3.3.3.1), with the enable signals connected to each. The latch enable signals are A_{DAV}* and B_{DAV}* input signals for the Port A and B data latches, respectively.

3.3.3.4 Data Path Routing Multiplexer

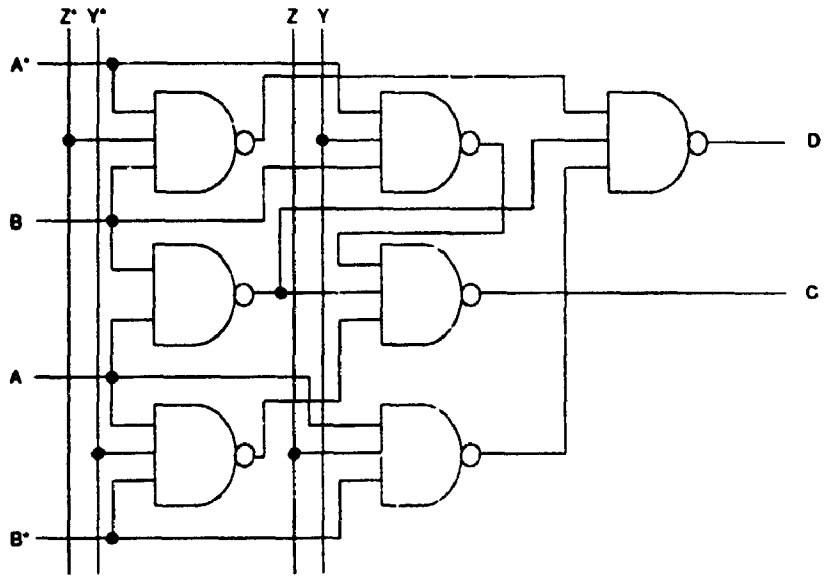
Data path routing is controlled by port configuration control bits Y and Z, and is realized using a multiplexer circuit. Boolean expressions for C_{data} and D_{data} outputs which result in the correct control, data input, and data output relationship (reduced to NAND, NOR, and INVERT functions for fabrication purposes) are as shown:

$$C_{data}=((A_{data}\bullet B_{data}'\bullet Y')'\bullet(A_{data}'\bullet B_{data}\bullet Y)'\bullet(A_{data}\bullet B_{data})')'$$

$$D_{data}=((A_{data}\bullet B_{data}'\bullet Z)'\bullet(A_{data}'\bullet B_{data}\bullet Z')'\bullet(A_{data}\bullet B_{data})')'$$

A one-bit data multiplexing cell is shown in figure 3.3.3.4.1.

Figure 3.3.?.4.1 One-bit, 2 × 2 data multiplexer.



The circuit shown was repeated a total of nine times: eight for data path routing, and an additional unit to route DAV* signals. The block of nine multiplexers implements the *Data Multiplexer* unit of figure 3.3.3.1.

3.3.3.5 Handshake Feedback Control Logic

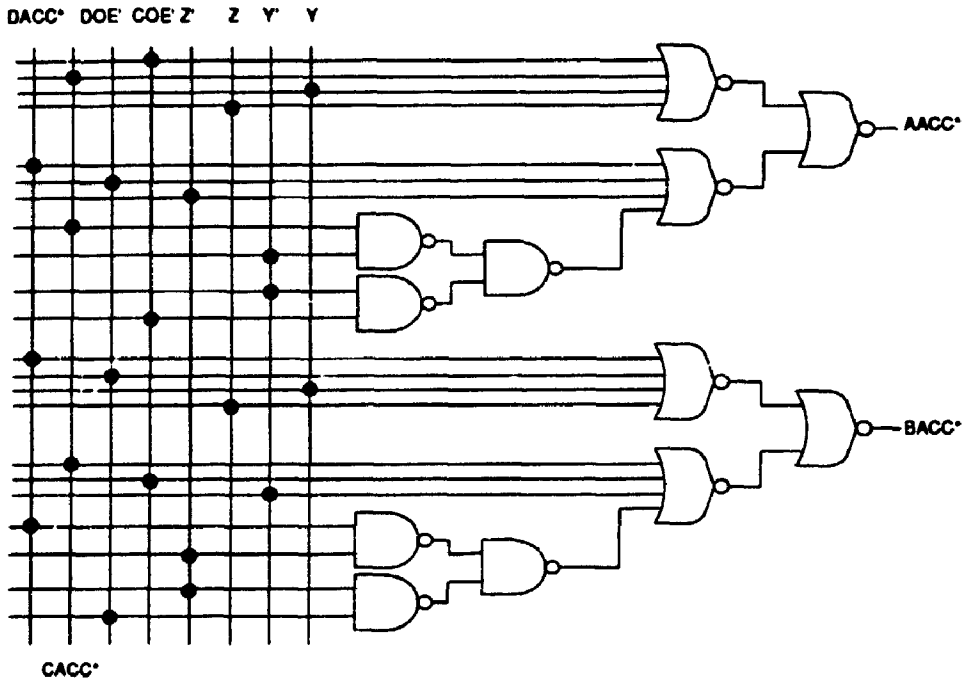
Since input data can be transmitted to either a single output port, or both (broadcast mode), routing of the handshake signals C_{ACC}* and D_{ACC}* to A_{ACC}* and B_{ACC}* is more complex than data path routing. Provision exists for broadcast modes, where both C_{ACC}* and D_{ACC}* must be asserted for the appropriate A_{ACC}* or B_{ACC}* to be activated. The following Boolean expressions (reduced to NAND, NOR and INVERT functions for fabrication purposes) for output signals A_{ACC}* and B_{ACC}* describe the necessary logical relations.

$$A_{ACC}^{*}=(\{C_{ACC}^{*}+C_{OE}'+Y+Z\}'+\{D_{ACC}^{*}+D_{OE}'+Z'+([Y'\cdot C_{OE}']'\cdot[Y'\cdot C_{ACC}^{*}]')'\}')'$$

$$B_{ACC}^{*}=(\{D_{ACC}^{*}+D_{OE}'+Y+Z\}'+\{C_{ACC}^{*}+C_{OE}'+Y'+([Z'\cdot D_{OE}']'\cdot[Z'\cdot D_{ACC}^{*}]')'\}')'$$

The handshake feedback control circuit prescribed by the above expressions is shown in figure 3.3.3.5.1.

Figure 3.3.3.5.1 Handshake Feedback Control circuit.



From the preceding discussion, a Muller-C element (Mead 1, 80) is an apparently suitable element for incorporation with the handshake feedback control circuitry, and would aid in synchronization of data transfers in the broadcast mode. The standard C-element introduces a potential timing hazard when the MC68230 PI/T is utilized in its interlocked handshake modes, since the signalling is not the standard 4-cycle protocol (Mead 2, 80). A related, but more easily accommodated hazard may occur when a C-element is not provided. The potential hazard exists when two processors are receiving broadcast data while executing receive instruction loops of significantly different lengths (*i.e.* one receiver is executing intervening instructions between input register read operations), or the data path length between receivers and the broadcasting device is significantly different (a path length with

delay approxi[illegible]ly equal to a one-byte transaction time with the nearest receiver). It is unreasonable to expect processors to execute different data receive loops, since the slowest receiver causes the others to wait, reducing efficiency. Thus, all receivers should execute an optimized receive-loop, and extra operations required of selected processors are performed subsequently. The hazard is not encountered if broadcast path length differences are below the limit. A latched version of the C-element is required if the hazard is to be eliminated altogether when the MC68230 PI/T is used. The specialized element was not incorporated to preserve the generality of the device design, and therefore its applicability for use with a wide range of peripheral devices. The restrictions on processor timing and propagation paths must therefore be observed in the prototype system.

3.3.3.6 Buffers

Since Port Control bits Y,Y' and Z,Z' drive many gates each, the signals are buffered before connection to the data path multiplexer by two series-connected inverters.

3.3.3.7 Power Distribution

The power connections labelled VDD and VSS are facilitated by the usual square metal pads. The I/O pad circuitry is powered separately from the internal circuitry. All of the external power pins must therefore be connected to VDD and VSS supplies for proper device operation.

3.3.4 Implementation and Fabrication

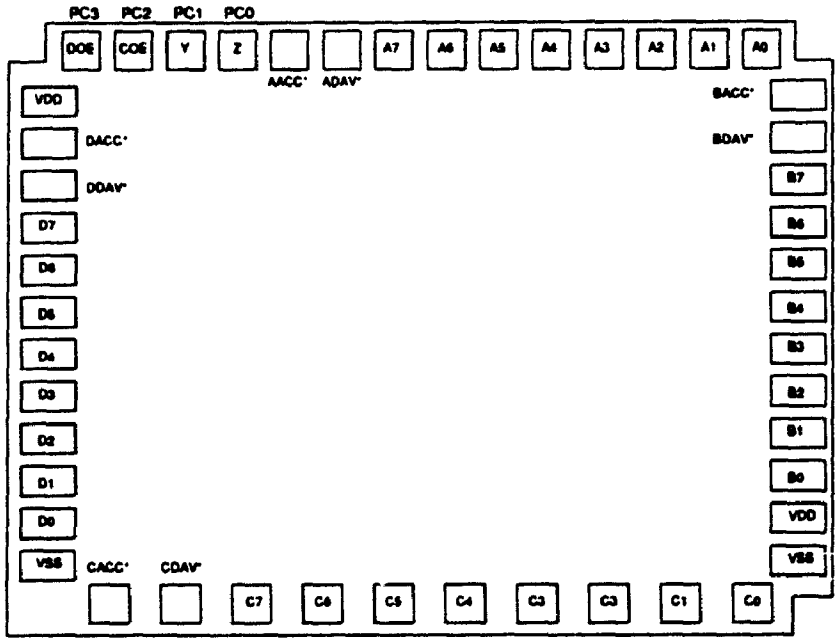
The PSR design is specified in the CIF geometrical specification language (Hon, 80), and is implemented using 3-micron, 2-metal layer CMOS technology. The design rules and fabrication restrictions are described in CMC (CMC, 89).

The devices are fabricated by Northern Telecom (Canada) on 4 mm square silicon dies, and are packaged in ceramic, 68-pin PGA chip carriers. Details of the I/O pad characteristics are documented in Chum (Chum, 89). whose library of pad and logic cell designs was made available to the author.

3.3.4.1 Pad Layout

The I/O pad designations are shown in a simplified diagram of the actual die in figure 3.3.4.1.1.

Figure 3.3.4.1.1 Simplified Die Layout.



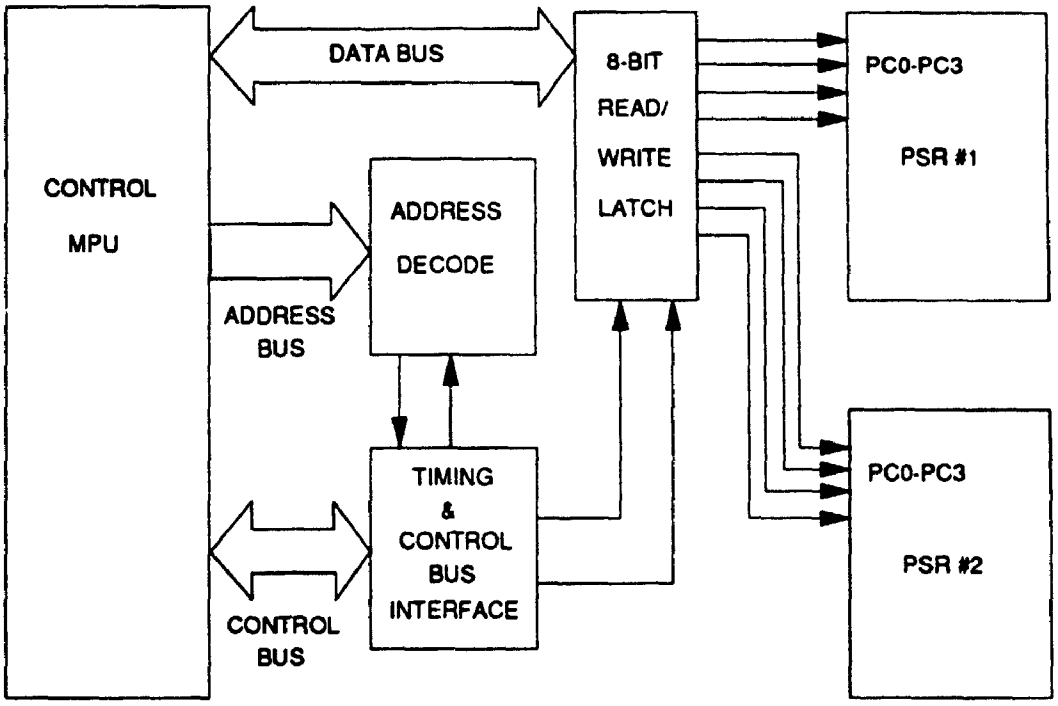
3.3.5 Interfacing Techniques

This section briefly discusses control processor and data channel interfacing, and data path-width expansion techniques.

3.3.5.1 Control Processor Interface

The port control bits PC0-PC3 must be controlled externally. A suggested method employs a read/write latch (such as a 74ALS666) which resides in the memory or I/O address map a control processor. Since only four bits are necessary to control a PSR, a single 74ALS666 may control two devices, while occupying only a single system address.

Figure 3.3.5.1.1 Typical control processor interface.



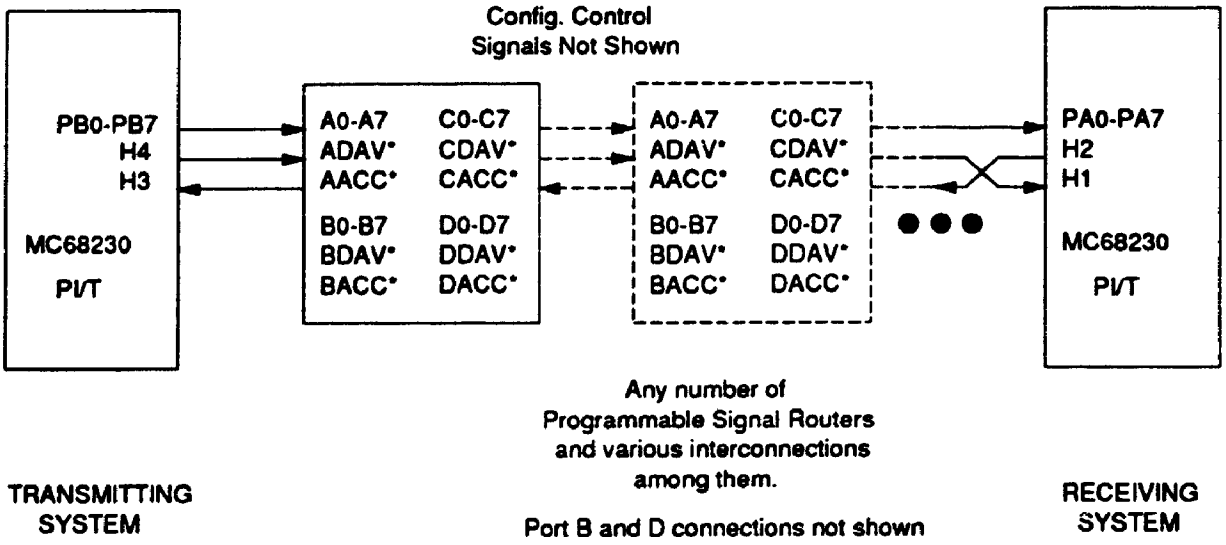
Integration of the read/write latch with the PSR device was considered, to reduce the need for external hardware. One was not included, however, to keep the module usefulness in systems with increased bus-width (section 3.3.5.3), and different timing specifications.

3.3.5.2 Data Port Interface

The data transfer timing specifications are non-restrictive. Any I/O device with or without handshaking capabilities may be used, since communication device-specific

circuitry does not exist in the PSR. The Motorola MC68230 PI/T implements an asynchronous 2-wire handshaking protocol which the device accommodates. Figure 3.3.5.2.1 shows how the MC68230 PI/T may be connected to a network of PSRs.

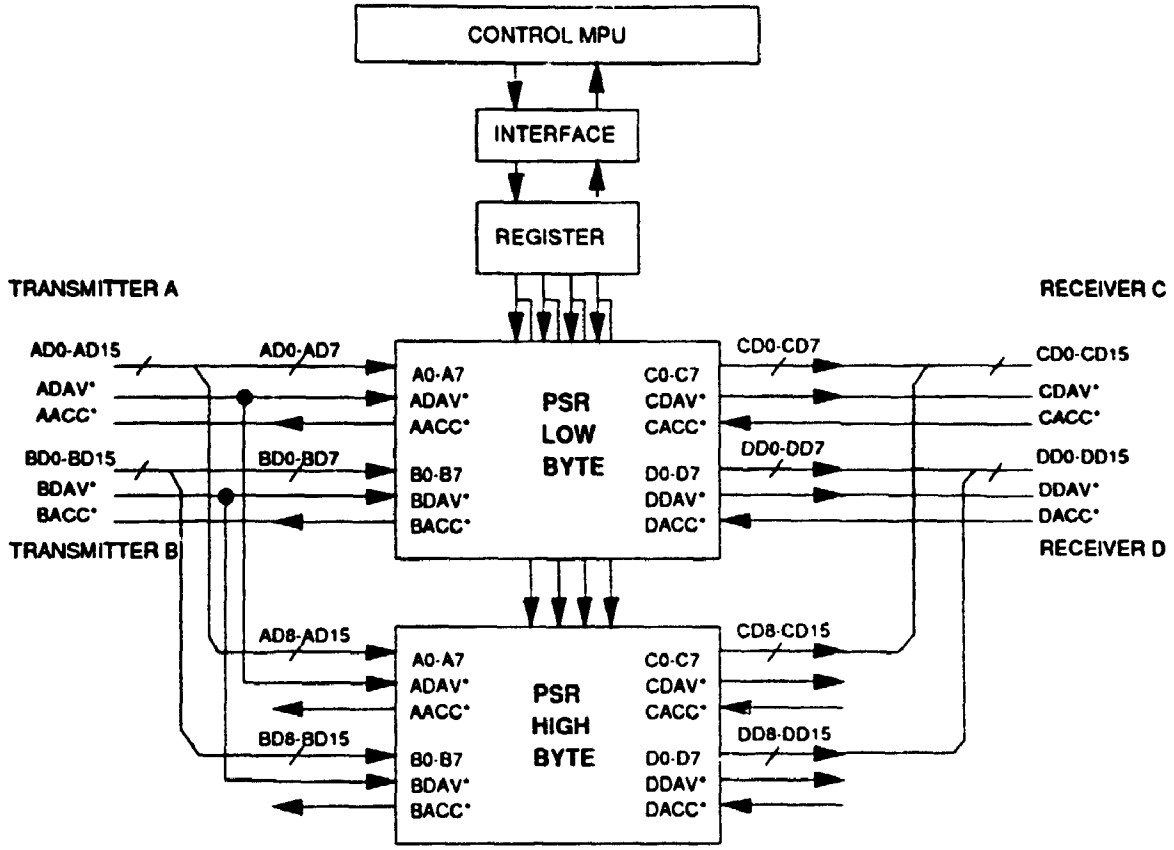
Figure 3.3.5.2.1 MC68230 PI/T connections to a network of PSRs.



3.3.5.3 Data Path-Width Expansion

The PSR is an 8-bit device, but may be used in 16-, 32-, and other multi-bit systems by incorporating additional devices connected in parallel. Since only one DAV* and one ACC* line per port is necessary, no supplemental circuitry is necessary. All of the DAV* input signals must be connected, since they are used to latch data internal to the device. Figure 3.3.5.3.1 shows a typical connection for 16-bit systems.

Figure 3.3.5.3.1 Typical connection for a 16-bit system.

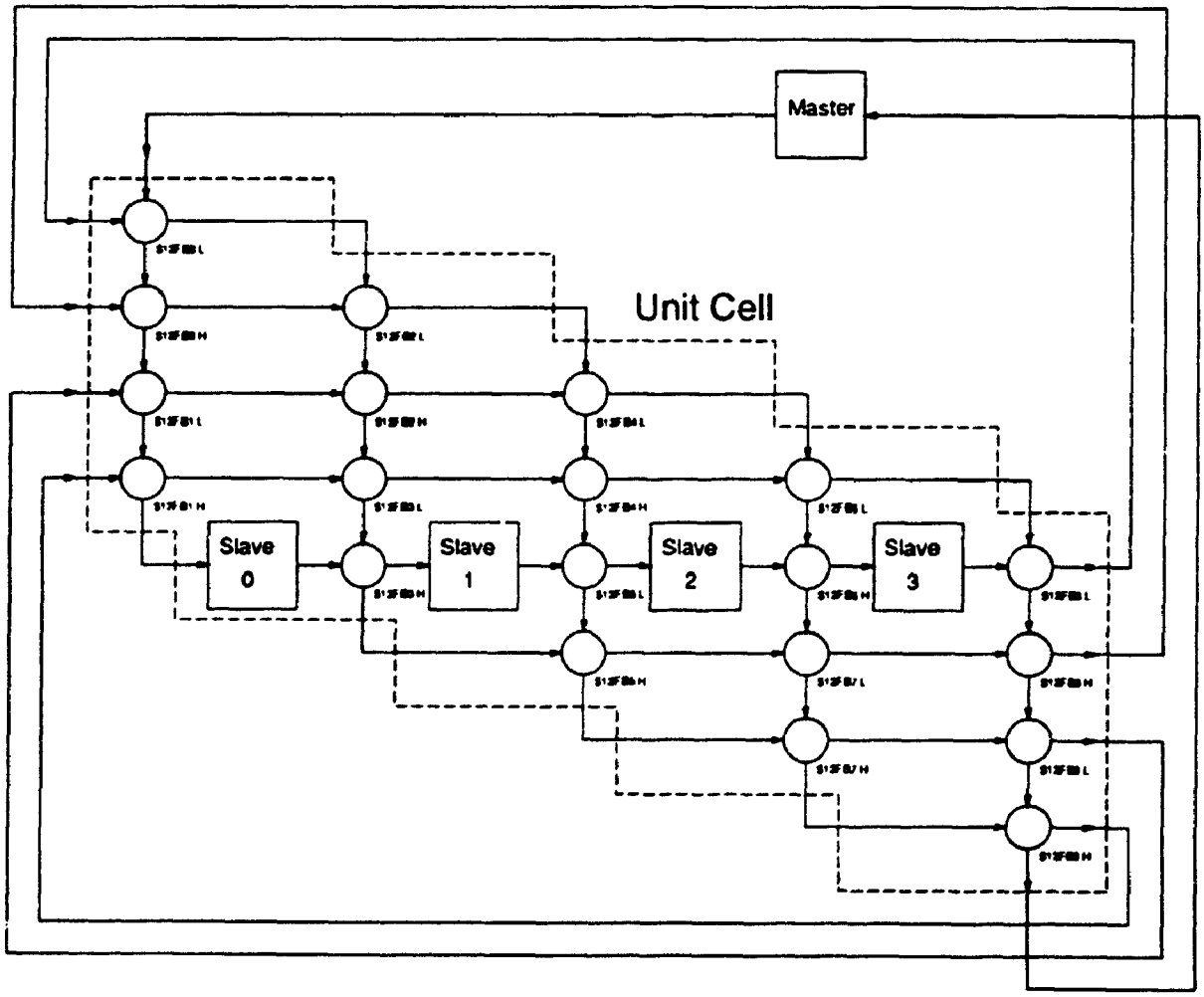


3.4 Interprocessor Communication Network

The interprocessor communication network (ICN) was designed to fulfill three major objectives: conflict-free communications among slaves in a 4-processor cell; physical geometric regularity of the cell; and external access to all slaves in the cell. Communications between slaves and the master processor utilize the network, as well. Slave processors possess a single input data port, and a single output data port, which are realized by the PI/T ports A and B, respectively. The network topology, which has been implemented for the study, is an offset ladder-like array of twenty PSRs. A block diagram of the prototype system's ICN is shown in figure 3.4.1. A schematic diagram of the ICN is shown in Appendix F. The PSR's are shown as circles, processor cells as rectangles, and data paths

as arrows, indicating their fixed direction. The hexadecimal addresses shown adjacent to the PSP's correspond to the master processor's memory addresses to which the devices are mapped; "L" indicates the lower order 4-bits of data at that byte address, and "H" indicates the most significant 4-bits. Data transfers across the ICN between processors obey the protocol indicated in figure 3.3.2.4.

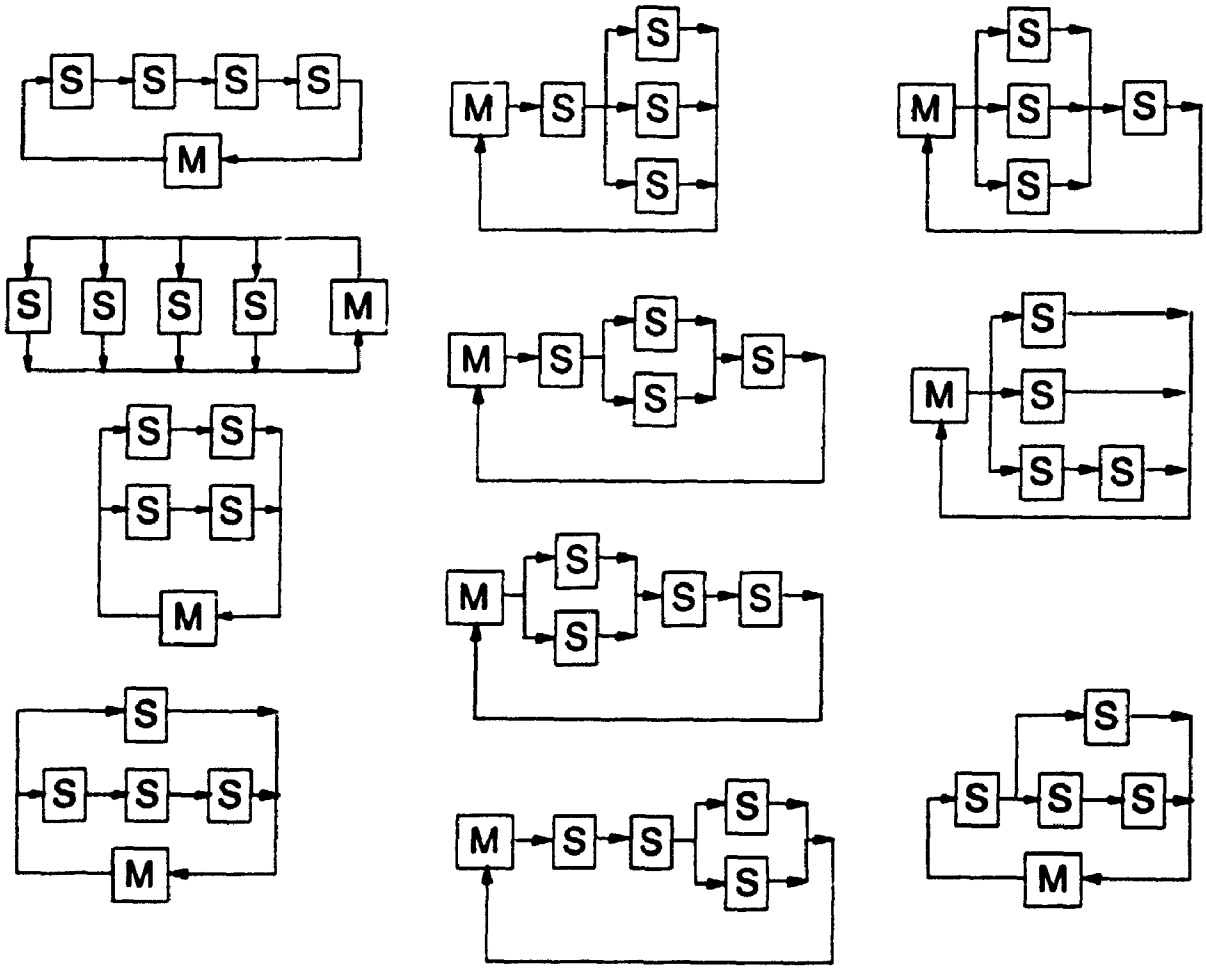
Figure 3.4.1 R· .ram of prototype system Interprocessor Communication Ne



The elements shown inside the dotted lines in figure 3.4.1 constitute the 4-slave cell. Data path arrows crossing the line are considered extra-cellular connections. Expansion of the system is facilitated by replicating the four-slave cell and interconnecting them at the extra-cellular connection points. The choice of connection sites has an influence on the simultaneous configurations which are possible in expanded systems. A full analysis of possible configurations for expanded systems, and their relation to the selection of connection sites is beyond the scope of the presented thesis; Snyder (Snyder, 82) theorizes that the number of configurations possible is related to the switch corridor width, and thus full reconfigurability of expanded systems can only be achieved by increasing the corridor width. Clearly, this is not possible with a fixed, cellular design, and some compromise must be settled upon. The network has distinct input and output sections, necessitated by the unidirectionality of the communication paths. The path from a given output to an input appears as a multi-branched, expanding and collapsing tree; branching is determined by PSR settings. Figure 3.4.1 shows the extra-cellular connections used in the prototype, unit-cell system. In the prototype system, the extra-cellular connections are such that any slave may output data to any of the five outputs at the lower-right corner of the figure, without limiting communication by other slaves to the remainder of the outputs. Likewise, any input signal at the upper left of the diagram can be routed to any of the slaves without conflict. Figure 3.4.2 shows a number of possible processor configurations; the slave indices are not distinguished in the figure since any slave can assume any position for a given topology. A rigorous, mathematical proof of the interconnection patterns available is beyond the scope of the presented thesis; the 4-processor combinations have been individually and explicitly verified. Figure 3.4.3 shows an eight-processor, two-cell system organization, with cells connected "end-to-end". Note that only a single master is used in systems of any size. When a 4-slave cell is considered, any slave transmitting has a choice of three remaining slaves for reception, and thus only three output signals must be routed back to the input

section of the ICN. What results is a four-slave cell with two input and two output channels, resembling the PSR device organization itself. The strategy has reduced redundancy, however it provides the same set of interconnection patterns for a 4-slave system (some restrictions on slave positions within a topology may apply). Thus, expanded systems may utilize such 2-input, 2-output cells, structured in a rectangular array. Due to the limited connections between cells, the number of possible configurations will be reduced; however, the structure of the system would still be suitable for a wide range of problems, and would resemble an array processor with locally reconfigurable node elements.

Figure 3.4.2 Available processor configurations.



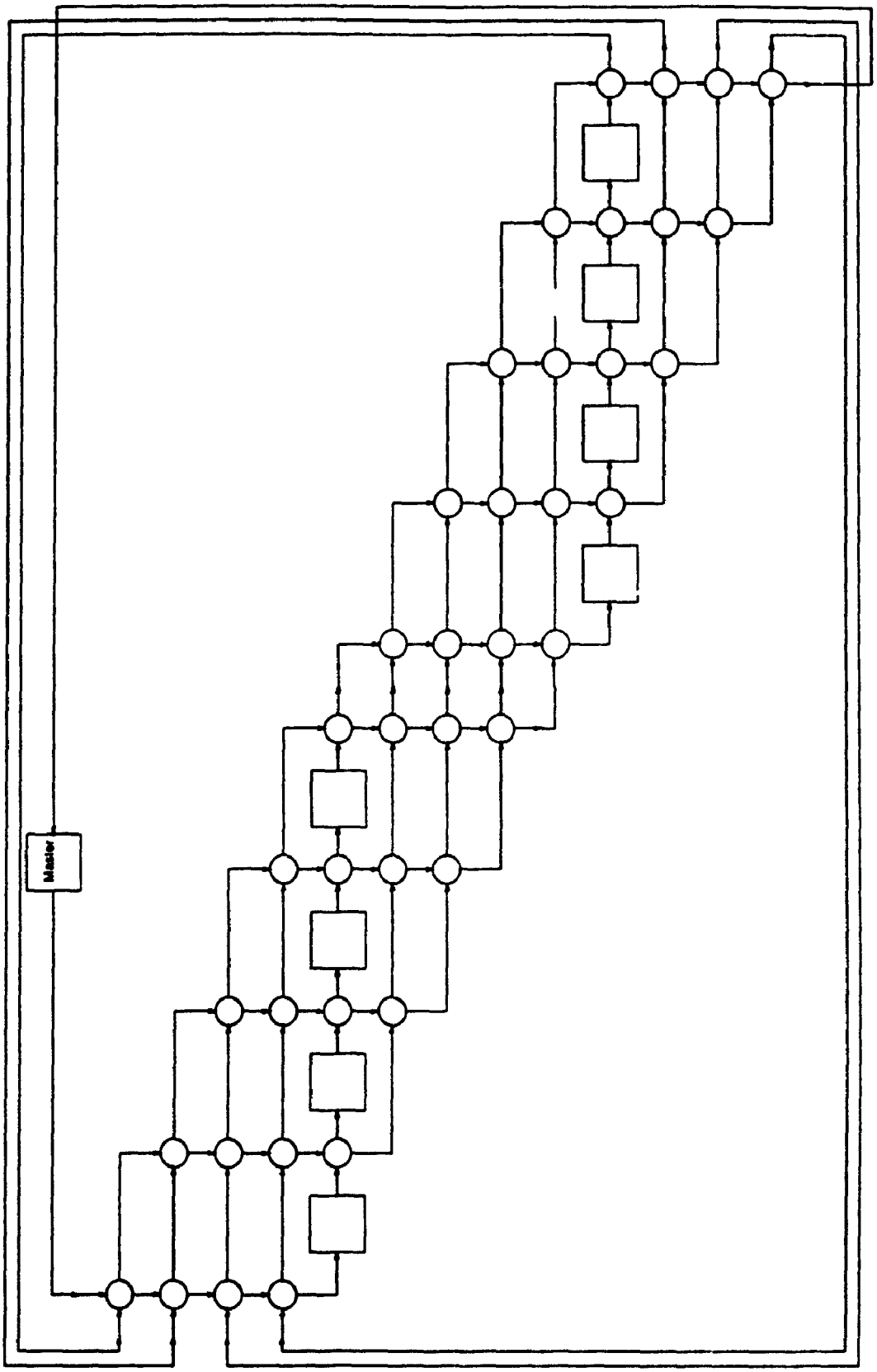


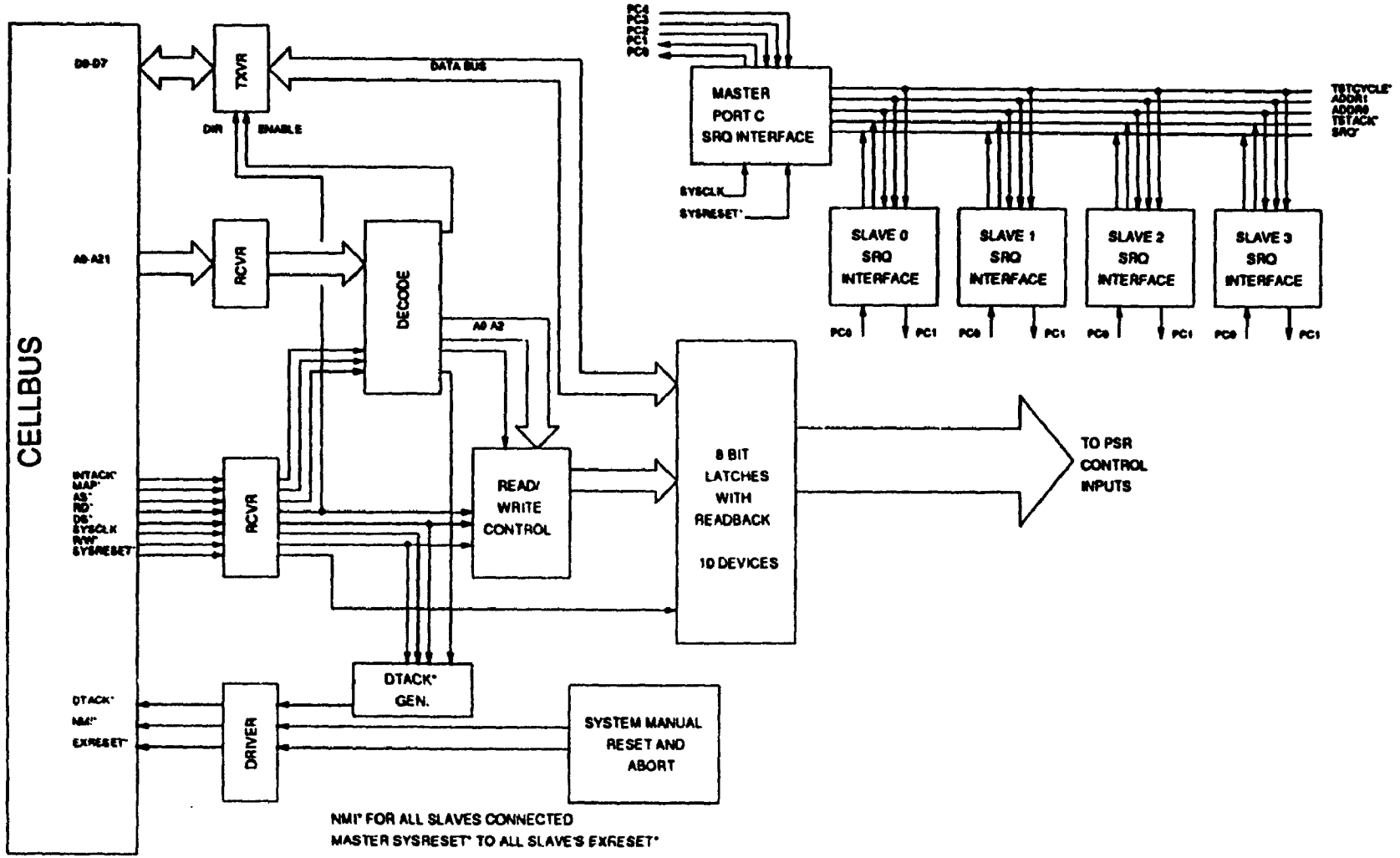
Figure 3.4.3 Expanded system with eight slave processors and master.

3.5 Network Controller Card

The network controller card (NCC) interfaces the master processor with the configuration control signals of the twenty PSRs in the ICN. The design is based on that presented in section 3.3.5.1. Ten 8-bit latches with read-back capability (74ALS666) are used as a register block for control of the network configuration. The registers occupy addresses $12FB0 to $12FB9 in the master's memory map. Their contents are continuously present on the latch output signals which are connected to the configuration control signals of the ICN. Since only four bits are required to control a PSR, a single latch controls two PSRs and occupies only a single memory address. The read-back ability of the latches is not entirely necessary, since the master could trace the history of register write operations. Incorporating read-back makes tracing non-compulsory. More importantly, the registers exist as read/write memory locations; programming flexibility is enhanced since all available instructions and memory addressing modes may be used to modify the register contents. From both hardware and software perspectives, the registers are treated as fast memory locations, and performance and flexibility is maximized.

The NCC also contains the necessary circuitry to implement the Service Request bus (SRQbus) interfaces. Alternatively, each slave processor could incorporate its SRQbus interface directly as part of its Memory and PI/T card, and a simple bus-backplane connection could be made among processors. The strategy causes slave cells to become non-identical by virtue of their SRQbus address decoder. It compromises the cellular nature of the slaves, and topologically constrains their design. Therefore, the SRQbus interfaces were located on the network controller card. The next section discusses the SRQbus and its operation in detail. Schematic diagrams of the NCC and SRQbus interfaces are shown in Appendix G.

Figure 3.5.1 Block diagram of Network Controller Card and SRQbus Interfaces.



3.6 Service Request Bus

The SRQbus provides the means whereby slave processors signal status information to the master. The information is used by the master to determine when a network reconfiguration may safely be performed. Modifications to logical interconnections must not occur while processing cells are using the path to exchange data. Since the master communicates with the slaves using the ICN as well, the very act of polling for status information via the ICN requires potentially hazardous reconfigurations. The SRQbus is a necessary element in the asynchronous system design. The SRQbus does not exchange data between system units; it implements a streamlined "ready/continue" signalling system. Slaves cannot determine the status of other slaves using the bus, information flow is limited to individual slave-master transactions. Operation of the bus is described below.

A slave interfaces to the SRQbus using its PI/T port C data port. Bit PC0 is the SRQSTROBE* output, and bit PC1 is the SRQACK input. A slave signals the master by momentarily asserting SRQSTROBE* low. A flip-flop in the SRQbus interface is toggled, asserting the SRQbus SRQ* signal continuously, as well as resetting SRQACK low. The slave monitors its SRQACK signal, until it is asserted (high) when the master acknowledges the request. The SRQ* signal (master bit PC0) is driven by all slave interfaces using open-collector devices; when the SRQ* signal is asserted, the master must resolve which slave has asserted the request. The master places an address on the SRQbus using its PC2 and PC3 outputs, and an "address valid" (TSTCYCLE*; bit PC4 output) signal is asserted. Each slave occupies a unique address on the SRQbus. If the currently addressed slave has requested service, that slave's SRQbus interface will assert TSTACK* (master bit PC1 input, also driven by all slave interfaces using open-collector devices). The master reads the state of TSTACK*, and negates TSTCYCLE*. This in turn toggles the addressed slave's SRQACK high. The slave recognizes that its request has been acknowledged by the master, and it returns to executing its program. Meanwhile, the master has recognized which slave

had asserted the request, and it updates its internal assessment of the application program's degree of completion. Timing diagrams showing typical operation of the SRQbus under single- and multiple-pending requests are shown in figure 3.6.1 and figure 3.6.2, respectively. The SRQbus interfaces are designed such that if a request is made by a slave during a period when the master happens to be polling that slave (TSTCYCLE* is active), the request is not acknowledged until the next polling cycle.

Consistent control must be exercised due to the streamlined nature of the signalling. A slave does not report why, how often, nor for how long it has been requesting service. A well-defined operating strategy for process synchronization must be adopted which necessarily couples both the hardware and software control of the system. A discussion of process synchronization and programming conventions employed in the system is deferred to a later section (section 3.8), after a description of the system software has been presented.

Figure 3.6.1 Timing diagram of SRQbus Protocol: single requestor.

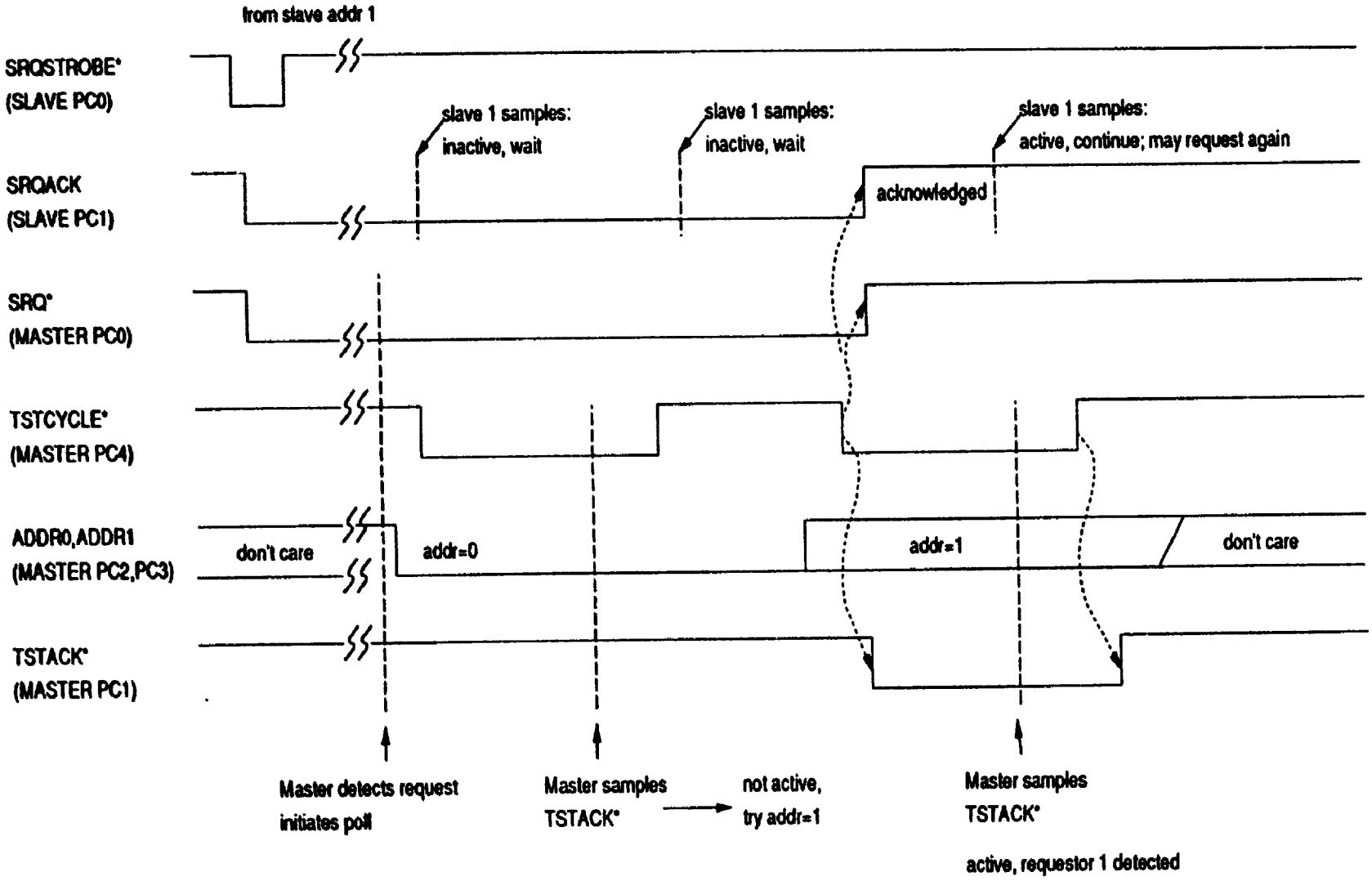
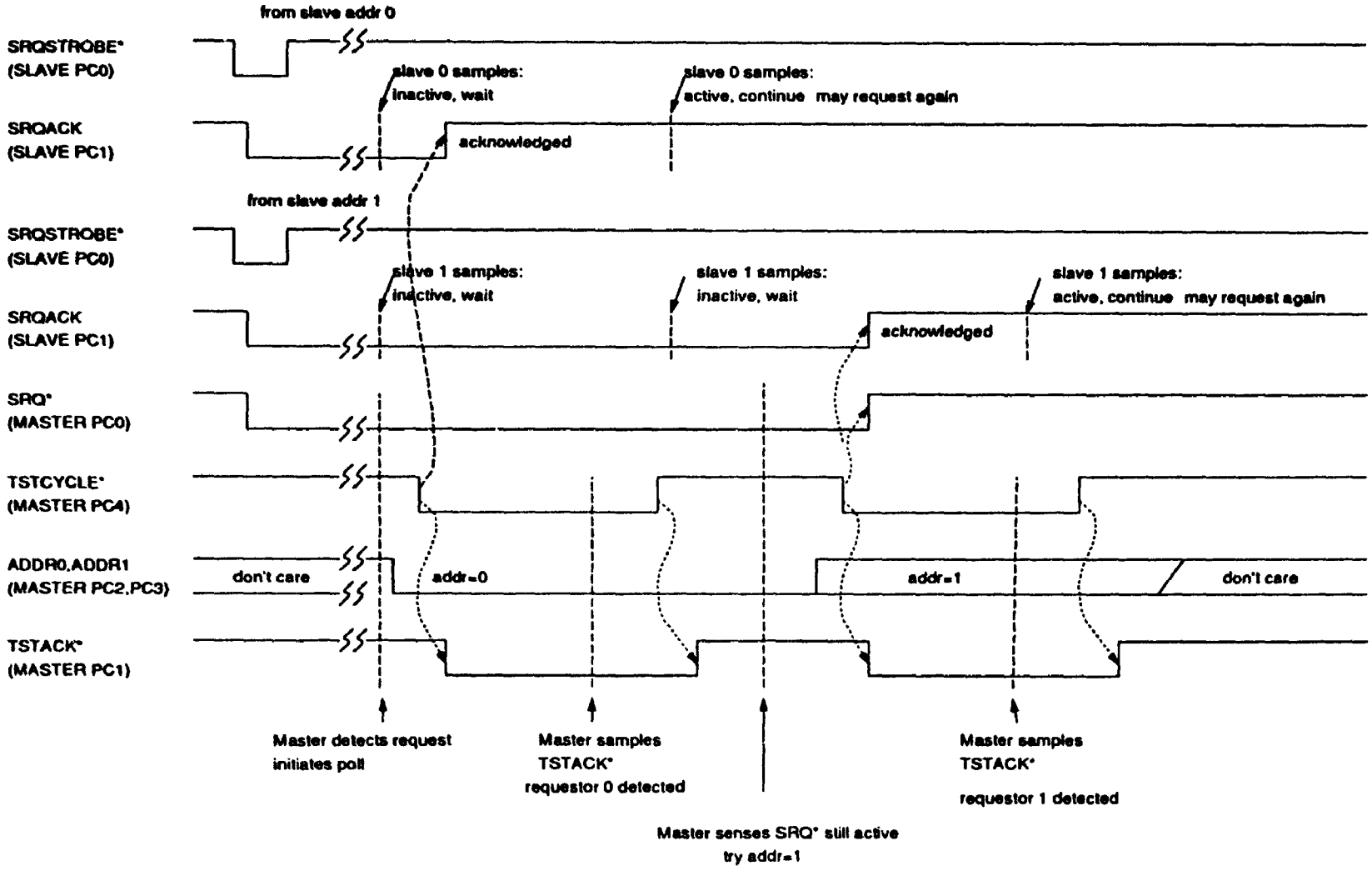


Figure 3.6.2 Timing diagram of SRQbus Protocol: multiple requestors



3.7 System Software for the Reconfigurable Multicomputer

This section focusses on the system-level programs developed for the RMCS as part of the thesis. The programs not only provide system control and supervisory functions, but make available a number of service routines via TRAP exception vectors, which may be utilized to simplify applications programming. Due to the unique nature of the RMCS, a commercially available operating system was not used. Such systems can place unsuitable and unnecessary restrictions on system design and control (memory requirements, system routine linkage, programming languages), and apply principles based on existing architectures and paradigms. Additionally, such packages invariably offer supplemental utilities unsuitable to the objectives of the prototype system; those which are useful often require slight modifications to adapt them to a unique system like the RMCS. The software for the RMCS is therefore optimized for the given system, and includes only those utilities and subroutines necessary to achieving the goals of the presented thesis.

3.7.1 General Description

The system software for the RMCS encompasses both the master and slave processors' ROM-based programs. Since the master processor is used for overall system control, and for providing the user and host interface, its system-level programs incorporate routines for I/O, exception handling, and ICN control. The master processor's monitor provides a versatile program development and debugging environment for the RMCS. Since the slave processor cells lack a user terminal or host interface, their system software is comparatively streamlined. The slave software does not incorporate error-condition exception processing (such as bus-error, address error, divide by zero, etc.). Slave tasks must execute without encountering those errors, unless custom exception handling routines are specified. Slave routines may be tested and debugged using the master processor during program development.

Overall system operation can be described as a sequence of *processing states*, to which all of the slave processors adhere. Processing state information is passed between slaves and the master using the SRQbus. The formalization is limited to the most basic processing phases of execution which are common to all programs. Individual programs require various degrees of interprocessor synchronization throughout their execution, and supplementary synchronization cycles are specified as part of the application program. The master processor monitors slave processing states throughout program execution, and thus all programs require code for both master and slaves.

The master processor's monitor program provides the RMCS with the necessary system initialization instructions, exception processing routines, and a comprehensive user interface. The combination of user commands, error-condition exception routines, and informative error messages, provides a powerful, straightforward program evaluation and debugging environment.

Programs executing on the master processor are permitted access to a subset of the monitor's serial input/output and code conversion routines in an address independent manner via the TRAP #15 service. The feature simplifies and shortens user programs, eliminating the need for I/O device drivers and code conversion subroutines to be included in each program. A series of TRAP utilities are included for the slave processors as well. Functions which facilitate block transfers of program code and data, address and byte count specifications, and user/supervisor state switching, are common to the master and slave utility-program repertoires. Slave processors have additional TRAP routines to request service from the master and await acknowledgement. Both hardware- and software-controlled means exist whereby processing states may be changed. The master processor has unique TRAP service routines for ICN configuration control, as well as detection and acknowledgement of slave service requests.

The master monitor program executes in the MC68008 supervisor state at all times. The user state is entered upon a GO command (to execute a user program) being issued. A safe transition back to the supervisor state occurs upon execution of the TRAP #0 instruction, which must be the last in a program. The master monitor program regains control, and further commands are accepted.

The starting addresses of the vectored exception service routines are copied to the exception vector table in RAM after a system power-on or hardware reset condition. The contents of the vector table may be modified directly, enabling the user to define and relocate custom exception service routines, *in lieu* of the existing routines in the monitor ROM. Since the ROM devices are significantly slower than the system RAM, routines which are preferred to execute at maximum speed are copied to RAM upon system restart. They may be accessed using the appropriate TRAP instructions (section 3.7.4). The feature is included to increase overall performance while maintaining ease of programming for the user. It is recognized that higher performance may be attained by executing the system utility routines as in-line code within an application, thereby eliminating the TRAP/RTE (call and return) instruction overhead. Under such conditions, however, code length is significantly increased, and a compromise in performance may be made in exchange for memory usage efficiency and programming ease.

3.7.2 System Requirements

The RESET vector consists of the initial program counter and the initial supervisor stack pointer. The RESET vector occupies the first eight bytes of the respective ROMs, and these locations are accessed during the first eight bus cycles after a power-on or hardware reset condition. The contents of the vector specify the absolute addresses of the initial system stack pointer (INITSSP) and the initial program counter (RESTART), which is the address of the first instruction of the monitor program.

The exception vector tables are located at memory addresses $0C through $3FF of each processor in the system, as required by the MC68008. A RAM workpage is required by the master monitor for temporary storage, and to hold copies of performance critical routines. It is located at an offset of -$210 from the first byte of ROM. In the APC's the ROMs are located at $10000 to $11FFF, and thus valid RAM must exist from $00 to $3FF, from $FDF0 to $FFFF, and from $FDF0 down to accommodate at least 128 bytes for the system stack. Slave system programs also use a RAM workpage to hold their copies of performance critical routines. It resides at an offset of -$110 from the first byte of ROM; slave processors begin their supervisor stacks at $FED0.

The master monitor program addresses two MC6850 ACIA peripheral devices; one assigned to the console terminal (at $12FFC and $12FFD), another assigned to a host communication link (at $12FFE and $12FFF). The serial communications protocol is defined as 8-bit, no parity, one stop-bit, clock ÷16, RTS low, receive interrupts disabled. Communication via the ports is non-interlocked, with no XON/XOFF protocol implemented. Both master and slave system routines initialize and access an MC68230 PI/T device, which incorporates three parallel I/O ports and a programmable timer. Port A is configured as an 8-bit input port in the *interlocked input handshake* mode, while port B is an 8-bit output port in the *interlocked output handshake* mode (MC68230, 83). Port C of the master and slaves are configured differently, since the slaves use the port to assert SRQSTROBE*, and to detect a SRQACK, while the master uses its port C to sense and acknowledge slave SRQs. The slave processors configure PC0 as an output, with PC1-PC7 defined as inputs. The master defines PC0, PC1 and PC5-PC7 as inputs; PC2-PC4 are outputs. The timer section of the PI/T is not addressed by the system code, since its configuration after RESET is passive. Timer programming and utilization are left as application program-specific functions.

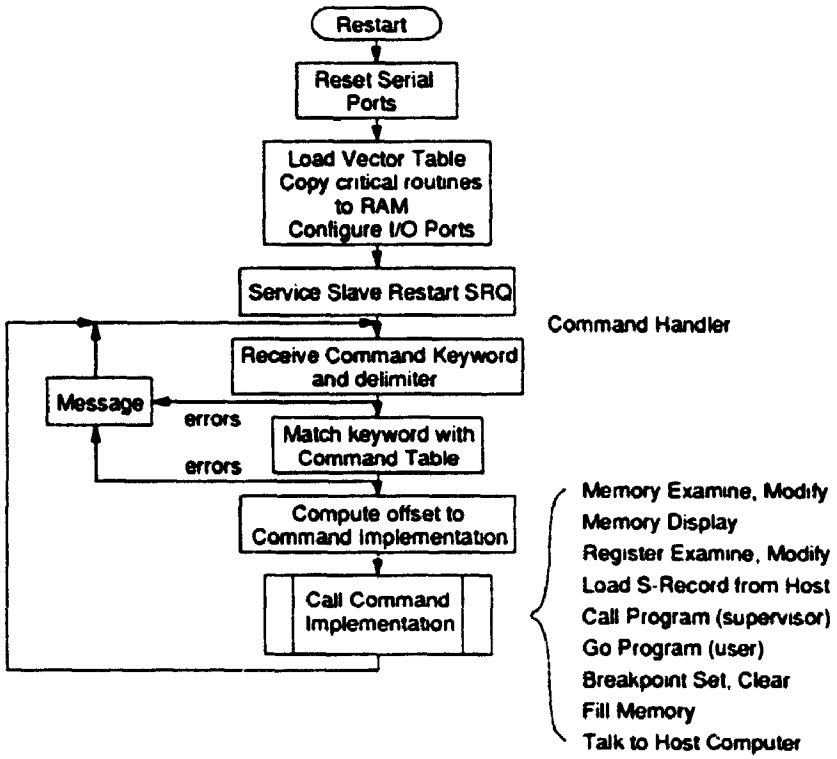
3.7.3 System Program Execution

This section describes master monitor program execution and content, and the slave system program execution.

3.7.3.1 Master Monitor Program Initialization

Execution of the master monitor program begins after a power-on or hardware reset condition, and the MPU's RESET vector fetch sequence is invoked. The addresses of the first instructions of the exception service routines are loaded into the exception vector table in RAM ($00-$3FF). The serial and parallel communication ports are configured, and initial values in the RAM workpage established. The MPU's status register is written to establish tracing off, supervisor state processing, and condition codes clear. Performance-critical routines are copied to RAM, and the first *service request* (RESET SRQ) signals from the slaves are all acknowledged. The command handler is entered upon completion of the initialization phase. The RMCS ICN configuration control registers are not explicitly written: they are automatically cleared by the hardware RESET, and therefore all processors in the system are isolated. A flowchart depicting master monitor program initialization is in figure 3.7.3.1.1.

Figure 3.7.3.1.1 Flowchart of Master Monitor program initialization.



3.7.3.2 Master Command Handler

The system prompt ">" is sent to the console terminal port upon entry to the command handler routine, to indicate that a user command is expected, and the system waits for it. Characters entered via the terminal keyboard are stored in the workpage at CMDSTK until a non-alphanumeric character is received. Valid command delimiters are **space** and **carriage return.** An invalid delimiter input results in an error message being issued, and the command handler is restarted.

Upon receipt of a valid delimiter, the input character sequence is compared with the command table in ROM. Command strings are one to three characters in length; extra characters are ignored. Upon recognition of the input command, the address offset to the command implementation routine is computed, and execution resumes as a subroutine call

to that location. Command implementations terminate with the RTS instruction to return control to the command handler, or, in the case of a system error, to the supervisor stack initialization sequence. The command handler ultimately regains control. The contents of internal registers are modified by a command call, with the exception of "**Register display/modify**", which necessarily does not affect register contents unless the user explicitly wishes to do so.

3.7.3.3 Master Command Implementations

The command implementation routines are called upon receipt of the command key-word string and a valid delimiter. If further input parameters are required, it is the first responsibility of the command implementation routine to acquire such data, and to respond to, and recover from, user-input errors. When the necessary input has been received, with appropriate delimiters, the command function is performed.

The available commands, command syntax, and their function, are listed in Appendix H.

3.7.3.4 Master Serial Data Input, Output, and Code Conversion Routines

A set of subroutines for I/O and code conversion are called by most of the higher level monitor routines. In most cases, the I/O functions modify only registers D0 and A0, but exceptions exist. For functions in which register contents are volatile, vital data must be preserved prior to invoking the routine.

Most of the I/O and code conversion routines can be accessed in an address independent manner via the TRAP #15, <utility #> instruction sequence. A detailed discussion of the TRAP #15 service, the available routines, parameter passing conventions, and register usage, can be found in Appendix H.

3.7.3.5 Error Handling and System Support

The MC68008 incorporates a variety of exception vectors that allow vectored "exception handler" service routines to execute following master processor error conditions, interrupts, instruction traps, or to facilitate software tracing.

The most serious exceptions are group 0 exceptions (RESET, address error, bus error), and as such, they precipitate a processing environment storage procedure that differs from those which occur following group 1 and 2 exceptions. A detailed discussion of group 0, 1, and 2 exceptions can be found in the MC68008 documentation (MC68008, 85).

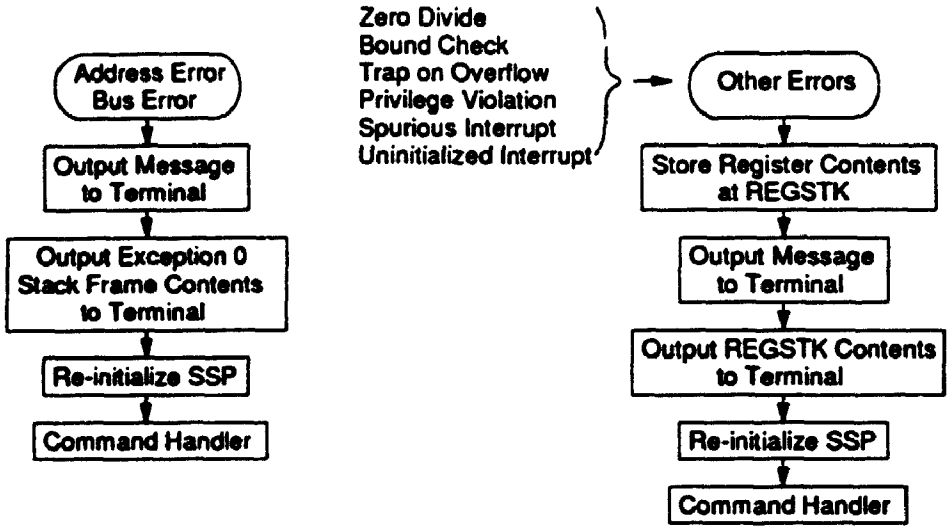
The RESET operation is well understood from the previous discussion (section 3.7.3.1), causing an entire system restart and re-initialization.

After a bus error or address error occurs, the exception routine sends a message to the console describing the error, and the contents of the group 0 exception stack frame are displayed, with informative annotation. The system is re-initialized, and the command handler re-entered.

Exceptions such as trap on overflow, boundary check, zero divide, privilege violation, spurious interrupt, and uninitialized interrupt are all handled in a similar manner. The register contents at the time of the exception are displayed on the console terminal, along with a message informing the user of the fault encountered. The supervisor stack pointer is re-initialized, and the command handler re-entered. Figure 3.7.3.5.1 shows flowcharts depicting error condition processing.

The MC68008 implements single-instruction tracing using a status register bit ("T") which, when set, causes a vectored service routine to be invoked following execution of each instruction in a program. The RMCS utilizes the trace option for single instruction, as well as selected *breakpoint* tracing. Breakpoint tracing also utilizes the ILLEGAL

Figure 3.7.3.5.1 Flowcharts of error condition processing.



instruction, and therefore the service routines for the Trace option, ILLEGAL instructions, and the Breakpoint Set/Remove command must be mutually compatible. The functions and their cooperation are described next.

A user enables the Trace option when calling a program (see Appendix H). Following execution of each instruction, the current register contents are displayed, and the user is prompted to type "Esc" to terminate the program, or any other key to continue execution. Entering "Esc" re-initializes the supervisor stack pointer, and the command handler resumes control. The routine checks the breakpoint flag and counter, since it also executes to service breakpoints defined by the user.

The ILLEGAL instruction exception routine executes when an illegal instruction op-code ($4AFA, $4AFB, $4AFC) is encountered. The breakpoint function of the system monitor uses the ILLEGAL op-code to cause breakpoint exceptions, along with a flag in the RAM workpage.

If the ILLEGAL instruction op-code is encountered, and no breakpoints are set (BRKNUM in the workpage is checked), it is treated as an ILLEGAL instruction, registers are displayed, along with an advisory message, the supervisor stack pointer is re-initialized, and the command handler re-entered.

If breakpoints are set, the ILLEGAL instruction exception routine searches the breakpoint address/data table (BRKSTK in the workpage), and replaces all of the original instructions. The original instructions were substituted with ILLEGAL op-codes and stacked by the breakpoint set/remove command implementation. A flag in the workpage (BRKFLG) is set, tracing is enabled, and an RTE instruction resumes program execution with the original instruction.

The original instruction is executed with tracing active, and thus the trace exception routine is invoked. The trace routine checks BRKFLG in the workpage, determining whether a normal trace or a breakpoint service is requested. The registers are displayed as usual and the user is prompted. If BRKFLG is set, the instructions swapped in by the ILLEGAL instruction exception handler are re-swapped out and replaced by the ILLEGAL instructions. Tracing is disabled, and an RTE executes, returning control back to the original program. Subsequent breakpoint ILLEGAL instructions are handled in the same manner. Figures 3.7.3.5.2 through 3.7.3.5.4 are flowcharts depicting Trace exception, ILLEGAL instruction exception, and Breakpoint Set/Remove command processing.

Figure 3.7.3.5.2 Flowchart of Trace exception processing.

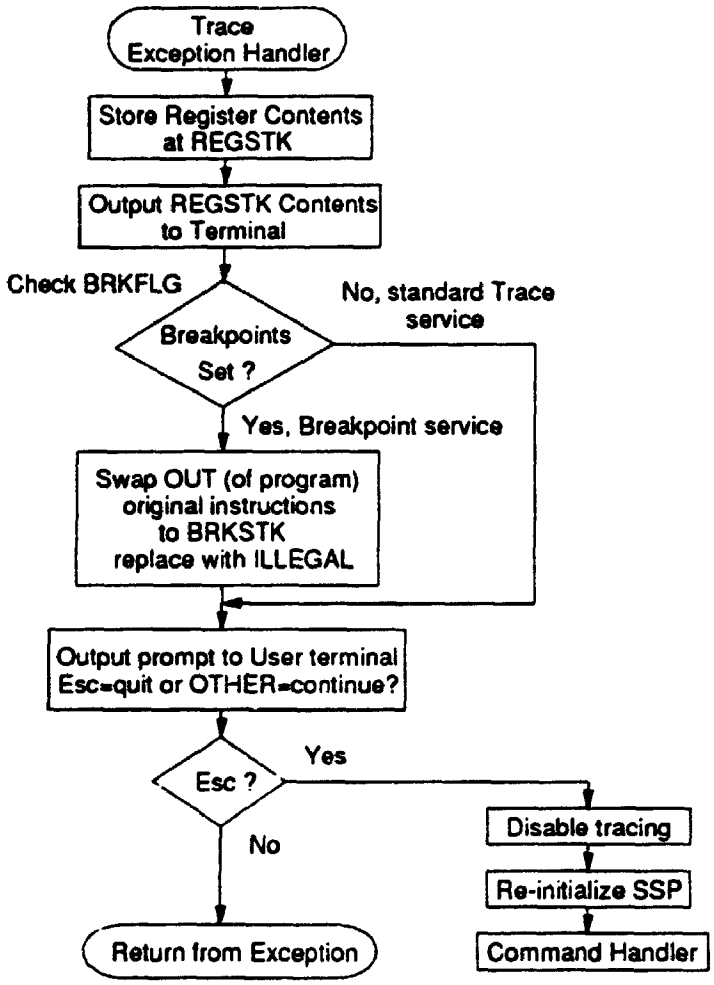


Figure 3.7.3.5.3 Flowchart of ILLEGAL instruction exception processing.

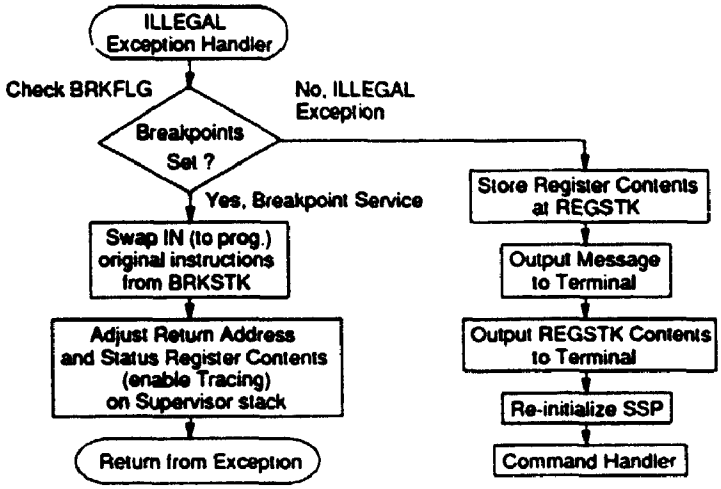
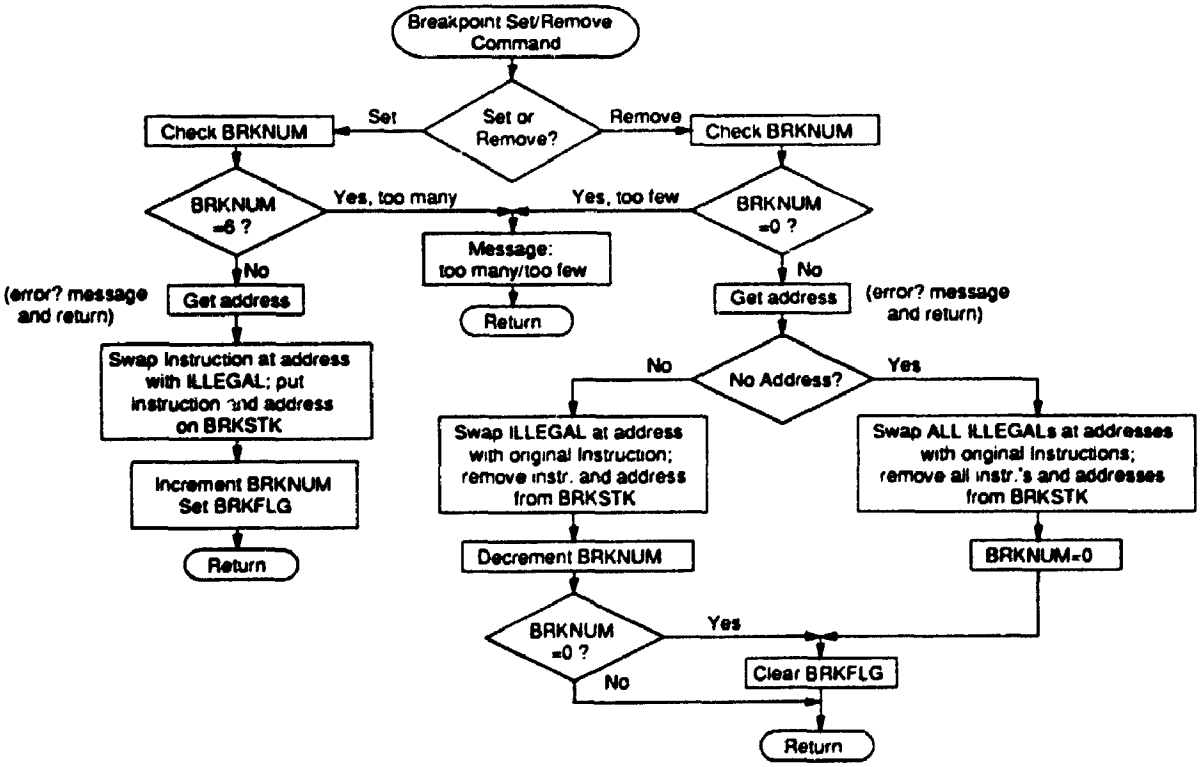


Figure 3.7.3.5.4 Flowchart of Breakpoint Set/Remove command processing.



A number of TRAP instructions are interpreted by the system to allow position independent access to a subset of routines in the master monitor ROM. The routine functions, input/output parameters, and calling statements are discussed in section 3.7.4.

3.7.3.6 Slave System Program

The slave processors in the system operate autonomously, executing the programs present in their respective local memories. The actual programs and data, however, are provided by the master processor. The slaves inform the master of their processing state by asserting the "service request" (SRQ*) signal, and await acknowledgement from the master processor (SRQACK) before entering the next processing state. The processing state transitions follow a prescribed order, of which the master processor keeps track throughout. There are various methods, both hardware and software, with which the master can force the slave's processing state to change, re-initializing the sequence as new programs and data become available.

After a power-on or hardware RESET signal has been received, the slave processor enters the *restart* processing state, which initializes PI/T ports in the same modes as those described in section 3.7.2. The performance-critical routines are copied to RAM, and the slave asserts its first SRQ (see section 3.6), and awaits the master's acknowledgement. After the SRQACK is received, the slave enters the *program accept* processing state.

The *program accept* state begins with an SRQ assertion, and acknowledgment is awaited. Once received, the slave expects six bytes of data to appear sequentially at its input port A, the first four bytes specify the program start address (Most Significant Byte received first), the last two bytes indicate the total block length to follow (MSB first). Data subsequently received are treated as the slave's program code, and it is stored at the specified start address in the slave's RAM. After the last byte is received, the slave enters the *data accept* processing state.

The *data accept* processing state transition is indicated to the master by another SRQ. Once acknowledgement is received, the slave once again expects an address and block length specifier, followed by the actual data for the program. If the program requires no input data, $0000 is specified in the block length specifier, with any valid address specifier (0-$FFFF). The *execute* state is then entered.

As before, the transition to the new state is signified to the master by a service request, and acknowledgement is awaited. The slave proceeds to execute the newly loaded program at the program start address. Execution of the program may involve many more SRQ/SRQACK transactions, and data exchanges with other processing elements. Transmissions and receptions need not always be preceded by addresses and byte counts; they are specified if necessary for the problem at hand. Any supplementary SRQ/SRQACK cycles (those above and beyond those explicitly addressed by the master system software), must be serviced by the master's control routine for the particular application program. Upon completion of the slave program (with an RTS instruction), the slave processor immediately re-enters the *data accept* state. The loop of *data accept* and *execute* is continuously repeated, since it is usual that a program is executed numerous times with various input data; reloading of the program code each time is unnecessary and undesirable.

There are methods for breaking the *data accept-execute* processing-state loop. The obvious one is an entire system RESET, which re-initializes all processors and peripherals, and causes the *restart* processing state to be entered by the slaves. The method is not recommended, however, and should only be used to recover from catastrophic system errors. Resetting the system causes the ICN configuration control registers to clear, which may not be desired. Use of the system ABORT switch causes the master to report its current register contents to the terminal, and forces all slave processors to enter the *program accept* state without re-initializing the peripherals; they are momentarily disabled and their status registers are reset by software. Two software methods exist which force selected slaves to

re-enter the *program accept* state. The first one provides an illegal start address ($FFFFXXXX) to a slave which has reached the *data accept* state, followed by any byte count. Illegal addresses are detected by the slave program, and cause the slave to re-enter the *program accept* state. Once the state is entered, an SRQ/SRQACK cycle is necessary before the new program code can be transmitted. Another method uses the TRAP #14 instruction within the slave program which, when encountered, immediately forces the slave into the *program accept* state. Whenever the *program accept* state is entered, the slave's supervisor stack pointer is reset to the value INITSSP, (its original value after a power-on or hardware RESET), and thus the stack does not grow indefinitely as new programs are loaded. When a program is loaded, the start address of that program is placed on the stack for future use by the *execute* state, to recall the original program start address. Since the slave application program is executed as a subroutine of the slave system program, the return address to the calling program is also present on the stack. It is imperative that slave programs do not terminate with the supervisor stack pointer content different from that which exists prior to slave program commencement.

A number of TRAP instructions are interpreted by the system to allow position independent access to a subset of routines in the slave system ROM. The routine functions, input/output parameters, and calling statements and conventions are discussed in section 3.7.4.

A simplified representation of slave system program execution, showing processing states and their transitional conditions is shown in figure 3.7.3.6.1. A flowchart of the slave system program is in figure 3.7.3.6.2

Figure 3.7.3.6.1 Representation of Slave Program Execution, showing state transitions.

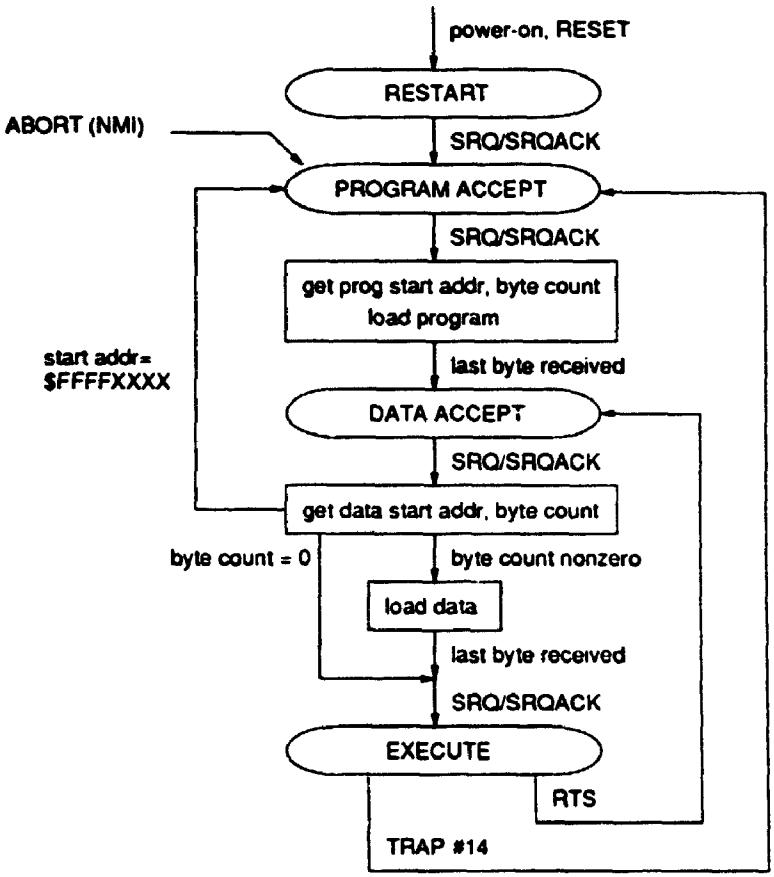
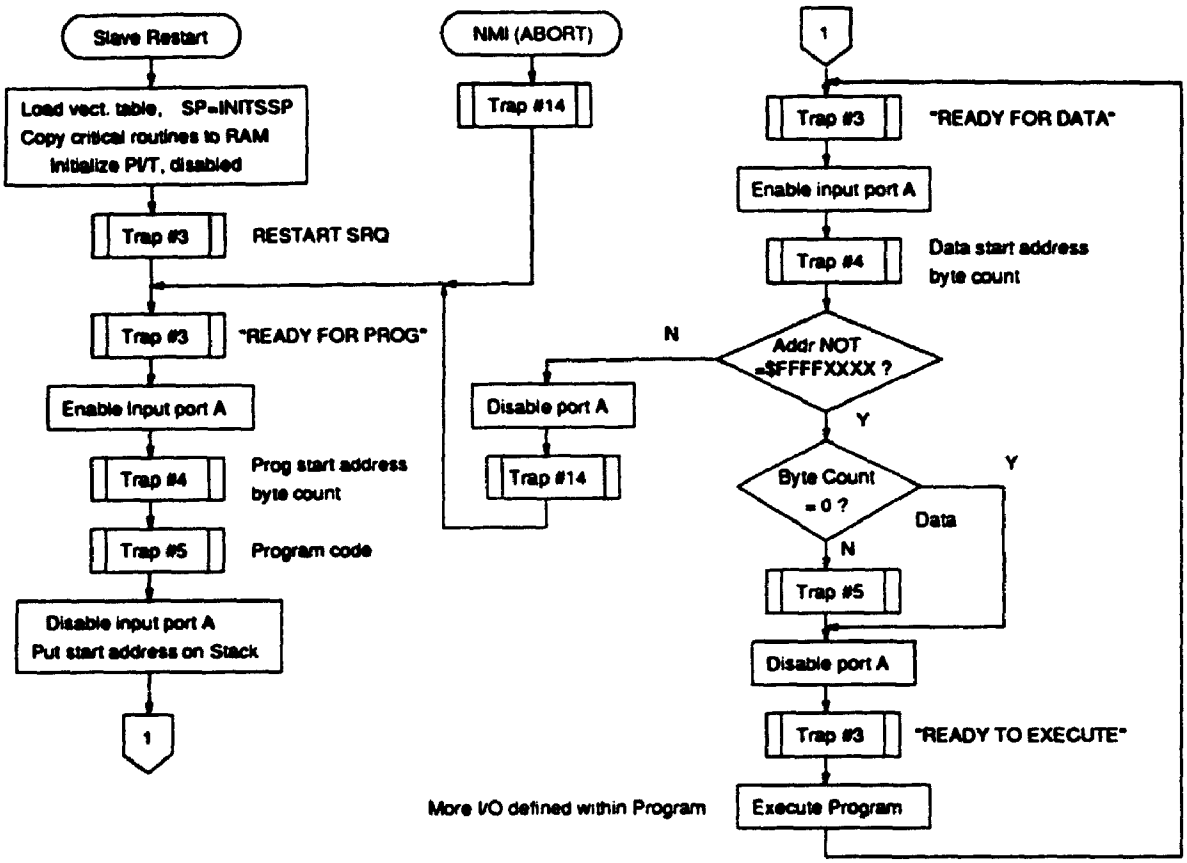


Figure 3.7.3.6.2 Flowchart of Slave System Program.



3.7.4 TRAP Exception Service Routines

This section describes the utility routines available in the RMCS system software. Some routines are available on both master and slave processors; others are processor specific. Distinctions are denoted throughout the section.

3.7.4.1 TRAP #0 Function (Master only)

The TRAP #0 instruction is used as the last instruction of a program executed in the user state, and it forces an orderly return to the supervisor state and monitor program

command handler. The status register contents saved on the stack as a result of the TRAP call are modified such that upon RTE instruction execution, processing resumes in the supervisor state, as required by the monitor program.

3.7.4.2 TRAP #1 and TRAP #2 Service (Master and Slaves)

The TRAP #1 and TRAP #2 services provide a convenient means with which a system programmer may toggle between supervisor and user processing states. The TRAP #1 instruction forces user state processing upon return from the handler routine, while TRAP #2 restores the supervisor state when executed. Valid stack pointers must be maintained throughout, to preserve system integrity. The instruction handlers compromise the privilege protection strategy of the MC68008 architecture and system; use of the instructions should only be undertaken by experienced programmers when specifying system-level or benchmarking code.

3.7.4.3 TRAP #3 Service: SRQASRT (Slaves only)

The TRAP #3 instruction is used by a slave program to cause an SRQ to be asserted by that slave processor, and acknowledgement from the master is awaited. It is used to synchronize activities of the master and slave.

An SRQ is asserted by strobing the PI/T PC0 output low, then high again. PC1 is immediately forced low by the SRQbus interface hardware, and it remains low until the master acknowledges (SRQACK) the requesting slave. After the SRQACK signal is restored high (the slave continuously polls the state of the PC1 signal), the exception subroutine is terminated, and control is returned to the calling program.

3.7.4.4 TRAP #4 Service: ADDRBYT (Master and Slaves)

The TRAP #4 instruction invokes a routine which reads a start address and byte count arriving at input port A. The start address is returned in register A0.L, and the byte count is returned in register D0.W. Other register contents are unaffected. Input port A must be enabled before the TRAP #4 instruction executes.

3.7.4.5 TRAP #5 Service: INDATA (Master and Slaves)

The TRAP #5 instruction invokes a routine which reads a sequential data block of length (in bytes) specified by the contents of D0.W arriving at input port A, and places the data in increasing contiguous memory, starting at the address specified in register A0.L. Register contents are unaffected. Input port A must be enabled before the TRAP #5 instruction executes.

An example program sequence to read a start address, byte count, and data block arriving at port A is as follows:

```
    MOVE.B #ENABLEA,PGCR          ;ENABLE PORT A HANDSHAKE PINS
    TRAP #4                       ;GET START ADDRESS,BYTE COUNT

{validity check address, zerc
check byte count}

    TRAP #5                       ;GET BLOCK DATA
    CLR.B PGCR                    ;DISABLE PORT A HANDSHAKE PINS
```

3.7.4.6 TRAP #6 Service: OUTDATA (Master and Slaves)

The TRAP #6 instruction calls a routine to write a data block of length (in bytes) specified by the contents of D0.W, from contiguous increasing memory addresses starting at the address specified in register A0.L, sequentially to output port B. Register contents are unaffected. Output port B must be enabled before the TRAP #6 instruction executes.

3.7.4.7 TRAP #7 Service: SNADBYT (Master and Slaves)

The TRAP #7 instruction is used to call a routine which sequentially writes six bytes of data to output port B. The first four bytes are the contents of A0.L, and are sent with MSB first. The last two bytes are the contents of D0.W, and are sent MSB first. The output data specify an address and a byte count to the receiving processor. Register contents are unaffected. Output port B must be enabled before the TRAP #7 instruction executes.

An example program sequence to send a start address, byte count, and data block out via port B is as follows:

```
        LEA START(PC),A0          ;BLOCK START ADDRESS
        LEA END1(PC),A1           ;LAST ADDRESS+1
        SUBA.L A0,A1              ;CALCULATE BLOCK LENGTH
        MOVE.W A1,D0              ;LENGTH IN D0.W
        MOVE.B #ENABLEB,PGCR      ;ENABLE B PORT HANDSHAKE PINS
        TRAP #7                   ;SEND ADDRESS, BYTE COUNT
        TRAP #6                   ;SEND DATA BLOCK
        CLR.B PGCR                ;DISABLE B PORT HANDSHAKE PINS
START   DC.B 0,1,2,3,4,5,6,7,8
END1    DS.B 1                    ;DUMMY FOR END1
```

3.7.4.8 TRAP #8 Service: NETCONF (Master only)

Ten bytes of data are written to the network configuration control registers located at memory addresses $12FB0-$12FB9 (inclusive). Register A0 contains the start address of a 10-byte block in memory which contain the desired control words for the new ICN configuration.

3.7.4.9 TRAP #9 Service: SRQACK (Master only)

The TRAP #9 instruction is used to respond to slave SRQs by detecting which slaves are currently asserting the SRQ* signal, and causing the appropriate SRQACK signals to be asserted. A 4-bit *expected SRQ mask* (ESM) is required by the routine in D0.B, with bit b_N set to signify that an SRQ from slave #N is expected, which must be acknowledged before exiting the routine. Only the expected SRQs are acknowledged. A rotating priority scheme is used within the routine. First, the routine waits for the SRQ* signal to go active (low),

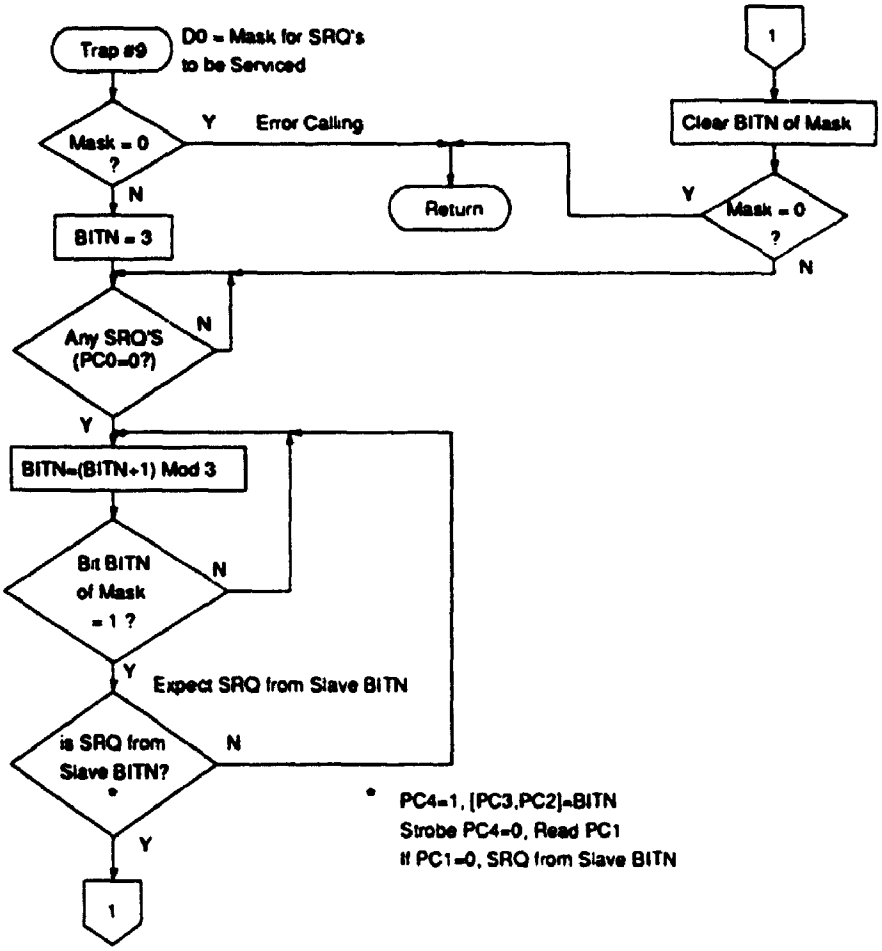
signifying that one or more SRQs are pending. Beginning with slave #0, the ESM is examined to determine whether that slave should be interrogated and acknowledged, or not. If bit #0 of the mask is not set, slave #0 is not polled, and other bits in the mask are examined. If the bit is set, the slave is polled as described in section 3.6. If slave #0 was requesting service, it now has been acknowledged and the bit in the ESM is subsequently cleared. Other bits of the ESM are tested and the corresponding slaves polled in the same manner, proceeding with slave #1, then #2 and #3. The subroutine terminates when the ESM has been cleared.

The possibility exists that a higher-numbered slave may be serviced before a lower-numbered one, even when both are expected, depending on the order and time between requests.

The strategy allows different SRQ priority schemes to be implemented: fixed, rotating or any variable-order. The exact strategy is determined by manipulation of the ESM and by appropriate ordering of calls to the TRAP #9 service routine. First-come, first served strategies are not possible with the TRAP #9 service, when slave-specific intermediate processing is necessary between requests by competing slaves, but such a protocol is easily implemented by directly controlling the master SRQbus I/O port.

A flowchart of the TRAP #9 utility routine is shown in figure 3.7.4.9.1.

Figure 3.7.4.9.1 Flowchart of the TRAP #9 (*SRQACK*) Service Routine.



3.7.4.10 TRAP #14 Service: ABORT (Slaves only)

The TRAP #14 instruction is used within a slave program to cause a transition to the *program accept* state. The supervisor stack pointer is re-initialized to the value INITSSP, ports A and B are disabled, their status reset, and the *program accept* processing state entered, starting with a SRQ/SRQACK cycle, as described above. The TRAP #14 exception vector is also the slave NMI vector; the same code is executed upon assertion of the non-maskable interrupt signal (system abort switch depressed).

3.7.5 Program Listings

Master and slave system program listings are presented in Appendix I.

3.8 Processor Synchronization Strategy

The previous sections have described individual elements of the RMCS. This section focusses on the operation of larger functional blocks of the system, that are used to establish and maintain processor synchronization throughout a given task.

The processor synchronization hardware in the RMCS is sufficiently flexible to support a variety of monitoring, scheduling, and control strategies. A programmer may adopt whatever method provides satisfactory performance and programming ease. Only the issues and conventions of *processing states* (and their transitions) must be regarded.

Processor synchronization is required to ensure error-free interprocessor communications. In the RMCS, processor synchronization is implemented through a combination of the interlocked data transfer protocol, the SRQbus protocol, and the maskable SRQ acknowledge service. All transactions are asynchronous; periods of unnecessary system inactivity may be avoided, and overall performance and reliability are enhanced.

At various times during execution of a program, data transfers between processors are required. Slave processors have no knowledge (at the hardware level) of the current network configuration; they simply transmit and receive data via their PI/T ports according to their programs. Synchronization during message transfers is provided by the slaves' PI/T hardware handshake protocol, and the ICN. It is the master's responsibility to keep valid communications paths available, and to ensure that slaves commence transactions in the prescribed order. It is necessary to prevent slaves from performing I/O functions until valid paths are established between the processors participating in the transaction. The SRQbus and maskable SRQ acknowledge strategy fulfill such a purpose. Processor synchronization may best be described using a specific example.

Suppose that the slaves are to perform a program where slave #0 computes some data, which must be sent to slave #2, and after more calculation, slave #0 must send data to slave #1. Slaves #1 and #2 do nothing beforehand, but wait for their respective input. Initially no paths exist between processors.

The slaves execute their programs after reaching the *execute* processing state. Slave #0 calculates, while slaves #1 and #2 request service of the master, and wait for acknowledgment. The master's segment of the application program includes code which modifies the ESM to specify that requests from slaves #0 and #2 are forthcoming, and the master waits for both. The SRQ from slave #2 is acknowledged, but not that from slave #1. Slave #2 continues with its program, in which it attempts to receive data via its input port by first asserting RFD*. Since no path exists, nor is there a ready transmitter available, DAV* remains negated, and slave #2 waits. Eventually, slave #0 completes its calculations, requests service, and is acknowledged by the master. It continues with its program, in which it attempts to transmit data, by first sensing the state of the RFD* signal. It finds the signal negated, since a path still does not exist. The master program continues, and eventually, it reconfigures the network so that slaves #0 and #2 may communicate. Once the path is established, slave #0 senses an asserted RFD* from slave #2, and a block of data is transferred according to the interlocked data transfer protocol described in section 3.3.2, figure 3.3.2.4. The block length may be pre-defined, requiring both slave #0 and slave #2 to have the length specified in their programs. The block length may alternatively be variable, in which case slave #0 must provide a block-length specifier, and slave #2 must expect the specifier at the start of the transmission, which it subsequently uses as the initial value in a byte counter.

During the transfer, the master proceeds with its own segment of the program, and reaches a point where it expects SRQs from both slaves #0 and #1 prior to commencing the second transfer. Slave #1 has had a request pending since the start, and it is finally

acknowledged. Slave #1 attempts to receive data at its port, but once again, no path exists, so the slave waits. Slave #0 completes its transfer to slave #2, and it requests service again. The master acknowledges, and once again slave #0 attempts to transfer data. It finds that it cannot, since although there is a path to slave #2 already established, that slave is presently not accepting data[1]. The master configures the necessary path between slave #0 and #1, and the data transfer takes place. A timing chart of the various events in the described process is shown in figure 3.8.1.

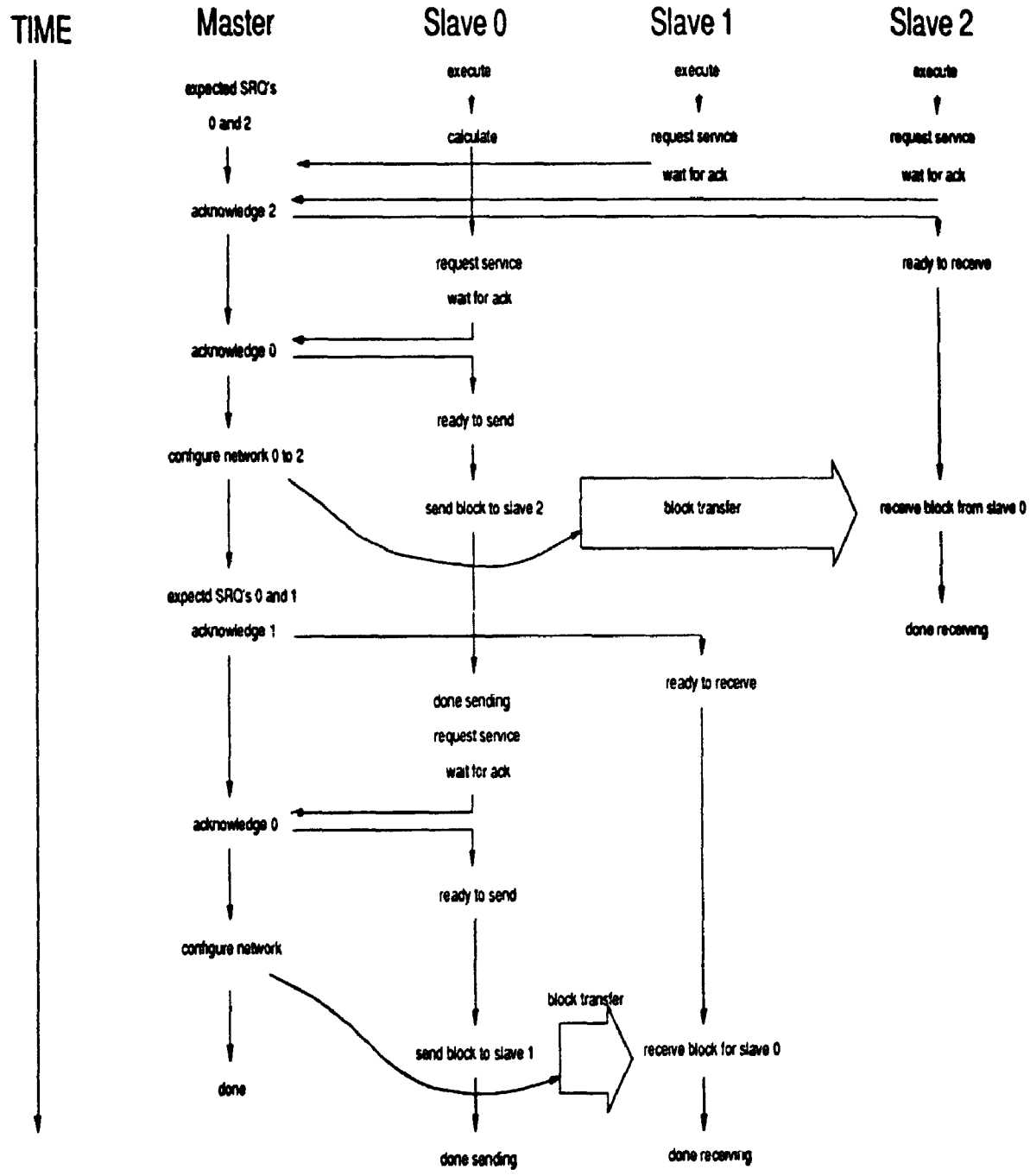
In the above example, the slaves reached their respective I/O routines prior to a path being configured, and thus the transfers took place upon institution of the paths. Often, the network may already have a correct path established. Suppose that initially a path existed between slaves #0 and #2. Upon slave #2's acknowledgement, it asserted RFD*, which would reach slave #0. Slave #0 is not attempting to send, and it will not until after its SRQ has been acknowledged. The master program in this case does not reconfigure the network after acknowledging slave #0's SRQ, since the proper configuration is already in place. The transfer commences after slave #0 is acknowledged, and the slave attempts to send data. Therefore, when paths between system processors already exist, an SRQ/SRQACK cycle is not necessarily required; the interlocked data transfer protocol is sufficient to establish and maintain processor synchronization throughout the transaction until a change in communications path configuration is required.

In the example, it was irrelevant whether the expected S' s within the two pairs occurred in a particular order. If, for whatever reason, the SRQ from slave #0 had to be acknowledged first, followed by that of slave #2, the master program must be encoded such

---

1 As a further complication, suppose that slave #2 was to accept more data after the slave #0 to slave #1 transaction, and it could reach its receive-data routine before the network was configured to support that transfer. An error could occur, with slave #2 receiving data that was intended for slave #1. In this case, an SRQ/SRQACK cycle must be used to prevent slave #2 from reaching its receive-data routine until the appropriate time.

Figure 3.8.1 Timing chart of events described in processor synchronization example.



that only a request from slave #0 was expected on the first call to the SRQ acknowledge service routine (TRAP #9). Upon return, the ESM would be re-written so that a request from slave #2 is expected and acknowledged. The maskable SRQ acknowledge strategy allows any ordering and combination of SRQs to be serviced, and thus any sequence of events can be controlled by the master processor cell.

The preceding sections describe the operating principles of the RMCS. The formalization of basic *processing states* and state transitions, available I/O subroutines, asynchronous interlocked data transfer protocol, SRQbus protocol, and maskable SRQ acknowledgement combine to provide a consistent and flexible processor synchronization strategy. Shared system busses and their inherent bottlenecks, shared memory (and the necessity for shared-data protection techniques), and common system clocking are non-existent in the RMCS. The system is asynchronous, with all transactions made on an event-driven or data-availability basis, offering a high degree of reliability, performance and programming ease.

4 Experiments

This chapter centres on the experiments used to characterize performance of the reconfigurable multicomputer system. The hardware tests focus on the programmable signal router devices and the interprocessor communication network. The details of the various performance test programs are presented. The method of measurement common to all performance tests is described, as well as the procedures used to determine the system overheads. The chapter also presents results obtained in performance testing of the reconfigurable multicomputer, and its critical functional blocks. Results are summarized graphically wherever appropriate; complete experimental data are presented in tabular form in Appendix A.

4.1 Hardware System Tests

The reconfigurable multicomputer system comprises a number of subsystems which were individually tested at various stages of its design and construction. The processing elements used in the system are identical to that described in Smeulders (Smeulders, 88), with the addition of the PI/T device and its Cellbus interface. The subsystems unique to the RMCS encompass the programmable signal router devices, the interprocessor communication network, the network controller card, and the SRQbus interconnection. The following sections describe the procedures used to characterize the PSR and ICN. The SRQbus response time test procedure is deferred to section 4.4, following presentation of the common performance test measurement method.

4.1.1 Programmable Signal Router Device Characteristics

The programmable signal router's electrical characteristics were determined to facilitate the final system design. Since Chum (Chum, 89) presents the input and output cell electrical characteristics, his results were used in the presented thesis. However, signal propagation delays, input set-up and hold times, reconfiguration delay, and output enable delays were measured, using procedures to be described. The tests were performed on a

sample of five devices, and critical minimum and maximum parameters are presented in table 4.1.1.5.1. All input signals used were CMOS standard for V_{dd}=5V, with 10 ns rise and fall times. All measurements were performed at the half-maximum signal amplitude level. Output signals were loaded with a single LSTTL device input (74LS244). The measurement instrument used was a Tektronix 2215A oscilloscope which provides a 1% of full scale reading accuracy.

4.1.1.1 Propagation Delay Time Measurements

With the port configuration fixed, port output signals enabled, and the input DAV* latching signal inactive (transparent mode for internal latches), a square wave signal was applied to a data input, while the corresponding data output was monitored. The time between transitions of the input and output signals was measured (at the half-maximum voltage level), yielding the data signal propagation delay times, t_{dpdr}, for rising signals, and t_{dpdf}, for falling signals.

The same method was used to determine the input DAV* to output DAV* propagation delay (t_{dvpd}), as well as the input ACC* to output ACC* propagation delay (t_{dcpd}).

4.1.1.2 Data Set-up and Hold Time Measurements

With the port configuration fixed, and port output signals enabled, a square wave signal with fixed duty-factor was applied to a DAV* input. A signal with variable pulse width was applied to a data input, and the corresponding data output was monitored. The time required from data input valid to DAV* asserted, which ensures that the output signal matches the input, is the minimum set-up time between data input and DAV* (t_{su}). It was determined by varying the data input pulse width until the input and output signal levels exhibited disagreements.

The hold time was measured in a similar manner, with the variable pulse-width signal applied to DAV*, and the fixed duty-factor signal applied to a data input. The time required from DAV* asserted to input data invalid, for which the output data signal matches the input level prior to assertion of DAV*, is the minimum hold time between data input and DAV* asserted (t_{bde}). It was determined by varying the DAV* pulse width until the output signal switched following assertion of the DAV* signal.

4.1.1.3 Reconfiguration Delay Time Measurements

With the output ports enabled, a square wave signal was applied to both the PC0 and PC1 configuration control pins, alternately switching the device between "A→C/B→D" and "A→D/B→C" configurations. A logical high voltage was applied to a port A data input, and a low was applied to a port B data input. The port C data output signal therefore switches in opposite phase (with some delay) to the input square wave, while the port D output is in phase. The time between the input configuration control signal transition to the output transitions was measured as the reconfiguration delay time (t_{rfd}).

4.1.1.4 Port Output Enable Delay Measurements

The device was configured in a fixed topological state (PC0, PC1 held at a constant voltage), with a square wave signal applied to port configuration control bits PC2 and PC3. Port A and port B data inputs were held low. When a port is disabled, its outputs are driven high, and thus port C and port D output signals switch in opposite phase to that of the signal applied at PC2 and PC3. The time between signal transitions represents the port output enable delay time (t_{oed}).

4.1.1.5 PSR Device Characteristics: Test Results

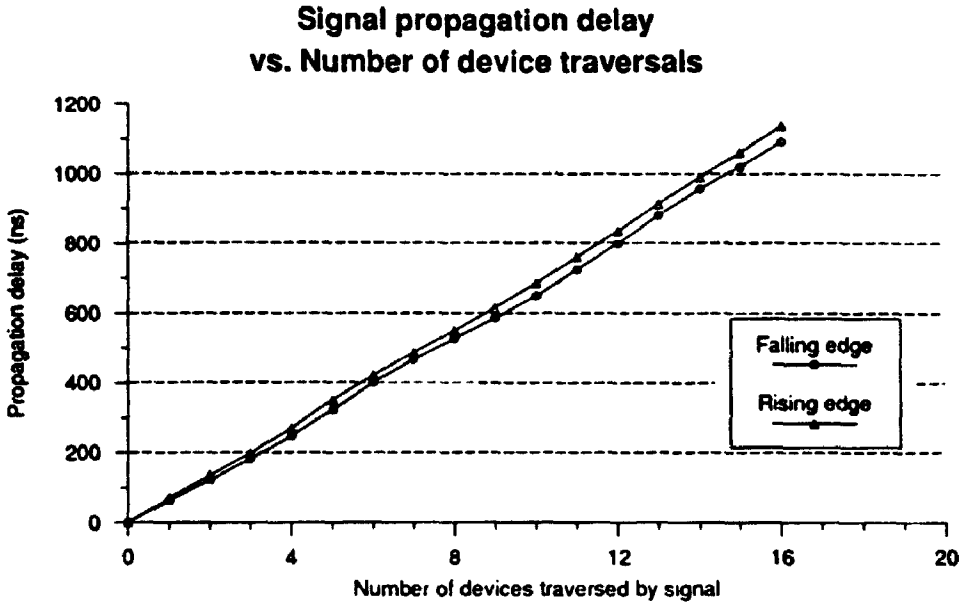
Table 4.1.1.5.1 Summary of PSR device characteristics.

Parameter	Description	min (ns) ±0.5	max (ns) ±1.0
t_{dpdr}, t_{dpdf}	Data signal propagation delay time, rising and falling	-	75
t_{dvpd}	DAV* signal propagation delay time	-	65
t_{dcpd}	ACC* signal propagation delay time	-	50
t_{su}	Set-up time, data valid to DAV* asserted	40	-
t_{hlda}	Hold time, DAV* asserted to data invalid	40	-
t_{rfd}	Reconfiguration delay time	-	90
t_{oed}	Output enable delay time	-	50

4.1.2 Interprocessor Communication Network Characteristics

The propagation delay of signals traversing the ICN was tested by applying an input signal to a network input port, and monitoring the output ports of successive PSR devices in the signal's path. The network configuration for the test was fixed. Loading for signals was provided by the successive devices in the network, the same loading condition present under normal system operation. Results were recorded for both rising and falling transitions, and are shown in figure 4.1.2.1. Appendix A shows tabulated results.

Figure 4.1.2.1 ICN Signal Propagation Delay characteristics.



4.2 Performance Test Measurement Method

The experiments presented in the following sections have higher complexity, and often span longer time intervals than can accurately be observed using the methods of the previously described experiments. They involve a high degree of software control over the hardware resources under test. For the data transfer rate and processor synchronization overhead tests, the measurement method does not necessarily yield absolute maximum ratings; the ICN and SRQbus have bandwidths much higher than the results imply. The method measures the actual system operating parameters using optimum control programs; the processor and PI/T performance is a factor, and must be incorporated.

Each processing element in the system incorporates a programmable timer which is used to measure program execution time (MC68230, 83). The device must be initialized to establish the desired counting mode, and an *initial count* value is written into its

"counter pre-load" register. When the timer is enabled, the content of the pre-load register is copied to the counter register, which is decremented once every 32 cycles of the 8 MHz system clock; 4 μs precision may be achieved in all measurements.

The master's timer is initialized during an unmeasured phase of program execution. Immediately prior to a section of code whose execution time is to be measured, a single instruction is used to enable the timer. When the program under test has been completed, the timer is disabled. The content of the counter register is subtracted from the initial count value to determine elapsed time. Overhead for timer control is very small (generally < 30μs, depending on the instructions and addressing modes chosen), and is generally insignificant compared to the execution time of the code under test. The pre-load and counter registers are 24-bits wide, allowing measurement of programs up to 67.1s long. For longer intervals, the device may assert an interrupt each time the counter register reaches zero, and the associated interrupt service routine maintains a count of the "roll-over" occurrences. At the end of execution, the roll-over count, along with the remaining counter register content may be used to determine overall elapsed time.

Due to the asynchronous nature of interprocessor communications and the SRQbus, a program which utilizes those elements may exhibit varying execution times. In other words, if the same program is performed with the same data repeatedly, the execution times are likely to be slightly different. The performance test programs (with the exception of the data-transfer rate programs) were written such that the measured portion of execution time is repeated 256 times, with the maximum, minimum, and average times logged throughout. The times are reported to the user terminal upon completion of the test. Tests are measured by the master processor for entire process execution time. In some tests, slave processor execution times are also of interest, and are measured by the slave's resident timers. At the end of a 256-pass test, the master processor extracts the slave times, and reports them to the user terminal.

4.3 Data Transfer Rate Measurement

The data communication rates between processing elements were measured using a single test program which performs transfers of data from master to individual slaves, slaves to master, and then once again from the master to all slaves using the broadcast mode. The tests utilize the TRAP service routines for block data transfers. Elapsed time to transfer data in blocks of 8,16,32,...,512 bytes is measured by the master processor, and reported to the user terminal. Flowcharts of the master and slave programs are shown in figures 4.3.1 and 4.3.2, respectively. The program code for the data transfer rate measurement test is presented in Appendix J.

The data transfer rate performance test results are shown in figures 4.3.3 and 4.3.4, with best-fit straight line approximations. The master-to-slave and slave-to-master results were combined to provide an average measure of data transfer rates independent of direction. A graph of the average data transfer rates with best-fit straight line approximation is shown in figure 4.3.5. Appendix A shows tabulated results.

Figure 4.3.1 Flowchart of Data Transfer Time measurement program: Master.

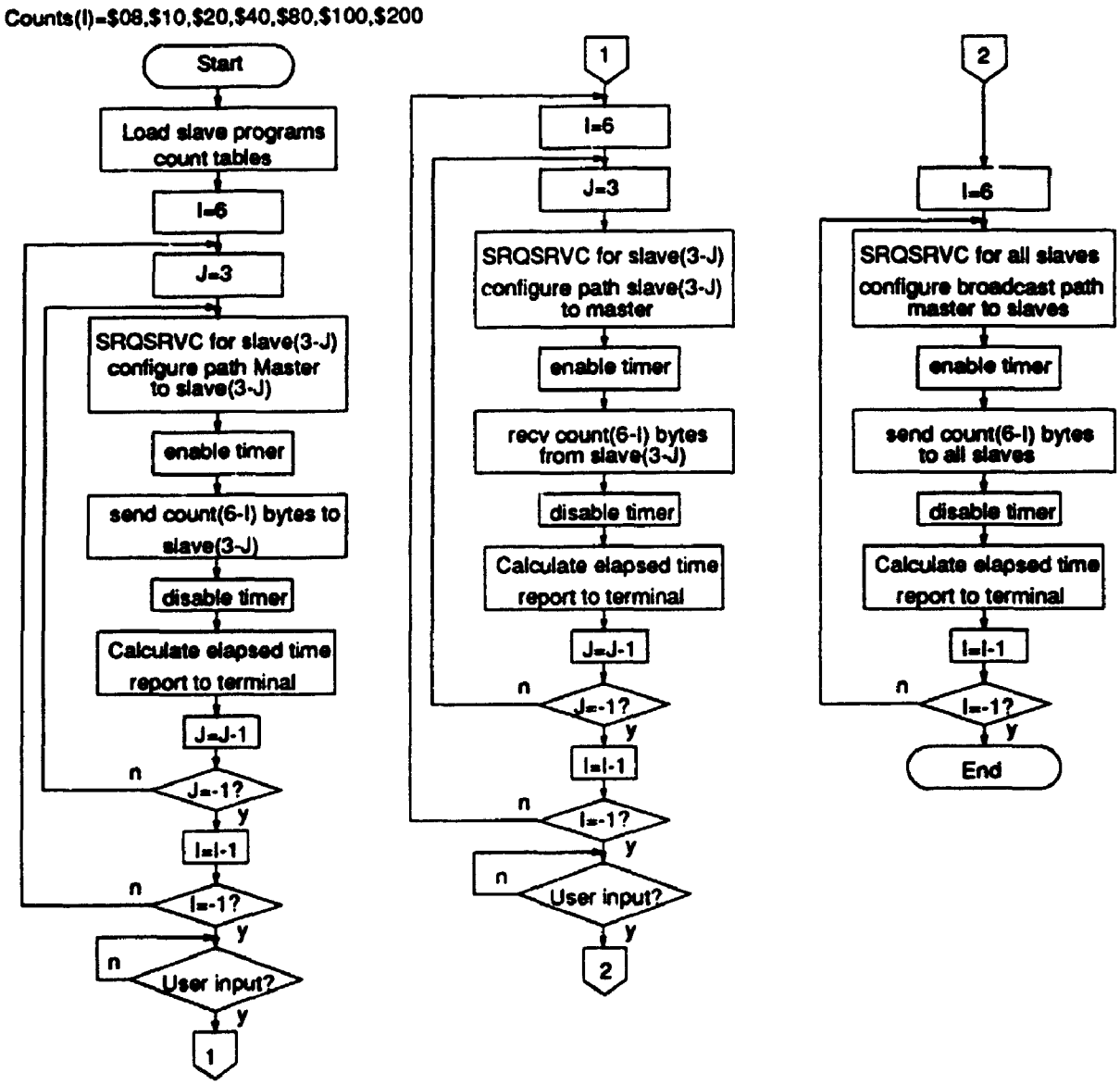


Figure 4.3.2 Flowchart of Data Transfer Time measurement program: Slaves.

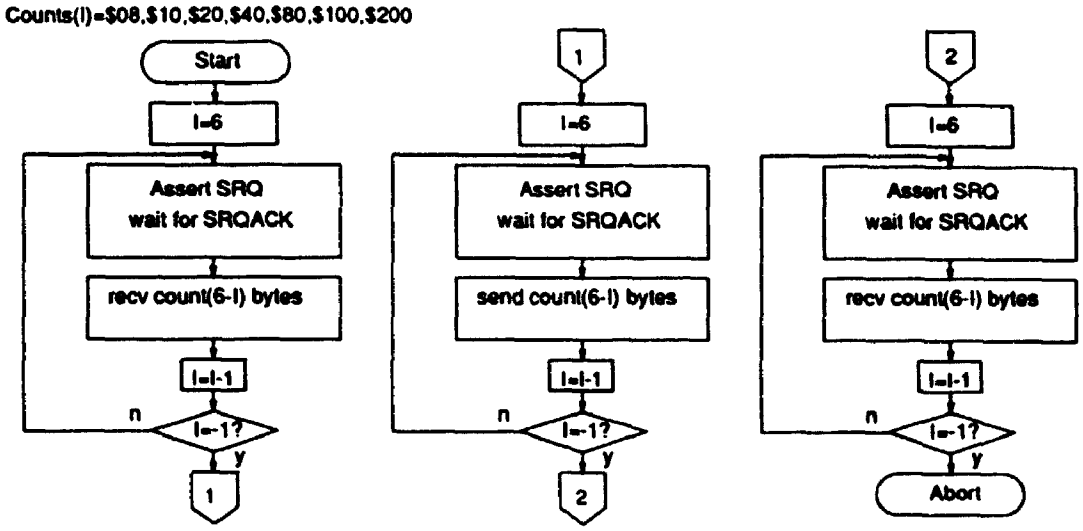
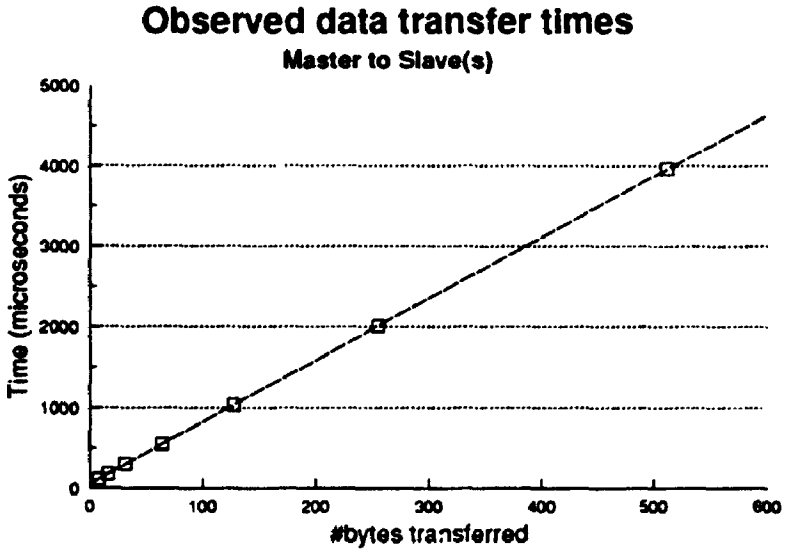
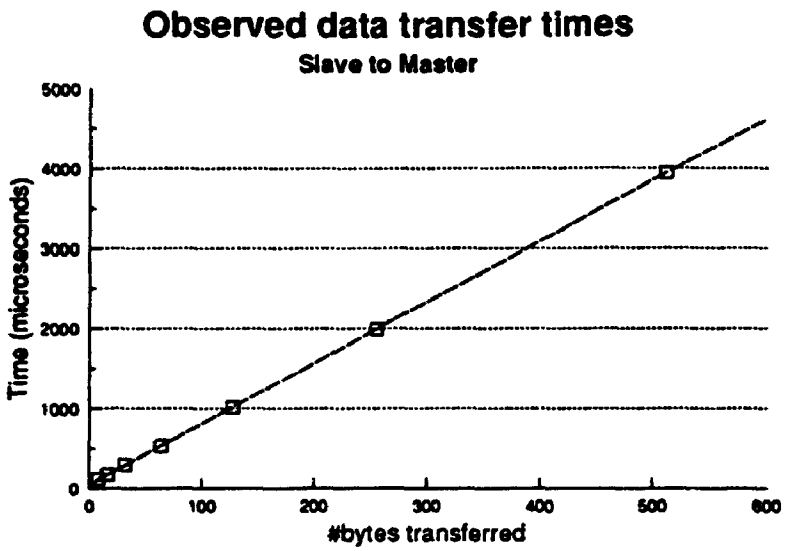


Figure 4.3.3 Data Transfer Rate test results: Master→Slave(s).



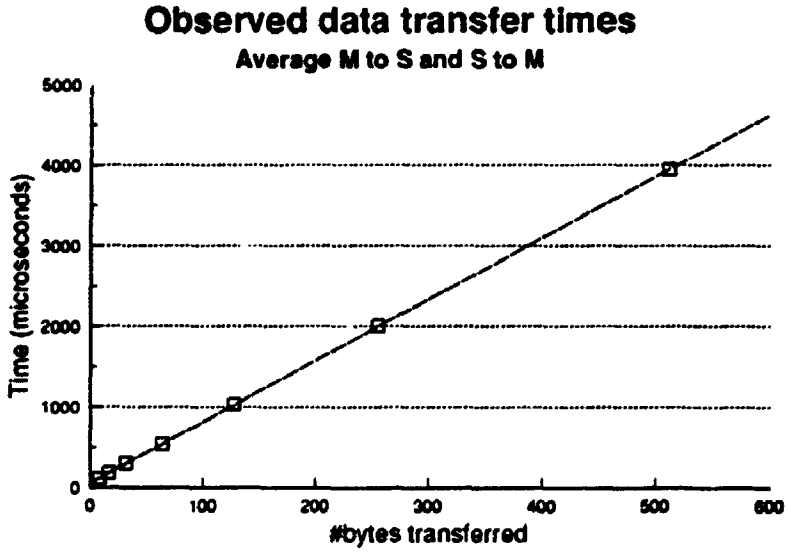
$$Time = [7.627 \times (byte\ count) + 59.16]\ \mu s$$

Figure 4.3.4 Data Transfer Rate test results: Slave→Master.



$$Time = [7.607 \times (byte\ count) + 48.35]\ \mu s$$

Figure 4.3.5 Data Transfer Rate test results: Average transfer times.



$$Time = [7.617 \times (byte\ count) + 53.76]\ \mu s$$

4.4 Processor Synchronization Overhead Measurement

Processor synchronization overhead was measured while the matrix multiplication experiments were conducted. The matrix multiplication measurement code was edited to measure the various processor synchronization intervals present in the code. The procedure was adopted since it was an opportunity to measure the minimum synchronization period; where a service request was guaranteed to be pending (see section 2.3.2) prior to the master processor calling the service request acknowledge routine. Due to the nature of the service routine code, the time to service a request is dependent on the source of the particular request, although differences are expected to be very small. The tests measured time to service single and multiple simultaneously pending requests. The code to measure the service time is as shown:

```
MOVEA.L #TCR,A4          ;POINTER TO TIMER CONTROL REGISTER
MOVE.B #ENABLTM,(A4)     ;ENABLE TIMER TO COUNT
MOVE.B #MASK,D0          ;SET UP SERVICE MASK IN D0
TRAP #SRQSRVC            ;CALL SRQ ACKNOWLEDGE ROUTINE
CLR.B (A4)               ;STOP TIMER
```

The appropriate mask is used for each of the tests (see section 3.7.4.9). The process was repeated 256 times, and the average times recorded are shown in table 4.4.1.

Table 4.4.1 Processor Synchronization Time test results.

Number of slaves acknowledged	Synchronization time (μs)
1	88
2	112
3	140
4	172

4.5 System Performance Tests

The experiments described in sections 4.3 and 4.4 measure "low-level" aspects of system operation which are common to all multiple processor tasks performed by the RMCS.

The experiments described in the following sections are considered "high level" tests, and reflect system performance in executing typical application programs. The applications make use of the ICN, and SRQbus, and therefore the performance of those system components influences overall test results.

4.5.1 Matrix Multiplication Performance Tests

The matrix multiplication performance tests were performed on the RMCS configured for parallel mode processing. The task was assigned among the slave processors on a row-by-row basis, as described in section 2.2.1. Experiments were performed using square matrices of dimension 4, 8, 12, 16 and 20. Floating-point input and output data were used in the tests, to preserve 23-bit precision in calculations. The floating-point format adopted is as shown in figure 4.5.1.1. Normalized numbers (most significant bit of the fraction is always set) are used throughout the tests, and no hidden bit is assumed. The exponents are biased by 128. The format permits special-case numbers to be represented, as well as a unique zero. Floating-point subroutines test input data for special values and attempted illegal operations. A summary of floating-point number special-cases is given in table 4.5.1.1.

Figure 4.5.1.1 Floating-point Number format.

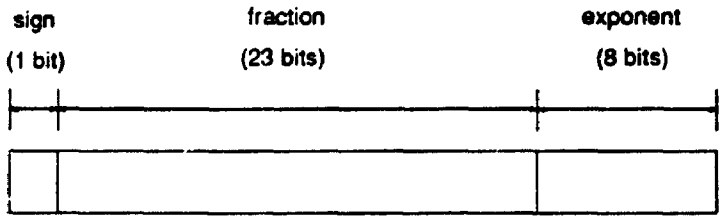


Table 4.5.1.1 Floating-point Number special cases.

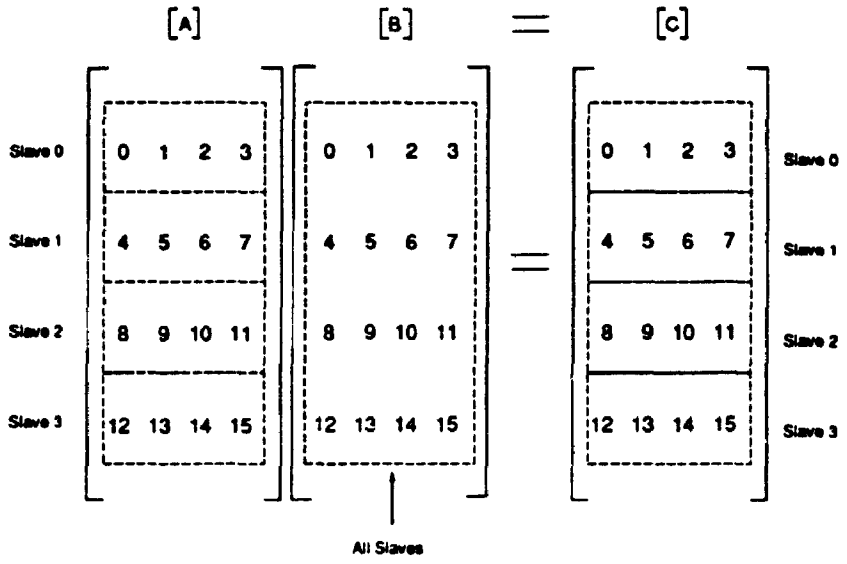
Fraction	Exponent		
	0	1-254	255
+0	0	not used	$+\infty$
-0	-0	not used	$-\infty$
≠0	not used	$-1^s \times (0.F) \times 2^{E-128}$ MSB[F] = 1	NaN

The full IEEE floating point standard is not used in the tests since the MC68000 instruction set provides simplified and more efficient computation when the standard outlined above is adopted. The IEEE standard also represents numbers smaller than 1.0×2^{-127} in non-normalized form, requiring additional operations to test whether a number is normalized or not, and to accommodate both forms. For the test programs used in the thesis, the additional instructions are superfluous.

Following the method outlined in section 2.2.1, where r (the number of processors) is 4, slave #0 calculates the first $n/4$ rows of the product matrix, slave #1 the next $n/4$ rows, and so on. To evaluate the matrix equation $[C]=[A][B]$, all of the elements of matrix $[B]$ must be available to all slaves. The elements of the $n/4$ rows of matrix $[A]$, which correspond to the rows of the product matrix for which the slave is responsible, are required by the respective slaves. Figure 4.5.1.2 shows the assignment strategy.

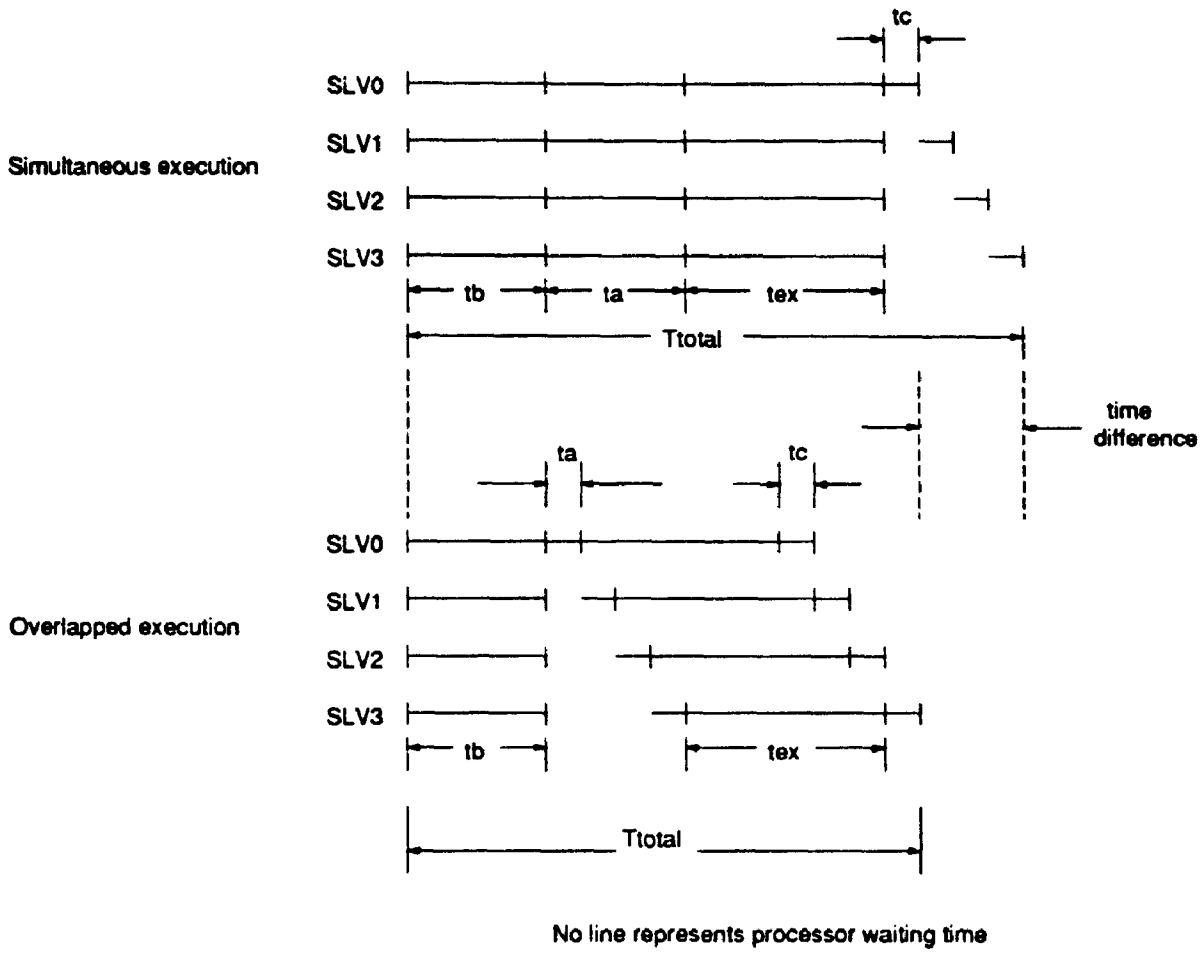
The partitionability of the problem, together with the control strategy of the RMCS, allows matrix multiplication to be performed in an *overlapped parallel* manner, where slave processors commence execution while others are receiving or awaiting arrival of their input data. The strategy is potentially faster than postponing calculation until input data has been received by all slaves. Figure 4.5.1.3 shows a set of timing diagrams comparing the two methods.

Figure 4.5.1.2 Matrix Multiplication Slave problem assignment.



In the diagram, t_b represents the time necessary to broadcast the entire matrix [*B*] to all slaves; t_a represents the time to send the matrix [*A*] data (in the simultaneous case, it is broadcast, in the overlapped case, 1/4 of the elements in matrix [*A*] are sent to each slave); and t_c represents the time required for a slave to up-load its results to the master. Processor synchronization intervals are neglected in the diagram. Clearly, the overlapped execution strategy will yield lower overall times. The diagram assumes that t_{ex}, the slave program execution time, is the same for each slave processor. In reality, due to the data-dependent execution time of the processor's multiply instruction, and special-case number actions in the floating-point add and multiply subroutines, t_{ex} may be highly dependent on the input matrix data. For this reason, two multicomputer programs were tested: the first with slave output data collected in order of increasing slave index (MATFPM); the second collects data on a first-come, first-served basis (MATFPFC). Tests were performed using different sets of input data: one with all elements of [*A*] and [*B*] equal to 0 (FPINDAT1), another with all elements equal to $7FFFFF80 (FPINDAT2), and a third consisting of elements equal to

Figure 4.5.1.3 Simultaneous versus Overlapped execution for Matrix Multiplication.



either $7FFFFF80 or $FFFFFF80 (FPINDAT3). The input data files and test programs are shown in Appendix J. Program flowcharts for the matrix multiplication tests (uniprocessor, multicomputer master and slave programs) are shown in figures 4.5.1.4 through 4.5.1.6. The floating-point addition and multiplication algorithms are outlined in Cavanaugh (Cavanaugh, 84). The multicomputer programs execute 256 times, with average, maximum, and minimum execution times recorded. To aid performance analysis, the slave processors measure their execution time, t_{ex}, and report it to the master at the end of the 256-pass run.

Figure 4.5.1.4 Flowchart for Uniprocessor Matrix Multiplication performance test.

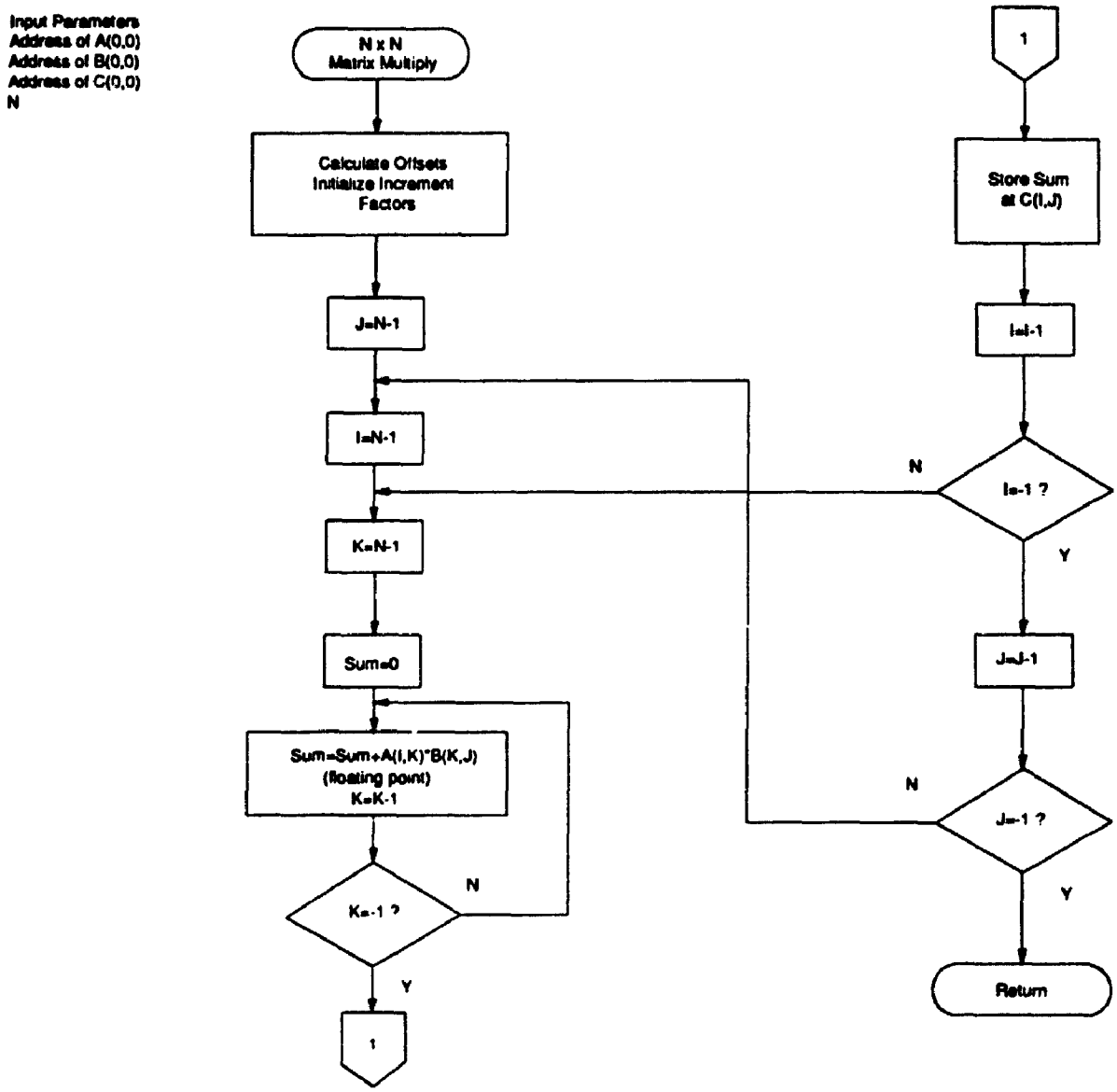


Figure 4.5.1.5 Flowchart for Multicomputer Matrix Multiplication performance test, Master program.

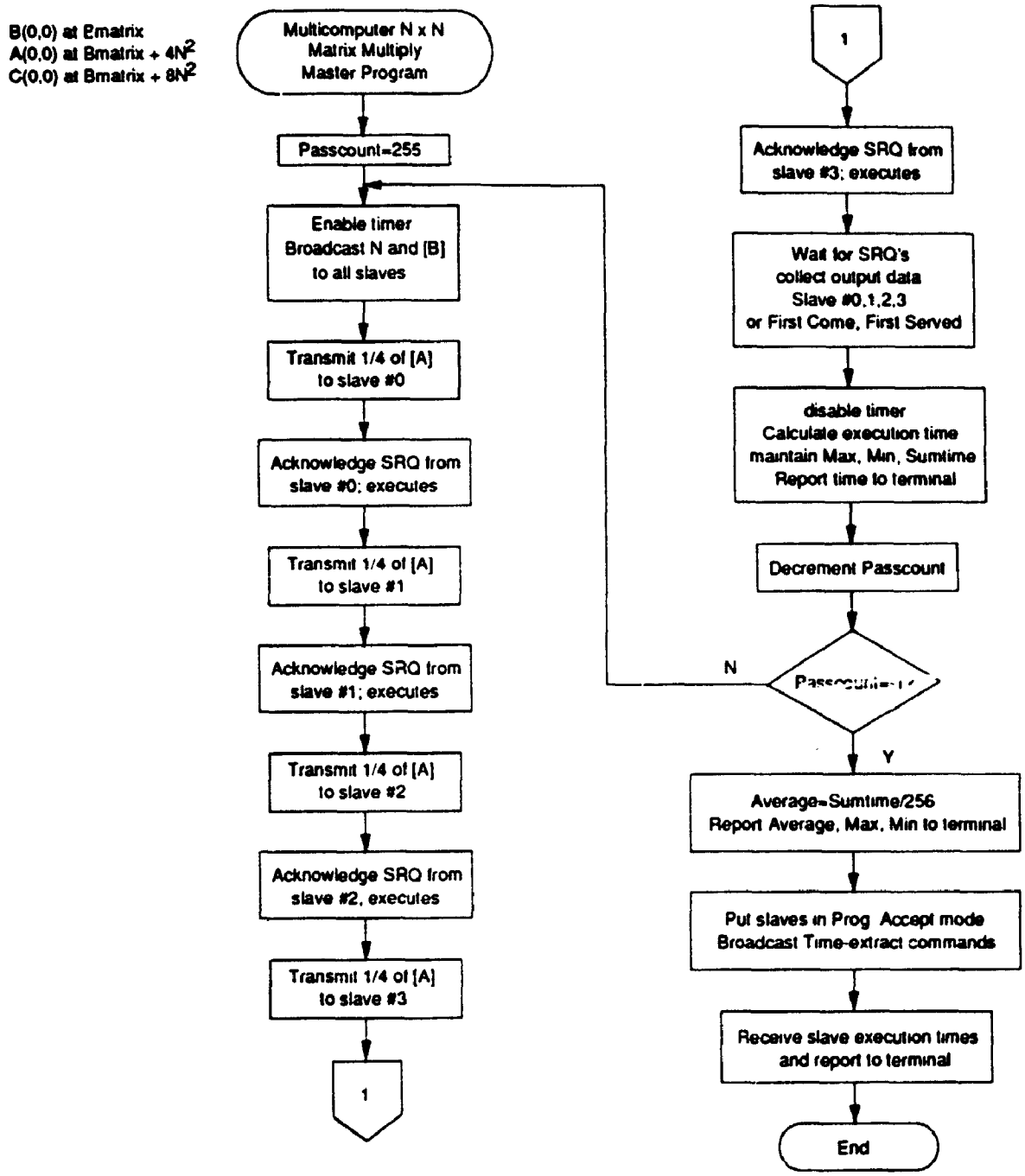
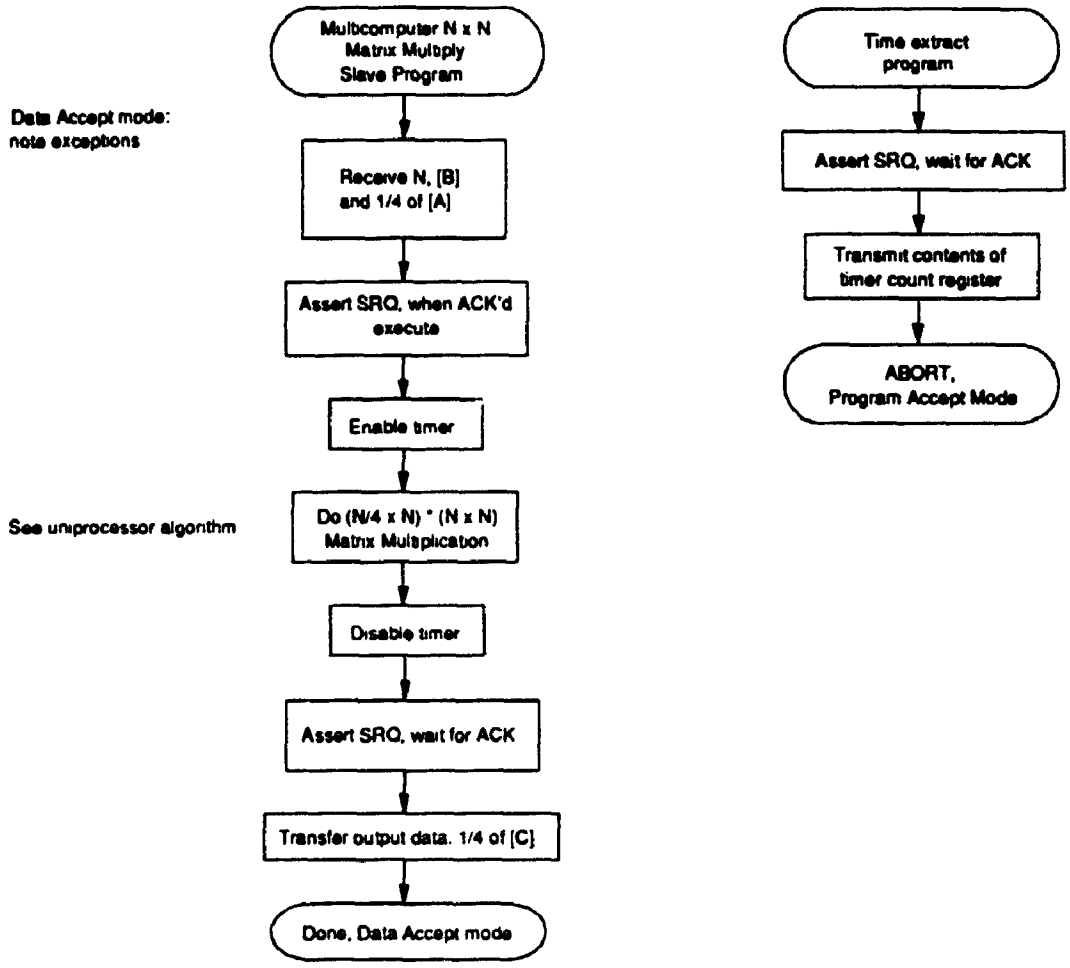


Figure 4.5.1.6 Flowchart for Multicomputer Matrix Multiplication performance test, Slave program.



Uniprocessor and overall multicomputer matrix multiplication performance test results are summarized in figures 4.5.1.7 through 4.5.1.12. Appendix A shows complete tabulated results, including individual slave execution times.

Figure 4.5.1.7 Matrix Multiplication tests, Uniprocessor and Multicomputer comparisons, input: FPINDAT1

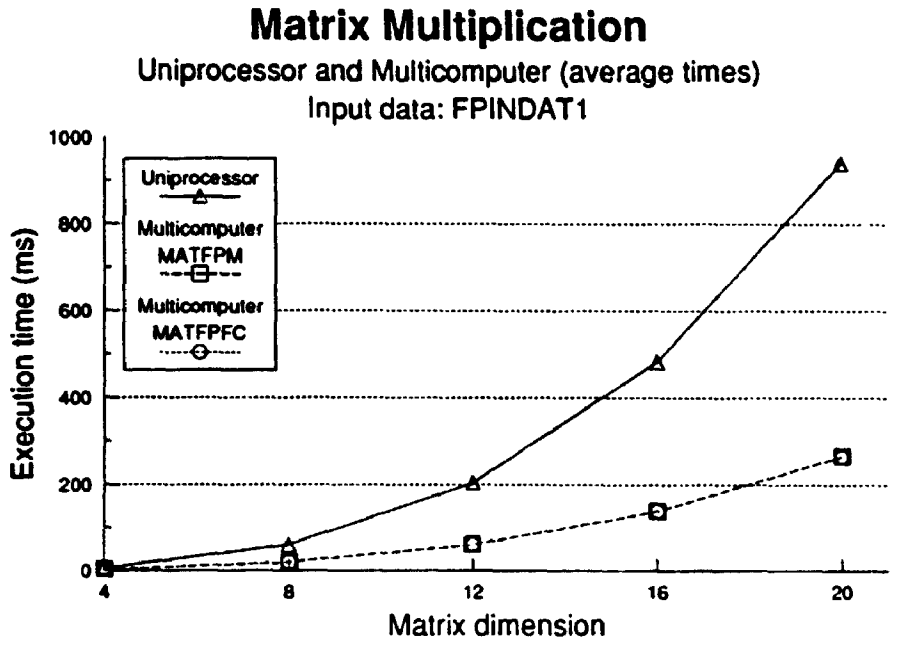


Figure 4.5.1.8 Matrix Multiplication tests, Uniprocessor and Multicomputer comparisons, input: FPINDAT2

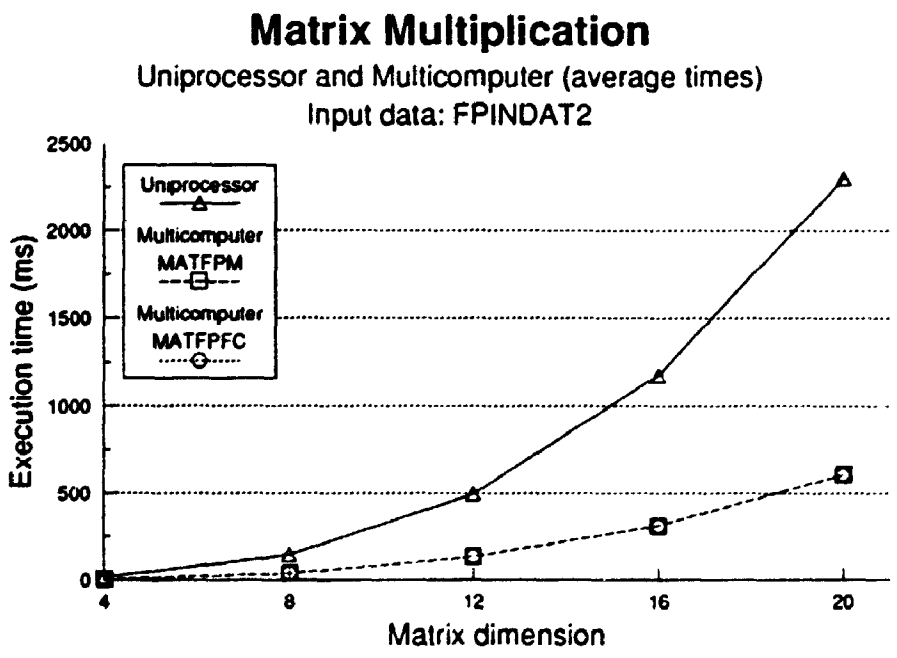


Figure 4.5.1.9 Matrix Multiplication tests, Uniprocessor and Multicomputer comparisons, input: FPINDAT3

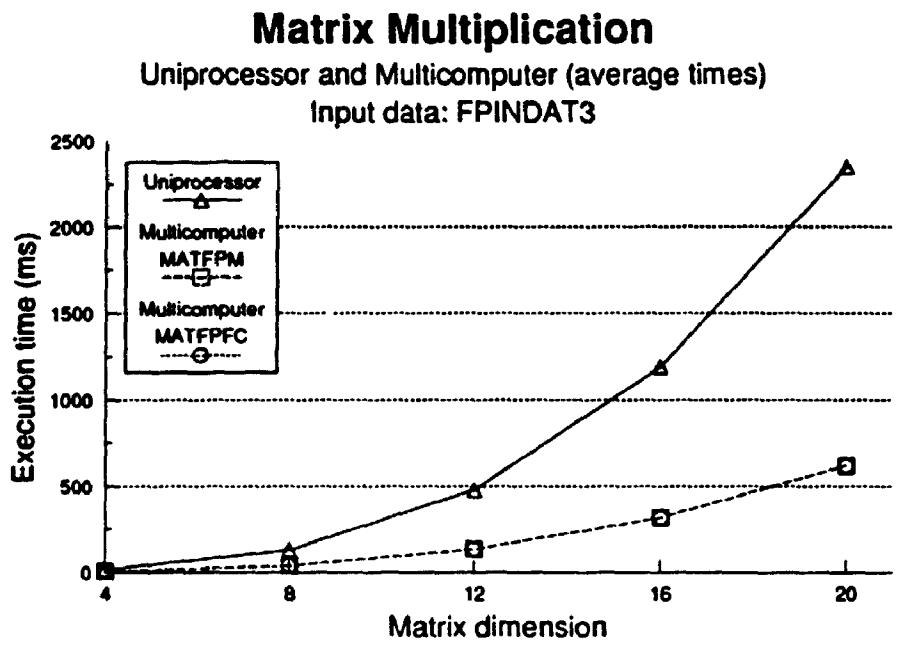


Figure 4.5.1.10 Matrix Multiplication tests, Multicomputer speed-up factors, input: FPINDAT1

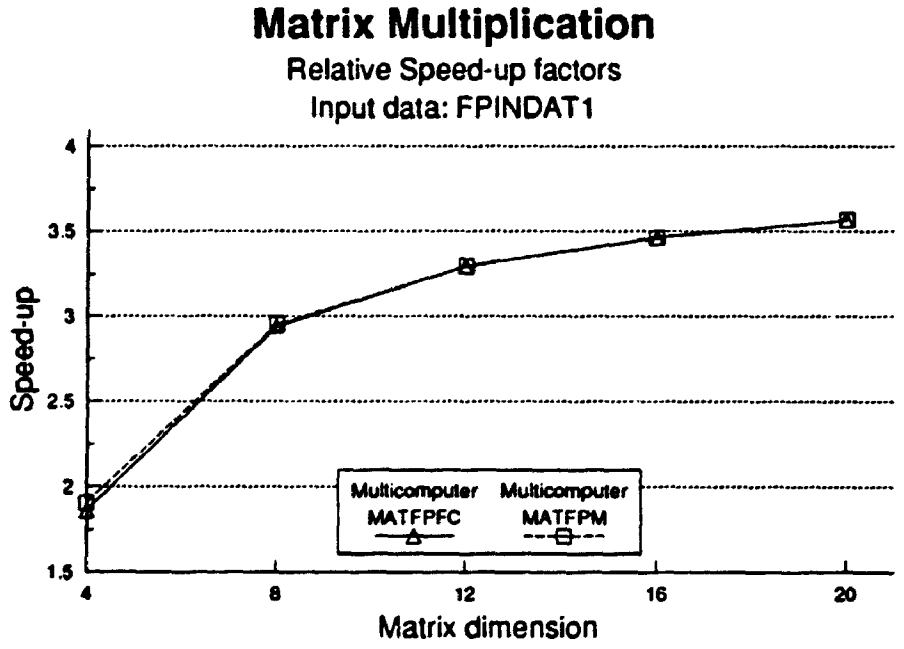


Figure 4.5.1.11 Matrix Multiplication tests, Multicomputer speed-up factors, input: FPINDAT2

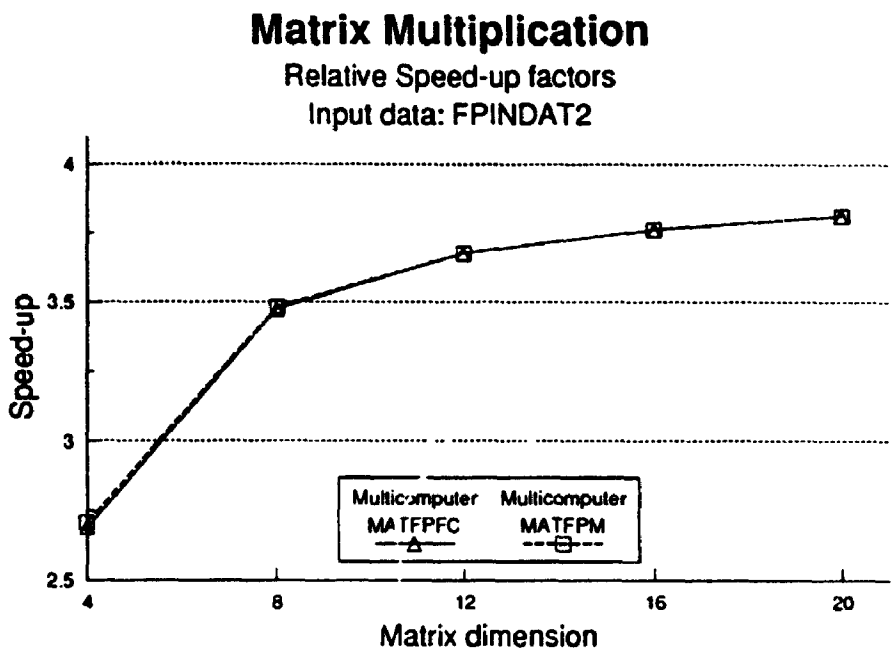
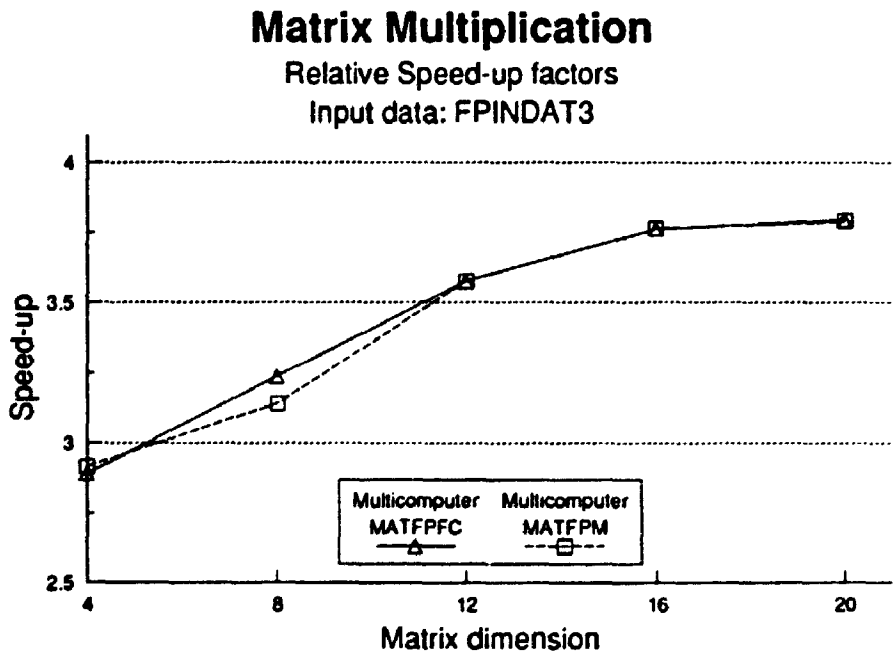


Figure 4.5.1.12 Matrix Multiplication tests, Multicomputer speed-up factors, input: FPINDAT3



4.5.2 Fast Fourier Transform Performance Tests

The FFT calculations are performed on the RMCS following the theory outlined in section 2.2.2, for both uniprocessor and multicomputer tests. All experiments are conducted using a 256 complex-point problem size. Due to the data-dependent multiplication time of the microprocessor, different input data sets are used to test performance: DC signals of strengths 0, $3FFF, and $D555 (NULLDC, FULLDC, MAXDC); signals comprising one and eight sinusoidal components (COS1, COS8); and a random noise signal (NOISE). All input signals used are complex, with scaled, 16-bit integer real and imaginary components. The imaginary components of input signals are set to $0000.

For each of the input data sets, four test programs were executed, each with a different data retrieval strategy. The first (PFFT1), uses a fixed-order retrieval strategy which waits for slave #0 to complete computations, then up-loads its output, then awaits slave #1's completion, and so on. Another test program (PFFT2) waits for all slaves to finish calculations, then up-loads their output data in order of increasing slave index. The third program (PFFT3) exploits slave #0's potentially faster completion time due to the absence of twiddle-factor multiplications in its part of the problem. After slave #0's data is up-loaded to the master, all the remaining slaves are expected to have finished, their SRQ's are serviced, and their output data collected in order of increasing slave index. The last program (PFFT4) collects data from slaves on a first-come, first-served basis. The program OPT256 is used to measure uniprocessor performance with all input data sets.

In all tests, the programs are repeated 256 times, with average, maximum and minimum times recorded. The slave processor execution times are also measured, and reported at the end of a 256-pass test. Other related tests include measurement of the 64-point FFTs within the slave programs, sum-and-twiddle factor multiplication times, as well as the *time to solution*, when the frequency domain data has been computed, but has not been reported to the master. The various processing-phase times are useful for subsequent performance

analysis in Chapter 5. Flowcharts for the uniprocessor and multicomputer FFT performance tests are shown in figures 4.5.2.1 through 4.5.2.3. Program code and input data files used in the tests are shown in Appendix L.

Figure 4.5.2.1 Flowchart for Uniprocessor FFT performance test.

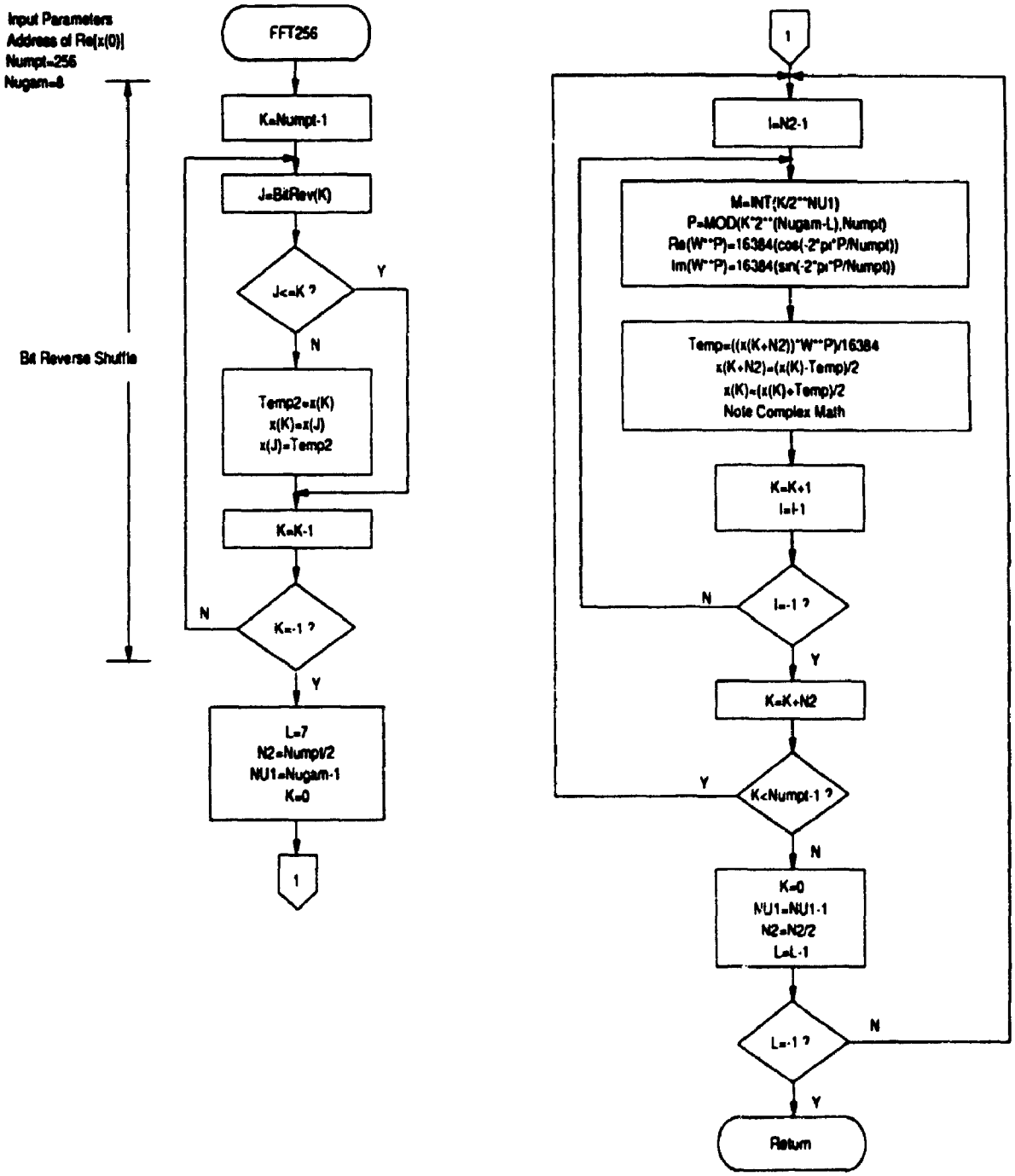


Figure 4.5.2.2 Flowchart for Multicomputer FFT performance test: Master program.

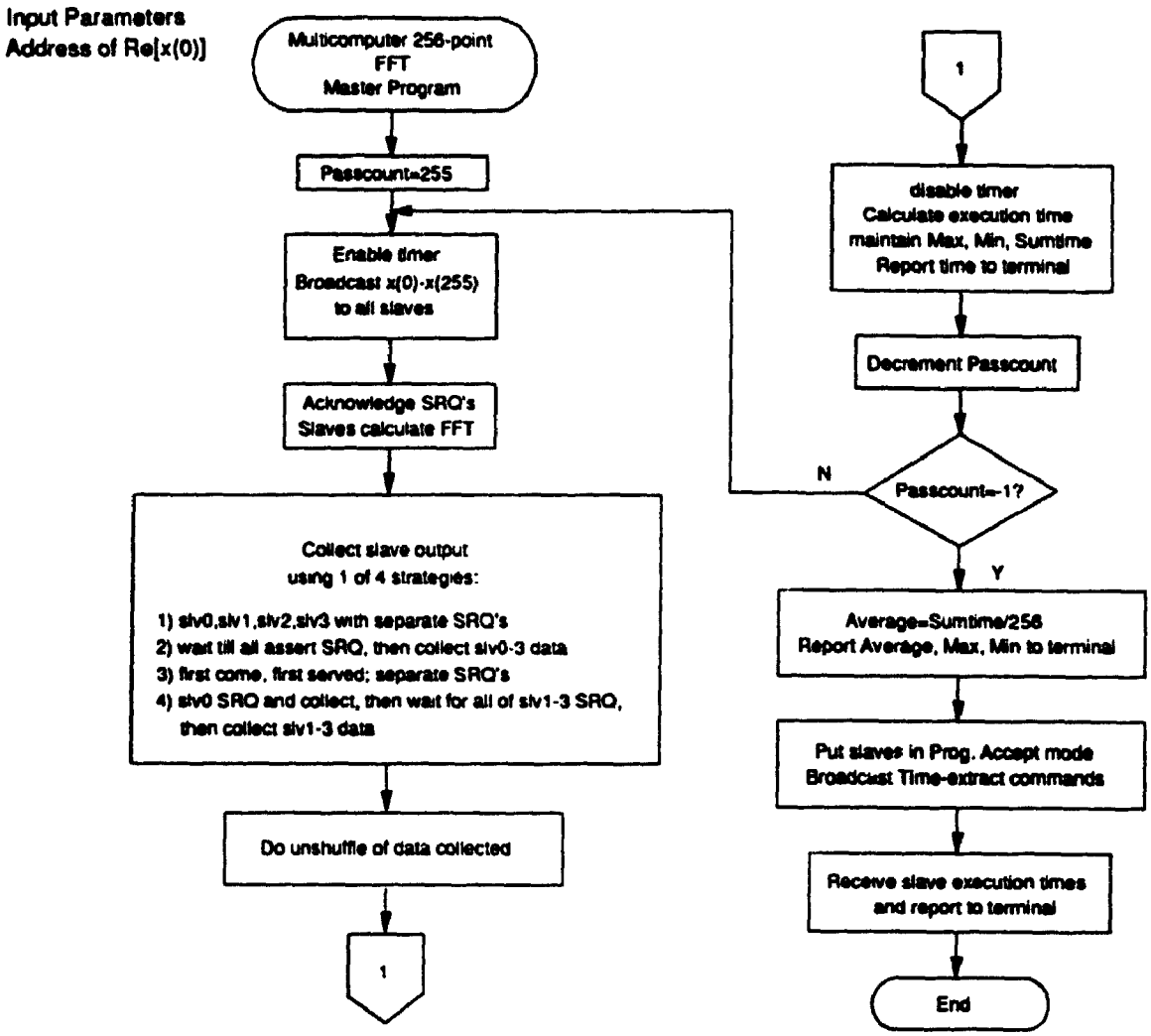
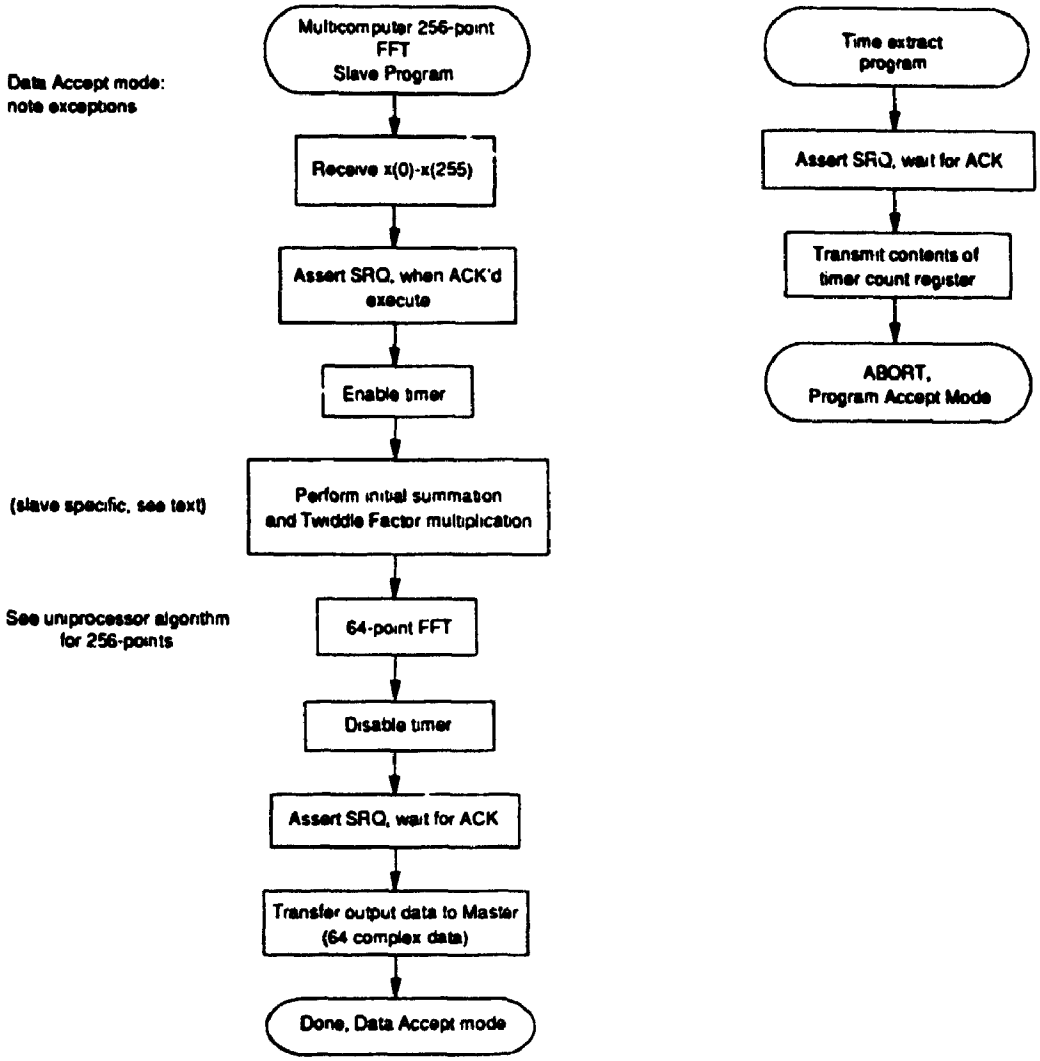


Figure 4.5.2.3 Flowchart for Multicomputer FFT performance test: Slave program.



Results of the FFT performance tests are summarized in tables 4.5.2.1 and 4.5.2.2 and figures 4.5.2.4 and 4.5.2.5. Appendix A shows complete, tabulated results, including individual slave execution times.

Table 4.5.2.1 Fast Fourier Transform tests: computation time speed-up factors summary.

Input filename	64-pt FFT only (average)	Average Slave Total time	Time to solution
NULLDC	5.28	4.15	3.21
FULLDC	5.28	4.16	3.21
MAXDC	5.32	4.20	3.26
COS1	5.29	4.19	3.19
COS8	5.27	4.17	3.23
NOISE	5.25	4.17	3.24

Table 4.5.2.2 Fast Fourier Transform tests: Multicomputer speed-up factors.

Input filename	Test program			
	PFFT1	PFFT2	PFFT3	PFFT4
NULLDC	2.62	2.53	2.63	2.62
FULLDC	2.62	2.53	2.63	2.62
MAXDC	2.64	2.57	2.64	2.64
COS1	2.65	2.52	2.63	2.64
COS8	2.66	2.56	2.67	2.66
NOISE	2.67	2.57	2.68	2.67

Figure 4.5.2.4 Fast Fourier Transform tests: Uniprocessor and Multicomputer execution time comparison.

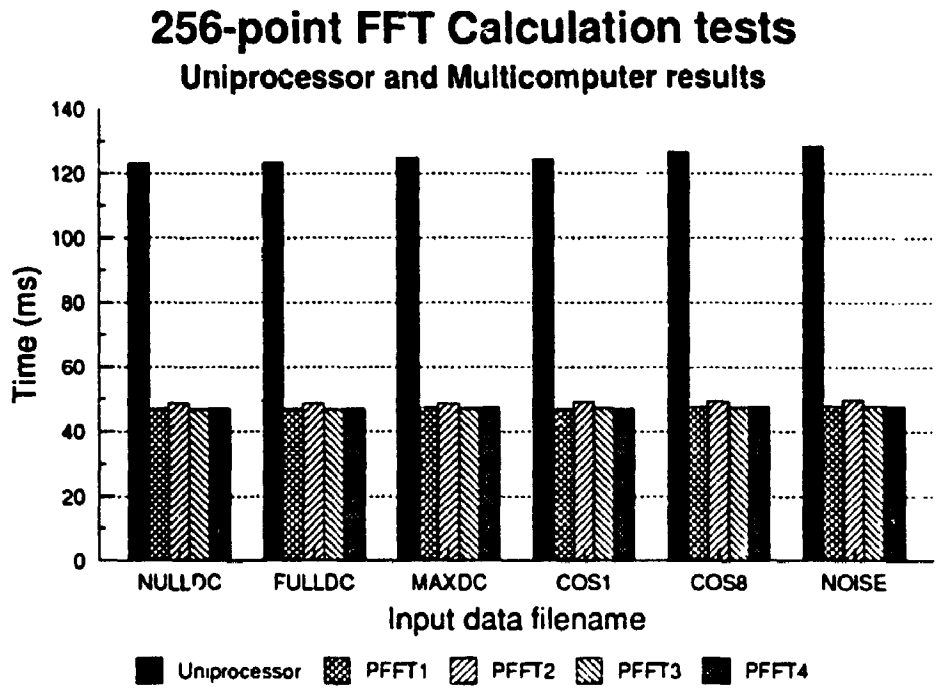
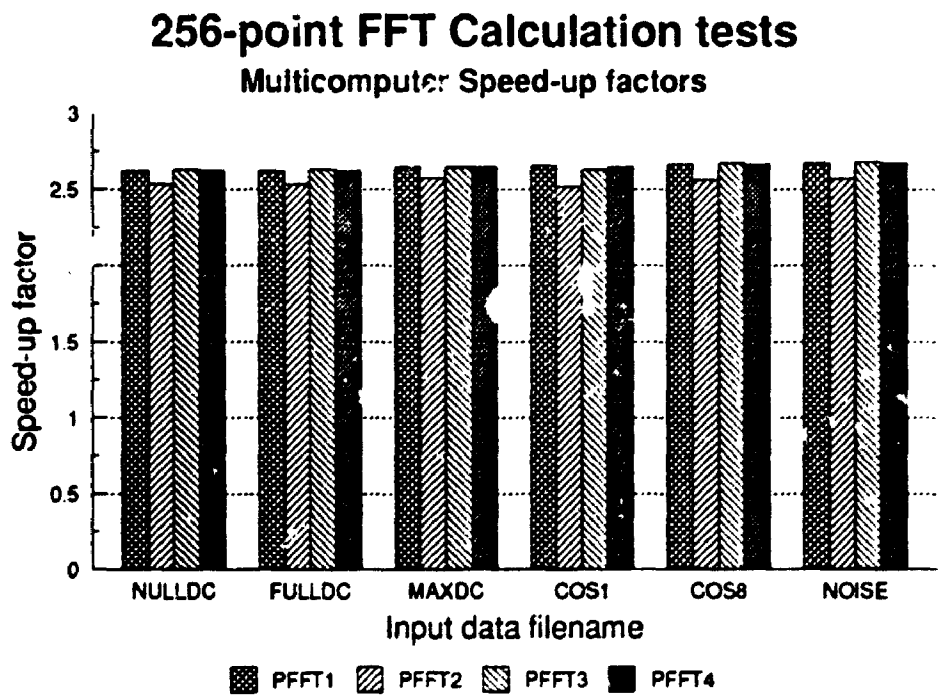


Figure 4.5.2.5 Fast Fourier Transform tests: Multicomputer speed-up factors.



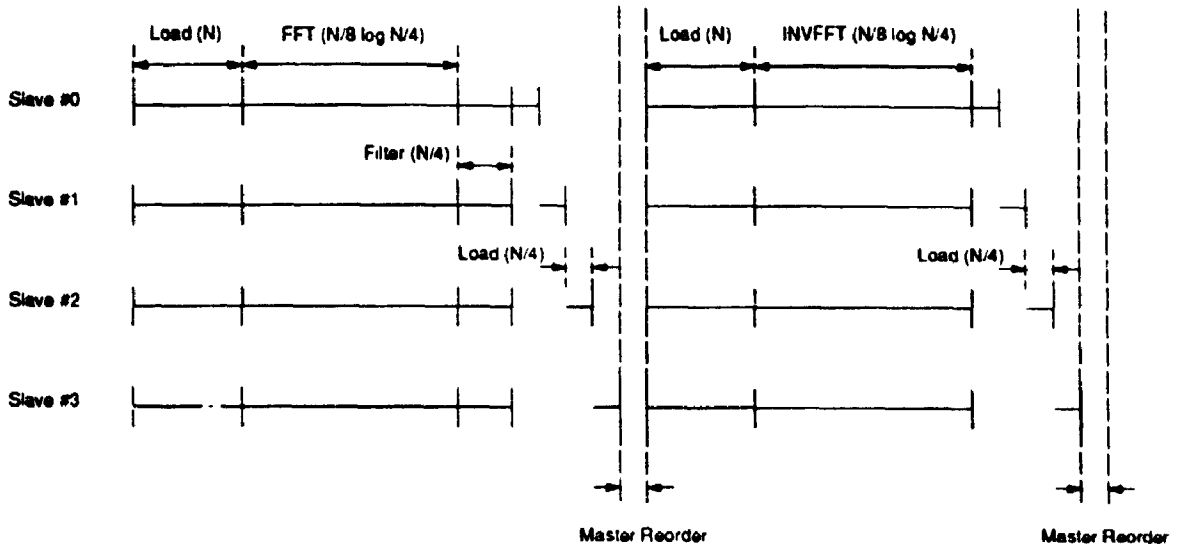
4.5.3 Frequency Domain Filtering Performance Tests

As outlined in section 2.2.3, the frequency domain filtering performance tests were performed using a parallel configuration. The section described two strategies for intermediate data redistribution and collection, hereinafter designated as "slave-to-master-to-slaves" (S→M→S) and "slave-to-slaves" (S→S). Sets of tests adopting each strategy were performed. Due to the data-dependent multiplication time, different input signals and filter types were used to observe minimum and maximum execution times. Two types of input signals were used: a DC signal with 0 amplitude (NULLDC), and a random noise signal (NOISE) (Appendix L). Several filters were used in the experiments, namely: an all-pass filter (ALLPASS; multiplies all components by 1), a no-pass filter (NOPASS; multiplies all components by 0), a low-pass filter, and a high-pass filter. In the low- and high-pass filters, the cut-off frequency was one-half of the Nyquist frequency (LPHLF and HPHLF, respectively). Complex data with 16-bit scaled integer components are used throughout.

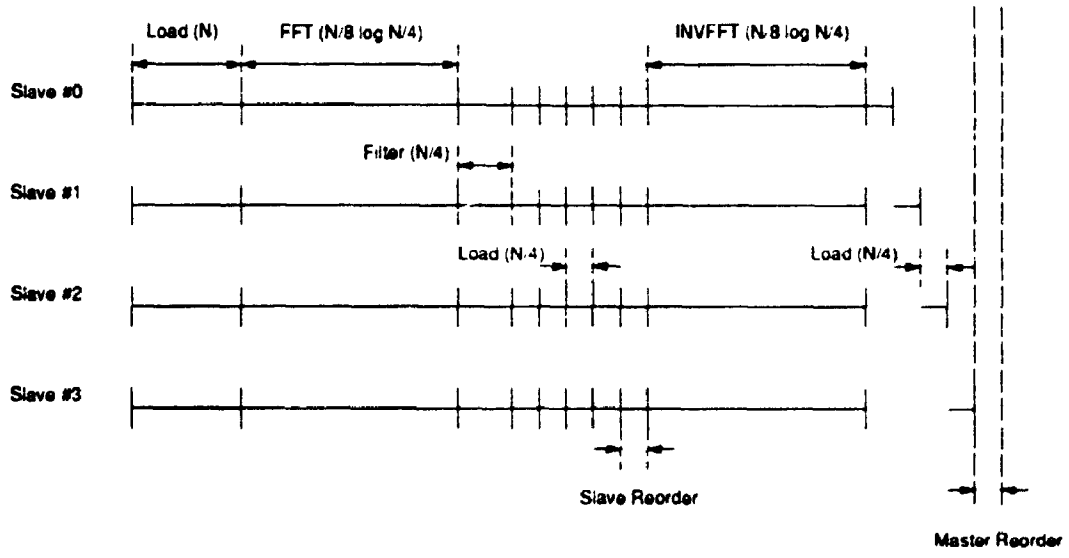
As in the FFT performance tests, four data collection strategies may be used following a transform. For each of the two intermediate data redistribution strategies, all four data collection strategies were tested. In the S→S strategy, it is necessary to wait for all slaves to complete the forward transform before data redistribution may commence. Figure 4.5.3.1 shows a timing diagram of the filtering process using the two strategies. From the diagram, it appears that the S→S strategy is faster than the S→M→S method. However, the S→S method requires more network reconfigurations, and more slaves must be serviced per SRQACK cycle. The increased overhead may reduce the savings provided by the S→S strategy, and therefore, both strategies were tested in the experiments.

The S→M→S tests use programs PFILNOM1, PFILNOM2, PFILNOM3, and PFILNOM4 (Appendix M), where in each case, data collection strategies correspond to those described in the previous section. The programs which employ the S→S strategy are

Figure 4.5.3.1 Timing diagram for comparison of S→S and S→M→S filtering strategies.



(a) Slave-Master-Slave strategy



(b) Slave-Slave strategy

called FSTFL1NM, FSTFL2NM, FSTFL3NM and FSTFL4NM (Appendix L); in each case, the final data collection strategy corresponds to those described in the previous section. To aid performance analysis, the slave execution times for forward FFT, filtering, and inverse

FFT were individually measured. For measurement of overall process times, the slave program code excluded self-timing operations, to decrease measurement overhead. Flowcharts for the frequency domain filtering performance tests (uniprocessor, multicomputer master and slave programs) are shown in figures 4.5.3.2 to 4.5.3.6. Program code and filter data for the tests are presented in Appendix M.

Figure 4.5.3.2 Flowchart for Uniprocessor Frequency Domain Filtering performance test.

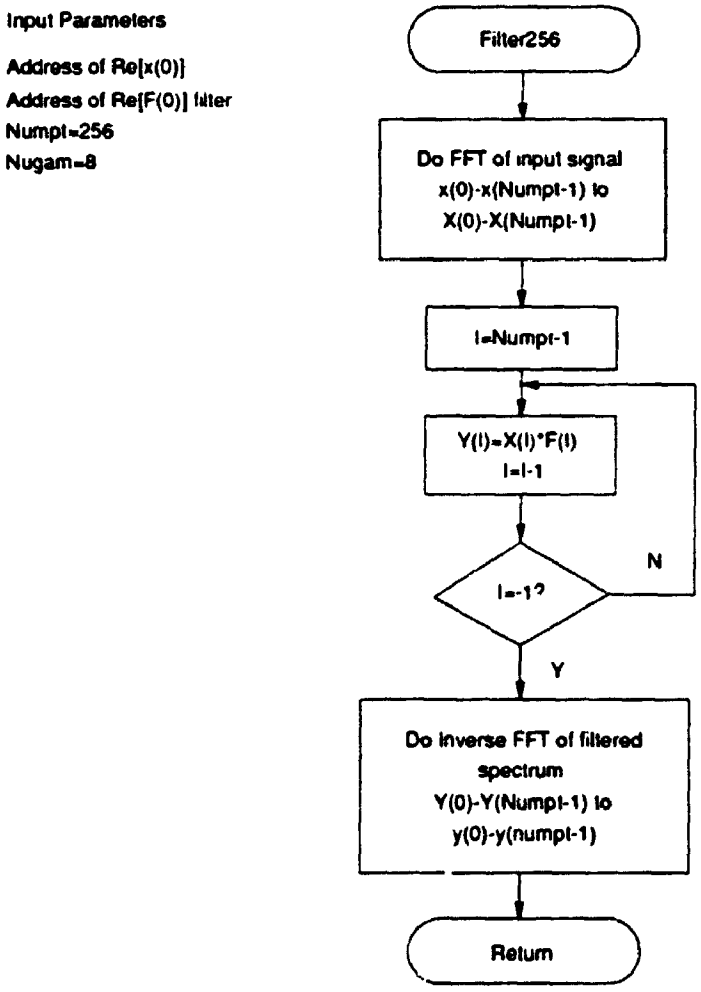


Figure 4.5.3.3 Flowchart for Multicomputer Frequency Domain Filtering performance test: Master program, S→M→S strategy.

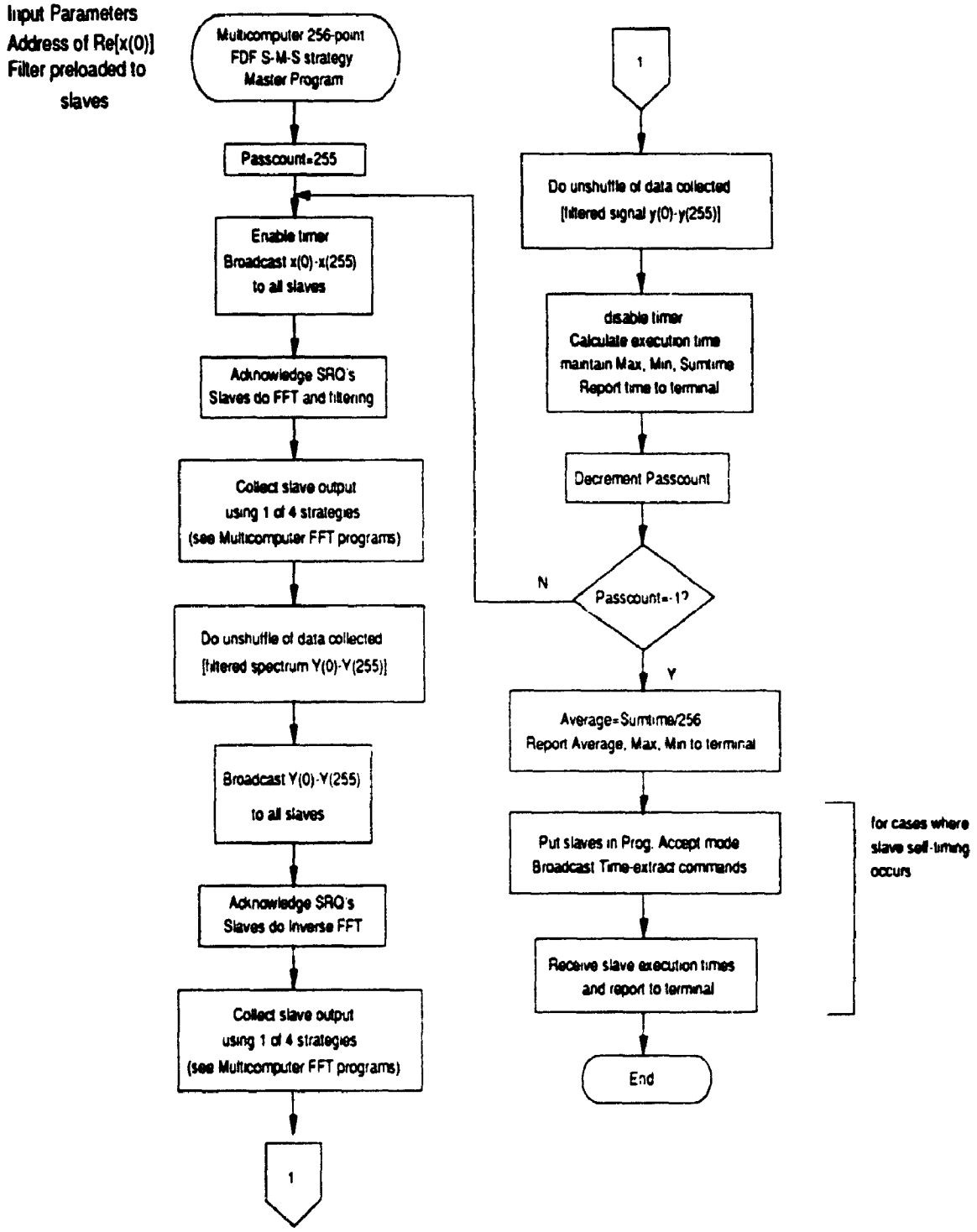


Figure 4.5.3.4 Flowchart for Multicomputer Frequency Domain Filtering performance test: Slave program, S→M→S strategy.

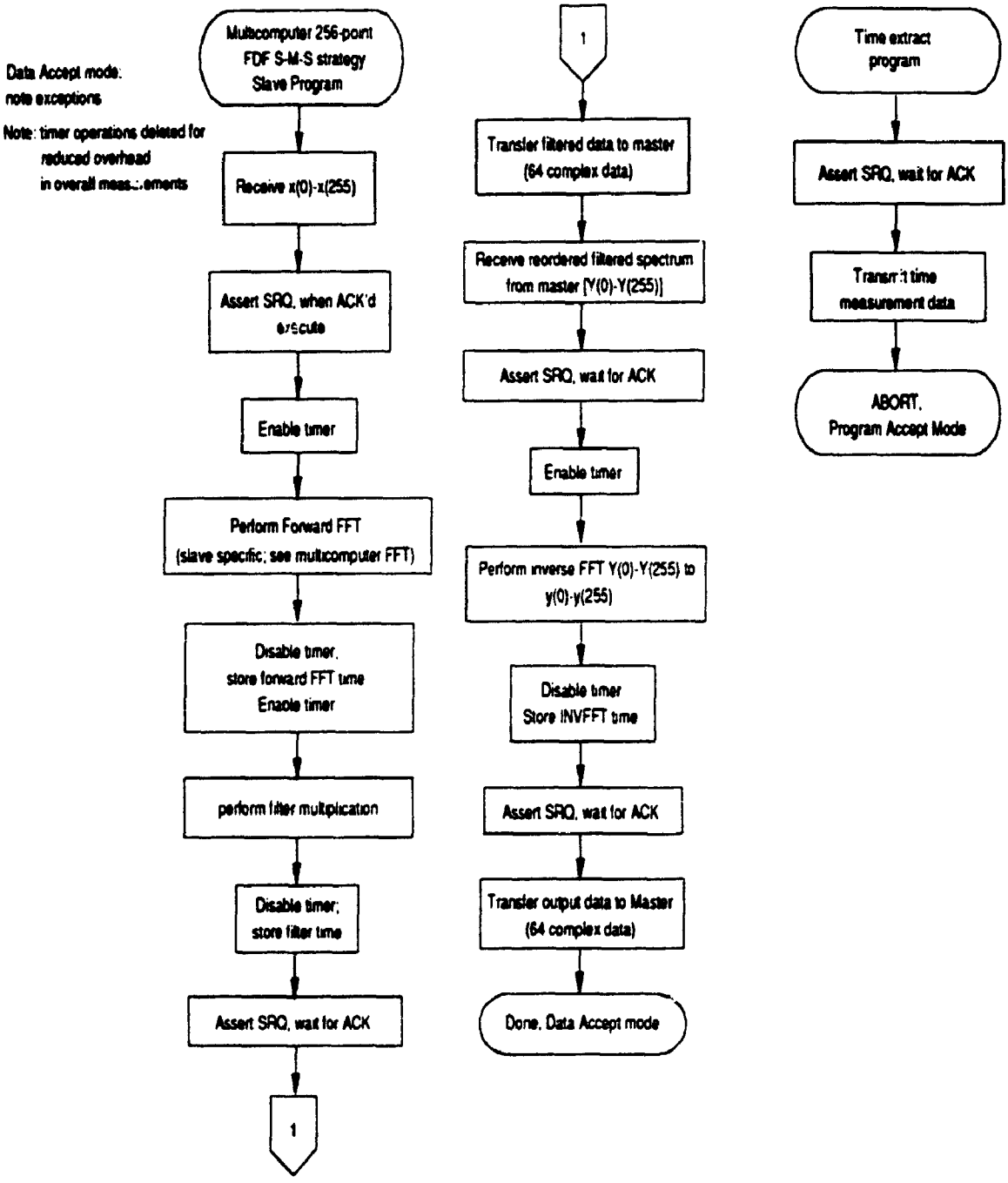


Figure 4.5.3.5 Flowchart for Multicomputer Frequency Domain Filtering performance test: Master program, S→S strategy.

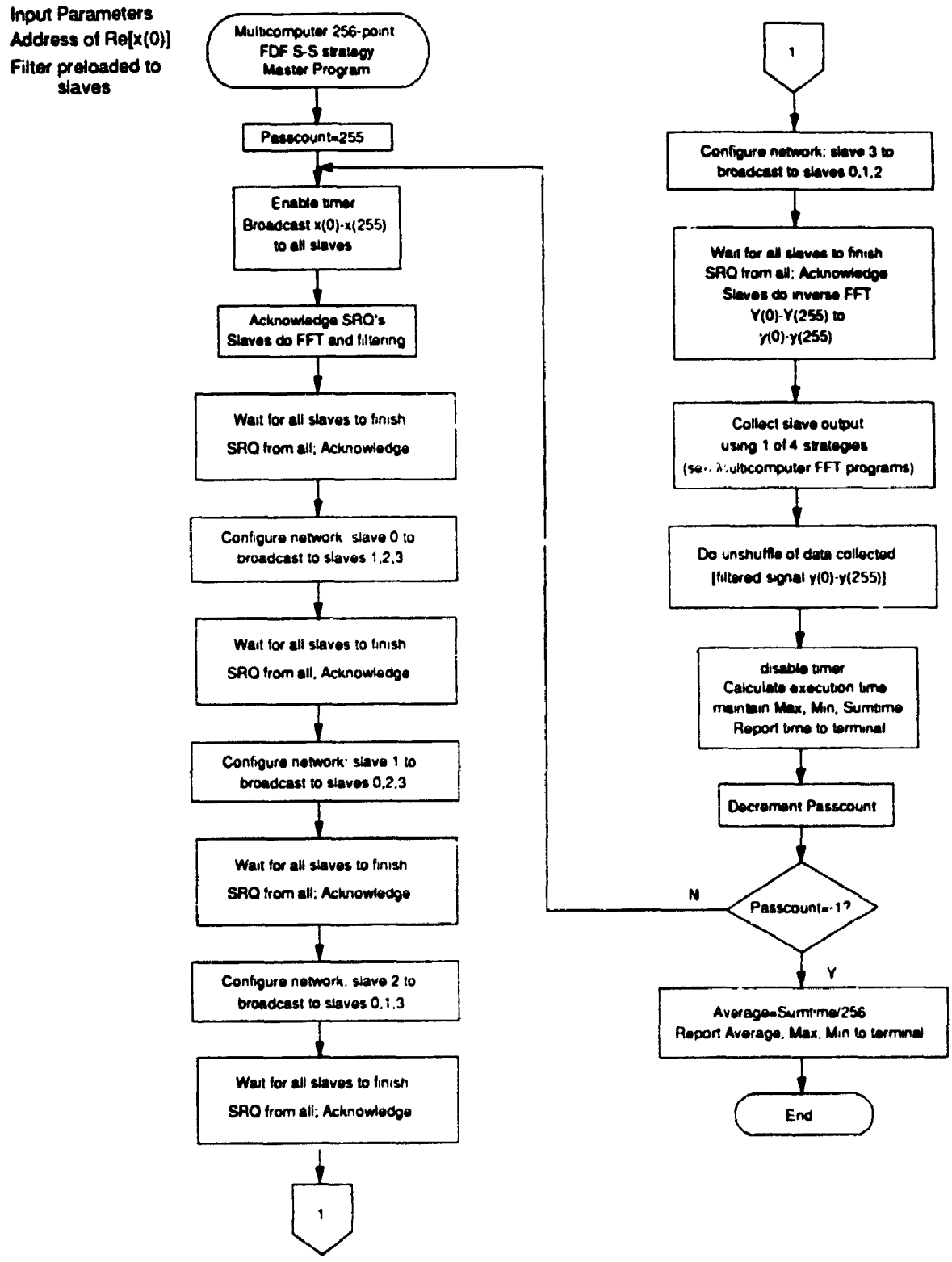
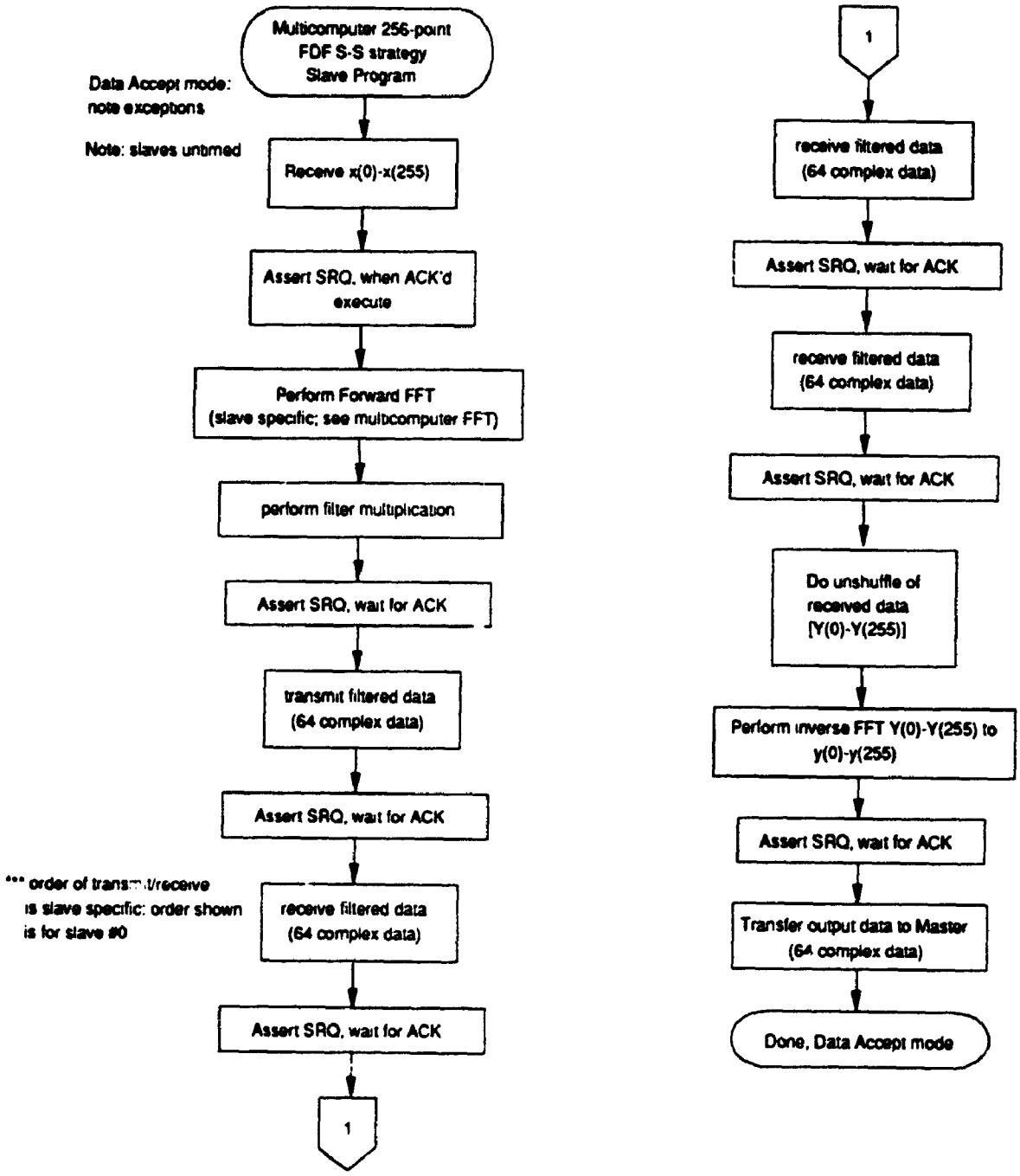


Figure 4.5.3.6 Flowchart for Multicomputer Frequency Domain Filtering performance test: Slave program, S→S strategy.



The program FILTER (Appendix M) is executed prior to the frequency domain filtering test programs. It performs a 4-shuffle of the naturally-ordered filter coefficients,

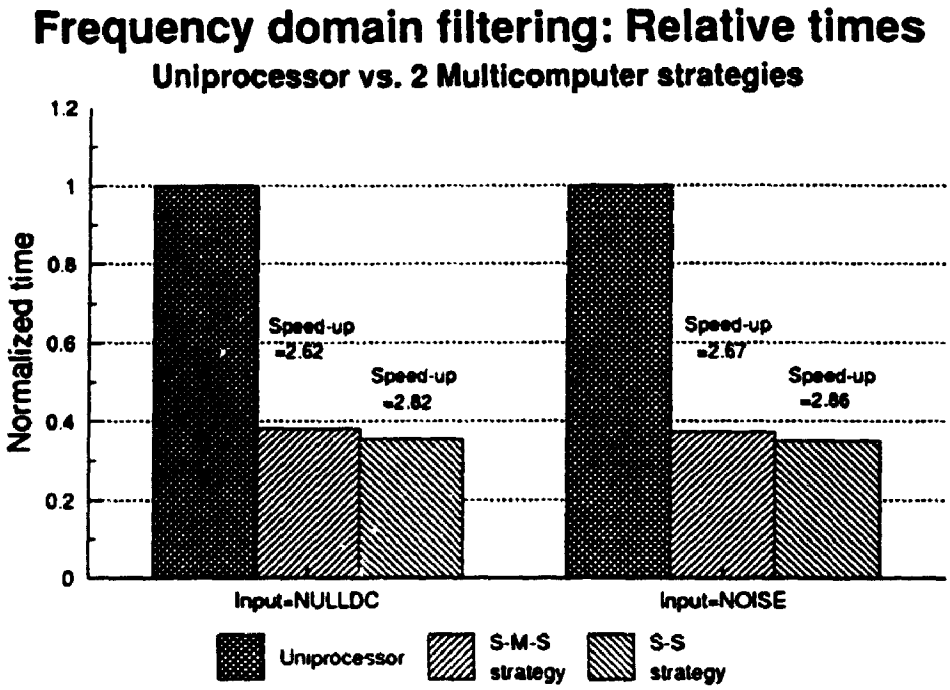
and distributes them among the slave processors, as required by the test programs. Experimental results are summarized in table 4.5.3.1 and figure 4.5.3.7. Appendix A shows additional tabulated results.

Table 4.5.3.1 Frequency Domain Filtering performance tests: summary of Multicomputer speed-up factors.

Input data	Program	Filter			
		ALLPASS	NOPASS	LPHLF	HPHLF
NULLDC	PFILNOM1	2.64	2.64	2.64	2.64
	PFILNOM2	2.55	2.55	2.55	2.55
	PFILNOM3	2.65	2.65	2.65	2.65
	PFILNOM4	2.64	2.64	2.64	2.64
	FSTFL1NM	2.83	2.83	2.83	2.83
	FSTFL2NM	2.77	2.77	2.77	2.77
	FSTFL3NM	2.83	2.83	2.83	2.83
	FSTFL4NM	2.83	2.83	2.83	2.83
NOISE	PFILNOM1	2.70	2.67	2.70	2.70
	PFILNOM2	2.61	2.58	2.60	2.60
	PFILNOM3	2.71	2.68	2.71	2.71
	PFILNOM4	2.70	2.67	2.70	2.70
	FSTFL1NM	2.88	2.85	2.86	2.86
	FSTFL2NM	2.82	2.79	2.82	2.82
	FSTFL3NM	2.89	2.86	2.88	2.88
	FSTFL4NM	2.88	2.85	2.88	2.88

For figure 4.5.3.7, the speed-up factors for a given data collection strategy and all filter types are averaged to provide an average speed-up factor for each strategy.

Figure 4.5.3.7 Frequency Domain Filtering performance tests: summary of average Multicomputer speed-up factors.



4.5.4 Alternating Series Calculation Performance Tests

As outlined in section 2.2.4, the alternating series calculation performance tests were performed using a parallel-pipeline configuration, with two processors in each pipeline. For the tests, calculation of $\pi/4$ is chosen, where:

$$\frac{\pi}{4} = \sum_{n=1}^{\infty} \frac{(-1)^{n-1}}{2n-1} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \ldots$$

The series was selected for its slow convergence to its true value. Tests using a large number of terms may be performed without convergence to 6-decimal places (Microsoft FORTRAN uses a single precision value of 0.7853982 as $\tan^{-1}(1.0) = \pi/4$, while a double precision calculation of the series using 32768 terms, yields a result of 0.7853905). The sub-tasks are naturally divided among the two processors in each parallel branch. The first processor uses the index integer to calculate a floating-point representation of successive

terms in the series ("divide" sub-task); the second processor computes the sum of the incoming terms ("add" sub-task); the master processor executes the final subtraction using the same floating-point addition routine, after collecting results from each adder. The two sub-tasks can be made very closely equal in execution time, which is the preferred condition for optimal pipeline performance. A number of programs were used in the tests, and in all cases, the user specifies the number of terms desired. A uniprocessor program was written (PI4INFO) which monitors the time for each sub-task to complete: data dependencies on execution time can be observed, and slave self-timing is not required in the multicomputer tests. The uniprocessor and multicomputer tests (PI4STST and PI4MTST, respectively) measure overall calculation time for the series. In the multicomputer program, the measurement interval begins when the master broadcasts the desired number of terms to the slaves, and ends following completion of the final subtraction.

In yet another test, the sub-task execution times are artificially increased and more accurately equalized using a *wait* loop in each subroutine (uniprocessor) and slave process (multicomputer). The aim was to observe performance under conditions of very long per-unit execution time, compared to inter-element communication time. The addition sub-task times are increased by 2.397 ms, while the division sub-task times are increased by 2.4225 ms. The uniprocessor and multicomputer programs PI4LSTST and PI4LMTST, respectively, were used for the comparison.

The floating-point representation used in the alternating series tests was the same as that used in the matrix multiplication tests. Similar floating-point addition routines are used throughout, without application-specific optimizations. Flowcharts for the uniprocessor and multicomputer programs (PI4STST and PI4MTST) are shown in figures 4.5.4.1 through 4.5.4.3. Flowcharts for the extended-time tests are not shown. The program code for all alternating series tests is presented in Appendix N. Tabulated results are presented in Appendix A.

Figure 4.5.4.1 Flowchart for Uniprocessor Alternating Series performance test.

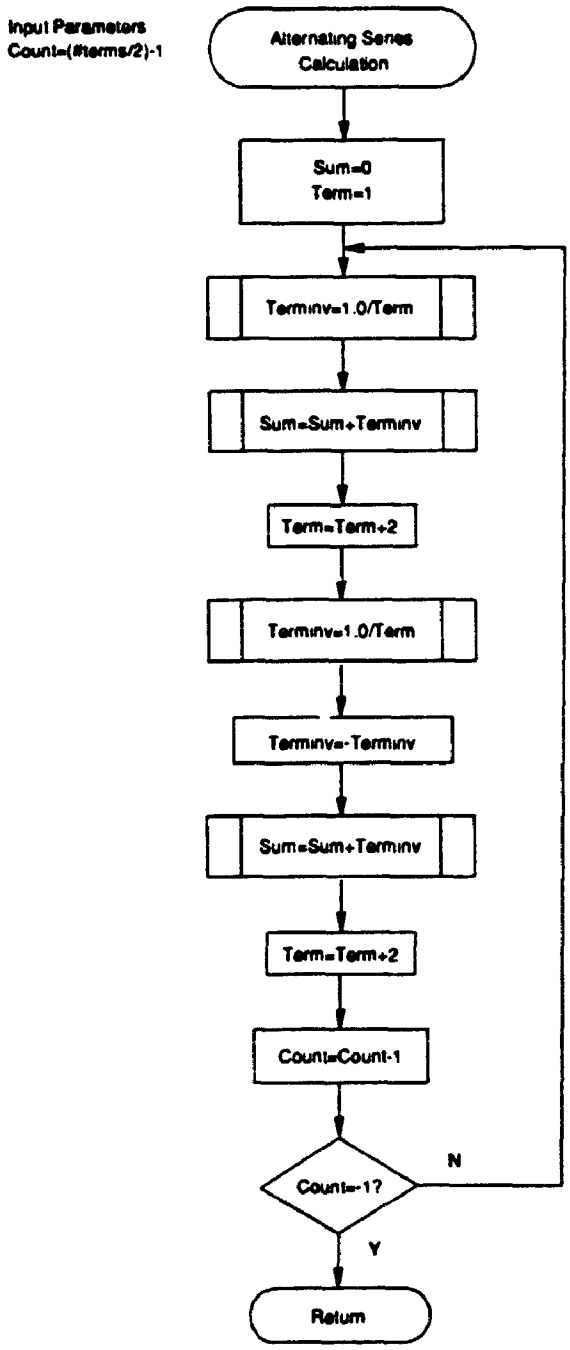


Figure 4.5.4.2 Flowchart for Multicomputer Alternating Series performance test: Master program.

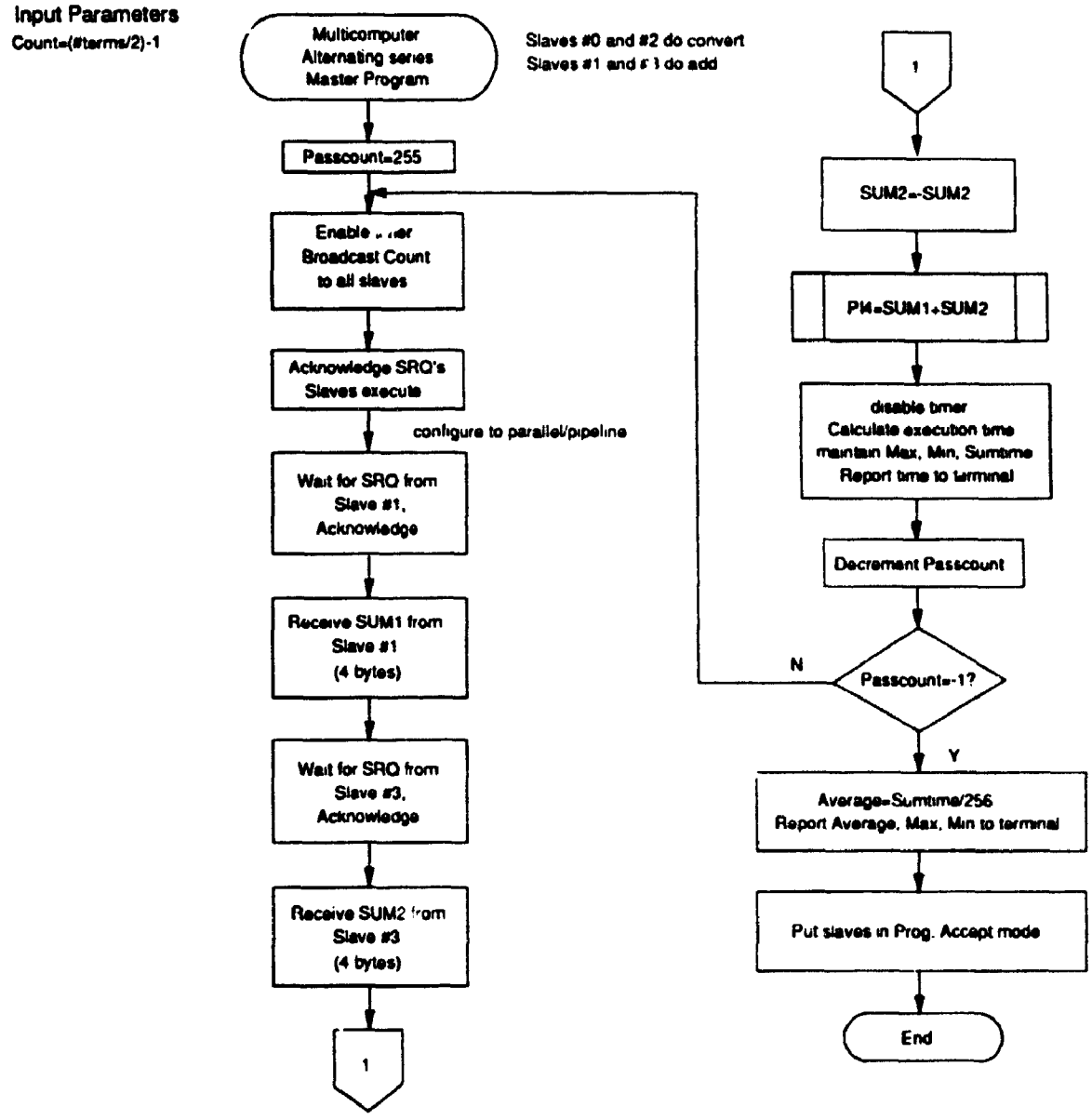
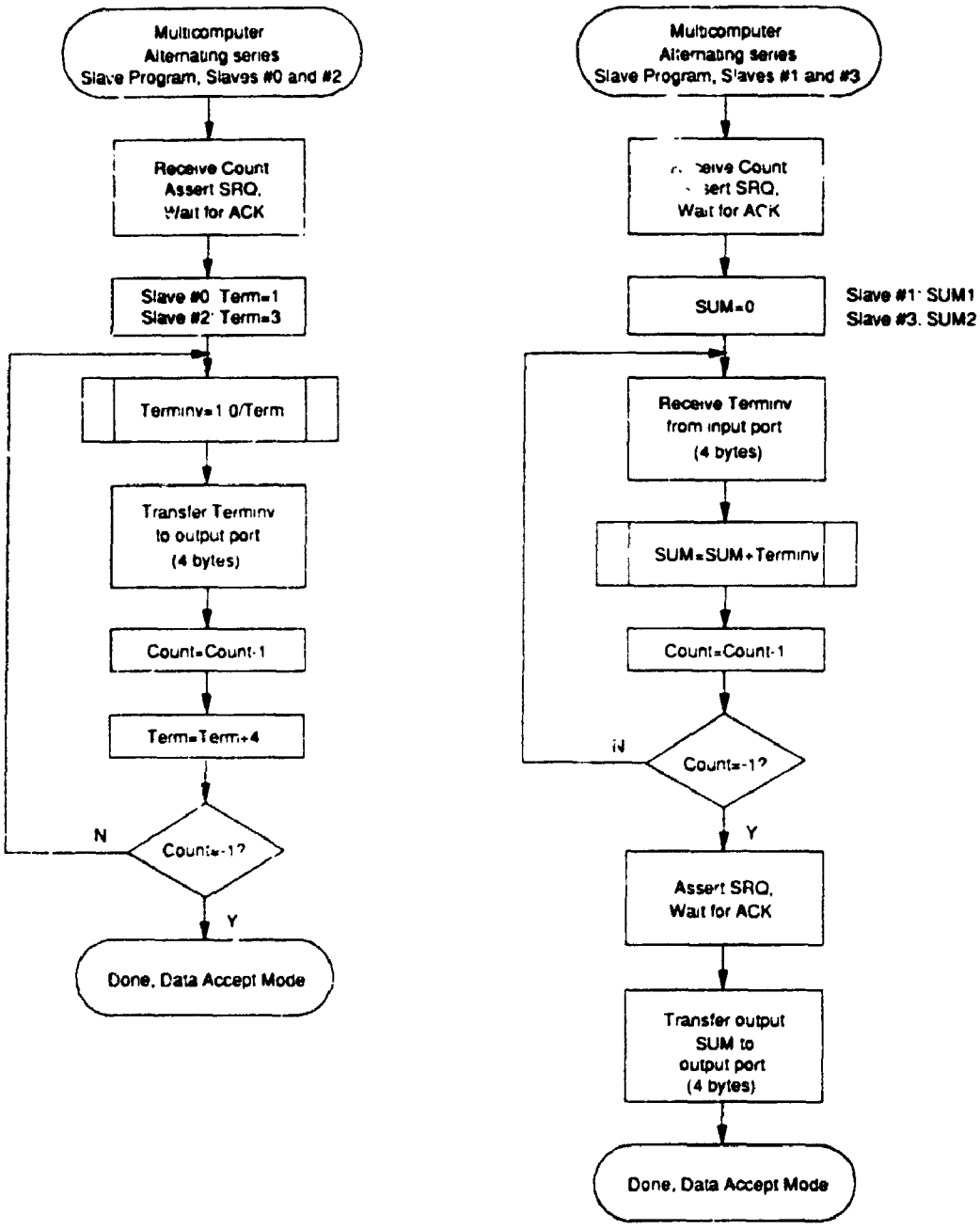


Figure 4.5.4.3 Flowchart for Multicomputer Alternating Series performance test: Slave programs.



Experimental results for the alternating series calculation performance tests are summarized in figures 4.5.4.4 and 4.5.4.5. Appendix N shows complete, tabulated results.

Figure 4.5.4.4 Alternating Series performance tests: Multicomputer speed-up factors.

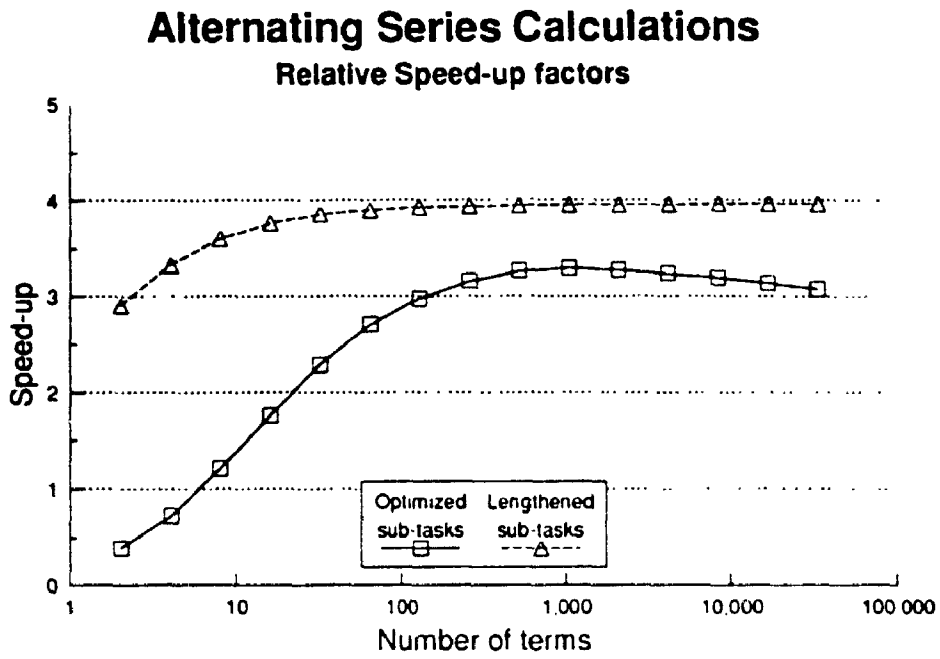
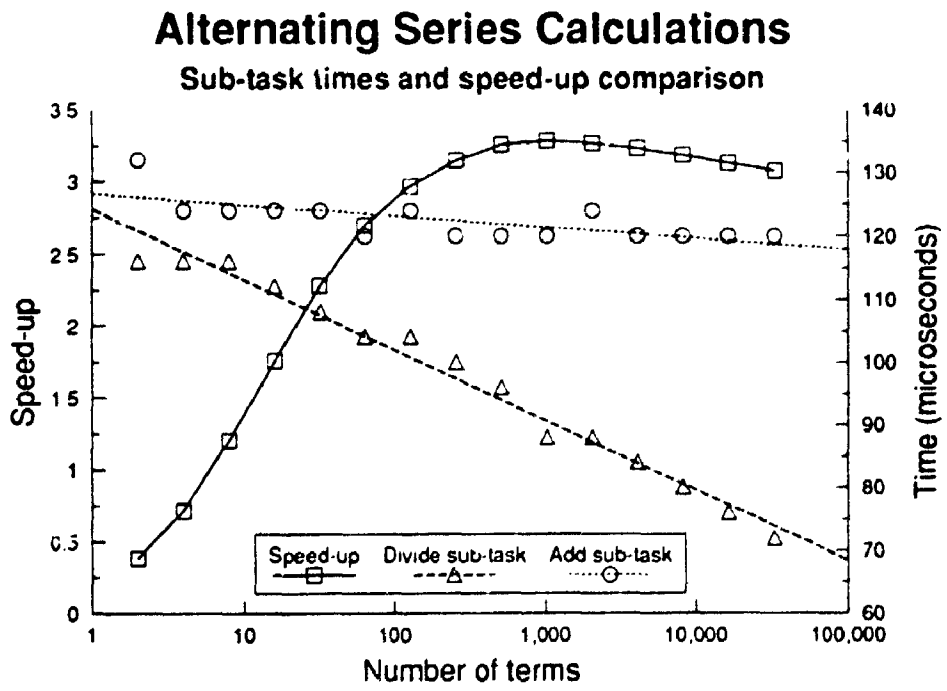


Figure 4.5.4.5 Alternating Series performance tests: Multicomputer speed-up factors (optimized) and sub-task times.



5 Conclusions and Recommendations

This chapter discusses the experimental results and develops a general performance model for the system. The model is used in analyses of the various tests, which are treated separately. The chapter concludes with recommendations for future research.

5.1 Programmable Signal Router Device and Interprocessor Communication Network characteristics

The PSR characteristics were needed prior to finalizing the ICN design. Measured reconfiguration and output enable delay times indicated that an addressed device is configured and ready 90 ns following modification of its configuration control register. For the 8 MHz microprocessors used in the system, reconfiguration and output enable speed is acceptable; no idle time is necessary to stabilize configuration. The observed minimum set-up and hold times for the latching DAV* signals are compatible with the PI/T specifications.

The 75 ns data propagation delay time is satisfactory for the ICN design. DAV* signal propagation delay is considerably shorter, which can cause system errors if compensation is not provided. In a network consisting of numerous PSR devices, the delay-time mismatch can result in a DAV* signal arriving at a destination before its associated data, and invalid states will be latched. The DAV* signal propagates faster since it bypasses the internal latch of the PSR. To compensate for the delay-time mismatch, each DAV* signal in the ICN design is buffered by a non-inverting driver, typically providing a 12 ns delay. Adding a short delay to each DAV* signal, rather than a few long delays at selected points in the system, guarantees that requisite timing characteristics are maintained for all configurations.

The linearity of the graph in figure 4.1.2.1 (ICN Signal Propagation Delay characteristics) confirms that there are negligible delay-time differences among the PSRs; small variations are manifest as deviations from a straight line on the graph. Such variations are common for semiconductor devices, as is a dependence on power supply levels. The

difference observed in rising- and falling-edge delay times is a function of signal loading, and output impedance shifts between the two output states. The effect is cumulative with devices traversed, as confirmed by the diverging trends in the figure.

Since the propagation delay of a signal traversing the network is a function of the number of PSR devices through which it passes, it is apparent that system performance is influenced by the selection of minimum-delay paths between processing elements. The effect of communication path length on overall system performance is addressed in the next section, since the processor element communication rate must also be considered.

5.2 Data Transfer Rate Performance tests

The data transfer rate tests produced the expected trend of linear increase in communication time with block length, as shown in figures 4.3.3 and 4.3.4. Regression analysis of the observed data shows a systematic overhead delay of 59.16 μs and 48.35 μs for output and input transfers, respectively. The dissimilarity is expected, since the TRAP service routine used for block input transfers comprises fewer instructions than the output routine. The extra instructions in the output routine establish synchronization with the receiver. The slopes of the two curves agree within 0.3% of each other; the difference can be attributed to the precision of the measurements.

The difference in execution time for input and output transfers of equal block length complicates efforts to account for, or predict, system performance. Using an average value for data transfers simplifies matters. Figure 4.3.5 shows the outcome of averaging the measured input and output characteristics. If the number of block output and block input transfers differs significantly throughout a program, differences between predictions and actual execution time may be excessive; under such circumstances, the more exact formulation is necessary.

In the experiments, it was shown that a signal traverses a PSR device in a maximum of 50 ns, while a byte is transferred between communicating processors in approximately 7.617 μs. Communications in the system are processor and PI/T device limited. A processor executes an instruction loop for each byte transferred, which entails polling PI/T status, reading or writing the PI/T data registers, modifying indices, maintaining a loop counter, and branching conditionally. The hardware data-transfer handshake is completed in significantly less time than the processor expends to execute the loop. Double buffering of the transfers also enhances the communication performance. Because of the high bandwidth of the ICN compared to a processing cell and its PI/T, an investigation of system performance as a function of communication path-length was not conducted. For systems in which the ICN's bandwidth is close to that of the I/O devices, communication path length can be expected to affect overall performance.

5.3 Processor Synchronization Overhead measurements

The results of the processor synchronization overhead measurements exhibited the anticipated trend: an increase in SRQ acknowledge time with number of slaves serviced. Synchronization time is small in comparison to that required to transfer moderately-sized data blocks, and therefore, processor synchronization overhead will affect overall performance minimally. Programs employing frequent synchronization cycles may suffer performance degradation. It is recommended that unnecessary synchronization cycles are avoided.

The measurements represent average times required to service pending requests. A different synchronization time is observed if the master enters the SRQACK routine before a slave asserts its service request. The asynchronous nature of the protocol and system introduces some variation in synchronization times, and both factors contribute to uncertainties in predictions of system performance.

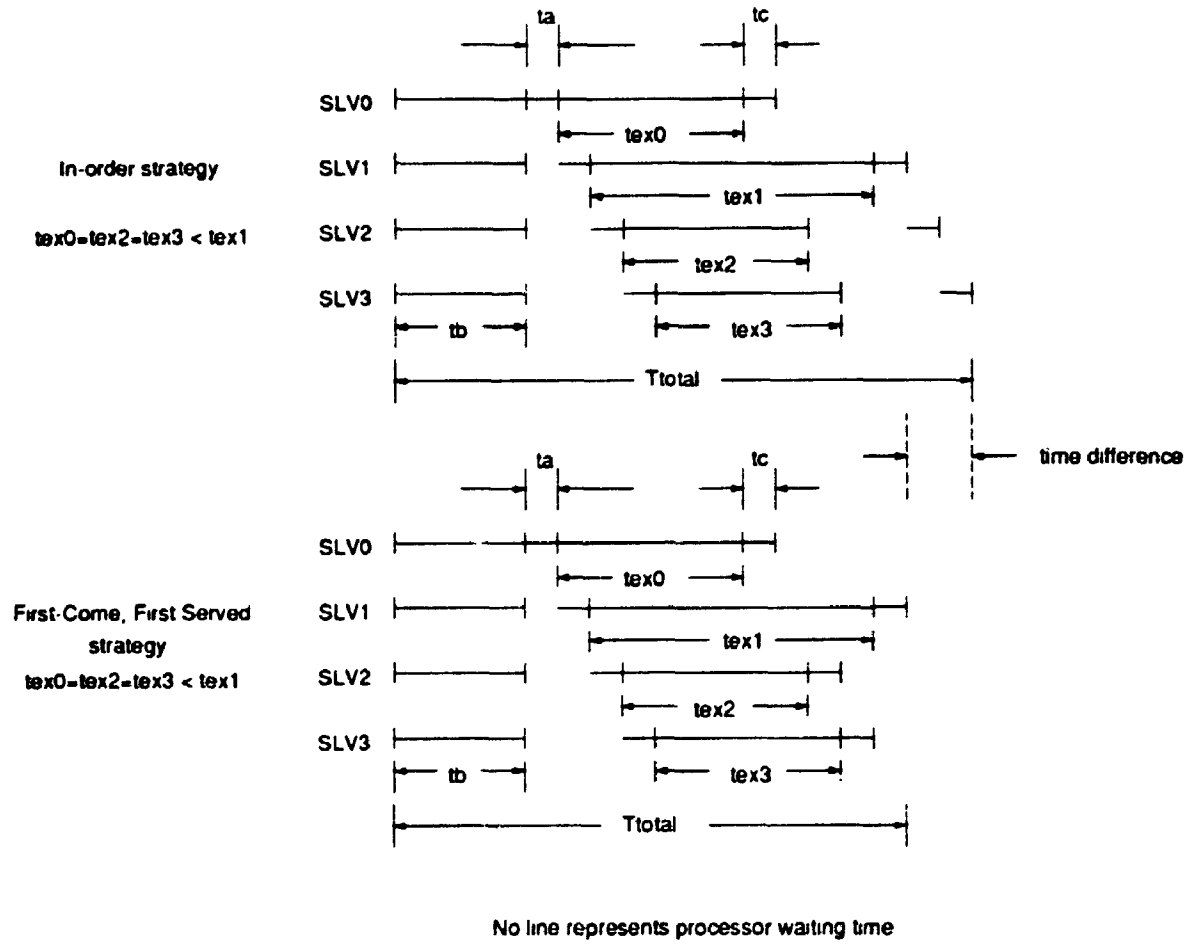
5.4 Matrix Multiplication Performance tests

Both the uniprocessor and multicomputer matrix multiplication tests indicated that execution time was strongly data dependant. The behaviour is due to exceptional processing of null operands in the floating-point multiply and add subroutines, as well as the data-dependent execution time of the microprocessor's multiply instruction. The varying execution time has significance to the multicomputer application, since it has a bearing on the choice of data collection strategy for maximum efficiency.

The multicomputer speed-up factor was consistently lower when null matrices were used as input, compared to other data sets. The slave calculation (or execution) time is significantly lower for the null matrix case, while the data transfer and synchronization overhead are constant for all input arrays. The larger percentage of overall processing time expended by overhead in the null data case accounts for the decreased speed-up factor.

Of the two data collection strategies tested (*In-Order*: MATFPM, and *First-Come, First-Served*: MATFPFC), the *FCFS* method was generally slower, particularly when slave execution times were equal or nearly equal. The *FCFS* protocol requires more instructions to implement. However, in cases where slave execution times differed significantly (particularly by more than the down-load time of 1/4 of the matrix [*A*]), the *FCFS* method was more efficient. Table A.4.4 (Appendix A) shows that, for matrix size 8×8, the execution time for slave #1 is more than 5 ms longer than that of the other slaves. Using the *In-Order* method, the master awaits slave #1 completion, while other slaves have completed and are requesting service. The *FCFS* strategy permits the up-loading of data from completed slaves to the master, while slave #1 continues to compute. The *FCFS* technique is recommended, since the potential benefit outweighs the small penalty paid when slave execution times are equal. Figure 5.4.1 uses the format of figure 4.5.1.3 to compare the two methods when slave execution times are unequal.

Figure 5.4.1 Multicomputer Matrix Multiplication timing: *In-Order* versus *First-Come, First-Served* data collection strategies for unequal slave execution times.



It is appropriate to introduce a model for performance of the multicomputer system. The model incorporates the measured execution time of the slaves, and the system overhead parameters determined in separate experiments. The model is used as an estimator of performance, and may highlight the existence and significance of other overhead thus far undisclosed. The least complex cases (*In-Order* data collection, equal slave times) are used to develop an approximate model for matrix multiplication.

Following the nomenclature of figure 4.5.1.3, the total execution time is given by:

$$T_{total} = t_b + 4t_a + t_{ex} + t_c \tag{5.4.1}$$

where t_{ex} is the measured slave execution time, t_b, t_a, and t_c are data transfer times, which can be modeled as proposed in section 2.3.3 as:

$$t_{tfr} = (byte\ \ count) \times m + b \tag{5.4.2}$$

Parameters m and b were determined experimentally in section 4.3. Additional factors influence T_{total}, which must be considered in the operational model.

Before matrix $[B]$ is broadcast, a header consisting of the destination address and byte count for the data is sent, requiring a transfer of six bytes, and a separate TRAP call. After matrix $[A]$ data is downloaded to each slave, a single-processor service request/acknowledge cycle occurs, initiating slave program execution in that slave. The time required is denoted as $TSYNC_1$. Another $TSYNC_1$ interval precedes the data upload phase from a completed slave. The overlap of upload operations with slave execution is such that only the $TSYNC_1$ interval from slave #3 contributes to the expression for total execution time. The equation for T_{total} is developed to include the additional factors in equations 5.4.3 to 5.4.7.

$$t_b = (6m + b) + 4N^2m + b = (4N^2 + 6)m + 2b \tag{5.4.3}$$

$$t_a = t_c = \frac{4N^2}{4}m + b \tag{5.4.4}$$

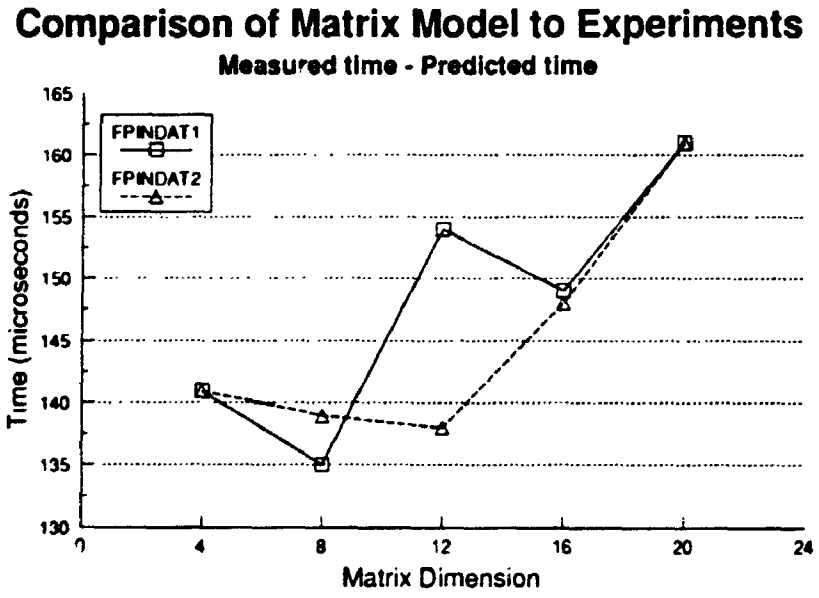
$$T_{total} = t_b + 4t_a + 4TSYNC_1 + t_{ex} + TSYNC_1 + t_c \tag{5.4.5}$$

$$T_{total} = (4N^2 + 6)m + 2b + 4\left(\frac{4N^2}{4}m + b\right) + t_{ex} + \frac{4N^2}{4}m + b + 5TSYNC_1 \tag{5.4.6}$$

$$T_{total} = (9N^2 + 6)m + 7b + 5TSYNC_1 + t_{ex} \qquad (5.4.7)$$

Using the experimentally determined average values for m (7.617 µs/byte), b (53.76 µs), $TSYNC_1$ (88 µs), and the measured t_{ex} (Appendix A) times, the difference between actual multicomputer performance and that predicted by the matrix multiplication model is plotted in figure 5.4.2.

Figure 5.4.2 Difference between actual and calculated Multicomputer execution times using the matrix multiplication model.



The model underestimates system performance, and some overhead remains to be accounted for. The error is low (3.5% in the worst case), however, if the model is to be accepted, the additional delays should be explained. Their sources have been determined to provide more complete insight into system operation.

Between the end-of-transmission of matrix [B] data and the start-of-transmission of matrix [A] data to each slave, the master processor performs a network reconfiguration and a register load instruction. Reconfiguration time is not necessarily equal for each

configuration; it is dependent upon the number of PSR's modified. A total of 40 μs is spent on reconfiguration and register re-load, in the time preceding slave #3 execution. A number of subsequent reconfigurations are concurrent with slave execution. The reconfiguration delay prior to slave #3's upload phase contributes to the calculation of T_{total}. If reconfiguration time for the slave-data upload phase is greater than that on the download phase, slave idle time is introduced. The total reconfiguration delay is approximately 45 μs.

The measured slave time, t_{ex}, does not encompass the time required by the slave's supervisory program (section 3.7.3.6) to initiate processing, nor does it include the time to initialize and arm the slave's timer. A total of 75 μs are consumed in the slave program for these purposes. An additional 10 μs can be attributed to the master's TRAP #7 exception routine, which is used to transmit the initial address and byte count; it includes additional instructions not present in the standard block output routine. About 130 μs of error in the matrix model has been explained. Further disagreement with measurements is small (worst case 0.3%), and apparently correlated with N. The error is likely a consequence of the accuracy in the data transfer rate parameter, m, used in the calculations. Furthermore, the $TSYNC_I$ value is a measurement of average acknowledge time for pending requests; asynchronism in the SRQ/SRQACK protocol gives rise to uncertainties.

The model may be modified to account for other types of supervisory operation, such as *FCFS* servicing, and for instances of unequal slave execution periods. Complications occur when differences in slave execution times are shorter than the data download time, t_a. Both synchronization strategies would service slaves in the same order, since slave completions are separated by the difference in the slave execution time plus t_a. Furthermore, if t_{ex} is less than $3t_a$, the multicomputer time is completely independent of t_{ex}, and is exclusively a function of communication time and the aforementioned overhead factors.

The approximate operational model, although proven to be inexact, provides a reasonable estimate of system performance. The problem-size dependent factors are incorporated, as are the significant system overheads. For the matrix multiplication tests, the model provides from 0.03% to 3.5% accuracy. The quest for further precision is unreasonable. The general procedure for modelling performance will be applied in discussions that follow.

The matrix multiplication speed-up factor curves (figures 4.5.1.10 through 4.5.1.12) may be considered with reference to the operational model. For all tests, the multicomputer speed-up factor approaches four as matrix size increases. Section 2.2.1 predicts that a speed-up factor of four is the highest attainable; for small problem sizes, it is observed to be significantly lower. Regression analysis was performed to determine approximate expressions for uniprocessor and multicomputer slave execution times. Results of the analyses are shown in table 5.4.1.

Table 5.4.1 Expressions for Uniprocessor and Multicomputer Matrix Multiplication Slave execution times.

	Input data set[1]		
	FPINDAT1 (ms)	FPINDAT2 (ms)	FPINDAT3 (ms)
Uniprocessor	$0.124N^{2.98}$	$0.272N^{3.02}$	$0.254N^{3.04}$
Multicomputer	$0.032N^{2.97}$	$0.069N^{3.01}$	$0.054N^{3.10}$

The approximate behaviour of $O(N^3)$ and $O(N^3)/4$ for the two methods, as predicted mathematically in section 2.1.1, is evident. If uniprocessor time is represented as KN^3, and slave execution time, t_{ex}, for the same input data set as $KN^3/4$, the speed-up curves can be

1 See Appendix K for matrix multiplication programs and input data.

described using the operational model for multicomputer matrix multiplication performance.

$$Matrix_{speed\text{-}up} = \frac{KN^3}{\frac{KN^3}{4} + (9N^2 + 6)m + 7b + 5TSYNC_1} \tag{5.4.8}$$

$$\lim_{N \to \infty} Matrix_{speed\text{-}up} = 4 \tag{5.4.9}$$

The $O(N^3)$ terms dominate the speed-up expression for large N. For small N, the system overhead makes up a more significant proportion of overall time, explaining the non-optimal speed-up factors for those cases. The consistently reduced speed-up factor of tests with null input data is rationalized by the lesser K value determined for those tests. The equations apply to cases with equal or nearly equal slave times, and *In-Order* data collection. They can be shown to closely agree with measurements made using the *FCFS* data-collection strategy in particular, but not uncommon, circumstances. It is not within the scope of the thesis to develop performance models for larger systems, since an expanded prototype against which to test the hypotheses has not been implemented. However, a larger system can be described similarly assuming that data transfer rates remain distance independent. More communication and reconfiguration would be necessary, but the processes remain either linearly or quadratically increasing with N, and computation time is $O(N^3)$. Therefore, for large problem sizes, expanded systems are expected to yield speed-up factors in matrix multiplication approaching N.

5.5 Fast Fourier Transform Performance tests

The FFT performance tests exhibited data-dependent execution times, although not to the degree observed in the matrix multiplication tests. The uniprocessor results differ by only 4% between best and worst cases. Individual slave execution times show correspondingly similar data dependence.

The four multicomputer programs showed varying performance (even with identical input data), as a consequence of the data collection strategies utilized. The *time to solution*, when the frequency domain spectra have been calculated, but not reported to the master, is independent of the data collection method. The time to solution measure is important to problems which require FFTs as sub-tasks of a larger problem, such as frequency domain filtering. It is not always necessary to report the output of the slave operations to the master; the locally determined results may be used by subsequent routines executing on their native processor.

Of the four data collection methods tested, best performance was delivered by PFFT3[2], where slave #0 is serviced first, its output data collected, followed by servicing of slaves #1, #2, and #3 with one SRQACK routine call. The lack of twiddle-factor multiplications justifies the assumption that slave #0 will complete first, which was observed in all tests. The first-come, first-served technique (PFFT4) provides nearly equal performance to that of PFFT3; it suffers due to the additional SRQACK routine calls. The program PFFT1 uses a separate SRQACK call for each slave as well, but services slaves in a fixed order. As observed in the matrix multiplication test, the strategy introduces unnecessary idle time in some slaves. The same criticism applies to the program which performed least efficiently, PFFT2, which waits until all slaves have completed before the data collection phase commences.

The speed-up factors observed in the experiments vary with input data set, and show slight deviation among three of the data collection methods. The technique of program PFFT2 is not recommended; the other strategies provide essentially the same speed-up factor, with PFFT3 marginally best.

---

2 See Appendix L for FFT programs and input data.

The speed-up factor determined by considering only slave execution times and uniprocessor time may appear dubious. The interval measured is that required by the slave to perform the sum-and-twiddle operations, as well as their local 64-point FFT. Theoretical analysis in section 2.2.2 yields a maximum speed-up of four, however, values in excess of that are observed. The incongruity underscores the shortcomings of performance prediction based solely on operation count of one particular type, in this case multiplication. The technique makes one of two assumptions:

- only those operations that dominate the overall execution time are taken into account, or
- the number of auxiliary instructions in a considered procedure is the same.

The latter circumstance is more practical, and applicable. For the sum-and-twiddle factor multiplication phase of slave execution, fewer auxiliary instructions per multiplication are present than in the 64-point FFT calculation phase.

The execution time of 64- and 256-point FFT programs can be discussed and compared in terms of the number of multiplications, since for these programs, the number of auxiliary instructions per multiply is nearly equal. Following the discussion of section 2.2.2, the number of multiplications for an FFT algorithm is $O(N \log_2 N)$. Performance of algorithms with N and $N/4$-points may be compared:

$$Ideal\ FFT_{speed\text{-}up} = \frac{KN \log_2(N)}{\frac{KN}{4} \log_2\left(\frac{N}{4}\right)} \qquad (5.5.1)$$

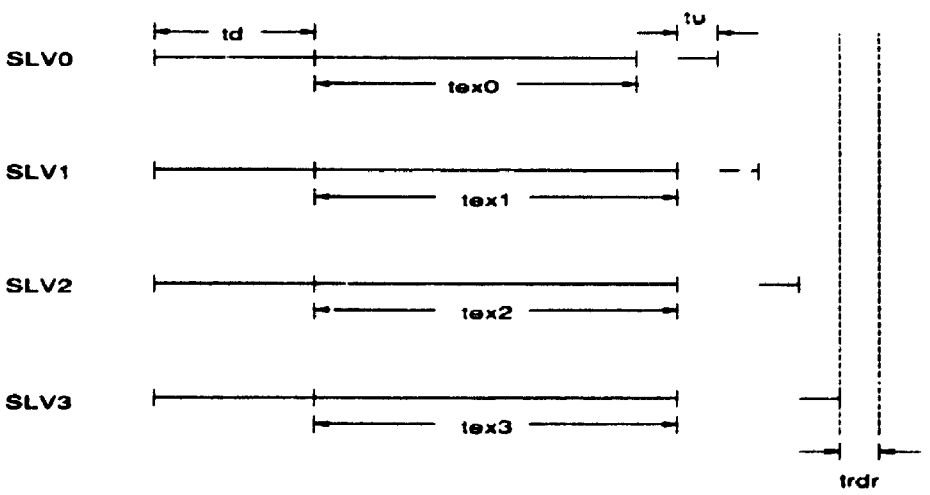
For increasing N, the function approaches four from above; for N=256 as in the experiments, $Ideal\ FFT_{speed\text{-}up}$ is 5.33. Measurements agree closely with this value; differences are attributable to:

- routine initialization instructions prior to the main computation loops,

- some degree of data dependence in the multiply operations, and
- the O(N) process of data shuffling prior to the main calculation loops.

The decreasing trend toward the speed-up factor of four affects the rate at which overall multicomputer performance approaches its maximum with increasing N. Experiments with larger data sets were not performed, but a model developed in a manner similar to that described in the previous section, may be adopted. An average value for K is derived from the measured FFT calculation times. The data collection strategy of program PFFT2 is used for the model, to minimize complexity[3]. A timing diagram for the process is shown in figure 5.5.1

Figure 5.5.1 Timing diagram for Multicomputer FFT calculation, data collection strategy as in program PFFT2.



An expression for total execution time is developed, following the procedure of the previous section.

$$T_{total} = t_d + TSYNC_4 + \max(t_{ex}) + TSYNC_4 + 4t_u + t_{rdr} \tag{5.5.2}$$

$$t_d = (4N + 6)m + b \tag{5.5.3}$$

---

3 Models for the other strategies must consider whether slave #0 completes its data up-load phase prior to or following the completion of calculations by the other slaves.

$$t_u = \frac{4N}{4}m + b \tag{5.5.4}$$

$$TSYNC_4 = 172\mu s \tag{5.5.5}$$

$$t_{rdr} = O(N) \approx RN = 2.269ms \qquad R = 8.86\mu s \tag{5.5.6}$$

$$t_{ex} = t_{sum\&twiddle} + t_{FFT64} \tag{5.5.7}$$

$$t_{sum\&twiddle} = O(N) \approx SN = 6.824ms \qquad S = 26.7\mu s \tag{5.5.8}$$

$$t_{FFT64} = O\left(\frac{N}{4}\log_2\left(\frac{N}{4}\right)\right) \approx K\left(\frac{N}{4}\log_2\left(\frac{N}{4}\right)\right) \qquad K = 61.34\mu s \tag{5.5.9}$$

$$T_{total} = (8N + 6)m + 5b + 2TSYNC_4 + (S + R)N + K\left(\frac{N}{4}\log_2\left(\frac{N}{4}\right)\right) \tag{5.5.10}$$

The parameter R is derived from the master reorder time per data element; S uses the maximum sum-and-twiddle time observed (Appendix A).

Substitution of the parameters m, b, $TSYNC_4$, S, R, K, and N give estimates of T_{total} which agree with measurements within 1.8%. Reconfiguration and other master overhead has been neglected. The FFT model may be applied to predict system performance in calculating FFTs with various N.

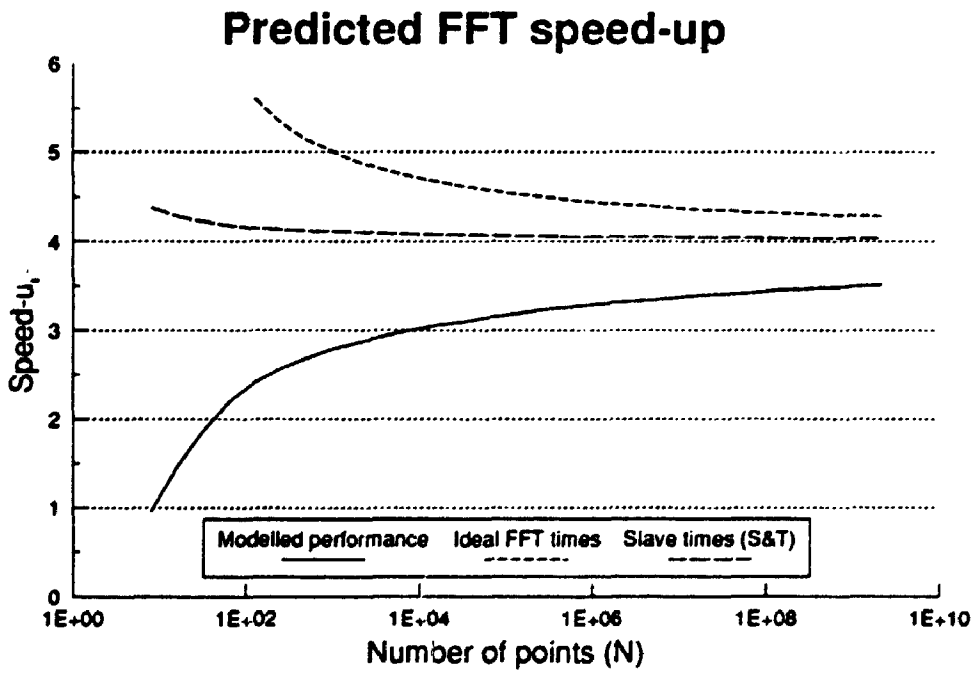
The $O(N \log_2 N)$ behaviour of the FFT algorithm ensures that the speed-up factor for the multicomputer problem will increase with increasing problem size, since other system overheads are $O(N)$ or constant. The rate at which it approaches the maximum will be slower than that observed for matrix multiplication, where the slave time increased by a factor of N greater than other operations. The speed-up factor expression for the multicomputer FFT problem is:

$$FFT_{speed\text{-}up} = \frac{KN\log_2(N)}{(8N+6)m + 5b + 2TSYNC_4 + (S+R)N + K\left(\frac{N}{4}\log_2\left(\frac{N}{4}\right)\right)} \tag{5.5.11}$$

$$\lim_{N \to \infty} \quad FFT_{speed\text{-}up} \quad = \quad 4 \tag{5.5.12}$$

Figure 5.5.2 plots FFT model speed-up for various N. Also shown is a speed-up curve considering only slave execution time (including the sum-and-twiddle step), and an "ideal" curve based on the *Ideal* $FFT_{speed\text{-}up}$ function.

Figure 5.5.2 Predicted FFT speed-up vs. problem size.



The diagram shows that the multicomputer performance can be expected to increase toward a speed-up of four, although very slowly. Problem sizes of the magnitudes suggested on the graph cannot be undertaken with the RMCS in its present form; additional memory, well beyond the addressing capabilities of the MC68008 microprocessor, is required. Small increases in speed-up from that predicted by the model may be achieved by utilizing alternate data collection strategies, since the model is based upon the least efficient method.

5.6 Frequency Domain Filtering Performance tests

The frequency domain filtering performance tests exhibited execution time data dependence similar to the previous experiments. Not only did input data affect results, but also filter type. The variations, as before, are a consequence of the multiply instruction execution time.

Execution times for the inverse FFT were consistently shorter than those of the forward FFT calculations, in both uniprocessor and multicomputer applications. The inverse FFT algorithm does not include a $1/N$ scaling factor present in the forward FFT. The fewer instructions account for the data-independent proportion of the 5.6 to 10.4% difference in measured times for the two algorithms.

Of the four slave-to-master data collection strategies tested, the performance ranking corresponds to that observed in the FFT tests. The ranking remains consistent for both of the intermediate data distribution strategies, S→M→S and S→S. The result is expected, since the intermediate data distribution strategy for the S→S method is the same for all tests.

The tests proved that the S→S intermediate data distribution method was superior to the S→M→S strategy. If SRQACK and reconfiguration time were too great, the extra overhead required to implement the S→S procedure could cause inferior performance. The overhead has been shown to be sufficiently small so that such behaviour does not occur.

The strategy used to synchronize slaves for the intermediate data distribution phase of the S→S method is the one proven to be least efficient in the FFT tests. It is unavoidable, however, since all slaves must be ready to receive the broadcast data. The method introduces idle time in the slaves, especially slave #0. The data collection strategy selected for the final data upload phase can exploit differences in execution time of the slaves, as shown in the FFT performance tests.

The speed-up factors of 2.5 to 2.7, exhibited by programs using the S→M→S method, correspond closely with those observed in the FFT performance tests with similar data collection techniques. The S→S procedure shows higher speed up (about 2.8 to 2.9). The smaller variation in the S→S results are a consequence of using a common intermediate slave synchronization technique. Since the performance of the test is in many ways similar to the FFT performance, detailed modelling and analysis follows similar reasoning, and results show similar trends with problem size changes.

In the frequency domain filtering tests, the reconfigurability of the system was utilized differently than in the matrix multiplication and FFT experiments. For the latter tests, the system was cast as parallel; the configuration was modified only to permit slave-to-master communications. In the filtering tests, the overall process was undertaken as a parallel problem, however, use of additional communications topologies improved performance.

5.7 Alternating Series Calculation Performance tests

The alternating series performance tests yielded results which confirmed the RMCS's suitability to serial problems, while it also demonstrated high performance in a non-conventional system configuration. The experiments also focus attention on issues of task equalization, minimization of communications, and effects of varying sub-task execution time on speed-up factor.

The observed speed-up factor approaches the maximum value of four as the number of terms increases, as is expected for the configuration (figure 4.5.4.4). In the tests with optimized sub-tasks, maximum speed-up falls short of four, mainly due to communication overhead. The time to transfer four bytes is approximately one-fifth to one-quarter of the processing element computation time per term, which is a significant proportion of overall time. The unequal execution times of the processing elements for each sub-task also reduce performance. The tests with lengthened and equalized sub-tasks displayed speed-up approaching the theoretical maximum. The overhead incurred by communication of positive and negative term results, and the final subtraction operation by the master, had negligible effect on speed-up factor, except in cases with small problem size.

The varying execution times of the serially connected elements are responsible for the observed peak in the speed-up factor curve for the optimized experiment. It is well-known that the throughput of a pipeline is limited by the slowest element. It is also generally assumed that the throughput of each element remains constant regardless of input data. Data dependence of execution time is evident in the measurements, and most extreme in the division process. Not only is the divide instruction data-dependent, but so is the alignment of the denominator term, which predominates. In the tests, speed-up factor is affected by the mismatch in element throughput; the disparity is aggravated as problem size increases. The outcome is a maximum in the speed-up factor plot, indicating a problem size for which the system and program are optimal (figure 4.5.4.5). In the tests, only two elements per serial leg are employed; longer pipelines with data-dependent behaviour per unit may give rise to additional local maxima in the speed-up characteristic. In tests with lengthened sub-tasks, variation in execution times for the elements is relatively reduced, and peaking of the speed-up curve is not evident. The behaviour of the speed-up factor curve may be clarified by examining an analytical expression for the calculation times for both the uniprocessor and multicomputer processes.

Let the divide and add sub-task times be represented by:

$$D = Wi + X \tag{5.7.1}$$

$$A = Yi + Z \tag{5.7.2}$$

Uniprocessor computation time for a problem with N terms is given by:

$$U = \sum_{i=0}^{N-1} (Wi + X) + (Yi + Z) \tag{5.7.3}$$

Let F represent the time to communicate the maximum number of terms to the slaves, collect terms and compute the final sum by the master processor, and C represent the fixed communication time to transfer intermediate results to subsequent pipeline elements. The multicomputer time can be approximated by:

$$M = F + \frac{1}{2} \sum_{i=0}^{N-1} \max(Wi + X + C, Yi + Z + C) \tag{5.7.4}$$

$$Speed\text{-}up = \frac{U}{M} = \frac{\sum_{i=0}^{N-1} (Wi + X) + (Yi + Z)}{F + \frac{1}{2} \sum_{i=0}^{N-1} \max(Wi + X + C, Yi + Z + C)} \tag{5.7.5}$$

The peak in the speed-up factor characteristic does not appear at exactly the point where the two sub-task times differ by C, due to the integration effect in the expression for both uniprocessor and multicomputer execution times.

It may be concluded that, as in any serial design, optimum performance is achieved when the time for each process step is the same, and communication time is minimized. In the RMCS, the functional units are of higher complexity than those encountered in more common, special purpose pipelines. It follows that inherently serial problems can be subdivided among pipeline stages without restrictions imposed by limited capability of available elements. The system affords considerable flexibility, and equalized sub-task

assignment may be pursued. Sub-task division must also consider the communications required for various strategies, and a balance must be achieved for optimal system performance.

5.8 Conclusions

This section presents general conclusions related to the stated objectives of the work.

5.8.1 Performance

The RMCS has been shown to provide an improvement in performance over a similar-class uniprocessor system. The degree of improvement depends upon a number of factors, most importantly, the amount of communication required between processing elements throughout a program. The experiments have shown that issues such as network reconfiguration time and slave processor synchronization are practically insignificant in comparison. With the present communications limitations, for some problems, speed-up approaching the theoretical limits can be achieved. The improvement is not only observed for a single, particular configuration, but for a number of distinctly different topologies. The system has displayed a degree of flexibility that contributed to overall performance. For some problems, modest performance increases were observed. Perfect speed-up in all tasks is an unrealistic goal, since it is rare that a general-purpose research system would always match performance of the dedicated system. However, the main objective was to offer performance enhancement in a range of applications, not just in a limited class of problems. This objective has been achieved.

At first it would seem that the design philosophy of single, unidirectional communications ports in each processing element would hinder performance. The trend in high-performance computing is to increase connectivity and communications bandwidth by utilizing numerous, bidirectional I/O ports for each processor. Although the strategy is sound, it is best exploited when the processing element can perform multiple

communications simultaneously. Most microprocessors are restricted in this sense, and thus if the reconfiguration speed for the network is sufficient, performance of the single port design can be comparable to that with multiple ports. Adoption of single, unidirectional channels provides a decrease in hardware and software complexity with minimal performance penalty, due to the flexibility of the ICN, and efficiency of its control.

5.8.2 Reconfigurability, Cellularity, and System Expansion

The four-slave (single-cell) prototype system offers complete reconfigurability; any desired interconnection pattern may be established among the slave processing elements involving a single input and output. Paths form several outputs to a single slave input must be multiplexed. Analytical proof of interconnection topologies was not within the scope of the presented thesis; however, the network properties of five input connections and five output connections and tree-like topology between them provides the desired characteristic of full reconfigurability. It was determined that reconfiguration of the network is efficient, and that the process of reconfiguration may overlap with slave operations. Reconfiguration time is of concern in expanded systems, since it necessarily increases with the number of PSR devices present. In the experiments, it was infrequent that all configuration control registers were modified during a reconfiguration cycle; only selected PSR's were altered. Accordingly, reconfiguration overhead is expected to remain insignificant even for large systems.

The cellularity component of the design objective has been met; expansion of the system may be accomplished by replication of the four-slave cell. The cellular nature of the design makes the system suitable for integration to larger scales. The selection of interconnection sites for cells influences the number of configurations available, since a PSR in either of the broadcast modes effectively blocks its remaining input path. It is recommended that subsequent placement of cells follows a trend toward a square system topology. The strategy provides a greater abundance of alternate routes whereby signals

may circumvent blocked paths. The prototype system comprises twenty PSRs, and communications paths selected for the experiments were of insufficient length to cause performance degradation due to signal propagation delays. In expanded systems, demand for interconnections with longer path lengths is probable. Minimum path length selection would be a more prominent issue with respect to performance.

The slave processors are cells from both hardware and software perspectives. Distinctions between slaves do not hinge on their hardware nor software, and logical designations are not topologically constrained. The SRQbus interface hardware embodies the single hardware element (an address decoder) which distinguishes slaves from one another. If the SRQbus interface is physically located on the master's NCC (as it is in the prototype system), the slave processor cells may be implemented identically. An alternative design replicates the SRQbus interface circuitry as part of each slave cell, with only the address decoder unique to each slave. The slaves would conveniently connect to the SRQbus using a backplane. In such a design, however, slave hardware assigns to each a fixed designation, and places intrinsic restrictions on system size. System expansion is limited by the number of address signals available. The homogeneous slave design adopted for the prototype system was preferred, since the restrictions are not cast at the slave's hardware level.

5.8.3 Suitability of the System for General Computations

The experiments conducted in the study attest to the reconfigurable multicomputer system's suitability for general purpose computations. The programs used to characterize performance were specifically chosen to be diverse. Both parallel configurations and a parallel-pipeline configuration were tested in the experiments, and a substantial increase in performance was observed. Testing with every possible configuration and every application program is certainly impractical. Since the reconfiguration and synchronization strategies of the system were shown to be efficient, it can be reasonably expected that algorithms

which naturally map onto a given topology will be performed efficiently by the RMCS. Problems which would benefit from reconfiguration at intermediate stages of computation are also suitable. The constituent processing elements were selected for their general purpose nature, and therefore slave cor. putations are not limited to a particular class of problem. Furthermore, the homogeneity of processing elements does not limit nor favour particular operations to be performed on particular slaves. The control strategy of the system is sufficiently flexible to permit any sequence of operations to be regulated by the master in an efficient, coherent manner.

The reconfigurability of the system provides an efficient method by which an application may be tuned for increased performance. Machines with fixed architectures are best suited to particular classes of problems. Problems or sub-tasks of larger problems arise for which the special-purpose machine is ill-suited, but must be solved nonetheless. Assignment of those tasks to the supervisory unit, a single cell, or a standard uniprocessor are options. The solutions are subsequently computed in the usual, serial manner. The RMCS may assume the optimum configuration for the problem at hand, and for each specialized sub-task, if advantageous. System overhead cannot be ignored, and it must not offset the benefits offered by redistributing the problem. A broad range of computations may be undertaken, each completed faster than a comparable uniprocessor, without the need for a specialized machine for each type of problem encountered. The RMCS may not equal the speed of a special-purpose system in executing its intended class of problem, but due to its flexibility, it is capable of efficiently solving a broader range of tasks. The RMCS will surpass special-purpose systems in problem classes beyond their scope.

The RMCS offers an additional degree of freedom for optimizing a task not available in fixed architectures. A static system requires the programmer to partition a problem

according to the machine structure. The RMCS empowers the programmer to subdivide the problem according to the optimum computation strategy, and to specify the machine configurations best suited to accomplish it.

5.8.4 Suitability of the System to Other Applications

The thesis has focussed on the RMCS's performance in arithmetically intensive applications, issues of computational speed-up factor and system control overhead, and the system's flexibility to afford optimal interprocessor configurations. Use of the system as a general purpose, high performance machine with variable degrees of parallelism has been stressed. The immediate goal has been to determine the advantages and practical limitations of the prototype system, providing insight into methods for increasing the performance of future systems with similar architecture. The RMCS design is useful in applications where raw computational horsepower is not the most attractive asset. This section briefly discusses a few of these applications.

The RMCS is suitable as a generalized test platform for investigating optimum algorithms and structures for proposed, special-purpose machines. The system may be used as a hardware emulator to experiment with task partitioning, scheduling, and control strategies. Complications and bottlenecks may be identified. Experiments will influence and validate the special-purpose system design at lower cost than individual prototype construction.

The autonomous nature of the processing cells may be exploited to perform many, unrelated functions simultaneously. The experiments conducted for the study emphasize applications where the slave processors cooperate in fulfilling a single, common objective. Tasks may be assigned to execute entirely on a slave processor, while other, separate tasks execute on others. Problems may be assigned to a subset of the available slaves to be solved

in some parallel fashion, while other sets of slaves collaborate to solve different problems. The master's role of single-task overseer would be expanded to multiple-task overseer and task scheduler.

5.9 Recommendations and Future Research

The thesis has been successful in meeting its objective of investigating the suitability of the RMCS architecture in achieving high performance in broad range of applications. The factors which limit performance were identified. Future research may focus on:

- increasing performance of the four-slave cell,
- increased performance and connectivity by replicated expansion of the system,
- increased system integration,
- development of applications for which the system is suitable, and
- studying system programming methodologies.

This section briefly discusses each, with attention to primary issues and their implications.

5.9.1 Increasing System Performance

A fundamental method for enhancing performance of the RMCS would adopt a higher performance microprocessor as the elementary processing cell. The results achieved so far may be approximately scaled. However, available I/O devices generally do not parallel the processing performance increase provided by current state-of-the-art processors; the mismatch complicates direct scaling of the results. Speed-up is hindered more significantly by communications speed, and applications with more coarse-grained parallelism are favoured. The range of applications for which the system is cost-effective (especially expanded systems with many cells) may be diminished. The systems would be utilized most often in modes where distinct tasks are assigned to each processor, or to a few processors. For computation intensive tasks, the combination of processing power and flexibility would

provide exceptional performance. The study has shown that the most significant obstacle to overall speed-up is interprocessor data transfer rate. If the system is to maintain its applicability to a very broad range of problems, any increase in processing element performance must be accompanied by correspondingly augmented communications bandwidth.

Since the microprocessor chosen offers high performance at low cost, methods for increasing performance which do not replace it merit consideration. The speed-up of the four-slave cell will be improved by decreasing communication overhead, and a direct memory access controller (DMAC) would serve this purpose. Available devices can sustain block data transfers between a slave's memory and PI/T at over ten times the rate attained under processor control. The DMAC however, introduces increased hardware and software complexity, making large-scale integration of the system more difficult, and programming less straightforward. For the experiments performed, speed-up of tasks would achieve maximum levels for problems of smaller size; problems with even finer-grained parallelism would benefit. The effect of data propagation delays while traversing the network would become more significant at shorter path lengths, as well.

A design objective for the prototype design was to permit system expansion by replication, and it was satisfied. System performance may therefore be enhanced through replicated expansion. The strategy does not modify the hardware or software complexity of the individual processing cell. Not only would more processors be available for computations, but more complex and varied interconnection patterns would be available. Practical considerations such as physical size and cable lengths may be limiting factors. Enhanced integration of the four-slave cell would be essential. Current VLSI technology does not support the entire cell (as defined in the prototype) on one chip, but reduction to a single board per cell is possible. Fabrication technology would support production of single-chip, reconfigurable cells comprising processors of lesser complexity and capacity.

Multi-cell systems would be inexpensive, and their implementation simplified. Although the functional capabilities of the processing cells would be reduced, the flexibility and potentially higher communications rates would yield exceptional performance in systems with even a modest number of simple cells.

5.9.2 Other Applications for the System

The performance tests conducted in the study utilized parallel and parallel-pipelined configurations. A test using a single, serial configuration was not conducted. A suitable application which is related to the previous tests is calculation of two-dimensional fast Fourier transforms. Each element in the pipeline computes one or a set of vectors in the individual row and column transforms. A complete discussion of pipelined, two-dimensional FFT computation is in Appendix O. Calculation of 2-d FFTs may also be performed in a parallel mode, where each processor is assigned a fraction of the row and column transforms. The two methods would compete in terms of performance, communications time and memory requirements being the decisive factors in determining superiority.

The frequency domain filtering tests exploited the reconfigurability feature of the system to a higher degree than the others; modifications to interprocessor connections were made for purposes beyond slave-to-master data reporting. The filtering tests clearly demonstrated the advantages provided by the system flexibility. The system is uniquely suited to problems which would benefit from intermediate reconfigurations. Network alterations can be accomplished efficiently, and introduce little overhead. The problems sought are those which have various processing phases, of which some phases would be most suitably performed in a parallel mode, the others in pipelined or alternate modes. Future research will target algorithms and applications where performance using hybrid modes exceeds that attainable with a single mode.

Aside from the applications outlined herein, the RMCS is suitable for carrying out various forms of fault-tolerant computations, either through processor redundancy or dynamic replacement of faulty processors. System software modifications are necessary to fulfill fault-tolerant control strategies.

5.9.3 Software Issues: Programmability and Control

The RMCS implements a coherent system control strategy with a combination of specialized hardware protocols and the software constructs of *processing states*. The control is sufficiently flexible to permit reliable and efficient task synchronization for all configurations and circumstances. The definition of slave processing states, transition regulations, and TRAP-service routines, combine to provide a cogent software framework for applications programming.

Programs used throughout the study were written using MC68000 assembly language. Although the astute programmer can often produce more highly optimized code than the best compilers, development time is longer than when high-level languages are used. A first step in easing the programming task is to adopt a language such as C, with compilers modified to implement code for I/O and system control which corresponds to that required by the RMCS monitor system.

High-level language support is only a first step, however, and does not address more fundamental issues of system programmability. Programming of a single computer is considered a formidable task, no matter what language is used. Evidence can be seen in the industry: there is a significant time lag between a hardware advance and the availability of programs and operating systems which adequately exploit the new capabilities. The RMCS prototype demands that a programmer write code for not only the master cell, but compatible programs for four other processors! In some applications, slave programs are identical,

however, frequent reconfiguration results in higher programming complexity. Expanded systems will present magnified difficulties. The solution lies in an advanced or "intelligent" compiler.

The "intelligent" compiler would generate the appropriate slave and master-control code automatically. The desired processor configurations for specific sections of code would be specified by the programmer, using a suitable nomenclature. The compiler would generate code to download programs and data to the slaves, monitor program execution, and redistribute data in preparation for subsequent operations. Further evolution of the "intelligent" compiler would yield the "brilliant" compiler, wherein the user does not specify system configurations. The compiler itself determines optimum topologies based on program properties and communications delays inherent to competing strategies. The "brilliant" compiler is a profoundly challenging problem related to artificial intelligence. Versions with "incomplete insight" (only certain, common constructs are recognized and optimized) however, could prove to be useful in decreasing program development time for the RMCS, and in affording access to such a system to a broader range of users.

5.10 Concluding Comments

The reconfigurable multicomputer system embodies concepts which can have an influence on the future direction of high-performance computing. It represents a departure from the trend of special-purpose, fixed-topology systems. Its flexibility provides high performance by permitting exploitation of varying degrees of parallelism in a broad range of applications. The prototype system is only one implementation of the general concept: details such as processing cell performance and complexity may be modified to realize systems with varying levels of performance, size, applicability, and cost. The thesis has established the practical viability of the concept, demonstrated its advantages, as well as underscored its shortcomings, and suggested methods for their alleviation.

Appendix A: Tabulated Experimental Results

A.1 Hardware System test results

Table A.1.1 PSR device characteristics.

Parameter	Description	min (ns)	max (ns)
t_{dpdr}, t_{dpdf}	Data signal propagation delay time, rising and falling	-	75
t_{dvpd}	DAV* signal propagation delay time	-	65
t_{dcpd}	ACC* signal propagation delay time	-	50
t_{su}	Set-up time, data valid to DAV* asserted	40	-
t_{hlda}	Hold time, DAV* asserted to data invalid	40	-
t_{rfd}	Reconfiguration delay time	-	90
t_{oed}	Output enable delay time	-	50

Interprocessor Communication Network Characteristics

Table A.1.2 Signal Propagation Delay times.

Number of PSR device traversals	Delay time falling signal (ns)	Delay time rising signal (ns)
1	63.0	71.0
2	122.5	136.0
3	181.5	199.0
4	247.0	270.0
5	321.5	350.5
6	401.0	422.0
7	467.0	487.0
8	526.5	549.5
9	584.5	614.0
10	647.5	685.0
11	724.0	760.0
12	799.0	835.0
13	880.0	914.0
14	957.0	990.5
15	1020.5	1060.0
16	1090.0	1138.0

A.2 Data Transfer Rate Performance test results

Table A.2.1 Data Transfer Rate performance.

Block Length (bytes)	Transfer Mode		
	master→one slave time (μs)	master→all slaves time(μs)	slave→master time(μs)
8	120	120	112
16	180	180	172
32	304	304	292
64	548	548	532
128	1036	1036	1020
256	2012	2012	1996
512	3964	3964	3944

A.3 Processor Synchronization Time test results

Table A.3.1 Processor Minimum Synchronization Time test results.

Number of slaves acknowledged	Synchronization time (μs)
1	88
2	112
3	140
4	172

A.4 Matrix Multiplication Performance test results

Table A.4.1 Uniprocessor Matrix Multiplication tests: measured execution times.

Matrix Dimension	Execution times for various input data (ms)		
	FPINDAT1	FPINDAT2	FPINDAT3
4 × 4	7.760	17.848	17.916
8 × 8	60.876	145.564	132.064
12 × 12	204.296	494.924	478.196
16 × 16	482.944	1174.464	1188.636
20 × 20	941.756	2300.656	2347.068

Table A.4.2 Multicomputer Matrix Multiplication tests: measured Slave execution times, input data: FPINDAT1.

Matrix Dimension	Slave execution times (ms)			
	Slave #0	Slave #1	Slave #2	Slave #3
4 × 4	1.968	1.968	1.968	1.968
8 × 8	15.248	15.248	15.248	15.248
12 × 12	51.084	51.084	51.084	51.084
16 × 16	120.764	120.764	120.764	120.764
20 × 20	235.468	235.468	235.468	235.468

Table A.4.3 Multicomputer Matrix Multiplication tests: measured Slave execution times, input data: FPINDAT2.

Matrix Dimension	Slave execution times (ms)			
	Slave #0	Slave #1	Slave #2	Slave #3
4×4	4.492	4.492	4.492	4.492
8×8	36.420	36.420	36.420	36.420
12×12	123.760	123.760	123.760	123.760
16×16	293.644	293.644	293.644	293.644
20×20	575.192	575.192	575.192	575.192

Table A.4.4 Multicomputer Matrix Multiplication tests: measured Slave execution times, input data: FPINDAT3.

Matrix Dimension	Slave execution times (ms)			
	Slave #0	Slave #1	Slave #2	Slave #3
4×4	4.040	4.040	4.040	4.040
8×8	31.382	36.668	31.382	31.382
12×12	120.968	120.896	118.040	122.840
16×16	297.188	294.664	296.372	297.188
20×20	585.200	584.672	590.488	586.828

Table A.4.5 Multicomputer Matrix Multiplication tests: overall execution times; program MATFPM, input: FPINDAT1.

Matrix Dimension	Execution times (ms)		
	Average	Maximum	Minimum
4×4	4.068	4.072	4.068
8×8	20.632	20.664	20.628
12×12	61.972	62.012	61.968
16×16	139.324	139.356	139.320
20×20	263.912	263.944	263.904

Table A.4.6 Multicomputer Matrix Multiplication tests: overall execution times; program MATFPM, input: FPINDAT2.

Matrix Dimension	Execution times (ms)		
	Average	Maximum	Minimum
4×4	6.592	6.600	6.592
8×8	41.808	41.836	41.800
12×12	134.632	134.668	134.628
16×16	312.204	312.224	312.200
20×20	603.636	603.668	603.628

Table A.4.7 Multicomputer Matrix Multiplication tests: overall execution times; program MATFPM, input: FPINDAT3.

Matrix Dimension	Execution times (ms)		
	Average	Maximum	Minimum
4×4	6.144	6.144	6.144
8×8	42.064	42.076	42.032
12×12	133.712	133.744	133.704
16×16	315.756	315.768	315.748
20×20	618.940	618.952	618.908

Table A.4.8 Multicomputer Matrix Multiplication tests: overall execution times; program MATFPFC, input: FPINDAT1.

Matrix Dimension	Execution times (ms)		
	Average	Maximum	Minimum
4×4	4.172	4.184	4.124
8×8	20.700	20.736	20.680
12×12	62.036	62.056	62.024
16×16	139.408	139.416	139.404
20×20	263.968	263.976	263.960

Table A.4.9 Multicomputer Matrix Multiplication tests: overall execution times; program MATFPFC, input: FPINDAT2.

Matrix Dimension	Execution times (ms)		
	Average	Maximum	Minimum
4×4	6.636	6.636	6.636
8×8	41.896	41.896	41.896
12×12	134.668	134.676	134.664
16×16	312.308	312.316	312.304
20×20	603.664	603.672	603.664

Table A.4.10 Multicomputer Matrix Multiplication tests: overall execution times; program MATFPFC, input: FPINDAT3.

Matrix Dimension	Execution times (ms)		
	Average	Maximum	Minimum
4×4	6.200	6.200	6.200
8×8	40.784	40.784	40.784
12×12	133.760	133.768	133.760
16×16	315.776	315.784	315.772
20×20	618.544	618.552	618.540

Table A.4.11 Multicomputer Matrix Multiplication tests: speed-up factors.

Matrix Dimension	Speed-up factors					
	FPINDAT1		FPINDAT2		FPINDAT3	
	MATFPM	MATFPFC	MATFPM	MATFPFC	MATFPM	MATFPFC
4×4	1.908	1.860	2.708	2.690	2.916	2.890
8×8	2.951	2.941	3.482	3.474	3.140	3.238
12×12	3.297	3.293	3.676	3.675	3.576	3.575
16×16	3.466	3.464	3.762	3.761	3.764	3.764
20×20	3.568	3.568	3.811	3.811	3.792	3.795

A.5 Fast Fourier Transform Performance test results

Table A.5.1 Fast Fourier Transform tests: Uniprocessor execution times.

Input data filename	Execution time (ms)
NULLDC	123.044
FULLDC	123.300
MAXDC	124.956
COS1	124.320
COS8	126.356
NOISE	128.188

Table A.5.2 Fast Fourier Transform tests: Multicomputer execution times.

Input filename		Test program (ms)			
		PFFT1	PFFT2	PFFT3	PFFT4
NULLDC	Average	46.936	48.656	46.780	46.884
	Maximum	46.952	48.664	46.788	46.892
	Minimum	46.936	48.648	46.772	46.876
FULLDC	Average	47.000	48.656	46.844	47.080
	Maximum	47.012	48.664	46.852	47.088
	Minimum	46.996	48.644	46.836	47.076
MAXDC	Average	47.412	48.656	47.256	47.372
	Maximum	47.424	48.664	47.260	47.380
	Minimum	47.408	48.644	47.248	47.364
COS1	Average	46.972	49.288	47.284	47.100
	Maximum	46.988	49.296	47.308	47.108
	Minimum	46.964	49.280	47.260	47.092
COS8	Average	47.520	49.356	47.364	47.552
	Maximum	47.536	49.368	47.376	47.560
	Minimum	47.516	49.344	47.352	47.548
NOISE	Average	47.972	49.812	47.840	47.940
	Maximum	47.984	49.820	47.848	47.948
	Minimum	47.972	49.800	47.836	47.936

Table A.5.3 Fast Fourier Transform tests: Slave execution time summary.

Input filename	Slave #0 (ms)		Slave #1 (ms)		Slave #2 (ms)		Slave #3 (ms)	
	Sum & Twid	Total	Sum & Twid	Total	Sum & Twid	Total	Sum & Twid	Total
NULLDC	4.880	28.184	6.824	30.128	6.816	30.120	6.816	30.124
FULLDC	4.880	28.248	6.824	30.128	6.816	30.120	6.816	30.124
MAXDC	4.880	28.656	6.824	30.128	6.816	30.120	6.816	30.124
COS1	4.880	28.216	6.824	30.128	6.816	30.780	6.816	30.124
COS8	4.880	28.768	6.824	30.828	6.816	30.816	6.816	30.812
NOISE	4.880	29.108	6.824	31.284	6.816	31.260	6.816	31.264

Table A.5.4 Fast Fourier Transform tests: Multicomputer *time to solution* summary.

Input filename	Average (ms)	Maximum (ms)	Minimum (ms)
NULLDC	38.388	38.392	38.384
FULLDC	38.388	38.392	38.384
MAXDC	38.388	38.392	38.384
COS1	39.020	39.024	39.020
COS8	39.088	39.096	39.084
NOISE	39.544	39.548	39.540

Table A.5.5 Fast Fourier Transform tests: computation time speed-up factors summary.

Input filename	64-pt FFT only (average)	Average Slave Total time	Time to solution
NULLDC	5.28	4.15	3.21
FULLDC	5.28	4.16	3.21
MAXDC	5.32	4.20	3.26
COS1	5.29	4.19	3.19
COS8	5.27	4.17	3.23
NOISE	5.25	4.17	3.24

Table A.5.6 Fast Fourier Transform tests: multicomputer average speed-up factors.

Input filename	Test program			
	PFFT1	PFFT2	PFFT3	PFFT4
NULLDC	2.62	2.53	2.63	2.62
FULLDC	2.62	2.53	2.63	2.62
MAXDC	2.64	2.57	2.64	2.64
COS1	2.65	2.52	2.63	2.64
COS8	2.66	2.56	2.67	2.66
NOISE	2.67	2.57	2.68	2.67

A.6 Frequency domain filtering performance test results

Table A.6.1 Frequency Domain Filtering performance tests: Uniprocessor results.

Input data	Filter	Forward FFT (ms)	Filter (ms)	Inverse FFT (ms)	Total (ms)
NULLDC	[1]	123.036	12.936	114.844	250.812
NOISE	ALLPASS	128.180	14.260	120.968	263.408
	NOPASS	128.180	14.260	114.844	257.280
	LPHLF	128.180	14.260	120.416	262.852
	HPHLF	128.180	14.260	120.436	262.872

1 For input data NULLDC, results were identical for all filter types.

Table A.6.2 Frequency Domain Filtering performance tests: Multicomputer results, S→M→S strategy.

Input data	Filter	Program	Average (ms)	Maximum (ms)	Minimum (ms)
NULLDC	[2]	PFILNOM1	94.872	94.920	94.844
		PFILNOM2	98.412	98.460	98.384
		PFILNOM3	94.568	94.600	94.540
		PFILNOM4	94.864	94.900	94.844
NOISE	ALLPASS	PFILNOM1	97.444	97.492	97.416
		PFILNOM2	101.100	101.148	101.068
		PFILNOM3	97.136	97.188	97.100
		PFILNOM4	97.584	97.648	97.52-
NOISE	NOPASS	PFILNOM1	96.236	96.280	96.204
		PFILNOM2	99.904	99.952	99.872
		PFILNOM3	95.932	95.980	95.888
		PFILNOM4	96.248	96.296	96.176
NOISE	LPHLF	PFILNOM1	97.380	97.424	97.336
		PFILNOM2	101.072	101.108	101.044
		PFILNOM3	97.068	97.136	97.012
		PFILNOM4	97.396	97.440	97.328
NOISE	HPHLF	PFILNOM1	97.384	97.428	97.348
		PFILNOM2	101.028	101.072	100.988
		PFILNOM3	97.076	97.156	97.040
		PFILNOM4	97.376	97.432	97.304

2 For input data NULLDC, results were identical for all filter types.

Table A.6.3 Frequency Domain Filtering performance tests: Multicomputer results, S→S strategy.

Input data	Filter	Program	Average (ms)	Maximum (ms)	Minimum (ms)
NULLDC	[3]	FSTFL1NM	88.776	88.792	88.760
		FSTFL2NM	90.628	90.644	90.616
		FSTFL3NM	88.624	88.640	88.608
		FSTFL4NM	88.760	88.780	88.748
NOISE	ALLPASS	FSTFL1NM	91.452	91.464	91.436
		FSTFL2NM	93.304	93.316	93.292
		FSTFL3NM	91.300	91.312	91.292
		FSTFL4NM	91.596	91.608	91.584
NOISE	NOPASS	FSTFL1NM	90.256	90.268	90.244
		FSTFL2NM	92.112	92.128	92.096
		FSTFL3NM	90.108	90.120	90.096
		FSTFL4NM	90.240	90.252	90.232
NOISE	LPHLF	FSTFL1NM	91.428	91.456	91.400
		FSTFL2NM	93.284	93.296	93.276
		FSTFL3NM	91.292	91.304	91.280
		FSTFL4NM	91.396	91.412	91.384
NOISE	HPHLF	FSTFL1NM	91.396	91.408	91.384
		FSTFL2NM	93.268	93.284	93.252
		FSTFL3NM	91.284	91.296	91.276
		FSTFL4NM	91.376	91.388	91.368

3 For input data NULLDC, results were identical for all filter types.

Table A.6.4 Frequency Domain Filtering performance tests: Multicomputer results; Slave execution times (common to both S→M→S and S→S strategies).

Input data	Filter	Processing Phase	Slave #0	Slave #1	Slave #2	Slave #3
NULLDC	[4]	Forward FFT	28.188	30.132	30.124	30.124
		Filter	3.336	3.336	3.336	3.336
		Inverse FFT	25.624	27.700	27.696	27.696
NOISE	ALLPASS	Forward FFT	29.108	31.284	31.264	31.268
		Filter	3.692	3.668	3.656	3.668
		Inverse FFT	26.820	28.868	28.892	28.892
NOISE	NOPASS	Forward FFT	29.108	31.284	31.264	31.268
		Filter	3.692	3.668	3.656	3.668
		Inverse FFT	25.624	27.700	27.696	27.696
NOISE	LPHLF	Forward FFT	29.108	31.284	31.264	31.268
		Filter	3.692	3.668	3.656	3.668
		Inverse FFT	26.740	28.860	28.872	28.844
NOISE	HPHLF	Forward FFT	29.108	31.284	31.264	31.268
		Filter	3.692	3.668	3.656	3.668
		Inverse FFT	26.768	28.812	28.832	28.856

4 For input data NULLDC, results were identical for all filter types.

Table A.6.5 Frequency Domain Filtering performance tests: summary of Multicomputer speed-up factors.

Input data	Program	Filter			
		ALLPASS	NOPASS	LPHLF	HPHLF
NULLDC	PFILNOM1	2.64	2.64	2.64	2.64
	PFILNOM2	2.55	2.55	2.55	2.55
	PFILNOM3	2.65	2.65	2.65	2.65
	PFILNOM4	2.64	2.64	2.64	2.64
	FSTFL1NM	2.83	2.83	2.83	2.83
	FSTFL2NM	2.77	2.77	2.77	2.77
	FSTFL3NM	2.83	2.83	2.83	2.83
	FSTFL4NM	2.83	2.83	2.83	2.83
NOISE	PFILNOM1	2.70	2.67	2.70	2.70
	PFILNOM2	2.61	2.58	2.60	2.60
	PFILNOM3	2.71	2.68	2.71	2.71
	PFILNOM4	2.70	2.67	2.70	2.70
	FSTFL1NM	2.88	2.85	2.86	2.86
	FSTFL2NM	2.82	2.79	2.82	2.82
	FSTFL3NM	2.89	2.86	2.88	2.88
	FSTFL4NM	2.88	2.85	2.88	2.88

Table A.7.2 Alternating Series performance tests: Uniprocessor and Multicomputer results; optimized Slave programs (PI4STST and PI4MTST).

Number of terms (hex)	Uniprocessor time (ms)	Multicomputer time (ms)			Speed-up
		Average	Max	Min	
2	0.392	1.036	1.040	1.028	0.381
4	0.872	1.220	1.224	1.220	0.715
8	1.824	1.516	1.520	1.516	1.203
10	3.680	2.096	2.100	2.096	1.756
20	7.316	3.208	3.212	3.204	2.283
40	14.524	5.376	5.388	5.376	2.702
80	28.636	9.640	9.652	9.640	2.971
100	56.376	17.920	17.924	17.916	3.147
200	110.932	34.040	34.052	34.036	3.259
400	218.220	66.404	66.424	66.376	3.288
800	428.988	131.272	131.304	131.260	3.268
1000	843.572	260.972	260.992	260.948	3.232
2000	1656.672	520.172	520.188	520.144	3.185
4000	3253.896	1039.216	1039.244	1039.200	3.131
8000	6390.168	2079.004	2079.024	2078.980	3.074

Table A.6.5 Frequency Domain Filtering performance tests: summary of Multicomputer speed-up factors.

Input data	Program	Filter			
		ALLPASS	NOPASS	LPHLF	HPHLF
NULLDC	PFILNOM1	2.64	2.64	2.64	2.64
	PFILNOM2	2.55	2.55	2.55	2.55
	PFILNOM3	2.65	2.65	2.65	2.65
	PFILNOM4	2.64	2.64	2.64	2.64
	FSTFL1NM	2.83	2.83	2.83	2.83
	FSTFL2NM	2.77	2.77	2.77	2.77
	FSTFL3NM	2.83	2.83	2.83	2.83
	FSTFL4NM	2.83	2.83	2.83	2.83
NOISE	PFILNOM1	2.70	2.67	2.70	2.70
	PFILNOM2	2.61	2.58	2.60	2.60
	PFILNOM3	2.71	2.68	2.71	2.71
	PFILNOM4	2.70	2.67	2.70	2.70
	FSTFL1NM	2.88	2.85	2.86	2.86
	FSTFL2NM	2.82	2.79	2.82	2.82
	FSTFL3NM	2.89	2.86	2.88	2.88
	FSTFL4NM	2.88	2.85	2.88	2.88

A.7 Alternating Series Calculation Performance test results

Table A.7.1 Alternating Series performance tests: execution times of sub-tasks division and addition (program PI4INFO).

Number of terms (hex)	Divide time (µs)		Add time (µs)	
	$(n-1)^{th}$ term	n^{th} term	$(n-1)^{th}$ term ADD	n^{th} term SUBTRACT
2	88	116	44	132
4	116	116	100	124
8	112	116	108	124
10	104	112	104	124
20	100	108	100	124
40	96	104	108	120
80	100	104	104	124
100	96	100	104	120
200	92	96	104	120
400	84	88	104	120
800	88	88	108	124
1000	84	84	104	120
2000	80	80	104	120
4000	76	76	104	120
8000	72	72	104	120

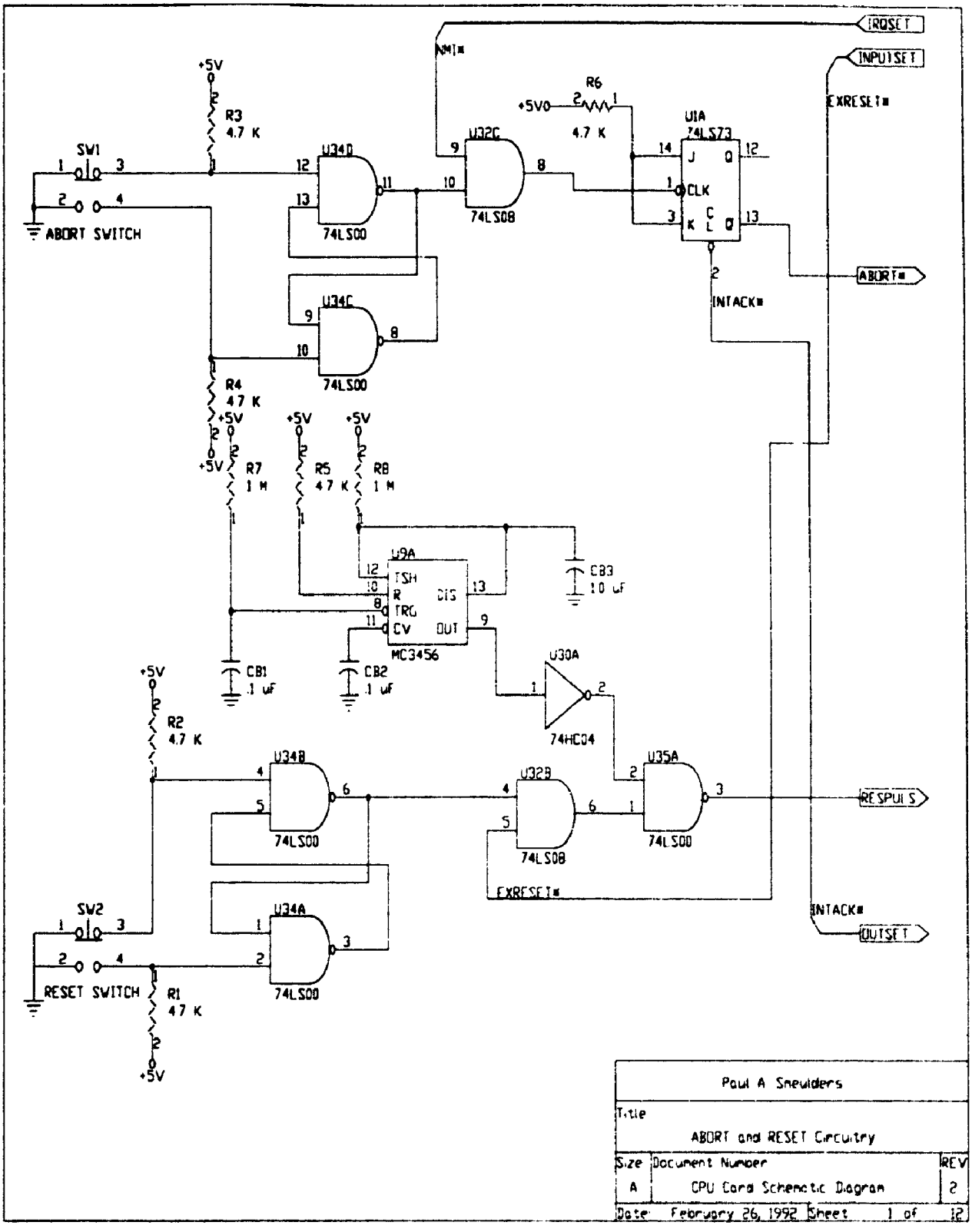
Table A.7.2 Alternating Series performance tests: Uniprocessor and Multicomputer results; optimized Slave programs (PI4STST and PI4MTST).

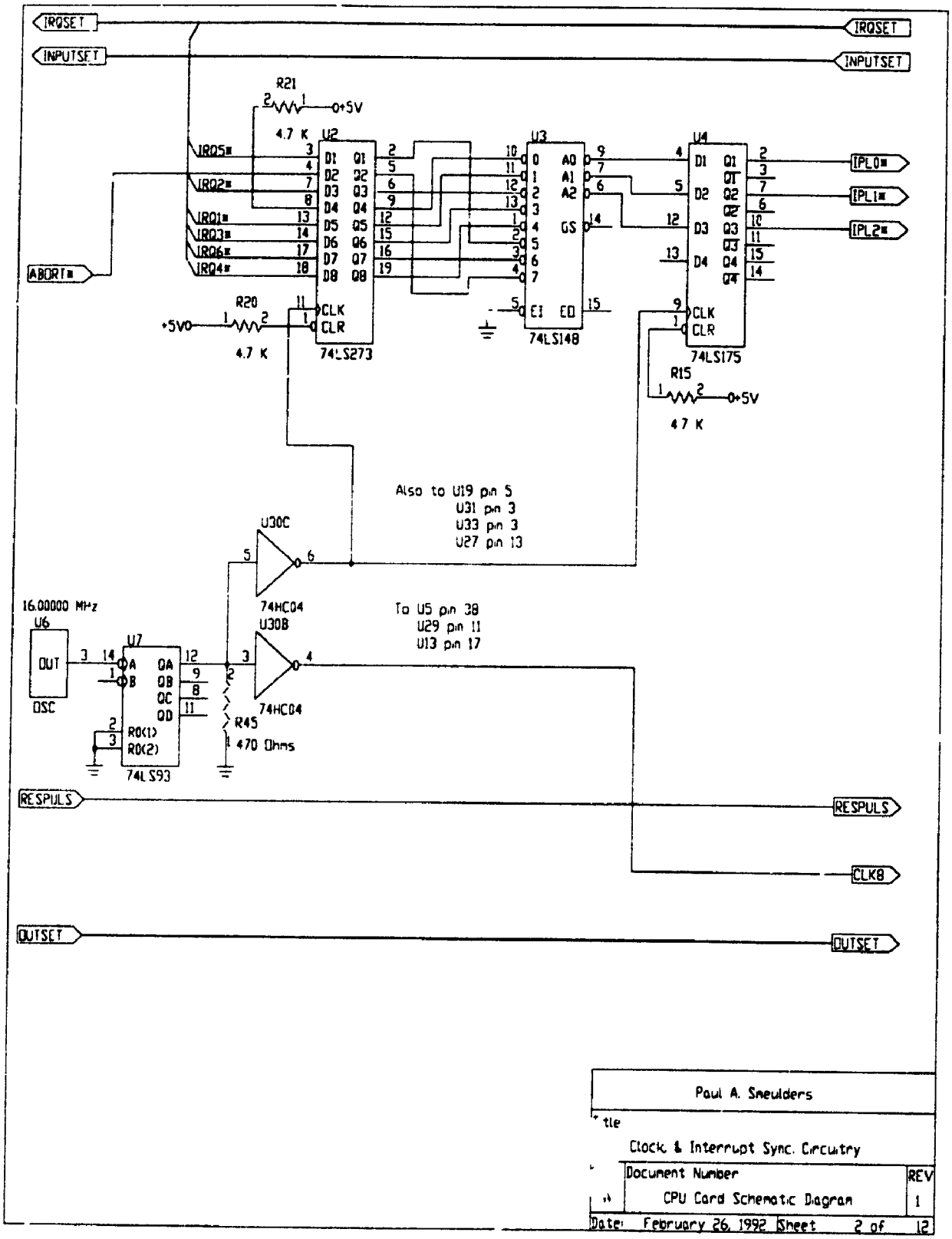
Number of terms (hex)	Uniprocessor time (ms)	Multicomputer time (ms)			Speed-up
		Average	Max	Min	
2	0.392	1.036	1.040	1.028	0.381
4	0.872	1.220	1.224	1.220	0.715
8	1.824	1.516	1.520	1.516	1.203
10	3.680	2.096	2.100	2.096	1.756
20	7.316	3.208	3.212	3.204	2.283
40	14.524	5.376	5.388	5.376	2.702
80	28.636	9.640	9.652	9.640	2.971
100	56.376	17.920	17.924	17.916	3.147
200	110.932	34.040	34.052	34.036	3.259
400	218.220	66.404	66.424	66.376	3.288
800	428.988	131.272	131.304	131.260	3.268
1000	843.572	260.972	260.992	260.948	3.232
2000	1656.672	520.172	520.188	520.144	3.185
4000	3253.896	1039.216	1039.244	1039.200	3.131
8000	6390.168	2079.004	2079.024	2078.980	3.074

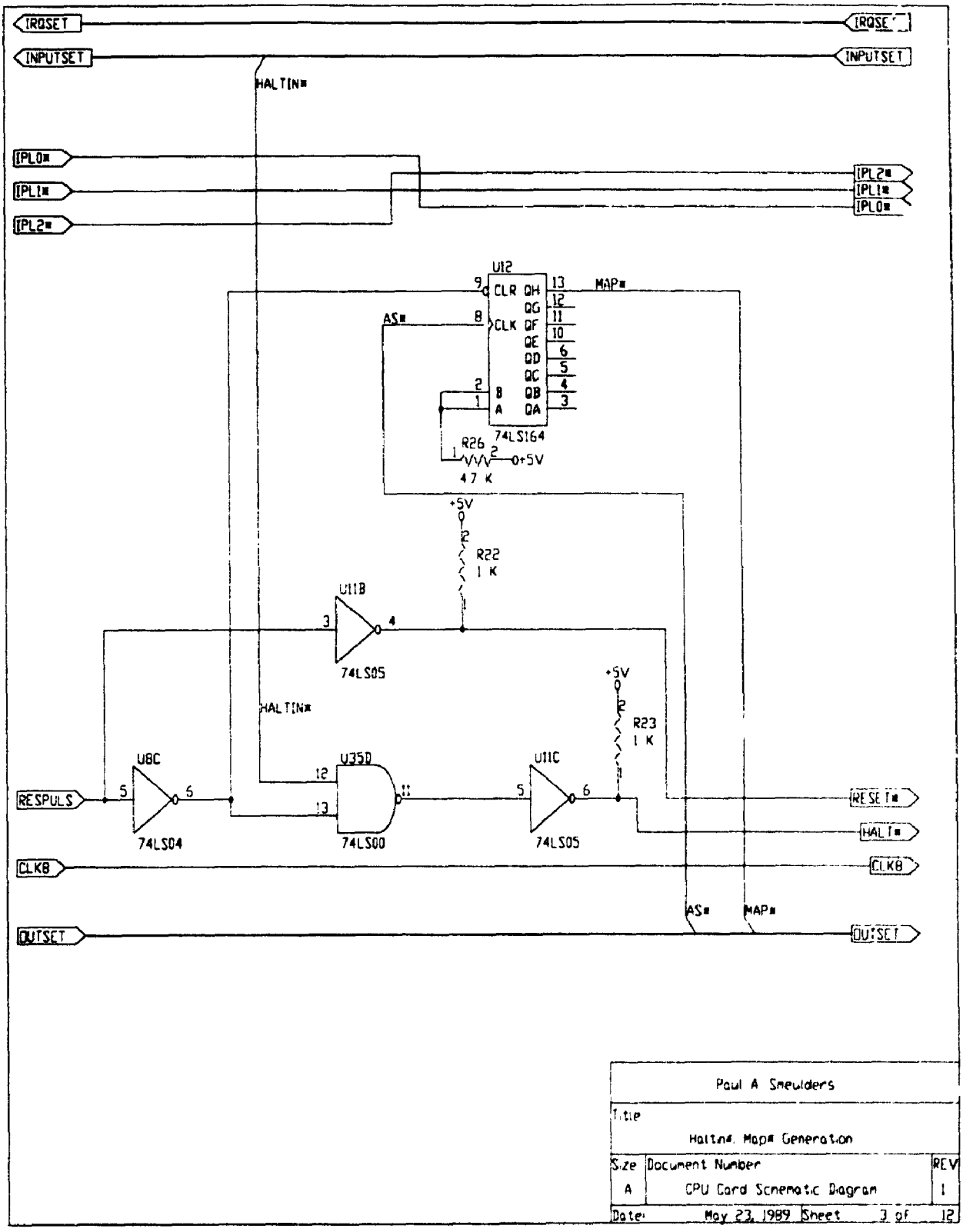
Table A.7.3 Alternating Series performance tests: Uniprocessor and Multicomputer results; lengthened Slave programs (PI4LSTST and PI4LMTST).

Number of terms (hex)	Uniprocessor time (ms)	Multicomputer Average time (ms)	Speed-up
2	10.060	3.464	2.904
4	20.208	6.080	3.324
8	40.496	11.240	3.603
10	81.024	21.532	3.763
20	162.004	42.080	3.850
40	323.900	83.132	3.896
80	647.388	165.144	3.920
100	1293.880	328.924	3.933
200	2585.940	655.920	3.942
400	5168.236	1308.976	3.948
800	10329.020	2613.120	3.952
1000	20643.640	5217.676	3.956
2000	41256.800	10418.690	3.960
4000	82424.160	20805.720	3.963
8000	164790.700	41554.930	3.965

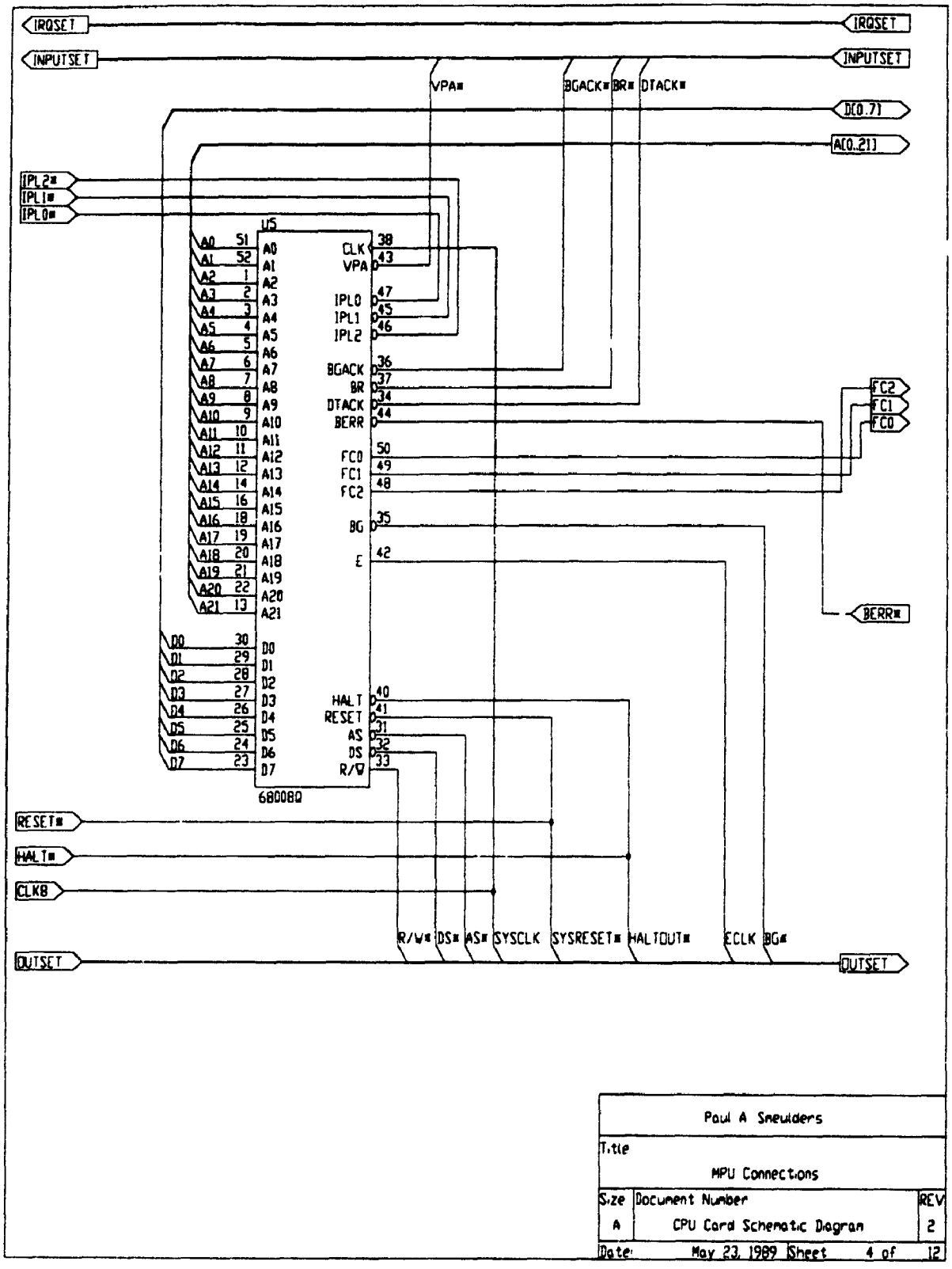
Appendix B: CPU Card Schematic Diagrams


IRQSET
NMI#
INPUTSET
EXRESET#
+5V
R3
4.7 K
SW1
ABORT SWITCH
U34D
74LS00
U32C
74LS08
R6
4.7 K
U1A
74LS73
J
CLK
K
Q
ABORT#
INTACK#
U34C
74LS00
R4
4.7 K
R7
1 M
R5
4.7 K
R8
1 M
U9A
TSH
R
TRG
CV
DIS
OUT
MC3456
CB3
10 uF
CB1
.1 uF
CB2
.1 uF
U30A
74HC04
R2
4.7 K
U34B
74LS00
U32B
74LS08
U35A
74LS00
RESPULS
SW2
RESET SWITCH
R1
4.7 K
U34A
74LS00
EXRESET#
INTACK#
OUTSET
Paul A Snewlders
Title
ABORT and RESET Circuitry
Size
A
Document Number
CPU Card Schematic Diagram
REV
2
Date: February 26, 1992 Sheet 1 of 12

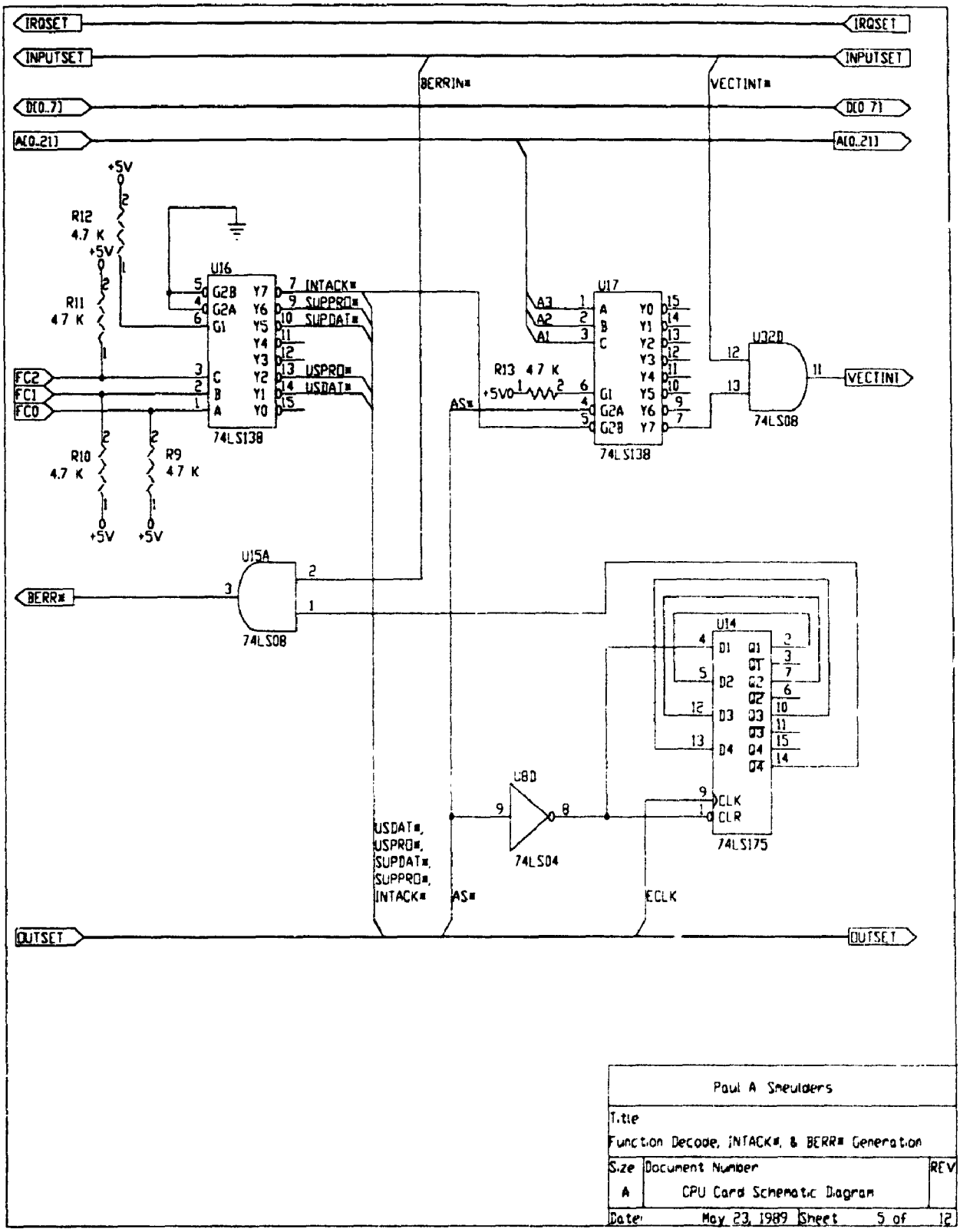

IRQSET
INPUTSET
R21
4.7 K
+5V
U2
IRQ5*
IRQ2*
IRQ1*
IRQ3*
IRQ6*
IRQ4*
ABORT*
R20
+5V
4.7 K
74LS273
U3
74LS148
U4
74LS175
IPL0*
IPL1*
IPL2*
R15
4.7 K
Also to U19 pin 5
U31 pin 3
U33 pin 3
U27 pin 13
U30C
74HC04
16.00000 MHz
U6
OUT
OSC
U7
74LS93
U30B
74HC04
To U5 pin 38
U29 pin 11
U13 pin 17
R45
470 Ohms
RESPULS
CLK8
OUTSET
Paul A. Sneulders
Clock & Interrupt Sync. Circuitry
Document Number
CPU Card Schematic Diagram
REV
1
Date: February 26, 1992 Sheet 2 of 12

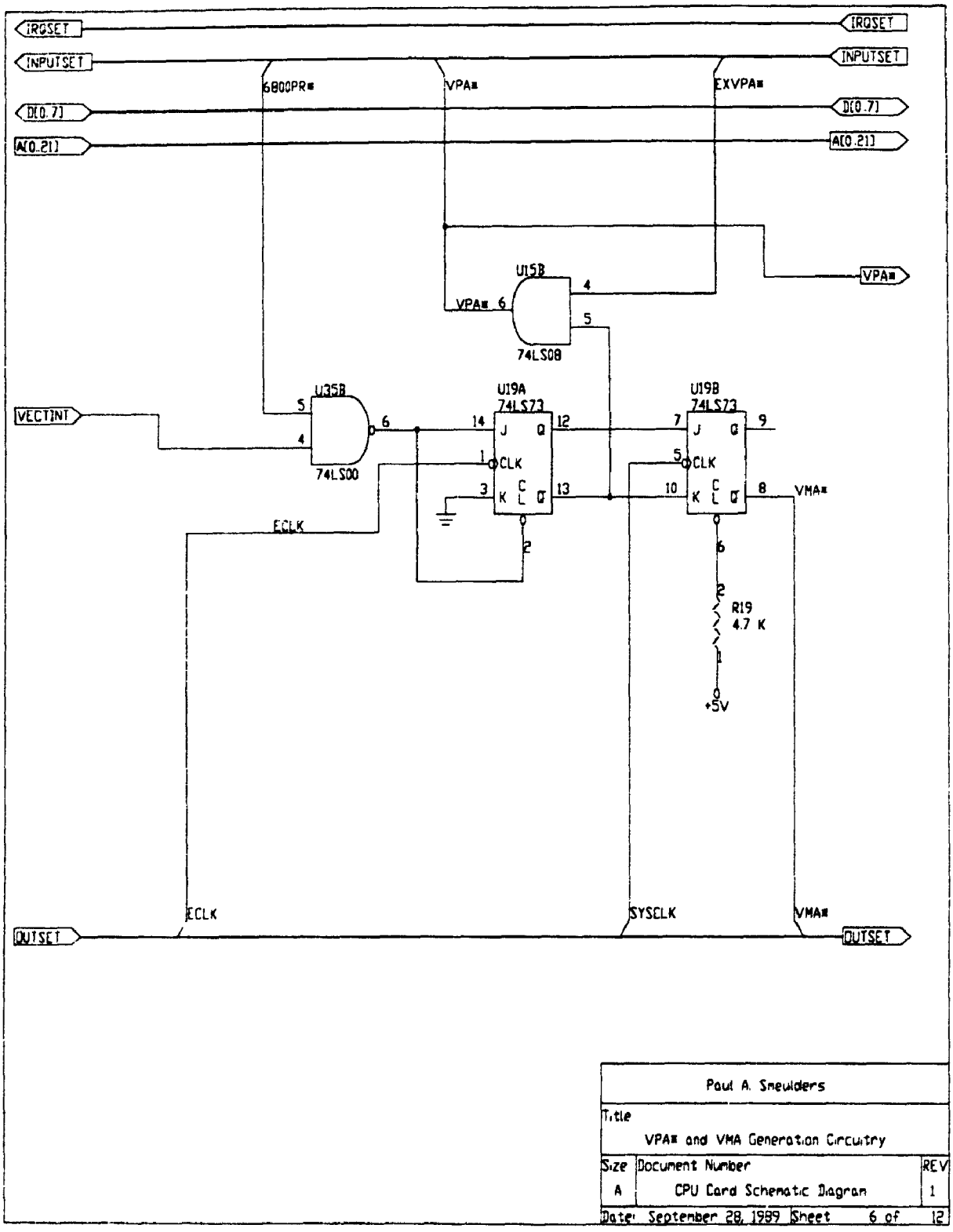


Paul A Smeulders		
Title: Halt*, Map* Generation		
Size: A	Document Number: CPU Card Schematic Diagram	REV: 1
Date: May 23, 1989	Sheet 3 of 12	

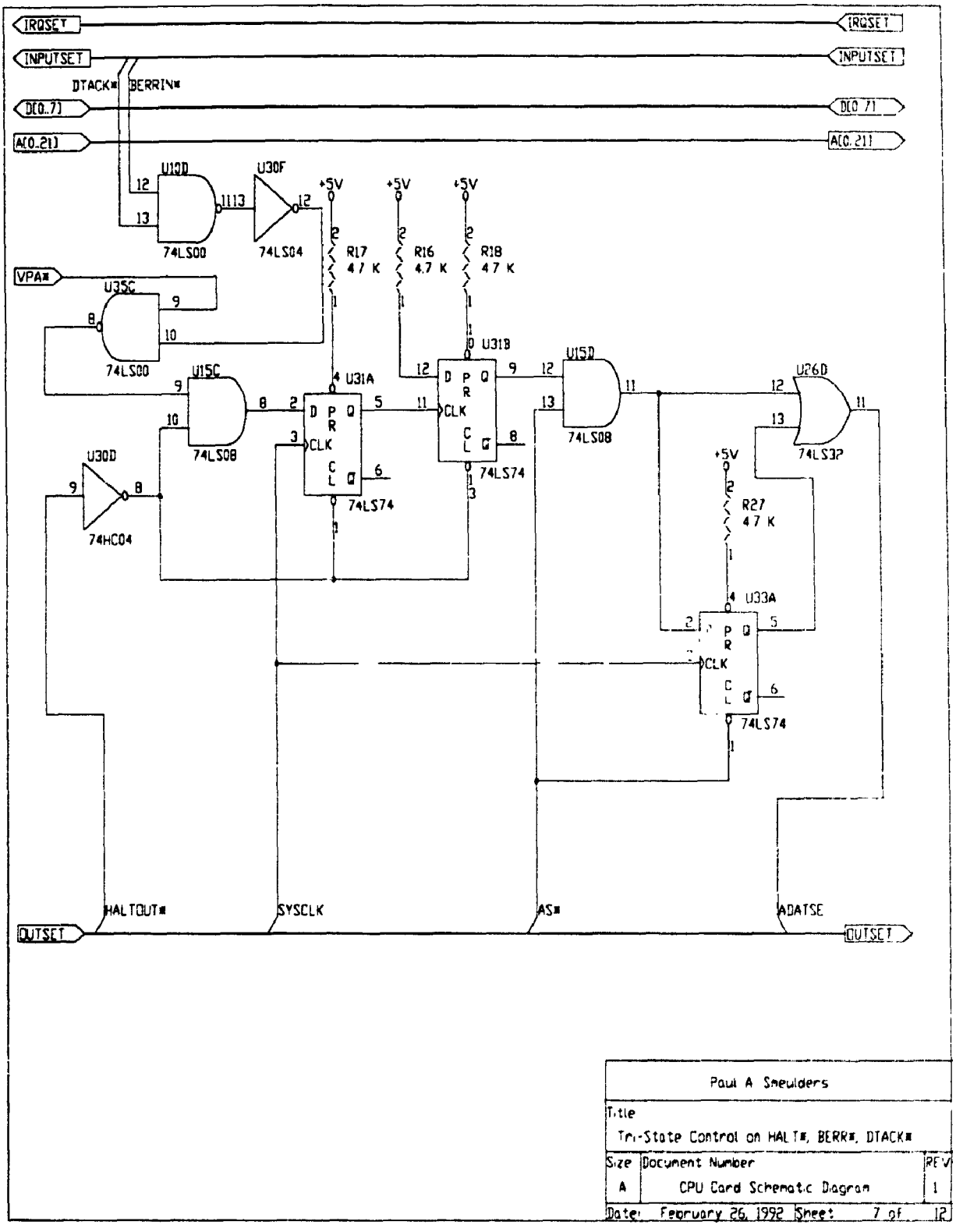


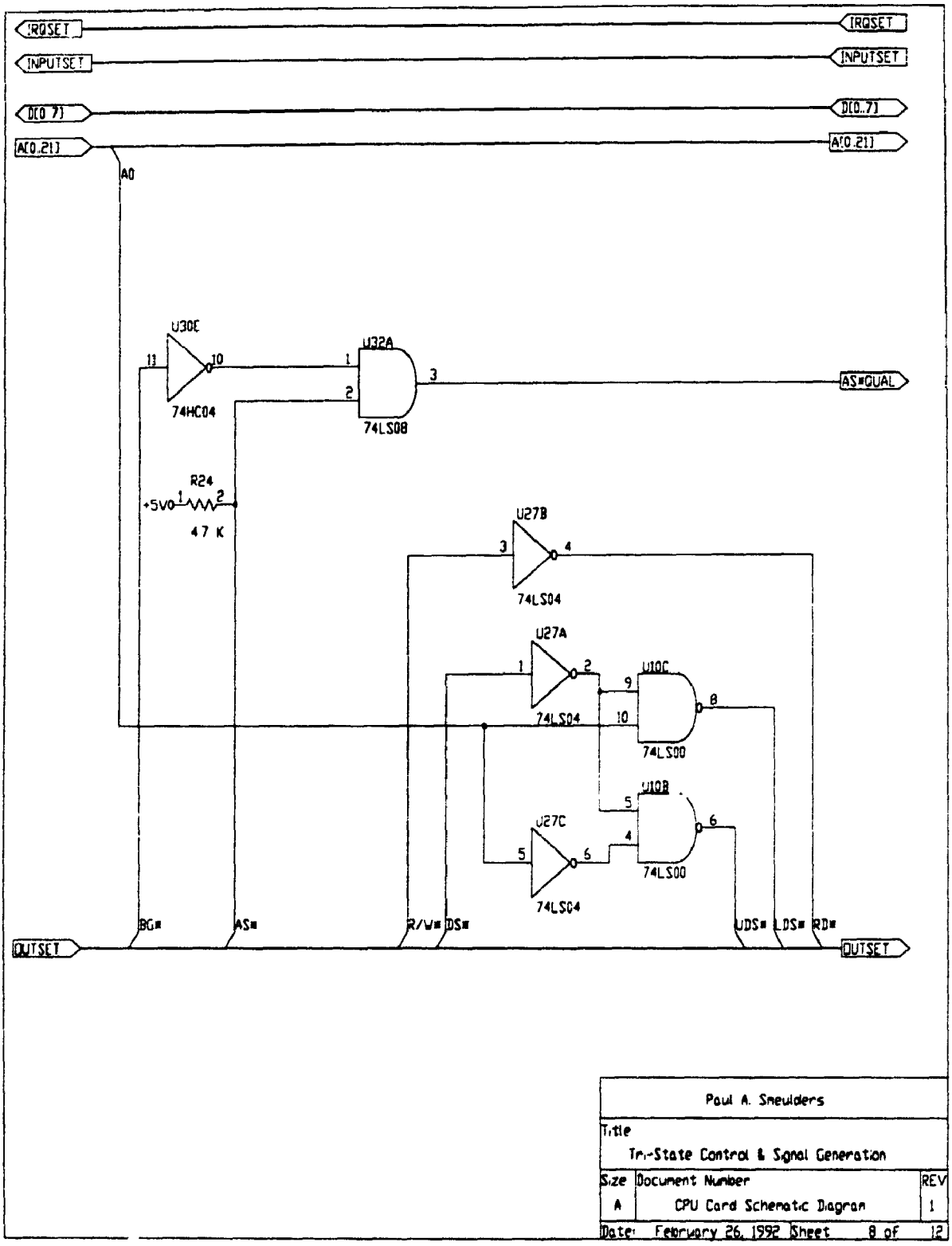
Paul A Sneulders		
Title: MPU Connections		
Size: A	Document Number: CPU Card Schematic Diagram	REV: 2
Date: May 23, 1989	Sheet: 4 of 12	


IRQSET
INPUTSET
D[0..7]
A[0..21]
BERRIN*
VECTINT*
R12
4.7 K
+5V
R11
4.7 K
U16
G2B
G2A
G1
C
B
A
Y7
Y6
Y5
Y4
Y3
Y2
Y1
Y0
INTACK*
SUPPRQ*
SUPDAT*
USPRQ*
USDAT*
74LS138
FC2
FC1
FC0
R10
4.7 K
R9
4.7 K
U17
A3
A2
A1
R13 4.7 K
AS*
74LS138
U32D
74LS08
VECTINT
U15A
BERR*
74LS08
U14
D1
D2
D3
D4
Q1
Q2
Q3
Q4
CLK
CLR
74LS175
U8D
74LS04
USDAT*,
USPRQ*,
SUPDAT*,
SUPPRQ*,
INTACK*
AS*
ECLK
OUTSET
Paul A Sneulders
Title
Function Decode, INTACK*, & BERR* Generation
Size
A
Document Number
CPU Card Schematic Diagram
REV
Date: May 23, 1989
Sheet 5 of 12

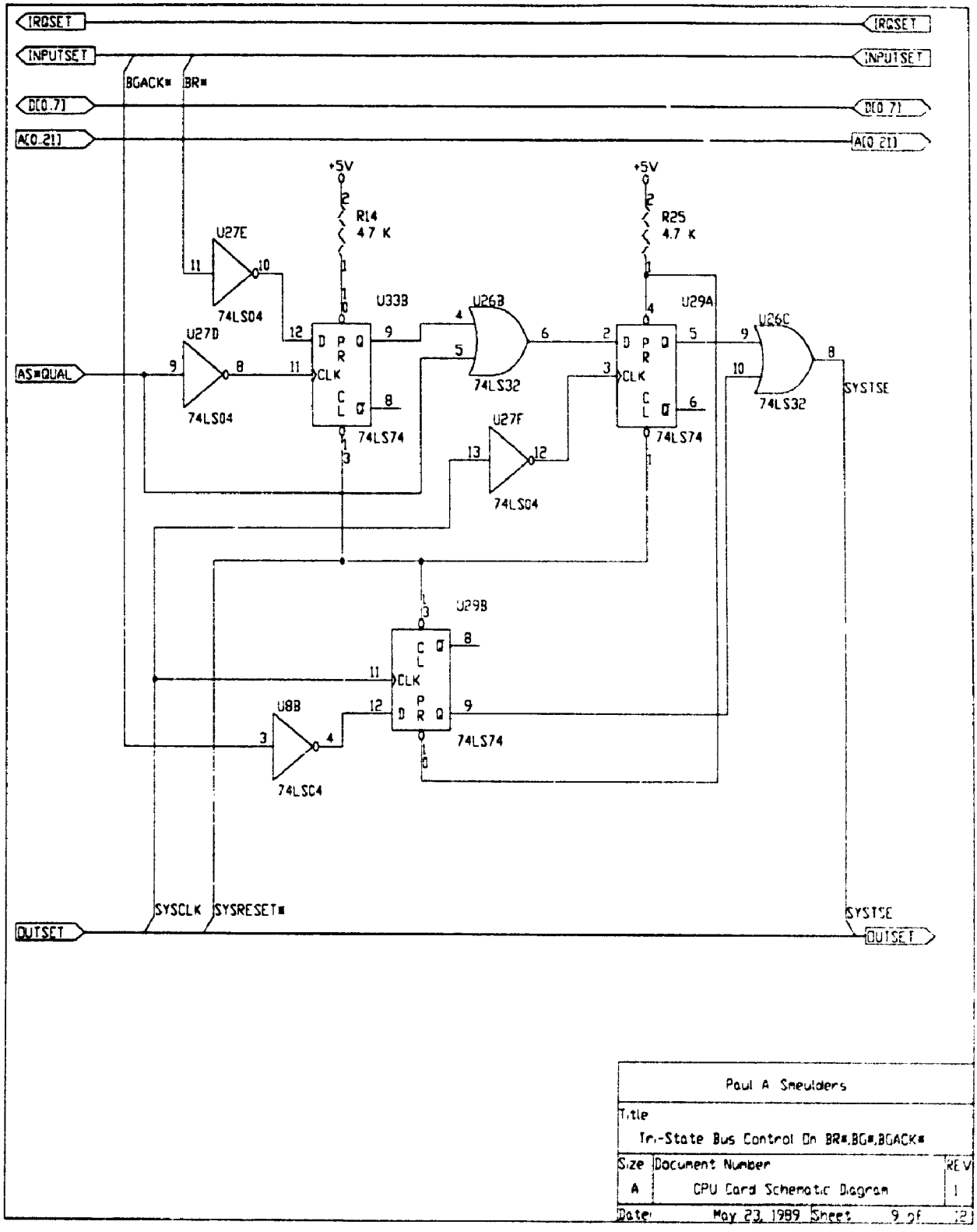


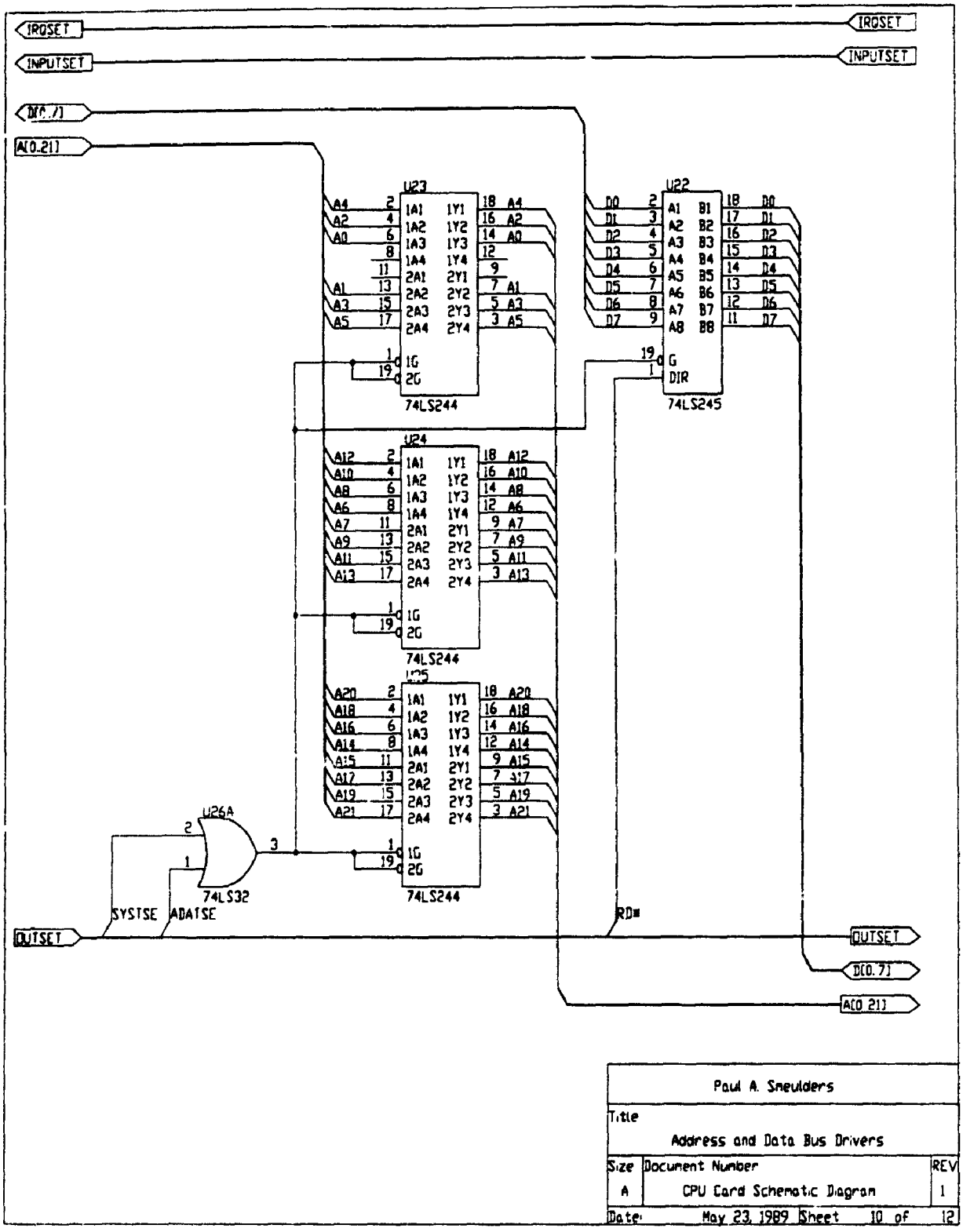
Paul A. Smeulders		
Title: VPA≡ and VMA Generation Circuitry		
Size: A	Document Number: CPU Card Schematic Diagram	REV: 1
Date: September 28, 1989	Sheet: 6 of	12

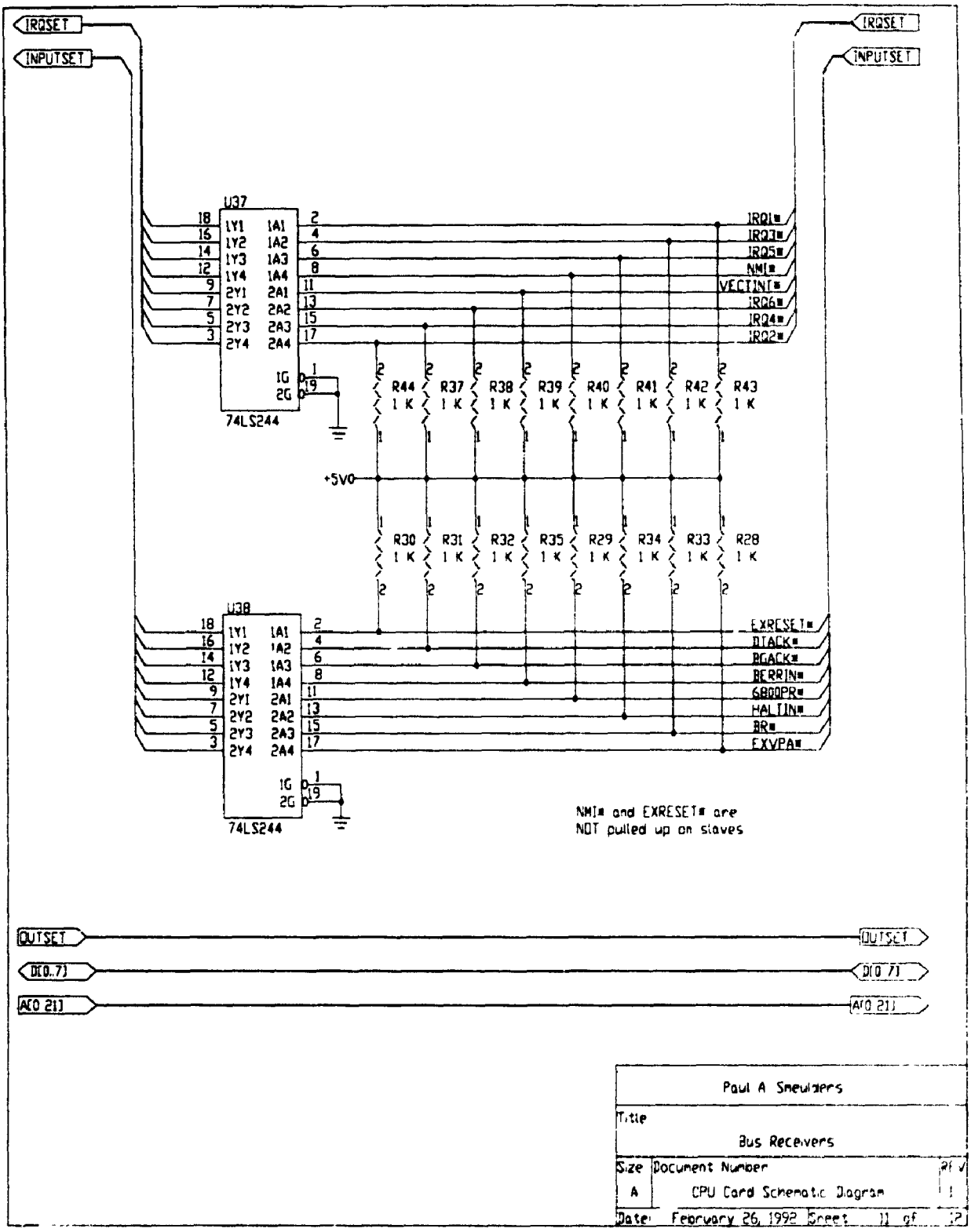

IRQSET
INPUTSET
DTACK#
BERRIN#
D[0..7]
A[0..21]
U10D
74LS00
U30F
74LS04
+5V
R17
4.7 K
R16
4.7 K
R18
4.7 K
VPA#
U35C
74LS00
U31B
U31A
74LS74
U15C
74LS08
U15D
74LS08
U26D
74LS32
U30D
74HC04
R27
4.7 K
U33A
74LS74
HALTOUT#
SYSCLK
AS#
ADATSE
OUTSET
Paul A Sneulders
Title
Tri-State Control on HALT#, BERR#, DTACK#
Size
A
Document Number
CPU Card Schematic Diagram
REV
1
Date: February 26, 1992
Sheet 7 of 12

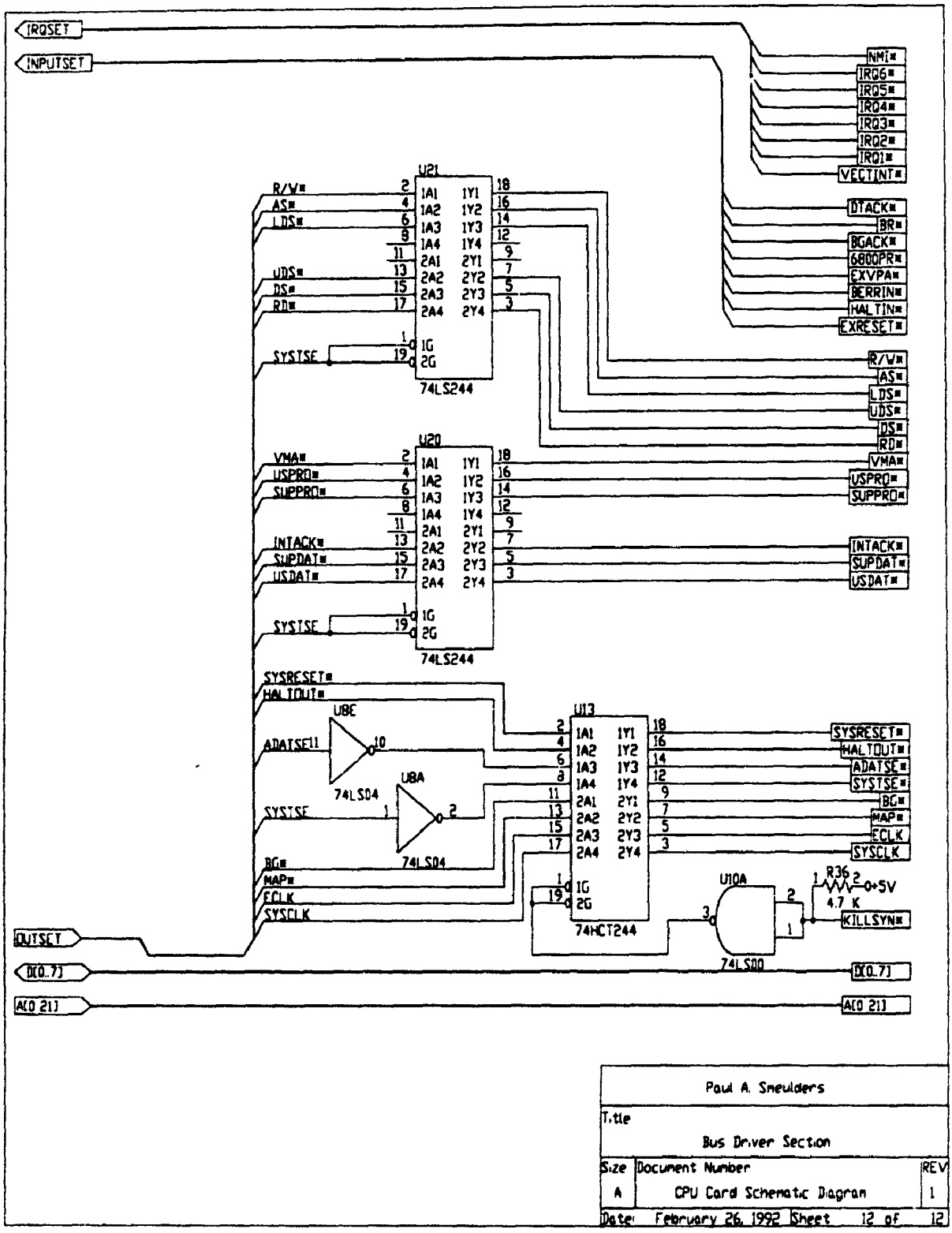


Paul A. Sneulders		
Title: Tri-State Control & Signal Generation		
Size: A	Document Number: CPU Card Schematic Diagram	REV: 1
Date: February 26, 1992	Sheet: 8 of 12	


IRQSET
INPUTSET
BGACK*
BR*
D[0.7]
A[0.21]
+5V
R14
4.7 K
R25
4.7 K
U27E
74LS04
U27D
74LS04
U33B
74LS74
U26B
74LS32
U29A
74LS74
U26C
74LS32
U27F
74LS04
U29B
74LS74
U8B
74LS04
AS*QUAL
SYSTSE
SYSCLK
SYSRESET*
OUTSET
Paul A Smeulders
Title
In-State Bus Control On BR*,BG*,BGACK*
Size
A
Document Number
CPU Card Schematic Diagram
REV
1
Date: May 23, 1989 Sheet 9 of 12

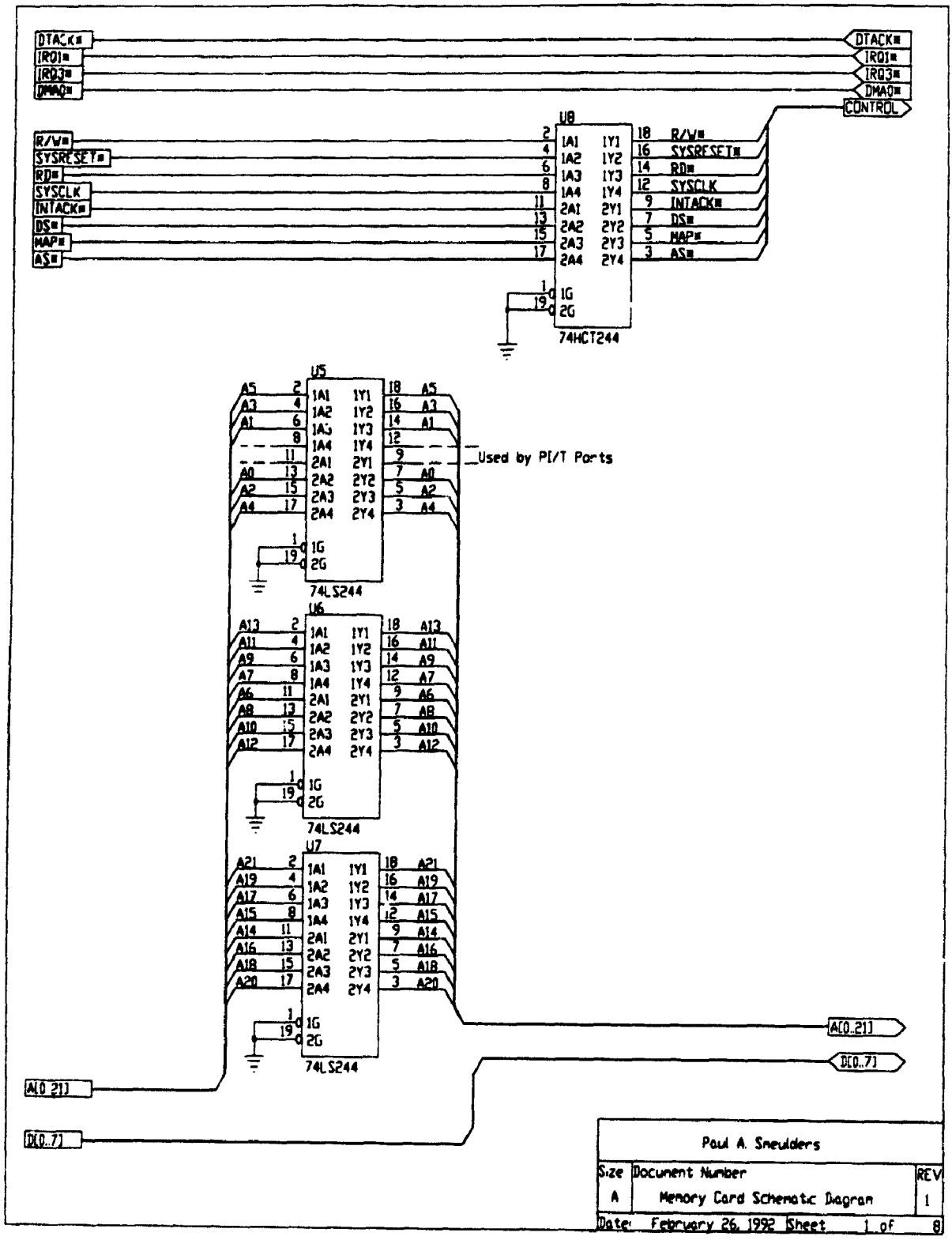

IRQSET
INPUTSET
D[0..7]
A[0..21]
U23
74LS244
U22
74LS245
U24
74LS244
U25
74LS244
U26A
74LS32
SYSTSE
ADATSE
OUTSET
RD
Paul A. Sneulders
Address and Data Bus Drivers
Size A
Document Number
CPU Card Schematic Diagram
REV 1
Date: May 23, 1989
Sheet 10 of 12

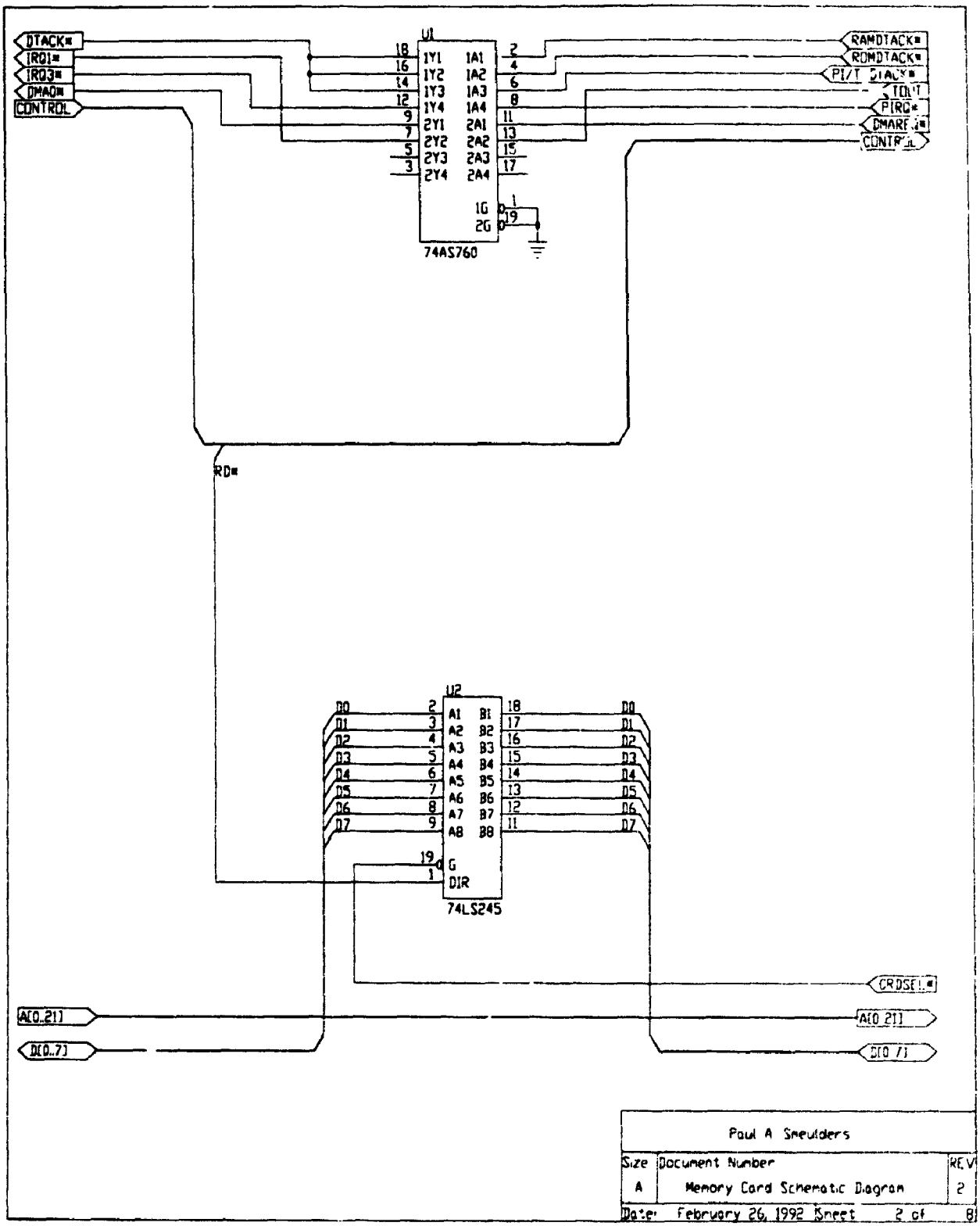

IRQSET
INPUTSET
U37
74LS244
R44 1 K
R37 1 K
R38 1 K
R39 1 K
R40 1 K
R41 1 K
R42 1 K
R43 1 K
+5VO
R30 1 K
R31 1 K
R32 1 K
R35 1 K
R29 1 K
R34 1 K
R33 1 K
R28 1 K
IRQ1*
IRQ3*
IRQ5*
NMI*
VECTINT*
IRQ6*
IRQ4*
IRQ2*
U38
74LS244
EXRESET*
DTACK*
BGACK*
BERRIN*
68000PR*
HALTIN*
BR*
EXVPA*
NMI* and EXRESET* are
NOT pulled up on slaves
OUTSET
D[0..7]
A[0 21]
Paul A Sneulders
Title
Bus Receivers
Size
A
Document Number
CPU Card Schematic Diagram
Date: February 26, 1992 Sheet 11 of 12



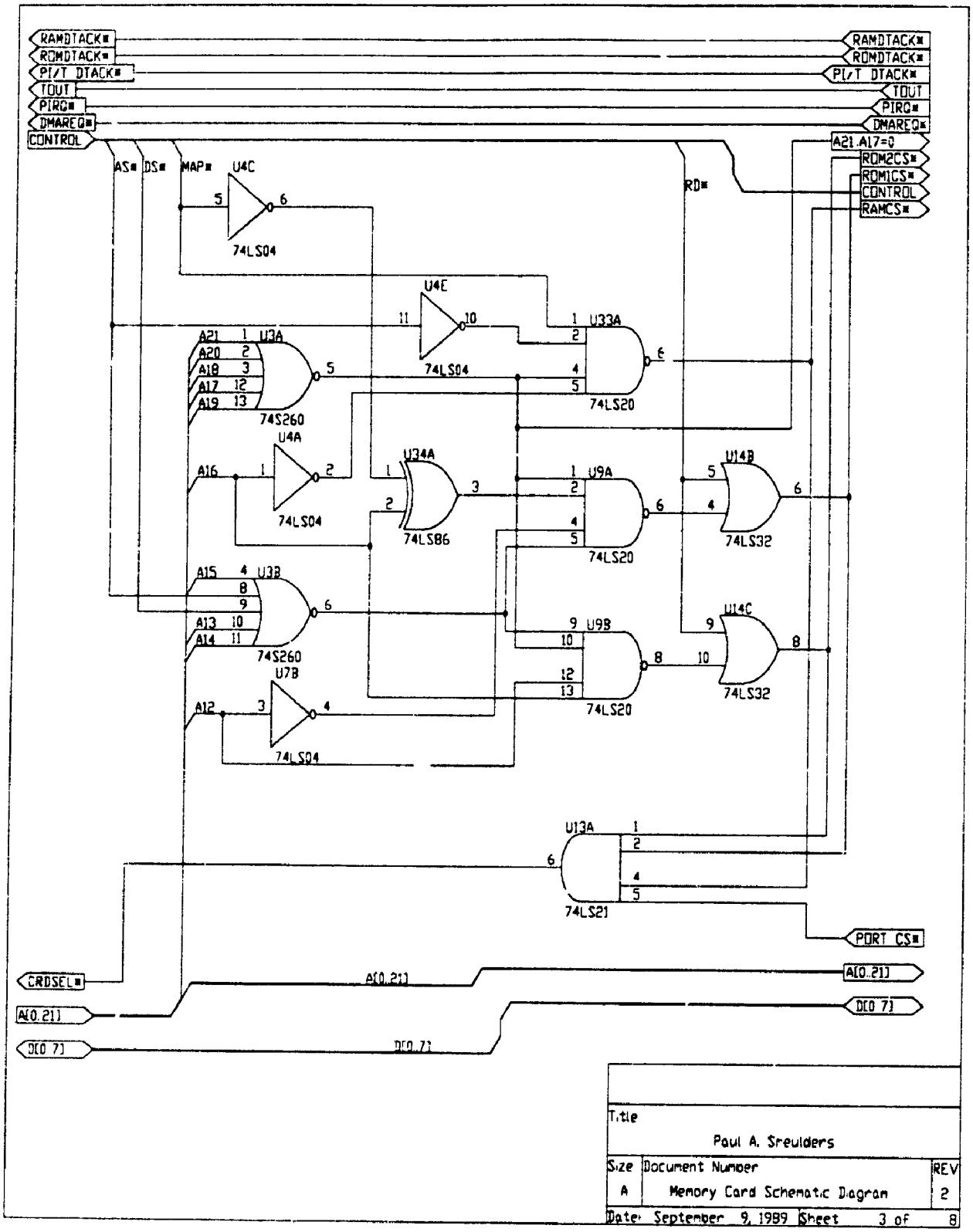
Paul A. Smeulders		
Title: Bus Driver Section		
Size: A	Document Number: CPU Card Schematic Diagram	REV: 1
Date: February 26, 1992	Sheet 12 of 12	

Appendix C: Memory and PI/T Card Schematic Diagrams

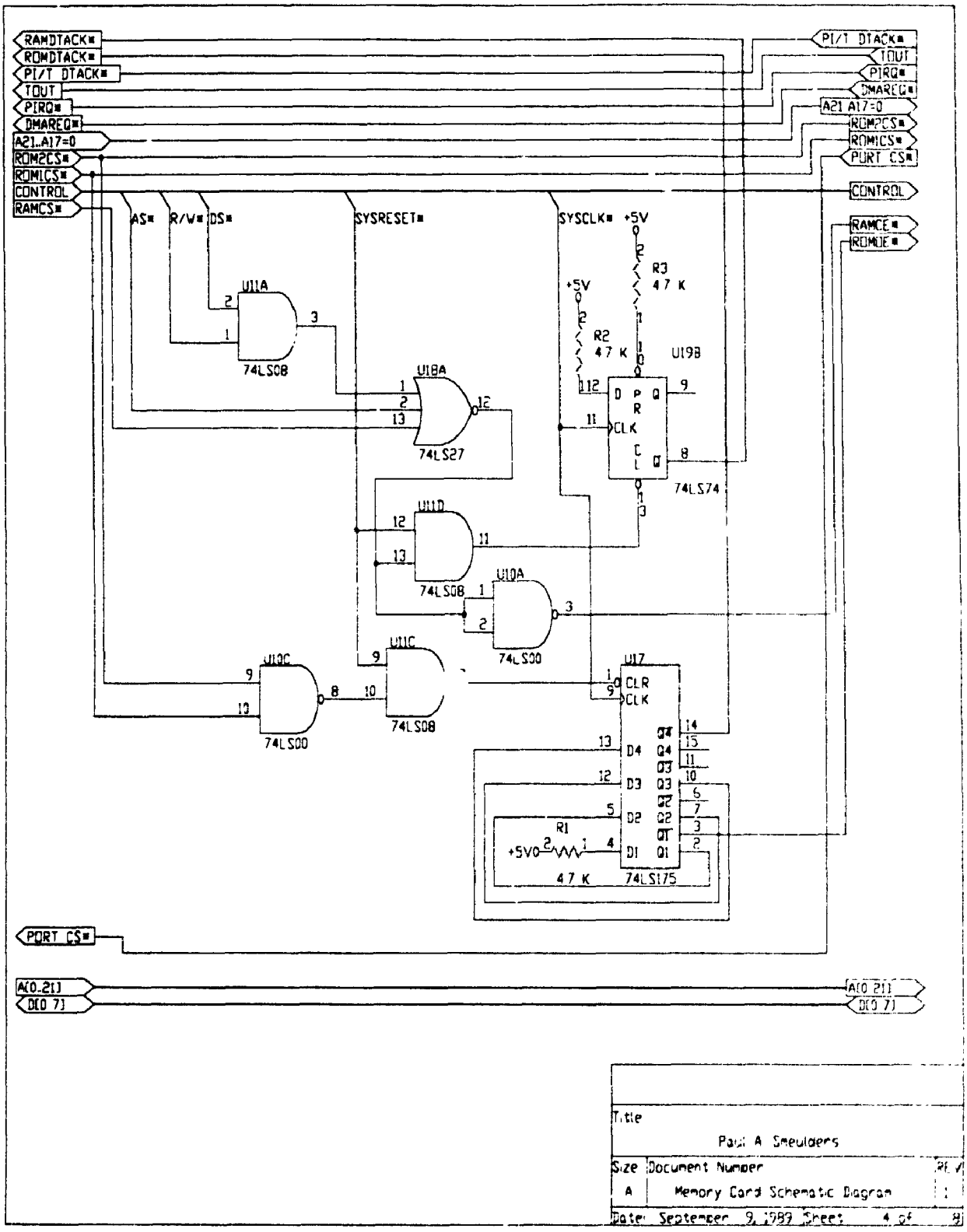

DTACK#
IRQ1#
IRQ3#
DMAQ#
CONTROL
U8
74HCT244
R/W#
SYSRESET#
RD#
SYSCLK
INTACK#
DS#
MAP#
AS#
U5
74LS244
Used by PI/T Ports
U6
74LS244
U7
74LS244
A[0..21]
D[0..7]
Paul A. Sneulders
Size
A
Document Number
Memory Card Schematic Diagram
REV
1
Date: February 26, 1992 Sheet 1 of 8

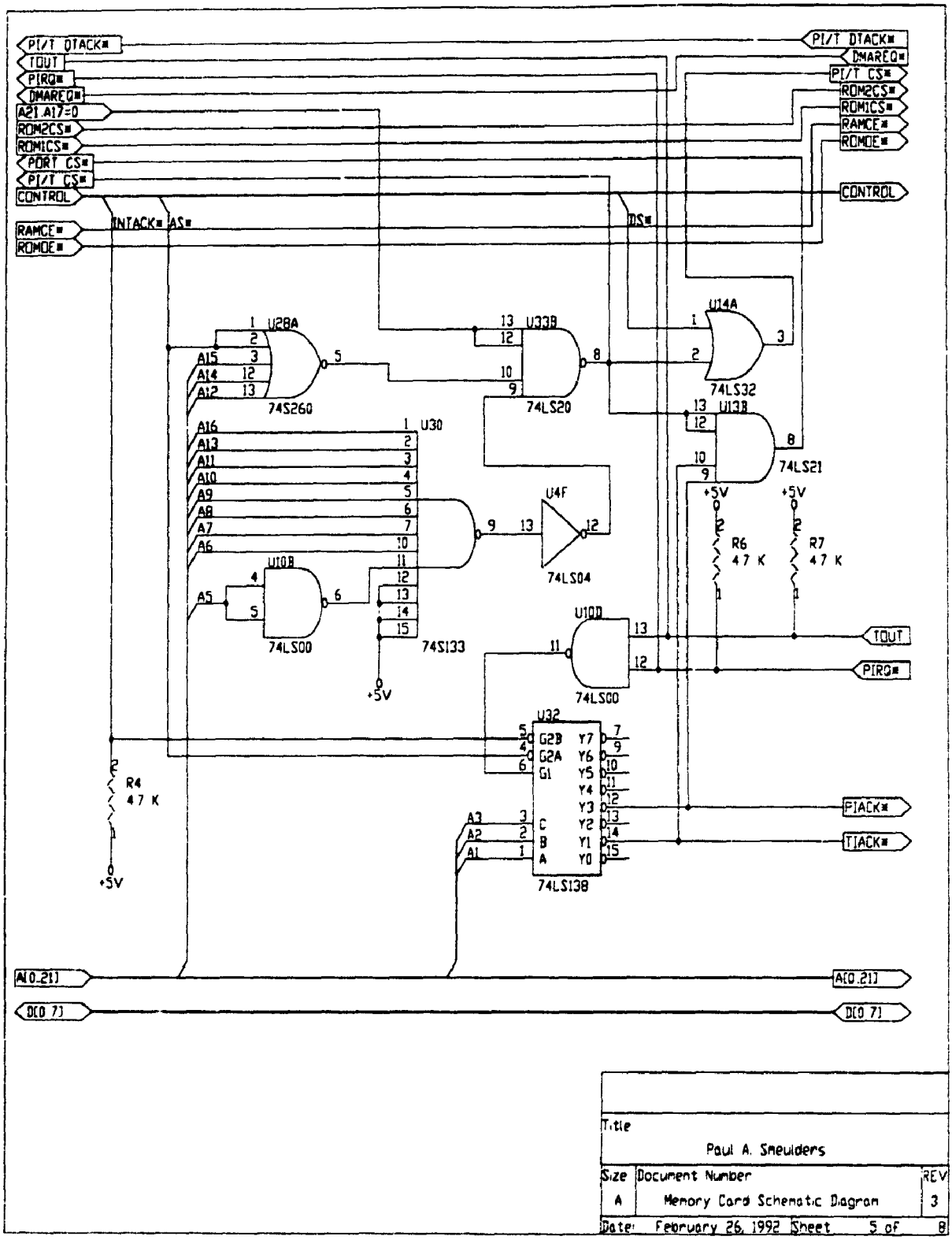


Paul A Sneulders		
Size: A	Document Number: Memory Card Schematic Diagram	REV: 2
Date: February 26, 1992	Sheet 2 of 8	

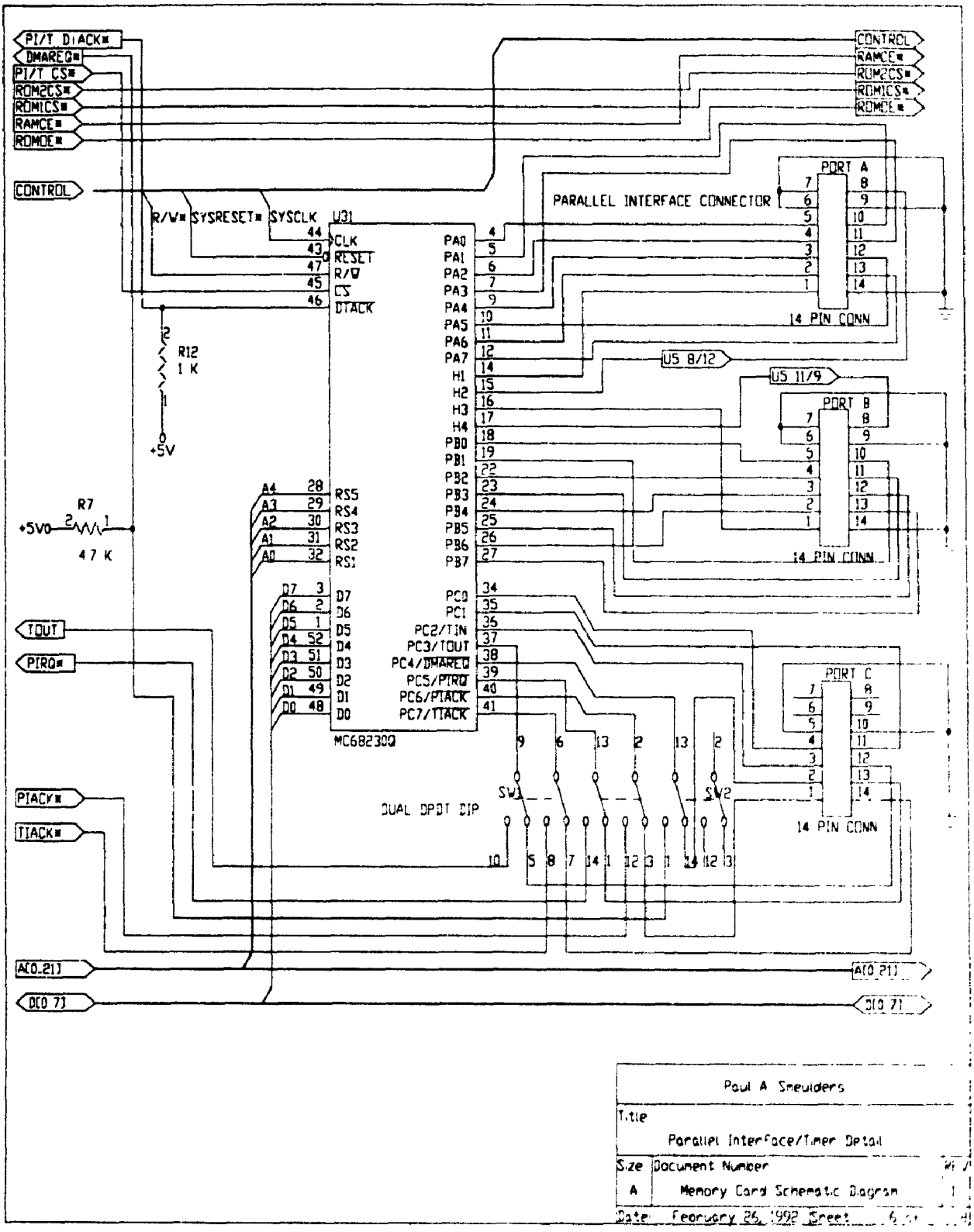


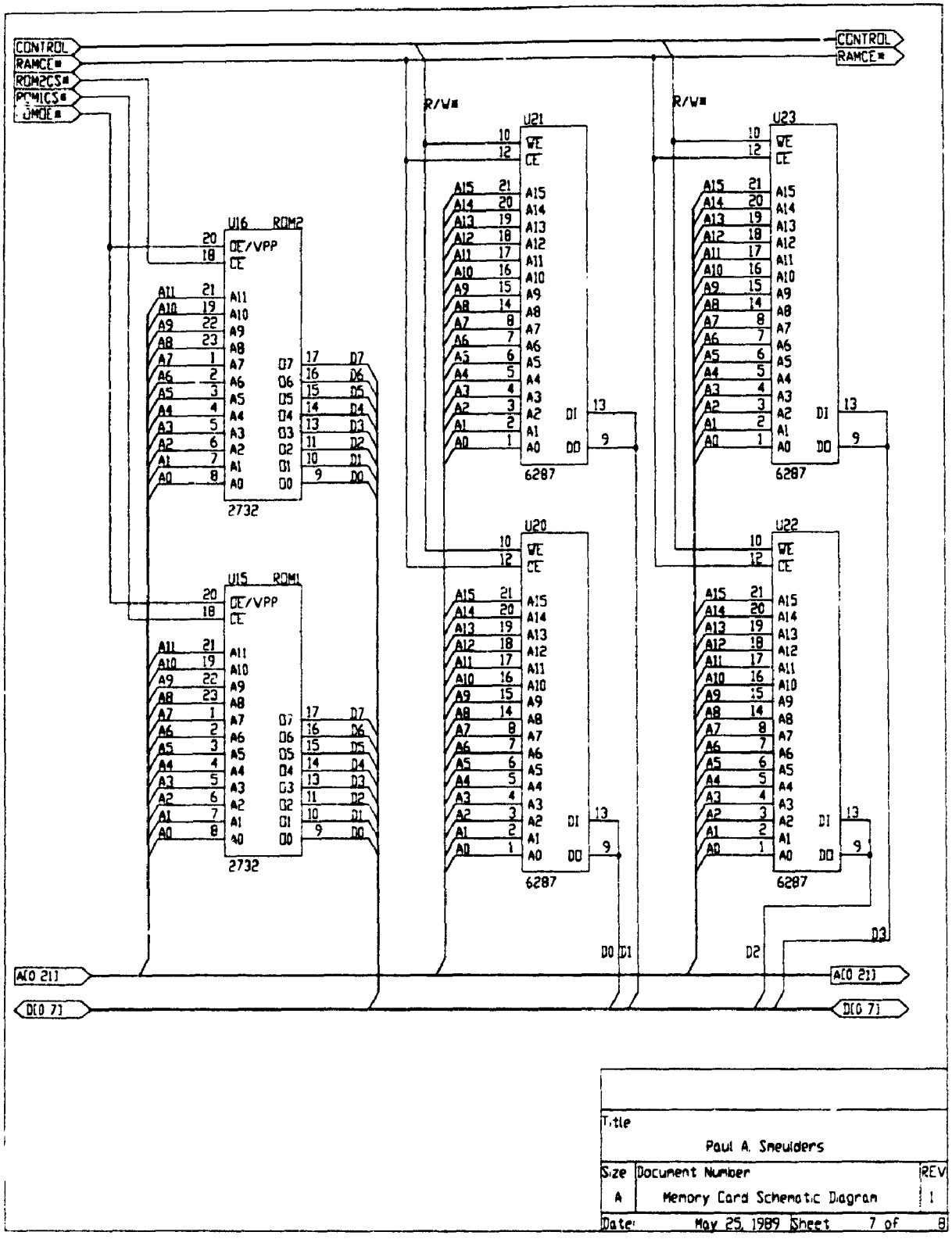
Title	Paul A. Sreulders	
Size: A	Document Number: Memory Card Schematic Diagram	REV: 2
Date: September 9, 1989	Sheet 3 of 8	


RAMDTACK*
ROMDTACK*
PI/T DTACK*
TOUT
PIRQ*
DMAREQ*
A21..A17=0
ROM2CS*
ROM1CS*
CONTROL
RAMCS*
AS*
R/W*
DS*
SYSRESET*
SYSCLK*
+5V
U11A
74LS08
U18A
74LS27
R3
4.7 K
R2
4.7 K
U19B
74LS74
U11D
74LS08
U10A
74LS00
U10C
74LS00
U11C
74LS08
U17
74LS175
R1
4.7 K
PORT CS*
A(0..21)
D(0..7)
PI/T DTACK*
TOUT
PIRQ*
DMAREQ*
A21..A17=0
ROM2CS*
ROM1CS*
PORT CS*
CONTROL
RAMCE*
ROMOE*
A(0..21)
D(0..7)
Title
Paul A Smeulders
Size
A
Document Number
Memory Card Schematic Diagram
Date: September 9, 1989 Sheet 4 of

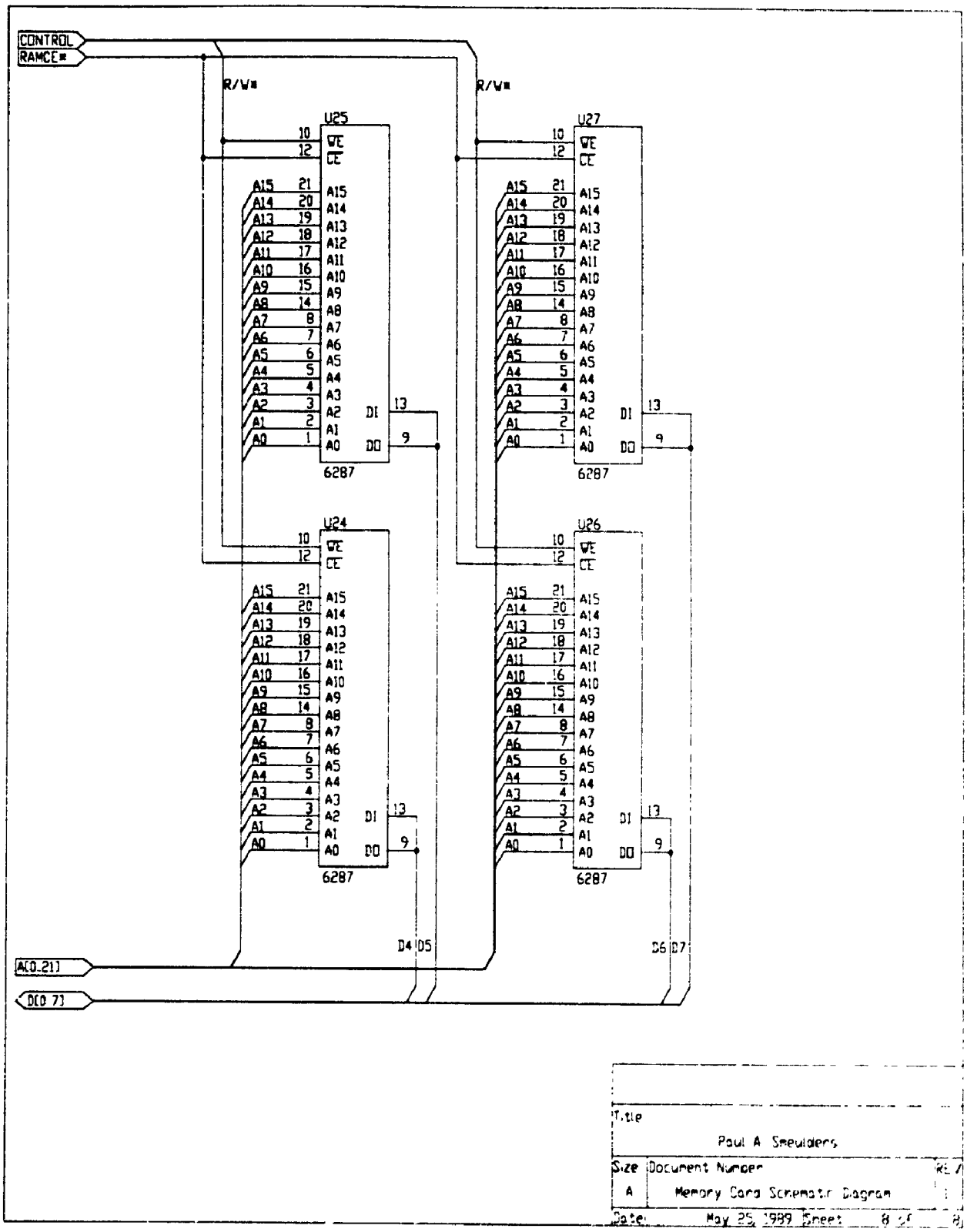


Title	Paul A. Smeulders	
Size: A	Document Number: Memory Card Schematic Diagram	REV: 3
Date: February 26, 1992	Sheet 5 of 8	

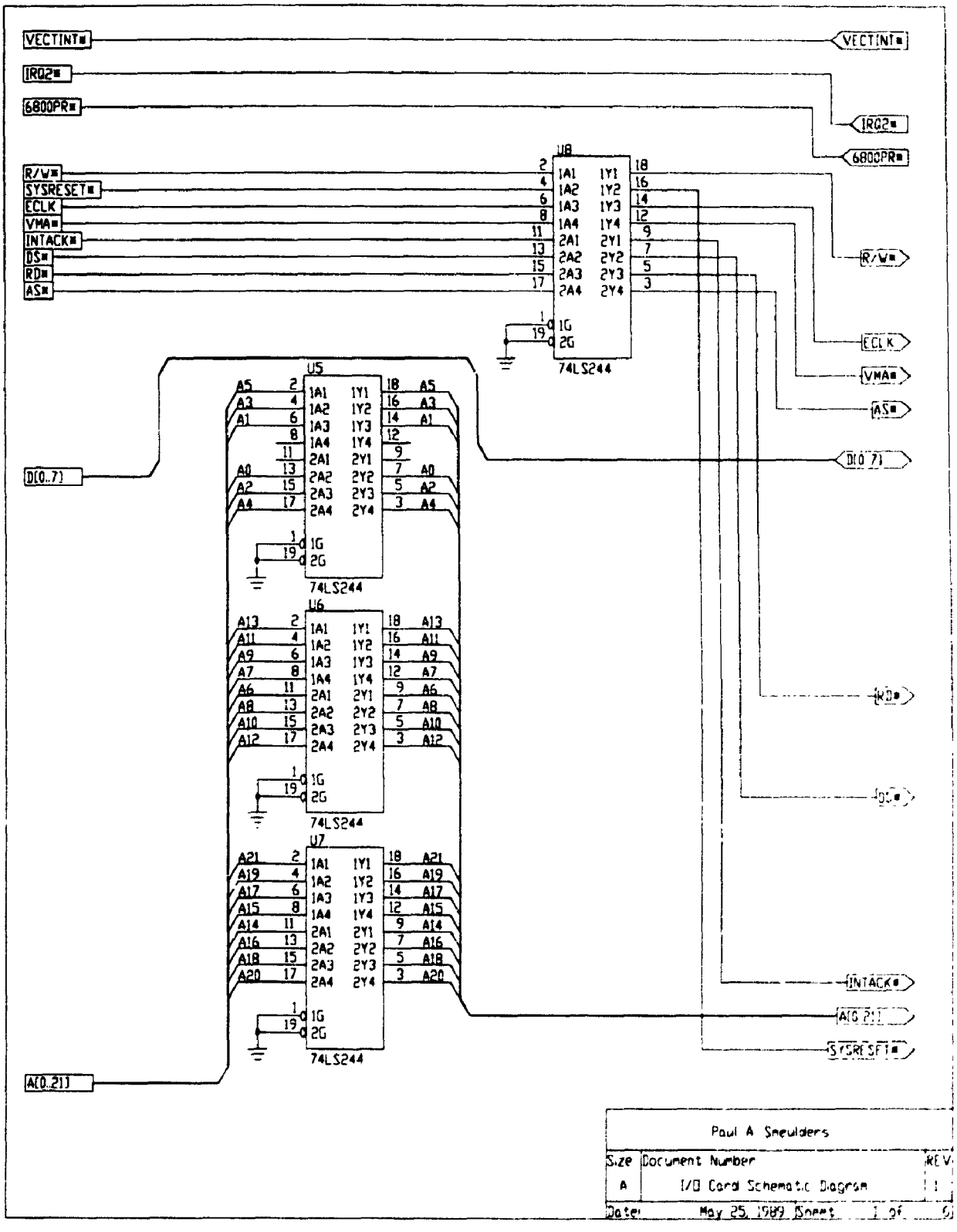

PI/T DIACK*
DMAREQ*
PI/T CS*
ROM2CS*
ROM1CS*
RAMCE*
ROMOE*
CONTROL
R/W*
SYSRESET*
SYSCLK
U31
MC68230Q
R12
1 K
+5V
R7
47 K
+5V0
TOUT
PIRQ*
PIACK*
TIACK*
A[0..21]
D[0..7]
PARALLEL INTERFACE CONNECTOR
PORT A
PORT B
PORT C
14 PIN CONN
U5 8/12
U5 11/9
DUAL DPDT DIP
SW1
SW2
Paul A Smeulders
Parallel Interface/Timer Detail
Memory Card Schematic Diagram
February 26, 1992

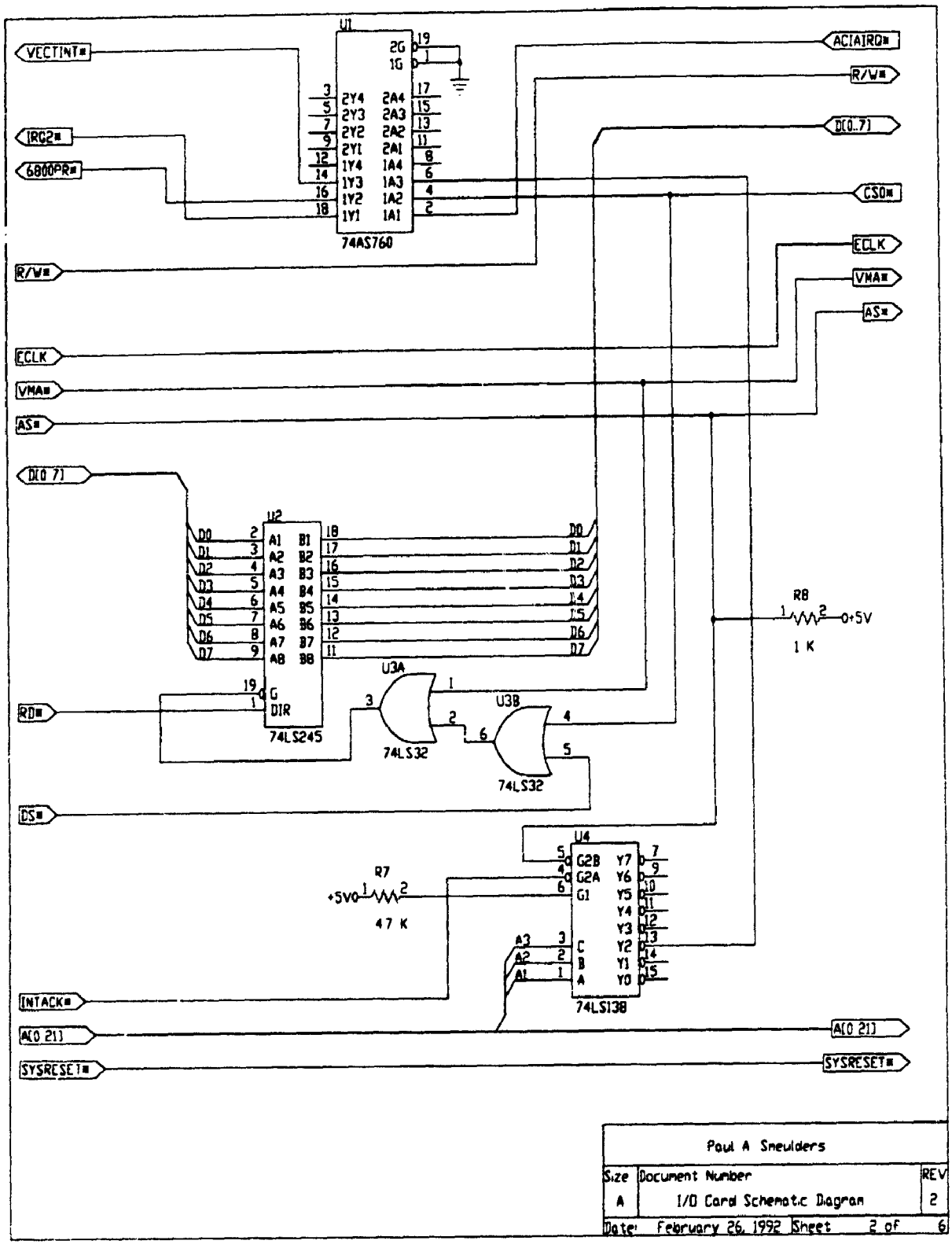


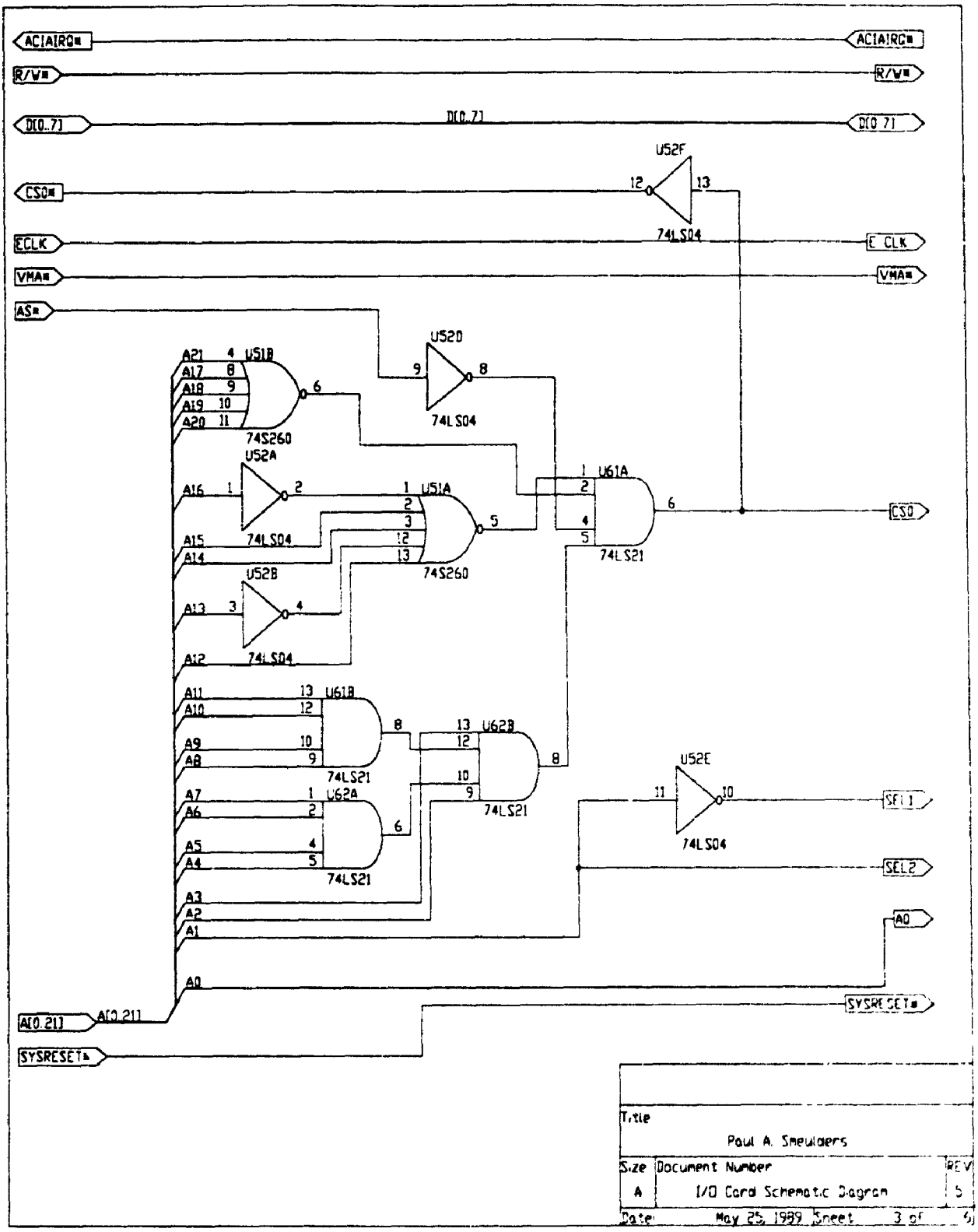
Title	Paul A. Smulders	
Size: A	Document Number: Memory Card Schematic Diagram	REV: 1
Date: May 25, 1989	Sheet 7 of 8	

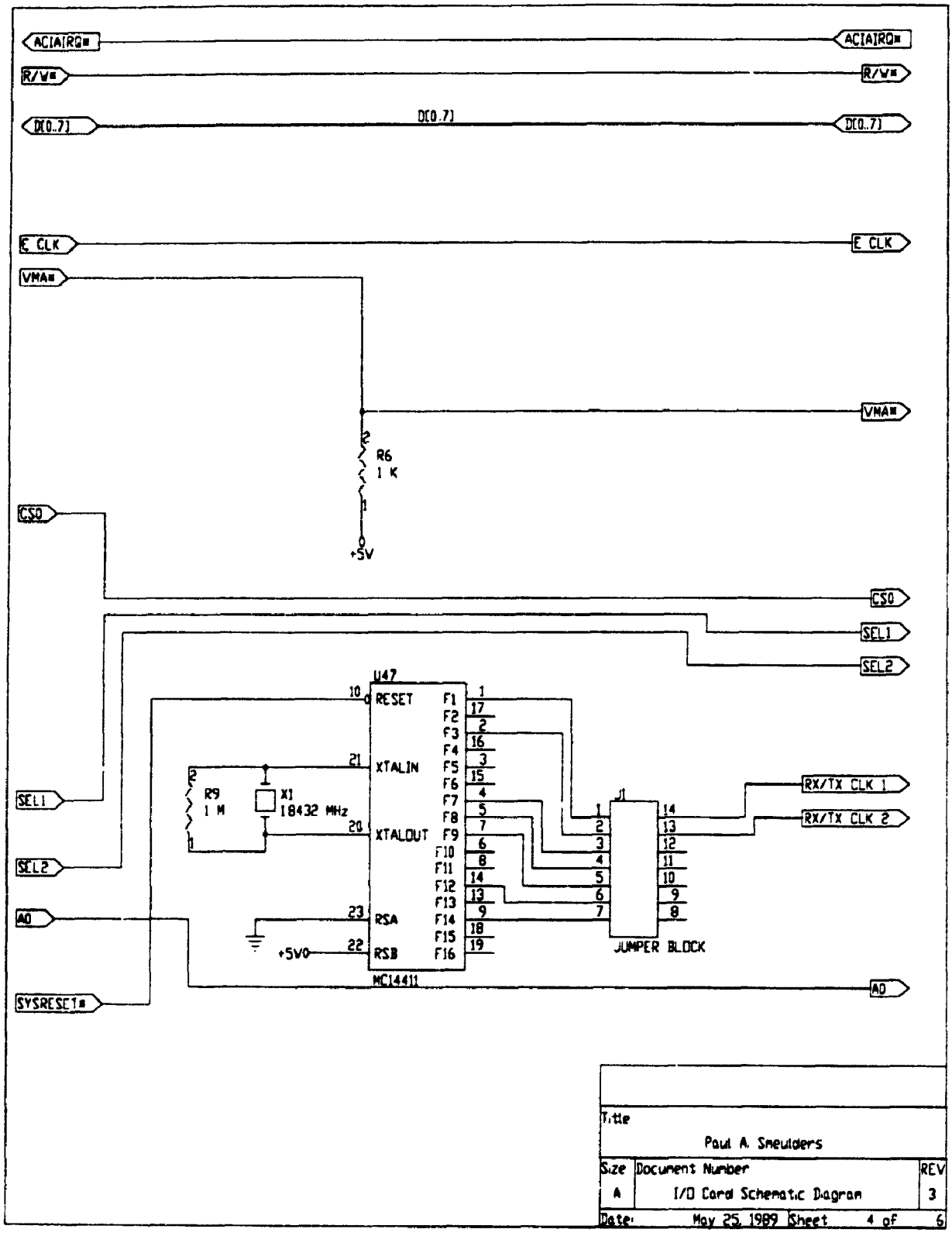

CONTROL
RAMCE*
R/W*
R/W*
U25
U27
U24
U26
6287
WE
CE
DI
DO
A15
A14
A13
A12
A11
A10
A9
A8
A7
A6
A5
A4
A3
A2
A1
A0
D4
D5
D6
D7
A[0..21]
D[0..7]
Paul A Smeulders
Memory Card Schematic Diagram
May 25, 1989

Appendix D: Serial Input/Output Card Schematic Diagrams

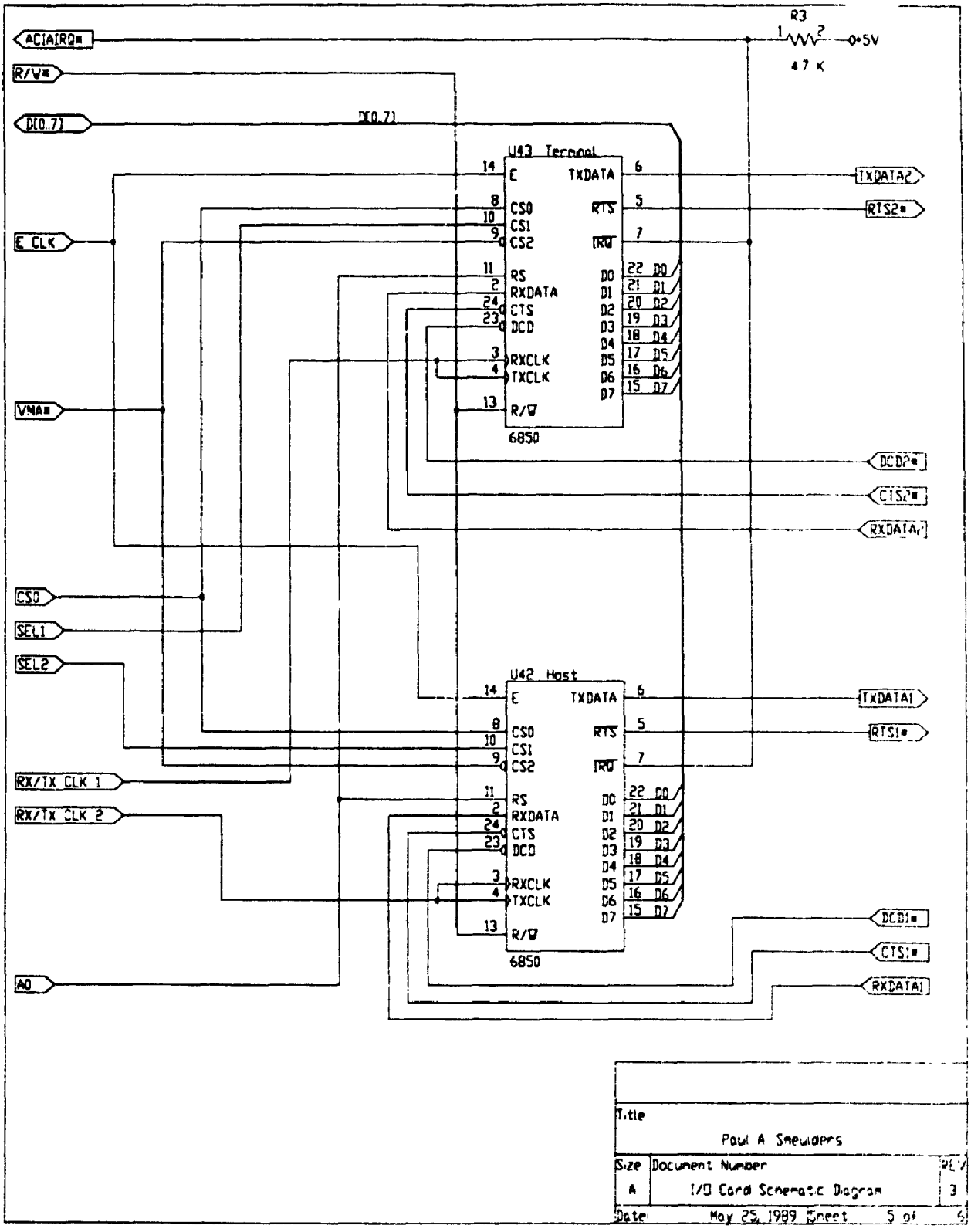

VECTINT*
IRQ2*
6800PR*
R/W*
SYSRESET*
ECLK
VMA*
INTACK*
DS*
RD*
AS*
D[0..7]
A[0..21]
U8
U5
U6
U7
74LS244
1A1 1Y1
1A2 1Y2
1A3 1Y3
1A4 1Y4
2A1 2Y1
2A2 2Y2
2A3 2Y3
2A4 2Y4
1G
2G
Paul A Smeulders
Size
A
Document Number
I/O Card Schematic Diagram
REV
Date: May 25, 1989 Sheet 1 of 6

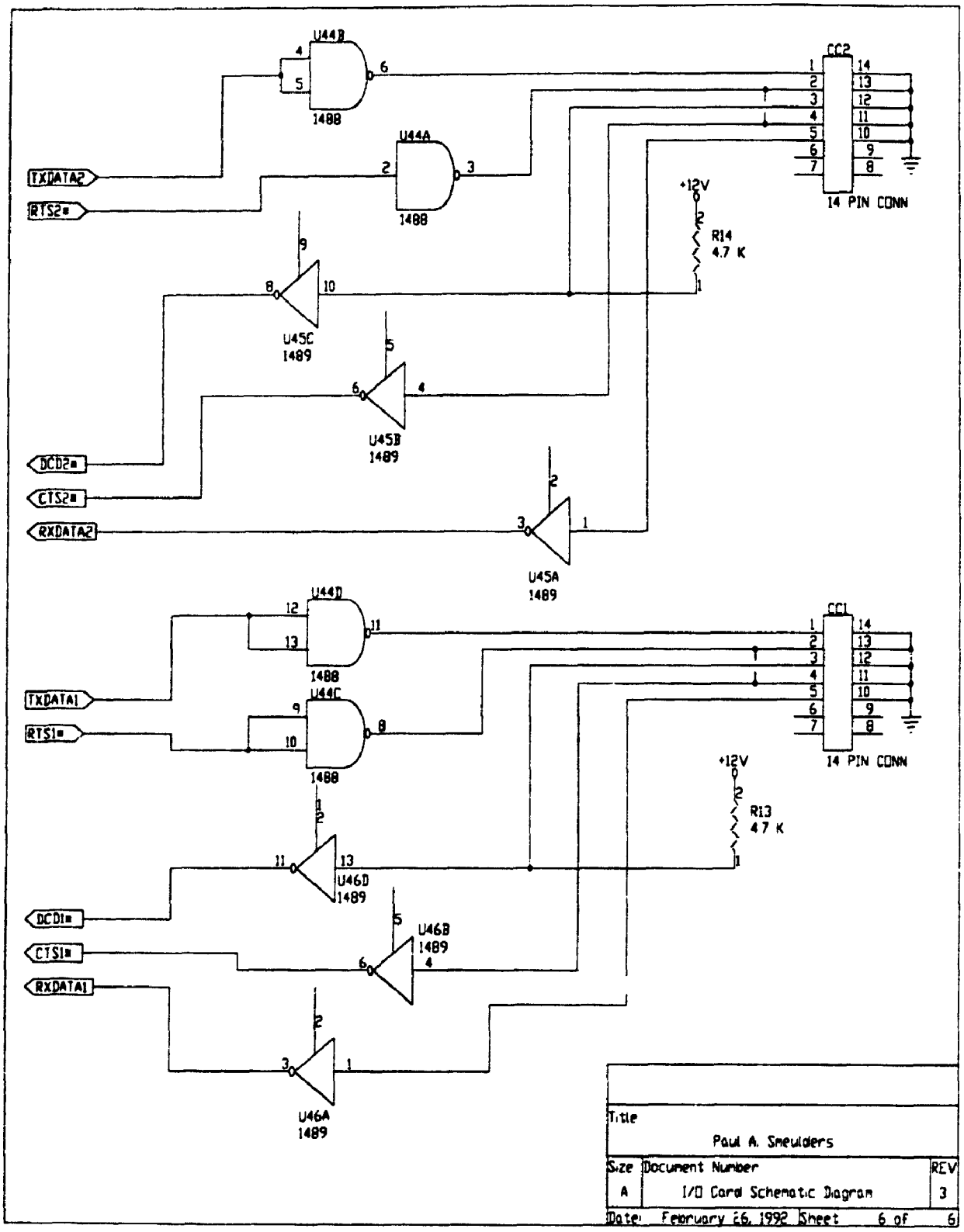

U1
74AS760
VECTINT*
IRQ2*
6800PR*
R/W*
ECLK
VMA*
AS*
D[0 7]
ACIAIRQ*
R/W*
D[0..7]
CS0*
ECLK
VMA*
AS*
U2
74LS245
RD*
U3A
74LS32
U3B
74LS32
R8
+5V
1 K
DS*
U4
74LS138
R7
+5V
47 K
INTACK*
A[0 21]
A[0 21]
SYSRESET*
SYSRESET*
Paul A Sneulders
Size
A
Document Number
I/O Card Schematic Diagram
REV
2
Date: February 26, 1992
Sheet 2 of 6


ACIAIRQ*
R/W*
D[0..7]
CS0*
ECLK
VMA*
AS*
A[0..21]
SYSRESET*
U52F
74LS04
U52D
74LS04
U51B
74S260
U52A
74LS04
U51A
74S260
U52B
74LS04
U61A
74LS21
U61B
74LS21
U62B
74LS21
U62A
74LS21
U52E
74LS04
E CLK
VMA*
CS0
SEL1
SEL2
A0
SYSRESET*
Title
Paul A. Smeulders
Size
A
Document Number
I/O Card Schematic Diagram
REV
5
Date: May 25, 1989
Sheet 3 of 6



Title	Paul A. Sneulders	
Size: A	Document Number: I/O Card Schematic Diagram	REV: 3
Date: May 25, 1989	Sheet 4 of 6	


ACIAIRQ■
R/W■
D[0.7]
E CLK
VMA■
CS0
SEL1
SEL2
RX/TX CLK 1
RX/TX CLK 2
AQ
R3
0-5V
4.7 K
U43 Terminal
U42 Host
6850
E
CS0
CS1
CS2
RS
RXDATA
CTS
DCD
RXCLK
TXCLK
R/W
TXDATA
RTS
IRQ
D0
D1
D2
D3
D4
D5
D6
D7
TXDATA2
RTS2■
DCD2■
CTS2■
RXDATA2
TXDATA1
RTS1■
DCD1■
CTS1■
RXDATA1
Title
Paul A Smeulders
Size
A
Document Number
I/O Card Schematic Diagram
REV
3
Date: May 25, 1989 Sheet 5 of 6

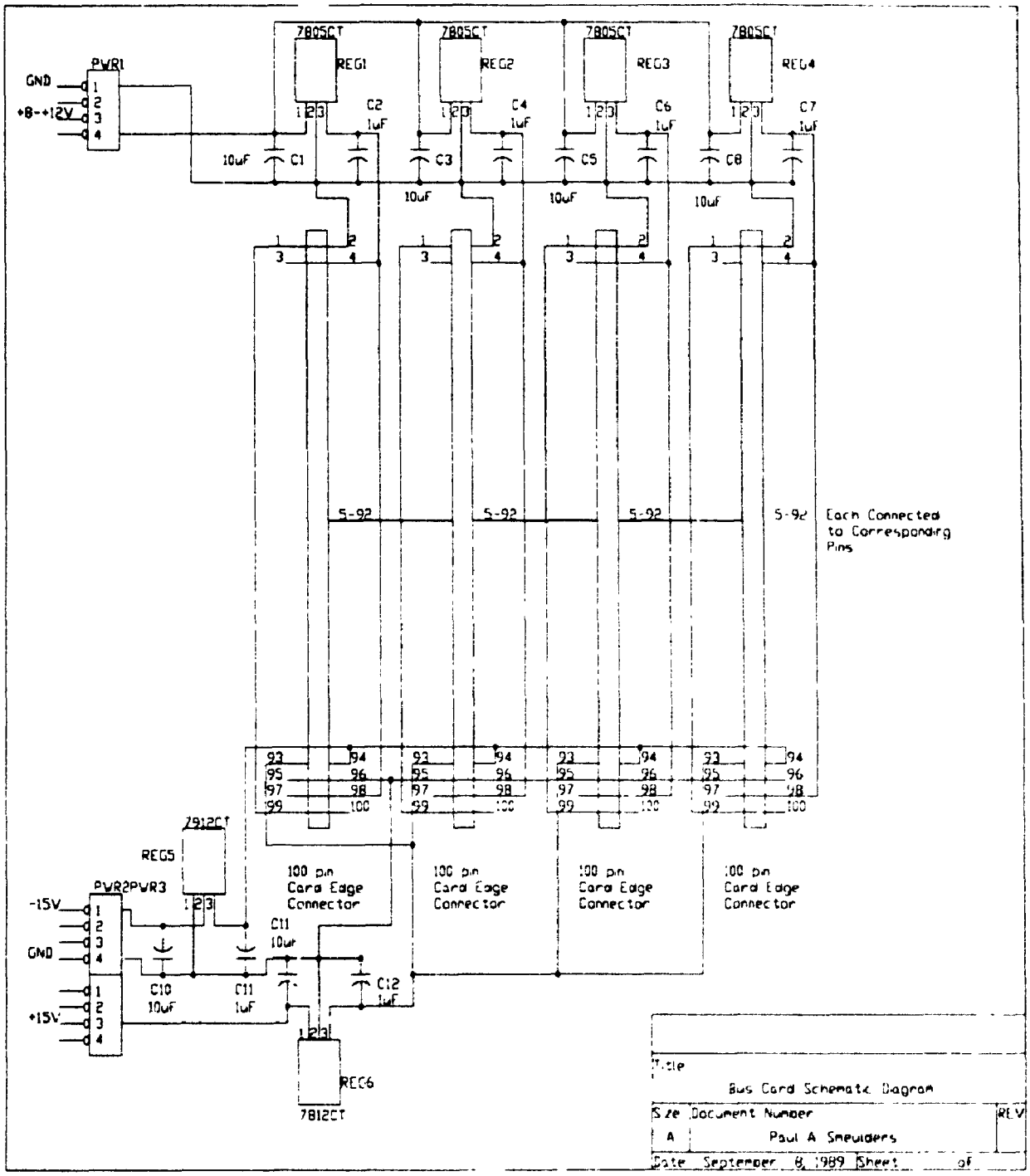

U44B
1488
U44A
1488
CC2
14 PIN CONN
TXDATA2
RTS2#
+12V
R14
4.7 K
U45C
1489
U45B
1489
DCD2#
CTS2#
RXDATA2
U45A
1489
U44D
1488
U44C
1488
CC1
14 PIN CONN
TXDATA1
RTS1#
+12V
R13
4.7 K
U46D
1489
DCD1#
CTS1#
RXDATA1
U46B
1489
U46A
1489
Title
Paul A. Smulders
Size
A
Document Number
I/O Card Schematic Diagram
REV
3
Date: February 26, 1992
Sheet 6 of 6

Appendix E: Printed Circuit Board Designs and Hardware Specifications

Bus Connector Pinout Specification

Pin	Mnemonic	Direction	Type	Pin	Mnemonic	Direction	Type
1	GND	I	POWER	2	GND	I	POWER
3	+5VDC	I	POWER	4	+5VDC	I	POWER
5				6			
7	D0	I/O	3 state	8	D4	I/O	3 state
9	D1	I/O	3 state	10	D5	I/O	3 state
11	D2	I/O	3 state	12	D6	I/O	3 state
13	D3	I/O	3 state	14	D7	I/O	3 state
15	A10	O	3 state	16	A21	O	3 state
17	A9	O	3 state	18	A20	O	3 state
19	A8	O	3 state	20	A19	O	3 state
21	A7	O	3 state	22	A18	O	3 state
23	A6	O	3 state	24	A17	O	3 state
25	A5	O	3 state	26	A16	O	3 state
27	A4	O	3 state	28	A15	O	3 state
29	A3	O	3 state	30	A14	O	3 state
31	A2	O	3 state	32	A13	O	3 state
33	A1	O	3 state	34	A12	O	3 state
35	A0	O	3 state	36	A11	O	3 state
37	RD*	O	3 state	38	R/W*	O	3 state
39	DS*	O	3 state	40	AS*	O	3 state
41	UDS*	O	3 state	42	LDS*	O	3 state
43	SYSCLK	O	3 state	44	SYSRESET*	O	3 state
45	EXRESET*	I	2 st OC	46	ECLK	O	3 state
47	DTACK*	I	2 st OC	48	EXVPA*	I	3 st. OC
49				50			
51				52			
53	BR*	I	2 st OC	54		O	3 state
55	BGACK*	I	2 st OC	56	VMA*	O	3 state
57	HALTIN*	I	2 st OC	58	HALTOUT	O	3 state
59	BERRIN*	I	2 st OC	60	MAP*	O	3 state
61	USDAT*	O	3 state	62	USPRO*	O	3 state
63	SUPDAT*	O	3 state	64	SUPPRO*	O	3 state
65	IRQ1*	I	2 st OC	66	IRQ2*	I	2 st OC
67	IRQ3*	I	2 st OC	68	IRQ4*	I	2 st OC
69	IRQ5*	I	2 st OC	70	IRQ6*	I	2 st OC
71	NMI*	I	2 st OC	72	6800PR*	I	2 st OC
73	INTACK*	O	3 state	74	VECTINT*	I	2 st OC
75	SYSTSE*	O	3 state	76	ADATSE*	O	3 state
77	KILLSYN*	I	2 st OC	78			
79	DMA0*			80	DMA1*		
81	DMA2*			82	DMA3*		
83				84			
85	BG*	O	3 state	86	BGCHI*	I (slave)	2 st OC
87	DACK0*			88	DACK1*		
89	DACK2*			90	DACK3*		
91				92			
93	+12VDC		POWER	94	-12VDC		POWER
95	GND		POWER	96	GND		POWER
97	+5 VDC		POWER	98	+5 VDC		POWER
99	GND		POWER	100	GND		POWER

Bus Card schematic diagram



Bus Card Parts List

Part Reference	Part Specification, Order Number	Quantity
REG1-REG4	LM7805CT regulator, TO-220 Pkg.	4
REG5	LM7812CT regulator, TO-220 Pkg.	1
REG6	LM7912CT regulator, TO-220 Pkg.	1
C1, C3, C5, C8, C10, C11	10 µF 35V tantalum capacitor	6
C2, C4, C6, C7, C9, C12	0.1 µF 35V tantalum capacitor	6
PWR1	8-pin header connector, 4 pins used, Panduit MPSS100-8-1	1
PWR2PWR3	8-pin header connector, 8 pins used, Panduit MPSS100-8-1	1
Card Edge Connectors	EDAC 345-100-520-202 100 pin connector	3
Heat Sinks	for TO-220 package	4
Female Header Connectors	Panduit CE100F22-2-1 (wired for +8V and GND to power supply)	2
Female Header Connectors	Panduit CE100F22-8-1 (wired pin 1 to -15V, pin 4 to gnd, pin 7 to +15V)	1
Header Connector Covers	Panduit EC100-8-1	2
Mounting Hardware	screws and nuts etc., as required by mounting chassis of choice	

CPU Card Parts List

Part Reference	Part Specification, Order Number	Quantity
U1, U19	74LS73	2
U2	74LS273	1
U3	74LS148	1
U4, U14	74LS175	2
U5	MC68008FN8	1
U6	16 MHz oscillator, M-TRON MTO-T1-S3-16.000000	1
U7	74LS93	1
U8, U27	74LS04	2
U9	MC3456	1
U10, U34, U35	74LS00	3
U11	74LS05	1
U12	74LS164	1
U15, U32	74LS08	2
U16, U17	74LS138	2
U13, U20, U21, U23, U24, U25, U37, U38	74LS244	8
U22	74LS245	1
U26	74LS32	1
U29, U31, U33	74LS74	3
U30	74HC04	1
CA1-CA4, CA15	10 µF 35V tantalum capacitor	5
CA5-CA14, CC1-CC13	1 µF 35V tantalum capacitor	24
CB1, CB2, CA16	0.1 µF ceramic capacitor	3
R1-R6, R9-R13, R24-R27	SIP resistor network 4608X-101-472	3
R7, R8	1 MΩ, 1/4 Watt resistor	2
R14-R21	SIP resistor network 4610X-101-472	1
R22, R23	1 KΩ, 1/4 Watt resistor	2
R37-R44, R28-R36	SIP resistor network 4610X-101-102	2
R45	470Ω, 1/4 Watt resistor	1
RESET, ABORT	Grayhill PC mount pushbutton switches, right angle. 39-201R (RESET), 39-201B (ABORT)	2
14 pin DIP sockets	Texas Instruments TI C-8414-02	19
16 pin DIP sockets	Texas Instruments TI C-8416-02	5
20 pin DIP sockets	Texas Instruments TI C-8420-02	10
52 pin PLCC socket	AMP 821551-1 52 pin PLCC socket	1

Memory & PI/T Card Parts List

Part Reference	Part Specification, Order Number	Quantity
U1	74AS760	1
U2	74LS245	1
U3, U28	74LS260	2
U4	74LS04	1
U5, U6, U7, U8	74LS244	4
U9, U33	74LS20	2
U10	74LS00	1
U11	74LS08	1
U13	74LS21	1
U14	74LS32	1
U15, U16	TMS 2732A-20JL EPROMS (or up to 45JL)	2
U17	74LS175	1
U18	74LS27	1
U19	74LS74	1
U20-U27	MC6287 SRAM	8
U30	74LS133	1
U31	MC68230FN8	1
U32	74LS138	1
U34	74LS86	1
R12	1 KΩ 1/4 Watt resistor	1
R1-R7	SIP resistor network, 4608X-101-472	1
CA10, CA11, C12, C13	10 µF tantalum capacitors	4
CAJ	0.1µF ceramic capacitor	1
CA1-CA9, CA14-CA23, CAA-CAI	1 µF tantalum capacitor	28
SW1, SW2	Grayhill 2 rocker DPDT w/ raised rockers, 76SD02	2
PORT A, PORT B, PORT C	Scotchflex 4 wall headers, right angled with long ejectors: 3314-5302, and female mates 3385-6014, with strain relief.	3 ea type.
	14 conductor ribbon cable	2 m
14 pin DIP sockets	Texas Instruments TI C-8414-02	12
16 pin DIP sockets	Texas Instruments TI C-8416-02	3
20 pin DIP sockets	Texas Instruments TI C-8420-02	6
22 pin DIP sockets	Texas Instruments TI C-8422-02 (remove crossbar, use a SIP sockets)	8
52 pin PLCC socket	AMP 821551-1 52 pin PLCC socket	1
	mounting hardware: screws, nuts	6

Serial I/O Card Parts List

Part Reference	Part Specification, Order Number	Quantity
U1	74AS760	1
U2	74LS245	1
U3	74LS32	1
U4	74LS138	1
U5, U6, U7, U8	74LS244	4
U42, U43	MC68B50P	2
U44	MC1488	1
U45, U46	MC1489	2
U47	MC14411	1
U51	74LS260	1
U52	74LS04	1
U61,U62	74LS21	2
X1	M-TRON MP2-2-1.843200 1.843200 MHz crystal	1
R9	1 MΩ 1/4 Watt resistor	1
R3, R7, R13, R14	4.7 KΩ 1/4 Watt resistor	4
R6, R8	1.0 KΩ 1/4 Watt resistor	2
CA10, CA11, C12-C15, CA14, CA17	10 µF tantalum capacitor	
CA1-CA9, C10, C11, CA16	1 µF tantalum capacitor	
CA18, CA19	0.1 µF ceramic capacitor	
14 pin DIP sockets	Texas Instruments TI C-8414-02	8
16 pin DIP sockets	Texas Instruments TI C-8416-02	1
20 pin DIP sockets	Texas Instruments TI C-8420-02	6
24 pin DIP sockets	Texas Instruments TI C-8424-02	3
CC1, CC2	Scotchflex 4 wall headers, right angled with long ejectors: 3314-5302, and female mates 3385-6014, with strain relief.	2 ea type.
	14 conductor ribbon cable	2 m
	25 pin D-subminiature connector shells	2
	25 pin D-subminiature male connector	1
	25 Pin D-subminiature Female Connector	1
	mounting hardware: screws, nuts	3

Barnacle Notes

NOTE: Schematic diagrams presented in the thesis are correct. Barnacles are required on the PC boards to exactly match circuit design.

Bus Card:

- No Barnacles
- Do not allow mounting bolt for "Controller Slot" card edge connector to short circuit the traces on the solder side of the board. Use an insulated bolt, or trim the traces before mounting.

CPU Card:

- Break trace connecting Pin 1 (common pin) of Rnet R24-R27 to the GND trace on the component side of the board. Connect barnacle wire from pin 1 of Rnet R24-R27 to the VCC pin (topmost) of CC1.
- The holes to the RESET and ABORT switches may need to be widened. This will remove the plating in the hole, but it is of no consequence, since all connections are made on the solder side to these components.
- The plating inside the holes of U5 most upper and left MUST be removed. These are extraneous holes caused by deficiencies in the layout tools, but traces were routed through them on both sides, and should not connect. Use a 0.050" drill bit to break the side to side connection.
- Solder in place R45 on non-component side.
- For slave cells, cut traces to pull-ups for NMI* and EXRESET*, and use barnacle to bypass the resistors.
- To drive slave NMI* and EXRESET* signals, a 74LS244 driver is employed.

Serial I/O Card:

- If 2 wire communications is desired (Tx and Rx on the RS232 ports), the RTS output must be connected to the CTS input. Place barnacle from Pin 2 to Pin 4 of the CC1 connector, and from Pin 2 to Pin 4 of the CC2 connector.
- Drill holes for communication port connectors after connectors are in place. DO NOT Drill a hole for the leftmost hole in the CC1 connector.
- The card was shipped slightly to tall. Sheer off the top (non-bus side) of the PC board to ensure all cards are the same height, making for easier mounting inside a hardware cabinet.
- Host connector connects to CC1. Use MALE 25 pin D-subminiature connector, and connect Tx to pin 2, Rx to pin 3, GND to pin 7. For IBM PC communications, connect pin 4 to pin 5, and connect pins 6,8 and 20 together.
- Terminal connector connects to CC2. Use FEMALE 25 pin D-subminiature connector, and connect Tx to pin 3, Rx to pin 2, GND to pin 7.
- Baud rate selection:
  - Header JA pin 1 is for Terminal Rx/Tx Clock
  - Header JA pin 2 is for Host Rx/Tx Clock
  - Pin numbering is left to right
  - Header JB:

Pin #	Frequency
1	153.6 kHz
2	76.8 kHz
3	19.2 kHz
4	9.6 kHz
5	4.8 kHz
6	1.2 kHz
7	2.4 kHz

Use wire wrap wire to connect appropriate baud rate generator pin to Rx/Tx clocks.

Memory and PI/T Card

- Barnacle GND side of CAA to GND side of CA8, and CAA GND to U20 pin 11
- Cut the trace on the SOLDER side of the card to U33 pin 4, close to the pin. Break the trace connecting U9 pin 5 to U3 pin 5 (use vertical trace between chips, see diagrams). Use barnacles to connect U3 pin 5 to U33 pin 4, and connect U3 pin 6 to U9 pin 5.
- Cut traces on the SOLDER side of the card just above AND below U13 pins 9 and 10. Use barnacles to connect the following sets of pins:
  - R2 to U1 pin 8
  - R3 to U1 pin 13
  - U13 pin 9 to U32 pin 12
  - U13 pin 10 to U32 pin 14
- Cut trace on SOLDER side to U31 pin 45 above the via hole to the left of and between U19 and U28. Cut the trace on the COMPONENT side to U28 pin 1, between the pin and the via hole.
  - U28 pin 1 to U28 pin 2
  - U13 pin 12 OR 13 to U14 pin 2
  - U14 pin 3 to U31 pin 45
  - U11 pin 2 to U14 pin 1
- Cut trace to pin 14 of ports A and B. Use barnacle to connect pins 6,8,9 and 14 to GND for both ports. Use barnacles to connect H2 and H4 to 74LS244 driver (U5), as shown in schematics.

Figure E.1 Connector pin numbering conventions

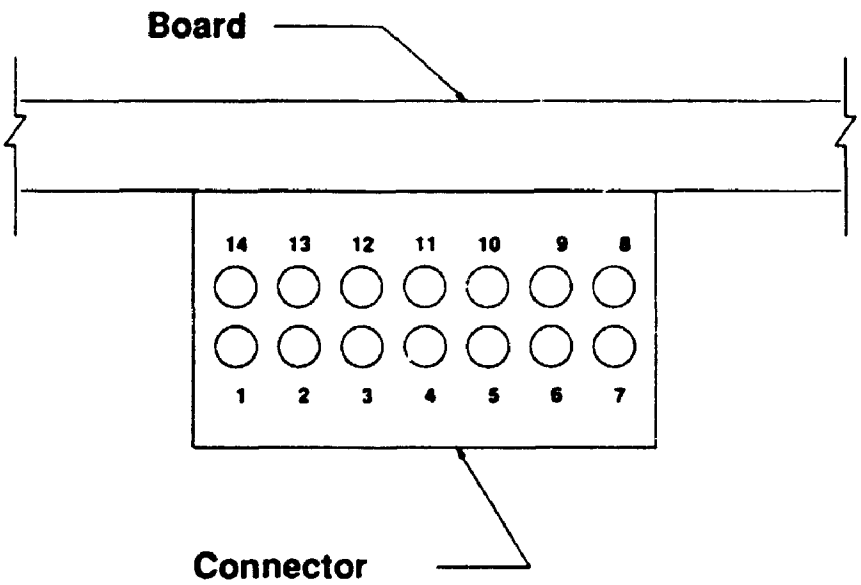


Figure E.2 Serial port ribbon cable conductor assignments

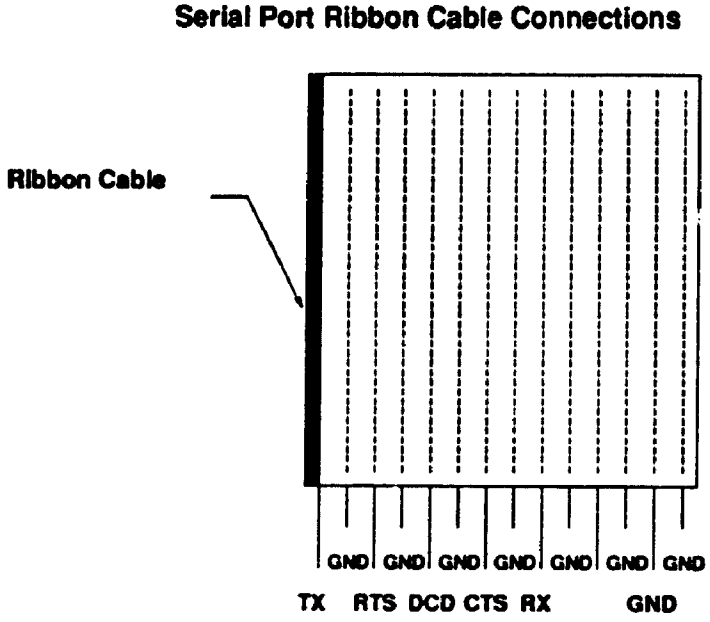


Figure E.3 Parallel port ribbon cable conductor assignments

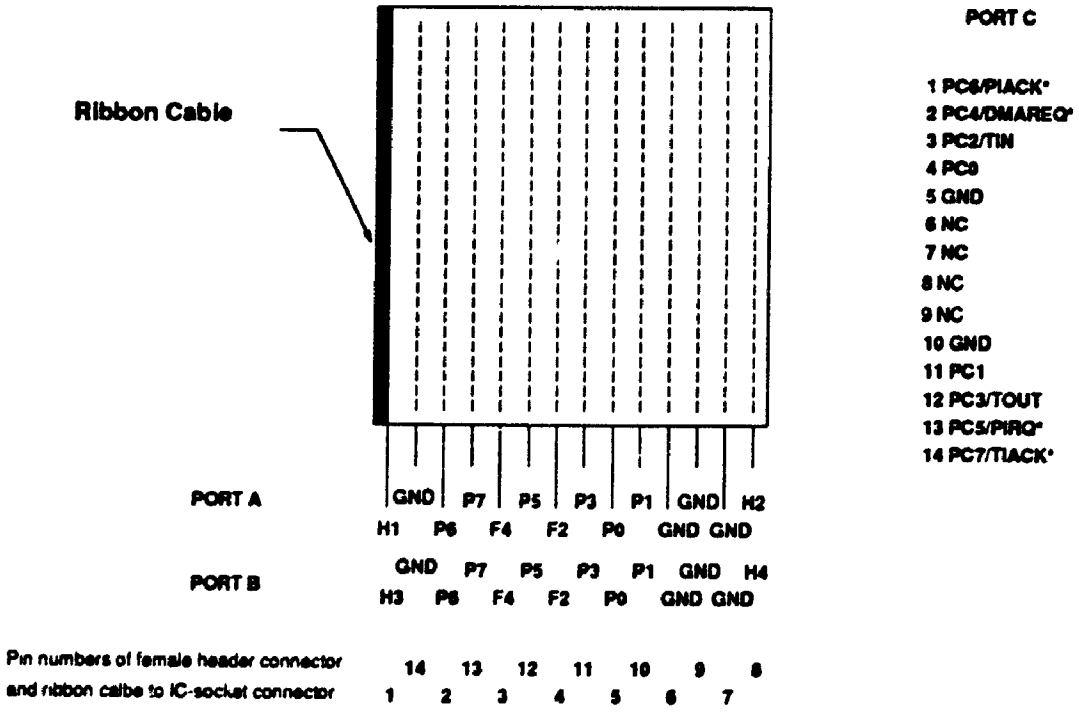
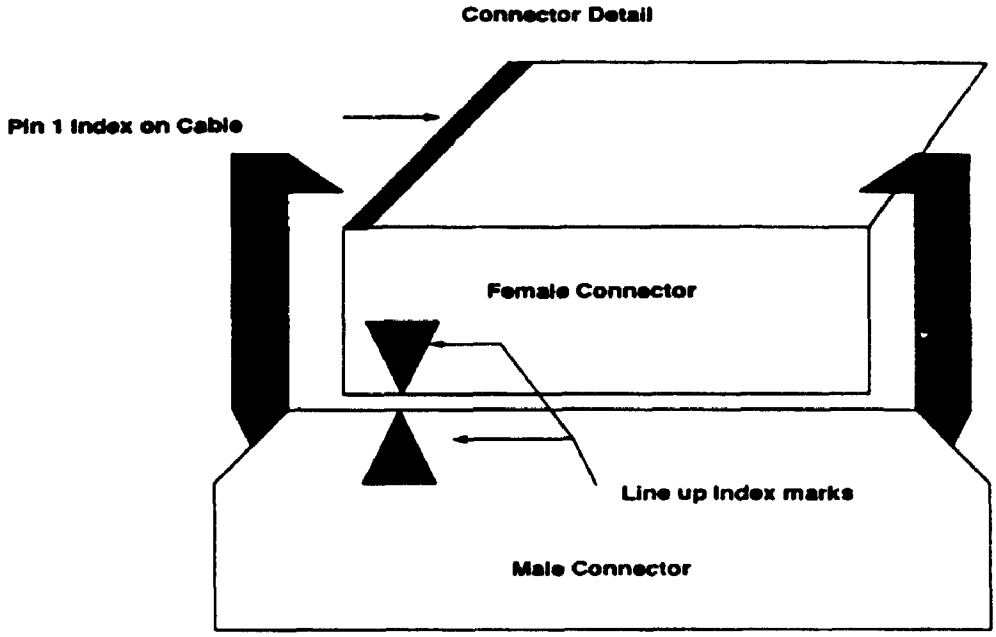


Figure E.4 Connector indexing

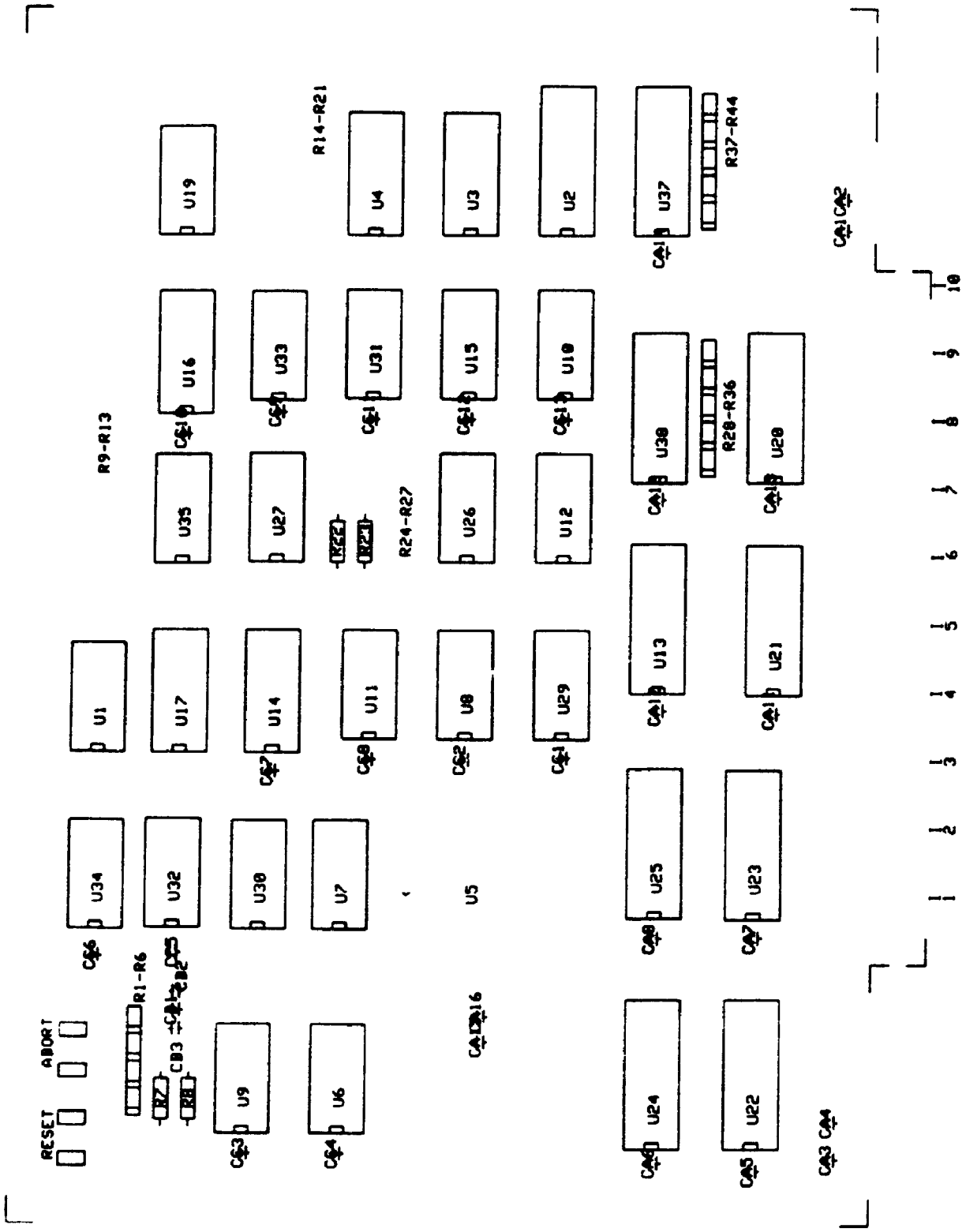


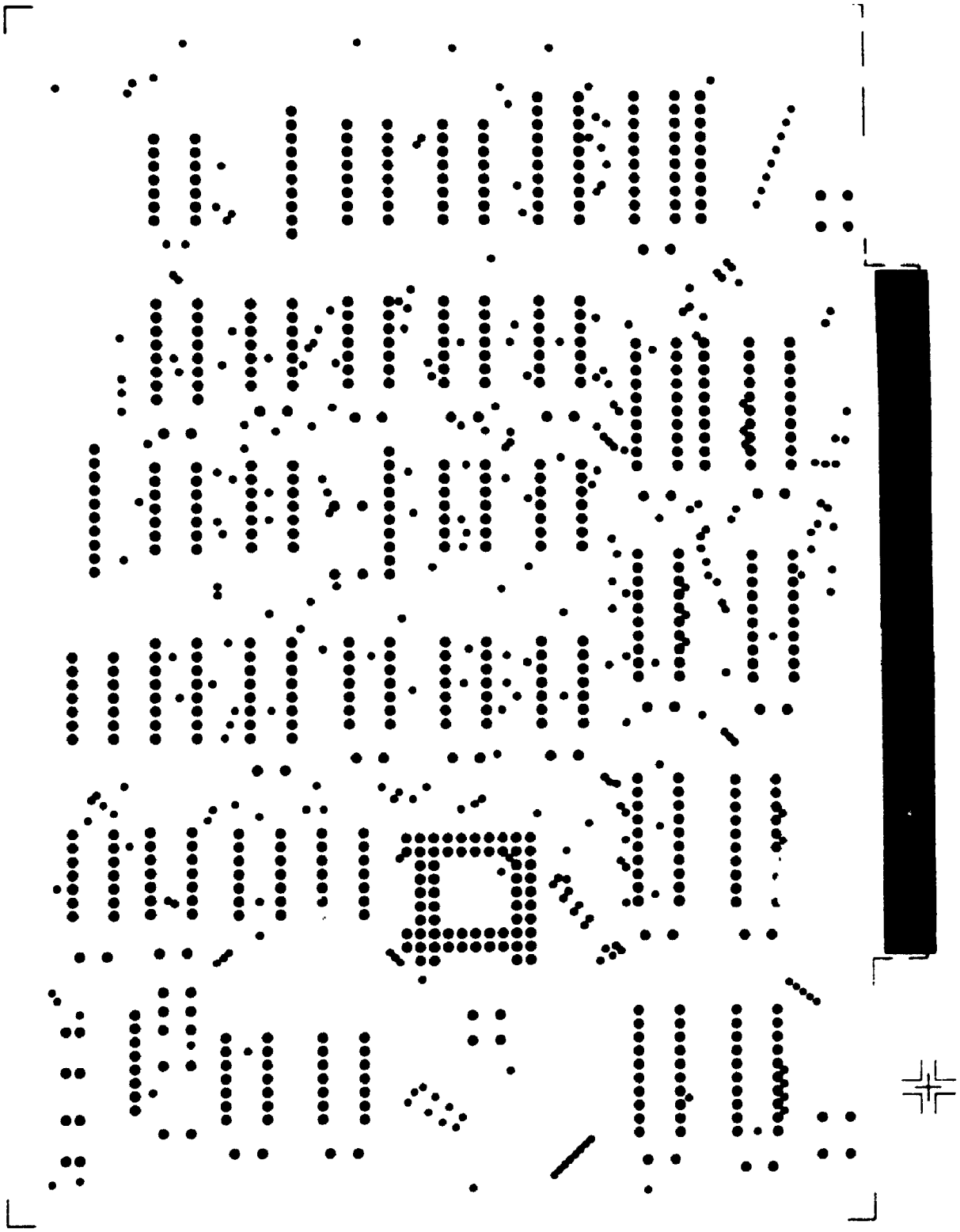
Network Controller Card Parts List

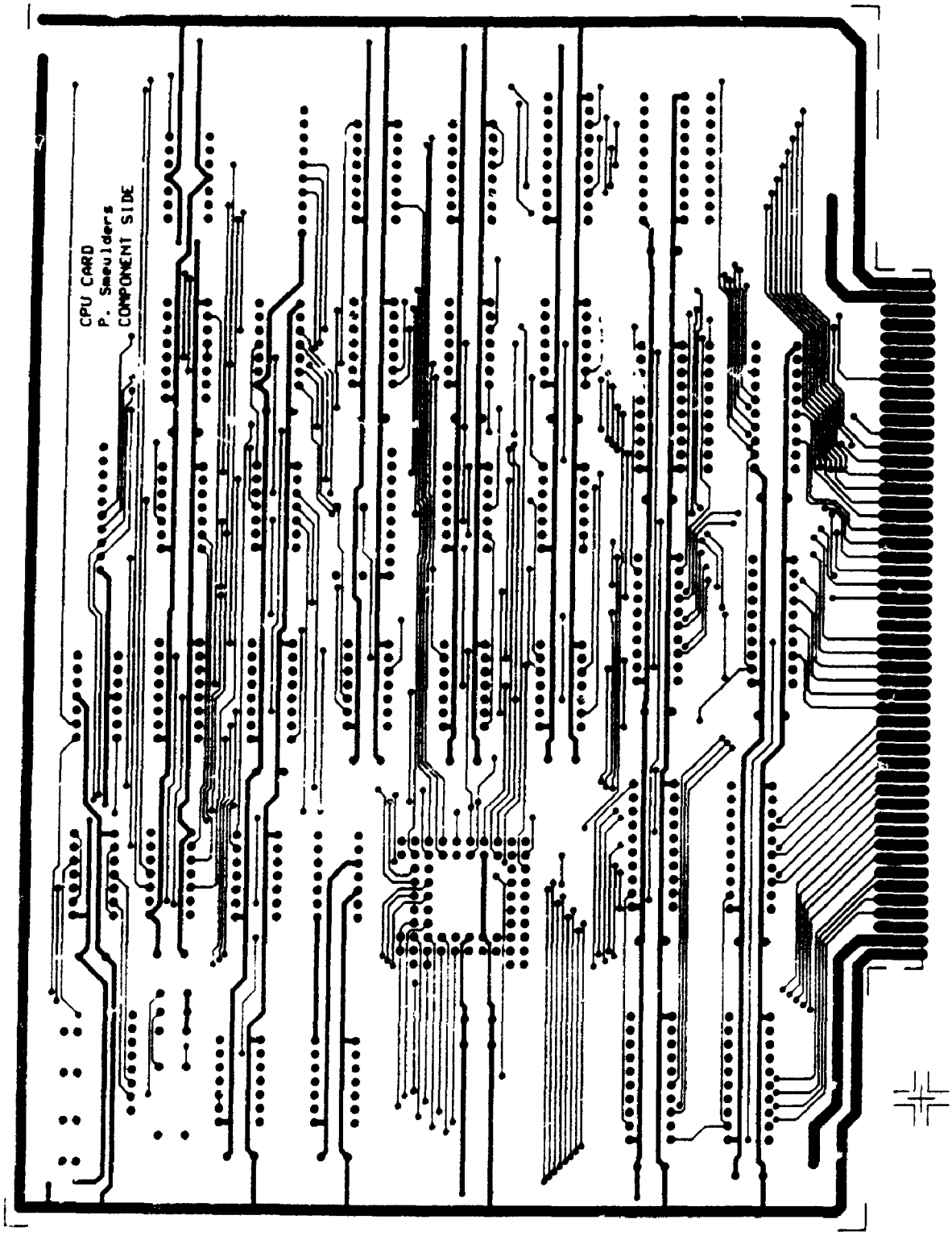
Part Reference	Part Specification, Order Number	Quantity
U6-U13	74ALS666	10
U16	74LS245	1
U17-U20	74LS244	4
U21-U24	74LS138	4
U25, U26, U33	74LS04	3
U27	74LS133	1
U28	74S260	1
U29, U50-U53	74LS74	5
U30, U47	74AS760	2
U31, U49	74LS00	2
U32	74LS11	1
U35, U40, U42, U46	74LS73	4
U36, U43	74LS139	2
U37, U44	74LS08	2
	14-pin wire-wrap sockets	20
	16-pin wire-wrap sockets	7
	20-pin wire-wrap sockets	14
	24-pin wire-wrap sockets (0.3 inch width)	10
JP1-JP10	40-pin wire-wrap dip socket	2
U34, U39, U41, U45, U48	Scotchflex 4 wall Headers, Right Angled with long Ejectors: 3314-5302, and Female Mates 3385-6014, with strain relief.	5 ea type.
R13	27 KΩ 1/4 Watt Resistor	1
R22, R23, R35, R36	1.0 KΩ 1/4 Watt Resistor	4
SW1, SW2	Microswitch momentary action SPDT pushbuttons 8N1021 with operators 8Z0062,8Z0063	2
System power switch	Microswitch Power-Duty alternate action DPST-DB N.O. pushbuttons AML31EBA4AD with button AML51F1OR	1
	10 µF Tantalum Capacitor	2
	1 µF Tantalum Capacitor	20
	0.1 µF Ceramic Capacitor	20
	Wrap-posts Vector T68	as req'd
	Vector wire-wrap perfboard, epoxy glass, 0.1" hole spacing, non clad	22.5 x 16 cm
	Cellbus card-edge male PCB to wire-wrap adapter, custom part, 100 contacts on .1" spacing	1

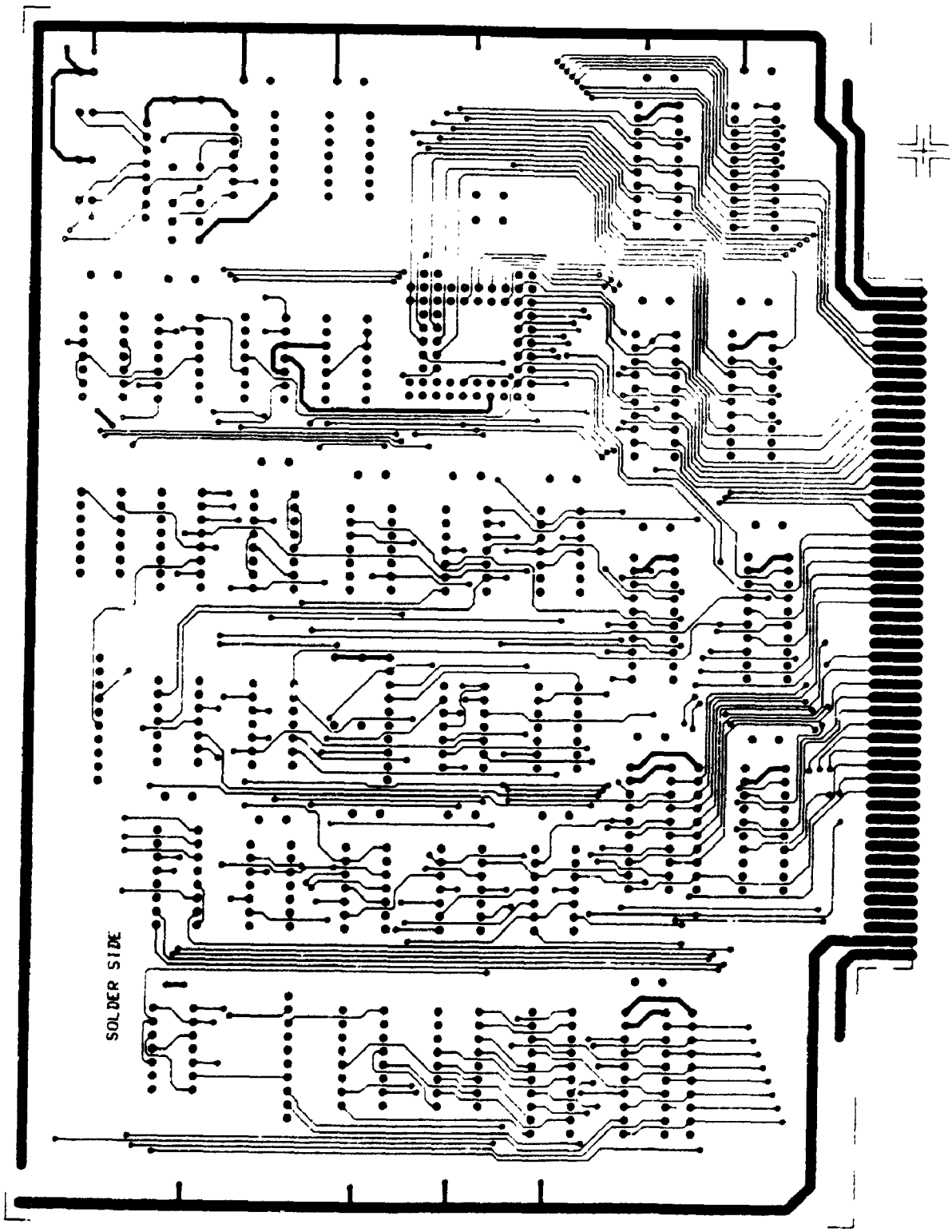
Interprocessor Communication Network Parts List

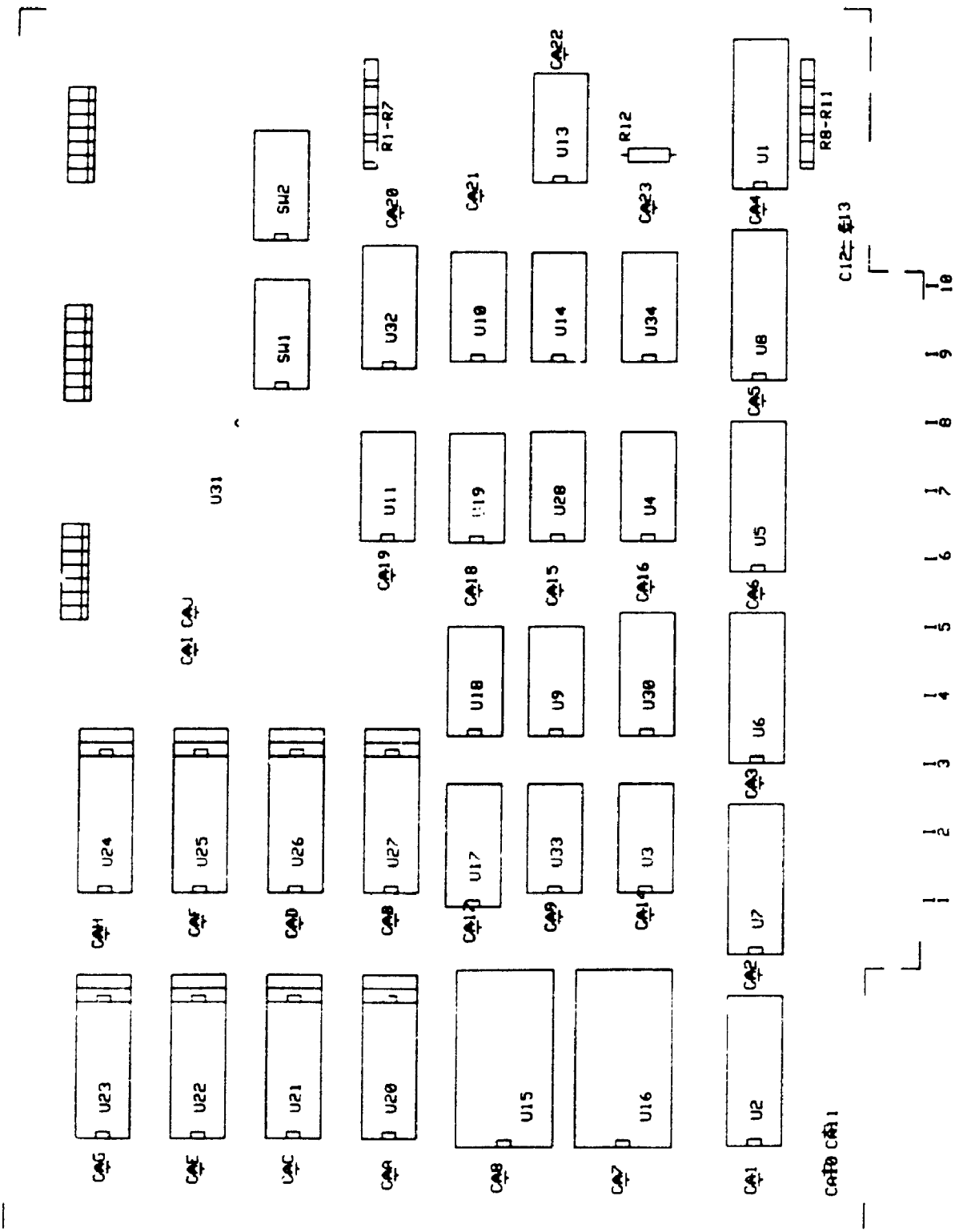
Part Reference	Part Specification, Order Number	Quantity
U0-U19	WECBS2	20
U20-U37	74LS244	18
JP1-JP20	40-pin wire-wrap socket	2
CMI, CMO, C0I, C0O, C1I, C1O, C2I, C2O, C3I, C3O	14-pin wire-wrap socket	10
R	SIP resistor network 4610X-101-331	13
	10 μF Tantalum Capacitor	2
	1 μF Tantalum Capacitor	38
	0.1 μF Ceramic Capacitor	38
	Ansley ribbon cable to dip socket connectors, 14-pin 609-M145H	10
	Wrap-posts with screw machine contact Vector R32	1000
	Wrap-posts Vector T68	as req'd
	20-pin wire-wrap sockets	18
	Vector wire-wrap perfboard, epoxy glass, 0.1" hole spacing, non-clad	22.5 x 20 cm

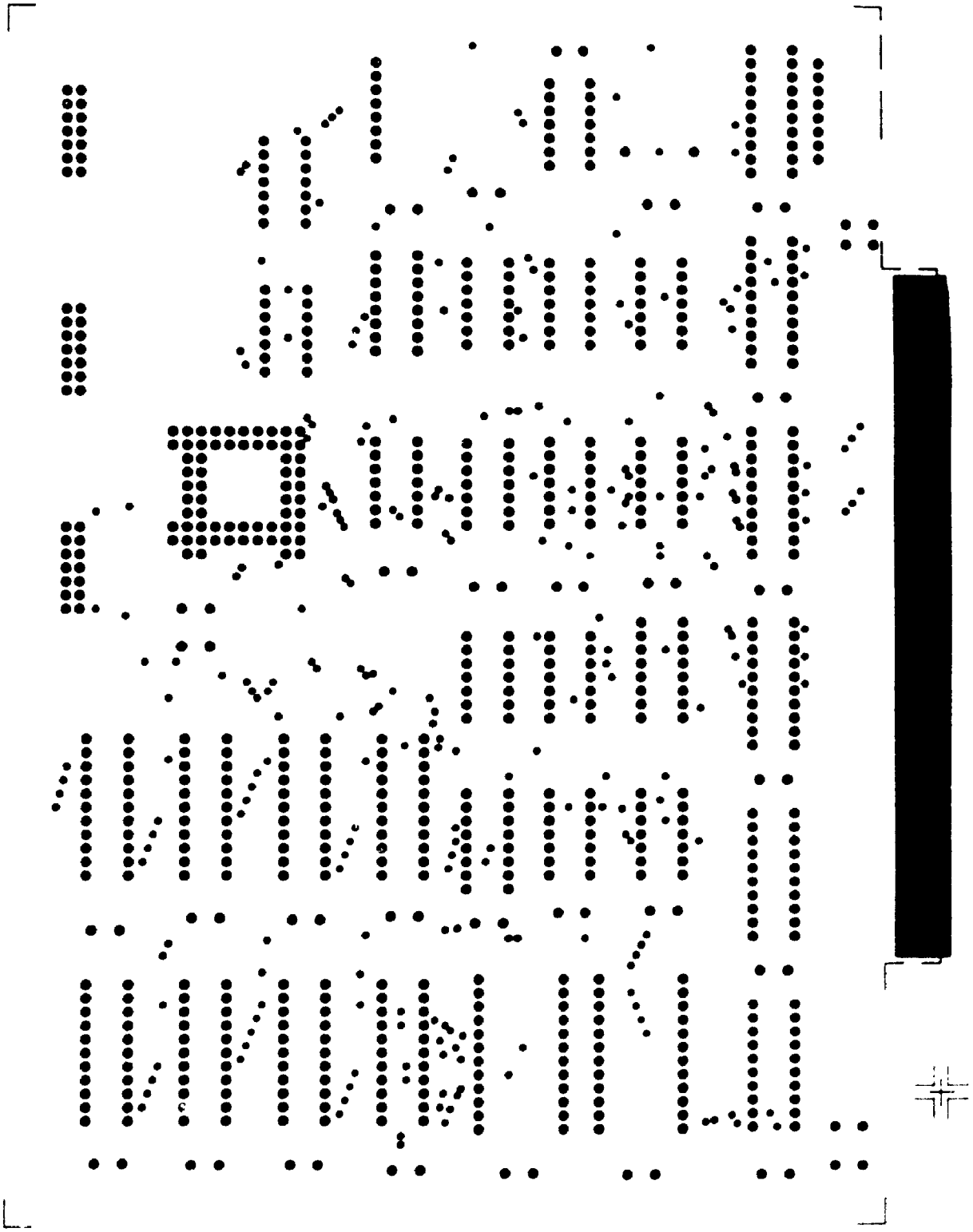

RESET
ABORT
R1-R6
R9-R13
R14-R21
R24-R27
R37-R44
R28-R36
U1
U2
U3
U4
U5
U6
U7
U8
U9
U10
U11
U12
U13
U14
U15
U16
U17
U18
U19
U20
U21
U22
U23
U24
U25
U26
U27
U28
U29
U30
U31
U32
U33
U34
U35
U36
U37
U38
1
2
3
4
5
6
7
8
9
10

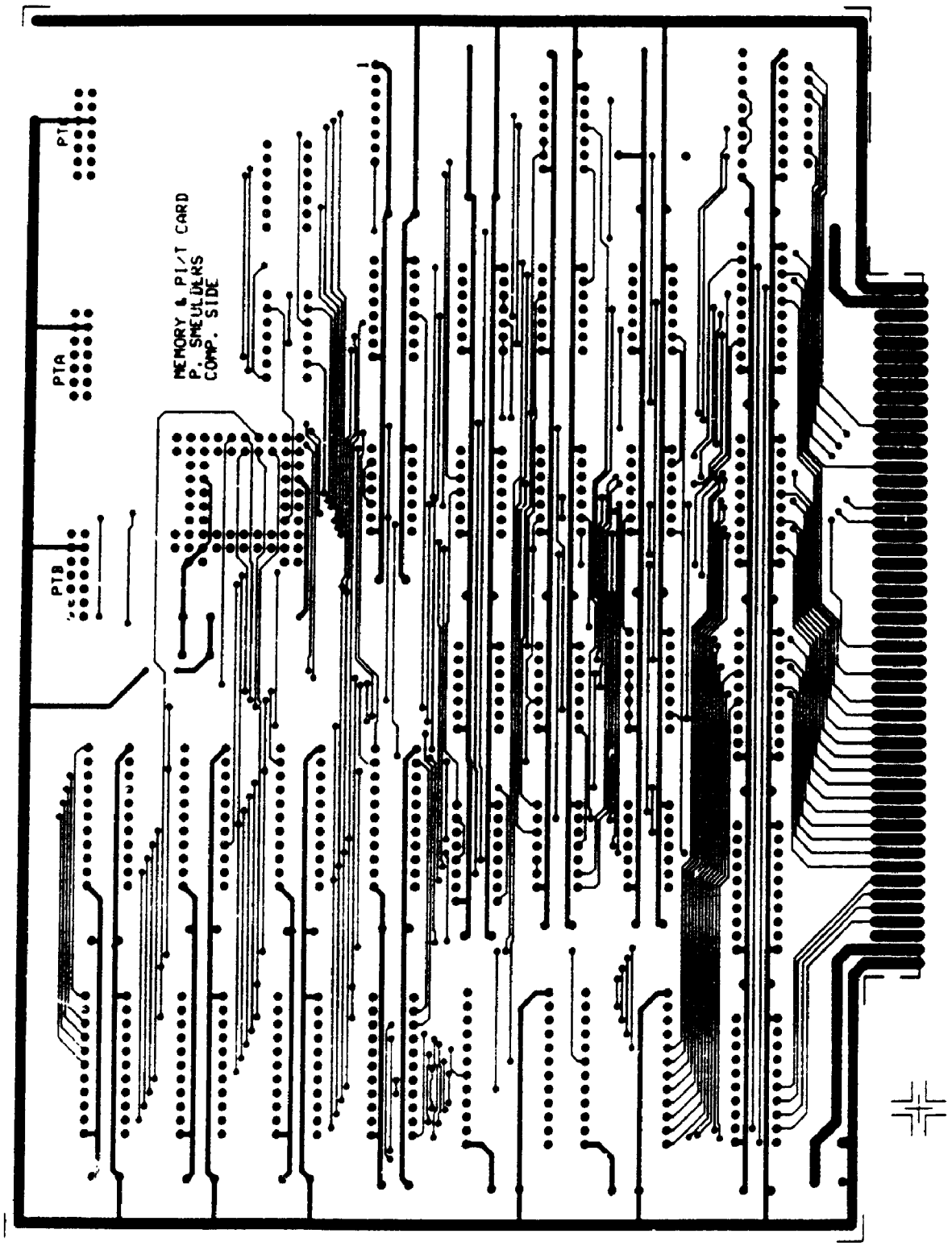


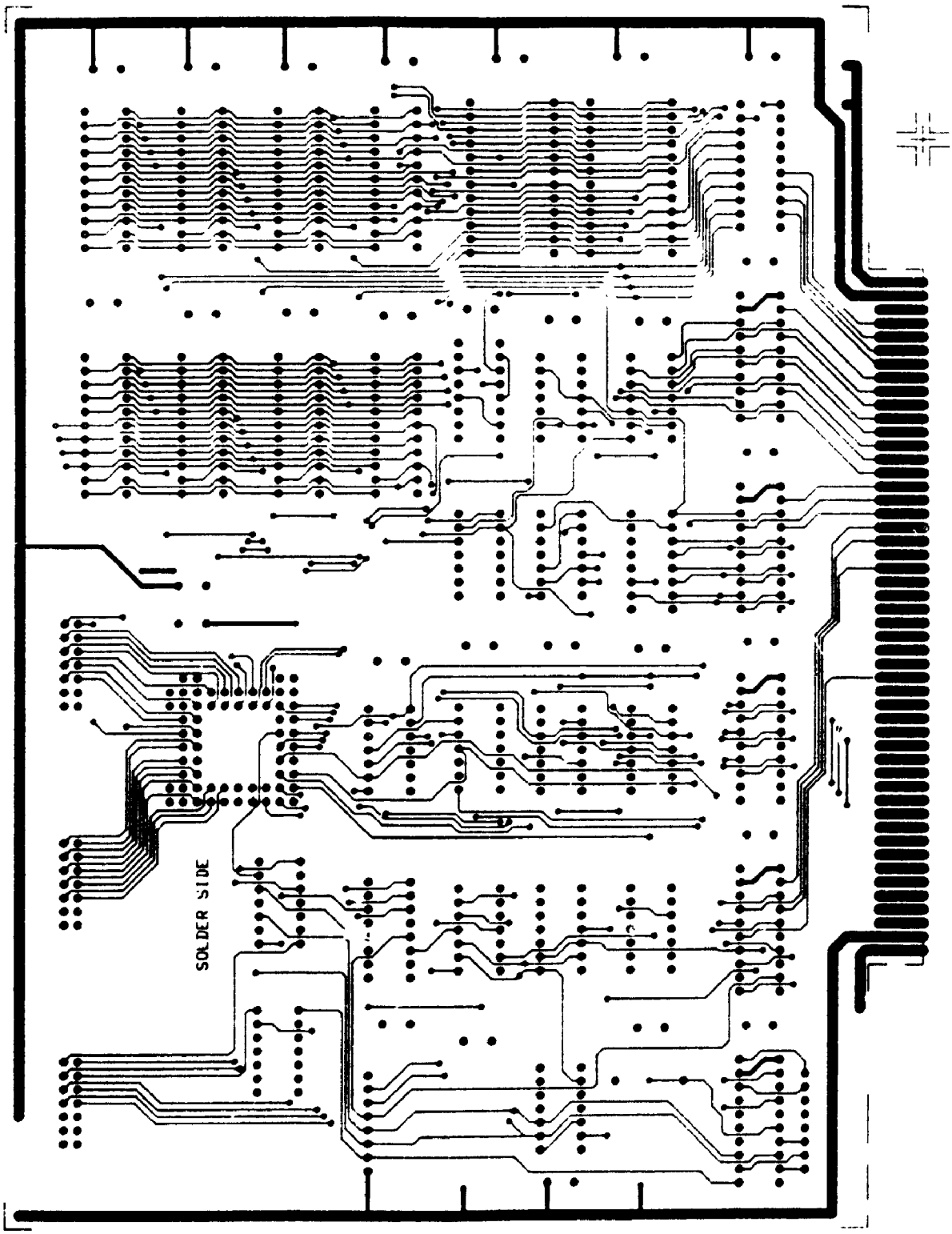

CPU CARD
P. Smeulders
COMPONENT SIDE

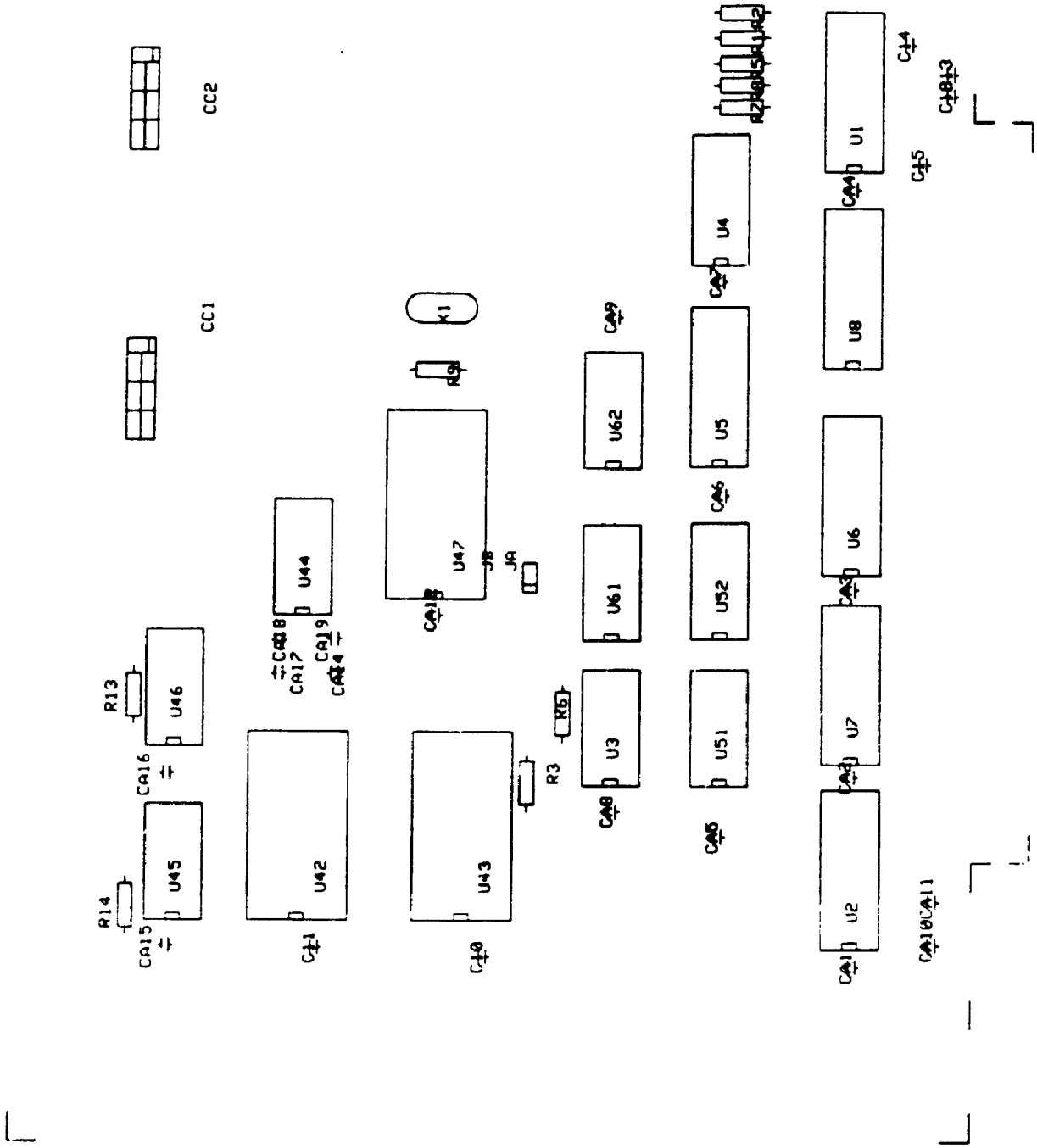

SOLDER SIDE

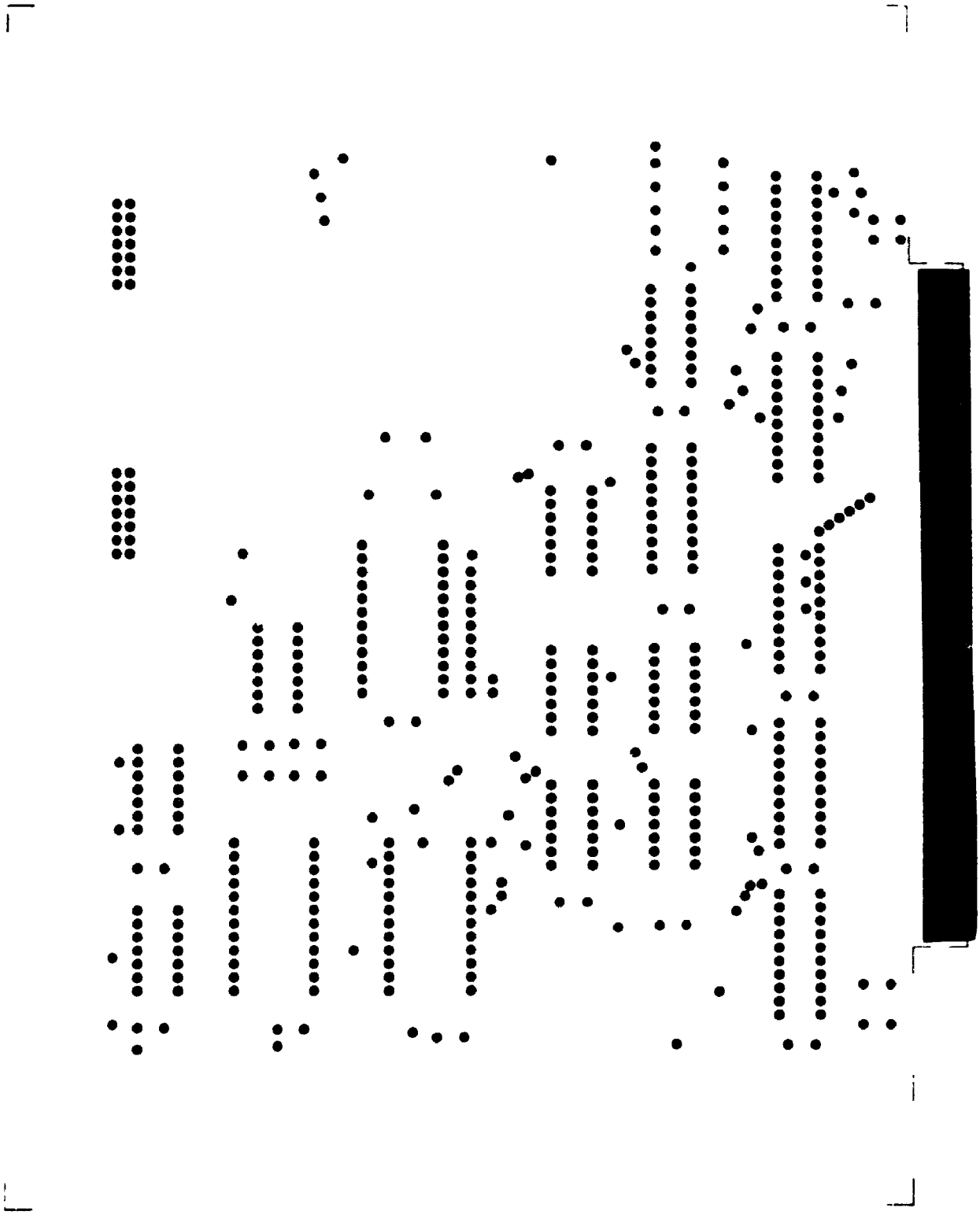

U1
U2
U3
U4
U5
U6
U7
U8
U9
U10
U11
U13
U14
U15
U16
U17
U18
U19
U20
U21
U22
U23
U24
U25
U26
U27
U28
U30
U31
U32
U33
U34
SW1
SW2
R1-R7
R8-R11
R12
C12
C13

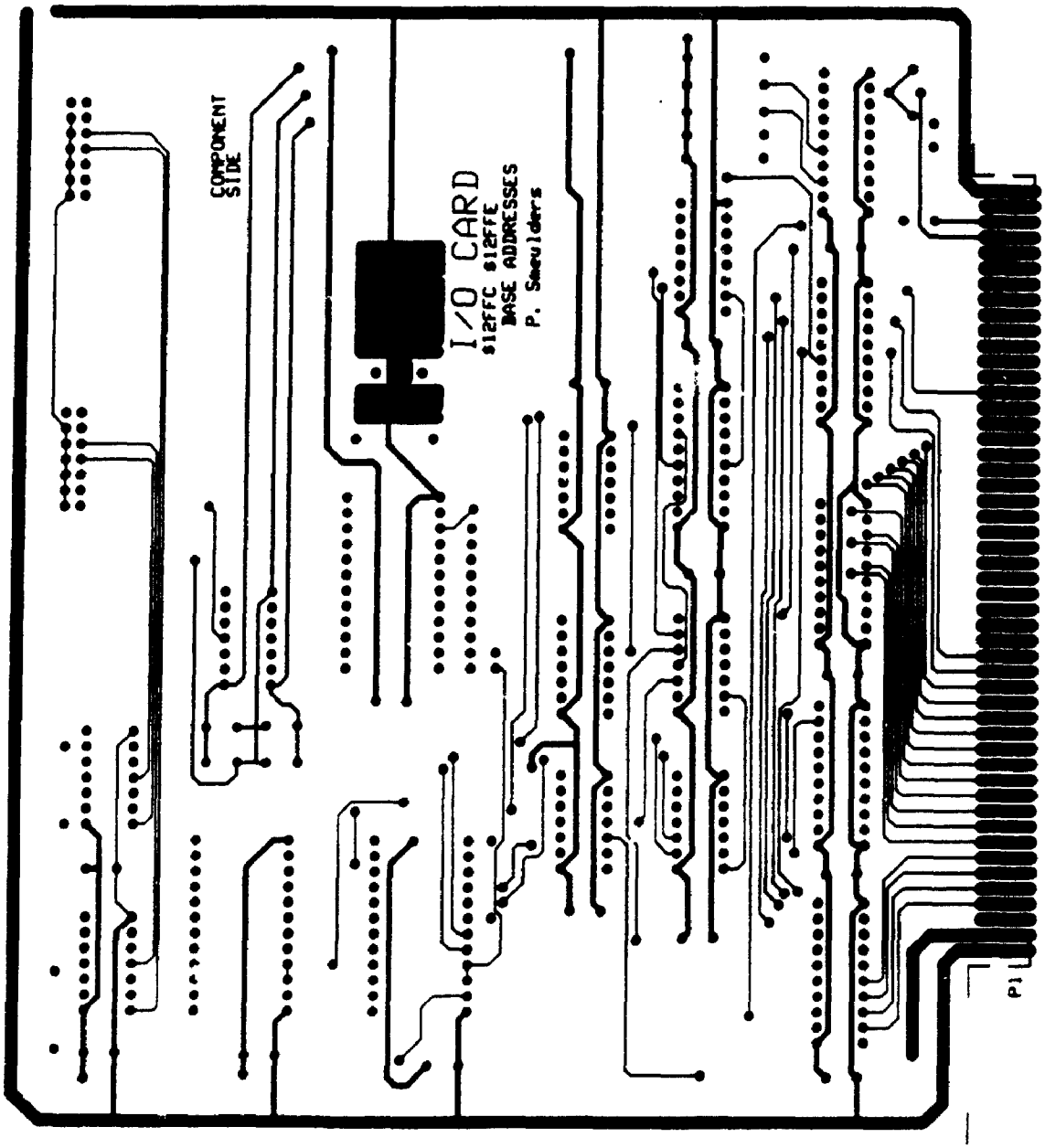


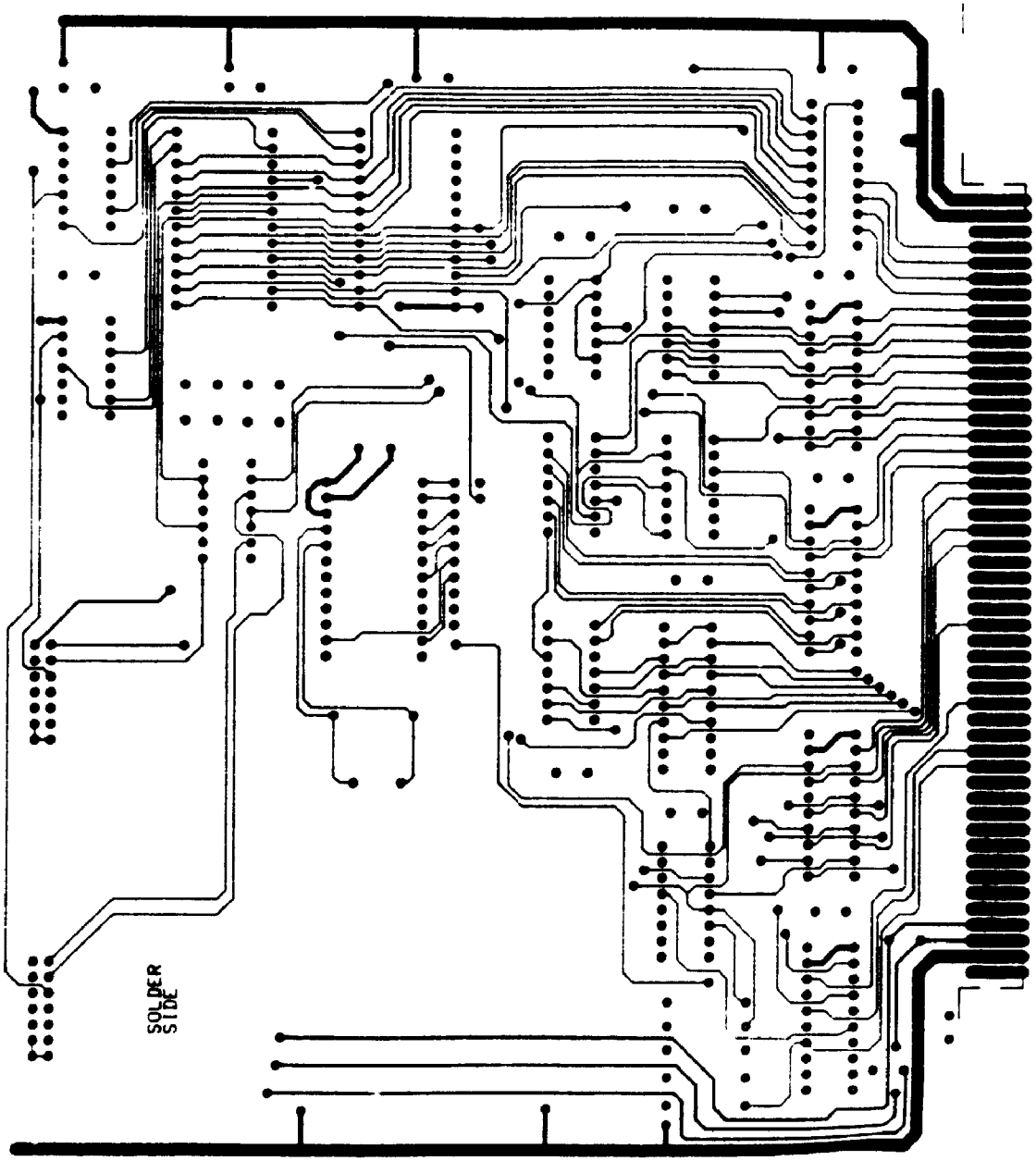

PTB
PTA
PT
MEMORY & PI/T CARD
P. SMEULDERS
COMP. SIDE

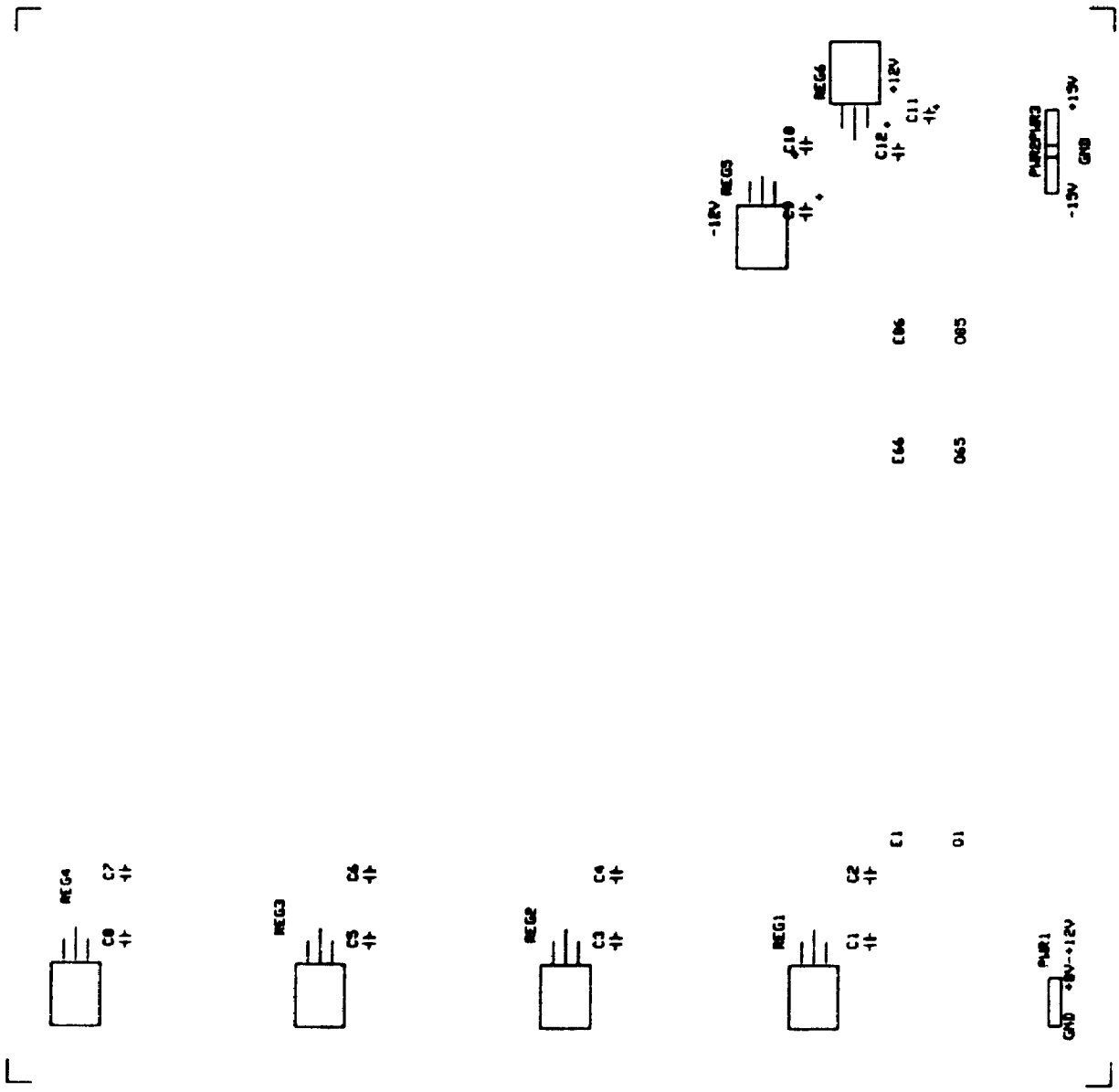

SOLDER SIDE

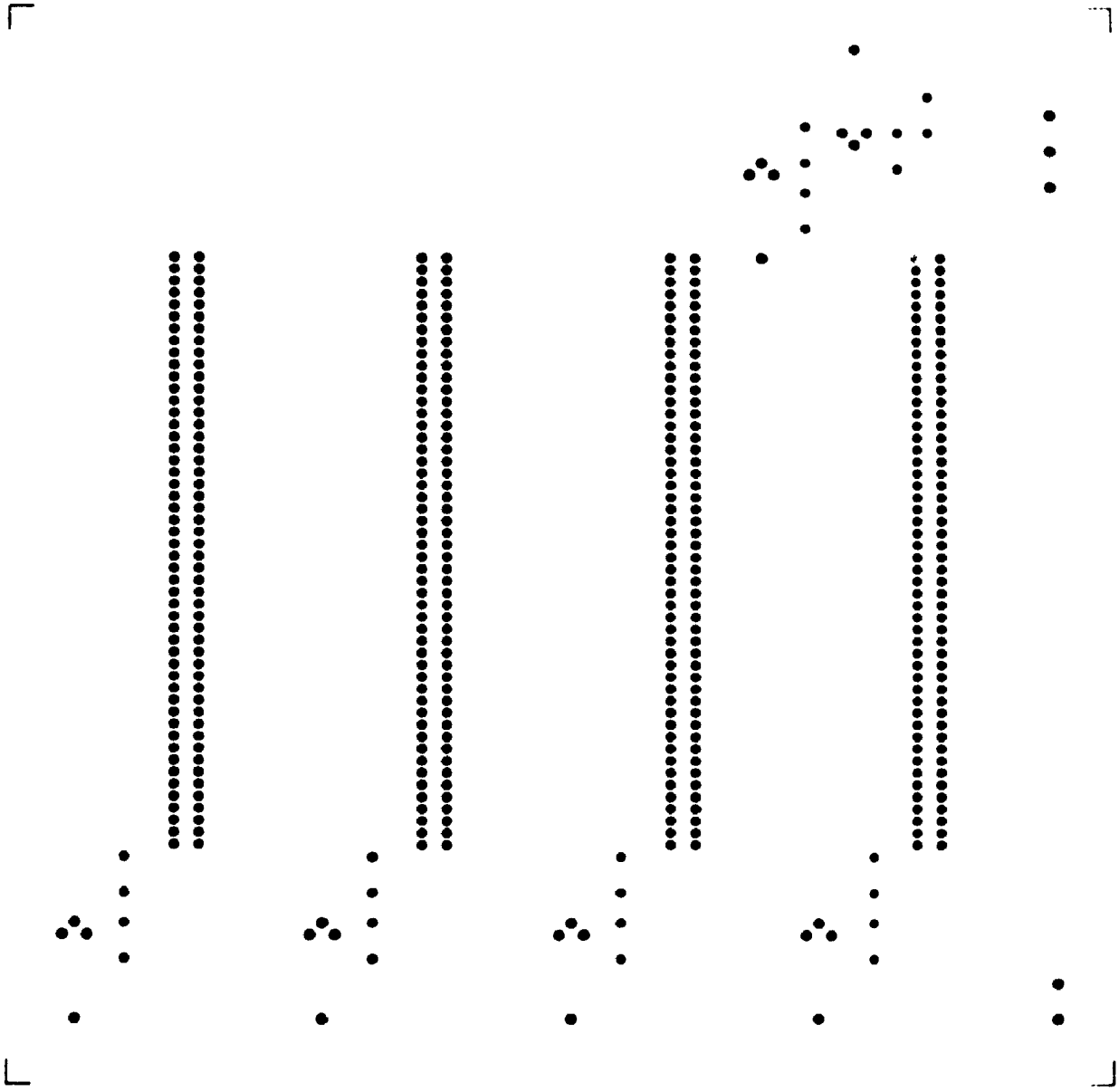

CC2
CC1
U46
U45
U44
U42
U47
U43
U62
U61
U3
U51
U52
U5
U4
U8
U6
U7
U2
U1
R13
R14
R3
R6
R9
X1
J8
J9
C4
C5
C10
C11

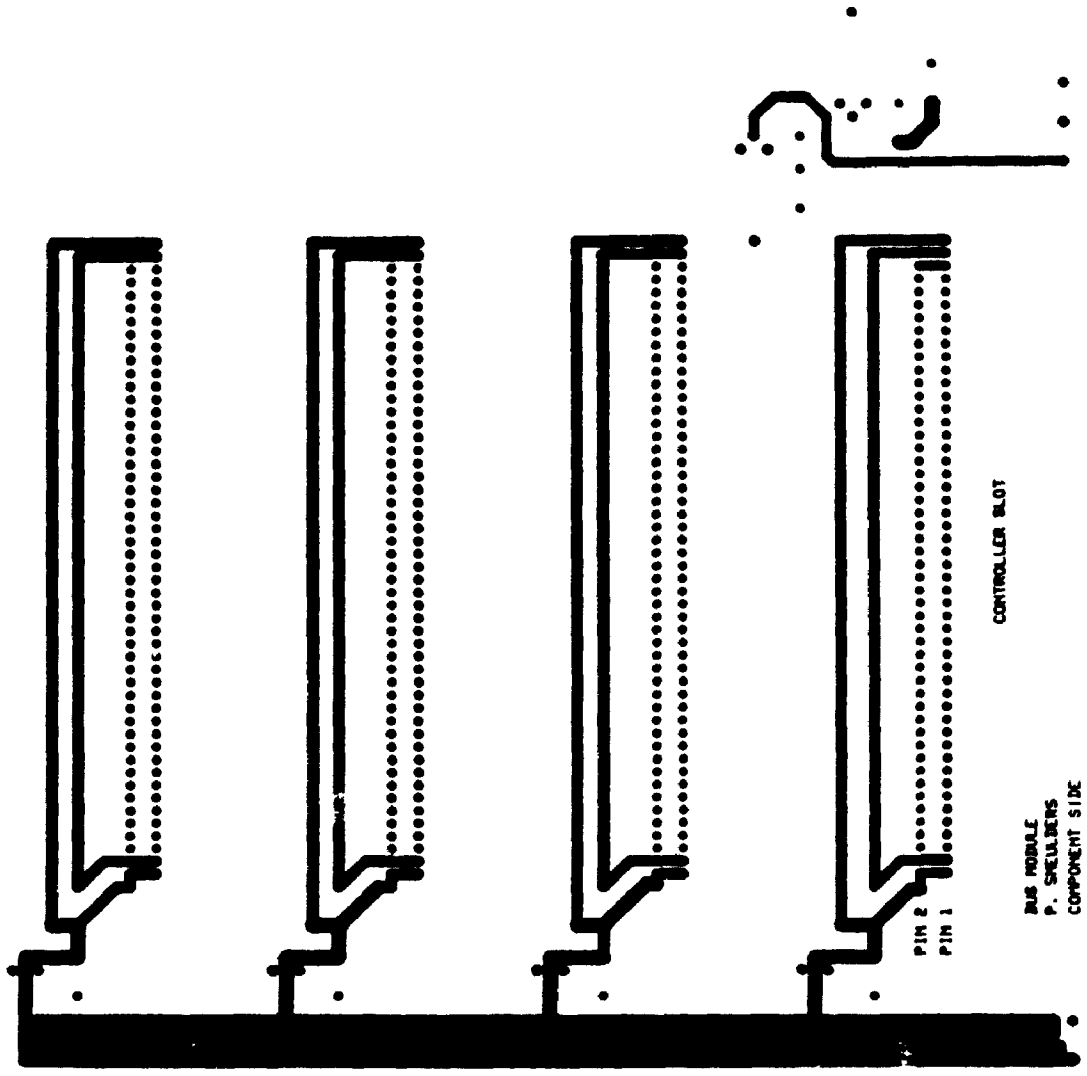


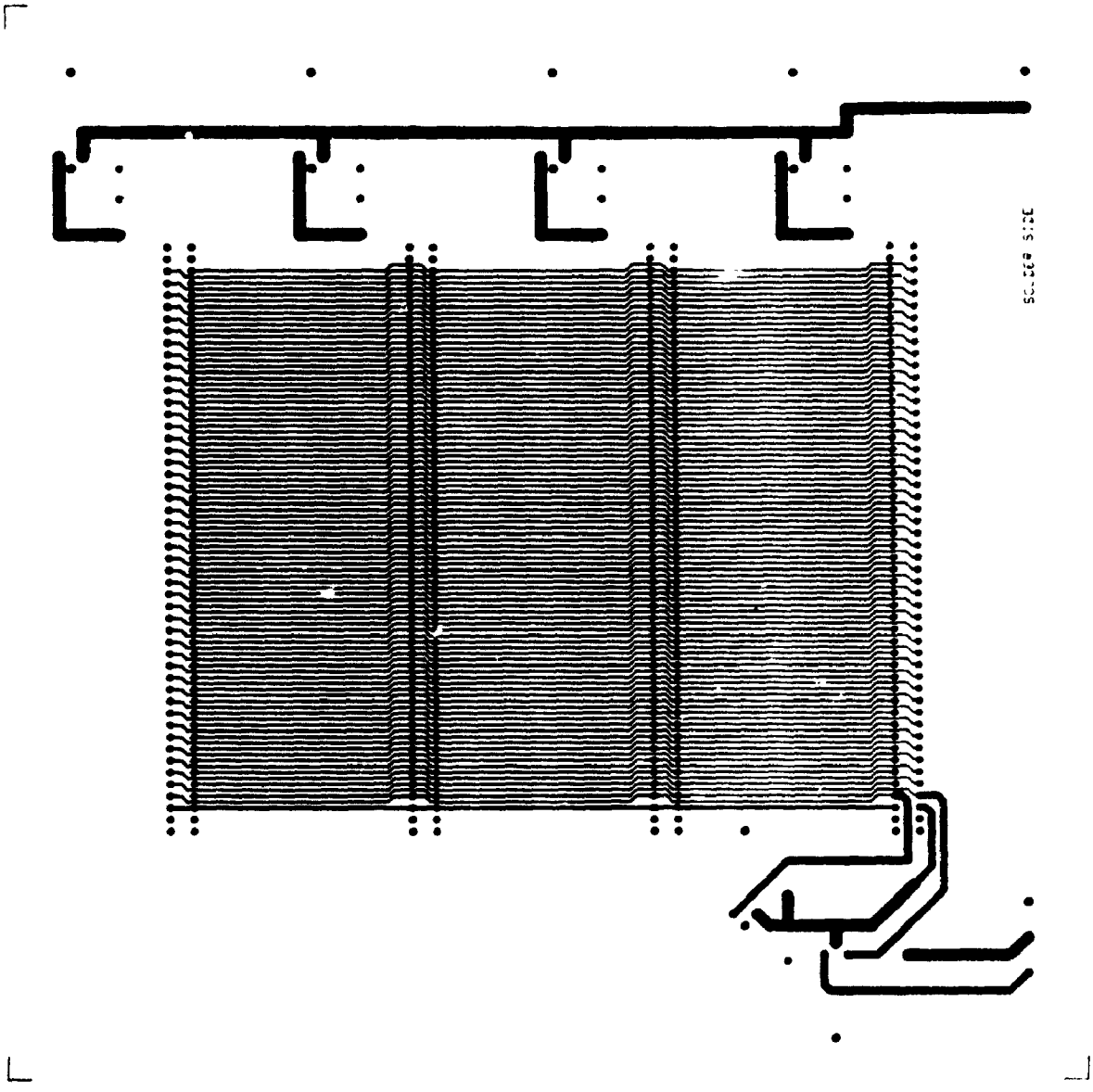

COMPONENT SIDE
I/O CARD
$12FFC $12FFE
BASE ADDRESSES
P. Saeulders
P1


SOLDER
SIDE

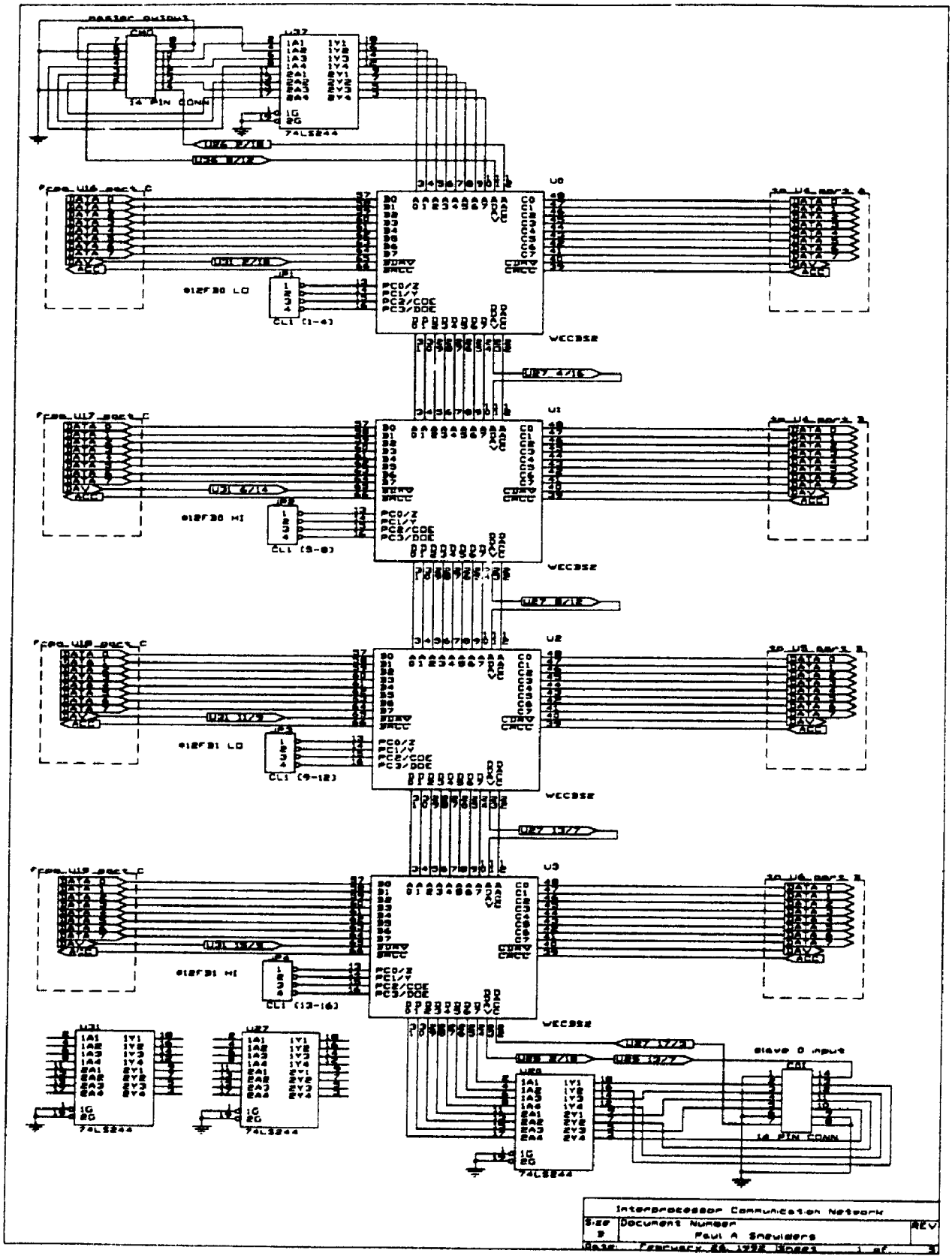

REG6
+12V
C11
C12
C10
REG5
-12V
C9
PWRPWR3
+15V
GND
-15V
C86
D85
C66
D65
C1
D1
REG4
C7
C8
REG3
C6
C5
REG2
C4
C3
REG1
C2
C1
PWR1
GND
+5V-+12V

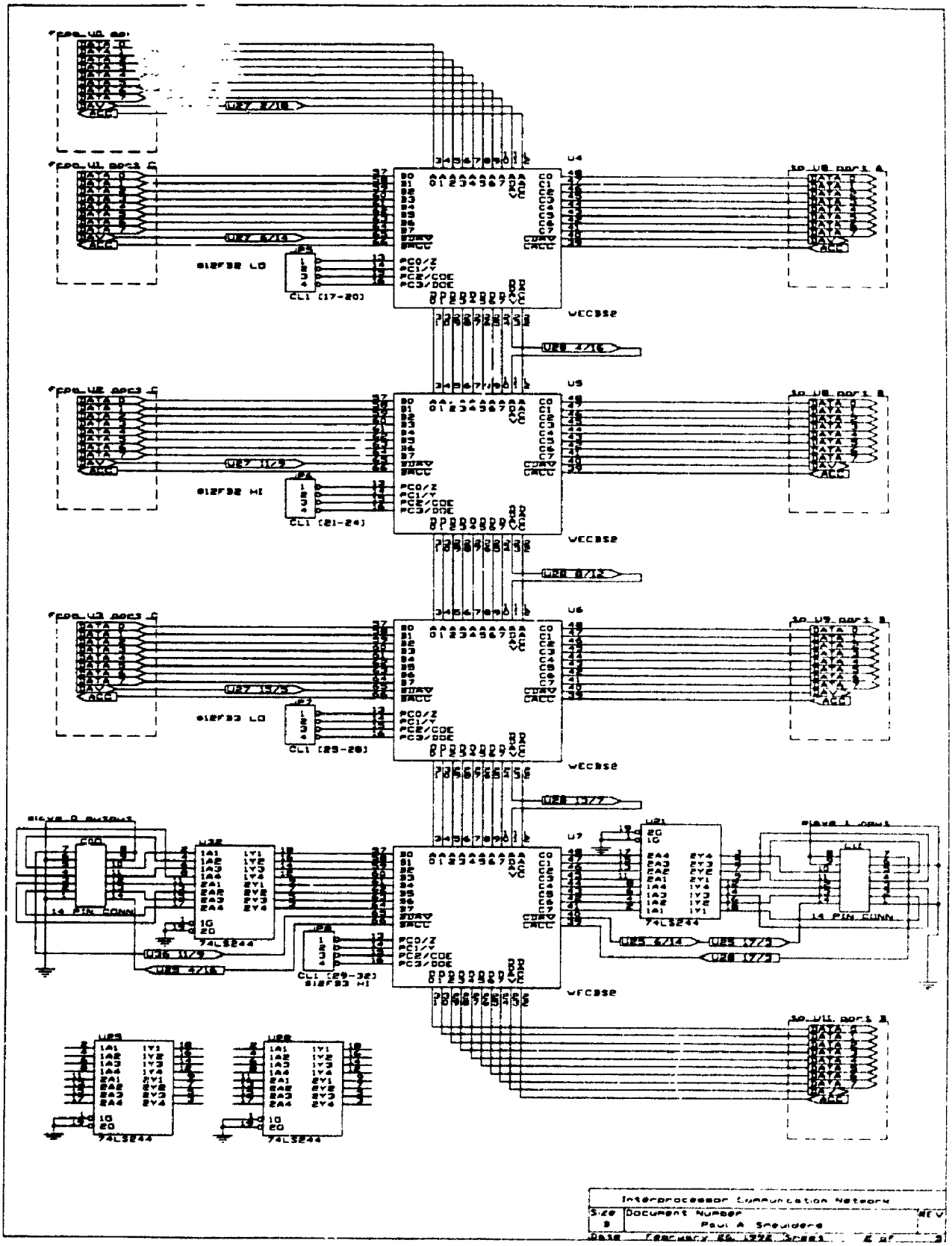


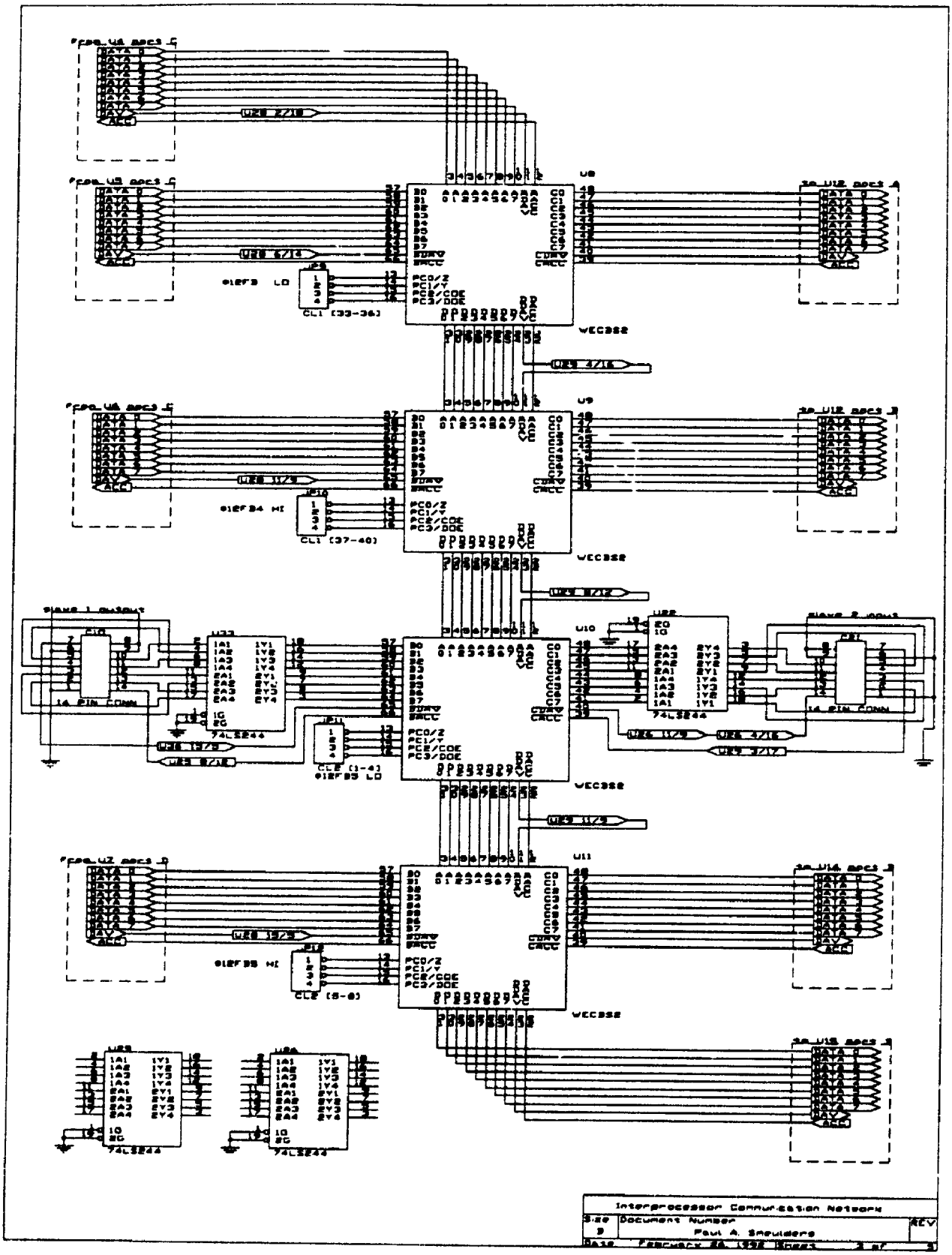

CONTROLLER SLOT
PIN 2
PIN 1
BUS MODULE
P. SHELDERS
COMPONENT SIDE

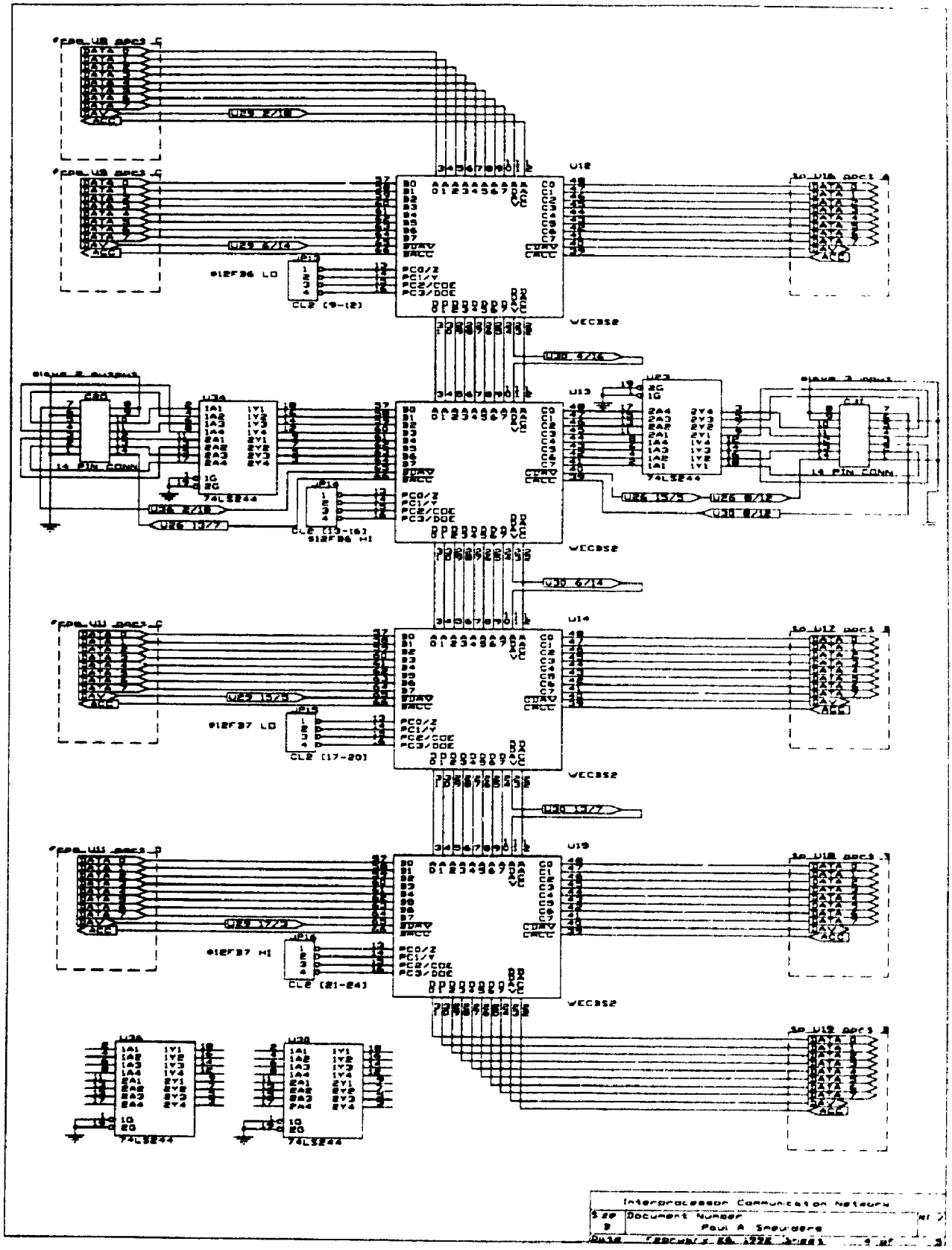


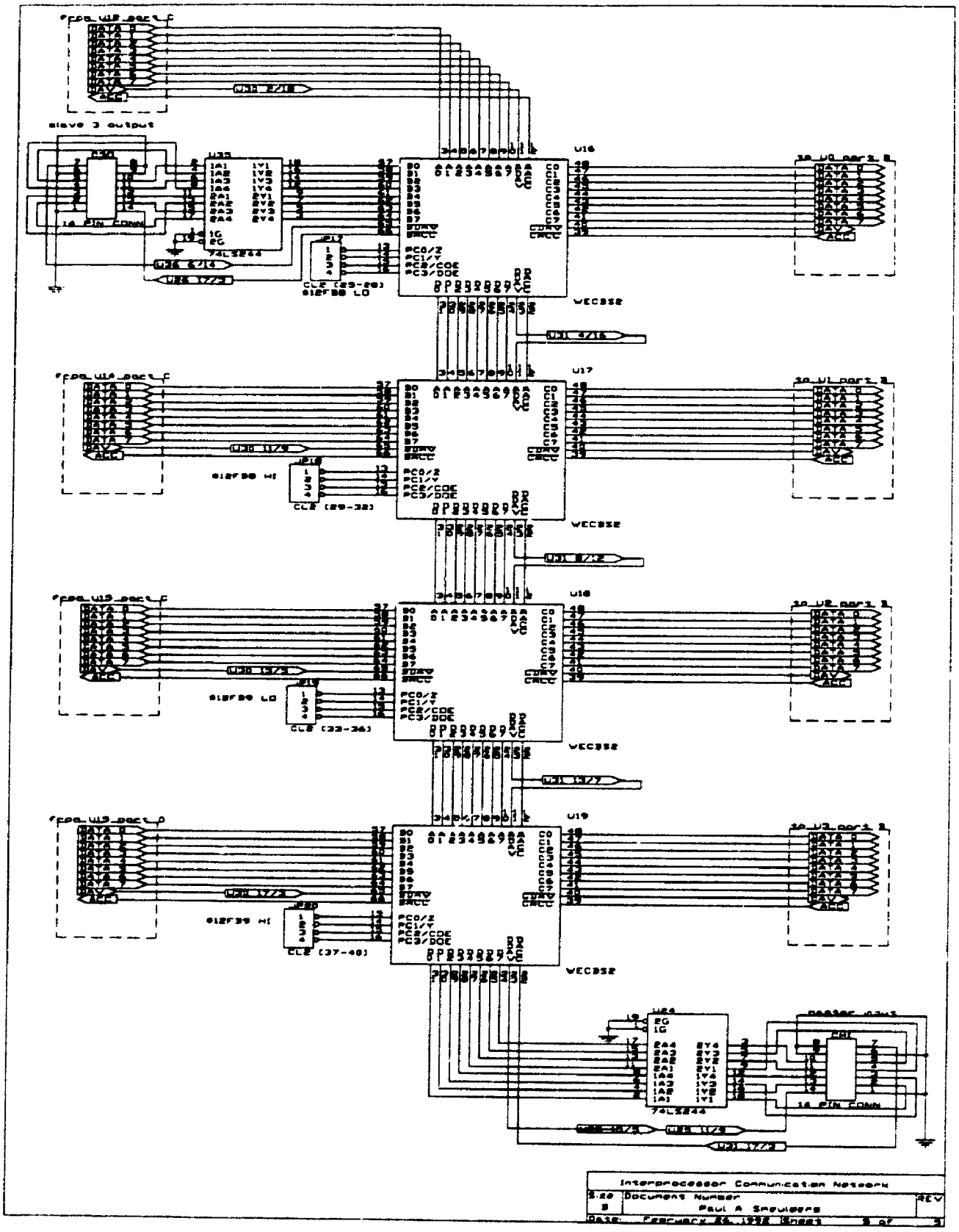
Appendix F: Interprocessor Communication Network Schematic Diagrams


74LS244
14 PIN CONN
Slave 0 input
Interprocessor Communication Network
Document Number
Paul A Snevelers
February 28, 1992

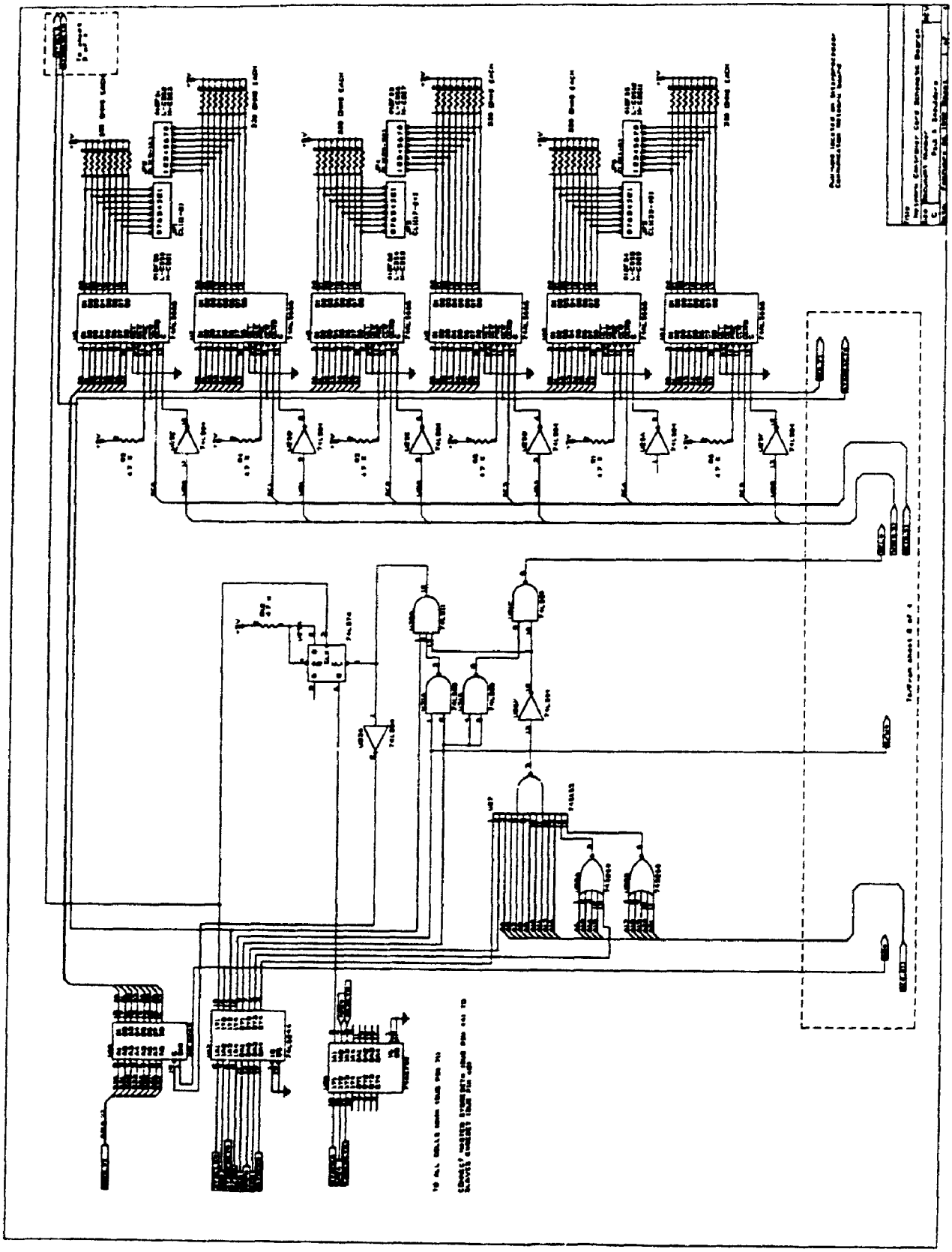

Interprocessor Communication Network
Paul A. Steinbacher

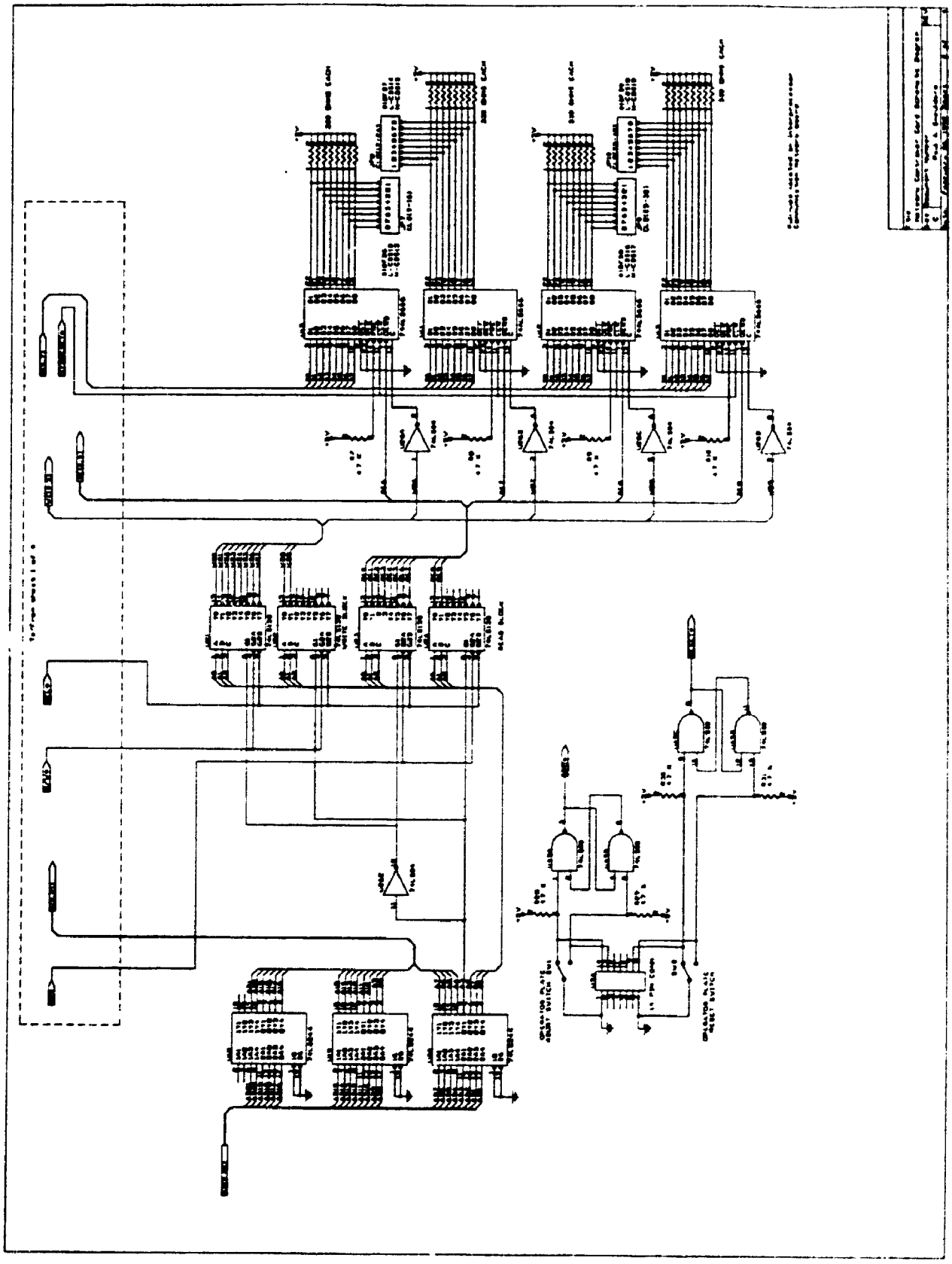

Interprocessor Communication Network
Document Number
Paul A. Smulders
U8
U9
U10
U11
74LS244
16 PIN CONN

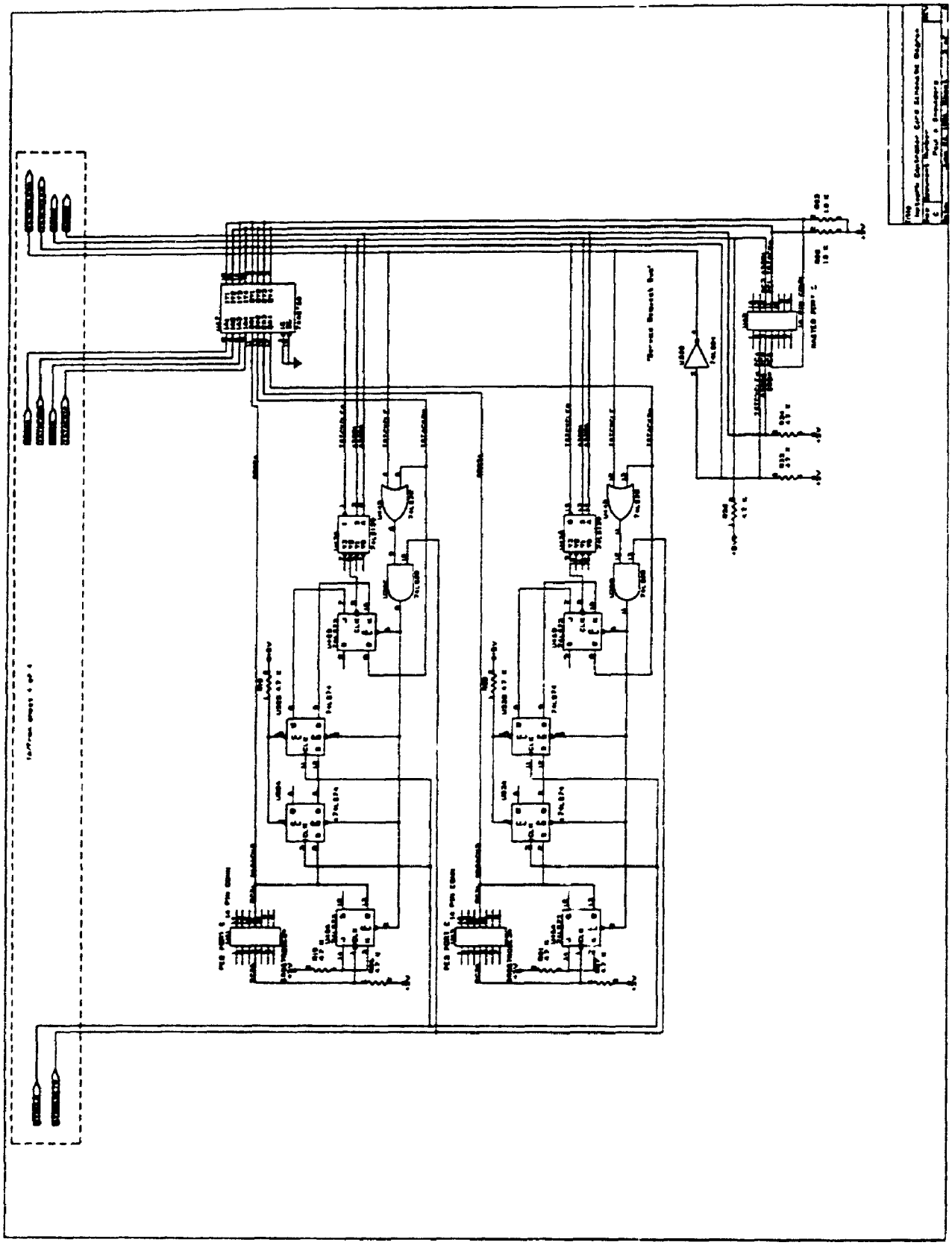

Interprocessor Communication Network
Document Number
Paul A. Snowberg

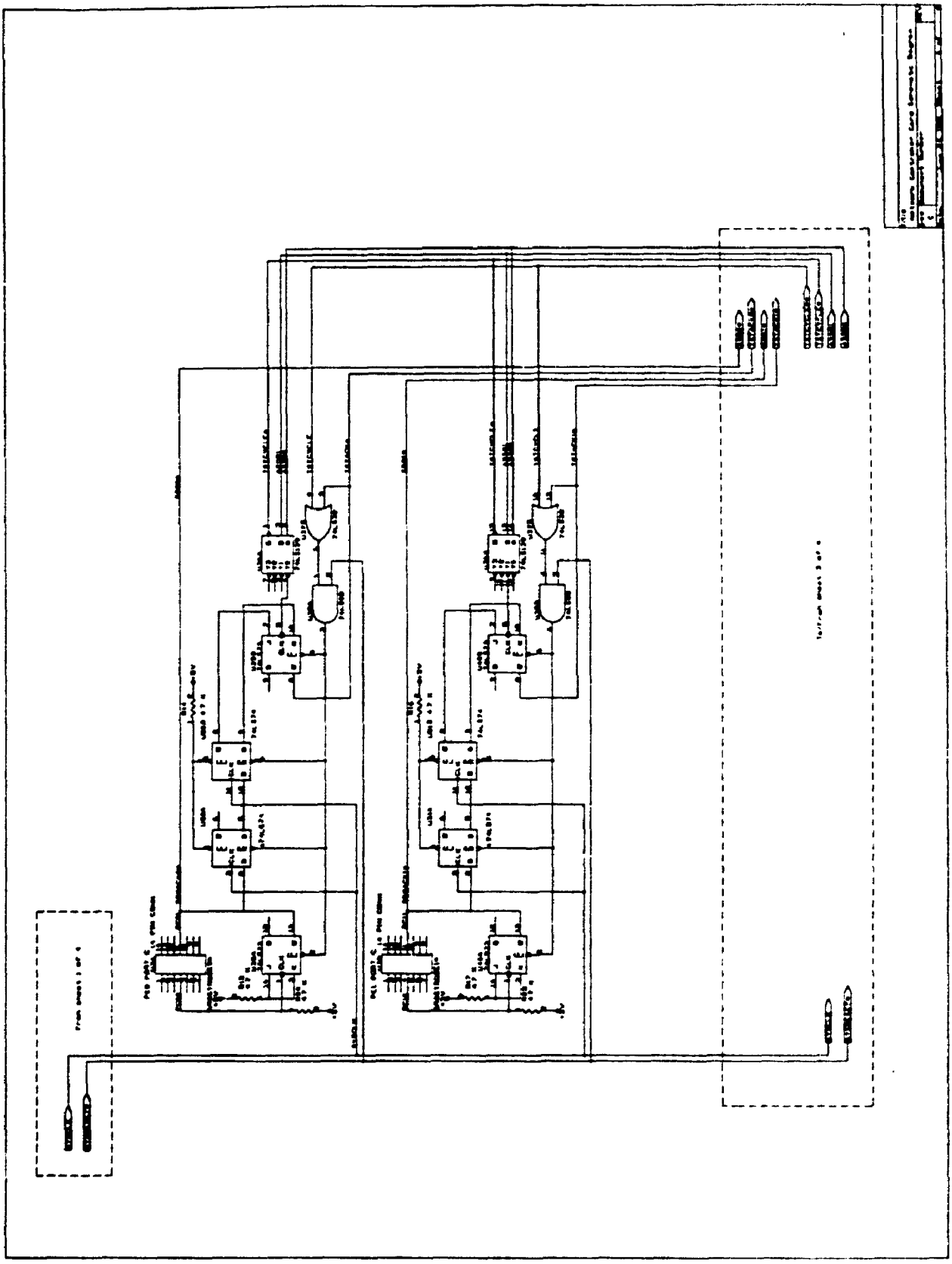

slave 3 output
74LS244
16 PIN CONN
WEC852
Interprocessor Communication Network
Document Number
Paul A Sagliocco

Appendix G: Network Controller Card Schematic Diagrams









Appendix H: Master Monitor Commands and TRAP #15 Utility Routines

This appendix describes RMCS monitor command syntax and the functions they perform. The TRAP #15 utility subroutines are also documented, with calling and parameter passing conventions, as well as register usage. The appendix is presented as a reference for programming the RMCS I/O functions. Other TRAP exception handling routines are documented in chapter 3 of the thesis.

H.1 Master monitor program command syntax and functions

The following notation is used:

spc	-	space character delimiter
cr	-	carriage return delimiter
lf	-	line feed character delimiter
^	-	"up arrow" character delimiter
<addr>	-	Address specification hexadecimal string, right justified, zero filled
[ ]	-	encloses optional parameters

All numerical input parameters must be entered as a hexadecimal string of characters.

H.1.1 Register examine/modify

Syntax: R spc OR R cr

Displays current contents of MPU registers and allows modification of registers D0-D7, A0-A6. The stack pointers (A7), the status register, and the program counter are not modifiable.

After the registers are displayed, the contents of D0 are re-displayed and the user is prompted for a change. Valid responses to prompts are:

cr	-	terminate the command
lf	-	skip to next register prompt
^	-	skip to previous register prompt
<hex data>cr	-	replace register contents with specified new data and terminate.
<hex data>lf	-	replace register contents with specified new data, skip to next register prompt.
<hex data>^	-	replace register contents with specified new data, skip to previous register prompt.

Other delimiters cause error messages, the register being examined is not updated, and the command terminated.

H.1.2 Display address space contents

Syntax: D spc [<addr1> spc [<addr2>]] cr

Displays the hexadecimal and ASCII equivalent contents of memory locations <addr1> to <addr2>. The value of <addr1> is rounded to the nearest lower 16 byte boundary, and the value of <addr2> is rounded to the nearest higher 16 byte boundary, and thus at least 16 bytes of data are displayed.

If no addresses are specified before the cr, locations $0 to $F are displayed.

If an <addr2> specification is not provided, locations <addr1> (rounded down) to <addr1>+16 are displayed. In this case, the spc character is required after the addr1 specification, followed by the cr.

H.1.3 Memory examine, modify

Syntax: M spc <addr> cr

Displays the contents of memory location specified by <addr>, and prompts the user for a change. Valid input to the prompts are:

cr	-	Terminate command, without modifying current location's contents.
lf	-	Display contents of next higher memory location, without modifying current location's contents.
^	-	Display contents of next lower memory location, without modifying current location's contents.
<hex data> cr	-	Update current location with specified data, terminate command.
<hex data> lf	-	Update current location with specified data, skip to next higher location.
<hex data> ^	-	Update current location with specified data, skip to next lower location.

H.1.4 Fill memory with data

Syntax: F spc OR F cr

Fill consecutive memory with a particular data byte. The user is prompted for further information, and must respond as shown:

'Input start address for Fill - >' <addr> cr

'Input Block Length to Fill - >' <addr> cr

'Input Byte to fill with - >' <hex data byte> cr

H.1.5 Talk to a host computer

Syntax: T cr

Enables transparent communication to host computer via the console terminal. All characters except control-D are echoed to the host. Typing control-D terminates communication and the command level is re-entered.

H.1.6 Load an S-format record into memory

Syntax: LOA spc OR LOA cr

An S-format record is loaded from the host communication link into memory at the addresses specified in the record, plus the offset specified. The user is prompted to provide the offset value, and the filename of the record as it exists on the host machine. Both responses are terminated by cr. The number of errors detected are reported when loading has been completed, or after it is aborted due to unrecoverable errors. The command handler is re-entered in either case.

H.1.7 Breakpoint Set/Remove

Syntax: B spc <addr> cr to set a breakpoint at addr.

B spc -<addr> cr to remove a breakpoint at addr.

B spc - cr to remove all breakpoints.

Program execution ceases AFTER the instruction at <addr> is executed, and the register contents are displayed. The user is prompted to type any key to continue with the program, or to type ESC to cancel program execution. A maximum of 6 breakpoints are allowed.

Breakpoints are realized by substituting the instruction at <addr> with $4AFC, the ILLEGAL instruction op-code. The original instruction and its address are stored in the RAM work-page. The original instruction is restored when the breakpoint is removed.

If a system hardware RESET condition occurs, the breakpoint counters and flags are cleared. If breakpoints were active prior to the RESET condition, the $4AFC op-codes will remain in the code in RAM. Therefore, breakpoints should be removed before resetting the system.

H.1.8 Call a supervisor state program

Syntax: C spc <addr>[spc T] cr

Commences execution of a program starting at <addr> as a supervisor-state subroutine of the monitor program. The program must end with the RTS ($4A75) op-code, to ensure orderly return to the command level. Breakpoints may be set to halt execution at specific addresses.

The optional T switch enables single instruction tracing. After each instruction is executed, the registers are displayed, and the user is prompted to continue. Typing ESC terminates the program, the supervisor stack pointer is re-initialized, the command level re-entered. Typing any other key allows program execution to proceed.

Tracing is not available when breakpoints are set.

H.1.9 GO execute a user state program

Syntax: G spc <addr>[spc T] cr

Commences execution of a program starting at <addr> in the user-state. The program must end with the TRAP #0 ($4E40) op-code to ensure orderly return to the supervisor-state monitor command level. Breakpoints may be set to halt execution at specific addresses.

The optional T switch enables single instruction tracing. After each instruction is executed, the registers are displayed, and the user is prompted to continue. Typing ESC terminates the program, the supervisor stack pointer is re-initialized, the command level re-entered. Typing any other key allows program execution to proceed.

Tracing is not available if breakpoints are set.

H.2 TRAP #15 handler

The monitor program uses several I/O and code conversion routines available in the ROM. The TRAP #15 handler allows a user's program to access a subset of the routines in an address independent manner.

The routines use the MPU registers to pass parameters, and thus it is the programmer's responsibility to provide correct data in the registers, and to preserve data appropriately before the TRAP #15 call.

To call a routine from within a program, the instruction syntax is:

TRAP #15

DC.W <number>

...where <number> specifies the access code for the desired routine.

It is important that a word sized <number> specification is used, to avoid address errors due to op-code fetches at odd addresses (if byte-sized specifiers are used) upon return from the TRAP #15 routine.

The following section describes the various service routines, their respective access codes, and register allocations.

H.2.1 TRAP #15 accessible routines

Code number	Routine name	Registers used	Volatile registers
0	SENDLO	D0,A6	-
1	SENDLOC	D0	-
2	SENDLOH	D0	-

Output the contents of register D0.L (longword) to the ACIA port address specified by the contents of register A6, the console port, or the host port, respectively.

Code number	Routine name	Registers used	Volatile registers
3	SENDWO	D0,A6	-
4	SENDWOC	D0	-
5	SENDWOH	D0	-

Output the contents of register D0.W (word) to the ACIA port address specified by the contents of register A6, the console port, or the host port, respectively.

Code number	Routine name	Registers used	Volatile registers
6	SENDBY	D0,A6	-
7	SENDBYC	D0	-
8	SENDBYH	D0	-

Output the contents of register D0.B (byte) to the ACIA port address specified by the contents of register A6, the console port, or the host port, respectively.

Code number	Routine name	Registers used	Volatile registers
9	SENDCLF	A6	-
10	SENDCLFC	-	-
11	SENDCLFH	-	-

Output a carriage return and line feed to the ACIA port address specified by the contents of register A6, the console port, or the host port, respectively.

Code number	Routine name	Registers used	Volatile registers
12	INCH	D0.B,A6	D0.B
13	INCHCON	D0.B	D0.B
14	INCHHOS	D0.B	D0.B

Obtain a character input from the ACIA port address specified by the contents of register A6, the console port, or the host port, respectively. The ASCII coded byte is returned in D0.B

Code number	Routine name	Registers used	Volatile registers
15	ASCBIN	D0	D0

Converts a single ASCII character in D0.B into its binary equivalent. The result is returned in the least significant 4-bits of D0.B. If the character is not in the range $0 to $F, it is returned in the most significant byte of D0.W, with the least significant byte set to $FF.

Code number	Routine name	Registers used	Volatile registers
16	COMESCO	D0,A0,A6	D0,A0
17	OUTMESC	D0,A0	D0,A0
18	OUTMESH	D0,A0	D0,A0

Output an ASCII string to the ACIA port address specified by the contents of register A6, the console port, or the host port, respectively. The string to be output begins at the address specified by the contents of register A0, and terminates with the EOT ($04) character. EOT is not sent. Upon return, D0.B contains EOT, and A0 points to the location following the EOT entry.

Code number	Routine name	Registers used	Volatile registers
19	PUT8HX	D0,D1,D2,A0	D0,D1,D2,A0
20	PUT6HX	D0,D1,D2,A0	D0,D1,D2,A0
21	PUT4HX	D0,D1,D2,A0	D0,D1,D2,A0
22	PUT2HX	D0,D1,D2,A0	D0,D1,D2,A0
23	PUT1HX	D0,A0	D0,A0

Convert the N (N = 1,2,4,6,8) 4-bit data in D0 into its ASCII equivalent, and store the result at the N sequential byte addresses specified by the contents of A0.

Code number	Routine name	Registers used	Volatile registers
24	DISBUF8	D0.L	-
25	DISBUF4	D0.W	-
26	DISBUF2	D0.B	-

Converts the contents of register D0.L, D0.W, D0.B, respectively, into their ASCII equivalents. Result is stored in the RAM work-page display buffer, DISBUF. The contents of the display buffer are then output to the console port.

Code number	Routine name	Registers used	Volatile registers
27	OUTDSBF	D0,A0	D0,A0

Output the contents of memory locations starting at the address specified by the contents of register A0, to the console port. The number of bytes to be output is specified by the contents of register D0.L.

Code number	Routine name	Registers used	Volatile registers
28	BLDNUM	D0,A0	D0,A0

Accept input from the console port as hexadecimal digits, until the first non-hex character is received. ASCII data accepted is converted to binary and returned, right justified and zero-filled in D0.L. If more than 8 hex digits are input before a non-hex character, only the last 8 digits are used.

A0.B contains the first non-hex character accepted. If no hex digits were input before the non-hex character, the most significant byte of A0.W is $FF. If valid hex digits were received first, the most significant byte of A0.W is cleared. The system of flagging the delimiters aids in error detection.

Code number	Routine name	Registers used	Volatile registers
29	BLDSTRG	D0,A0	D0,A0

Builds a sequential string of ASCII characters in memory starting at the location specified by the contents of register A0, the length of which is specified by the contents of register D0.L. Characters are accepted via the console port until no more characters are allowed, or until a Carriage Return is received. D0 will return with the number of characters remaining that could have been input. A0 returns with the address of the last byte stored, plus 1.

Appendix I: Master Monitor and Slave System Program Listings

The master monitor program is most easily assembled as a set of relocatable modules, with the individual modules subsequently linked to calculate all of the necessary relative offsets and external references. The method leaves many unknown operands in the individual listings, and the starting addresses of routines are not obvious. For this reason, the master monitor program was also assembled in an absolute manner so that all operands are calculated and included in the final listing.

The listing files generated by the absolute assembly of the master and slave programs follow.

I.1 Master Monitor Program Listing

```
                          1.          TTL ABSMONT
                          2.  *
                          3.  * ABSOLUTE ASSEMBLY OF MASTER MONITOR PROGRAM
                          4.  * March 14,1991
                          5.  ********************************************
                          6.  * THIS IS THE WORKPAGE AREA FOR THE ROM     *
                          7.  * ROUTINES                                  *
                          8.  ********************************************
                          9.  *
                         10.  * updated March 14,1991
                         11.  * Paul A. Smeulders
                         12.  * for use in multiprocessor system
                         13.  *
                         14.  * UPDATED OCT 3 1991
                         15.  * FIX ON OUTPUT PIT TRANSFERS
                         16.  *
                         17.          OPT P=68000
                         18.  *
                         19.  * EQUATES
                         20.  *
# 00000015               21.  ACSTAT  EQU $15
# 00002700               22.  NOINTRU EQU $2700
# 00010020               23.  RESTART EQU $00010020
# 0000FDE0               24.  INITSSP EQU $0000FDE0
# 00012FC0               25.  PGCR    EQU $12FC0
# 00012FC1               26.  PSRR    EQU $12FC1
# 00012FC2               27.  PADDR   EQU $12FC2
# 00012FC3               28.  PBDDR   EQU $12FC3
# 00012FC4               29.  PCDDR   EQU $12FC4
# 00012FC6               30.  PACR    EQU $12FC6
# 00012FC7               31.  PBCR    EQU $12FC7
# 00012FC8               32.  PADR    EQU $12FC8
# 00012FC9               33.  PBDR    EQU $12FC9
# 00012FCC               34.  PCDR    EQU $12FCC
# 00012FCD               35.  PPSR    EQU $12FCD
# 00000000               36.  INPUTS  EQU $00
# 000000FF               37.  OUTPUTS EQU $FF
# 00000030               38.  ACONT   EQU $30
# 00000070               39.  BCONT   EQU $70
# 0000001C               40.  CCONFIG EQU $1C
# 00000010               41.  ENABLEA EQU $10
# 00000020               42.  ENABLEB EQU $20
# 0000003E               43.  ASCGT   EQU $3E
# 00000020               44.  SPACE   EQU $20
# 0000000F               45.  LINCNT  EQU 15
# 00000007               46.  BLKCNT  EQU 7
# 0000002E               47.  ASCDOT  EQU $2E
# 00002020               48.  TOSPC   EQU $2020
# FFFFFFF0               49.  NIBMSK  EQU $FFFFFFF0
# 0000202D               50.  SPDASH  EQU $202D
# 0000203E               51.  SPASGT  EQU $203E
```

```
                  # 0000000D            52.  CR       EQU $0D
                  # 0000000A            53.  LF       EQU $0A
                  # 0000005E            54.  UPAR     EQU $5E
                  # 00000000            55.  DEFASTR EQU 00
                  # 000000FF            56.  DEFAEND EQU $FF
                  # 00000C80            57.  WATCONT EQU $C80
                  # 00000075            58.  BRKLEV   EQU $75
                  # 0000FF0D            59.  ZFFCR    EQU $FF0D
                  # 0000FF2D            60.  ZFFDSH   EQU $FF2D
                  # 00004AFC            61.  ZILLEG   EQU $4AFC
                  # 00000003            62.  MASK     EQU $03
                  # 00000006            63.  H3LEV    EQU $6
                  # 00000002            64.  H3S      EQU $02
                  # 00000000            65.  H1S      EQU $00
                  # 00012FB0            66.  CBS0 EQU $12FB0
                  # 00000000            67.  PC0      EQU $00
                  # 00000001            68.  PC1      EQU $01
                  # 00000004            69.  PC4      EQU $04
                  # 00000003            70.  ADRCONT EQU $03
                  # 00000001            71.  BYTCONT EQU $01
                  # 00000008            72.  SHIFT8   EQU $08
                  # 00000004            73.  EOT EQU 04
                  # 00000D0A            74.  CRLF EQU $0D0A
                  # 00012FFC            75.  ACIA1 EQU $12FFC
                  # 00012FFE            76.  ACIA2 EQU $12FFE
                  # 0000007F            77.  ASCMSK EQU $7F
                  # 00000012            78.  COUNTRG EQU 18
                  # 0000001B            79.  ESCCHR   EQU $1B
                                        80.  *
0000FDF0                                81.          ORG.L $10000-$210
0000FDF0  <3C>                          82.  TRAP4    DS.B $3C          ;BLOCKIN-ADDRBYT
0000FE2C  <24>                          83.  TRAP5    DS.B $24          ;BLOKOUT-BLOCKIN
0000FE50  <2E>                          84.  TRAP6    DS.B $2E          ;SNADBYT-BLOKOUT
0000FE7E  <34>                          85.  TRAP7    DS.B $34          ;NETCONF-SNADBYT
0000FEB2  <16>                          86.  TRAP8    DS.B $16          ;SENSAK-NETCONF
0000FEC8  <46>                          87.  TRAP9    DS.B $46          ;ENDCRIT-SENSAK
0000FF0E  <50>                          88.  REGSTOR DS.B $50
0000FF5E  <10>                          89.  DISBUF   DS.B $10
0000FF6E  <2>                           90.  STRGTMP DS.B $2
0000FF70  <20>                          91.  CMDSTK   DS.B $20
0000FF90  <10>                          92.  STRGBUF DS.B $10
0000FFA0  <1>                           93.  BRKNUM   DS.B 1
0000FFA1  <1>                           94.  BRKFLG   DS.B 1
0000FFA2  <24>                          95.  BRKSTK   DS.B $24
0000FFC6  <4>                           96.  TEMPRET DS.L 1
                                        97.  ********************************************
                                        98.  * MONITOR ENTRY POINT                      *
                                        99.  * THIS SECTION EXECUTED ON THE RESET VECTOR *
                                       100.  * AND WHEN A RESTART IS REQUIRED, WITH A NEW*
                                       101.  * SYSTEM STACK POINTER ALSO THE STATUS      *
                                       102.  * REGISTER IS LOADED TO ENABLE THE          *
                                       103.  * APPROPRIATE INTERRUPTS                    *
                                       104.  ********************************************
                                       105.  *
                                       106.  * THIS VERSION UPDATED MARCH 14,1991 TO
                                       107.  * SUPPORT PI/T AND MULTIPROCESSOR SYSTEM
                                       108.  * OPERATION.
                                       109.  * Paul A. Smeulders, 14/03/91
                                       110.  *
                                       111.  * DEFINE THE RESTART VECTOR
00010000                               112.          ORG.L $10000
00010000  0000FDE0                     113.          DC.L INITSSP
00010004  00010020                     114.          DC.L RESTART
                                       115.  *
                                       116.  * ENTRY POINT LOAD VECTOR TABLE
                                       117.  *
                                       118.
00010020                               119.          ORG.L RESTART
00010020  13FC 0003                    120.          MOVE.B #3,ACIA1
          00012FFC
00010028  13FC 0003                    121.          MOVE.B #3,ACIA2
          00012FFE
00010030  46FC 2700                    122.  RSTRT    MOVE.W #NOINTRU,SR      ;DISABLE INTS
00010034  21FC 0000FDE0                123.          MOVE.L #INITSSP,0
          0000
0001003C  21FC 00010020                124.          MOVE.L #RESTART,4
          0004
00010044  41FA 0C58                    125.          LEA BUSERR(PC),A0
00010048  21C8 0008                    126.          MOVE.L A0,8
0001004C  41FA 0C48                    127.          LEA ADDERR(PC),A0
```

```
00010050  21C8 000C        128.          MOVE.L A0,12
00010054  41FA 0D5A        129.          LEA ILLEG(PC),A0
00010058  21C8 0010        130.          MOVE.L A0,16
0001005C  41FA 0E4E        131.          LEA ZERDIV(PC),A0
00010060  21C8 0014        132.          MOVE.L A0,20
00010064  41FA 0E5E        133.          LEA CHKTRP(PC),A0
00010068  21C8 0018        134.          MOVE.L A0,24
0001006C  41FA 0E6E        135.          LEA TRPVTRP(PC),A0
00010070  21C8 001C        136.          MOVE.L A0,28
00010074  41FA 0E7E        137.          LEA PRIVTRP(PC),A0
00010078  21C8 0020        138.          MOVE.L A0,32
0001007C  41FA 0E8E        139.          LEA SPURINT(PC),A0
00010080  21C8 0060        140.          MOVE.L A0,96
00010084  41FA 0E9E        141.          LEA UNINIT(PC),A0
00010088  21C8 003C        142.          MOVE.L A0,60
0001008C  41FA 0EAE        143.          LEA LEV7INT(PC),A0
00010090  21C8 007C        144.          MOVE.L A0,124
00010094  41FA 0FD6        145.          LEA INTLEV2(PC),A0
00010098  21C8 0068        146.          MOVE.L A0,104
0001009C  41FA 0D7E        147.          LEA HTRACE(PC),A0
000100A0  21C8 0024        148.          MOVE.L A0,36
000100A4  41FA 1148        149.          LEA HTRAP0(PC),A0
000100A8  21C8 0080        150.          MOVE.L A0,128
000100AC  41FA 114E        151.          LEA HTRAP1(PC),A0
000100B0  21C8 0084        152.          MOVE.L A0,132
000100B4  41FA 114C        153.          LEA HTRAP2(PC),A0
000100B8  21C8 0088        154.          MOVE.L A0,136
000100BC  41FA FD32        155.          LEA TRAP4(PC),A0        ;PLACE IN ROM
000100C0  21C8 0090        156.          MOVE.L A0,144           ;AFTER OTHER
000100C4  41FA FD66        157.          LEA TRAP5(PC),A0        ;LOADS
000100C8  21C8 0094        158.          MOVE.L A0,148
000100CC  41FA FD82        159.          LEA TRAP6(PC),A0
000100D0  21C8 0098        160.          MOVE.L A0,152
000100D4  41FA FDA8        161.          LEA TRAP7(PC),A0
000100D8  21C8 009C        162.          MOVE.L A0,156
000100DC  41FA FDD4        163.          LEA TRAP8(PC),A0
000100E0  21C8 00A0        164.          MOVE.L A0,160
000100E4  41FA FDE2        165.          LEA TRAP9(PC),A0
000100E8  21C8 00A4        166.          MOVE.L A0,164
000100EC  41FA 111A        167.          LEA HTRAP15(PC),A0
000100F0  21C8 00BC        168.          MOVE.L A0,188
                           169.  *
                           170.  * COPY CRITICAL ROUTINES TO RAM
                           171.  *
000100F4  41FA 11C0        172.          LEA ADDRBYT(PC),A0      ,START ADDR
000100F8  43FA 12DA        173.          LEA ENDCRIT(PC),A1      ;END ADDR
000100FC  93C8             174.          SUBA.L A0,A1            ;SIZE OF BLK
000100FE  2009             175.          MOVE.L A1,D0            ;COUNTER
00010100  5380             176.          SUBQ.L #1,D0            ;ADJUST
00010102  43FA FCEC        177.          LEA TRAP4(PC),A1        ;DEST ADDR
00010106  12D8             178.  CPYLOP  MOVE.B (A0)+,(A1)+
00010108  51C8 FFFC        179.          DBRA D0,CPYLOP          ;COPY LOOP
                           180.  *
                           181.  * TABLE LOADED, INITIALIZE PERIPHERALS
                           182.  *
000101JC  13FC 0015        183.          MOVE.B #ACSTAT,ACIA1
          00012FFC
00010114  13FC 0015        184.          MOVE.B #ACSTAT,ACIA2
          00012FFE
                           185.  *
                           186.  * INITIALIZE PI/T
                           187.  * PORT A INPUT HANDSHAKE MODE
                           188.  * PORT B OUTPUT HANDSHAKE MODE
                           189.  * PORT C PC0: SLAVE IRQ* INPUT
                           190.  *        PC1: TSTACK* INPUT
                           191.  *        PC2: ADDR0 OUTPUT
                           192.  *        PC3: ADDR1 OUTPUT
                           193.  *        PC4: TSTCYCLE* OUTPUT
                           194.  * PC5-PC7 INPUTS, UNUSED
                           195.  *
0001011C  4239 00012FC0    196.          CLR.B PGCR      ;DISABLES SENSING A AND B
00010122  4239 00012FC1    197.          CLR.B PSRR      ;KILL INTERRUPTS
00010128  13FC 0000        198.          MOVE.B #INPUTS,PADDR
          00012FC2
00010130  13FC 00FF        199.          MOVE.B #OUTPUTS,PBDDR
          00012FC3
00010138  13FC 0030        200.          MOVE.B #ACONT,PACR
          00012FC6
00010140  13FC 0070        201.          MOVE.B #BCONT,PBCR
          00012FC7
```

```
00010148  4239 00012FC1   202.          CLR.B PSRR             ;FNCTN OF PORT C
0001014E  13FC 00FF       203.          MOVE.B #$FF,PCDR       ;INACT ON ENABLE
          00012FCC
00010156  13FC 001C       204.          MOVE.B #CCONFIG,PCDDR ;CONFIG DIRECTIONS
          00012FC4
                          205.  *
                          206.  * CLEAR THE BREAKPOINTS
0001015E  41FA FE40       207.          LEA BRKNUM(PC),A0
00010162  4250            208.          CLR.W (A0)
                          209.  * UPON RESTART ALL SLAVES WILL SIGNAL MASTER
                          210.  * THAT THEY ARE READY. SENSE THE CONDITION AND
                          211.  * ACKNOWLEDGE ALL SLAVES. THIS PUTS THEM IN
                          212.  * PROGRAM ACCEPT MODE
                          213.  *
00010164  103C 000F       214.          MOVE.B #$0F,D0          ;EXPECT ALL
00010168  4E49            215.          TRAP #9                 ;SLAVES
                          216.  *
                          217.  * BE SURE WERE ARE USING INITIAL SSP
                          218.  *
0001016A  2E7C 0000FDE0   219.  REINIT  MOVE.L #INITSSP,SP       ;LOAD WITH
                          220.  *                               ;INITIAL SP
                          221.  * NOW CAN ENABLE THE INTERRUPTS
                          222.  *
00010170  46FC 2000       223.          MOVE.W #$2000,SR        ;INT LEVEL=0
                          224.  *
                          225.  * NOW GOTO COMMAND LEVEL.
                          226.  *
                          227.  *       COMMAND HANDLER
                          228.  ********************************************
                          229.  * COMMAND HANDLER ROUTINE. ENTERED AFTER    *
                          230.  * SYSTEM INITIALIZATION. ACCEPTS USER INPUT *
                          231.  * AND FORCES JUMP TO SPECIFIED ROUTINE. DOES*
                          232.  * STRING COMPARISON ON EXISTANT COMMANDS    *
                          233.  * TO PRESERVE REGISTERS FOR EXAMINATION, A1 *
                          234.  * AND D1 ARE STACKED. A0 AND D0 ARE VOLTILE *
                          235.  * AS USUAL                                  *
                          236.  * COMMAND TABLE FORMAT:
                          237.  * -LENGTH OF ENTRY IN BYTES
                          238.  * -COMMAND STRING
                          239.  * -2 BYTE OFFSET TO COMMAND ROUTINE FROM     *
                          240.  *      CURRENT POSITION
                          241.  ********************************************
                          242.  * ROUTINES CAN BE CALLED AFTER THESE
                          243.  * DELIMITERS: SPACE  AND CR
                          244.  *
00010174  48E7 C0C0       245.  CMDLVL  MOVEM.L D0-D1/A0-A1,-(SP)
00010178  6100 12C8       246.          BSR SENCLFC             ;OUTPUT
0001017C  103C 003E       247.          MOVE.B #ASCGT,D0        ;'>' PROMPT
00010180  6100 12A8       248.          BSR SENDBYC
00010184  4281            249.          CLR.L D1          ;D1 IS CHR COUNTER
00010186  43FA FDE8       250.          LEA CMDSTK(PC),A1       ;PNT TO CMDSTK
0001018A  41FA 00EC       251.          LEA CMDTBL(PC),A0       ;PNT TO CMDTBL
0001018E  6100 12FA       252.  CMDMOR  BSR INCHCON     ;GET INPUT
00010192  0200 007F       253.          ANDI.B #ASCMSK,D0       ;MASK HI BIT
00010196  0C00 000D       254.          CMPI.B #CR,D0   ;CR IS DELIMITER?
0001019A  67 1E           255.          BEQ.S VALCR
0001019C  0C00 0020       256.          CMPI.B #SPACE,D0        ;SPACE?
000101A0  67 1E           257.          BEQ.S VALDEL
000101A2  0C00 0021       258.          CMPI.B #$21,D0  ;IS IT <!?
000101A6  6D 7A           259.          BLT.S INVALID
000101A8  0C00 007F       260.          CMPI.B #$7F,D0  ;IS IT DEL?
000101AC  67 74           261.          BEQ.S INVALID
                          262.  * CHR INPUT IS VALID, ECHO IT AND PLACE IT AT
                          263.  * CMDSTK IF IT IS NOT A DELIMITER
000101AE  6100 127A       264.          BSR SENDBYC
000101B2  12C0            265.          MOVE.B D0,(A1)+
000101B4  0601 0001       266.          ADDI.B #1,D1    ;MAINTAIN CHR COUNTER
000101B8  60 D4           267.          BRA.S CMDMOR    ;MOR INPUT CHRS
000101BA  6100 1286       268.  VALCR   BSR SENCLFC
000101BE  60 04           269.          BRA.S SKIPBYC
000101C0  6100 1268       270.  VALDEL  BSR SENDBYC      ;ECHO DELIMITER
000101C4  0C01 0000       271.  SKIPBYC CMPI.B #0,D1     ;ANY DATA?
000101C8  67 44           272.          BEQ.S ENDCMD
                          273.  * COMMAND STRING IS AT CMDSTK, BEGIN COMPARE
                          274.  * OF VALID STRINGS
000101CA  43FA FDA4       275.          LEA CMDSTK(PC),A1       ;RESTORE PNTR
000101CE  1010            276.  POLLTBL MOVE.B (A0),D0   ;GET LENGTH ENTRY
000101D0  6B 64           277.          BMI.S ERRCMD    ;TABLE DONE IF -1
000101D2  0240 00FF       278.          ANDI.W #$FF,D0  ;SET TO WORD
000101D6  5940            279.          SUBQ.W #4,D0    ;ADJUST THE INDEX
```

```
000101D8  1230 0001        280.  NEXTCHR MOVE.B 1(A0,D0.W),D1     ;GET CHR
000101DC  B231 0000        281.          CMP.B 0(A1,D0.W),D1      ;SAME CHRS?
000101E0  66 34            282.          BNE.S NEXTENT    ;NO GOOD NEXT ENTRY
000101E2  51C8 FFF4        283.          DBRA D0,NEXTCHR ;TRY NEXT CHR
                           284.  * COMMAND HAS BEEN FOUND, A0 IS STILL POINTING
                           285.  * TO ENTRY LENGTH OF CHOSEN CMD
000101E6  4280             286.          CLR.L D0
000101E8  4281             287.          CLR.L D1
000101EA  1010             288.          MOVE.B (A0),D0  ;RESTOR LENGTH
000101EC  3230 00FE        289.          MOVE.W -2(A0,D0.W),D1    ;GET OFFSET
000101F0  41F0 00FE        290.          LEA -2(A0,D0.W),A0       ;PNT TO OFFSET
000101F4  D1C1             291.          ADDA.L D1,A0     ;COMPUTE ADDR OF ROUT
000101F6  43FA FDCE        292.          LEA TEMPRET(PC),A1
000101FA  2288             293.          MOVE.L A0,(A1)
000101FC  4CDF 0303        294.          MOVEM.L (SP)+,D0-D1/A0-A1      ;RESTORE
                           295.  * THE FOLLOWING IS A STACK TRICK TO ALLOW
                           296.  * REGISTERS USED IN CMDLVL TO BE PRESERVED
                           297.  * REGISTERS WILL BE ALTERED BY COMMAND IMPS
                           298.  * BUT THE R COMMAND WILL NOT
                           299.  * THE SELECTED COMMAND IMP. ADDRESS IS PUT
                           300.  * ON THE STACK, AND THUS RTS WILL UNSTACK
                           301.  * THE ADDRES AND CAUSE JUMP.  THE RETURN
                           302.  * ADDRESS TO CMDLVL IS ALSO ON STACK, TO
                           303.  * BE USED BY THE COMIMP RTS INTSRUCTION
                           304.  * EFFECT IS SAME AS
                           305.  *       JSR(A0)
                           306.  * WHERE A0 IS JUMP ADDRESS
                           307.  * WITHOUT LOSING ORIGINAL CONTENTS OF A0
                           308.  *
00010200  487A 0008        309.          PEA RETCOM(PC)   ;FOR RETURN
00010204  2F3A FDC0        310.          MOVE.L TEMPRET(PC),-(SP)  ;4 DUMMY RTS
00010208  4E75             311.          RTS              ;GOTO TO ROUTINE
0001020A  6000 FF68        312.  RETCOM  BRA CMDLVL       ;GET NEXT COMMAND
0001020E  4CDF 0303        313.  ENDCMD  MOVEM.L (SP)+,D0-D1/A0-A1       ;RESTORE
00010212  6000 FF60        314.          BRA CMDLVL       ;NEXT COMMAND
00010216  1010             315.  NEXTENT MOVE.B (A0),D0  ;GET BACK LGTH  ENTRY
00010218  0280 000000FF    316.          ANDI.L #$FF,D0   ;MASK UNWANTED BITS
0001021E  D1C0             317.          ADDA.L D0,A0     ;ADJUST POINTER
00010220  60 AC            318.          BRA.S POLLTBL    ;LOOK AT NEXT ENTRY
00010222  6100 1206        319.  INVALID BSR SENDBYC      ;ECHO BAD DELIMITER
00010226  41FA 001E        320.          LEA INVAMES(PC),A0       ;INVALID MSG
0001022A  6100 123A        321.          BSR OUTMESC
0001022E  4CDF 0303        322.          MOVEM.L (SP)+,D0-D1/A0-A1 ;RESTORE
00010232  6000 FF40        323.          BRA CMDLVL
00010236  41FA 0028        324.  ERRCMD  LEA COMERMS(PC),A0
0001023A  6100 122A        325.          BSR OUTMESC
0001023E  4CDF 0303        326.          MOVEM.L (SP)+,D0-D1/A0-A1
00010242  6000 FF30        327.          BRA CMDLVL
                           328.  *
                           329.  *
00010246  20 20 69 73 20   330.  INVAMES DC.B '  is an Invalid Delimiter',EOT
          61 6E 20 49 6E
          76 61 6C 69 64
          20 44 65 6C 69
          6D 69 74 65 72
          04
00010260  55 6E 72 65 63   331.  COMERMS DC.B 'Unrecognized Command',EOT
          6F 67 6E 69 7A
          65 64 20 43 6F
          6D 6D 61 6E 64
          04
                           332.  *
                           333.  ******************************************
                           334.  * COMMAND TABLE                          *
                           335.  ******************************************
                           336.  *
          -- boundary align
00010276  0000             337.  EVEFIX  DC.W 00
    @ 00010278             338.  CMDTBL  EQU *
00010278  04 52            339.          DC.B 4,'R'
0001027A  0224             340.          DC.W REGLOOK-*
0001027C  04 44            341.          DC.B 4,'D'
0001027E  0022             342.          DC.W DISPMEM-*
00010280  04 4D            343.          DC.B 4,'M'
00010282  0158             344.          DC.W MEMEDT-*
00010284  04 46            345.          DC.B 4,'F'
00010286  030C             346.          DC.W FILLMEM-*
00010288  04 54            347.          DC.B 4,'T'
0001028A  03CE             348.          DC.W TALK-*
0001028C  06 4C 4F 41      349.          DC.B 6,'LOA'
```

```
00010290  0426              350.          DC.W DOWNLOD-*
00010292  04 43             351.          DC.B 4,'C'
00010294  08D2              352.          DC.W CALSPRG-*
00010296  04 47             353.          DC.B 4,'G'
00010298  0946              354.          DC.W GOUPRG-*
0001029A  04 42             355.          DC.B 4,'B'
0001029C  0726              356.          DC.W BRKHAN-*
0001029E  FF                357.          DC.B -1
                            358.  *
                            359.  *
                            360.  *       COMMAND IMPLEMENTATIONS
                            361.  *
                            362.  ********************************************
                            363.  * COMMAND IMPLEMENTATIONS FOR DISPLAY,      *
                            364.  * MEMORY CHANGE, REGISTERS,FILL,            *
                            365.  * TALK,AND DOWNLOAD
                            366.  ********************************************
                            367.  *
                            368.  * DISPLAY MEMORY CONTENTS ROUTINE
                            369.  * CALLED ONLY BY THE COMMAND HANDLER AS
                            370.  * A SUBROUTINE AFTER D SPACE. MORE INPUT
                            371.  * NEEDED AND IS HANDLED BY THIS MODULE. WILL
                            372.  * PUT START ADDRESS IN A1, END
                            373.  * ADDRESS IN A2. ADDRESS BLOCK IS ROUNDED
                            374.  * TO THE NEAREST 16 BYTES. NEARLY ALL OF THE
                            375.  * REGISTERS TAKE A REAL BEATING
                            376.  *
                            377.  *
                            378.  *
        -- boundary align
000102A0  2C7C 00012FFC     379.  DISPMEM MOVEA.L #ACIA1,A6       ;DEFINE PORT
000102A6  6100 12CA         380.          BSR BLDNUM       ;GET FIRST ADDRESS
000102AA  3208              381.          MOVE.W A0,D1
000102AC  0C01 0020         382.          CMPI.B #SPACE,D1         ;SPC IS VALID
000102B0  6600 00BA         383.          BNE ERRDISP
000102B4  08C1 000F         384.          BSET #15,D1      ;DATA PRESENT?
000102B8  6600 009C         385.          BNE DEFSTRT      ;USE A DEFAULT VALUE
000102BC  2240              386.          MOVE.L D0,A1     ;START ADDRESS
000102BE  6100 12B2         387.  INENDAR BSR BLDNUM       ;GET NEXT DATA
000102C2  3208              388.          MOVE.W A0,D1
000102C4  0C01 000D         389.          CMPI.B #CR,D1    ;CR IS VALID DELIM
000102C8  6600 00A2         390.          BNE ERRDISP
000102CC  08C1 000F         391.          BSET #15,D1      ;DATA PRESENT?
000102D0  6600 008E         392.          BNE DEFEND       ;USE DEFAULT VALUE
000102D4  2440              393.          MOVE.L D0,A2     ;LOAD END ADDRESS
000102D6  72F0              394.  DODISP  MOVE.L #NIBMSK,D1        ;NRMLIZE ADDR
000102D8  2409              395.          MOVE.L A1,D2             ;TO 16 BYTE
000102DA  260A              396.          MOVE.L A2,D3             ;BOUNDS.
000102DC  C481              397.          AND.L D1,D2
000102DE  C681              398.          AND.L D1,D3
000102E0  2443              399.          MOVEA.L D3,A2
000102E2  2242              400.          MOVEA.L D2,A1
                            401.  *
                            402.  * BEGIN TO SEND OUT DISPLAY, 1ST THE HEADER
                            403.  *
000102E4  6100 1108         404.          BSR SENDCLF
000102E8  41FA 0090         405.  MORBLK  LEA HEADER(PC),A0
000102EC  6100 1178         406.          BSR OUTMESC
000102F0  6100 10FC         407.          BSR SENDCLF              ;CR LF
                            408.  *
                            409.  * NOW OUTPUT THE ADDRESS AND DATA INFORMATION
                            410.  *
000102F4  363C 2020         411.          MOVE.W #TOSPC,D3
000102F8  7A07              412.          MOVE.L #BLKCNT,D5        ;BLOCK COUNTER
000102FA  7C0F              413.  ANUTHLN MOVE.L #LINCNT,D6        ;LINE COUNTER
                            414.  * OUTPUT THE ADDRESS
000102FC  2009              415.          MOVE.L A1,D0             ;NOTE THAT
000102FE  6100 1226         416.          BSR DISBUF8              ;OUTPUT USES
00010302  3003              417.          MOVE.W D3,D0             ;DISBUF, THESE
00010304  6100 10D6         418.          BSR SENDWO               ;LOCS CHANGING
                            419.  * AND NOW CAN SEND THE ACTUAL DATA
00010308  1019              420.  MORBYTE MOVE.B (A1)+,D0          ;GET THE DATA
0001030A  6100 124C         421.          BSR DISBUF2              ;SEND THE DATA
0001030E  3003              422.          MOVE.W D3,D0             ;SEND 1 SPACE
00010310  6100 10D0         423.          BSR SENDBY
00010314  51CE FFF2         424.          DBRA D6,MORBYTE          ;DO 16 LCATONS
00010318  7C0F              425.          MOVE.L #LINCNT,D6        ;RESET COUNT
```

```
                                426.  * SEND OUT THE ACSII EQUIVALENTS
0001031A  6100 10C0             427.          BSR SENDWO                ;2 SPACES
0001031E  93FC 00000010         428.          SUBA.L #$10,A1            ;RESET PNTR
00010324  1019                  429.  MORASC  MOVE.B (A1)+,D0           ;GET DATA
00010326  0C00 007E             430.          CMPI.B #$7E,D0            ;DO RANGE CHK
0001032A  6E 10                 431.          BGT.S DOTOUT              ;20 TO 7E
0001032C  0C00 0020             432.          CMPI.B #$20,D0
00010330  6D 0A                 433.          BLT.S DOTOUT
00010332  6100 10AE             434.  NORMOUT BSR SENDBY                ;SEND BYTE
00010336  51CE FFEC             435.          DBRA D6,MORASC            ;DO 16 TIMES
0001033A  60 06                 436.          BRA.S CHKDUN
0001033C  103C 002E             437.  DOTOUT  MOVE.B #ASCDOT,D0
00010340  60 F0                 438.          BRA.S NORMOUT
                                439.  *
                                440.  * A WHOLE SINGLE LINE HAS BEEN SENT, DO CR LF
                                441.  * AND CHECK FOR MORE
                                442.  *
00010342  6100 10AA             443.  CHKDUN  BSR SENDCLF
00010346  B3CA                  444.          CMPA.L A2,A1              ;WHEN A1>A2
00010348  6E 0A                 445.          BGT.S DUNMEM              ;WILL QUIT
0001034A  51CD FFAE             446.          DBRA D5,ANUTHLN           ;ELSE MORE LN
0001034E  6100 109E             447.          BSR SENDCLF               ;AND NEW HDR
00010352  60 94                 448.          BRA MORBLK                ;IF NECESSARY
00010354  4E75                  449.  DUNMEM  RTS                       ;BACK TO COMD
00010356  227C 00000000         450.  DEFSTRT MOVE.L #DEFASTR,A1
0001035C  6000 FF60             451.          BRA INENDAR
00010360  2449                  452.  DEFEND  MOVEA.L A1,A2        ;DISPLAY 16 BYTES
00010362  D5FC 0000000F         453.          ADDA.L #$F,A2
00010368  6000 FF6C             454.          BRA DODISP
0001036C  6100 10D4             455.  ERRDISP BSR SENCLFC
00010370  41FA 004C             456.          LEA.L BADDEL(PC),A0
00010374  6100 10F0             457.          BSR OUTMESC
00010378  60 DA                 458.          BRA.S DUNMEM              ;TO CMD LEVEL
                                459.
                                460.  *
                                461.  * DATA SECTION HEADER
                                462.  *
0001037A  20 20 20 20 20        463.  HEADER  DC.B '          0  1  2  3  4  5  6  '
          20 20 20 20 20
          30 20 20 31 20
          20 32 20 20 33
          20 20 34 20 20
          35 20 20 36 20
          20
00010399  37 20 20 38 20        464.          DC.B '7  8  9  A  B  C  D  E  F     '
          20 39 20 20 41
          20 20 42 20 20
          43 20 20 44 20
          20 45 20 20 46
          20 20 20 20 20
000103B7  41 53 43 49 49        465.          DC.B 'ASCII '
          20
000103BD  04                    466.          DC.B EOT
000103BE  20 49 6E 76 61        467.  BADDEL  DC.B ' Invalid Command Delimiter',EOT
          6C 69 64 20 43
          6F 6D 6D 61 6E
          64 20 44 65 6C
          69 6D 69 74 65
          72 04
                                468.  *
                                469.  *
                                470.  * MEMORY EXAMINE,MODIFY ROUTINE
                                471.  * ALLOWS USER TO CHANGE MEMORY BYTE BY BYTE
                                472.  * COMMAND LEVEL IS THE CALLER, AFTER  M SPC
                                473.  * MORE INPUT REQUIRED, HANDLED BY THIS
                                474.  * MODULE. WILL LOAD A1
                                475.  * WITH THE FIRST ADDRESS TO MODIFY
                                476.  *
                                477.  *
                                478.  *
                                479.  *
          -- boundary align
000103DA  2C7C 00012FFC         480.  MEMEDT  MOVE.L #ACIA1,A6          ;TERM PORT
000103E0  6100 1190             481.          BSR BLDNUM          ;GET INPUT
000103E4  3208                  482.          MOVE.W A0,D1
000103E6  0C01 000D             483.          CMPI.B #CR,D1       ;VALID DELIM IS CR
000103EA  6600 0088             484.          BNE ERRMEM
000103EE  08C1 000F             485.          BSET #15,D1         ;DATA PRESENT?
000103F2  6600 0078             486.          BNE DEFMEM          ;USE A DEFAULT VALUE
000103F6  2240                  487.          MOVEA.L D0,A1       ;LOAD ADDRESS
```

```
000103F8  6100 0FF4        488.  MEMMOR  BSR SENDCLF
000103FC  2009             489.          MOVE.L A1,D0              ;SET UP DISPLY
000103FE  6100 1126        490.          BSR DISBUF8               ,OUTPT ADDRESS
00010402  303C 203E        491.          MOVE.W #SPASGT,D0         ;SP >
00010406  6100 0FD4        492.          BSR SENDWO
0001040A  1011             493.          MOVE.B (A1),D0            ;GET DATA
0001040C  6100 114A        494.          BSR DISBUF2               ;OUTPT DATA
00010410  303C 202D        495.          MOVE.W #SPDASH,D0         ;SP -
00010414  6100 0FC6        496.          BSR SENDWO
                           497.  *
                           498.  * NOW WAIT FOR USER INPUT
                           499.  *
00010418  6100 1158        500.  WATIN   BSR BLDNUM
                           501.  *
                           502.  * DETERMINE WHAT INPUT IS AND WHAT TO DO
                           503.  *
0001041C  3208             504.          MOVE.W A0,D1     ;KEEP A COPY OF THE CHR
                           505.  * DETERMINE VALID DELIMITER
0001041E  0C01 000D        506.          CMPI.B #CR,D1    ;IS IT A CR?
00010422  67 1C            507.          BEQ.S CRINM
00010424  0C01 000A        508.          CMPI.B #LF,D1    ;IS IT LF?
00010428  67 20            509.          BEQ.S LFINM
0001042A  0C01 005E        510.          CMPI.B #UPAR,D1 ; IS IT ^?
0001042E  67 26            511.          BEQ.S UPARINM
00010430  1001             512.          MOVE.B D1,D0     ; INVALID DELIMITER
00010432  6100 0FAE        513.          BSR SENDBY       ;CHECK FOR ERROR
00010436  41FA 0046        514.          LEA WRONGDA(PC),A0       ;ERROR MESS
0001043A  6100 102A        515.          BSR OUTMESC
0001043E  4E75             516.          RTS
00010440  6100 0020        517.  CRINM   BSR CHKUPT       ;CHECK FOR UPDATE
00010444  6100 0FA8        518.          BSR SENDCLF
00010448  4E75             519.          RTS
0001044A  6100 0016        520.  LFINM   BSR CHKUPT
0001044E  D3FC 00000001    521.          ADDA.L #1,A1     ;PNT TO NEXT ADDR
00010454  60 A2            522.          BRA.S MEMMOR
00010456  6100 000A        523.  UPARINM BSR CHKUPT
0001045A  93FC 00000001    524.          SUBA.L #1,A1
00010460  60 96            525.          BRA.S MEMMOR
00010462  0801 000F        526.  CHKUPT  BTST #15,D1       ;NEW DATA?
00010466  66 02            527.          BNE.S NOUPT
00010468  1280             528.          MOVE.B D0,(A1)   ;UPDATE ONLY BYTE
0001046A  4E75             529.  NOUPT   RTS
0001046C  227C 00000000    530.  DEFMEM  MOVEA.L #DEFASTR,A1 ;DEFAULT TO 0
00010472  60 84            531.          BRA.S MEMMOR
00010474  41FA FF48        532.  ERRMEM  LEA BADDEL(PC),A0 ; BAD DELIM MESS
00010478  6100 0FEC        533.          BSR OUTMESC
0001047C  4E75             534.          RTS              ;TO COMMAND LEVEL
                           535.  *
                           536.  *
0001047E  20 20 49 53 20   537.  WRONGDA DC.B '  IS AN ILLEGAL INPUT CHARACTER'
          41 4E 20 49 4C
          4C 45 47 41 4C
          20 49 4E 50 55
          54 20 43 48 41
          52 41 43 54 45
          52
0001049D  04               538.          DC.B EOT
                           539.  *
                           540.  *
                           541.  * REGISTER VIEW/MODIFY SUBROUTINE
                           542.  * CALLED BY THE COMMAND LEVEL AS A SUBROUTINE
                           543.  * NOTE THAT USER CAN ONLY MODIFY D0-D7,A0-A6
                           544.  * IN ORDER THAT SYSTEM ORDER IS PRESERVED
                           545.  *
0001049E  2F08             546.  REGLOOK MOVE.L A0,-(SP)
000104A0  41FA FACC        547.          LEA STRGTMP(PC),A0
000104A4  4210             548.          CLR.B (A0)
000104A6  205F             549.          MOVE.L (SP)+,A0
000104A8  6100 0C34        550.          BSR STOREG       ;STORE REGISTERS
000104AC  6100 0F94        551.          BSR SENCLFC     ;CR, LF
000104B0  6100 0C7A        552.          BSR OUTREG       ;DISPLAY ALL REGISTERS
000104B4  43FA FA58        553.          LEA REGSTOR(PC),A1       ;PNT  REGSTOR
000104B8  45FA 009C        554.          LEA REGTIT(PC),A2        ;PNT TO TTLS
000104BC  2649             555.          MOVEA.L A1,A3    ;SAVE COPY OF REGSTOR
000104BE  284B             556.          MOVEA.L A3,A4
000104C0  D9FC 00000038    557.          ADDA.L #56,A4
000104C6  6100 0F7A        558.  REGMOR  BSR SENCLFC
000104CA  2012             559.          MOVE.L (A2),D0   ;GET TITLE
000104CC  6100 0F2C        560.          BSR SENDLOC      ;PRINT TITLE
000104D0  2011             561.          MOVE.L (A1),D0   ;GET DATA
```

```
000104D2  6100 1052        562.          BSR DISBUF8      ;PRINT DATA
000104D6  303C 202D        563.          MOVE.W #SPDASH,D0        ;SPACE -
000104DA  6100 0F36        564.          BSR SENDWOC      ;PRINT SPC -
000104DE  4281             565.          CLR.L D1
                           566.  *
                           567.  * WAIT FOR USER INPUT
                           568.  *
000104E0  6100 1090        569.  WATINR  BSR BLDNUM
000104E4  3208             570.          MOVE.W A0,D1     ;COPY DATA
000104E6  0C01 000D        571.          CMPI.B #CR,D1    ;IS IT CR?
000104EA  67 48            572.          BEQ.S CRINR
000104EC  0C01 000A        573.          CMPI.B #LF,D1    ;IS IT LF?
000104F0  67 16            574.          BEQ.S LFINR
000104F2  0C01 005E        575.          CMPI.B #UPAR,D1 ;IS IT ^?
000104F6  67 26            576.          BEQ.S UPARINR
                           577.  * INVALID DELIMITER
000104F8  1001             578.          MOVE.B D1,D0     ;OFFENDING CHR
000104FA  6100 0F2E        579.          BSR SENDBYC
000104FE  41FA FF7E        580.          LEA WRONGDA(PC),A0     ;ERROR MSGE
00010502  6100 0F62        581.          BSR OUTMESC
00010506  4E75             582.          RTS
00010508  6100 0042        583.  LFINR   BSR UPDATER      ;UPDATE REG?
0001050C  D3FC 00000004    584.          ADDA.L #4,A1     ;POINT NEXT DATA
00010512  D5FC 00000004    585.          ADDA.L #4,A2     ;POINT TO NEXT TTL
00010518  B3CC             586.          CMPA.L A4,A1     ;RANGE CHECK
0001051A  6E 20            587.          BGT.S DUNCHG     ;QUIT AFTER A6
0001051C  60 A8            588.          BRA.S REGMOR     ;MOR STUFF
0001051E  6100 002C        589.  UPARINR BSR UPDATER      ;UPDATE REG?
00010522  93FC 00000004    590.          SUBA.L #4,A1     ;PREVIOUS ADDRESS
00010528  95FC 00000004    591.          SUBA.L #4,A2     ;PREVIOUS TITLE
0001052E  B3CB             592.          CMPA.L A3,A1     ;RANGE CHECK
00010530  6D 0A            593.          BLT.S DUNCHG     ;TOO FAR BACK?
00010532  60 92            594.          BRA REGMOR
00010534  6100 0016        595.  CRINR   BSR UPDATER      ;UPDATE REG?
00010538  6100 0F08        596.          BSR SENCLFC
0001053C  224B             597.  DUNCHG  MOVEA.L A3,A1    ;POINT TO REGSTOR
0001053E  4CD9 7FFF        598.          MOVEM.L (A1)+,D0-D7/A0-A6 ;RESTORE REG
00010542  43FA F9CA        599.          LEA REGSTOR(PC),A1 ;RESTORE A1 CONTENT
00010546  2269 0024        600.          MOVEA.L 36(A1),A1
0001054A  4E75             601.          RTS
0001054C  0801 000F        602.  UPDATER BTST #15,D1      ;NEW DATA?
00010550  66 02            603.          BNE.S NOUPDAT    ;NO UPDATE IF SET
00010552  2280             604.          MOVE.L D0,(A1)   ;UPDATE MEMORY FOR REG
00010554  4E75             605.  NOUPDAT RTS
                           606.  *
                           607.  *
00010556  44 30 2D 20 44   608.  REGTIT  DC.B 'D0- D1- D2- D3- D4- D5- D6- '
          31 2D 20 44 32
          2D 20 44 33 2D
          20 44 34 2D 20
          44 35 2D 20 44
          36 2D 20
00010572  44 37 2D 20 41   609.          DC.B 'D7- A0- A1- A2- A3- A4- A5- '
          30 2D 20 41 31
          2D 20 41 32 2D
          20 41 33 2D 20
          41 34 2D 20 41
          35 2D 20
0001058E  41 36 2D 20      610.          DC.B 'A6- '
                           611.  *
                           612.  *
                           613.  * FILL MEMORY BLOCK SUBROUTINE
                           614.  * PROVIDES SOMETHING TO DO ON V 1.00
                           615.  * CALLED BY COMMAND LEVEL WHEN FILL COMMAND
                           616.  * RECOGNIZED.
                           617.  *
00010592  6100 0EAE        618.  FILLMEM BSR SENCLFC
00010596  41FA 0066        619.          LEA STRTMES(PC),A0       ;PROMPTS
0001059A  6100 0ECA        620.          BSR OUTMESC              ;START ADDRESS
0001059E  6100 0FD2        621.          BSR BLDNUM
000105A2  6100 0E9E        622.          BSR SENCLFC
000105A6  B0FC 000D        623.          CMPA.W #CR,A0            ;VALID DELIM?
000105AA  66 3E            624.          BNE.S NOFILL
000105AC  2240             625.          MOVEA.L D0,A1            ;A1 IS INDEX
000105AE  41FA 006E        626.          LEA BLKMES(PC),A0        ;BLK LENGTH?
000105B2  6100 0EB2        627.          BSR OUTMESC
000105B6  6100 0FBA        628.          BSR BLDNUM
000105BA  6100 0E86        629.          BSR SENCLFC
000105BE  B0FC 000D        630.          CMPA.W #CR,A0            ;VALID DELIM?
000105C2  66 26            631.          BNE.S NOFILL
```

```
000105C4  2200              632.          MOVE.L D0,D1             ;D1 IS COUNTER
000105C6  41FA 0074         633.          LEA FILMES(PC),A0        ;FILL BYTE?
000105CA  6100 0E9A         634.          BSR OUTMESC
000105CE  6100 0FA2         635.          BSR BLDNUM
000105D2  6100 0E6E         636.          BSR SENCLFC
000105D6  B0FC 000D         637.          CMPA.W #CR,A0            ;VALID DELIM?
000105DA  66 0E             638.          BNE.S NOFILL
000105DC  0481 00000001     639.          SUBI.L #1,D1             ;SET FOR DBRA
000105E2  12C0              640.  FILLOOP MOVE.B D0,(A1)+          ;FILL WTH BYTE
000105E4  51C9 FFFC         641.          DBRA D1,FILLOOP
000105E8  4E75              642.          RTS
000105EA  3008              643.  NOFILL  MOVE.W A0,D0     ;
000105EC  6100 0E3C         644.          BSR SENDBYC      ;ECHO BAD DELIMITER
000105F0  41FA FE8C         645.          LEA WRONGDA(PC),A0       ;ERROR MESAGE
000105F4  6100 0E70         646.          BSR OUTMESC
000105F8  6100 0E48         647.          BSR SENCLFC
000105FC  4E75              648.          RTS
                            649.  *
000105FE  49 6E 70 75 74    650.  STRTMES DC.B 'Input Start Address for Fill - '
          20 53 74 61 72
          74 20 41 64 64
          72 65 73 73 20
          66 6F 72 20 46
          69 6C 6C 20 2D
          20
0001061D  04                651.          DC.B EOT
0001061E  49 6E 70 75 74    652.  BLKMES  DC.B 'Input Block Length to Fill - '
          20 42 6C 6F 63
          6B 20 4C 65 6E
          67 74 68 20 74
          6F 20 46 69 6C
          6C 20 2D 20
0001063B  04                653.          DC.B EOT
0001063C  49 6E 70 75 74    654.  FILMES  DC.B 'Input Byte to Fill with - ',EOT
          20 42 79 74 65
          20 74 6F 20 46
          69 6C 6C 20 77
          69 74 68 20 2D
          20 04
                            655.  *
                            656.  * TALK FUNCTION ROUTINE. CALLED BY COMMAND
                            657.  * LEVEL. ENABLES TRANSPARENT COMMUNICATION
                            658.  * WITH A HOST ON ACIA2 PORT
                            659.  *
          -- boundary align
00010638  287C 00012FFC    660.  TALK    MOVEA.L #ACIA1,A4        ;CONSOLE CHAN
0001065E  2A7C 00012FFE    661.          MOVEA.L #ACIA2,A5        ;HOST PORT
00010664  1014              662.  CHKLOOP MOVE.B (A4),D0           ;GET STAT
00010666  0800 0004         663.          BTST #4,D0       ;FRAMING ERROR?
0001066A  66 2E             664.          BNE.S SENBRAK    ;SEND BREAK LEVEL
0001066C  0800 0000         665.          BTST #0,D0       ;DATA PRESENT?
00010670  66 12             666.          BNE.S CONHOST    ;SEND TO HOST IF SO
00010672  0815 0000         667.  CHKHOST BTST #0,(A5)     ;DATA ON HOST CHAN?
00010676  67 EC             668.          BEQ.S CHKLOOP    ;BACK IF NOT
                            669.  * DATA PRESENT FROM HOST, ECHO TO CONSOLE
00010678  102D 0001         670.  HOSTCON MOVE.B 1(A5),D0 ;GET DATA
0001067C  2C4C              671.          MOVEA.L A4,A6    ;POINT TO CONSOLE
0001067E  6100 0D62         672.          BSR SENDBY
00010682  60 E0             673.          BRA.S CHKLOOP    ;DO AGAIN
00010684  102C 0001         674.  CONHOST MOVE.B 1(A4),D0 ;GET DATA
00010688  0200 007F         675.          ANDI.B #ASCMSK,D0 ;STRIP HI BIT
0001068C  0C00 0004         676.          CMPI.B #EOT,D0   ;IS IT EOT?
00010690  67 22             677.          BEQ.S TALKDUN    ;END IF NOT
00010692  2C4D              678.          MOVEA.L A5,A6
00010694  6100 0D4C         679.          BSR SENDBY       ;ECHO TO HOST
00010698  60 D8             680.          BRA.S CHKHOST    ;NOW CHECK HOST CHAN
0001069A  102C 0001         681.  SENBRAK MOVE.B 1(A4),D0  ;CLEAR THE ERROR
0001069E  1ABC 0075         682           MOVE.B #BRKLEV,(A5) ;RECONFIG PORT
000106A2  223C 00000C80     683.          MOVE.L #WATCONT,D1  ;INIT COUNTER
000106A8  4E71              684.  WATLOOP NOP
000106AA  51C9 FFFC         685.          DBRA D1,WATLOOP    ;SEND FOR 20 MS
000106AE  1ABC 0015         686.          MOVE.B #ACSTAT,(A5) ;PUT BACK TO
000106B2  60 B0             687.          BRA CHKLOOP      ;NORMAL
000106B4  4E75              688.  TALKDUN RTS
                            689.  *
                            690.  * COMMAND FUNCTION DOWNLOAD
                            691.  * ALLOWS THE DOWNLOAD OF AN S-RECORD FROM
                            692.  * THE HOST CHANNEL. MANY REGISTERS TRASHED
                            693.  * BUT IT DOESN'T MATTER.
                            694.  * PROMPTS USER TO INPUT OFFSET OF LOAD,
```

```
                                695.  * AND PROMPTS FOR FILENAME.
                                696.  * CALLS TO OUTMESC,BLDNUM,SENCLFC,BLDSTRG
                                697.  *          OUTMESH,SENCLFH,DISBUF8,LOADER
                                698.  *
                                699.  * STRGBUF IS IN WKPGE
                                700.  *
000106B6  41FA 00BC             701.  DOWNLOD LEA LOFFSET(PC),A0  ;PROMPT FOR OFST
000106BA  6100 0DAA             702.          BSR OUTMESC
000106BE  6100 0EB2             703.          BSR BLDNUM        ;GET THE OFFSET
000106C2  B0FC 000D             704.          CMPA.W #CR,A0
000106C6  6600 0080             705.          BNE ERRLOD
000106CA  2240                  706.          MOVE.L D0,A1      ;A1 HOLDS OFFSET
000106CC  6100 0D74             707.  FILSPEC BSR SENCLFC
000106D0  41FA 00BE             708.          LEA FILNAM(PC),A0   ;PROMPT FOR FILE
000106D4  6100 0D90             709.          BSR OUTMESC
000106D8  41FA F8B6             710.          LEA STRGBUF(PC),A0
000106DC  700D                  711.          MOVEQ.L #$0D,D0       ;CHR LIMIT
000106DE  6100 0EDC             712.          BSR BLDSTRG           ;BUILD CHR STRING
000106E2  0C00 0000             713.          CMPI.B #0,D0      ;SEE IF TOO LONG
000106E6  6700 0056             714.          BEQ TOOLONG
000106EA  10BC 0004             715.          MOVE.B #EOT,(A0)      ;INSERT EOT
                                716.  * modification 07/27/88 for pc compatibilty
                                717.  * wait for echo of command line
000106EE  41FA 00F0             718.          LEA TYPMES(PC),A0   ;GET TYPE MESSAGE
000106F2  6100 005E             719.          BSR SENBOTH
000106F6  41FA F898             720.          LEA STRGBUF(PC),A0  ;GET STRING
000106FA  6100 0056             721.          BSR SENBOTH
000106FE  6100 0D42             722.          BSR SENCLFC           ;CRLF TO TERM
00010702  6100 0D4A             723.          BSR SENCLFH
00010706  41FA 010F             724.          LEA LOADMES(PC),A0  ;LOADING...
0001070A  6100 0D5A             725.          BSR OUTMESC
0001070E  2C7C 00012FFE         726.          MOVEA.L #ACIA2,A6   ;POINT TO ACIA 2
00010714  6100 012A             727.          BSR LOADER
00010718  2C7C 00012FFC         728.          MOVEA.L #ACIA1,A6   ;POINT TO ACIA1
0001071E  6100 0D2E             729.          BSR SENCLFH
00010722  41FA 00C2             730.          LEA DNLOMES(PC),A0  ;FINISH MESSAGE
00010726  6100 0D3E             731.          BSR OUTMESC
0001072A  2002                  732.          MOVE.L D2,D0          ;GET ERROR COUNT
0001072C  6100 0DF8             733.          BSR DISBUF8           ;SEND ERRORS
00010730  41FA 00D1             734.          LEA EDLOMES(PC),A0  ;END OF MESSAGE
00010734  6100 0D30             735.          BSR OUTMESC
00010738  6100 0D08             736.          BSR SENCLFC
0001073C  4E75                  737.          RTS                   ;FINISH LOAD
0001073E  41FA 0080             738.  TOOLONG LEA LONGMES(PC),A0  ;TOO MANY CHRS
00010742  6100 0D22             739.          BSR OUTMESC           ;IN FILENAME
00010746  60 84                 740.          BRA FILSPEC
00010748  41FA 00D8             741.  ERRLOD  LEA NOLD(PC),A0       ;INVALID OFFSET
0001074C  6100 0D18             742.          BSR OUTMESC           ;ABORT LOAD
00010750  4E75                  743.          RTS
00010752  1018                  744.  SENBOTH MOVE.B (A0)+,D0
00010754  1200                  745.          MOVE.B D0,D1
00010756  0C00 0004             746.          CMPI.B #EOT,D0
0001075A  67 16                 747.          BEQ.S ENDBOTH
0001075C  6100 0CCC             748.          BSR SENDBYC
00010760  6100 0CD2             749.          BSR SENDBYH
00010764  6100 0D32             750.  WATECHO BSR INCHHOS
00010768  0200 007F             751.          ANDI.B #ASCMSK,D0
0001076C  B001                  752.          CMP.B D1,D0
0001076E  66 F4                 753.          BNE.S WATECHO
00010770  60 E0                 754.          BRA.S SENBOTH
00010772  4E75                  755.  ENDBOTH RTS
                                756.  *
00010774  49 6E 70 75 74        757.  LOFFSET DC.B 'Input Load Offset in HEX > '
          20 4C 6F 61 64
          20 4F 66 66 73
          65 74 20 69 6E
          20 48 45 58 20
          3E 20
0001078F  04                    758.          DC.B EOT
00010790  49 6E 70 75 74        759.  FILNAM  DC.B 'Input Filename to Load (<1-7'
          20 46 69 6C 65
          6E 61 6D 65 20
          74 6F 20 4C 6F
          61 64 20 28 3C
          31 2D 37
000107AC  3E 63 68 72 73        760.          DC.B '>chrs.<1-3> chrs) >',EOT
          2E 3C 31 2D 33
          3E 20 63 68 72
          73 29 20 3E 04
000107C0  46 69 6C 65 6E        761.  LONGMES DC.B 'Filename is Too Long, Re-ent'
```

```
          61 6D 65 20 69
          73 20 54 6F 6F
          20 4C 6F 6E 67
          2C 20 52 65 2D
          65 6E 74
000107DC  65 72 2E 04      762.          DC.B 'er.',EOT
000107E0  54 59 50 45 20   763.  TYPMES   DC.B 'TYPE ',EOT
          04
000107E6  4C 6F 61 64 20   764.  DNLOMES DC.B 'Load Complete.  There were $'
          43 6F 6D 70 6C
          65 74 65 2E 20
          20 54 68 65 72
          65 20 77 65 72
          65 20 24
00010802  04               765.          DC.B EOT
00010803  20 45 72 72 6F   766.  EDLOMES DC.B ' Errors Encountered',EOT
          72 73 20 45 6E
          63 6F 75 6E 74
          65 72 65 64 04
00010817  4C 6F 61 64 69   767.  LOADMES DC.B 'Loading...',EOT
          6E 67 2E 2E 2E
          04
00010822  49 6E 76 61 6C   768.  NOLD    DC.B 'Invalid Offset Specification'
          69 64 20 4F 66
          66 73 65 74 20
          53 70 65 63 69
          66 69 63 61 74
          69 6F 6E
0001083E  04               769.          DC.B EOT
                           770.  *
                           771.  *
                           772.  * SUBROUTINE LOADER
                           773.  * DOES THE ACTUAL DOWNLOAD OF THE S RECORD
                           774.  * DUMPED FROM THE HOST CHANNEL.
                           775.  * INITIAL CONDITIONS:
                           776.  * A1 CONTAINS THE ADDRESS OFFSET VALUE
                           777.  * A6 POINTS TO PORT TO LOAD FROM
                           778.  * ERROR COUNT IS RETURNED IN D2
                           779.  * REGISTER USAGE:
                           780.  * D0 ...DATA TRANSFERS VIA INCH,ASCHEX
                           781.  * D1 ...BYTE COUNT REGISTER
                           782.  * D2 ...ERROR COUNT ACCUMULATOR
                           783.  * D3 ...ADDRESS BYTE COUNTER
                           784.  * D4 ...TEMP. REGISTER FOR MATH OPN'S
                           785.  *       ON ADDRESS VALUES
                           786.  * D6 ...CHECKSUM ACCUMULATOR
                           787.  * A0 ...CONSOLE MESSAGE POINTER
                           788.  * A1 ...HOLDS OFFSET FOR ADDRESSES
                           789.  * A2 ...LOAD ADDRESS FROM S-RECORD
                           790.  * CALLS TO GRABNSM,INCH,ASCHEX
                           791.  *
          -- boundary align
00010840  4280             792.  LOADER  CLR.L D0
00010842  4281             793.          CLR.L D1
00010844  4282             794.          CLR.L D2
00010846  4286             795.          CLR.L D6
00010848  4287             796.          CLR.L D7
                           797.  * LOOK FOR HEADER RECORD
0001084A  6100 0C5A        798.  STRTLP  BSR INCH         ;GET 1ST CHR
0001084E  0200 007F        799.          ANDI.B #ASCMSK,D0
00010852  0C00 0053        800.          CMPI.B #'S',D0   ;IS IT 'S'?
00010856  66 F2            801.          BNE.S STRTLP     ;WAIT TILL SO
00010858  6100 0C4C        802.          BSR INCH         ;GET NEXT CHR
0001085C  0200 007F        803.          ANDI.B #ASCMSK,D0
00010860  0C00 0030        804.          CMPI.B #'0',D0   ;IS IT '0'?
00010864  6600 008A        805.          BNE NOHDR        ;FATAL, NO HEADER
                           806.  * GET BYTE COUNTER AND ADJUST FOR CORRECT
                           807.  * ACTION
00010868  6100 014C        808.          BSR GETCUNT
                           809.  * QUICKLY ZIP THROUGH HEADER RECORD
0001086C  6100 0114        810.  GETHEAD BSR GRABNSM      ;BYTE,COUNT,SUM
00010870  51C9 FFFA        811.          DBRA D1,GETHEAD ;DO TILL DONE
00010874  6100 012E        812.          BSR CHKSUM       ;VERIFY CHECKSUM
                           813.  * LOOK FOR NEXT LEADING S
00010878  6100 0C2C        814.  SXLOOP  BSR INCH
0001087C  0200 007F        815.          ANDI.B #ASCMSK,D0
00010880  0C00 0053        816.          CMPI.B #'S',D0   ;IS IT 'S'?
00010884  66 F2            817.          BNE.S SXLOOP
                           818.  *FIND NEXT DIGIT AND DECIDE APPROPRIATE
                           819.  *ACTION
```

```
00010886  6100 0C1E        820.          BSR INCH
0001088A  0200 007F        821.          ANDI.B #ASCMSK,D0
0001088E  6100 0C56        822.          BSR ASCBIN       ;CONVERT TO BINARY
00010892  0C00 0003        823.          CMPI.B #3,D0     ;IS IT > 3?
00010896  6E 40            824.          BGT.S ENDREC     ;ENDRECORD IF SO
00010898  4283             825.          CLR.L D3         ;PREPARE ADDR COUNT
0001089A  0400 0001        826.          SUBI.B #1,D0
0001089E  1600             827           MOVE.B D0,D3     ;ADDR COUNT ESTAB.
000108A0  6100 0114        828.  GETDATA BSR GETCUNT     ;GET BYTE COUNTER
                           829.  *READ IN BYTES AND FORM LOAD ADDRESS
000108A4  4284             830.          CLR.L D4
000108A6  6100 00DA        831.  LOADDR  BSR GRABNSM      ;GET DATA,KEEP SUM
000108AA  8807             832.          OR.B D7,D4       ;D4 BUILDS LOAD ADDR
000108AC  E184             833.          ASL.L #8,D4      ;POSITION SGNFCNCE
000108AE  0481 00000001    834.          SUBI.L #1,D1     ;ADJUST BYTE COUNTER
000108B4  51CB FFF0        835.          DBRA D3,LOADDR   ;DO TILL NO ADDRESS
000108B8  6100 00C8        836.          BSR GRABNSM      ;DO SHIFT CORRECT
000108BC  8807             837.          OR.B D7,D4       ;NO. OF TIMES
000108BE  0481 00000001    838.          SUBI.L #1,D1
000108C4  2444             839.          MOVEA.L D4,A2    ;A2 HAS LOAD ADDRESS
                           840.  *NOW SIMPLY READ DATA AND STORE AT (A1+A2)
000108C6  D5C9             841.          ADDA.L A1,A2
000108C8  6100 00B8        842.  LOADDAT BSR GRABNSM
000108CC  14C7             843.          MOVE.B D7,(A2)+  ;STORE OFF BYTE
000108CE  51C9 FFF8        844.          DBRA D1,LOADDAT  ;DO TILL DONE
000108D2  6100 00D0        845.          BSR CHKSUM       ;VERIFY CHECKSUM
000108D6  60 A0            846.          BRA.S SXLOOP     ;GET NEXT RECORD
                           847.  *END RECORD HANDLER, JUST READ AND DO ERROR
                           848.  *CHECK
000108D8  0C00 0007        849.  ENDREC  CMPI.B #7,D0     ;IS IT < 7?
000108DC  6D 22            850.          BLT.S NOCOD      ;FATAL ERROR IF SO
000108DE  6100 00D6        851.          BSR GETCUNT
000108E2  6100 009E        852.  ENDDAT  BSR GRABNSM      ;JUST READ QUICKLY
000108E6  51C9 FFFA        853.          DBRA D1,ENDDAT
000108EA  6100 00B8        854.          BSR CHKSUM
000108EE  4E75             855.  LOADRET RTS              ;END OF LOADER
                           856.  *FATAL ERROR ROUTINES
000108F0  41FA 0022        857.  NOHDR   LEA NOHDMES(PC),A0 ;NO HEADER ERROR
000108F4  6100 0B70        858.          BSR OUTMESC
000108F8  6100 0B48        859.          BSR SENCLFC
000108FC  7401             860.          MOVEQ #1,D2      ;1 ERROR
000108FE  60 EE            861.          BRA.S LOADRET    ;QUIT LOAD
00010900  41FA 0047        862.  NOCOD   LEA NOCODMS(PC),A0 ;
00010904  6100 0B60        863.          BSR OUTMESC
00010908  6100 0B38        864.          BSR SENCLFC
0001090C  0682 00000001    865.          ADDI.L #1,D2     ;ADJUST ERROR COUNT
00010912  60 DA            866.          BRA.S LOADRET
                           867.  *
00010914  46 41 54 41 4C   868.  NOHDMES DC.B 'FATAL ERROR: S0 Header Record'
          20 45 52 52 4F
          52 3A 20 53 30
          20 48 65 61 64
          65 72 20 52 65
          63 6F 72 64
00010931  20 4D 69 73 73   869.          DC.B ' Missing. Aborting Load',EOT
          69 6E 67 2E 20
          41 62 6F 72 74
          69 6E 67 20 4C
          6F 61 64 04
00010949  46 41 54 41 4C   870.  NOCODMS DC.B 'FATAL ERROR: Invalid s Record'
          20 45 52 52 4F
          52 3A 20 49 6E
          76 61 6C 69 64
          20 73 20 52 65
          63 6F 72 64
00010966  20 45 6E 63 6F   871.          DC.B ' Encountered. Aborting Load'
          75 6E 74 65 72
          65 64 2E 20 41
          62 6F 72 74 69
          6E 67 20 4C 6F
          61 64
00010981  04               872.          DC.B EOT
                           873.  *
                           874.  * SUBROUTINE GRABNSM
                           875.  * READ 2 ASCII CHRS AND CONVERTS TO BINARY
                           876.  * RETURNS BYTE IN D7, MAINTAINS SUM IN D6
                           877.  *
00010982  6100 0B22        878.  GRABNSM BSR INCH           ;GET DATA
00010986  0200 007F        879.          ANDI.B #ASCMSK,D0
0001098A  6100 0B5A        880.          BSR ASCBIN
```

```
0001098E  E900            881.          ASL.B #4,D0       ;POSITION SIGNF.
00010990  1E00            882.          MOVE.B D0,D7
00010992  6100 0B12       883.          BSR INCH
00010996  0200 007F       884.          ANDI.B #ASCMSK,D0
0001099A  6100 0B4A       885.          BSR ASCBIN
0001099E  8E00            886.          OR.B D0,D7        ;FORM BYTE IN D7
000109A0  DC07            887.          ADD.B D7,D6       ;KEEP CHKSUM
000109A2  4E75            888.          RTS
                          889.  *
                          890.  * SUBROUTINE CHKSUM
                          891.  * READS THE LAST BYTE OF RECORD,UPDATES
                          892.  * COUNTER,CHECKS VALIDITY OF CHECKSUM BYTE
                          893.  * AND INCREMENTS ERROR ACCUMULATOR ON ERRORS
                          894.  * CLEARS THE CHKSUM ACCUMULATOR
000109A4  61 DC           895.  CHKSUM  BSR GRABNSM
000109A6  0C06 00FF       896.          CMPI.B #$FF,D6  ;CHKSUM CORRECT?
000109AA  67 06           897.          BEQ.S SUMOK     ;DONT INC ERR IF SO
000109AC  0682 00000001   898.          ADDI.L #1,D2
000109B2  4286            899.  SUMOK   CLR.L D6        ;CLEAR FOR NEXT REC
000109B4  4E75            900.          RTS
                          901.  *
                          902.  * SUBROUTINE GETCUNT
                          903.  * SIMPLY READS THE BYTE COUNT DATA AND
                          904.  * ADJUSTS THE COUNTING REGISTER FOR DBRA
                          905.  * BYTECOUNT DATA INCLUDE BYTECOUNT,CHKSUM
                          906.  * AND ALL DATA.  # DATA TO BE STORED IS
                          907.  * BYTECOUNT-2, AND THEN NEED TO DECREMENT
                          908.  * BY ONE MORE FOR DBRA
000109B6  4281            909.  GETCUNT CLR.L D1
000109B8  61 C8           910.          BSR GRABNSM   ;GET COUNT,MANTAN SUM
000109BA  1207            911.          MOVE.B D7,D1  ;TRANSFER COUNT
000109BC  0401 0002       912.          SUBI.B #2,D1  ;ADJUST COUNTER
000109C0  4E75            913.          RTS
                          914.  *
                          915.  *
                          916.  * BREAKPOINT INSERT/REMOVE COMMAND
                          917.  * CALLED FROM COMMAND LEVEL. INSERTS THE
                          918.  * ILLEGAL OPCODE AT SPEC'D ADDRESS, PUTS
                          919.  * REPLACED INSTRUCTION ON BRKSTK ALONG
                          920.  * WITH THE REPLACMENT ADDRESS. REMOVING
                          921.  * BKPOINT REVERSES PROCEDURE
                          922.  *
000109C2  6100 0BAE       923.  BRKHAN  BSR BLDNUM       ;GET FURTHER INPUT
000109C6  B0FC 000D       924.          CMPA.W #CR,A0   ;ADDRESS CR ?
000109CA  67 16           925.          BEQ.S BRKEST    ;ESTABLISH BKPNT
000109CC  B1FC 0000FF2D   926.          CMPA.L #ZFFDSH,A0 ;MINUS SIGN ?
000109D2  67 56           927.          BEQ.S BRKREM    ;REMOVE BKPNT
000109D4  6100 0A6C       928.  BADINP  BSR SENCLFC
000109D8  41FA F9E4       929.          LEA BADDEL(PC),A0 ;BAD INPUT
000109DC  6100 0A88       930.          BSR OUTMESC
000109E0  4E75            931.          RTS
000109E2  2240            932.  BRKEST  MOVEA.L D0,A1   ;A1 HAS BRK ADDRESS
000109E4  3011            933.          MOVE.W (A1),D0  ;CHK IF ILLEGAL
000109E6  0C40 4AFC       934.          CMPI.W #ZILLEG,D0
000109EA  67 30           935.          BEQ.S RDYSET
000109EC  41FA F5B2       936.          LEA BRKNUM(PC),A0 ;A0 PNTS BRKNUM
000109F0  4280            937.          CLR.L D0
000109F2  1010            938.          MOVE.B (A0),D0  ;GET BRKNUM
000109F4  0C00 0006       939.          CMPI.B #6,D0    ;6 BRKS SET?
000109F8  6C 14           940.          BGE.S ALLBRK    ;ERROR IF SO
000109FA  C0FC 0006       941.          MULU #6,D0      ;ADJUST OFFSET
000109FE  2189 0002       942.          MOVE.L A1,2(A0,D0.W) ; STORE ADDRES
00010A02  3191 0006       943.          MOVE.W (A1),6(A0,D0.W) ;KEEP OLD INS
00010A06  32BC 4AFC       944.          MOVE.W #ZILLEG,(A1) ;RPLC WITH ILLEG
00010A0A  5210            945.          ADDQ.B #1,(A0)  ;INCREMENT BRKNUM
00010A0C  4E75            946.          RTS
00010A0E  6100 0A32       947.  ALLBRK  BSR SENCLFC
00010A12  41FA 00F3       948.          LEA ALBKMES(PC),A0 ;ERROR MESSAGE
00010A16  6100 0A4E       949.          BSR OUTMESC     ;6 BRKS SET
00010A1A  4E75            950.          RTS
00010A1C  6100 0A24       951.  RDYSET  BSR SENCLFC
00010A20  41FA 010F       952.          LEA RDYMES(PC),A0
00010A24  6100 0A40       953.          BSR OUTMESC
00010A28  4E75            954.          RTS
                          955.  *
00010A2A  41FA F574       956.  BRKREM  LEA BRKNUM(PC),A0  ;GET #BKPTS
00010A2E  0C10 0000       957.          CMPI.B #0,(A0)    ;NONE SET ?
00010A32  67 5C           958.          BEQ.S NONSET
00010A34  6100 0B3C       959.          BSR BLDNUM      ;GET NEXT INPUT
00010A38  B0FC 000D       960.          CMPA.W #CR,A0   ;ADDRESS SPEC'D
```

```
00010A3C  66 60              961.          BNE.S KILBRKS    ;NO KILL ALL BKS
00010A3E  2240               962.          MOVEA.L D0,A1    ;A1 HAS ADD TO KILL
00010A40  41FA F55E          963.          LEA BRKNUM(PC),A0
00010A44  2448               964.          MOVEA.L A0,A2    ;COPY
00010A46  4280               965.          CLR.L D0
00010A48  1010               966.          MOVE.B (A0),D0
00010A4A  D1FC 00000002      967.          ADDA.L #2,A0     ;ADJUST POINTER
00010A50  B3D0               968.  LKMOR   CMPA.L (A0),A1 ;SEARCH ADDR
00010A52  67 1A              969.          BEQ.S BRKFND
00010A54  5300               970.          SUBQ.B #1,D0
00010A56  67 08              971.          BEQ.S NOTBRK
00010A58  D1FC 00000006      972.          ADDA.L #6,A0
00010A5E  60 F0              973.          BRA.S LKMOR
00010A60  6100 09E0          974.  NOTBRK  BSR SENCLFC
00010A64  41FA 006A          975.          LEA NOTEXMS(PC),A0 ;BK NOT EXIST
00010A68  6100 09FC          976.          BSR OUTMESC
00010A6C  4E75               977.          RTS
                             978.  *BREAKPNT ADR FOUND,A0 POINTS, D0 HAS #LEFT
                             979.  * +1
00010A6E  32A8 0004          980.  BRKFND  MOVE.W 4(A0),(A1) ;REPLACE INST
00010A72  5300               981.          SUBQ.B #1,D0   ;LAST BREAKPOINT?
00010A74  67 16              982.          BEQ.S NOSHFT
00010A76  5380               983.          SUBQ.L #1,D0   ;SET FOR DBRA
00010A78  20A8 0006          984.  FIXUP   MOVE.L 6(A0),(A0) ;ADCMDST
00010A7C  3168 000A 0004     985.          MOVE.W 10(A0),4(A0)
00010A82  D1FC 00000006      986.          ADDA.L #6,A0
00010A88  51C8 FFEE          987.          DBRA D0,FIXUP
00010A8C  5312               988.  NOSHFT  SUBQ.B #1,(A2)  ;DECREMENT BRKNUM
00010A8E  4E75               989.          RTS
00010A90  6100 09B0          990.  NONSET  BSR SENCLFC
00010A94  41FA 005E          991.          LEA NONEXMS(PC),A0
00010A98  6100 09CC          992.          BSR OUTMESC
00010A9C  4E75               993.          RTS
00010A9E  B1FC 0000FF0D      994.  KILBRKS CMPA.L #ZFFCR,A0 ;KILL ALL?
00010AA4  6600 FF2E          995.          BNE BADINP
00010AA8  41FA F4F6          996.          LEA BRKNUM(PC),A0
00010AAC  2448               997.          MOVEA.L A0,A2    ;SAVE A COPY
00010AAE  4280               998.          CLR.L D0
00010AB0  1010               999.          MOVE.B (A0),D0   ;HOW MANY BRKS?
00010AB2  67 DC             1000.          BEQ.S NONSET     ;JUST IN CASE
00010AB4  5300              1001.          SUBQ.B #1,D0     ;ADJUST FOR DBRA
00010AB6  D1FC 00000002     1002.          ADDA.L #2,A0     ;ADJUST POINTER
00010ABC  2250              1003.  MORFIX  MOVEA.L (A0),A1 ;ADDRESS TO FIX
00010ABE  32A8 0004         1004.          MOVE.W 4(A0),(A1) ;RESTORE INST
00010AC2  D1FC 00000006     1005.          ADDA.L #6,A0     ;ADJUST
00010AC8  5312              1006.          SUBQ.B #1,(A2) ;DECREMENT BRKMUM
00010ACA  51C8 FFF0         1007.          DBRA D0,MORFIX
00010ACE  4E75              1008.          RTS
00010AD0  53 70 65 63 69    1009.  NOTEXMS DC.B 'Specified Breakpoint does'
          66 69 65 64 20
          42 72 65 61 6B
          70 6F 69 6E 74
          20 64 6F 65 73
00010AE9  20 6E 6F 74 20    1010.          DC.B ' not exist',EOT
          65 78 69 73 74
          04
00010AF4  4E 6F 20 42 72    1011.  NONEXMS DC.B 'No Breakpoints Set',EOT
          65 61 6B 70 6F
          69 6E 74 73 20
          53 65 74 04
00010B07  4D 61 78 69 6D    1012.  ALBKMES DC.B 'Maximum Number of Breakpoint'
          75 6D 20 4E 75
          6D 62 65 72 20
          6F 66 20 42 72
          65 61 6B 70 6F
          69 6E 74
00010B23  73 20 41 6C 72    1013.          DC.B 's Already Set',EOT
          65 61 64 79 20
          53 65 74 04
00010B31  42 72 65 61 6B    1014.  RDYMES  DC.B 'Breakpoint is already set or'
          70 6F 69 6E 74
          20 69 73 20 61
          6C 72 65 61 64
          79 20 73 65 74
          20 6F 72
00010B4D  20 69 73 20 49    1015.          DC.B ' is ILLEGAL instruction.'
          4C 4C 45 47 41
          4C 20 69 6E 73
          74 72 75 63 74
          69 6F 6E 2E
```

```
00010B65  04                1016.          DC.B EOT
                            1017.  *
                            1018.  *
                            1019.  * CALL A PROGRAM TO BE EXECUTED AS A
                            1020.  * SUPERVISOR STATE SUBROUTINE OF THE MONITOR
                            1021.  * PROGRAM MUST END IN RTS. COMMAND LEVEL
                            1022.  * RE-ENTERED. TRACE OPTION AVAILABLE
                            1023.  *
00010B66  6100 0A0A         1024.  CALSPRG BSR BLDNUM       ;GET FURTHER INPUT
00010B6A  B0FC 000D         1025.          CMPA.W #CR,A0    ;CR GOES
00010B6E  66 12             1026.          BNE.S TRACEN
00010B70  46FC 2000         1027.          MOVE.W #$2000,SR ;SUP MODE,NO TRACE
00010B74  2040              1028.          MOVEA.L D0,A0    ;JUMP ADDRES
00010B76  4E90              1029.          JSR (A0)
00010B78  007C 2000         1030.          ORI.W #$2000,SR ;ENSURE SUP STATE
00010B7C  027C 2FFF         1031.          ANDI.W #$2FFF,SR ;DISABLE TRACING
00010B80  4E75              1032.          RTS
00010B82  2F00              1033.  TRACEN  MOVE.L D0,-(SP)  ;SAVE JUMP ADDPESS
00010B84  B0FC 0020         1034.          CMPA.W #SPACE,A0 ;IS IT SPACE?
00010B88  66 44             1035.          BNE.S BADCAL
00010B8A  6100 08FE         1036.          BSR INCHCON      ;GET NEXT CHR
00010B8E  6100 089A         1037.          BSR SENDBYC      ;ECHO
00010B92  0200 007F         1038.          ANDI.B #ASCMSK,D0 ;
00010B96  0C00 0054         1039.          CMPI.B #'T',D0   ;T SPEC'D ?
00010B9A  66 32             1040.          BNE.S BADCAL
00010B9C  6100 08EC         1041.          BSR INCHCON
00010BA0  6100 0888         1042.          BSR SENDBYC
00010BA4  0200 007F         1043.          ANDI.B #ASCMSK,D0
00010BA8  0C00 000D         1044.          CMPI.B #CR,D0    ;CR WILL GO
00010BAC  66 20             1045.          BNE.S BADCAL
00010BAE  700A              1046.          MOVEQ #LF,D0   ;QUICK LF
00010BB0  6100 0878         1047.          BSR SENDBYC
00010BB4  41FA F3EA         1048.          LEA BRKNUM(PC),A0 ;SEE IF BKPNTS
00010BB8  0C10 0000         1049.          CMPI.B #0,(A0)
00010BBC  6600 0076         1050.          BNE SETBRK
00010BC0  205F              1051.          MOVE.L (SP)+,A0  ;GET JUMP ADDRESS
00010BC2  46FC A000         1052.          MOVE.W #$A000,SR ;SUP, TRACE ON
00010BC6  4E90              1053.          JSR (A0)         ;DO ROUTINE
00010BC8  46FC 2000         1054.          MOVE.W #$2000,SR ;TRACE OFF
00010BCC  4E75              1055.          RTS
00010BCE  6100 0872         1056.  BADCAL  BSR SENCLFC
00010BD2  41FA F7EA         1057.          LEA BADDEL(PC),A0
00010BD6  6100 088E         1058.          BSR OUTMESC
00010BDA  201F              1059.          MOVE.L (SP)+,D0  ;RESTORE STACK
00010BDC  4E75              1060.          RTS
                            1061.  *
                            1062.  *
                            1063.  * CALL A PROGRAM TO BE EXECUTED IN THE USER
                            1064.  * STATE. MUST END IN TRAP #0,TO RESUME MONITOR
                            1065.  * LEVEL EXECUTION. ELSE TROUBLE!!!
                            1066.  *
00010BDE  6100 0992         1067.  GOUPRG  BSR BLDNUM       ;GET FURTHER INPUT
00010BE2  B0FC 000D         1068.          CMPA.W #CR,A0    ;CR GOES
00010BE6  66 08             1069.          BNE.S TRACON
00010BE8  2040              1070.          MOVEA.L D0,A0    ;JUMP ADDRESS
00010BEA  46FC 0000         1071.          MOVE.W #0000,SR ;USER STATE
00010BEE  4ED0              1072.          JMP (A0)         ;USER PROGRAM
00010BF0  2F00              1073.  TRACON  MOVE.L D0,-(SP) ;SAVE JUMP ADDRESS
00010BF2  B0FC 0020         1074.          CMPA.W #SPACE,A0 ;SPACE?
00010BF6  66 D6             1075.          BNE.S BADCAL
00010BF8  6100 0890         1076.          BSR INCHCON      ;MORE INPUT
00010BFC  6100 082C         1077.          BSR SENDBYC
00010C00  0200 007F         1078.          ANDI.B #ASCMSK,D0
00010C04  0C00 0054         1079.          CMPI.B #'T',D0   ;T SPEC'D?
00010C08  66 C4             1080.          BNE S BADCAL
00010C0A  6100 087E         1081.          BS' INCHCON      ;NOW CR NECES.
00010C0E  6100 081A         1082.          B. . SENDBYC
00010C12  0200 007F         1083.          ANDI.B #ASCMSK,D0
00010C16  0C00 000D         1084.          CMPI.B #CR,D0    ;CR WILL GO
00010C1A  66 B2             1085.          BNE.S BADCAL
00010C1C  700A              1086.          MOVEQ #LF,D0
00010C1E  6100 080A         1087.          BSR SENDBYC
00010C22  41FA F37C         1088.          LEA BRKNUM(PC),A0 ;BKPNTS ACTIVE?
00010C26  0C10 0000         1089.          CMPI.B #0,(A0)   ;ERROR IF SO
00010C2A  66 0.             1090.          BNE.S SETBRK
00010C2C  205F              1091.          MOVE.L (SP)+,A0 ;GET JUMP ADDRESS
00010C2E  46FC 8000         1092.          MOVE.W #$8000,SR ; TRACE ON USER
00010C32  4ED0              1093.          JMP (A0)         ;DO USER ROUTINE
                            1094.  *
00010C34  6100 080C         1095.  SETBRK  BSR SENCLFC
```

```
00010C38  41FA 000A       1096.           LEA SETBKMS(PC),A0
00010C3C  6100 0828       1097.           BSR OUTMESC
00010C40  201F            1098.           MOVE.L (SP)+,D0  ;CLEAN UP
00010C42  4E75            1099.           RTS
00010C44  42 72 65 61 6B  1100.  SETBKMS DC.B 'Breakpoints are active. Trace'
          70 6F 69 6E 74
          73 20 61 72 65
          20 61 63 74 69
          76 65 2E 20 54
          72 61 63 65
00010C61  20 6E 6F 74 20  1101.           DC.B ' not available.',CR,LF,'Remove'
          61 76 61 69 6C
          61 62 6C 65 2E
          0D 0A 52 65 6D
          6F 76 65
00010C78  20 41 4C 4C 20  1102.           DC.B ' ALL Breakpoints to use Trace'
          42 72 65 61 6B
          70 6F 69 6E 74
          73 20 74 6F 20
          75 73 65 20 54
          72 61 63 65
00010C95  04              1103.           DC.B EOT
                          1104.  *
                          1105.  * ERROR HANDLING AND SYS SUPPORT
                          1106.  ********************************************
                          1107.  * THIS SECTION OF CODE CONTAINS ROUTINES FOR *
                          1108.  * ERROR HANDLING SUCH AS BUS ERROR, ADDRESS  *
                          1109.  * ERROR, INTERRUPTS, AND ALL THOSE OTHER     *
                          1110.  * THINGS THAT COULD POSSIBLY GO WRONG.IN MOST*
                          1111.  * CASES, THE USER IS SIMPLY ADVISED OF THE   *
                          1112.  * TYPE OF ERROR THAT OCCURRED, AND THE       *
                          1113.  * REGISTERS ARE PRESENTED TO THE CONSOLE.    *
                          1114.  * BUS AND ADDRESS ERROR, SINCE THEY HAVE A   *
                          1115.  * SPECIAL FORM OF EXCEPTION STACK FRAME, HAVE*
                          1116.  * A SPECIAL ROUTINE TO INDICATE THE CONTENTS *
                          1117.  * OF THE FRAME                               *
                          1118.  ********************************************
                          1119.  * 4-16-88 TRACE HANDLING AND BREAKPOINT
                          1120.  * HANDLING ADDED
                          1121.  *
                          1122.  * updated march 14,1991
                          1123.  * to handle new trap routines for
                          1124.  * multiprocessor system. parallel
                          1125.  * interface control of data transfers
                          1126.  * and service of SRQ's
                          1127.  *
                          1128.  *
                          1129.  * BUS ERROR AND ADDRESS ERROR HANDLING
                          1130.  * SUBROUTINES
                          1131.  * PRINTS AN ADVISORY MESSAGE,DISPLAYS
                          1132.  * CONTENTS OF THE GROUP 0 EXCEPTION STACK
                          1133.  * FRAME, AND THE SYSTEM IS RESTARTED
                          1134.  * CALLS TO EXCEPT0,OUTMESC
                          1135.  *
00010C96  41FA 0035       1136.  ADDERR  LEA ADERMES(PC),A0 ;POINT TO ADD.ERROR
00010C9A  6000 0006       1137.          BRA ERRWRIT       ;MESSAGE & SEND IT
00010C9E  41FA 001A       1138.  BUSERR  LEA BUERMES(PC),A0 ;POINT TO BUS ERROR
00010CA2  6100 07C2       1139.  ERRWRIT BSR OUTMESC       ;MESSAGE & SEND IT
00010CA6  6100 079A       1140.          BSR SENCLFC
00010CAA  6100 004A       1141.          BSR EXCEPT0       ;SEND STACK FRAME
00010CAE  41FA 0034       1142.          LEA STATMES(PC),A0 ;POINT TO RESTART
00010CB2  6100 07B2       1143.          BSR OUTMESC       ;MESSAGE & SEND IT
00010CB6  6000 F378       1144.          BRA RSTRT         ;QUIT
                          1145.  *
00010CBA  42 75 73 20 45  1146.  BUERMES DC.B 'Bus Error Occurred',EOT
          72 72 6F 72 20
          4F 63 63 75 72
          72 65 64 04
00010CCD  41 64 64 72 65  1147.  ADERMES DC.B 'Address Error Occurred',EOT
          73 73 20 45 72
          72 6F 72 20 4F
          63 63 75 72 72
          65 64 04
00010CE4  52 65 73 74 61  1148.  STATMES DC.B 'Restarting System',EOT
          72 74 69 6E 67
          20 53 79 73 74
          65 6D 04
                          1149.  *
                          1150.  *
                          1151.  *EXCEPTION 0 ERROR HANDLING ROUTINE
```

```
                                1152.  * CALLED BY GROUP 0 EXCEPTIONS
                                1153.  * OUTPUTS CONTENTS OF EXCEPTION STACK FRAME
                                1154.  * CALLS TO SENDLOC,SENDWOC,SENDBYC,PUT8HX,
                                1155.  * PUT4HX,PUT2HX,SENCLFC,
                                1156.  *
                                1157.  * TRASHES SEVERAL REGISTERS
                                1158.  *
00010CF6  7403                  1159.  EXCEPT0 MOVEQ #3,D2       ;INIT A LOOP COUNTER
00010CF8  4DFA 008C             1160.          LEA REGHD(PC),A6 ;POINT TO TTLS
00010CFC  4281                  1161.          CLR.L D1         ;ZERO
00010CFE  2016                  1162.  EXCT0   MOVE.L (A6),D0    ;GET TTL STRING
00010D00  6100 06F8             1163.          BSR SENDLOC      ;SEND IT OUT
00010D04  122E 0005             1164.          MOVE.B 5(A6),D1  ;GET OFFSET SPEC
00010D08  082E 0000 0004        1165.          BTST.B #0,4(A6)  ;TEST FOR LONG OR WD
00010D0E  67 18                 1166.          BEQ.S EXCT0WR    ;WORD LENGTH DATA
00010D10  2037 1000             1167.          MOVE.L 0(SP,D1.W),D0 ;GET VAL FROM STK
00010D14  6100 0810             1168.          BSR DISBUF8      ;CONVERT,STORE,SEND
00010D18  6100 0728             1169.          BSR SENCLFC      ;MAKE IT LOOK PRETTY
00010D1C  DDFC 00000006         1170.  EXCT0VR ADDA.L #6,A6     ;ADJUST POINTER
0001CD22  51CA FFDA             1171.          DBRA D2,EXCT0   ;CONTINUE 'ILL DONE
00010D26  60 0E                 1172.          BRA.S EXCT0CD    ;BEGIN CODE WORD OUT
                                1173.  *ENTRY BEFORE CODEWORD IS LONG
00010D28  3037 1000             1174.  EXCT0WR MOVE.W 0(SP,D1.W),D0 ;GET WORD DATA
00010D2C  6100 0810             1175.          BSR DISBUF4      ;CONVERT,STORE SEND
00010D30  6100 0710             1176.          BSR SENCLFC
00010D34  60 E6                 1177.          BRA.S EXCT0VR    ;CONTINUE AS NORMAL
                                1178.  *
                                1179.  * SEND THE CODE WORD
                                1180.  *
00010D3e  2016                  1181.  EXCT0CD MOVE.L (A6),D0    ;GET 1ST TTL FOR CODE
00010D38  6100 06C0             1182.          BSR SENDLOC      ;
00010D3C  4280                  1183.          CLR.L D0
00010D3E  322F 0004             1184.          MOVE.W 4(SP),D1  ;GET THE CODE WORD
00010D42  0281 0000001F         1185.          ANDI.L #$1F,D1   ;MASK MISC SET BITS
00010D48  E859                  1186.          ROR.W #4,D1      ;POSITION R/W  BIT
00010D4A  1001                  1187.          MOVE.B D1,D0     ;COPY TO D0
00010D4C  6100 080A             1188.          BSR DISBUF2      ;CONVERT STORE & SEND
00010D50  6100 06F0             1189.          BSR SENCLFC      ;CR LF
00010D54  202E 0006             1190.          MOVE.L 6(A6),D0  ;GET THE NEXT TTL
00010D58  6100 06A0             1191.          BSR SENDLOC      ;SEND TO CONSOLE
00010D5C  0881 0000             1192.          BCLR #0,D1       ;CLEAR THE R/W BIT
00010D60  E359                  1193.          ROL.W #1,D1      ;POSITION THE IN BIT
00010D62  1001                  1194.          MOVE.B D1,D0     ;COPY TO D0
00010D64  6100 07F2             1195.          BSR DISBUF2      ;SEND TO CONSOLE
00010D68  6100 06D8             1196.          BSR SENCLFC
00010D6C  202E 000C             1197.          MOVE.L 12(A6),D0 ;GET NEXT TTL
00010D70  6100 0688             1198.          BSR SENDLOC
00010D74  0881 0000             1199.          BCLR #0,D1      ;CLEAR THE IN BIT
00010D78  E759                  1200.          ROL.W #3,D1      ;POSITION FC BITS
00010D7A  1001                  1201.          MOVE.B D1,D0     ;COPY CONTENTS
00010D7C  6100 07DA             1202.          BSR DISBUF2      ;SEND TO THE CONSOLE
00010D80  6100 06C0             1203.          BSR SENCLFC
00010D84  4E75                  1204.          RTS
                                1205.  *
                                1206.  * DATA SECTION FOR EXCEPT0
                                1207.  *
00010D86  50 43 2D 20 01 0E     1208.  REGHD   DC.B 'PC- ',1,14
00010D8C  53 52 2D 20 00 0C     1209.          DC.B 'SR- ',0,12
00010D92  49 52 2D 20 00 0A     1210.          DC.B 'IR- ',0,10
00010D98  41 41 2D 20 01 06     1211.          DC.B 'AA- ',1,6
00010D9E  52 57 2D 20 00 04     1212.          DC.B 'RW- ',0,4
00010DA4  49 4E 2D 20 00 04     1213.          DC.B 'IN- ',0,4
00010DAA  46 43 2D 20 00 04     1214.          DC.B 'FC- ',0,4
                                1215.
                                1216.  *
                                1217.  * OTHER ERRORS SUBROUTINE
                                1218.  * USED TO HANDLE THE FOLLOWING ERROR COND:
                                1219.  *      ILLEGAL INSTRUCTION
                                1220.  *      TRACE EXCEPTION
                                1221.  *      ZERO DIVIDE
                                1222.  *      CHK INSTRUCTION TRAP
                                1223.  *      TRAPV INSTRUCTION EXCEPTION
                                1224.  *      PRIVILEGE VIOLATION
```

```
                        1225.  *      SPURIOUS INTERRUPT
                        1226.  *      UNINITIALIZED INTERRUPT
                        1227.  * IN EACH CASE, ONLY AN ERROR MESSAGE IS
                        1228.  * GIVEN TO ADVISE OF THE TYPE OF ERROR THAT
                        1229.  * OCCURRED, AND THE REGISTER CONTENTS
                        1230.  * PREVIOUS TO THE EXCEPTION ARE DISPLAYED
                        1231.  * CALLS STOREG,OUTMESC,OUTREG,RESTART
                        1232.  ********************************************
                        1233.  * 4-16-88 CHECKS BRKNUM TO SEE IF ILLEGAL
                        1234.  * INSTRUCTION EXISTS FOR BREAKPOINT PURPOSES
                        1235.  * IF NOT, TREAT AS ILLEGAL
                        1236.  *
00010DB0  48E7 8080     1237.  ILLEG   MOVEM.L D0/A0,-(SP)
00010DB4  41FA F1EA     1238.          LEA BRKNUM(PC),A0   ;CHEK BRKPTS SET
00010DB8  0C10 0000     1239.          CMPI.B #0,(A0)
00010DBC  66 18         1240.          BNE.S BREKPT        ;DO BREAK SERVICE
00010DBE  41FA F1AE     1241.          LEA STRGTMP(PC),A0 ;POINT
00010DC2  10BC 0001     1242.          MOVE.B #1,(A0)     ;SET FLAG
00010DC6  4CDF 0101     1243.          MOVEM.L (SP)+,D0/A0
00010DCA  6100 0312     1244.          BSR STOREG         ;SAVE THE REGISTERS
00010DCE  41FA 0194     1245.          LEA ILLMES(PC),A0 ;POINT TO ILL. MESS
00010DD2  6000 017C     1246.          BRA COMERR
00010DD6  4280          1247.  BREKPT  CLR.L D0
00010DD8  1010          1248.          MOVE.B (A0),D0     ;GET #BKPTS
00010DDA  D1FC 00000002 1249.          ADDA.L #2,A0       ;POINT TO BRKSTK
                        1250.  * RESTORE THE ORIGINAL INSTRUCTIONS FOR EXE.
00010DE0  6100 0014     1251.          BSR SWAP           ;DO BKPT SWAP
00010DE4  41FA F1BB     1252.          LEA BRKFLG(PC),A0 ;POINT TO BRKFLG
00010DE8  10BC 0001     1253.          MOVE.B #1,(A0)     ;SET THE FLAG
00010DEC  4CDF 0101     1254.          MOVEM.L (SP)+,D0/A0 ;RESTORE
00010DF0  0057 8000     1255.          ORI.W #$8000,(SP) ;MODIFY SR TO TRACE
00010DF4  4E73          1256.          RTE
                        1257.  * SWAP SUBROUTINE
                        1258.  *D0 HAS #SWAPS,A0 POINTS TO BRKSTK
                        1259.  *
00010DF6  48E7 4040     1260.  SWAP     MOVEM.L D1/A1,-(SP)  ;WORK SPACE
00010DFA  0480 00000001 1261.          SUBI.L #1,D0          ;ADJUST COUNT
00010E00  2250          1262.  MRSWP   MOVEA.L (A0),A1       ;SWAP ADDRESS
00010E02  3228 0004     1263.          MOVE.W 4(A0),D1       ;DATA TO SWAP
00010E06  3151 0004     1264.          MOVE.W (A1),4(A0)     ;DO SWAP
00010E0A  3281          1265.          MOVE.W D1,(A1)        ;
00010E0C  D1FC 00000006 1266.          ADDA.L #6,A0          ;POINT TO NEXT
00010E12  51C8 FFEC     1267.          DBRA D0,MRSWP
00010E16  4CDF 0202     1268.          MOVEM.L (SP)+,D1/A1  ;RESTORE
00010E1A  4E75          1269.          RTS
                        1270.  *
                        1271.  *
                        1272.  * TRACE EXCEPTION HANDLER ROUTINE.
                        1273.  * EXECUTES ON TRACE EXCEPTIONS. DISPLAYS
                        1274.  * REGISTERS, WAITS FOR INPUT. IF BREAKPOINT
                        1275.  * SERVICE, RESTORES ILLEGAL INSTR. BKPNTS
                        1276.  *
00010E1C  2F08          1277.  HTRACE  MOVE.L A0,-(SP)
00010E1E  41FA F14E     1278.          LEA STRGTMP(PC),A0       ;INDCTE EXCP
00010E22  10BC 0001     1279.          MOVE.B #1,(A0)     ;CALLING STOREG
00010E26  205F          1280.          MOVE.L (SP)+,A0
00010E28  6100 02B4     1281.          BSR STOREG
00010E2C  6100 02FE     1282.          BSR OUTREG
00010E30  48E7 8080     1283.          MOVEM.L D0/A0,-(SP)      ;WORK SPACE
00010E34  4280          1284.          CLR.L D0
00010E36  41FA F169     1285.          LEA BRKFLG(PC),A0  ;BRKPNT SERVICE?
00010E3A  1010          1286.          MOVE.B (A0),D0
00010E3C  67 14         1287.          BEQ.S WHATNXT       ;NO JUST SHOW
00010E3E  4210          1288.  BRKFIX  CLR.B (A0)      ;CLEAR FLAG
00010E40  1028 FFFF     1289.          MOVE.B -1(A0),D0    ;GET BRKNUM
00010E44  D1FC 00000001 1290.          ADDA.L #1,A0        ;POINT BRKSTK
00010E4A  61 AA         1291.          BSR SWAP            ;EXCHANGE
00010E4C  026F 2FFF 0008 1292.          ANDI.W #$2FFF,8(SP) ;DSBL TRCE ON RET
00010E52  41FA 0026     1293.  WHATNXT LEA CONTMES(PC),A0 ;PROMPT
00010E56  6100 060E     1294.          BSR OUTMESC
00010E5A  6100 062E     1295.          BSR INCHCON         ;WAIT FOR RESPONSE
00010E5E  0200 007F     1296.          ANDI.B #ASCMSK,D0
00010E62  0C00 001B     1297.          CMPI.B #ESCCHR,D0  ;IS IT ESC?
00010E66  67 06         1298.          BEQ.S QUITEX
00010E68  4CDF 0101     1299.          MOVEM.L (SP)+,D0/A0 ;RESTORE
00010E6C  4E73          1300.          RTE
                        1301.  * DO THIS WHEN USER ESCAPES. RESTORE REGS
                        1302.  * AND REINITIALIZE SYSTEM STACK, BE SURE
                        1303.  * SUPERVISOR, GO TO MONITOR
00010E6E  4CDF 0101     1304.  QUITEX  MOVEM.L (SP)+,D0/A0
```

```
00010E72  46FC 2000       1305.          MOVE.W #$2000,SR
00010E76  6000 F2F2       1306.          BRA REINIT
                          1307.  *
00010E7A  54 79 70 65 20  1308.  CONTMES DC.B 'Type ESC to quit, any other'
          45 53 43 20 74
          6F 20 71 75 69
          74 2C 20 61 6E
          79 20 6F 74 68
          65 72
00010E95  20 6B 65 79 20  1309.          DC.B ' key to continue...',CR,LF,EOT
          74 6F 20 63 6F
          6E 74 69 6E 75
          65 2E 2E 2E 0D
          0A 04
                          1310.  *
          -- boundary align
00010EAC  2F08            1311.  ZERDIV  MOVE.L A0,-(SP)
00010EAE  41FA F0BE       1312.          LEA STRGTMP(PC),A0 ;POINT
00010EB2  10BC 0001       1313.          MOVE.B #1,(A0)
00010EB6  205F            1314.          MOVE.L (SP)+,A0
00010EB8  6100 0224       1315.          BSR STOREG         ;SAVE THE REGISTERS
00010EBC  41FA 00C6       1316.          LEA ZERMES(PC),A0 ;POINT TO MESS
00010EC0  6000 008E       1317.          BRA COMERR
00010EC4  2F08            1318.  CHKTRP  MOVE.L A0,-(SP)
00010EC6  41FA F0A6       1319.          LEA STRGTMP(PC),A0 ;POINT
00010ECA  10BC 0001       1320.          MOVE.B #1,(A0)
00010ECE  205F            1321.          MOVE.L (SP)+,A0    ;SET EXCEPTION FLAG
00010ED0  6100 020C       1322.          BSR STOREG         ;SAVE THE REGISTERS
00010ED4  41FA 00C6       1323.          LEA CHKMES(PC),A0 ;POINT TO MESS
00010ED8  6000 0076       1324.          BRA COMERR
00010EDC  2F08            1325.  TRPVTRP MOVE.L A0,-(SP)
00010EDE  41FA F08E       1326.          LEA STRGTMP(PC),A0 ;POINT
00010EE2  10BC 0001       1327.          MOVE.B #1,(A0)
00010EE6  205F            1328.          MOVE.L (SP)+,A0
00010EE8  6100 01F4       1329.          BSR STOREG         ;SAVE THE REGISTERS
00010EEC  41FA 00CA       1330.          LEA TRVMES(PC),A0 ;POINT TO MESS
00010EF0  6000 005E       1331.          BRA COMERR
00010EF4  2F08            1332.  PRIVTRP MOVE.L A0,-(SP)
00010EF6  41FA F076       1333.          LEA STRGTMP(PC),A0 ;POINT
00010EFA  10BC 0001       1334.          MOVE.B #1,(A0)
00010EFE  205F            1335.          MOVE.L (SP)+,A0
00010F00  6100 01DC       1336.          BSR STOREG         ;SAVE THE REGISTERS
00010F04  41FA 00D0       1337.          LEA PRVMES(PC),A0 ;POINT TO MESS
00010F08  6000 0046       1338.          BRA COMERR
00010F0C  2F08            1339.  SPURINT MOVE.L A0,-(SP)
00010F0E  41FA F05E       1340.          LEA STRGTMP(PC),A0 ;POINT
00010F12  10BC 0001       1341.          MOVE.B #1,(A0)
00010F16  205F            1342.          MOVE.L (SP)+,A0
00010F18  6100 01C4       1343.          BSR STOREG         ;SAVE THE REGISTERS
00010F1C  41FA 00D8       1344.          LEA SPRMES(PC),A0 ;POINT TO MESS
00010F20  6000 002E       1345.          BRA COMERR
00010F24  2F08            1346.  UNINIT  MOVE.L A0,-(SP)
00010F26  41FA F046       1347.          LEA STRGTMP(PC),A0 ;POINT
00010F2A  10BC 0001       1348.          MOVE.B #1,(A0)
00010F2E  205F            1349.          MOVE.L (SP)+,A0
00010F30  6100 01AC       1350.          BSR STOREG         ;SAVE THE REGISTERS
00010F34  41FA 00DF       1351.          LEA UNTMES(PC),A0 ;POINT TO MESS
00010F38  6000 0016       1352.          BRA COMERR
00010F3C  2F08            1353.  LEV7INT MOVE.L A0,-(SP)
00010F3E  41FA F02E       1354.          LEA STRGTMP(PC),A0 ;POINT
00010F42  10BC 0001       1355.          MOVE.B #1,(A0)
00010F46  205F            1356.          MOVE.L (SP)+,A0
00010F48  6100 0194       1357.          BSR STOREG         ;SAVE THE REGISTERS
00010F4C  41FA 00EB       1358.          LEA ABTMES(PC),A0 ;POINT TO MESS
00010F50  6100 0514       1359.  COMERR  BSR OUTMESC        ;SEND APP. MESSAGE
00010F54  6100 01D6       1360.          BSR OUTREG         ;SHOW REGISTERS
00010F58  41FA 00E7       1361.          LEA REINMES(PC),A0 ;PNT TO RESTART MES
00010F5C  6100 0508       1362.          BSR OUTMESC
00010F60  6000 F208       1363.          BRA REINIT          ;START OVER
                          1364.  *
                          1365.  * ERROR MESSAGES
                          1366.  *
00010F64  49 6C 6C 65 67  1367.  ILLMES  DC.B 'Illegal Instruction Exception'
          61 6C 20 49 6E
          73 74 72 75 63
          74 69 6F 6E 20
          45 78 63 65 70
          74 69 6F 6E
00010F81  0D 0A 04        1368.          DC.B CR,LF,EOT
00010F84  5A 65 72 6F 20  1369.  ZERMES  DC.B 'Zero Divide Exception'
```

```
                44 69 76 69 64
                65 20 45 78 63
                65 70 74 69 6F
                6E
00010F99        0D 0A 04              1370.             DC.B CR,LF,EOT
00010F9C        43 68 6B 20 49        1371.  CHKMES      DC.B 'Chk Instruction Exception'
                6E 73 74 72 75
                63 74 69 6F 6E
                20 45 78 63 65
                70 74 69 6F 6E
00010FB5        0D 0A 04              1372.             DC.B CR,LF,EOT
00010FB8        54 72 61 70 56        1373.  TRVMES      DC.B 'TrapV Instruction Exception'
                20 49 6E 73 74
                72 75 63 74 69
                6F 6E 20 45 78
                63 65 70 74 69
                6F 6E
00010FD3        0D 0A 04              1374.             DC.B CR,LF,EOT
00010FD6        50 72 69 76 69        1375.  PRVMES      DC.B 'Privilege Violation Exception'
                6C 65 67 65 20
                56 69 6F 6C 61
                74 69 6F 6E 20
                45 78 63 65 70
                74 69 6F 6E
00010FF3        0D 0A 04              1376.             DC.B CR,LF,EOT
00010FF6        53 70 75 72 69        1377.  SPRMES      DC.B 'Spurious Interrupt Exception'
                6F 75 73 20 49
                6E 74 65 72 72
                75 70 74 20 45
                78 63 65 70 74
                69 6F 6E
00011012        0D 0A 04              1378.             DC.B CR,LF,EOT
00011015        55 6E 69 6E 69        1379.  UNTMES      DC.B 'Uninitialized Interrupt Excepti'
                74 69 61 6C 69
                7A 65 64 20 49
                6E 74 65 72 72
                75 70 74 20 45
                78 63 65 70 74
                69
00011034        6F 6E 0D 0A 04        1380.             DC.B 'on',CR,LF,EOT
00011039        41 62 6F 72 74        1381.  ABTMES      DC.B 'Abort',CR,LF,EOT
                0D 0A 04
00011041        52 65 69 6E 69        1382.  REINMES     DC.B 'Reinitializing Supervisor Stack'
                74 69 61 6C 69
                7A 69 6E 67 20
                53 75 70 65 72
                76 69 73 6F 72
                20 53 74 61 63
                6B
00011060        20 50 6F 69 6E        1383.             DC.B ' Pointer',CR,LF,EOT
                74 65 72 0D 0A
                04
                                      1384.  *
                                      1385.  * LEVEL 2 INTERRUPT HANDLER
                                      1386.  * ACIA INTERRUPT FROM EITHER PORT 1 OR 2
                                      1387.  * TO BE USED WHEN RECEIVE INTRPTS ARE ENABLED
                                      1388.  * SIMPLY WRITES A MESSAGE TO CONSOLE, AND
                                      1389.  * ECHOES THE DATA
                                      1390.  * POLLED I/O WILL REALLY BE USED, BUT THIS
                                      1391.  * ROUTINE IS INCLUDED FOR COMPLETENESS
                                      1392.  *
                                      1393.  *CALLS OUTMESC SENDBYC
                                      1394.  *
                -- boundary align
0001106C        48E7 8082             1395.  INTLEV2    MOVEM.L D0/A0/A6,-(SP) ;WORK SPACE
00011070        41FA 003A             1396.             LEA IN2MES(PC),A0      ;PNT TO MESGE
00011074        6100 03F0             1397.             BSR OUTMESC
00011078        2C7C 00012FFC         1398.             MOVE.L #ACIA1,A6       ,PNT TO ACIA1
0001107E        0816 0007             1399.             BTST.B #7,(A6)         ;IRQ BIT SET?
00011082        67 12                 1400.             BEQ.S AC2INT           ,SERV ACIA2
00011084        41FA 0038             1401.             LEA AC1MES(PC),A0      ;PNT TO MESS
00011088        6100 03DC             1402.             BSR OUTMESC            ;DISPLAY NUMBER
0001108C        102E 0001             1403.             MOVE.B 1(A6),D0        ;GET THE DATA
00011090        6100 0398             1404.             BSR SENDBYC
00011094        60 10                 1405.             BRA.S DUNINT2
00011096        41FA 0036             1406.  AC2INT     LEA AC2MES(PC),A0      ;PNT TO MESS
0001109A        6100 03CA             1407.             BSR OUTMESC
0001109E        102E 0003             1408.             MOVE.B 3(A6),D0        ;GET DATA
000110A2        6100 0386             1409.             BSR SENDBYC          ;ECHO DATA TO CON
000110A6        4CDF 4101             1410.  DUNINT2    MOVEM.L (SP)+,D0/A0/A6 ;REST ENVIRON.
```

```
000110AA  4E73                1411.          RTE                     ;RETURN
                              1412.  *
000110AC  49 6E 74 65 72      1413.  IN2MES  DC.B 'Interrupt Level 2',EOT
          72 75 70 74 20
          4C 65 76 65 6C
          20 32 04
000110BE  41 43 49 41 20      1414.  AC1MES  DC.B 'ACIA #1 DATA = ',EOT
          23 31 20 44 41
          54 41 20 3D 20
          04
000110CE  41 43 49 41 20      1415.  AC2MES  DC.B 'ACIA #2 DATA = ',EOT
          23 32 20 44 41
          54 41 20 3D 20
          04
                              1416.  *
                              1417.  *
                              1418.  * SUBROUTINE STOREG
                              1419.  * THIS ROUTINE IS CALLED BY ANY ROUTINE THAT
                              1420.  * INTENDS TO DISPLAY REGISTERS. OPERATES IN
                              1421.  * SUPERVISOR STATE ONLY. ON STACK IS PCH AND
                              1422.  * PCL OF CALLER, AND THEN PCH AND PCL OF THE
                              1423.  * SECOND CALLER. IF AN EXCEPTION MADE THE
                              1424.  * CALL, NOTE THAT SR IS ON THE STACK TOO,
                              1425.  * AND CONTENTS OF STRGTMP IS 1, ELSE 0
                              1426.  *
000110DE  40E7                1427.  STOREG  MOVE.W SR,-(SP)           ;SAVE STATUS
000110E0  2F0E                1428.          MOVE.L A6,-(SP)   ;PUT A5,A6 IN STK
000110E2  2F0D                1429.          MOVE.L A5,-(SP)          ;KEEP A6
000110E4  4BFA EE88           1430.          LEA STRGTMP(PC),A5  ;PNT TO STRGTMP
000110E8  4DFA EE24           1431.          LEA REGSTOR(PC),A6  ;PNT TO REGSTOR
000110EC  48D6 1FFF           1432.          MOVEM.L D0-D7/A0-A4,(A6)  ;STORE OFF
000110F0  DDFC 00000034       1433.          ADDA.L #52,A6       ;INCREMENT TO NEXT
000110F6  2CDF                1434.          MOVE.L (SP)+,(A6)+  ;STORE A5
000110F8  2CDF                1435           MOVE.L (SP)+,(A6)+  ;STORE A6
000110FA  2F0D                1436.          MOVE.L A5,-(SP)     ;SAVE A5
000110FC  4E6D                1437.          MOVE.L USP,A5       ;GET USP
000110FE  2CCD                1438.          MOVE.L A5,(A6)+     ;STOR OFF USP
00011100  2A5F                1439.          MOVE.L (SP)+,A5     ;RESTOR PNTER
00011102  2CCF                1440.          MOVE.L SP,(A6)+     ;STORE OFF SP
00011104  0815 0000           1441.          BTST.B #0,(A5)      ;TEST IF EXCEPTION
00011108  67 0C               1442.          BEQ.S NOTEXC        ;BRA NOT EXCEPTION
                              1443.  * EXCEPTION CAUSED THE CALL, WASTE PREVIOUS SR
0001110A  3A5F                1444.          MOVE.W (SP)+,A5
                              1445.  * NOW, GET SR BEFORE THE EXCEPTION OCCURRED
0001110C  3CEF 0004           1446.          MOVE.W 4(SP),(A6)+
00011110  2CEF 0006           1447.          MOVE.L 6(SP),(A6)+
00011114  60 06               1448.          BRA.S DUNSTRG
                              1449.  * EXCEPTION DIDN'T CAUSE CALL, NO SR ON STACK
00011116  3CDF                1450.  NOTEXC  MOVE.W (SP)+,(A6)+  ;STORE STATUS
00011118  2CEF 0004           1451.          MOVE.L 4(SP),(A6)+  ;PC OF CALLER
0001111C  4DFA EDF0           1452.  DUNSTRG LEA REGSTOR(PC),A6  ;PNT BCK  REGSTOR
                              1453.  * LET'S RESTORE A5 AND A6 TO THEIR STATE
                              1454.  * BEFORE CALL
00011120  DDFC 00000034       1455.          ADDA.L #52,A6   ;POINT TO WHERE A5 IS
00011126  2A5E                1456.          MOVEA.L (A6)+,A5        ;RESTORE A5
00011128  2C56                1457.          MOVEA.L (A6),A6         ;RESTORE A6
0001112A  4E75                1458.          RTS
                              1459.  *
                              1460.  * SUBROUTINE OUTREG
                              1461.  * ROUTINE TO DISPLAY CONTENTS OF ALL OF THE
                              1462.  * REG CONTENTS PREVIOUS TO THE CALL TO REGSTOR
                              1463.  * ASSUME MEMORY SETUP AS DEFINED BY REGSTOR
                              1464.  * NO VOLATILE REGISTERS
                              1465.  * REG LIST CODE: CHR CHR - SPACE,CODE,OFFSET
                              1466.  * WHERE CODE =1 FOR LONGWORD,0 FOR WORD
                              1467.  *
                              1468.  *
0001112C  48E7 E0C0           1469.  OUTREG  MOVEM.L D0-D2/A0-A1,-(SP) ;GET  ROOM
00011130  7412                1470.          MOVEQ #COUNTRG,D2       ;INIT CNTR REG
00011132  43FA EDDA           1471.          LEA REGSTOR(PC),A1      ;REGSTOR ADDR
00011136  41FA 0044           1472.          LEA REGLIST(PC),A0      ;POINT TTL TBLE
0001113A  4281                1473.          CLR.L D1
0001113C  2010                1474.  MORREG  MOVE.L (A0),D0          ;GET TTL IN D0
0001113E  6100 02BA           1475.          BSR SENDLOC
00011142  1228 0005           1476.          MOVE.B 5(A0),D1         ;GET THE OFFSET
00011146  0828 0000 0004      1477.          BTST.B #0,4(A0)         ;IS IT A WORD?
0001114C  67 1A               1478.          BEQ.S WRDREG            ;HANDLE WRD REG
0001114E  2031 1000           1479.          MOVE.L 0(A1,D1.W),D0    ;GET THE VALUE
00011152  6100 03D2           1480.          BSR DISBUF8             ;CNVRT,STR SEND
00011156  6100 02EA           1481.          BSR SENCLFC             ;PRETTY DISPLAY
```

```
0001115A  D1FC 00000006  1482.  STILMOR ADDA.L #6,A0            ;ADJUST PNTR
00011160  51CA FFDA      1483.          DBRA D2,MORREG          ;DONE YET?
00011164  6000 0010      1484.          BRA ALREGOT             ;FINISH OFF
00011168  3031 1000      1485.  WRDREG  MOVE.W 0(A1,D1.W),D0   ;GET THE WORD
0001116C  6100 03D0      1486.          BSR DISBUF4             ;CNVRT  SEND IT
00011170  6100 02D0      1487.          BSR SENCLFC             ;MAKE IT NICE
00011174  60 E4          1488.          BRA.S STILMOR
00011176  4CDF 0307      1489.  ALREGOT MOVEM.L (SP)+,D0-D2/A0-A1 ;RSTR ENVIN.
0001117A  4E75           1490.          RTS                     ;OUTTA HERE!!!!
                         1491.  *
                         1492.  *
                         1493.  * DATA SECTION FOR OUTREG
                         1494.  *
0001117C  50 43 2D 20 01 1495.  REGLIST DC.B 'PC- ',1,70
          46
00011182  53 52 2D 20 00 1496.          DC.B 'SR- ',0,68
          44
00011188  53 53 2D 20 01 1497.          DC.B 'SS- ',1,64
          40
0001118E  55 53 2D 20 01 1498.          DC.B 'US- ',1,60
          3C
00011194  41 36 2D 20 01 1499.          DC.B 'A6- ',1,56
          38
0001119A  41 35 2D 20 01 1500.          DC.B 'A5- ',1,52
          34
000111A0  41 34 2D 20 01 1501.          DC.B 'A4- ',1,48
          30
000111A6  41 33 2D 20 01 1502.          DC.B 'A3- ',1,44
          2C
000111AC  41 32 2D 20 01 1503.          DC.B 'A2- ',1,40
          28
000111B2  41 31 2D 20 01 1504.          DC.B 'A1- ',1,36
          24
000111B8  41 30 2D 20 01 1505.          DC.B 'A0- ',1,32
          20
000111BE  44 37 2D 20 01 1506.          DC.B 'D7- ',1,28
          1C
000111C4  44 36 2D 20 01 1507.          DC.B 'D6- ',1,24
          18
000111CA  44 35 2D 20 01 1508.          DC.B 'D5- ',1,20
          14
000111D0  44 34 2D 20 01 1509.          DC.B 'D4- ',1,16
          10
000111D6  44 33 2D 20 01 1510.          DC.B 'D3- ',1,12
          0C
000111DC  44 32 2D 20 01 1511.          DC.B 'D2- ',1,8
          08
000111E2  44 31 2D 20 01 1512.          DC.B 'D1- ',1,4
          04
000111E8  44 30 2D 20 01 1513.          DC.B 'D0- ',1,0
          00
                         1514.  *
                         1515.  * TRAP #0 HANDLER ROUTINE
                         1516.  * TRAP #0 IS USED BY USER ROUTINE TO CALL
                         1517.  * THE MONITOR COMMAND LEVEL, AND A SWITCH
                         1518.  * TO THE SUPERVISOR STATE OCCURS.
                         1519.  * CHANGES PROCESSOR STATE, DISABLE TRACING
                         1520.  * CLEANS OFF EXCEPTION STACK, BRANCH TO
                         1521.  * COMMAND LEVEL
000111EE  46FC 2000      1522.  HTRAP0  MOVE.W #$2000,SR ;TRACE OFF, SUP MOD
000111F2  DFFC 0000000A  1523.          ADDA.L #10,SP  ;WIPE OUT EXCEPT STK
000111F8  6000 EF7A      1524.          BRA CMDLVL
                         1525.  *TRAP #1 AND TRAP #2 ROUTINES
                         1526.  *USED TO MAKE PROCESSOR STATE SWITCHES FOR
                         1527.  *SYSTEM TEST PURPOSES.  NOTE STACKS OF BOTH
                         1528.  *TYPES SHOULD BE DEFINED
000111FC  0257 DFFF      1529.  HTRAP1  ANDI.W #$DFFF,(SP) ;CLEAR BIT TO USER
00011200  4E73           1530.          RTE                ;USER MODE EXEC.
00011202  0057 2000      1531.  HTRAP2  ORI.W #$2000,(SP) ;SET BIT FOR SUP.
00011206  4E73           1532.          RTE                ;SUPER MODE.
                         1533.  *
                         1534.  * TRAP #15 HANDLER
                         1535.  * TRAP 15 IS USED BY USER STATE PROGS TO
                         1536.  * CALL ROUTINES IN THE MONITOR
                         1537.  * SYNTAX IS:
                         1538.  * TRAP #15
                         1539.  * DC.W (NUMBER)
                         1540.  * WHERE (NUMBER) IS 0 TO 29, DETERMINES
                         1541.  * MONITOR ROUTINE TO CALL
                         1542.  * REGISTERS MUST BE SET UP FOR SERVICE!
```

```
                                1543.  * A5 NEVER USED BY ROUTINES CALLED
                                1544.  * REGISTERS TRANSPARENT TO THIS ROUTINE
                                1545.  * BUT ALTERED ACCORDING TO ROUTINE ACCESSED
00011208  48E7 0104             1546.  HTRAP15 MOVEM.L D7/A5,-(SP) ;WORK SPACE
0001120C  4287                  1547.          CLR.L D7
0001120E  2A6F 000A             1548.          MOVEA.L 10(SP),A5   ;GET PC
00011212  54AF 000A             1549.          ADDQ.L #2,10(SP)    ;ADJUST FOR NEXT
00011216  3E15                  1550.          MOVE.W (A5),D7      ;GET VECTOR #
00011218  0C47 001D             1551.          CMPI.W #29,D7       ;OUT OF RANGE?
0001121C  6E 18                 1552.          BGT.S ERTRP15
0001121E  CEFC 0002             1553.          MULU #2,D7       ;ADJUST TO POINT
00011222  4BFA 0056             1554.          LEA TBL15(PC),A5    ;POINT TO TABLE
00011226  4BF5 7000             1555.          LEA (A5,D7.W),A5    ;POINT TO ENTRY
0001122A  3E15                  1556.          MOVE.W (A5),D7      ;GET OFFSET
0001122C  DBC7                  1557.          ADDA.L D7,A5        ;CALCULATE JUMP
0001122E  2E1F                  1558.          MOVE.L (SP)+,D7     ;RESTORE D7
00011230  4E95                  1559.          JSR (A5)            ;DO JUMP
00011232  2A5F                  1560.          MOVE.L (SP)+,A5     ;A5 NOT USED EVER
00011234  4E73                  1561.          RTE
00011236  41FA 000A             1562.  ERTRP15 LEA TRP15MS(PC),A0 ;BAD CALL NUMBER
0001123A  6100 022A             1563.          BSR OUTMESC         ;USER WIPED OUT
0001123E  6000 EF2A             1564.          BRA REINIT  ;USE INIT SSP BIG ERROR
                                1565.  *
                                1566.  *
00011242  42 61 64 20 54        1567.  TRP15MS DC.B 'Bad TRAP #15 call Number..'
          52 41 50 20 23
          31 35 20 63 61
          6C 6C 20 4E 75
          6D 62 65 72 2E
          2E
0001125C  52 65 69 6E 69        1568.          DC.B 'Reinitializing Supervisor SP'
          74 69 61 6C 69
          7A 69 6E 67 20
          53 75 70 65 72
          76 69 73 6F 72
          20 53 50
00011278  04                    1569.          DC.B EOT
                                1570.  *
          -- boundary align
0001127A  015C                  1571.  TBL15    DC.W SENDLO-*
0001127C  017E                  1572.          DC.W SENDLOC-*
0001127E  0186                  1573.          DC.W SENDLOH-*
00011280  015C                  1574.          DC.W SENDWO-*
00011282  0190                  1575.          DC.W SENDWOC-*
00011284  0198                  1576.          DC.W SENDWOH-*
00011286  015C                  1577.          DC.W SENDBY-*
00011288  01A2                  1578.          DC.W SENDBYC-*
0001128A  01AA                  1579.          DC.W SENDBYH-*
0001128C  0162                  1580.          DC.W SENDCLF-*
0001128E  01B4                  1581.          DC.W SENCLFC-*
00011290  01BE                  1582.          DC.W SENCLFH-*
00011292  0214                  1583.          DC.W INCH-*
00011294  01F6                  1584.          DC.W INCHCON-*
00011296  0202                  1585.          DC.W INCHHOS-*
00011298  024E                  1586.          DC.W ASCBIN-*
0001129A  01DE                  1587.          DC.W COMESCO-*
0001129C  01CA                  1588.          DC.W OUTMESC-*
0001129E  01D2                  1589.          DC.W OUTMESH-*
000112A0  0212                  1590.          DC.W PUT8HX-*
000112A2  0218                  1591.          DC.W PUT6HX-*
000112A4  021C                  1592.          DC.W PUT4HX-*
000112A6  0222                  1593.          DC.W PUT2HX-*
000112A8  0228                  1594.          DC.W PUTCHR-*
000112AA  027C                  1595.          DC.W DISBUF8-*
000112AC  0292                  1596.          DC.W DISBUF4-*
000112AE  02AA                  1597.          DC.W DISBUF2-*
000112B0  025E                  1598.          DC.W OUTDSBF-*
000112B2  02C0                  1599.          DC.W BLDNUM-*
000112B4  0308                  1600.          DC.W BLDSTRG-*
                                1601.  *
                                1602.  *
000112B6  48E7 6060             1603.  ADDRBYT MOVEM.L D1-D2/A1-A2,-(SP) ;SAVE OFF ENV
000112BA  7203                  1604.          MOVEQ.L #ADRCONT,D1      ;ADDR BYT COUNT
000112BC  227C 00012FCD         1605.          MOVEA.L #PPSR,A1         ;STATUS REG
000112C2  247C 00012FC8         1606.          MOVEA.L #PADR,A2         ;A DATA REG
000112C8  7400                  1607.          MOVEQ.L #H1S,D2          ;BIT TO TEST
000112CA  4280                  1608.          CLR.L D0
000112CC  E188                  1609.  LOOP1A  LSL.L #SHIFT8,D0         ;ADJUST SIGNIF
000112CE  0511                  1610.  LOOP1   BTST.B D2,(A1)           ;BYTE READY?
000112D0  67 FC                 1611.          BEQ.S LOOP1              ;WAIT TILL SO
```

```
000112D2  1012           1612.          MOVE.B (A2),D0          ;GET BYTE
000112D4  51C9 FFF6      1613.          DBRA D1,LOOP1A          ;DO 4 TIMES
000112D8  2040           1614.          MOVEA.L D0,A0           ;ADDRESS->A0
000112DA  4280           1615.          CLR.L D0
000112DC  323C 0001      1616.          MOVE.W #BYTCONT,D1      ;DATA BYT COUNT
000112E0  E188           1617.  LOOP2A  LSL.L #SHIFT8,D0        ;ADJUST SIGNIF
000112E2  0511           1618.  LOOP2   BTST.B D2,(A1)          ;BYTE READY?
000112E4  67 FC          1619.          BEQ.S LOOP2             ;WAIT TILL SO
000112E6  1012           1620.          MOVE.B (A2),D0          ;GET BYTE
000112E8  51C9 FFF6      1621.          DBRA D1,LOOP2A          ;DO TWICE
000112EC  4CDF 0606      1622.          MOVEM.L (SP)+,D1-D2/A1-A2 ;RESTORE
000112F0  4E73           1623.          RTE
                         1624.  *
                         1625.  * EXCEPTION ROUTINE BLOCKIN (TRAP #5) TO READ
                         1626.  * (D0.W)
                         1627.  * BYTES OF DATA TO MEMORY STARTING AT (A0.L)
                         1628.  * ASSUMES PORT CONFIGURED, ENABLED BEFORE ENTRY
                         1629.  * AND BYTE COUNT > 0 ! REGISTERS TRANSPARENT
                         1630.  *
000112F2  48E7 C0E0      1631.  BLOCKIN MOVEM.L D0-D1/A0-A2,-(SP)
000112F6  5340           1632.          SUBQ.W #1,D0            ;ADJUST FOR DBRA
000112F8  227C 00012FCD  1633.          MOVEA.L #PPSR,A1        ;PORT STATUS
000112FE  247C 00012FC8  1634.          MOVEA.L #PADR,A2        ;A DATA REG
00011304  7200           1635.          MOVEQ.L #H1S,D1
00011306  0311           1636.  LOOP3   BTST.B D1,(A1)          ;BYTE READY?
00011308  67 FC          1637.          BEQ.S LOOP3             ;WAIT TILL SO
0001130A  10D2           1638.          MOVE.B (A2),(A0)+       ;READ & STORE
0001130C  51C8 FFF8      1639.          DBRA D0,LOOP3
00011310  4CDF 0703      1640.          MOVEM.L (SP)+,D0-D1/A0-A2 ;RESTORE
00011314  4E73           1641.          RTE
                         1642.  *
                         1643.  * EXCEPTION ROUTINE BLOCKOUT (TRAP #6) TO
                         1644.  * OUTPUT DATA TO
                         1645.  * OUTPUT PORT B.  BYTE COUNT > 0 IS IN D0.W
                         1646.  * START ADDRESS IN A0.L.  ASSUMES PORT B IS
                         1647.  * CONFIGURED AND ENABLED. WAITS FOR RECEIVER
                         1648.  * READY TO RECEIVE.  REGISTERS TRANSPARENT.
                         1649.  * ASSUMES RECEIVER IS "AWARE" WHERE TO PUT
                         1650.  * DATA ITSELF.  OTHERWISE, APPEND DEST ADDRESS
                         1651.  * TO DATA BLOCK, THEN BYTE COUNT (WORD) OF THE
                         1652.  * DATA BLOCK TO BE SENT, THEN DATA BLOCK ITSELF
                         1653.  * NOTE THAT BYTE COUNT SENT TO THIS ROUTINE
                         1654.  * MUST BE [DATA BYTE COUNT]+6!!, ADDRESS
                         1655.  * ADJUSTED TOO!
                         1656.  *
00011316  48E7 C0E0      1657.  BLOKOUT MOVEM.L D0-D1/A0-A2,-(SP)
0001131A  5340           1658.          SUBQ.W #1,D0            ;ADJUST FOR DBRA
0001131C  7206           1659.          MOVEQ.L #H3LEV,D1       ;BIT TO TEST
0001131E  227C 00012FCD  1660.          MOVEA.L #PPSR,A1
00011324  247C 00012FC9  1661.          MOVEA.L #PBDR,A2
0001132A  0311           1662.  LPREDY  BTST.B D1,(A1)          ;RCVR READY?
0001132C  66 FC          1663.          BNE.S LPREDY            ;WAIT TILL SO
0001132E  7202           1664.          MOVEQ.L #H3S,D1         ;BIT TO TEST
00011330  0311           1665.  LPOUT   BTST.B D1,(A1)          ;BYTE READY?
00011332  67 FC          1666.          BEQ.S LPOUT             ;WAIT TILL SO
00011334  1498           1667.          MOVE.B (A0)+,(A2)       ;SEND BYTE
00011336  51C8 FFF8      1668.          DBRA D0,LPOUT           ;DO D0 TIMES
0001133A  0311           1669.  LPOUTA  BTST.B D1,(A1)  ;ENSURE LAST BYTE GONE
0001133C  67 FC          1670.          BEQ.S LPOUTA    ;
0001133E  4CDF 0703      1671.          MOVEM.L (SP)+,D0-D1/A0-A2 ;RESTORE
00011342  4E73           1672.          RTE
                         1673.  *
                         1674.  * EXCEPTION ROUTINE SNADBYT (TRAP #7)
                         1675.  * SENDS DEST ADDRESS AND BYTE COUNT OF OUTGOING
                         1676.  * DATA. RARELY USED BY SLAVE, BUT HERE FOR
                         1677.  * GENERALITY.  ALTERNATIVELY, USE FIX DESCRIBED
                         1678.  * IN TRAP6 HANDLER.  DOES A CHECK TO SEE IF
                         1679.  * RECEIVER READY (H3LEV=0) BEFORE SENDING
                         1680.  * DEST START ADDRESS IN A0.L, BYTE COUNT IN
                         1681.  * D0.W.  ALL REGISTERS TRANSPARENT
                         1682.  *
00011344  48E7 E0F0      1683.  SNADBYT MOVEM.L D0-D2/A0-A3,-(SP)
00011348  264F           1684.          MOVEA.L SP,A3   ;KLUDGE BYTE POSTINC
0001134A  7205           1685.          MOVEQ.L #$05,D1         ;BYTE COUNTER
0001134C  7406           1686.          MOVEQ.L #H3LEV,D2
0001134E  227C 00012FCD  1687.          MOVE.L #PPSR,A1
00011354  247C 00012FC9  1688.          MOVE.L #PBDR,A2
0001135A  3700           1689.          MOVE.W D0,-(A3) ;USE STACK FOR STORAGE
0001135C  2708           1690.          MOVE.L A0,-(A3) ;THEN PULL OFF IN ORDER
0001135E  0511           1691.  REDYLP  BTST.B D2,(A1)  ;RECEIVER READY?
```

```
00011360  66 FC          1692.          BNE.S REDYLP
00011362  7402           1693.          MOVEQ.L #H3S,D2 ;SENSE BIT
00011364  0511           1694.  OUTLP    BTST.B D2,(A1)
00011366  67 FC          1695.          BEQ.S OUTLP
00011368  149B           1696.          MOVE.B (A3)+,(A2)
0001136A  51C9 FFF8      1697.          DBRA D1,OUTLP           ;SEND 6 BYTES
0001136E  0511           1698.  OUTLPA   BTST.B D2,(A1)  ;ENSURE LAST BYTE GONE
00011370  67 FC          1699.          BEQ.S OUTLPA    ;
00011372  4CDF 0F07      1700.          MOVEM.L (SP)+,D0-D2/A0-A3 ;RESTORE
00011376  4E73           1701.          RTE
                         1702.  *
                         1703.  * EXCEPTION ROUTINE NETCONF
                         1704.  * EXPECTS A POINTER TO A 10 BYTE BLOCK IN WHICH
                         1705.  * CONFIGURATION CONTROL WORDS RESIDE, 1ST BYTE
                         1706.  * FOR CBS1 (HI 4 BITS) AND CBS0 (LO 4 BITS),
                         1707.  * LAST FOR CBS18 AND CBS19
                         1708.  * SIMPLY COPIES TABLE TO CBS CONTROL REGISTERS
                         1709.  * AS FAST AS POSSIBLE. ALL REGISTERS TRANS
                         1710.  *
00011378  48E7 00C0      1711.  NETCONF MOVEM.L A0-A1,-(SP,
0001137C  227C 00012FB0  1712.          MOVEA.L #CBS0,A1
00011382  22D8           1713.          MOVE.L (A0)+,(A1)+
00011384  22D8           1714.          MOVE.L (A0)+,(A1)+
00011386  32D8           1715.          MOVE.W (A0)+,(A1)+
00011388  4CDF 0300      1716.          MOVEM.L (SP)+,A0-A1
0001138C  4E73           1717.          RTE
                         1718.  *
                         1719.  * EXCEPTION ROUTINE SENSAK
                         1720.  * EXPECTS A 4 BIT MASK IN D0.B. EACH BIT SET
                         1721.  * INDICATES SLAVE # FROM WHOM TO EXPECT AN
                         1722.  * INTERRUPT. WAIT ONLY FOR THOSE INTERRUPTS
                         1723.  * AND ACKNOWLEDGE.
                         1724.  * USES FIXED PRIORITY SCHEME WHEN POLLING
                         1725.  * 0->1->2->3. AND FIRST COME FIRST SERVED.
                         1726.  * TO DO ROUND ROBIN, USE MULTIPLE
                         1727.  * CALLS WITH 1 BIT SET. HANDLE PRIORITY
                         1728.  * ROTATION IN CALLING PROGRAM
                         1729.  *
0001138E  0C00 0000      1730.  SENSAK  CMPI.B #$00,D0  ;ANY EXPECTED?
00011392  67 3E          1731.          BEQ.S DONSERV   ;QUIT IF SO
00011394  48E7 F880      1732.          MOVEM.L D0-D4/A0,-(SP)
00011398  7804           1733.          MOVEQ.L #PC4,D4 ;TSTCYCLE* PIN BIT4
0001139A  7203           1734.          MOVEQ.L #3,D1   ;D1 IS BITN
0001139C  1401           1735.          MOVE.B D1,D2    ;D2 IS A MASK
0001139E  207C 00012FCC  1736.          MOVEA.L #PCDR,A0 ;A0 POINTS TO PCDR
000113A4  0810 0000      1737.  INTEST  BTST.B #PC0,(A0) ;INT PENDING?
000113A8  66 FA          1738.          BNE.S INTEST
000113AA  5201           1739.  ADDMOD4 ADDQ.B #1,D1
000113AC  C202           1740.          AND.B D2,D1     ;DO MOD 3 ADDITION
000113AE  0300           1741.          BTST.L D1,D0    ;IS BITN SET?
000113B0  67 F8          1742.          BEQ.S ADDMOD4
000113B2  1601           1743.          MOVE.B D1,D3    ;INQUIRE IF INT
000113B4  E50B           1744.          LSL.B #2,D3     ;FROM SLAVE BITN
000113B6  09C3           1745.          BSET.L D4,D3    ;ADDR & TSTCYCLE=1
000113B8  1083           1746.          MOVE.B D3,(A0)  ;PLACE IN PCDR
000113BA  0990           1747.          BCLR.B D4,(A0)  ;TSTCYCLE* ACTIVE
000113BC  1610           1748.          MOVE.B (A0),D3  ;READ RESPONSE
000113BE  09D0           1749.          BSET.B D4,(A0)  ;TSTCYCLE* INACTIVE
000113C0  0803 0001      1750.          BTST.L #PC1,D3  ;WAS TSTACK* ACTIVE?
000113C4  66 E4          1751.          BNE.S ADDMOD4   ;SLAVE BITN NOT IRQ
                         1752.  *                       ;TEST NEXT ONE
000113C6  0380           1753.          BCLR.L D1,D0    ;CLEAR BIT IN MASK
000113C8  0C00 0000      1754.          CMPI.B #0,D0    ;TEST IF DONE
000113CC  66 D6          1755.          BNE.S INTEST    ;GET OTHER EXPECTED
                         1756.  *                       ;INTERRUPTS
000113CE  4CDF 011F      1757.          MOVEM.L (SP)+,D0-D4/A0  ;RESTORE
000113D2  4E73           1758.  DONSERV RTE
000113D4  <2>            1759.  ENDCRIT DS.B 2
                         1760.  *
                         1761.  *       Low level I/O Routines
                         1762.  ********************************************
                         1763.  * The following are the lowest level I/O    *
                         1764.  * routines to be used by the monitor        *
                         1765.  * functions.                                *
                         1766.  * Paul A. Smeulders, January 31, 1988       *
                         1767.  ********************************************
                         1768.  ********************************************
                         1769.  *  OUTPUT SUBROUTINES
                         1770.  *  SENDLO     SEND LONGWORD
                         1771.  *  SENDWO     SEND WORD
```

```
                        1772.  *  SENDBY     SEND BYTE
                        1773.  *  SENDCLF    SEND CARRIAGE RETURN, LINEFEED
                        1774.  *  DATA TO BE SENT IS IN D0, ADDRESS OF I/O
                        1775.  *  PORT IS IN A6. OTHER REGISTERS ARE
                        1776.  *  UNAFFECTED.  NOTE THAT FOR CRLF, SINCE
                        1777.  *  DATA IS KNOWN, WHATEVER MIGHT BE IN D0 IS
                        1778.  *  STACKED, AND UNSTACKED, AND IS THUS
                        1779.  *  UNAFFECTED
                        1780.  ********************************************
                        1781.  *
000113D6  4840          1782.  SENDLO  SWAP D0          ;POSITION MSW IN LSW
000113D8  61 02         1783.          BSR.S SENDWO     ;SEND THE WORD
000113DA  4840          1784.          SWAP D0          ;RESTORE ORDER
000113DC  E058          1785.  SENDWO  ROR #8,D0        ;POSITION BYTE
000113DE  61 02         1786.          BSR.S SENDBY     ;SEND THE BYTE
000113E0  E158          1787.          ROL #8,D0        ;REPOSITION BYTE
000113E2  0816 0001     1788.  SENDBY  BTST #1,(A6)     ;TDRE BIT SET?
000113E6  67 FA         1789.          BEQ SENDBY       ;WAIT UNITL SO
000113E8  1D40 0001     1790.          MOVE.B D0,1(A6)  ;SEND THE BYTE
000113EC  4E75          1791.          RTS              ;RETURN TO CALLER
000113EE  2F00          1792.  SENDCLF MOVE.L D0,-(SP)  ;STACK D0
000113F0  303C 0D0A     1793.          MOVE.W #CRLF,D0  ;ESTABLISH DATA
000113F4  61 E6         1794.          BSR SENLWO       ;SEND AS A WORD
000113F6  201F          1795.          MOVE.L (SP)+,D0  ;UNSTACK D0
000113F8  4E75          1796.          RTS              ;AND RETURN
                        1797.  *
                        1798.  * Routines that can be called by the user to
                        1799.  * do same functions as sendlo,sendwo,sendby,
                        1800.  * except address of acia not necessarily is
                        1801.  * in A6.
                        1802.  *
000113FA  2F0E          1803.  SENDLOC MOVE.L A6,-(SP)     ;STACK A6
000113FC  2C7C 00012FFC 1804.          MOVE.L #ACIA1,A6  ;LOAD A6
00011402  60 08         1805.          BRA.S COMLO       ;DO COMMON
00011404  2F0E          1806.  SENDLOH MOVE.L A6,-(SP)     ;STACK A6
00011406  2C7C 00012FFE 1807.          MOVE.L #ACIA2,A6  ;LOAD A6
0001140C  61 C8         1808.  COMLO   BSR SENDLO        ;SEND IT
0001140E  2C5F          1809.          MOVE.L (SP)+,A6   ;RESTORE A6
00011410  4E75          1810.          RTS               ;RETURN
00011412  2F0E          1811.  SENDWOC MOVE L A6,-(SP)    ;STACK A6
00011414  2C7C 00012FFC 1812.          MOVE.L #ACIA1,A6  ;LOAD ADDRESS
0001141A  60 08         1813.          BRA.S COMWO       ;DO COMMON
0001141C  2F0E          1814.  SENDWOH MOVE.L A6,-(SP)    ;STACK A6
0001141E  2C7C 00012FFE 1815.          MOVE.L #ACIA2,A6  ;LOAD ADDRESS
00011424  61 B6         1816.  COMWO   BSR SENDWO        ;SEND IT
00011426  2C5F          1817           MOVE.L (SP)+,A6   ;RESTORE A6
00011428  4E75          1818.          RTS               ;RETURN
0001142A  2F0E          1819.  SENDBYC MOVE.L A6,-(SP)    ;STACK A6
0001142C  2C7C 00012FFC 1820.          MOVE.L #ACIA1,A6  ;LOAD ADDRESS
00011432  60 08         1821.          BRA.S COMBY       ;DO COMMON
00011434  2F0E          1822.  SENDBYH MOVE.L A6,-(SP)    ;STACK A6
00011436  2C7C 00012FFE 1823.          MOVE.L #ACIA2,A6  ;LOAD ADDRESS
0001143C  61 A4         1824.  COMBY   BSR SENDBY        ;SEND IT
0001143E  2C5F          1825.          MOVE.L (SP)+,A6   ;RESTORE A6
00011440  4E75          1826.          RTS               ;RETURN
00011442  48E7 8002     1827.  SENCLFC MOVEM.L D0/A6,-(SP)     ;STACK A6
00011446  2C7C 00012FFC 1828.          MOVE.L #ACIA1,A6  ;LOAD A6
0001144C  60 0A         1829.          BRA.S COMCRLF     ;
0001144E  48E7 8002     1830.  SENCLFH MOVEM.L D0/A6,-(SP)
00011452  2C7C 00012FFE 1831.          MOVE.L #ACIA2,A6
00011458  303C 0D0A     1832.  COMCRLF MOVE.W #CRLF,D0
0001145C  6100 FF7E     1833.          BSR SENDWO
00011460  4CDF 4001     1834.          MOVEM.L (SP)+,D0/A6
00011464  4E75          1835.          RTS
                        1836.  *
                        1837.  * OUTPUT MESSAGE SUBROUTINE
                        1838.  * SENDS AN ASCII MESSAGE STRING POINTED TO BY
                        1839.  * A0, UNTIL EOT IS ENCOUNTERED. EOT IS NOT
                        1840.  * SENT.DATA IS ALREADY ASCII CODED, NOTE
                        1841.  * FACILITY TO HAVE MESSAGE TO CONSOLE OR HOST
                        1842.  * SINCE A6 IS STACKED AND PROPERLY LOADED
                        1843.  *
00011466  2F0E          1844.  OUTMESC MOVE.L A6,-(SP)     ;STACK A6
00011468  2C7C 00012FFC 1845.          MOVE.L #ACIA1,A6  ;LOAD A6
0001146E  60 08         1846.          BRA.S COMESCO     ;DO COMMON
00011470  2F0E          1847.  OUTMESH MOVE.L A6,-(SP)     ;STACK A6
00011472  2C7C 00012FFE 1848.          MOVE.L #ACIA2,A6  ;LOAD A6
00011478  1018          1849.  COMESCO MOVE.B (A0)+,D0    ;GET BYTE
0001147A  0C00 0004     1850.          CMPI.B #EOT,D0    ;IS IT EOT?
0001147E  67 06         1851.          BEQ.S ENDMESS     ;QUIT IF SO
```

```
00011480  6100 FF60      1852.          BSR SENDBY          ;SEND THE BYTE
00011484  60 F2          1853.          BRA.S COMESCO       ;REPEAT
00011486  2C5F           1854.  ENDMESS MOVE.L (SP)+,A6     ;RESTORE A6
00011488  4E75           1855.          RTS                 ;AND RETURN
                         1856.  *
                         1857.  * INPUT CHARACTER ROUTINE
                         1858.  * CAN BE USED TO RECEIVE CHARACTERS FROM
                         1859.  * EITHER THE HOST OR THE CONSOLE. AGAIN
                         1860.  * CALLING THE APPROPRIATE ROUTINE WILL
                         1861.  * LOAD A6 WITH THE PORT POINTER
                         1862.  * ELSE CALL INCH IF A6 LOADED ALREADY
                         1863.  * CHARACTER RECEIVED IS PLACED IN D0.B
                         1864.  *
0001148A  2F0E           1865.  INCHCON MOVE.L A6,-(SP)     ;SAVE A6
0001148C  2C7C 00012FFC  1866.          MOVE.L #ACIA1,A6    ;LOAD A6
00011492  61 12          1867.          BSR.S INCH          ;GET CHAR
00011494  2C5F           1868.          MOVE.L (SP)+,A6     ;RESTORE A6
00011496  4E75           1869.          RTS                 ;RETURN
00011498  2F0E           1870.  INCHHOS MOVE.L A6,-(SP)     ;SAVE A6
0001149A  2C7C 00012FFE  1871.          MOVE.L #ACIA2,A6    ;LOAD A6
000114A0  61 04          1872.          BSR.S INCH          ;GET CHAR
000114A2  2C5F           1873.          MOVE.L (SP)+,A6     ;RESTORE A6
0001' ,A4 4E75           1874.          RTS                 ;RETURN
000114A6  0816 0000      1875.  INCH    BTST #0,(A6)        ;RDRF BIT SET?
000114AA  67 FA          1876.          BEQ.S INCH          ;WAIT TILL SO
000114AC  102E 0001      1877.          MOVE.B 1(A6),D0     ;STORE BYTE AT D0
000114B0  4E75           1878.          RTS
                         1879.  *
                         1880.  * ROUTINE TO CONVERT DATA IN D0 INTO ASCII
                         1881.  * FORMAT FOR OUTPUT.THE RESULT IS STORED IN
                         1882.  * IN THE 8 BYTES POINTED TO BY A0. REGISTERS
                         1883.  * D0,D1,AND D2 ARE TRASHED. A0 POINTS TO
                         1884.  * SOME VALID PLACE IN THE WORKSHEET RAM AREA
                         1885.  *
000114B2  4840           1886.  PUT8HX  SWAP D0             ;EXCHANGE WORDS
000114B4  61 0A          1887.          BSR.S PUT4HX        ;CONVERT LOWER WORD
000114B6  4840           1888.          SWAP D0             ;RESTORE D0
000114B8  60 06          1889.          BRA.S PUT4HX        ;CONVERT WORD
000114BA  4840           1890.  PUT6HX  SWAP D0             ;TRANSPOSE WORDS
000114BC  61 0A          1891.          BSR.S PUT2HX        ;CONVERT BYTE
000114BE  4840           1892.          SWAP D0             ;RESTORE
000114C0  3200           1893.  PUT4HX  MOVE.W D0,D1        ;SAVE D0
000114C2  E058           1894.          ROR.W #8,D0         ;SET UP
000114C4  61 02          1895.          BSR.S PUT2HX        ;CONVERT
000114C6  3001           1896.          MOVE.W D1,D0        ;RESTORE D0
000114C8  3400           1897.  PUT2HX  MOVE.W D0,D2        ;SAVE OFF D0
000114CA  E858           1898.          ROR.W #4,D0         ;SET UP NIBBLE
000114CC  61 02          1899.          BSR.S PUTCHR        ;CONVERT AND STORE
000114CE  3002           1900.          MOVE.W D2,D0        ;RESTORE D0
000114D0  C03C 000F      1901.  PUTCHR  AND.B #$0F,D0       ;MASK OFF HIGH NBBL
000114D4  803C 0030      1902.          OR.B #$30,D0        ;ADD 30
000114D8  0C00 0039      1903.          CMPI.B #$39,D0      ;IS IT <9?
000114DC  6F 04          1904.          BLE.S SAVECHR       ;SAVE IF NOT
000114DE  0600 0007      1905.          ADDI.B #7,D0        ;ADJUST FOR A TO F
000114E2  10C0           1906.  SAVECHR MOVE.B D0,(A0)+     ;STORE OFF CHAR.
000114E4  4E75           1907.          RTS
                         1908.  *
                         1909.  * ROUTINE TO CONVERT A SINGLE ASCII CHARACTER
                         1910.  * BINARY EQUIVALENT. IF IT DOES NOT FALL IN
                         1911.  * RANGE $0 TO $F, ERROR OCCURS, THE LSB OF D0
                         1912.  * IS SET TO $FF, AND THE OFFENDING ASCII VALUE
                         1913.  * IS IN THE MSB OF D0.W
                         1914.  *
                         1915.  *
000114E6  0C00 0030      1916.  ASCBIN  CMPI.B #$30,D0      ;IS IT <0?
000114EA  6D 1A          1917.          BLT.S ERRNOBI       ;
000114EC  0C00 0039      1918.          CMPI.B #$39,D0      ;IS IT <=9?
000114F0  6F 0E          1919.          BLE.S OKHEX         ;BRANCH IF SO
000114F2  0C00 0041      1920.          CMPI.B #$41,D0      ;A LETTER?
000114F6  6D 0E          1921.          BLT.S ERRNOBI       ;INVALID
000114F8  0C00 0046      1922.          CMPI.B #$46,D0      ; <=F?
000114FC  6E 08          1923.          BGT.S ERRNOBI       ;STILL NO GOOD
000114FE  5F00           1924.          SUB.B #7,D0         ;ADJUST FOR A-F
00011500  C03C 000F      1925.  OKHEX   AND.B #$F,D0        ;MASK,& PUT IN D0
00011504  4E75           1926.          RTS                 ;RETURN
00011506  E140           1927.  ERRNOBI ASL.W #8,D0         ;MOVE OVER D0
00011508  0040 00FF      1928.          ORI.W #$FF,D0       ;SET ERROR FLAG
0001150C  4E75           1929.          RTS
                         1930.  *
                         1931.  *OUTPUT CONSECUTIVE MEMORY CONTENTS SUBROUTINE
```

```
                        1932.  *SENDS ASCII DATA STORED IN CONSECUTIVE
                        1933.  *LOCATIONS THE START OF WHICH IS POINTED TO BY
                        1934.  *A0
                        1935.  *LENGTH OF MESSAGE IN BYTES IS IN D0
                        1936.  *
0001150E  2F01          1937.  OUTDSBF MOVE.L D1,-(SP) ;PRESERVE D1
00011510  2200          1938.          MOVE.L D0,D1    ;USE D1 TO COUNT
00011512  0481 00000001 1939.          SUBI.L #1,D1    ;ADJUST FOR DBRA
00011518  1018          1940.  OUTMOR  MOVE.B (A0)+,D0 ;GET THE DATA
0001151A  6100 FF0E     1941.          BSR SENDBYC     ;OUT TO CONSOLE
0001151E  51C9 FFF8     1942.          DBRA D1,OUTMOR  ;CONTINUE TILL DONE
00011522  221F          1943.          MOVE.L (SP)+,D1 ;RESTORE D1
00011524  4E75          1944.          RTS             ;ALL DONE
                        1945.  *
                        1946.  * CREATE AND DISPLAY CONTENTS OF A DISPLAY
                        1947.  * BUFFER 8,4,OR 2 BYTES LONG. BINARY NUMBER
                        1948.  * TO BE SENT IS IN D0. CONVERTS AND STORES,
                        1949.  * THEN DISPLAYS WHAT IS IN THE DISPLAY BUFFER
                        1950.  * DATA IS IN D0, OTHER REGISTERS TRANSPARENT
                        1951.  *
00011526  48E7 E080      1952.  DISBUF8 MOVEM.L D0-D2/A0,-(SP) ;CREATE WORK SP
0001152A  41FA EA32     1953.          LEA DISBUF(PC),A0   ;GET ADD OF DSPLAY
0001152E  61 82         1954.          BSR PUT8HX          ;DO LGWD CONVRSION
00011530  41FA EA2C     1955.          LEA DISBUF(PC),A0   ;RESET POINTER
00011534  7008          1956.          MOVEQ #8,D0         ;INITIALIZE COUNTR
00011536  61 D6         1957.          BSR OUTDSBF         ;SEND THE BUFFER
00011538  4CDF 0107     1958.          MOVEM.L (SP)+,D0-D2/A0 ;RESTORE ENVIRO
0001153C  4E75          1959.          RTS
0001153E  48E7 E080     1960.  DISBUF4 MOVEM.L D0-D2/A0,-(SP) ;CREATE WORK SP
00011542  41FA EA1A     1961.          LEA DISBUF(PC),A0   ;GET ADD OF DISPLA
00011546  6100 FF78     1962.          BSR PUT4HX          ;DO WD CONVERSION
0001154A  41FA EA12     1963.          LEA DISBUF(PC),A0   ;RESET POINTER
0001154E  7004          1964.          MOVEQ #4,D0         ;INITIALIZE COUNTR
00011550  61 BC         1965.          BSR OUTDSBF         ;SEND THE BUFFER
00011552  4CDF 0107     1966.          MOVEM.L (SP)+,D0-D2/A0 ;RESTORE ENVIRO
00011556  4E75          1967.          RTS
00011558  48E7 E080     1968.  DISBUF2 MOVEM.L D0-D2/A0,-(SP) ;CREATE WORK SP
0001155C  41FA EA00     1969.          LEA DISBUF(PC),A0   ;GET ADD OF DISPY
00011560  6100 FF66     1970.          BSR PUT2HX          ;DO BYTE CONVERSN
00011564  41FA E9F8     1971.          LEA DISBUF(PC),A0   ;RESET POINTER
00011568  7002          1972.          MOVEQ #2,D0         ;INITIALIZE CONTER
0001156A  61 A2         1973.          BSR OUTDSBF         ;SEND THE BUFFER
0001156C  4CDF 0107     1974.          MOVEM.L (SP)+,D0-D2/A0 ;RESTORE ENVIRO
00011570  4E75          1975.          RTS
                        1976.  *
                        1977.  * SUBROUTINE BLDNUM
                        1978.  * ACCEPTS INPUT FROM THE TERMINAL PORT AS HEX
                        1979.  * DIGITS UNTIL THE FIRST NONHEX DIGIT
                        1980.  * ENCOUNTERED, THE DELIMITER. D0 RETURNS THE
                        1981.  * BINARY NUMBER ENTERED, ZERO FILLED, BUT LAST
                        1982.  * 8 DIGITS USED. A0.B HAS THE DELIMITER, IF
                        1983.  * NO HEX DIGITS BEFORE DELIMITER, A0.W =
                        1984.  * $FF<DEL>, IF DIGITS DO OCCURR, A0.W=$00<DEL>
                        1985.  * D0,A0 TRASHED, OTHERS TRANSPARENT
                        1986.  *
00011572  48E7 6000      1987.  BLDNUM  MOVEM.L D1-D2,-(SP)      ;WORKING ROOM
00011576  4281          1988.          CLR.L D1
00011578  4282          1989.          CLR.L D2
0001157A  6100 FF0E     1990.  GETINP  BSR INCHCON      ;RECEIVE CHR
0001157E  6100 FEAA     1991.          BSR SENDBYC      ;ECHO CHR
00011582  0200 007F     1992.          ANDI.B #ASCMSK,D0        ;STRIP HI BIT
00011586  6100 FF5E     1993.          BSR ASCBIN       ;GET BINARY
0001158A  0C00 00FF     1994.          CMPI.B #$FF,D0   ;CHECK FOR ERROR
0001158E  67 0A         1995.          BEQ.S ENDNUM     ;DONE ON ERROR
00011590  08C2 0000     1996.          BSET #0,D2       ;SET D2 FLAG ON
00011594  E981          1997.          ASL.L #4,D1      ;SET UP FOR NEW DATA
00011596  8200          1998.          OR.B D0,D1       ;APPEND DIGIT
00011598  60 E0         1999.          BRA.S GETINP     ;MORE INPUT
0001159A  E040          2000.  ENDNUM  ASR.W #8,D0      ;POSITION DELIMITER
0001159C  0240 00FF     2001.          ANDI.W #$00FF,D0          ;KEEP IT
000115A0  08C2 0000     2002.          BSET #0,D2       ;FLAG SET?
000115A4  67 10         2003.          BEQ.S NOHEXIN
000115A6  0280 0000FFFF 2004.  NUMSTOR ANDI.L #$FFFF,D0 ;NAIL THE HIGH BITS
000115AC  2040          2005.          MOVEA.L D0,A0    ;PUT IN DELMTR $ CODE
000115AE  2001          2006.          MOVE.L D1,D0     ;RETURN NUMBER
000115B0  4CDF 0006     2007.          MOVEM.L (SP)+,D1-D2      ;RESTORE OLD
000115B4  4E75          2008.          RTS
000115B6  0040 FF00     2009.  NOHEXIN ORI.W #$FF00,D0 ;SET UP NO HEX CODE
000115BA  60 EA         2010.          BRA.S NUMSTOR    ;DO AS USUAL
                        2011.  *
```

```
                                2012.  *
                                2013.  * SUBROUTINE BLDSTRG
                                2014.  * BUILDS A LINEAR STRING OF CHARACTERS AT
                                2015.  * ASCENDING ADDRESSES STRAT WHERE A0 POINTS
                                2016.  * INPUT IS FROM THE CONSOLE
                                2017.  * A0 HAS START ADDRESS FOR STRING
                                2018.  * D0 HAS LIMIT ON NUMBER OF CHRS
                                2019.  * RECEIVES UNTIL NO MORE CHRS ALLOWED, OR
                                2020.  * UNTIL CR. CHRS RECEIVED ARE ECHOED
                                2021.  * D0 RETURNS WITH # CHRS REMAINING
                                2022.  *
000115BC  2F01                  2023.  BLDSTRG MOVE.L D1,-(SP)   ;SAVE D1
000115BE  2200                  2024.          MOVE.L D0,D1
000115C0  0481 00000001         2025.          SUBI.L #1,D1      ;ADJUST FOR DBRA
000115C6  6100 FEC2             2026.  NEXCHR  BSR INCHCON       ;GET CONSOLE INPUT
000115CA  0200 007F             2027.          ANDI.B #ASCMSK,D0  ;STRIP HI BIT
000115CE  0C00 000D             2028.          CMPI.B #CR,D0     ;IS IT CR?
000115D2  67 0A                 2029.          BEQ.S STRGBLT     ;DONE IF SO
000115D4  10C0                  2030.          MOVE.B D0,(A0)+   ;STORE OF CHR
000115D6  6100 FE52             2031.          BSR SENDBYC       ;ECHO CHR
000115DA  51C9 FFEA             2032.          DBRA D1,NEXCHR
000115DE  2001                  2033.  STRGBLT MOVE.L D1,D0   ;HAVE D0 RETURN #LEFT
000115E0  0680 00000001         2034.          ADDI.L #1,D0   ;AND MAKE REASONABLE
000115E6  6100 FE5A             2035.          BSR SENCLFC    ;DO CLFC REGARDLESS
000115EA  221F                  2036.          MOVE.L (SP)+,D1 ;RESTORE D1
000115EC  4E75                  2037.          RTS
000115EE                        2038.          END
    0 Errors
```

I.2 Slave System Program Listing

```
                                1.          TTL SLAVEROM
                                2.  *
                                3.  * SLAVE ROM PROGRAM
                                4.  *
                                5.  * EQUATES
     # 00012FC0                  6.  PGCR     EQU $12FC0
     # 00012FC1                  7.  PSRR     EQU $12FC1
     # 00012FC2                  8.  PADDR    EQU $12FC2
     # 00012FC3                  9.  PBDDR    EQU $12FC3
     # 00012FC4                 10.  PCDDR    EQU $12FC4
     # 00012FC6                 11.  PACR     EQU $12FC6
     # 00012FC7                 12.  PBCR     EQU $12FC7
     # 00012FC8                 13.  PADR     EQU $12FC8
     # 00012FC9                 14.  PBDR     EQU $12FC9
     # 00012FCC                 15.  PCDR     EQU $12FCC
     # 00012FCD                 16.  PPSR     EQU $12FCD
     # 00010020                 17.  RESTART  EQU $10020
     # 0000FED0                 18.  INITSSP  EQU $FED0
     # 00000010                 19.  ENABLEA  EQU $10
     # 00000020                 20.  ENABLEB  EQU $20
     # FFFF0000                 21.  ADDRILL  EQU $FFFF0000
     # 00000000                 22.  NODATA   EQU $0
     # 00000003                 23.  ADRCONT  EQU $03
     # 00000008                 24.  SHIFT8   EQU $08
     # 00000001                 25.  BYTCONT  EQU $01
     # 00000006                 26.  H3LEV    EQU $06
     # 00000002                 27.  H3S      EQU $02
     # 00000000                 28.  H1S      EQU $00
     # 00000004                 29.  H1LEV    EQU $04
     # 00000000                 30.  INPUTS   EQU $00
     # 000000FF                 31.  OUTPUTS  EQU $FF
     # 00000001                 32.  ONEOUT   EQU $01
     # 00000030                 33.  CONFIGA  EQU $30
     # 00000070                 34.  CONFIGB  EQU $70
     # 00002000                 35.  CLERCCR  EQU $2000
     # 0000DFFF                 36.  CLRUS    EQU $DFFF
     # 00002000                 37.  SETSUP   EQU $2000
     # 00000700                 38.  INTMASK  EQU $0700
     # 000000FF                 39.  RESSTAT  EQU $FF
                                40.  *
                                41.  * HIGH RAM WILL BE USED FOR DOWNLOAD/UPLOAD
                                42.  * PROGRAMS SO THAT THEY WILL RUN AT MAXIMUM
                                43.  * SPEED. ROUTINES ARE COPIED FROM ROM TO
                                44.  * RAM AT START-UP.  ROUTINES ARE THEMELVES
                                45.  * WRITTEN AS POSITION INDEPENDANT, SO WILL
                                46.  * EXECUTE PROPERLY AFTER COPIED
                                47.  *
                                48.  *
                                49.  * DEFINE RESET VECTOR FOR STARTUP
                                50.  *
00010000                        51.          ORG.L $10000
00010000  0000FED0              52.          DC.L INITSSP
00010004  00010020              53.          DC.L RESTART
                                54.  *
00010020                        55.          ORG.L $10020
                                56.  *
                                57.  * COPY VECTORS TO VECTOR TABLE
                                58.  * NOT ALL ERROR CONDITIONS HANDLERS IMPLEMENTED
                                59.  * DUE TO ERROR CONDITION RECOGNITION DIFFICULTY
                                60.  * AMONG CELLS.  THIS CAN BE A FUTURE SOFTWARE
                                61.  * PROJECT.  IMPLICIT ASSUMPTION THAT ALL SLAVE
                                62.  * PROGRAMS ARE RUNNABLE.
                                63.  *
00010020  21FC 0000FED0 0000    64.          MOVE.L #INITSSP,0
00010028  21FC 00010020 0004    65.          MOVE.L #RESTART,4
00010030  007C 0700             66.          ORI.W #INTMASK,SR       ;MASK ALL INTS
00010034  41FA 01CE             67.          LEA TRAP14(PC),A0       ;NEWPROG TRAP
00010038  21C8 00B8             68.          MOVE.L A0,184
0001003C  21C8 007C             69.          MOVE.L A0,124           ,NMI VECTOR
00010040  41FA 01F6             70.          LEA TRAP1(PC),A0        ,CALL USER TRAP
00010044  21C8 0084             71.          MOVE.L A0,132
00010048  41FA 01F4             72.          LEA TRAP2(PC),A0        ;CALL SUP TRAP
0001004C  21C8 0088             73.          MOVE.L A0,136
00010050  41FA FF86             74.          LEA TRAP3(PC),A0        ;ASSERT INT AND
```

```
00010054  21C8 008C        75.          MOVE.L A0,140           ;WAIT TRAP
00010058  41FA FEBC        76.          LEA TRAP4(PC),A0        ;ADDRBYT TRAP
0001005C  21C8 0090        77.          MOVE.L A0,144
00010060  41FA FEF0        78.          LEA TRAP5(PC),A0        ;BLOCKIN TRAP
00010064  21C8 0094        79.          MOVE.L A0,148
00010068  41FA FF0C        80.          LEA TRAP6(PC),A0        ;BLOKOUT TRAP
0001006C  21C8 0098        81.          MOVE.L A0,152
00010070  41FA FF32        82.          LEA TRAP7(PC),A0        ;SNADBYT TRAP
00010074  21C8 009C        83.          MOVE.L A0,156
                           84.  *
                           85.  * COPY CRITICAL ROUTINES TO HIGH RAM
                           86.  *
00010078  41FA 0076        87.          LEA DOADDAT(PC),A0 ;START ADDR OF BLK
0001007C  43FA 0186        88.          LEA TRAP14(PC),A1 ;END ADDR OF BLK
00010080  93C8             89.          SUBA.L A0,A1     ;SIZE OF BLK
00010082  2009             90.          MOVE.L A1,D0     ;D0 IS COUNTER
00010084  5380             91.          SUBQ.L #1,D0     ;ADJUST FOR DBRA
00010086  43FA FE58        92.          LEA LOADDAT(PC),A1
0001008A  12D8             93.  CPYLOP  MOVE.B (A0)+,(A1)+
0001008C  51C8 FFFC        94.          DBRA D0,CPYLOP
                           95.  *
                           96.  * INITIALIZE PARALLEL INTERFACE
                           97.  * PORT A INPUT, HANDSHAKE MODE,DISABLED
                           98.  * PORT B OUTPUT, HANDSHAKE MODE, DISABLED
                           99.  * PORT C INPUT, SAVE FOR PC0, OUTPUT = 1.
                          100.  *
00010090  4239 00012FC0    101.          CLR.B PGCR         ;CONTROL REG=0
00010096  4239 00012FC1    102.          CLR.B PSRR         ;DISABLE ANY INTS
0001009C  13FC 0000        103.          MOVE.B #INPUTS,PADDR
          00012FC2
000100A4  13FC 00FF        104.          MOVE.B #OUTPUTS,PBDDR
          00012FC3
000100AC  13FC 00FF        105.          MOVE.B #$FF,PCDR  ,ENSURE PC0=1 ON EN
          00012FCC
000100B4  13FC 0001        106.          MOVE.B #ONEOUT,PCDDR
          00012FC4
000100BC  13FC 0030        107.          MOVE.B #CONFIGA,PACR ;PORT A HANDSHAKE
          00012FC6
000100C4  13FC 0070        108.          MOVE.B #CONFIGB,PBCR ;PORT B HANDSHAKE
          00012FC7
000100CC  13FC 00FF        109           MOVE.B #RESSTAT,PPSR ;RESET STATUS
          00012FCD
                          110.  *
                          111.  * NOW INDICATE "READY AFTER RESTART" TO MASTER
                          112.  *
000100D4  4E43             113.          TRAP #3          ;READY AFTER RESTART
                          114.  *
                          115.  * NOW DOWNLOAD A PROGRAM FROM MASTER
                          116.  * MUST ASSERT "READY" ON THE INTERRUPT BUS OF
                          117.  * OF PORT C.  MASTER KEEPS TRACK OF READY CALLS
                          118.  * SUPERVISOR MODE OPERATION
                          119.  * NOTE THAT WHEN SLAVE PUT INTO READY FOR
                          120.  * PROGRAM MODE AFTER NMI,TRAP14,OR ILLEGAL
                          121.  * ADDRESS, MUST FIRST SERVICE AN SRQ FROM
                          122.  * THAT SLAVE
                          123.  *
000100D6  4E43             124.  NEWPROG TRAP #3          ;READY FOR PROGRAM
                          125.  *
                          126.  * PROGRAM WILL COME IN ON INPUT PORT A,
                          127.  * 1ST 4 BYTES
                          128.  * ARE START ADDRESS (HIGH BYTE FIRST), NEXT 2
                          129.  * ARE BYTE COUNT (MAX 64K). MUST ENABLE RECVR
                          130.  * PORT FIRST
                          131.  *
000100D8  13FC 0010        132.          MOVE.B #ENABLEA,PGCR
          00012FC0
000100E0  4E44             133.          TRAP #4          ;GET ADDR,BYTE COUNT
                          134.  *
                          135.  * START ADDRESS IN A0.L BYTE COUNT IN D0.W
                          136.  * CALL BLOCK INPUT ROUTINE
                          137.  *
000100E2  4E45             138.          TRAP #5          ;GET DATA BLOCKIN
                          139.  *
                          140.  * DISABLE INPUT PORT A, PUT START ADDRESS
                          141.  * ON SUPERVISOR STACK, INDICATE DONE TO
                          142.  * MASTER
                          143.  *
000100E4  4239 00012FC0    144.          CLR.B PGCR
000100EA  2F08             145.          MOVE.L A0,-(SP) ;START OF PROG ON STACK
000100EC  6000 FDF2        146.          BRA LOADDAT
```

```
                                147.  *
                                148.  * SUBROUTINE LOADDAT (DUMMY)
                                149.  * DOWNLOAD DATA FOR PROGRAM, GET START
                                150.  * ADDRESS AND BYTE COUNT, CHECK FOR VALIDITY
                                151.  * IF INVALID, RE-EXECUTE PROGRAM. INVALID
                                152.  * ADDRESS IS INDICATION FROM MASTER THAT A
                                153.  * NEW PROGRAM IS TO BE LOADED, NOT JUST NEW
                                154.  * DATA FOR THE EXISTING PROGRAM
                                155.  *
000100F0  4E43                  156.  DOADDAT TRAP #3          ;READY FOR DATA
000100F2  13FC 0010             157.          MOVE.B #ENABLEA,PGCR
          00012FC0
000100FA  4E44                  158.          TRAP #4          ;ADDRESS AND BYTE COUNT
000100FC  2208                  159.          MOVE.L A0,D1
000100FE  4241                  160.          CLR.W D1                    ;MASK LOW WORD
00010100  B2BC FFFF0000         161.          CMP.L #ADDRILL,D1           ;LEGAL ADDRES?
00010106  67 16                 162.          BEQ.S DONPROG               ;QUIT IF NOT
00010108  B07C 0000             163.          CMP.W #NODATA,D0            ;ANY DATA?
0001010C  67 02                 164.          BEQ.S NONEWDT
0001010E  4E45                  165.          TRAP #5                     ;GET NEW BLOCK
00010110  4239 00012FC0         166.  NONEWDT CLR.B PGCR                  ;DISABLE PORT A
00010116  4E43                  167.          TRAP #3          ;READY TO EXECUTE
00010118  2057                  168.          MOVEA.L (SP),A0             ;GET PROG START
0001011A  4E90                  169.          JSR (A0)                    ;EXECUTE PROG
0001011C  60 D2                 170.          BRA.S DOADDAT               ;LOAD NEW DATA
0001011E  4239 00012FC0         171.  DONPROG CLR.B PGCR                  ;DISABLE PORT A
00010124  4E4E                  172.          TRAP #14                    ;LOAD NEW PROG
                                173.  *
                                174.  * EXCEPTION ROUTINE ADDRBYT (TRAP #4)
                                175.  * GETS START ADDRESS AND BYTE COUNT OF INCOMING
                                176.  * DATA STREAM, RETURNS ADDRESS IN A0 L, COUNT
                                177.  * D0.L (MAX 64K). ASSUMES INPUT PORT A
                                178.  * CONFIGURED AND ENABLED BEFORE  TRY
                                179.  *
00010126  48E7 6060            180.  ADDRBYT MOVEM.L D1-D2/A1-A2,-(SP) ;SAVE OFF ENV
0001012A  7203                  181.          MOVEQ.L #ADRCONT,D1         ;ADDR BYT COUNT
0001012C  227C 00012FCD         182.          MOVEA.L #PPSR,A1            ;STATUS REG
00010132  247C 00012FC8         183.          MOVEA.L #PADR,A2            ;A DATA REG
00010138  7400                  184.          MOVEQ.L #H1S,D2             ;BIT TO TEST
0001013A  4280                  185.          CLR.L D0
0001013C  E188                  186.  LOOP1A  LSL.L #SHIFT8,D0            ;ADJUST SIGNIF
0001013E  0511                  187.  LOOP1   BTST.B D2,(A1)              ;BYTE READY?
00010140  67 FC                 188.          BEQ.S LOOP1                 ,WAIT TILL SO
00010142  1012                  189.          MOVE.B (A2),D0              ;GET BYTE
00010144  51C9 FFF6             190.          DBRA D1,LOOP1A              ;DO 4 TIMES
00010148  2040                  191.          MOVEA.L D0,A0               ,ADDRESS->A0
0001014A  4280                  192.          CLR.L D0
0001014C  323C 0001             193.          MOVE.W #BYTCONT,D1          ;DATA BYT COUNT
00010150  E188                  194.  LOOP2A  LSL.L #SHIFT8,D0            ;ADJUST SIGNIF
00010152  0511                  195.  LOOP2   BTST.B D2,(A1)              ,BYTE READY?
00010154  67 FC                 196.          BEQ.S LOOP2                 ;WAIT TILL SO
00010156  1012                  197.          MOVE.B (A2),D0              ,GET BYTE
00010158  51C9 FFF6             198.          DBRA D1,LOOP2A              ;DO TWICE
0001015C  4CDF 0606             199.          MOVEM.L (SP)+,D1-D2/A1-A2 ;RESTORE
00010160  4E73                  200.          RTE
                                201.  *
                                202.  * EXCEPTION ROUTINE BLOCKIN (TRAP #5) TO READ
                                203.  * (D0.W)
                                204.  * BYTES OF DATA TO MEMORY STARTING AT (A0.L)
                                205.  * ASSUMES PORT CONFIGURED, ENABLED BEFORE ENTRY
                                206.  * AND BYTE COUNT > 0 ! REGISTERS TRANSPARENT
                                207.  *
00010162  48E7 C0E0            208.  BLOCKIN MOVEM.L D0-D1/A0-A2,-(SP)
00010166  5340                  209.          SUBQ.W #1,D0                ;ADJUST FOR DBRA
00010168  227C 00012FCD         210.          MOVEA.L #PPSR,A1            ;PORT STATUS
0001016E  247C 00012FC8         211.          MOVEA.L #PADR,A2            ;A DATA REG
00010174  7200                  212.          MOVEQ.L #H1S,D1
00010176  0311                  213.  LOOP3   BTST.B D1,(A1)              ;BYTE READY?
00010178  67 FC                 214           BEQ.S LOOP3                 ;WAIT TILL SO
0001017A  10D2                  215           MOVE.B (A2),(A0)+           ;READ & STORE
0001017C  51C8 FFF8             216.          DBRA D0,LOOP3
00010180  4CDF 0703             217.          MOVEM.L (SP)+,D0-D1/A0-A2 ;RESTORE
00010184  4E73                  218.          RTE
                                219.  *
                                220.  * EXCEPTION ROUTINE BLOCKOUT (TRAP #6) TO
                                221.  * OUTPUT DATA TO
                                222.  * OUTPUT PORT B.  BYTE COUNT > 0 IS IN D0.W
                                223.  * START ADDRESS IN A0.L.  ASSUMES PORT B IS
                                224.  * CONFIGURED AND ENABLED. WAITS FOR RECEIVER
                                225.  * READY TO RECEIVE.  REGISTERS TRANSPARENT
```

```
                                226.  * ASSUMES RECEIVER IS "AWARE" WHERE TO PUT
                                227.  * DATA ITSELF.  OTHERWISE, APPEND DEST ADDRESS
                                228.  * TO DATA BLOCK, THEN BYTE COUNT (WORD) OF THE
                                229.  * DATA BLOCK TO BE SENT, THEN DATA BLOCK ITSELF
                                230.  * NOTE THAT BYTE COUNT SENT TO THIS ROUTINE
                                231.  * MUST BE [DATA BYTE COUNT]+6!!, ADDRESS
                                232.  * ADJUSTED TOO!
                                233.  *
00010186  48E7 C0E0            234.  BLOKOUT MOVEM.L D0-D1/A0-A2,-(SP)
0001018A  5340                  235           SUBQ.W #1,D0                ;ADJUST FOR DBRA
0001018C  7206                  236.          MOVEQ.L #H3LEV,D1           ;BIT TO TEST
0001018E  227C 00012FCD         237.          MOVEA.L #PPSR,A1
00010194  247C 00012FC9         238.          MOVEA.L #PBDR,A2
0001019A  0311                  239.  LPREDY   BTST.B D1,(A1)             ;RCVR READY?
0001019C  66 FC                 240.          BNE.S LPREDY                ;WAIT TILL SO
0001019E  7202                  241.          MOVEQ.L #H3S,D1             ;BIT TO TEST
000101A0  0311                  242.  LPOUT    BTST.B D1,(A1)              ;BYTE READY?
000101A2  67 FC                 243.          BEQ.S LPOUT                 ;WAIT TILL SO
000101A4  1498                  244.          MOVE.B (A0)+,(A2)           ;SEND BYTE
000101A6  51C8 FFF8             245.          DBRA D0,LPOUT               ;DO D0 TIMES
000101AA  0311                  246.  LPOUTA   BTST.B D1,(A1)  ;ENSURE LAST BYTE GONE
000101AC  67 FC                 247.          BEQ.S LPOUTA
000101AE  4CDF 0703             248.          MOVEM.L (SP)+,D0-D1/A0-A2 ;RESTORE
000101B2  4E73                  249.          RTE
                                250.  *
                                251.  * EXCEPTION ROUTINE SNADBYT (TRAP #7)
                                252.  * SENDS DEST ADDRESS AND BYTE COUNT OF OUTGOING
                                253.  * DATA. RARELY USED BY SLAVE, BUT HERE FOR
                                254.  * GENERALITY.  ALTERNATIVELY, USE FIX DESCRIBED
                                255.  * IN TRAP6 HANDLER.  DOES A CHECK TO SEE IF
                                256.  * RECEIVER READY (H3LEV=0) BEFORE SENDING
                                257.  * DEST START ADDRESS IN A0.L, BYTE COUNT IN
                                258.  * D0.W.  ALL REGISTERS TRANSPARENT
                                259.  *
000101B4  48E7 E0F0             260.  SNADBYT MOVEM.L D0-D2/A0-A3,-(SP)
000101B8  264F                  261.          MOVEA.L SP,A3    ;KLUDGE BYTE POSTINC
000101BA  7205                  262.          MOVEQ.L #$05,D1             ;BYTE COUNTER
000101BC  7406                  263.          MOVEQ.L #H3LEV,D2
000101BE  227C 00012FCD         264.          MOVE.L #PPSR,A1
000101C4  247C 00012FC9         265.          MOVE.L #PBDR,A2
000101CA  3700                  266.          MOVE.W D0,-(A3) ;USE STACK FOR STORAGE
000101CC  2708                  267.          MOVE.L A0,-(A3) ;THEN PULL OFF IN ORDER
000101CE  0511                  268.  REDYLP   BTST.B D2,(A1)  ;RECEIVER READY?
000101D0  66 FC                 269.          BNE.S REDYLP
000101D2  7402                  270.          MOVEQ.L #H3S,D2 ;SENSE BIT
000101D4  0511                  271.  OUTLP    BTST.B D2,(A1)
000101D6  67 FC                 272.          BEQ.S OUTLP
000101D8  149B                  273.          MOVE.B (A3)+,(A2)
000101DA  51C9 FFF8             274.          DBRA D1,OUTLP               ;SEND 6 BYTES
000101DE  0511                  275.  OUTLPA   BTST.B D2,(A1)  ;ENSURE LAST BYTE GONE
000101E0  67 FC                 276.          BEQ.S OUTLPA
000101E2  4CDF 0F07             277.          MOVEM.L (SP)+,D0-D2/A0-A3 ;RESTORE
000101E6  4E73                  278.          RTE
                                279.  *
                                280.  * TRAP HANDLER ROUTINES
                                281.  *
                                282.  * TRAPS #14,1,2 AND NMI CAN BE IN ROM, TRAP #3
                                283.  * WILL BE COPIED TO RAM FOR MAX PERFORMANCE
                                284.  *
                                285.  * TRAP #3 SERVICE ROUTINE (DUMMY)
                                286.  * ASSERTS AN INTERRUPT AT MASTER VIA PORT C PC0
                                287.  * WAITS FOR ACKNOWLEDGE, ASSUMES PORT C
                                288.  * CONFIGURED, NOT ASSERTING AN IRQ ON ENTRY
                                289.  * REGISTERS TRANSPARENT
                                290.  *
000101E8  48E7 8080             291.  DTRAP3   MOVEM.L D0/A0,-(SP)
000101EC  207C 00012FCC         292.          MOVEA.L #PCDR,A0
000101F2  4280                  293.          CLR.L D0
000101F4  0190                  294.  PULSE    BCLR.B D0,(A0)  ;FORCE PC0 LOW
000101F6  01D0                  295.          BSET.B D0,(A0)  ;FORCE PC0 HI
000101F8  5200                  296.          ADDQ.B #1,D0
000101FA  0110                  297.  WAITAK   BTST.B D0,(A0)  ;CHECK PC1
000101FC  67 FC                 298           BEQ.S WAITAK    ;WAIT TILL HI
                                299.  *
                                300.  * INTERRUPT HAS BEEN ACKNOWLEDGED BY
                                301   * MASTER, CAN RETURN
                                302.  *
000101FE  4CDF 0101             303           MOVEM.L (SP)+,D0/A0
00010202  4E73                  304.          RTE
                                305.  *
```

```
                                306.  * TRAP 14,1,2 ROUTINES, LEAVE IN ROM, NON
                                307.  * CRITICAL PERFORMANCE
                                308.  * TRAP 1 CALLS USERS STATE
                                309.  * TRAP 2 CALL SUP STATE
                                310.  * TRAP 14 PUTS SLAVE IN PROGRAM ACCEPT MODE
                                311.  * WITH A CLEAN STACK. CAN BE USED IN SLAVE
                                312.  * PROGRAMS TO KILL A PROGRAM AND DOWNLOAD
                                313.  * ANOTHER, OR A SOFTWARE NMI FROM MASTER!
                                314.  * USES TRAP #14 TO REDUCE CONFUSION BETWEEN
                                315.  * MASTER'S TRAP #0
                                316.  * ALSO NMI ROUTINE HERE, RESET STATUS
                                317.  * OF PORTS IN CASE SOME DATA HELD
                                318.  *
                                319.  * MAY 1,1991 CHECK STATUS OF PC1 IF HI,
                                320.  * DO NOTHING, IF LOW, MUST STROBE
                                321.  *
00010204  207C 00012FCC          322.  TRAP14  MOVEA.L #PCDR,A0
0001020A  0810 0001             323.          BTST.B #1,(A0)
0001020E  66 0C                 324.          BNE.S NOSTRB
00010210  08D0 0000             325.          BSET.B #0,(A0)   ,ENSURE A NEGATIVE
00010214  0890 0000             326.          BCLR.B #0,(A0)   ;EDGE
00010218  08D0 0000             327.          BSET.B #0,(A0)
0001021C  46FC 2000             328.  NOSTRB  MOVE.W #CLERCCR,SR  ;CCR=0,NOTRACE,SUP
00010220  2E7C 0000FED0         329.          MOVEA.L #INITSSP,SP ;NEW STACK POINTER
00010226  4239 00012FC0         330.          CLR.B PGCR              ;DISABLE PORTS
0001022C  13FC 00FF             331.          MOVE.B #RESSTAT,PPSR    ;RESET STATUS
          00012FCD
00010234  6000 FEA0             332.          BRA NEWPROG          ;CAN'T DO RTE,
                                333.  *                            ;NO STACK!
                                334.  *
                                335.  *
00010238  0257 DFFF             336.  TRAP1   ANDI.W #CLRUS,(SP)       ,CLEAR USER BIT
0001023C  4E73                  337.          RTE
0001023E  0057 2000             338.  TRAP2   ORI.W #SETSUP,(SP)       ;SET SUPER BIT
00010242  4E73                  339.          RTE
                                340.  *
                                341.  * AREA FOR ROUTINES
                                342.  *
0000FEE0                        343.          ORG.L $10000-$120
0000FEE0  <36>                  344.  LOADDAT DS.B ADDRBYT-DOADDAT
0000FF16  <3C>                  345.  TRAP4   DS.B BLOCKIN-ADDRBYT
0000FF52  <24>                  346.  TRAP5   DS.B BLOKOUT-BLOCKIN
0000FF76  <2E>                  347.  TRAP6   DS.B SNADBYT-BLOKOUT
0000FFA4  <34>                  348.  TRAP7   DS.B DTRAP3-SNADBYT
0000FFD8  <1C>                  349.  TRAP3   DS.B TRAP14-DTRAP3
0000FFF4                        350.          END
    0 Errors
```

Appendix J: Data Transfer Rate Measurement Program

```
                                    1.          TTL tfrtim
                                    2.  *
                                    3.  * THIS PROGRAM measures data transfer rates
                                    4.  * for blocks to and from each slave, and to
                                    5.  * all slaves in broadcast mode
                                    6.  *
                                    7.  *
                                    8.  *
                                    9.  * EQUATES
     # 00012FCC                     10.  PCDR EQU $12FCC
     # 00012FC0                     11.  PGCR EQU $12FC0
     # 00000010                     12.  ENABLEA EQU $10
     # 00000020                     13.  ENABLEB EQU $20
     # 00000003                     14.  SRQASRT EQU 3
     # 00000004                     15.  ADBYTIN EQU 4
     # 00000007                     16.  ADBYTOT EQU 7
     # 00000005                     17.  INDATA  EQU 5
     # 00000006                     18.  OUTDATA EQU 6
     # 00000008                     19.  NETCNF  EQU 8
     # 00000009                     20.  SRQSRVC EQU 9
     # 0000000E                     21.  ABORT   EQU 14
     # 00000001                     22.  SLV0    EQU $01
     # 00000002                     23.  SLV1    EQU $02
     # 00000004                     24.  SLV2    EQU $04
     # 00000008                     25.  SLV3    EQU $08
     # 0000000F                     26.  ALLSLV  EQU $0F
     # 00012FB0                     27.  PSR01   EQU $12FB0
     # 00012FB1                     28.  PSR23   EQU $12FB1
     # 00012FB2                     29.  PSR45   EQU $12FB2
     # 00012FB3                     30.  PSR67   EQU $12FB3
     # 00012FB4                     31.  PSR89   EQU $12FB4
     # 00012FB5                     32.  PSR1011 EQU $12FB5
     # 00012FB6                     33.  PSR1213 EQU $12FB6
     # 00012FB7                     34.  PSR1415 EQU $12FB7
     # 00012FB8                     35.  PSR1617 EQU $12FB8
     # 00012FB9                     36.  PSR1819 EQU $12FB9
     # 00012FD0                     37.  TCR     EQU $12FD0
     # 00012FD2                     38.  TPLR    EQU TCR+2
     # 00012FD6                     39.  TCNTR   EQU TCR+6
     # 0000000A                     40.  SENCLFC EQU 10
     # 00000018                     41.  DISBUF8 EQU 24
     # 00000016                     42.  DISBUF2 EQU 22
     # 00FFFFFF                     43.  CNTSTRT EQU $00FFFFFF
     # 00000001                     44.  ENABLTM EQU 1
     # 00000011                     45.  OUTMESC EQU 17
     # 00000100                     46.  PASSCNT EQU 256
     # 00000008                     47.  DIV256  EQU 8
     # 00000004                     48.  EOT     EQU 4
     # 00000007                     49.  CNTLEN  EQU 7
     # 0000000D                     50.  INCHCON EQU 13
                                    51.  *
                                    52.  ******************
                                    53.  * MASTER PROGRAM *
                                    54.  ******************
                                    55.  *
00001000                            56.       ORG.L $1000
00001000  <400>                   57.  DUMTAB  DS.B $400
00001400  0008 0010 0020           58.  CNTABL       DC.W $8,$10,$20,$40,$80,$100,$200
          0040 0080 0100
          0200
0000140E  BB BB 00 00 00           59.  CONOUT  DC.B $BB,$BB,$00,$00,$00,$00,$00,$00,$00,$00
          00 00 00 00 00
00001418  04 00 BB 4B 00           60.          DC.B $04,$00,$BB,$4B,$00,$00,$00,$00,$00,$00
          00 00 00 00 00
00001422  04 00 04 00 EB           61.          DC.B $04,$00,$04,$00,$BB,$04,$00,$00,$00,$00
          04 00 00 00 00
0000142C  04 00 04 00 04           62.          DC.B $04,$00,$04,$00,$04,$00,$4B,$00,$00,$00
          00 4B 00 00 00
00001436  00 00 00 80 00           63.  CONIN   DC.B $00,$00,$00,$80,$00,$80,$00,$80,$00,$80
          80 00 80 00 80
00001440  00 00 00 00 00           64.          DC.B $00,$00,$00,$00,$00,$B8,$00,$80,$00,$80
          B8 00 80 00 80
0000144A  00 00 00 00 00           65.          DC.B $00,$00,$00,$00,$00,$00,$80,$BB,$00,$80
          00 80 BB 00 80
00001454  00 00 00 00 00           66.          DC.B $00,$00,$00,$00,$00,$00,$00,$00,$B8,$BB
          00 00 00 B8 BB
0000145E  BD BB BD CB BD           67.  BRDCAST DC.B $BD,$BB,$BD,$CB,$BD,$8C,$CB,$8B,$B8,$8B
          8C CB 8B B8 8B
                                    68   *
                                    69   * CONFIGURE NETWORK TO BROADCAST STATE, ALL PROGRAMS
```

```
                                70.  * EXACTLY THE SAME!
                                71.  * RESPOND TO ALL SLAVE'S RFP SRQ, AND LOAD
                                72.  * THE SLAVE PROGRAM DOWN
                                73.  *
00001468  41FA FFF4             74.  STRTTFR LEA BRDCAST(PC),A0
0000146C  4E48                  75.          TRAP #NETCNF
0000146E  103C 000F             76.          MOVE.B #ALLSLV,D0
00001472  4E49                  77.          TRAP #SRQSRVC
00001474  41FA 02BC             78.          LEA SLVPRO(PC),A0        ;STRT ADDR
00001478  43FA 0326             79.          LEA ENDSLV(PC),A1        ;END ADDR+1
0000147C  93C8                  80.          SUBA.L A0,A1
0000147E  3009                  81.          MOVE.W A1,D0
00001480  13FC 0020             82.          MOVE.B #ENABLEB,PGCR
          00012FC0
00001488  4E47                  83.          TRAP #ADBYTOT
0000148A  4E46                  84.          TRAP #OUTDATA
0000148C  4239 00012FC0         85.          CLR.B PGCR
                                86.  *
                                87.  * ALL PROGRAMS LOADED, EACH ASSERTING READY FOR
                                88.  * DATA SRQ.  RESPOND TO ALL, THEN DOWNLOAD THE
                                89.  * CNTABL DATA IN BROADCAST MODE.
                                90.  *
00001492  41FA FFCA             91.          LEA BRDCAST(PC),A0
00001496  4E48                  92.          TRAP #NETCNF             ;CONFIG BRDCAST
00001498  103C 000F             93.          MOVE.B #ALLSLV,D0
0000149C  4E49                  94.          TRAP #SRQSRVC
0000149E  41FA FF60             95.          LEA CNTABL(PC),A0        ;START OF DATA
000014A2  303C 000E             96.          MOVE.W #CNTLEN*2,D0      ;D0 HAS BYTE COUNT
                                97.  *
                                98.  * SEND START ADDRESS AND BYTE COUNT TO SLAVES
                                99.  * AND THE CNTABL DATA
                               100.  *
000014A6  13FC 0020             101.          MOVE.B #ENABLEB,PGCR
          00012FC0
000014AE  4E47                  102.          TRAP #ADBYTOT    ;ADDRESS AND BYTE COUNT
000014B0  4E46                  103.          TRAP #OUTDATA    ;SEND OFF THE CNTABL
000014B2  4239 00012FC0         104.          CLR.B PGCR       ;
                                105.  *
                                106.  * RESPOND TO THE SLAVE rte SRQ
                                107.  *
000014B8  103C 000F             108.          MOVE.B #ALLSLV,D0
000014BC  4E49                  109.          TRAP #SRQSRVC
                                110.  *
                                111.  * NOW PERFORM THE DOWNLOADING AND UPLOADING
                                112.  * OF THE DATA, AND MEASURE TRANSFER TIMES
                                113.  *
000014BE  3E3C 0006             114.          MOVE.W #CNTLEN-1,D7      ;OUTR COUNTER
000014C2  43FA FF3C             115.          LEA CNTABL(PC),A1        ;POINT TO COUNT
000014C6  3C3C 0003             116.  OUTRLP1 MOVE.W #3,D6             ;INNER COUNTER
000014CA  3F3C 0001             117.          MOVE.W #SLV0,-(SP)       ;SLV MASK
000014CE  3F3C 0000             118.          MOVE.W #0,-(SP)          ;SLV NUMBER
000014D2  302F 0002             119.  INRLP1  MOVE.W 2(SP),D0          ;SLV MASK
000014D6  4E49                  120.          TRAP #SRQSRVC            ;RESPOND SRQ
000014D8  3017                  121.          MOVE.W (SP),D0           ;SLAVE # IN D0
000014DA  41FA FF32             122.          LEA CONOUT(PC),A0        ;PNT TO NET CODE
000014DE  6100 00B8             123.          BSR SETNET               ;CONFIG NET
000014E2  3011                  124.          MOVE.W (A1),D0           ;GET COUNT
000014E4  41FA FB1A             125.          LEA DUMTAB(PC),A0        ;DUMMY DATA
000014E8  6100 00B8             126.          BSR SENDOUT              ;SEND DATA
000014EC  E1EF 0002             127.          ASL.W 2(SP)              ;ADJUST MASK
000014F0  5257                  128.          ADD.W #1,(SP)            ;ADJUST SLAVE #
000014F2  51CE FFDE             129.          DBRA D6,INRLP1   ;SAME COUNT EACH SLV
000014F6  D3FC 00000002         130.          ADDA.L #2,A1     ;PNT NEXT COUNT
000014FC  201F                  131.          MOVE.L (SP)+,D0 ;CLEAN STACK
000014FE  51CF FFC6             132.          DBRA D7,OUTRLP1  ; REPEAT EACH COUNT
00001502  4E4F                  133.          TRAP #15
00001504  000A                  134.          DC.W SENCLFC
00001506  41FA 020F             135.          LEA KEYMES(PC),A0
0000150A  4E4F                  136.          TRAP #15
0000150C  0011                  137.          DC.W OUTMESC
0000150E  4E4F                  138.          TRAP #15
00001510  000D                  139.          DC.W INCHCON
00001512  4E4F                  140.          TRAP #15
00001514  000A                  141           DC.W SENCLFC
                                142.  *
                                143.  * DO AGAIN FOR SLAVE TO MASTER
                                144.  *
00001516  3E3C 0006             145.          MOVE.W #CNTLEN-1,D7      ;OUTR COUNTER
0000151A  43FA FEE4             146.          LEA CNTABL(PC),A1        ;POINT TO COUNT
0000151E  3C3C 0003             147.  OUTRLP2 MOVE.W #3,D6             ;INNER COUNTER
```

```
00001522  3F3C 0001       148.          MOVE.W #SLV0,-(SP)       ;SLV MASK
00001526  3F3C 0000       149.          MOVE.W #0,-(SP)          ;SLV NUMBER
0000152A  302F 0002       150.  INRLP2  MOVE.W 2(SP),D0          ;SLV MASK
0000152E  4E49            151.          TRAP #SRQSRVC            ;RESPOND SRQ
00001530  3017            152.          MOVE.W (SP),D0           ;SLAVE # IN D0
00001532  41FA FF02       153.          LEA CONIN(PC),A0         ;PNT TO NET CODE
00001536  6100 0060       154.          BSR SETNET               ;CONFIG NET
0000153A  3011            155.          MOVE.W (A1),D0           ;GET COUNT
0000153C  41FA FAC2       156.          LEA DUMTAB(PC),A0        ;DUMMY DATA
00001540  6100 00D0       157.          BSR RECIN                ;SEND DATA
00001544  E1EF 0002       158.          ASL.W 2(SP)
00001548  5257            159.          ADD.W #1,(SP)            ;ADJUST SLAVE #
0000154A  51CE FFDE       160.          DBRA D6,INRLP2  ;SAME COUNT EACH SLV
0000154E  D3FC 00000002   161.          ADDA.L #2,A1    ;PNT NEXT COUNT
00001554  201F            162.          MOVE.L (SP)+,D0 ;CLEAN STACK
00001556  51CF FFC6       163.          DBRA D7,OUTRLP2  ; REPEAT EACH COUNT
                          164.  *
                          165.  * DO AGAIN FOR BROADCAST MODE
                          166.  *
0000155A  4E4F            167.          TRAP #15
0000155C  000A            168.          DC.W SENCLFC
0000155E  41FA 01B7       169.          LEA KEYMES(PC),A0
00001562  4E4F            170.          TRAP #15
00001564  0011            171.          DC.W OUTMESC
00001566  4E4F            172.          TRAP #15
00001568  000D            173.          DC.W INCHCON
0000156A  4E4F            174.          TRAP #15
0000156C  000A            175.          DC.W SENCLFC
0000156E  41FA FEEE       176.          LEA BRDCAST(PC),A0
00001572  4E48            177.          TRAP #NETCNF
00001574  3E3C 0006       178.          MOVE.W #CNTLEN-1,D7      ;OUTR COUNTER
00001578  43FA FE86       179.          LEA CNTABL(PC),A1        ;POINT TO COUNT
0000157C  103C 000F       180.  OUTRLP3 MOVE.B #ALLSLV,D0        ,SLV MASK
00001580  4E49            181.          TRAP #SRQSRVC            ,RESPOND SRQ
00001582  3011            182.          MOVE.W (A1),D0           ;GET COUNT
00001584  41FA FA7A       183.          LEA DUMTAB(PC),A0        ;DUMMY DATA
00001588  6100 00F8       184.          BSR SENDBRD              ,SEND DATA
0000158C  D3FC 000C0002   185.          ADDA.L #2,A1     ,PNT NEXT COUNT
00001592  51CF FFE8       186.          DBRA D7,OUTRLP3  ; REPEAT EACH COUNT
                          187.  *
                          188.  * DONE
                          189.  *
00001596  4E75            190.          RTS
                          191.  *
                          192.  * SUBROUTINE SETNET
                          193.  * SLAVE NUMBER IS IN D0, A0 POINTS TO CONIN OR
                          194.  * CONOUT
00001598  C0FC 000A       195.  SETNET  MULU #10,D0
0000159C  D1C0            196.          ADDA.L D0,A0
0000159E  4E48            197.          TRAP #NETCNF
000015A0  4E75            198.          RTS
                          199.  *
                          200.  * SUBROUTINE SENDOUT
                          201.  * BYTE COUNT IN D0, ADDRESS TO SEND IN A0
                          202.  * SLAVE # AT 4(SP) BEFORE ENTRY!
                          203.  * REPORTS SLV #, COUNT, AND TIME TO TERMINAL
                          204.  *
000015A2  3F00            205.  SENDOUT MOVE.W D0,-(SP) ;SAVE COUNT
000015A4  4239 00012FD0   206.          CLR.B TCR
000015AA  23FC 00FFFFFF   207.          MOVE.L #CNTSTRT,TCR+2
          00012FD2
000015B4  13FC 0001       208.          MOVE.B #ENABLTM,TCR ;COUNT HAS BEGUN
          00012FD0
000015BC  13FC 0020       209.          MOVE.B #ENABLEB,PGCR
          00012FC0
000015C4  4E46            210.          TRAP #OUTDATA
000015C6  4239 00012FC0   211.          CLR.B PGCR
000015CC  4239 00012FD0   212.          CLR.B TCR        ;STOP TIMER
                          213.  *
                          214.  * NOW OUTPUT THE MESSAGES
                          215.  *
000015D2  41FA 010E       216.          LEA MES1(PC),A0
000015D6  4E4F            217.          TRAP #15
000015D8  0011            218.          DC.W OUTMESC
000015DA  4280            219.          CLR.L D0
000015DC  301F            220.          MOVE.W (SP)+,D0 ,GET COUNT
000015DE  4E4F            221.          TRAP #15
000015E0  0018            222.          DC.W DISBUFB
000015E2  41FA 0106       223.          LEA MES2(PC),A0
000015E6  4E4F            224.          TRAP #15
```

```
000015E8  0011              225.          DC.W OUTMESC
000015EA  302F 0004         226.          MOVE.W 4(SP),D0 ;GET SLAVE #
000015EE  4E4F              227.          TRAP #15
000015F0  0016              228.          DC.W DISBUF2
                            229.  *
                            230.  * CALCULATE TIME
                            231.  *
000015F2  41FA 0106         232.          LEA MES3(PC),A0
000015F6  4E4F              233.          TRAP #15
000015F8  0011              234.          DC.W OUTMESC
000015FA  2239 00012FD6     235.          MOVE.L TCR+6,D1
00001600  203C 00FFFFFF     236.          MOVE.L #CNTSTRT,D0
00001606  9081              237.          SUB.L D1,D0
00001608  4E4F              238.          TRAP #15
0000160A  0018              239.          DC.W DISBUF8
0000160C  4E4F              240.          TRAP #15
0000160E  000A              241.          DC.W SENCLFC
00001610  4E75              242.          RTS
                            243.  *
                            244.  * SUBROUTINE RECIN
                            245.  * BYTE COUNT IN D0, ADDRESS IN A0
                            246.  * RECEIVE BYTE AND MEASURE AND REPORT
                            247.  * TIME
                            248.  *
00001612  3F00              249.  RECIN   MOVE.W D0,-(SP) ;SAVE COUNT
00001614  4239 00012FD0     250.          CLR.B TCR
0000161A  23FC 00FFFFFF     251.          MOVE.L #CNTSTRT,TCR+2
          00012FD2
00001624  13FC 0001         252.          MOVE.B #ENABLTM,TCR ;COUNT HAS BEGUN
          00012FD0
0000162C  13FC 0010         253.          MOVE.B #ENABLEA,PGCR
          00012FC0
00001634  4E45              254.          TRAP #INDATA
00001636  4239 00012FC0     255.          CLR.B PGCR
0000163C  4239 00012FD0     256           CLR.B TCR       ;STOP TIMER
                            257.  *
                            258.  * NOW OUTPUT THE MESSAGES
                            259.  *
00001642  41FA 009E         260.          LEA MES1(PC),A0
00001646  4E4F              261.          TRAP #15
00001648  0011              262.          DC.W OUTMESC
0000164A  4280              263.          CLR.L D0
0000164C  301F              264.          MOVE.W (SP)+,D0 ;GET COUNT
0000164E  4E4F              265.          TRAP #15
00001650  0018              266.          DC.W DISBUF8
00001652  41FA 0096         267.          LEA MES2(PC),A0
00001656  4E4F              268.          TRAP #15
00001658  0011              269.          DC.W OUTMESC
0000165A  302F 0004         270.          MOVE.W 4(SP),D0 ;GET SLAVE #
0000165E  4E4F              271.          TRAP #15
00001660  0016              272.          DC.W DISBUF2
                            273.  *
                            274.  * CALCULATE TIME
                            275.  *
00001662  41FA 0096         276.          LEA MES3(PC),A0
00001666  4E4F              277.          TRAP #15
00001668  0011              278.          DC.W OUTMESC
0000166A  2239 00012FD6     279.          MOVE.L TCR+6,D1
00001670  203C 00FFFFFF     280.          MOVE.L #CNTSTRT,D0
00001676  9081              281.          SUB.L D1,D0
00001678  4E4F              282.          TRAP #15
0000167A  0018              283.          DC.W DISBUF8
0000167C  4E4F              284.          TRAP #15
0000167E  000A              285.          DC.W SENCLFC
00001680  4E75              286.          RTS
                            287.  *
                            288.  * SUBROUTINE SENDBRD
                            289.  * BYTE COUNT IN D0, ADDRESS IN A0
                            290.  * SEND OUT DATA AND REPORT TIMES
                            291.  *
00001682  3F00              292.  SENDBRD MOVE.W D0,-(SP) ;SAVE COUNT
00001684  4239 00012FD0     293.          CLR.B TCR
0000168A  23FC 00FFFFFF     294.          MOVE.L #CNTSTRT,TCR+2
          00012FD2
00001694  13FC 0001         295.          MOVE.B #ENABLTM,TCR ;COUNT HAS BEGUN
          00012FD0
0000169C  13FC 0020         296.          MOVE.B #ENABLEB,PGCR
          00012FC0
000016A4  4E46              297.          TRAP #OUTDATA
000016A6  4239 00012FC0     298           CLR.B PGCR
```

```
000016AC  4239 00012FD0   299.          CLR.B TCR       ;STOP TIMER
                          300.   *
                          301.   * NOW OUTPUT THE MESSAGES
                          302.   *
000016B2  41FA 002E       303.          LEA MES1(PC),A0
000016B6  4E4F            304.          TRAP #15
000016B8  0011            305.          DC.W OUTMESC
000016BA  4280            306.          CLR.L D0
000016BC  301F            307.          MOVE.W (SP)+,D0 ;GET COUNT
000016BE  4E4F            308.          TRAP #15
000016C0  0018            309.          DC.W DISBUF8
                          310.   *
                          311.   * CALCULATE TIME
                          312.   *
000016C2  41FA 0041       313.          LEA MESBD(PC),A0
000016C6  4E4F            314.          TRAP #15
000016C8  0011            315.          DC.W OUTMESC
000016CA  2239 00012FD6   316.          MOVE.L TCR+6,D1
000016D0  203C 00FFFFFF   317.          MOVE.L #CNTSTRT,D0
000016D6  9081            318.          SUB.L D1,D0
000016D8  4E4F            319.          TRAP #15
000016DA  0018            320.          DC.W DISBUF8
000016DC  4E4F            321.          TRAP #15
000016DE  000A            322.          DC.W SENCLFC
000016E0  4E75            323.          RTS
                          324.   *
                          325.   * MESSAGES
                          326.   *
000016E2  43 6F 75 6E 74  327.   MES1     DC.B 'Count= ',EOT
          3D 20 04
000016EA  20 53 6C 61 76  328.   MES2     DC.B ' Slave number= ',EOT
          65 20 6E 75 6D
          62 65 72 3D 20
          04
000016FA  20 20 20 54 69  329.   MES3 DC.B '   Time = ',EOT
          6D 65 20 3D 20
          04
00001705  20 42 72 6F 61  330.   MESBD    DC.B ' Broadcast time= ',EOT
          64 63 61 73 74
          20 74 69 6D 65
          3D 20 04
00001717  53 74 72 69 6B  331.   KEYMES   DC.B 'Strike any key to continue',EOT
          65 20 61 6E 79
          20 6B 65 79 20
          74 6F 20 63 6F
          6E 74 69 6E 75
          65 04
                          332.   *
                          333.   ****************************************
                          334.   * SLAVE PROGRAM
                          335.   ****************************************
00001732  3E3C 0006       336.   SLVPRO   MOVE.W #CNTLEN-1,D7
00001736  43FA FCC8       337.            LEA CNTABL(PC),A1
0000173A  3019            338.   LOOP1    MOVE.W (A1)+,D0          ;COUNT
0000173C  41FA F8C2       339.            LEA DUMTAB(PC),A0        ,DEST
00001740  4E43            340.            TRAP #SRQASRT            ,SRQ
00001742  13FC 0010       341.            MOVE.B #ENABLEA,PGCR
          00012FC0
0000174A  4E45            342.            TRAP #INDATA
0000174C  4239 00012FC0   343.            CLR.B PGCR
00001752  51CF FFE6       344.            DBRA D7,LOOP1
00001756  3E3C 0006       345.            MOVE.W #CNTLEN-1,D7
0000175A  43FA FCA4       346.            LEA CNTABL(PC),A1
0000175E  3019            347.   LOOP2    MOVE.W (A1)+,D0          ,COUNT
00001760  41FA F89E       348.            LEA DUMTAB(PC),A0        ,DEST
00001764  4E43            349.            TRAP #SRQASRT            ;SRQ
00001766  13FC 0020       350.            MOVE.B #ENABLEB,PGCR
          00012FC0
0000176E  4E46            351.            TRAP #OUTDATA
00001770  4239 00012FC0   352.            CLR B PGCR
00001776  51CF FFE6       353.            DBRA D7,LOOP2
                          354.   *
                          355.   * AND ONCE MORE FOR BROADCASTING
                          356.   * INPUT
0000177A  3E3C 0006       357.            MOVE W #CNTLEN-1,D7
0000177E  43FA FC80       358.            LEA CNTABL(PC),A1
00001782  3019            359.   LOOP3    MOVE.W (A1)+,D0          ,COUNT
00001784  41FA F87A       360             LEA DUMTAB(PC),A0        ,DEST
00001788  4E43            361.            TRAP #SRQASRT            ,SRQ
0000178A  13FC 0010       362             MOVE.B #ENABLEA,PGCR
```

```
          00012FC0
00001792  4E45              363.          TRAP #INDATA
00001794  4239 00012FC0     364.          CLR.B PGCR
0000179A  51CF FFE6         365.          DBRA D7,LOOP3
0000179E  4E4E              366.          TRAP #ABORT
                            367.  *
                            368.  * DONE!
                            369.  *
000017A0  <1>               370.  ENDSLV  DS.B 1
000017A1                    371.          END
    0 Errors
```

A Reconfigurable Multicomputer System: Implementation and Performance

Volume II

by

Paul Anthony Smeulders

Faculty of Engineering Science
Department of Electrical Engineering

Submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy

Faculty of Graduate Studies
The University of Western Ontario
London, Ontario
July 1992

Table of Contents (Vol. II)

Appendix K: Matrix Multiplication Test Programs and Input Data

K.1 Program MATFP1: Uniprocessor Matrix Multiplication

```
                                    1.          TTL MATFP1
                                    2.  * MATRIX MULTIPLY PROGRAM FOR FLOATING POINT
                                    3.  * INPUT AND OUTPUT DATA.  CONSISTS OF A
                                    4.  * CONTROL ROUTINE FOR USER INTERFACE, AND THE
                                    5.  * ACTUAL MATRIX MULTIPLY ROUTINE
                                    6.  * DOES A x B = C FOR N x N MATRICES
                                    7.  * MARCH 2,1992
                                    8.  *
      # 0000000D                    9.  CR       EQU $0D
      # 0000000A                   10.  LF       EQU $0A
      # 00000004                   11.  EOT      EQU $04
      # 00000011                   12.  OUTMESC  EQU 17
      # 0000001C                   13.  BLDNUM   EQU 28
      # 00012FD0                   14.  TCR      EQU $12FD0
      # 00000018                   15.  DISBUF8  EQU $18
      # 0000000A                   16.  SENCLFC  EQU $A
      # 00FFFFFF                   17.  CNTSTRT  EQU $00FFFFFF
      # 000000FF                   18.  BGEXP    EQU $FF
      # 0000001F                   19   SGNMSK   EQU 31
      # 00000080                   20.  BIAS     EQU 128
      # 00000009                   21.  ALSB     EQU 9
      # 00000008                   22.  ANTLSB   EQU 8
      # 00000020                   23.  ARNDDR   EQU $20
      # 80000000                   24.  ABSGN    EQU $80000000
      # 00000017                   25.  SIGBITS  EQU 23
      # 0000001D                   26.  NORMTST  EQU 29
      # 00000007                   27.  MLSB     EQU 7
      # 00000006                   28.  MNTLSB   EQU 6
      # 00000080                   29   MRNDDR   EQU $80
      # 00000001                   30.  ENABLTM  EQU 01
                                   31   *
00001000                           32           ORG.L $1000
00001000  4E4F                     33   MATWORD TRAP #15
00001002  000A                     34.          DC.W SENCLFC
00001004  41FA 00F0                35.          LEA TITLE(PC),A0          ,TITLE PROMPT
00001008  4E4F                     36           TRAP #15                  ,SERVICE
0000100A  0011                     37           DC.W OUTMESC
0000100C  41FA 0122                38.          LEA ADR1(PC),A0           ,GET ADDR
00001010  6100 0028                39.          BSR GETCHK                ,FOR A(0,0)
00001014  2840                     40.          MOVE.L D0,A4              ,HOLD TEMP
00001016  41FA 0133                41.          LEA ADR2(PC),A0           ,GET ADDR
0000101A  6100 001E                42.          BSR GETCHK                ,FOR B(0,0)
0000101E  2240                     43           MOVE.L D0,A1              ,HOLD ADDRESS
00001020  41FA 0146                44           LEA ADR3(PC),A0           ,GET ADR
00001024  6100 0014                45           BSR GETCHK                ,FOR C(0,0)
00001028  2440                     46           MOVE.L D0,A2
0000102A  41FA 0159                47.          LEA DIMEN(PC),A0          ,GET N
0000102E  6100 000A                48           BSR GETCHK                ,N IN D0.L
00001032  204C                     49.          MOVE.L A4,A0              ,SET UP FOR MAT
00001034  6100 0036                50.          BSR MATRIX
00001038  4E75                     51           RTS                       ,DONE
                                   52.  *
                                   53   * SUBROUTINE GETCHK
                                   54   * OUTPUTS PROMPT, ACQUIRES HEX DATA, DOES ERROR
                                   55.  * CHECK, REPEAT IF NECESSARY, RETURNS INPUT
                                   56.  * DATA IN D0.L
                                   57   *
0000103A  48E7 4080                58.  GETCHK  MOVEM.L D1/A0,-(SP)       ,WORK SPACE
0000103E  4281                     59   REDO    CLR.L D1
00001040  4E4F                     60           TRAP #15                  ,SEND PROMPT
00001042  0011                     61           DC.W OUTMESC
00001044  4E4F                     62.          TRAP #15                  ,CALL BLDNUM
00001046  001C                     63.          DC.W BLDNUM
00001048  3208                     64.          MOVE.W A0,D1
0000104A  B23C 000D                65.          CMP.B #CR,D1              ,VALID DELIMTER?
0000104E  66 0C                    66           BNE.S ERROR
00001050  E089                     67           LSR.L #8,D1
00001052  4A81                     68           TST.L D1
00001054  66 06                    69           BNE.S ERROP
00001056  4CDF 0102                70           MOVEM.L (SP)+,A0/D1       ,RESTORE
0000105A  4E75                     71           RTS
0000105C  41F9 0000119C            72   ERROP   LEA ERRMES,A0             ,ERROR MESSAGE
00001062  4E4F                     73           TRAP #15
00001064  0011                     74           DC.W OUTMESC
00001066  206F 0004                75           MOVEA.L 4(SP),A0 ,POINT TO OLD PROMPT
0000106A  60 D2                    76           BRA.S REDO
                                   77   *
                                   78   * SUBROUTINE FOR N x N MATRIX MULTIPLY
```

```
                                    79.  * [A] x [B] = C
                                    80   * FLOATING POINT INPUT AND OUTPUT
                                    81.  * DATA ASSUMED IN MEMORY
                                    82.  * A0 = ADDRESS OF A(0,0)
                                    83.  * A1 = ADDRESS OF B(0,0)
                                    84.  * A2 = ADDRESS OF C(0,0)
                                    85.  * D0 = N
                                    86.  *
                                    87.  * SET UP COUNTERS AND INDICES
0000106C  4239 00012FD0             88.  MATRIX  CLR.B TCR
00001072  23FC 00FFFFFF             89.          MOVE.L #CNTSTRT,TCR+2
          00012FD2
0000107C  13FC 0001                 90.          MOVE.B #ENABLTM,TCR ;ENABLE TIMER
          00012FD0
00001084  3C00                      91.          MOVE.W D0,D6    ;D6=N
00001086  3406                      92.          MOVE.W D6,D2   ;D2=N
00001088  C4C2                      93.          MULU D2,D2     ;D2=N**2
0000108A  E582                      94.          ASL.L #2,D2    ;D2=4*N**2
0000108C  2A02                      95.          MOVE.L D2,D5   ;D5=4*N**2
0000108E  5982                      96.          SUBQ.L #4,D2   ;D2=4*N**2-4
00001090  2A42                      97.          MOVE.L D2,A5   ;A5=4*N**2-4
00001092  264D                      98.          MOVEA.L A5,A3  ;HOLD FOR LATER
00001094  2606                      99.          MOVE.L D6,D3
00001096  E583                     100.          ASL.L #2,D3    ;D3=4*N
00001098  5386                     101.          SUBQ.L #1,D6
0000109A  2E06                     102.          MOVE.L D6,D7   ;D7=N-1
0000109C  2846                     103.          MOVEA.L D6,A4  ;SAVE FOR LATER
0000109E  D5C5                     104.          ADDA.L D5,A2   ;A2 IS ADDR OF C(N,N)
                                   105.  *
                                   106.  * NOW DO THE MULTIPLY
                                   107.  *
000010A0  4285                     108   OUTRLP  CLR.L D5       ;HOLDS RESULT OF ADDS
000010A2  2802                     109           MOVE.L D2,D4
000010A4  2C4D                     110           MOVE.L A5,A6
000010A6  2030 4800                111.  INRLP   MOVE.L 0(A0,D4.L),D0    ;GET A(I,J)
000010AA  2231 E800                112.          MOVE.L 0(A1,A6.L),D1    ;DO MUL B(J,K)
000010AE  6100 01F6                113           BSR MULTFP              ;RESULT IN D0
000010B2  2205                     114.          MOVE.L D5,D1
000010B4  6100 010E                115.          BSR ADDFP
000010B8  2A00                     116.          MOVE.L D0,D5
000010BA  5984                     117           SUBQ.L #4,D4          ;D4=D4-4
000010BC  9DC3                     118.          SUBA.L D3,A6          ;D5=D5-D3
000010BE  51CE FFE6                119.          DBRA D6,INRLP         ;DO N TIMES
000010C2  2505                     120.          MOVE.L D5,-(A2)       ;D0=N-1 AGAIN
000010C4  2C0C                     121.          MOVE.L A4,D6
000010C6  9BFC 00000004            122           SUBA.L #4,A5
000010CC  51CF FFD2                123.          DBRA D7,OUTRLP        ;DO N TIMES
000010D0  2A4B                     124           MOVE.L A3,A5          ;A5=2*N**2-2
000010D2  2E0C                     125.          MOVE.L A4,D7          ;D7=N-1
000010D4  9483                     126.          SUB.L D3,D2           ;D2=D2-D3
000010D6  6C C8                    127.          BGE.S OUTRLP
000010D8  4239 00012FD0            128.          CLR.B TCR       ,DISABLE TIMER
000010DE  2239 00012FD6            129.          MOVE.L TCR+6,D1
000010E4  203C 00FFFFFF            130.          MOVE.L #CNTSTRT,D0
000010EA  9081                     131.          SUB.L D1,D0
000010EC  4E4F                     132.          TRAP #15
000010EE  000A                     133.          DC.W SENCLFC
000010F0  4E4F                     134.          TRAP #15
000010F2  0018                     135.          DC.W DISBUF8
000010F4  4E75                     136.          RTS                   ;AND RETURN
                                   137.  *
                                   138.  * DATA SECTION
                                   139   *
000010F6  0A 0D 4E 20 78           140   TITLE   DC.B LF,CR,'N x N Matrix Multiply Progr'
          20 4E 20 4D 61
          74 72 69 78 20
          4D 75 6C 74 69
          70 6C 79 20 50
          72 6F 67 72
00001113  61 6D 0D 0A              141           DC.B 'am',CR,LF
00001117  46 6C 6F 61 74           142.          DC.B 'Floating point version',CR,LF,EOT
          69 6E 67 20 70
          6F 69 6E 74 20
          76 65 72 73 69
          6F 6E 0D 0A 04
00001130  49 6E 70 75 74           143   ADR1    DC.B 'Input Address of A(0,0) > ',EOT
          20 41 64 64 72
          65 73 73 20 6F
          66 20 41 28 30
          2C 30 29 20 3E
          20 04
```

```
0000114B  0A 0D 49 6E 70  144.  ADR2    DC.B LF,CR,'Input Address of B(0,0) > '
          75 74 20 41 64
          64 72 65 73 73
          20 6F 66 20 42
          28 30 2C 30 29
          20 3E 20
00001167  04              145.          DC.B EOT
00001168  0A 0D 49 6E 70  146.  ADR3    DC.B LF,CR,'Input Address of C(0,0) > '
          75 74 20 41 64
          64 72 65 73 73
          20 6F 66 20 43
          28 30 2C 30 29
          20 3E 20
00001184  04              147.          DC.B EOT
00001185  0A 0D 49 6E 70  148.  DIMEN   DC.B LF,CR,'Input Dimension N > ',EOT
          75 74 20 44 69
          6D 65 6E 73 69
          6F 6E 20 4E 20
          3E 20 04
0000119C  49 6E 76 61 6C  149.  ERRMES  DC.B 'Invalid Address or Data Specifica'
          69 64 20 41 64
          64 72 65 73 73
          20 6F 72 20 44
          61 74 61 20 53
          70 65 63 69 66
          69 63 61
000011BD  74 69 6F 6E 0D  150.          DC.B 'tion',CR,LF,EOT
          0A 04
                          151.  *
                          152.  * SUBROUTINE ADDFP
                          153.  *
                          154.  * FLOATING POINT ADDITION FOR 32-BIT
                          155.  * FLOATING POINT FORMAT:
                          156.  * SIGN (1)| FRACTION (23) | EXPONENT (8)
                          157   * WHERE EXPONENT IS BIASED EXCESS 128;
                          158   * FRACTION IS NORMALIZED
                          159.  * 0 = $00000000
                          160   * (SIGN) INFINTY = $(8 OR 0)0000FF
                          161.  * NaN = $XXXXXXXX
                          162.  *       WHERE XXXXXX IS NOT AS INFINITY
                          163.  *
                          164.  * DOES A*B=C WHERE
                          165.  * A IS IN D0
                          166.  * B IS IN D1
                          167.  * C RETURNS IN D0
                          168.  * IF A OR B = NaN, SETS C=NaN = $FFFFFFFF
                          169   * IF A OR B = +-INFINITY, SET C=NaN
                          170.  *
000011C4  48E7 3F00       171.  ADDFP   MOVEM.L D2-D7,-(SP)      ,SAVE OFF
000011C8  0C00 00FF       172.          CMPI.B #BGEXP,D0         ,A=NaN OR inf?
000011CC  6700 00BC       173.          BEQ CNANAD               ,C=NaN IF SO
000011D0  0C01 00FF       174.          CMPI B #BGEXP,D1         ,B=NaN OR inf?
000011D4  6700 00B4       175.          BEQ CNANAD               ,C=NaN IF SO
000011D8  4A81            176.          TST.L D1                 ,B=0?
000011DA  6700 0096       177.          BEQ UNSTCKA              ;C=A IF SO
000011DE  4A80            178.          TST.L D0                 ,A=0?
000011E0  6700 00AE       179.          BEQ CISB                 ,C=B IF SO
000011E4  4243            180.          CLR.W D3
000011E6  4244            181.          CLR.W D4
000011E8  1600            182.          MOVE B D0,D3             ,(D3 W)=EXPA
000011EA  1801            183.          MOVE B D1,D4             ,(D4 W)=EXPB
000011EC  9644            184.          SUB.W D4,D3              ,(D3.W)=SHFTCNT
000011EE  0C43 0017       185           CMPI W #SIGBITS,D3       ,TOO BIG? C=A
000011F2  6C00 007E       186.          BGE UNSTCKA
000011F6  0C43 FFE9       187.          CMPI.W #-SIGBITS,D3      ,TOO SMALL? C=B
000011FA  6F00 0094       188           BLE CISB
                          189.  *
                          190.  * HERE, MUST DO ADD
                          191.  * D5, D6 GET FRACTIONS, D7 CARRIES SIGN
                          192.  *
000011FE  7E00            193           MOVEQ.L #0,D7
00001200  0800 001F       194.          BTST.L #SGNMSK,D0        ,SIGN OF A
00001204  67 04           195.          BEQ.S POSTVE
00001206  08C7 001F       196.          BSET.L #SGNMSK,D7        ,C IS NEG
0000120A  2A00            197.  POSTVE  MOVE.L D0,D5             ,(D5)=AF
0000120C  2C01            198           MOVE L D1,D6             ,(D6)=BF
0000120E  4205            199.          CLR.B D5
00001210  4206            200.          CLR.B D6                 ,KILL EXPS
00001212  E38D            201           LSL.L #1,D5
00001214  E38E            202.          LSL.L #1,D6              ,ALIGN
00001216  7400            203           MOVEQ L #0,D2            ,
```

```
00001218  1400              MOVE.B D0,D2            ;CEXP=AEXP
0000121A  4A43              TST.W D3                ;TEST SC
0000121C  67 06             BEQ.S NOADJ             ;IF=0 NO SHFTNG
0000121E  6B00 0074         BMI SHFTA               ;NEG, SHFT A
00001222  E6AE              LSR.L D3,D6             ;SHIFTB
00001224  2600              MOVE.L D0,D3            ;COPY OF A
00001226  B383              EOR.L D1,D3             ,SIGNS
00001228  0803 001F         BTST.L #SGNMSK,D3       ;
0000122C  6700 0052         BEQ SAMSGN              ;EQUAL SIGNS?
00001230  4486              NEG.L D6                ;NEG B
00001232  DC85              ADD.L D5,D6             ;D6 HAS THE SUM
00001234  6500 0042         BCS NCMPC               ;IF CAR=0, NOT=0
00001238  0847 001F         BCHG.L #SGNMSK,D7       ;COMPLEMENT SUM
0000123C  4486              NEG.L D6
0000123E  0806 001F         BTST.L #SGNMSK,D6       ;NORMALIZED?
00001242  66 06             BNE.S CNORMD
00001244  E38E              LSL.L #1,D6             ;ADJUST FRACTION
00001246  5342              SUBQ.W #1,D2            ;ADJUST EXP
00001248  60 F4             BRA.S NORMC
                            *
                            * BOUND CHECK ON EXP
                            *
0000124A  4A42              CNORMD  TST.W D2
0000124C  6F00 002E         BLE CZERO
00001250  0C42 00FF         CMPI.W #BGEXP,D2
00001254  6700 0046         BEQ CINFAD
                            *
                            * ROUND OFF FRACTION
                            *
00001258  0806 0009         ROUNDC  BTST.L #ALSB,D6
0000125C  66 0A             BNE.S NOROUN
0000125E  0806 0008         BTST.L #ANTLSB,D6
00001262  67 04             BEQ.S NOROUN
00001264  0046 0020         ORI.W #ARNDDR,D6
                            *
                            * ALIGN RESULT
                            *
00001268  E28E              NOROUN  LSR.L #1,D6             ;ROOM FOR SIGN
0000126A  4206              CLR.B D6                ;ROOM FOR EXP
                            *
                            * WITH BOUND CHECKS ON EXP, HAS LEADING ZEROES
                            *
0000126C  8486              OR.L D6,D2
0000126E  8487              OR.L D7,D2
00001270  2002              MOVE.L D2,D0
00001272  4CDF 00FC         UNSTCKA MOVEM.L (SP)+,D2-D7
00001276  4E75              RTS
                            *
00001278  4A86              NCMPC   TST.L D6                ;C=0?
0000127A  66 C2             BNE.S NORMC             ;NO, NORMALIZE
0000127C  4280              CZERO   CLR.L D0
0000127E  60 F2             BRA UNSTCKA
00001280  DC85              SAMSGN  ADD.L D5,D6             ;D6 = SUM FRACT
00001282  64 D4             BCC.S ROUNDC
00001284  E296              ROXR.L #1,D6            ;IF CARRY,
00001286  5242              ADDQ.W #1,D2            ,ADJUST RESULT
00001288  60 C0             BRA.S CNORMD            ;GO ROUND
0000128A  7000              CNANAD  MOVEQ.L #0,D0
0000128C  5380              SUBQ.L #1,D0            ;C=NaN
0000128E  60 E2             BRA UNSTCKA
00001290  2001              CISB    MOVE.L D1,D0
00001292  60 DE             BRA UNSTCKA
00001294  4443              SHFTA   NEG.W D3                ,POS SHIFT CNT
00001296  E6AD              LSR.L D3,D5             ;ADJUST A FRACT
00001298  1401              MOVE.B D1,D2            ;USE BEXP=CEXP
0000129A  60 88             BRA NOADJ
0000129C  7000              CINFAD  MOVEQ.L #0,D0
0000129E  103C 00FF         MOVE.B #BGEXP,D0        ;C=inf
000012A2  8087              OR.L D7,D0              ;TAKE SIGN
000012A4  60 CC             BRA UNSTCKA             ;QUIT
                            *
                            *
                            * SUBROUTINE MULTFP
                            *
                            * FLOATING POINT MULTIPLICATION FOR 32-BIT
                            * FLOATING POINT FORMAT:
                            * SIGN (1)| FRACTION (23) | EXPONENT (8)
                            * WHERE EXPONENT IS BIASED EXCESS 128,
                            * FRACTION IS NORMALIZED
                            * 0 = $00000000
                            * (SIGN) INFINTY = $(8.OR.0)00000FF
```

```
                                285.  * NaN = $XXXXXXXF
                                286.  *       WHERE XXXXXX IS NOT AS INFINITY
                                287.  *
                                288.  * DOES A*B=C WHERE
                                289.  * A IS IN D0
                                290.  * B IS IN D1
                                291.  * C RETURNS IN D0
                                292.  * IF A OR B = NaN, SETS C=NaN = $FFFFFFFF
                                293.  * IF A OR B = +-INFINITY, SET C=+-INFINITY
                                294.  *
000012A6  48E7 3F00             295.  MULTFP  MOVEM.L D2-D7,-(SP)      ;SAVE OFF
000012AA  0C00 00FF             296.          CMPI.B #BGEXP,D0         ,A=NaN OR inf?
000012AE  6700 009A             297.          BEQ AFUNNY               ,SERVICE IF SO
000012B2  4A80                  298.          TST.L D0                 ,A=0?
000012B4  6700 00C2             299.          BEQ AZERO
000012B8  0C01 00FF             300           CMPI.B #BGEXP,D1         ;B=NaN OR inf?
000012BC  6700 00C4             301.          BEQ BFUNNY               ;SERVICE IF SO
000012C0  4A81                  302.          TST.L D1                 ;B=0?
000012C2  6700 00BA             303.          BEQ ZERES                ,A IS VALID, B=0
                                304   *                                ,THEN C=0
000012C6  2800                  305           MOVE.L D0,D4             ,D4 HAS A
000012C8  2601                  306.          MOVE.L D1,D3             ,D3 HAS B
000012CA  0884 001F             307.          BCLR.L #SGNMSK,D4        ;KILL SIGN OF A
000012CE  0883 001F             308.          BCLR.L #SGNMSK,D3        ;KILL SIGN OF B
000012D2  E08C                  309.          LSR.L #8,D4              ,D4 HAS A(F)
000012D4  E08B                  310.          LSR.L #8,D3              ;D3 HAS B(F)
000012D6  4245                  311.          CLR.W D5                 ;EXP CALC
000012D8  4246                  312.          CLR.W D6
000012DA  1A00                  313           MOVE.B D0,D5             ,A(E) IN D5 W
000012DC  1C01                  314.          MOVE.B D1,D6             ,B(E) IN D6 W
000012DE  DA46                  315.          ADD.W D6,D5              ,C(E) IN D5 W,
                                316   *                                ,DBL BIAS
000012E0  0445 0080             317.          SUBI.W #BIAS,D5          ;C(E) IN D5 W,
                                318   *                                ,SNG BIAS
000012E4  6F00 0098             319.          BLE ZERES                ,0 IF TOO SMALL
000012E8  0C45 00FF             320.          CMPI.W #BGEXP,D5         ,TOO BIG?
000012EC  6C00 0076             321.          BGE CINFM                ,C=inf IF SO
                                322.  *
                                323.  * CAN PROCEED WITH MULTIPLY OPERATIONS
                                324.  *
000012F0  2E04                  325.          MOVE L D4,D7     ,(D7) = AhAl
000012F2  C8C3                  326.          MULU W D3,D4     ,(D4) = Al*Bl
000012F4  2407                  327           MOVE.L D7,D2
000012F6  4842                  328           SWAP.W D2        ,(D2 W) = Ah
000012F8  C4C3                  329.          MULU.W D3,D2     ,(D2) = Ah*Bl
000012FA  4843                  330.          SWAP W D3        ,(D3) = BlBh
000012FC  3C07                  331           MOVE.W D7,D6     ,(D6 W) = Al
000012FE  CCC3                  332           MULU D3,D6       ,(D6) = Al*Bh
00001300  DC82                  333           ADD.L D2,D6      ,(D6) = Al*Bh+Ah*Bl
00001302  2407                  334.          MOVE.L D7,D2
00001304  4842                  335.          SWAP.W D2        ,(D2.W) = Ah
00001306  C4C3                  336.          MULU.W D3,D2     ,(D2)=Ah*Bh WON'T OVREL
00001308  4842                  337.          SWAP.W D2
0000130A  4242                  338.          CLR.W D2         ,CANT DO ASL #16,D2
0000130C  D486                  339.          ADD.L D6,D2      ,(D2)=Al*Bh+AhBl
                                340.  *                        ;+(Ah*Bh)SHFT16
0000130E  4244                  341.          CLR.W D4         ;LSW NOT SIGNIFICANT
00001310  4844                  342.          SWAP W D4        ,ALIGN FOR ADD
00001312  D484                  343.          ADD L D4,D2      ,(D2)=C(F) W/ MORE BITS
                                344   *
                                345.  * NORMALIZE FRACTION, THEN ROUND
                                346.  *
00001314  0802 001D             347           BTST.L #NORMTST,D2       ,NORMED ALREADY?
00001318  66 08                 348.          BNE.S NOSHFT             ,
0000131A  E382                  349           ASL.L #1,D2              ,NORMALIZE
0000131C  5345                  350           SUBQ W #1,D5             ,C(E) ADJUST
0000131E  4A45                  351.          TST.W D5                 ,TOO SMALL ?
00001320  67 5C                 352           BEQ.S ZERES              ,WON'T BE =0,
                                353.  *                                ,TESTED BEFORE
00001322  0802 0007             354   NOSHFT  BTST.L #MLSB,D2          ,NEED RND UP?
00001326  66 0A                 355           BNE.S NORNDUP
00001328  0802 0006             356           BTST.L #MNTLSB,D2        ,NEED RND UP?
0000132C  67 04                 357           BEQ S NORNDUP
0000132E  0002 0080             358           ORI.B #MRNDDR,D2         ,RND UP,
                                359   *                                ,SET BIT LSB
00001332  E382                  360.  NORNDUP ASL L #1,D2              ,ALIGN C(F)
00001334  1405                  361.          MOVE B D5,D2             ,INS EXPON C(E)
00001336  2801                  362.  DOSIGN  MOVE.L D1,D4             ,(D4) = AhAl
00001338  B184                  363.          EOR.L D0,D4              ,EOR FOR SIGN
0000133A  0284 80000000         364           ANDI L #ABSGN,D4         ,KEEP JUST MSBIT
00001340  8484                  365.          OR L D4,D2               ,FP RSLT IN D2 L
```

```
00001342  2002              366.          MOVE.L D2,D0            ;RESULT IN D0.L
00001344  4CDF 00FC         367.  UNSTCKM MOVEM.L (SP)+,D2-D7
00001348  4E75              368.          RTS
                            369.  *
                            370.  * EXCEPTIONS
                            371.  *
0000134A  2600              372.  AFUNNY  MOVE.L D0,D3
0000134C  0283 7FFFFF00     373.          ANDI.L #$7FFFFF00,D3    ;CLR SGN, EXP
00001352  67 06             374.          BEQ.S AINF              ;IF ZERO A=inf
00001354  7000              375.  CNANM   MOVEQ.L #0,D0           ;(A OR B)=NaN?
                            376.  *                             ;THEN FLAG C=NaN
00001356  5380              377.          SUBQ.L #1,D0            ;C=NaN
00001358  60 EA             378.          BRA.S UNSTCKM
0000135A  0C01 00FF         379.  AINF    CMPI.B #BGEXP,D1        ;B=inf OR NaN?
0000135E  67 0C             380.          BEQ.S BCHECK
00001360  4A81              381.          TST.L D1                ;B=0?
00001362  67 F0             382.          BEQ.S CNANM             ;Y: inf*0=NaN,
                            383.  *                             ;N: inf
00001364  7400              384.  CINFM   MOVEQ.L #0,D2           ;C=+-inf
00001366  143C 00FF         385.          MOVE.B #BGEXP,D2        ;EXP=$FF
0000136A  60 CA             386.          BRA DOSIGN              ;SGN A AND B
0000136C  2801              387.  BCHECK  MOVE.L D1,D4
0000136E  0284 7FFFFF00     388.          ANDI.L #$7FFFFF00,D4    ;CLR SGN, EXP
00001374  67 EE             389.          BEQ.S CINFM             ;IF 0, B=+-inf,
                            390   *                             ;A=+-inf,C=+-inf
00001376  60 DC             391.          BRA.S CNANM             ;ELSE B=NaN:
                            392.  *                             ;THEN C=NaN
00001378  0C01 00FF         393   AZERO   CMPI.B #BGEXP,D1       ;B=inf OR NaN?
0000137C  67 D6             394.          BEQ.S CNANM             ;Y:
                            395.  *                             ;0*(INF OR NaN)
                            396.  *                             ;=NaN
0000137E  7000              397.  ZERES   MOVEQ.L #0,D0           ;ELSE 0*VALID=0
00001380  60 C2             398.          BRA UNSTCKM             ;
00001382  2801              399.  BFUNNY  MOVE.L D1,D4
00001384  0284 7FFFFF00     400.          ANDI.L #$7FFFFF00,D4    ;CLR SIGN, EXP
0000138A  67 D8             401.          BEQ.S CINFM             ;A VALID;
                            402.  *                             ;B=+-inf
                            403.  *                             ;C=+-inf
0000138C  60 C6             404.          BRA.S CNANM             ;B=NaN: C=NaN
                            405.  *
                            406.  * DONE EXCEPTIONS
                            407.  *
0000138E                    408.          END
    0 Errors
```

K.2 Program MATFPM: Multicomputer Matrix Multiplication, In-order data collection

```
                                1           TTL matfpm
                                2   *
                                3.  * THIS PROGRAM PERFORMS multiplication
                                4.  * of 2 N x N matrices, N divisible by 4
                                5   * (maximum 20 x 20) on the RMCS
                                6   * does C = A x B
                                7   * this version downloads the common B matrix
                                8   * to all using broadcast, then loads the
                                9   * appropriate rows of the A matrix to the
                               10   * respective slaves, allowing execution to
                               11   * overlap with loading
                               12   *
                               13   * March 3, 1992
                               14.  *
                               15   * FLOATING POINT INPUT, OUTPUT
                               16.  * executes 256 times, with averages etc
                               17   * output
                               18   *
                               19   * EQUATES
                               20   *
# 00012FCC                     21.  PCDR EQU $12FCC
# 00012FC0                     22   PCCR EQU $12FC0
# 00000010                     23   ENABLEA EQU $10
# 00000020                     24   ENABLEB EQU $20
# 00000003                     25   SRQASRT EQU 3
# 00000004                     26.  ADBYTIN EQU 4
# 00000007                     27   ADBYTOT EQU 7
# 00000005                     28   INDATA  EQU 5
# 00000006                     29   OUTDATA EQU 6
# 00000008                     30.  NETCNF  EQU 8
# 00000009                     31   SRQSRVC EQU 9
# 0000000E                     32   ABORT   EQU 14
# 00000001                     33   SLV0    EQU $01
# 00000002                     34   SLV1    EQU $02
# 00000004                     35.  SLV2    EQU $04
# 00000008                     36   SLV3    EQU $08
# 0000000F                     37   ALLSLV  EQU $0F
# 00012FB0                     38   PSR01   EQU $12FB0
# 00012FB1                     39   PSR23   EQU $12FB1
# 00012FB2                     40   PSR45   EQU $12FB2
# 00012FB3                     41   PSR67   EQU $12FB3
# 00012FB4                     42.  PSR89   EQU $12FB4
# 00012FB5                     43   PSR1011 EQU $12FB5
# 00012FB6                     44   PSR1213 EQU $12FB6
# 00012FB7                     45   PSR1415 EQU $12FB7
# 00012FB8                     46   PSR1617 EQU $12FB8
# 00012FB9                     47   PSR1819 EQU $12FB9
# 00012FD0                     48.  TCR     EQU $12FD0
# 00012FD2                     49   TPLR    EQU TCR+2
# 00012FD6                     50   TCNTR   EQU TCR+6
# 0000000A                     51.  SENCLFC EQU 10
# 00000018                     52.  DISBUF8 EQU 24
# 00000016                     53.  DISBUF2 EQU 22
# 00FFFFFF                     54   CNTSTRT EQU $00FFFFFF
# 00000001                     55.  ENABLTM EQU 1
# 00000011                     56.  OUTMESC EQU 17
# 00000100                     57.  PASSCNT EQU 256
# 00000008                     58   DIV256  EQU 8
# 00000004                     59   EOT     EQU 4
# 0000000D                     60   CR      EQU $0D
# 0000000A                     61.  LF      EQU $0A
# 000000FF                     62   BGEXP   EQU $FF
# 0000001F                     63.  SGNMSK  EQU 31
# 00000080                     64   BIAS    EQU 128
# 00000009                     65.  ALSB    EQU 9
# 00000008                     66   ANTLSB  EQU 8
# 00000020                     67   ARNDDR  EQU $20
# 80000000                     68   ABSGN   EQU $80000000
# 00000017                     69   SIGBITS EQU 23
# 0000001D                     70   NORMTST EQU 29
# 00000007                     71   MLSB    EQU 7
# 00000006                     72   MNTLSB  EQU 6
# 00000080                     73   MRNDDR  EQU $80
                               74.
                               75   *
                               76   ******************
                               77   * MASTER PROGRAM *
```

```
                                78   ******************
                                79.  *
00000FFE                        80.       ORG.L $1000-2
00000FFE  <2>                   81.  NSPEC   DS.B $2 ;MUST HAVE N SPECIFIER HERE
00001000  <640>                 82.  BMATRIX DS.B $640
00001640  <640>                 83.  AMATRIX DS.B $640
00001C80  <640>                 84.  CMATRIX DS.B $640
000022C0  <4>                   85.  TEMPCNT DS.B $4      ;TEMP FOR SLAVE SELF TIME
000022C4  BD BB BD CB BD        86.  BRDCAST DC.B $BD,$BB,$BD,$CB,$BD,$8C,$CB,$8B
          8C CB 8B
000022CC  B8 8B                 87.           DC.B $B8,$8B
                                88.  *
                                89.  * CONFIGURE NETWORK TO BROADCAST STATE, ALL
                                90.  * PROGRAMS EXACTLY THE SAME!
                                91.  * RESPOND TO ALL SLAVE'S RFP SRQ, AND LOAD
                                92.  * THE SLAVE PROGRAM DOWN
                                93.  *
000022CE  287C 00012FD0         94.  STRTMAT MOVEA.L #TCR,A4          ;POINTER TIMER
000022D4  4214                  95.          CLR.B (A4)               ;DISABLE F'SURE
000022D6  2F3C 00FFFFFF         96.          MOVE.L #CNTSTRT,-(SP)    ;MIN TIME
000022DC  2F3C 00000000         97.          MOVE.L #$00,-(SP)        ;MAX TIME
000022E2  2F3C 00000000         98.          MOVE.L #$00,-(SP)        ;RUN'G SUM TIME
000022E8  2E3C 000000FF         99.          MOVE.L #PASSCNT-1,D7     ;COUNTER
000022EE  41FA FFD4            100.          LEA BRDCAST(PC),A0
000022F2  4E48                 101.          TRAP #NETCNF
000022F4  103C 000F            102.          MOVE.B #ALLSLV,D0
000022F8  4E49                 103.          TRAP #SRQSRVC
000022FA  41FA 0386            104.          LEA SLVPRO(PC),A0        ;STRT ADDR
000022FE  43FA 05F2            105.          LEA SLVPRO2(PC),A1       ;END ADDR+1
00002302  93C8                 106           SUBA.L A0,A1
00002304  3009                 107.          MOVE.W A1,D0
00002306  13FC 0020            108.          MOVE.B #ENABLEB,PGCR
          00012FC0
0000230E  4E47                 109.          TRAP #ADBYTOT
00002310  4E46                 110.          TRAP #OUTDATA
00002312  4239 00012FC0        111.          CLR.B PGCR
                               112.  *
                               113.  * ALL PROGRAMS LOADED, EACH ASSERTING READY FOR
                               114.  * DATA SRQ.  RESPOND TO ALL, THEN DOWNLOAD THE
                               115.  * BMATRIX DATA IN BROADCAST MODE. NOTE THAT THE
                               116   * BYTE COUNT IS ACTUALLY LONGER TO ALLOW
                               117.  * CONSECUTIVE LOADING OF THE PROPER AMATRIX TO
                               118.  * EACH SLAVE FOLLOWING ITS BMATRIX
                               119.  *
                               120.  * LOAD TIMER PRE-LOAD REGISTER HERE
                               121.  *
00002318  41FA FFAA            122.  REEXE   LEA BRDCAST(PC),A0
0000231C  4E48                 123           TRAP #NETCNF             ;CONFIG BRDCAST
0000231E  297C 00FFFFFF        124           MOVE.L #CNTSTRT,2(A4)
          0002
00002326  4281                 125.          CLR.L D1
00002328  103C 000F            126.          MOVE.B #ALLSLV,D0
0000232C  4E49                 127.          TRAP #SRQSRVC
0000232E  41FA ECCE            128.          LEA NSPEC(PC),A0         ;START OF DATA
00002332  3210                 129.          MOVE.W (A0),D1   ;D1 HOLDS N
00002334  2401                 130.          MOVE.L D1,D2
00002336  C4C2                 131.          MULU D2,D2       ;D2 HOLDS N**2
00002338  2602                 132.          MOVE.L D2,D3     ;D3=4N**2/4
0000233A  E582                 133           ASL.L #2,D2      ;D2=4N**2
0000233C  2003                 134.          MOVE.L D3,D0
0000233E  D082                 135.          ADD.L D2,D0      ;D0=4N**2+N**2+2
00002340  5480                 136.          ADDQ L #2,D0     ;BYTE COUNT TO BRDCAST
                               137.  *
                               138.  * SEND START ADDRESS AND BYTE COUNT TO SLAVES
                               139.  * ENABLE TIMER TOO!
                               140.  *
00002342  18BC 0001            141.  TMSTRT  MOVE.B #ENABLTM,(A4)
00002346  13FC 0020            142           MOVE.B #ENABLEB,PGCR
          00012FC0
0000234E  4E47                 143.          TRAP #ADBYTOT    ;ADDRESS AND BYTE COUNT
                               144.  *
                               145.  * MODIFY D0 TO HAVE ONLY BMATRIX BYTE COUNT+2
                               146.  * SEND IT OUT...SLAVES WAIT FOR REST
                               147.  *
00002350  2002                 148.          MOVE.L D2,D0
00002352  5480                 149.          ADDQ.L #2,D0
00002354  4E46                 150.          TRAP #OUTDATA    ;SEND OFF THE B MATRIX
00002356  4239 00012FC0        151.          CLR.B PGCR       ;
                               152.  *
                               153.  * NOW (A0)+(D0) POINTS TO START OF THE AMATRIX,
                               154   * THE FIRST (4*N**2)/4 ELEMENTS OF WHICH GO TO
```

```
                          155.  * SLAVE 0.  CONFIGURE NETWORK
                          156.  *
0000235C  13FC 00BB       157.          MOVE.B #$BB,PSR01        ;TO SLV0 ONLY
          00012FB0
00002364  D1C0            158.          ADDA.L D0,A0
00002366  3003            159.          MOVE.W D3,D0     ,ADDITIONAL BYTE COUNT
00002368  13FC 0020       160.          MOVE.B #ENABLEB,PGCR     ;ENABLE PORT
          00012FC0
00002370  4E46            161.          TRAP #OUTDATA    ;SEND SLAVE 0 PORTION
00002372  4239 00012FC0   162.          CLR.B PGCR       ;
                          163.  *
                          164.  * RESPOND TO SLAVE 0'S RTE SRQ, AND LET EXECUTE
                          165.  *
00002378  103C 0001       166.          MOVE.B #SLV0,D0
0000237C  4E49            167.          TRAP #SRQSRVC
                          168.  *
                          169.  * NEXT CHUNK OF DATA GOES TO SLAVE 1, STARTS
                          170.  * AT A0+D3, AND HAS BYTE COUNT D3
                          171.  * CONFIGURE NETWORK TO SLAVE 1
                          172.  *
0000237E  13FC 00BB       173.          MOVE.B #$BB,PSR45
          00012FB2
00002386  13FC 0004       174.          MOVE.B #$04,PSR01        ,TO SLV1 ONLY
          00012FB0
0000238E  D1C3            175.          ADDA.L D3,A0
00002390  3003            176.          MOVE.W D3,D0
00002392  13FC 0020       177.          MOVE.B #ENABLEB,PGCR
          00012FC0
0000239A  4E46            178.          TRAP #OUTDATA
0000239C  4239 00012FC0   179.          CLR.B PGCR
                          180.  *
                          181.  * RESPOND TO SLAVE 1'S RTE SRQ, AND LET EXECUTE
                          182.  *
000023A2  103C 0002       183           MOVE.B #SLV1,D0
000023A6  4E49            184           TRAP #SRQSRVC
                          185   *
                          186.  * NEXT CHUNK OF DATA GOES TO SLAVE 2, STARTS
                          187.  * AT A0+D3, AND HAS BYTE COUNT D3
                          188.  * CONFIGURE NETWORK TO SLAVE 2
                          189.  *
000023A8  13FC 00BB       190.          MOVE.B #$BB,PSR89
          00012FB4
000023B0  13FC 0004       191.          MOVE.B #$04,PSR45        ,TO SLV2 ONLY
          00012FB2
000023B8  D1C3            192.          ADDA.L D3,A0
000023BA  3003            193.          MOVE.W D3,D0
000023BC  13FC 0020       194           MOVE B #ENABLEB,PGCR
          00012FC0
000023C4  4E46            195.          TRAP #OUTDATA
000023C6  4239 00012FC0   196.          CLR.B PGCR
                          197.  *
                          198.  * RESPOND TO SLAVE 2'S RTE SRQ, AND LET EXECUTE
                          199.  *
000023CC  103C 0004       200           MOVE.B #SLV2,D0
000023D0  4E49            201.          TRAP #SRQSRVC
                          202.  *
                          203.  * NEXT CHUNK OF DATA GOES TO SLAVE 3, STARTS
                          204.  * AT A0+D3, AND HAS BYTE COUNT D3
                          205.  * CONFIGURE NETWORK TO SLAVE 3
                          206   *
000023D2  13FC 0004       207.          MOVE.B #$04,PSR89        ,TO SLV3 ONLY
          00012FB4
000023DA  D1C3            208           ADDA.L D3,A0
000023DC  3003            209           MOVE.W D3,D0
000023DE  13FC 0020       210.          MOVE.B #ENABLEB,PGCR
          00012FC0
000023E6  4E46            211.          TRAP #OUTDATA
000023E8  4239 00012FC0   212.          CLR B PGCR
                          213.  *
                          214   * RESPOND TO SLAVE 3'S RTE SRQ, AND LET EXECUTE
                          215   *
000023EE  103C 0008       216           MOVE.B #SLV3,D0
000023F2  4E49            217           TRAP #SRQSRVC
                          218   *
                          219   * ALL DATA IS LOADED DOWN TO SLAVES, AND ARE
                          220   * RESPOND TO INDIVIDUAL SLAVE SRQ'S AND UPLOAD
                          221.  * THE RESULTS TO THE CMATRIX ARRAY
                          222   * BYTE COUNT FROM EACH SLAVE WILL BE
                          223   * ((4*N**2)/4)=D3
                          224   *
000023F4  41FA F88A       225           LEA CMATRIX(PC),A0 ,POINTER TO OUTPUT
```

```
000023F8  103C 0001        226.          MOVE.B #SLV0,D0
000023FC  4E49             227.          TRAP #SRQSRVC    ;RESPOND TO SLAV0
000023FE  3003             228.          MOVE.W D3,D0
00002400  13FC 0010        229.          MOVE.B #ENABLEA,PGCR     ;ENABLE INPUT
          00012FC0
00002408  4E45             230.          TRAP #INDATA
0000240A  4239 00012FC0    231.          CLR.B PGCR
00002410  D1C3             232.          ADDA.L D3,A0     ;NEW POINTER
00002412  103C 0002        233.          MOVE.B #SLV1,D0
00002416  4E49             234.          TRAP #SRQSRVC    ;RESPOND TO SLAV1
00002418  13FC 00BC        235.          MOVE.B #$BC,PSR1011
          00012FB5
00002420  3003             236.          MOVE.W D3,D0
00002422  13FC 0010        237.          MOVE.B #ENABLEA,PGCR     ;ENABLE INPUT
          00012FC0
0000242A  4E45             238.          TRAP #INDATA
0000242C  4239 00012FC0    239.          CLR.B PGCR
00002432  D1C3             240.          ADDA.L D3,A0     ;NEW POINTER
00002434  103C 0004        241.          MOVE.B #SLV2,D0
00002438  4E49             242.          TRAP #SRQSRVC    ;RESPOND TO SLAV2
0000243A  13FC 00BB        243.          MOVE.B #$BB,PSR1415
          00012FB7
00002442  3003             244.          MOVE.W D3,D0
00002444  13FC 0010        245.          MOVE.B #ENABLEA,PGCR     ;ENABLE INPUT
          00012FC0
0000244C  4E45             246.          TRAP #INDATA
0000244E  4239 00012FC0    247.          CLR.B PGCR
00002454  D1C3             248.          ADDA.L D3,A0     ;NEW POINTER
00002456  103C 0008        249.          MOVE.B #SLV3,D0
0000245A  4E49             250.          TRAP #SRQSRVC    ;RESPOND TO SLAV3
0000245C  13FC 00BB        251.          MOVE.B #$BB,PSR1819
          00012FB9
00002464  3003             252.          MOVE.W D3,D0
00002466  13FC 0010        253.          MOVE.B #ENABLEA,PGCR     ;ENABLE INPUT
          00012FC0
0000246E  4E45             254.          TRAP #INDATA
00002470  4239 00012FC0    255.          CLR.B PGCR
                           256.  *
                           257.  * PROGRAM IS DONE, CAN STOP TIMER, AND
                           258.  * OUTPUT NUMBER OF 32 CLOCK INTERVALS
                           259.  * PASSED. FIRST DISABLE TIMER
                           260.  *
00002476  4214             261.          CLR.B (A4)
00002478  222C 0006        262.          MOVE.L 6(A4),D1 ;GET NEW COUNT
0000247C  203C 00FFFFFF    263.          MOVE.L #CNTSTRT,D0
00002482  9081             264.          SUB.L D1,D0      ;GET ELAPSED COUNT
                           265.  *
                           266.  * OUTPUT CR, LF, THEN THE COUNT IN HEX
                           267.  *
00002484  4E4F             268.          TRAP #15
00002486  000A             269.          DC.W SENCLFC
00002488  4E4F             270.          TRAP #15
0000248A  0018             271.          DC.W DISBUF8
                           272.  *
                           273.  * NOW KEEP TRACK OF STATISTICS
                           274.  *
0000248C  D197             275.          ADD.L D0,(SP)    ;ADD TO SUM
0000248E  B0AF 0004        276.          CMP.L 4(SP),D0   ;CHECK MAX
00002492  6F 04            277.          BLE.S NOTMAX     ;D0 NOT MAX
00002494  2F40 0004        278.          MOVE.L D0,4(SP)  ;NEW MAX
00002498  B0AF 0008        279.  NOTMAX  CMP.L 8(SP),D0   ;CHECK MIN
0000249C  6C 04            280.          BGE.S NOTMIN     ;D0 NOT MIN
0000249E  2F40 0008        281.          MOVE.L D0,8(SP)  ;NEW MIN
000024A2  51CF FE74        282.  NOTMIN  DBRA D7,REEXE    ;DO D7 TIMES
                           283.  *
                           284   * WHEN HERE, PROGRAM HAS BEEN EXECUTED 256 TIMES
                           285.  * SEND AN ILLEGAL ADDRESS FOR DATA, FORCING
                           286.  * SLAVES TO PROGRAM ACCEPT MODE
                           287.  *
000024A6  41FA FE1C        288.          LEA BRDCAST(PC),A0
000024AA  4E48             289.          TRAP #NETCNF
000024AC  103C 000F        290.          MOVE.B #ALLSLV,D0
000024B0  4E49             291.          TRAP #SRQSRVC
000024B2  207C FFFF0000    292.          MOVEA.L #$FFFF0000,A0    ;ILLEGAL ADDR
000024B8  13FC 0020        293.          MOVE.B #ENABLEB,PGCR
          00012FC0
000024C0  4E47             294           TRAP #ADBYTDT    ;THAT'LL FIX 'EM
000024C2  4239 00012FC0    295.          CLR.B PGCR
                           296.  *
                           297   * OUTPUT STATISTICS
                           298.  *
```

```
000024C8  4E4F             299.          TRAP #15
000024CA  000A             300.          DC.W SENCLFC
000024CC  4E4F             301.          TRAP #15
000024CE  000A             302.          DC.W SENCLFC
000024D0  41FA 0104        303.          LEA AVEMES(PC),A0
000024D4  4E4F             304.          TRAP #15
000024D6  0011             305.          DC.W OUTMESC
000024D8  201F             306.          MOVE.L (SP)+,D0 ;GET SUM
000024DA  E080             307.          ASR.L #DIV256,D0        ;DO AVERAGE
000024DC  4E4F             308.          TRAP #15
000024DE  0018             309.          DC.W DISBUF8    ;OUTPUT AVERAGE
000024E0  4E4F             310.          TRAP #15
000024E2  000A             311.          DC.W SENCLFC
000024E4  41FA 0117        312.          LEA MAXMES(PC),A0
000024E8  4E4F             313.          TRAP #15
000024EA  0011             314.          DC.W OUTMESC
000024EC  201F             315.          MOVE.L (SP)+,D0 ;GET MAX
000024EE  4E4F             316.          TRAP #15
000024F0  0018             317.          DC.W DISBUF8
000024F2  4E4F             318.          TRAP #15
000024F4  000A             319.          DC.W SENCLFC
000024F6  41FA 012C        320.          LEA MINMES(PC),A0
000024FA  4E4F             321.          TRAP #15
000024FC  0011             322.          DC.W OUTMESC
000024FE  201F             323.          MOVE.L (SP)+,D0 ;GET MIN
00002500  4E4F             324.          TRAP #15
00002502  0018             325.          DC.W DISBUF8
00002504  4E4F             326.          TRAP #15
00002506  000A             327.          DC.W SENCLFC
                           328.  *
                           329.  * AFTER 256 RUNS, LOAD THE SECOND SLAVE
                           330.  * PROGRAM TO ACQUIRE THE TIMING DATA
                           331.  * IN THE SLAVES TIMER COUNT REGISTERS
                           332.  *
00002508  41FA FDBA        333.          LEA BRDCAST(PC),A0      ,BROADCAST MODE
0000250C  4E48             334.          TRAP #NETCNF
                           335.  *
                           336.  * RESPOND TO SLAVES RFP
                           337.  *
0000250E  103C 000F        338.          MOVE.B #ALLSLV,D0
00002512  4E49             339.          TRAP #SRQSRVC
00002514  41FA 03DC        340.          LEA SLVPRO2(PC),A0
00002518  43FA 03F4        341.          LEA ENDSLV(PC),A1
0000251C  93C8             342.          SUBA.L A0,A1
0000251E  3009             343.          MOVE.W A1,D0
00002520  13FC 0020        344.          MOVE.B #ENABLEB,PGCR
          00012FC0
00002528  4E47             345.          TRAP #ADBYTOT
0000252A  4E46             346.          TRAP #OUTDATA
0000252C  4239 00012FC0    347.          CLR.B PGCR      ;PROGRAM SENT
                           348.  *
                           349.  * RESPOND TO SLAVES RFD
                           350.  *
00002532  103C 000F        351.          MOVE.B #ALLSLV,D0
00002536  4E49             352.          TRAP #SRQSRVC
                           353.  *
                           354.  * PROGRAM HAS NO INPUT DATA, SEND DUMMY
                           355.  * ADDRESS AND ZERO BYTE COUNT
                           356.  * AND THEN NO DATA
                           357.  *
00002538  4240             358.          CLR.W D0
0000253A  13FC 0020        359.          MOVE.B #ENABLEB,PGCR
          00012FC0
00002542  4E47             360.          TRAP #ADBYTOT
00002544  4239 00012FC0    361.          CLR.B PGCR
                           362.  *
                           363.  * RESPOND TO READY TO EXECUTE
                           364.  *
0000254A  103C 000F        365.          MOVE.B #ALLSLV,D0
0000254E  4E49             366.          TRAP #SRQSRVC
                           367.  *
                           368.  * SLAVES EXECUTE AND ASSERT AN SRQ WHEN
                           369.  * READY TO REPORT. RESPOND ONE AT A TIME
                           370.  * DETERMINE TIME ELAPSED AND REPORT TO
                           371.  * CONSOLE
                           372.  *
00002550  103C 0001        373.          MOVE.B #SLV0,D0 ,SLAVE 0
00002554  4281             374.          CLR.L D1
00002556  61 32            375.          BSR.S GETSLTM
00002558  13FC 00BC        376.          MOVE.B #$BC,PSP1011 ,SLAVE 1
          00012FB5
```

```
00002560  5281              377.          ADDQ.L #1,D1
00002562  103C 0002         378.          MOVE.B #SLV1,D0
00002566  61 22             379.          BSR.S GETSLTM
00002568  13FC 00BB         380.          MOVE.B #$BB,PSR1415 ;SLAVE 2
          00012FB7
00002570  5281              381.          ADDQ.L #1,D1
00002572  103C 0004         382.          MOVE.B #SLV2,D0
00002576  61 12             383.          BSR.S GETSLTM
00002578  13FC 00BB         384.          MOVE.B #$BB,PSR1819 ;SLAVE3
          00012FB9
00002580  5281              385.          ADDQ.L #1,D1
00002582  103C 0008         386           MOVE.B #SLV3,D0
00002586  61 02             387.          BSR.S GETSLTM
                            388.  *
                            389.  * NOW FINALLY DONE!
                            390.  *
00002588  4E75              391.          RTS
                            392.  *
                            393.  * SUBROUTINE TO GET TIMING DATA FROM SLAVE
                            394.  * SPECIFIED IN D1.B, WITH MASK IN D0.B
                            395.  *
0000258A  48E7 C0C0         396.  GETSLTM MOVEM.L D0-D1/A0-A1,-(SP) ;SAVE ENV
0000258E  4E49              397.          TRAP #SRQSRVC
00002590  7004              398.          MOVEQ.L #$4,D0  ;BYTE COUNT
00002592  41FA FD2C         399.          LEA TEMPCNT(PC),A0
00002596  13FC 0010         400.          MOVE.B #ENABLEA,PGCR    ;EN`BLE INPUT
          00012FC0
0000259E  4E45              401.          TRAP #INDATA
000025A0  4239 00012FC0     402.          CLR.B PGCR
000025A6  2248              403.          MOVEA.L A0,A1
000025A8  41FA 00A1         404.          LEA MESSA(PC),A0
000025AC  4E4F              405.          TRAP #15
000025AE  0011              406.          DC.W OUTMESC
000025B0  1001              407.          MOVE.B D1,D0
000025B2  4E4F              408.          TRAP #15
000025B4  0016              409.          DC.W DISBUF2
000025B6  41FA 00AD         410.          LEA MESSB(PC),A0
000025BA  4E4F              411.          TRAP #15
000025BC  0011              412.          DC.W OUTMESC
000025BE  2211              413.          MOVE.L (A1),D1
000025C0  203C 00FFFFFF     414.          MOVE.L #CNTSTRT,D0 ;CALC ELAPSED
000025C6  9081              415.          SUB.L D1,D0
000025C8  4E4F              416.          TRAP #15
000025CA  0018              417.          DC.W DISBUF8
000025CC  4E4F              418.          TRAP #15
000025CE  000A              419.          DC.W SENCLFC
000025D0  4CDF 0303         420.          MOVEM.L (SP)+,D0-D1/A0-A1
000025D4  4E75              421.          RTS
                            422.  *
                            423.  *
000025D6  41 76 65 72 61    424.  AVEMES  DC.B 'Average time ( x 4 for microseco'
          67 65 20 74 69
          6D 65 20 28 20
          78 20 34 20 66
          6F 72 20 6D 69
          63 72 6F 73 65
          63 6F
000025F6  6E 64 73 29 3D    425.          DC.B 'nds)= ',EOT
          20 04
000025FD  4D 61 78 69 6D    426.  MAXMES  DC.B 'Maximum time ( x 4 for microseco'
          75 6D 20 74 69
          6D 65 20 28 20
          78 20 34 20 66
          6F 72 20 6D 69
          63 72 6F 73 65
          63 6F
0000261D  6E 64 73 29 3D    427.          DC.B 'nds)= ',EOT
          20 04
00002624  4D 69 6E 69 6D    428.  MINMES  DC.B 'Minimum time ( x 4 for microseco'
          75 6D 20 74 69
          6D 65 20 28 20
          78 20 34 20 66
          6F 72 20 6D 69
          63 72 6F 73 65
          63 6F
00002644  6E 64 73 29 3D    429.          DC.B 'nds)= ',EOT
          20 04
0000264B  45 78 65 63 75    430.  MESSA   DC.B 'Execution time by slave #',EOT
          74 69 6F 6E 20
          74 69 6D 65 20
          62 79 20 73 6C
```

```
          61 76 65 20 23
          04
00002665  20 20 28 20 78  431.  MESSB   DC.B '   ( x 4 for microseconds) = ',EOT
          20 34 20 66 6F
          72 20 6D 69 63
          72 6F 73 65 63
          6F 6E 64 73 29
          20 3D 20 04
                          432.  ********************
                          433.  * ALL SLAVE'S PROGRAM *
                          434.  ********************
                          435.  *
                          436.  * HAVE LOADED N AT NSPEC, FOLLOWED BY BMATRIX
                          437.  * AND THE AMATRIX. BMATRIX IS N x N, AND WORD
                          438.  * LENGTH ELEMENTS. BMATRIX STARTS AT NSPEC+2
                          439.  * AND THE AMATRIX STARTS AT BMATRIX+(2*N**2)
                          440.  * ONCE DISCOVERED, CAN USE THE MATRIX
                          441.  * MULTIPLY SIMILAR TO THAT IN MESC.
                          442.  *
00002682  4239 00012FD0    443.  SLVPRO  CLR.B TCR        ;DISABLE TIMER
00002688  23FC 00FFFFFF    444.          MOVE.L #CNTSTRT,TCR+2 ;LOAD MAX COUNT
          00012FD2
00002692  13FC 0001        445.          MOVE.B #ENABLTM,TCR      ;ENABLE TIMER
          00012FD0
0000269A  4286             446.          CLR.L D6
0000269C  43FA E962        447.          LEA BMATRIX(PC),A1
000026A0  3C29 FFFE        448.          MOVE.W -2(A1),D6         ,D6 HOLDS N
000026A4  3A06             449.          MOVE.W D6,D5
000026A6  CAC5             450.          MULU D5,D5       ;D5=N**2
000026A8  2405             451.          MOVE.L D5,D2     ;D2=N**2
000026AA  E585             452.          ASL.L #2,D5      ;D5=4N**2
000026AC  2A45             453.          MOVEA.L D5,A5    ;
000026AE  9BFC 00000004    454.          SUBA.L #4,A5     ;A5=4N**2-4
000026B4  2049             455.          MOVE.L A1,A0
000026B6  D1C5             456.          ADDA.L D5,A0     ;A0 POINTS TO AMATRIX
000026B8  45FA F5C6        457.          LEA CMATRIX(PC),A2       ,CMATRIX POINTER
                          458.  *
                          459.  *
                          460.  * HERE, DO (N/4 x N) X (N x N) MATRIX MULTIPLY
                          461.  *
                          462.  *
000026BC  D5C2             463.  MATRIX  ADDA.L D2,A2     ;A2=END OF C=A2+N**2
000026BE  5982             464.          SUBQ.L #4,D2     ;D2=N**2-4
000026C0  264D             465.          MOVEA.L A5,A3    ;A3=4N**2-4
000026C2  2606             466.          MOVE.L D6,D3     ;D3=N
000026C4  E583             467.          ASL.L #2,D3      ;D3=4N
000026C6  5386             468.          SUBQ.L #1,D6     ;D6=N-1
000026C8  2E06             469.          MOVE.L D6,D7     ;D7=N-1
000026CA  2846             470.          MOVEA.L D6,A4    ;A4=N-1
000026CC  4285             471.  OUTRLP  CLR.L D5        ;HOLDS RESULT OF ADDS
000026CE  2802             472.          MOVE.L D2,D4
000026D0  2C4D             473.          MOVE.L A5,A6
000026D2  2030 4800        474.  INRLP   MOVE.L 0(A0,D4.L),D0     ;GET A(I,J)
000026D6  2231 E800        475.          MOVE.L 0(A1,A6.L),D1     ;DO MUL B(J,K)
000026DA  6100 012E        476.          BSR MULTFP               ;RESULT IN D0
000026DE  2205             477.          MOVE.L D5 D1
000026E0  6100 0046        478.          BSR ADDF
000026E4  2A00             479.          MOVE.L D0,D5
000026E6  5984             480.          SUBQ.L #4,D4          ;D4=D4-4
000026E8  9DC3             481.          SUBA.L D3,A6          ;D5=D5-D3
000026EA  51CE FFE6        482.          DBRA D6,INRLP         ,DO N TIMES
000026EE  2505             483.          MOVE.L D5,-(A2)       ,D0=N-1 AGAIN
000026F0  2C0C             484.          MOVE.L A4,D6
000026F2  9BFC 00000004    485.          SUBA.L #4,A5
000026F8  51CF FFD2        486.          DBRA D7,OUTRLP        ,DO N TIMES
000026FC  2A4B             487.          MOVE.L A3,A5          ,A5=2*N**2-2
000026FE  2E0C             488.          MOVE.L A4,D7          ,D7=N-1
00002700  9483             489.          SUB.L D3,D2           ,D2=D2-D3
00002702  6C C8            490.          BGE.S OUTRLP
                          491.  *
                          492.  * CALCULATION DONE, REST IS OVERHEAD FOR UPLOADING
                          493.  * STOP TIMER HERE
                          494.  *
00002704  4239 00012FD0    495.          CLR.B TCR
                          496.  *
0000270A  41FA F574        497           LEA CMATRIX(PC),A0       ;POINT TO OUTPUT
0000270E  303A E8EE        498.          MOVE.W NSPEC(PC),D0      ,GET N AGAIN
00002712  C1C0             499.          MULS D0,D0
```

```
00002714  4E43              500.          TRAP #SRQASRT           ;ASSERT SRQ
00002716  13FC 0020         501.          MOVE.B #ENABLEB,PGCR
          00012FC0
0000271E  4E46              502.          TRAP #OUTDATA
00002720  4239 00012FC0     503.          CLR.B PGCR
00002726  4E75              504.          RTS
                            505.  *
                            506.  * SUBROUTINE ADDFP
                            507.  *
                            508.  * FLOATING POINT ADDITION FOR 32-BIT
                            509.  * FLOATING POINT FORMAT:
                            510.  * SIGN (1)| FRACTION (23) | EXPONENT (8)
                            511.  * WHERE EXPONENT IS BIASED EXCESS 128;
                            512.  * FRACTION IS NORMALIZED
                            513.  * 0 = $00000000
                            514.  * (SIGN) INFINTY = $(8.OR.7)0000FF
                            515.  * NaN = $XXXXXXFF
                            516.  *       WHERE XXXXXX IS NOT AS INFINITY
                            517.  *
                            518.  * DOES A*B=C WHERE
                            519.  * A IS IN D0
                            520.  * B IS IN D1
                            521   * C RETURNS IN D0
                            522.  * IF A OR B = NaN, SETS C=NaN = $FFFFFFFF
                            523.  * IF A OR B = +-INFINITY, SET C=NaN
                            524.  *
00002728  48E7 3F00         525.  ADDFP   MOVEM.L D2-D7,-(SP)     ;SAVE OFF
0000272C  0C00 00FF         526.          CMPI.B #BGEXP,D0        ;A=NaN OR inf?
00002730  6700 00BC         527.          BEQ CNANAD              ;C=NaN IF SO
00002734  0C01 00FF         528           CMPI.B #BGEXP,D1        ;B=NaN OR inf?
00002738  6700 00B4         529.          BEQ CNANAD              ;C=NaN IF SO
0000273C  4A81              530           TST.L D1                ;B=0?
0000273E  6700 0096         531.          BEQ UNSTCKA             ;C=A IF SO
00002742  4A80              532.          TST.L D0                ,A=0?
00002744  6700 00AE         533.          BEQ CISB                ;C=B IF SO
00002748  4243              534           CLR.W D3
0000274A  4244              535.          CLR.W D4
0000274C  1600              536.          MOVE.B D0,D3            ;(D3.W)=EXPA
0000274E  1801              537           MOVE.B D1,D4            ;(D4.W)=EXPB
00002750  9644              538.          SUB.W D4,D3             ,(D3.W)=SHFTCNT
00002752  0C43 0017         539.          CMPI.W #SIGBITS,D3      'TOO BIG? C=A
00002756  6C00 007E         540           BGE UNSTCKA
0000275A  0C43 FFE9         541.          CMPI.W #-SIGBITS,D3     ,TOO SMALL? C=B
0000275E  6F00 0094         542.          BLE CISB
                            543   *
                            544.  * HERE, MUST DO ADD
                            545   * D5, D6 GET FRACTIONS; D7 CARRIES SIGN
                            546.  *
00002762  7E00              547           MOVEQ L #0,D7
00002764  0800 001F         548.          BTST.L #SGNMSK,D0       ,SIGN OF A
00002768  67 04             549           BEQ.S POSTVE
0000276A  0EC7 001F         550           BSET.L #SGNMSK,D7       ,C IS NEG
0000276E  2A00              551   POSTVE  MOVE.L D0,D5            ,(D5)=AF
00002770  2C01              552           MOVE.L D1,D6            ;(D6)=BF
00002772  4205              553.          CLR.B D5
00002774  4206              554.          CLR.B D6                ,KILL EXPS
00002776  E38D              555.          LSL.L #1,D5
00002778  E38E              556.          LSL.L #1,D6             ;ALIGN
0000277A  7400              557.          MOVEQ.L #0,D2           ,
0000277C  1400              558.          MOVE.B D0,D2            ,CEXP=AEXP
0000277E  4A43              559           TST.W D3                ,TEST SC
00002780  67 0E             560.          BEQ.S NOADJ             ;IF=0 NO SHFTNG
00002782  6B00 0074         561           BMI SHFTA               ;NEG, SHFT A
00002786  E04E              562.  SHFTB   LSR.L D3,D6             ;SHIFTB
00002788  2600              563   NOADJ   MOVE.L D0,D3            ;COPY OF A
0000278A  B383              564.          EOR.L D1,D3             ;SIGNS
0000278C  0803 001F         565.          BTST.L #SGNMSK,D3       ;
00002790  6700 0052         566.          BEQ SAMSGN              ;EQUAL SIGNS?
00002794  448E              567.  DIFFSGN NEG.L D6                ;NEG B
00002796  DC 86             568           ADD.L D5,D6             ;D6 HAS THE SUM
00002798  6500 0042         569.          BCS NCMPC               ;IF CAR=0, NOT=0
0000279C  0F17 001F         570.  COMPC   BCHG.L #SGNMSK,D7       ;COMPLEMENT SUM
000027A0  44 86             571.          NEG.L D6
000027A2  0806 001F         572.  NORMC   BTST.L #SGNMSK,D6       ;NORMALIZED?
000027A6  66 0E             573           BNE.S CNORMD
000027A8  E38E              574.          LSL.L #1,D6             ;ADJUST FRACTION
000027AA  53 82             575           SUBQ.W #1,D2            ;ADJUST EXP
000027AC  60 F4             576           BRA.S NORMC
                            577.  *
                            578   * BOUND CHECK ON EXP
                            579.  *
```

```
000027AE  4A42          580.  CNORMD  TST.W D2
000027B0  6F00 002E     581.          BLE CZERO
000027B4  0C42 00FF     582           CMPI.W #BGEXP,D2
000027B8  6700 0046     583.          BEQ CINFAD
                        584.  *
                        585.  * ROUND OFF FRACTION
                        586.  *
000027BC  0806 0009     587.  ROUNDC  BTST.L #ALSB,D6
000027C0  66 0A         588.          BNE.S NOROUN
000027C2  0806 0008     589.          BTST.L #ANTLSB.D6
000027C6  67 04         590.          BEQ S NOROUN
000027C8  0046 0020     591           ORI.W #ARNDDR,D6
                        592.  *
                        593.  * ALIGN RESULT
                        594   *
000027CC  E28E          595.  NCROUN  LSR.L #1,D6              ,ROOM FOR SIGN
000027CE  4206          596.          CLR.B D6                ,ROOM FOR EXP
                        597   *
                        598   * WITH BOUND CHECKS ON EXP, HAS LEADING ZEROES
                        599   *
000027D0  8486          600.          OR L D6,D2
000027D2  8487          601.          OR.L D7,D2
000027D4  2002          602.          MOVE.L D2,D0
000027D6  4CDF 00FC     603   UNSTCKA MOVEM.L (SP)+,D2-D7
000027DA  4E75          604.          RTS
                        605   *
000027DC  4A86          606   NCMPC   TST L D6                ,C=0?
000027DE  66 C2         607           BNE.S NORMC             ,NO, NORMALIZE
000027E0  4280          608   CZERO   CLR L D0
000027E2  60 F2         609.          BRA UNSTCKA
000027E4  DC85          610   SAMSGN  ADD L D5,D6             ,D6 = SUM FRACT
000027E6  64 D4         611.          BCC.S ROUNDC
000027E8  E296          612           ROXR L #1,D6            ,IF CARRY,
000027EA  5242          613           ADDQ W #1,D2            ,ADJUST RESULT
000027EC  60 C0         614.          BRA S CNORMD            ,GO ROUND
000027EE  7000          615   CNANAD  MOVEQ L #0,D0
000027F0  5380          616           SUBQ L #1,D0            ,D=NaN
000027F2  60 D2         617           BRA UNSTCKA
000027F4  2001          618.  CISB    MOVE L D1,D0
000027F6  60 DE         619           BRA UNSTCKA
000027F8  4443          620   SHFTA   NEG W D3                ,POS SHIFT CNT
000027FA  E6AD          621           LSR L D3,D5             ,ADJUST A FRACT
000027FC  1401          622           MOVE B D1,D2            ,USE BEXP=AEXP
000027FE  60 88         623           BRA NOADJ
00002800  7000          624   CINFAD  MOVEQ L #0,D0
00002802  103C 00FF     625           MOVE B #BGEXP,D0        ,D=inf
00002806  8087          626           OR L D7,D0              ,TAKE SIGN
00002808  60 CC         627           BRA UNSTCKA             ,QUIT
                        628   *
                        629   *
                        630   * SUBROUTINE MULTFP
                        631   *
                        632   * FLOATING POINT MULTIPLICATION FOR 32-BIT
                        633   * FLOATING POINT FORMAT
                        634   * SIGN (1)| FRACTION (23) | EXPONENT (8)
                        635   * WHERE EXPONENT IS BIASED EXCESS 126,
                        636   * FRACTION IS NORMALIZED
                        637   * 0 = $00000000
                        638   * (SIGN) INFINITY = $(8 OR 0)0000FF
                        639   * NaN = $XXXXXXXX
                        640   *       WHERE XXXXXX IS NOT AS INFINITY
                        641   *
                        642   * DOES A*B=C WHERE
                        643   * A IS IN D0
                        644   * B IS IN D1
                        645   * C RETURNS IN D0
                        646   * IF A OR B = NaN, SETS C=NaN = $FFFFFFFF
                        647   * IF A OR B = +-INFINITY, SET C=+-INFINITY
                        648   *
0000280A  48E7 3F00     649   MULTFP  MOVEM L D2-D7,-(SP)     ,SAVE OFF
0000280E  0C00 00FF     650           CMPI.B #BGEXP,D0        ,A=NaN OR inf?
00002812  6700 009A     651           BEQ AFUNNY              ,SERVICE IF SO
00002816  4A80          652           TST.L D0                ,A=0?
00002818  6700 00C2     653           BEQ AZERO
0000281C  0C01 00FF     654           CMPI.B #BGEXP,D1        ,B=NaN OR inf?
00002820  6700 00C4     655           BEQ BFUNNY              ;SERVICE IF SO
00002824  4A81          656           TST.L D1                ,B=0?
00002826  6700 00BA     657           BEQ ZERES               ,A IS VALID, B=0
                        658   *                               ,THEN C=0
0000282A  2800          659           MOVE.L D0,D4            ;D4 HAS A
0000282C  2601          660           MOVE.L D1,D3            ,D3 HAS B
```

```
0000282E  0884 001F      661           BCLR.L #SGNMSK,D4       ;KILL SIGN OF A
00002832  0883 001F      662.          BCLR.L #SGNMSK,D3       ;KILL SIGN OF B
00002836  E08C           663.          LSR.L #8,D4             ;D4 HAS A(F)
00002838  E08B           664           LSR.L #8,D3             ;D3 HAS B(F)
0000283A  4245           665.          CLR.W D5                ;EXP CALC
0000283C  4246           666.          CLR.W D6
0000283E  1A00           667.          MOVE.B D0,D5            ;A(E) IN D5.W
00002840  1C01           668.          MOVE.B D1,D6            ;B(E) IN D6.W
00002842  DA46           669.          ADD.W D6,D5             ;C(E) IN D5.W,
                         670.  *                               ;DBL BIAS
00002844  0445 0080      671.          SUBI.W #BIAS,D5         ;C(E) IN D5.W,
                         672.  *                               ;SNG BIAS
00002848  6F00 0098      673.          BLE ZERES               ;0 IF TOO SMALL
0000284C  0C45 00FF      674.          CMPI.W #BGEXP,D5        ;TOO BIG?
00002850  6C00 0076      675.          BGE CINFM               ,C=inf IF SO
                         676.  *
                         677.  * CAN PROCEED WITH MULTIPLY OPERATIONS
                         678.  *
00002854  2E04           679.          MOVE L D4,D7      ,(D7) = AhAl
00002856  C8C3           680.          MULU.W D3,D4      ,(D4) = Al*Bl
00002858  2407           681.          MOVE.L D7,D2
0000285A  4842           682.          SWAP.W D2         ;(D2.W) = Ah
0000285C  C4C3           683.          MULU.W D3,D2      ;(D2) = Ah*Bl
0000285E  4843           684.          SWAP.W D3         ;(D3) = BlBh
00002860  3C07           685.          MOVE.W D7,D6      ,(D6.W) = Al
00002862  CCC3           686.          MULU D3,D6        ;(D6) = Al*Bh
00002864  DC82           687.          ADD.L D2,D6       ;(D6) = Al*Bh+Ah*Bl
00002866  2407           688.          MOVE.L D7,D2
00002868  4842           689.          SWAP.W D2         ;(D2.W) = Ah
0000286A  C4C3           690.          MULU.W D3,D2      ;(D2)=Ah*Bh WON'T OVRFL
0000286C  4842           691.          SWAP.W D2
0000286E  4242           692.          CLR.W D2          ;CANT DO ASL #16,D2
00002870  D486           693.          ADD.L D6,D2       ;(D2)=Al*Bh+AhBl
                         694.  *                         ;+(Ah*Bh)SHFT16
00002872  4244           695           CLR.W D4          ,LSW NOT SIGNIFICANT
00002874  4844           696.          SWAP W D4         ;ALIGN FOR ADD
00002876  D484           697.          ADD L D4,D2       ;(D2)=C(F) W/ MORE BITS
                         698.  *
                         699.  * NORMALIZE FRACTION, THEN ROUND
                         700.  *
00002878  0802 001D      701.          BTST.L #NORMTST,D2      ;NORMED ALREADY?
0000287C  66 08          702.          BNE.S NOSHFT            ;
0000287E  E382           703.          ASL.L #1,D2             ;NORMALIZE
00002880  5345           704.          SUBQ.W #1,D5            ,C(E) ADJUST
00002882  4A45           705.          TST.W D5                ;TOO SMALL ?
00002884  67 5C          706.          BEQ.S ZERES             ,WON'T BE <0,
                         707   *                               ;TESTED BEFORE
00002886  0802 0007      708.  NOSHFT  BTST L #MLSB,D2         ;NEED RND UP?
0000288A  66 0A          709.          BNE.S NORNDUP
0000288C  0802 0006      710.          BTST.L #MNTLSB,D2       ,NEED RND UP?
00002890  67 04          711.          BEQ.S NORNDUP
00002892  0002 0080      712.          ORI.B #MRNDDR,D2        ;RND UP,
                         713.  *                               ;SET BIT LSB
00002896  E362           714.  NORNDUP ASL.L #1,D2             ;ALIGN C(F)
00002898  1405           715.          MOVE.B D5,D2            ;INS EXPON C(E)
0000289A  2801           716.  DOSIGN  MOVE.L D1,D4            ;(D4) = AhAl
0000289C  B184           717.          EOR.L D0,D4             ;EOR FOR SIGN
0000289E  0284 80000000  718.          ANDI.L #ABSGN,D4        ;KEEP JUST MSBIT
000028A4  8484           719.          OR.L D4,D2              ;FP RSLT IN D2.L
000028A6  2002           720.          MOVE.L D2,D0            ,RESULT IN D0.L
000028A8  4CDF 00FC      721.  UNSTCKM MOVEM.L (SP)+,D2-D7
000028AC  4E75           722.          RTS
                         723.  *
                         724.  * EXCEPTIONS
                         725.  *
000028AE  2600           726.  AFUNNY  MOVE.L D0,D3
000028B0  0283 7FFFFF00  727.          ANDI.L #$7FFFFF00,D3    ;CLR SCN, EXP
000028B6  67 06          728.          BEQ.S AINF              ;IF ZERO A=inf
000028B8  7000           729.  CNANM   MOVEQ.L #0,D0           ;(A OR B)=NaN?
                         730.  *                               ;THEN FLAG C=NaN
000028BA  5380           731.          SUBQ.L #1,D0            ;C=NaN
000028BC  60 EA          732.          BRA.S UNSTCKM
000028BE  0C01 00FF      733.  AINF    CMPI.B #BGEXP,D1        ;B=inf OR NaN?
000028C2  67 0           734.          BEQ.S BCHECK
000028C4  4A81           735.          TST.L D1                ;B=0?
000028C6  67 F0          736           BEQ.S CNANM             ;Y: inf*0=NaN,
                         737.  *                               ;N: inf
000028C8  7400           738.  CINFM   MOVEQ.L #0,D2           ;C=+-inf
000028CA  143C 00FF      739.          MOVE B #BGEXP,D2        ;EXP=$FF
000028CE  60 CA          740.          BRA DOSIGN              ;SGN A AND B
000028D0  2801           741.  BCHECK  MOVE L D1,D4
```

```
000028D2  0284 7FFFFF00    742.          ANDI.L #$7FFFFF00,D4     ;CLR SGN, EXP
000028D8  67 EE            743.          BEQ.S CINFM              ;IF 0, B=+-inf,
                           744.  *                                ,A=+-inf,C=+-inf
000028DA  60 DC            745.          BRA.S CNANM              ,ELSE B=NaN
                           746.  *                                ,THEN C=NaN
000028DC  0C01 00FF        747.  AZERO   CMPI.B #BGEXP,D1         ,B=inf OR NaN?
000028E0  67 D6            748.          BEQ.S CNANM              ,Y:
                           749.  *                                ;0*(INF OR NaN)
                           750.  *                                ,=NaN
000028E2  7000             751.  ZERES   MOVEQ L #0,D0            ,ELSE 0*VALID=0
000028E4  60 C2            752.          BRA UNSTCKM              ;
000028E6  2801             753.  BFUNNY  MOVE.L D1,D4
000028E8  0284 7FFFFF00    754           ANDI.L #$7FFFFF00,D4     ,CLR SIGN, EXP
000028EE  67 D8            755.          BEQ.S CINFM              ,A VALID,
                           756.  *                                ,B=+-inf
                           757.  *                                ,C=+-inf
000028F0  60 C6            758.          BRA.S CNANM              ;B=NaN, C=NaN
                           759.  *
                           760.  * DONE EXCEPTIONS
                           761   *
                           762.  * SLAVE PROGRAM #1 IS DONE'
                           763   *
                           764.  *
                           765.  ****************************
                           766.  * ALL SLAVES PROG 2
                           767.  ****************************
                           768   *
                           769.  * PROGRAM TO REPORT CONTENTS OF THE
                           770.  * COUNTER REGISTER
                           771.  *
000028F2  207C 00012FD6    772.  SLVPRO2 MOVEA.L #TCR+6,A0        ,POINTER
000028F8  7004             773           MOVEQ L #$4,D0           ,BYTE COUNT
000028FA  4E43             774.          TRAP #SRQASPT            ,ASSERT SRQ
000028FC  13FC 0020        775           MOVE B #ENABLEB,PGCR     ,OUTPUT ENABLED
          00012FC0
00002904  4E46             776           TRAP #OUTDATA            ,SEND BLOCK
00002906  4239 00012FC0    777.          CLR B PGCR
0000290C  4E4E             778           TRAP #ABORT      ,GOTO PROG ACCEPT MODE
0000290E  <1>              779   ENDSLV  DS.B 1
0000290F                   780           END
    0 Errors
```

K.3 Program MATFPFC: Multicomputer Matrix Multiplication: First-Come, First Served data collection

```
                                1.          TTL MATFPFC
                                2.  *
                                3.  * THIS PROGRAM PERFORMS multiplication
                                4.  * of 2 N x N matrices, N divisible by 4
                                5.  * (maximum 20 x 20) on the RMCS.
                                6.  * does C = A x B
                                7.  * this version downloads the common B matrix
                                8.  * to all using broadcast, then loads the
                                9.  * appropriate rows of the A matrix to the
                               10.  * respective slaves, allowing execution to
                               11.  * overlap with loading
                               12.  *
                               13.  * March 3, 1992
                               14.  *
                               15.  * FLOATING POINT INPUT, OUTPUT
                               16.  * executes 256 times, with averages etc
                               17.  * output
                               18.  *
                               19.  * MARCH 7, 1992
                               20.  * THIS VERSION DOES FIRST COME FIRST
                               21.  * SERVE DATA UPLOAD
                               22   *
                               23.  * EQUATES
                               24.  *
# 00012FCC                     25.  PCDR EQU $12FCC
# 00012FC0                     26.  PGCR EQU $12FC0
# 00000010                     27   ENABLEA EQU $10
# 00000020                     28   ENABLEB EQU $20
# 00000003                     29.  SPQASRT EQU 3
# 00000004                     30.  ADBYTIN EQU 4
# 00000007                     31.  ADBYTOT EQU 7
# 00000005                     32.  INDATA  EQU 5
# 00000006                     33   OUTDATA EQU 6
# 00000008                     34.  NETCNF  EQU 8
# 00000009                     35.  SRQSRVC EQU 9
# 0000000E                     36.  ABORT   EQU 14
# 00000001                     37.  SLV0    EQU $01
# 00000002                     38.  SLV1    EQU $02
# 00000004                     39.  SLV2    EQU $04
# 00000008                     40.  SLV3    EQU $08
# 0000000F                     41.  ALLSLV  EQU $0F
# 00012FB0                     42   PSR01   EQU $12FB0
# 00012FB1                     43   PSR23   EQU $12FB1
# 00012FB2                     44   PSR45   EQU $12FB2
# 00012FB3                     45.  PSR67   EQU $12FB3
# 00012FB4                     46.  PSR89   EQU $12FB4
# 00012FB5                     47.  PSR1011 EQU $12FB5
# 00012FB6                     48.  PSR1213 EQU $12FB6
# 00012FB7                     49.  PSR1415 EQU $12FB7
# 00012FB8                     50   PSR1617 EQU $12FB8
# 00012FB9                     51.  PSR1819 EQU $12FB9
# 00012FD0                     52   TCR     EQU $12FD0
# 00012FD2                     53.  TPLR    EQU TCR+2
# 00012FD6                     54.  TCNTR   EQU TCR+6
# 0000000A                     55   SENCLFC EQU 10
# 00000018                     56.  DISBUF8 EQU 24
# 00000016                     57   DISBUF2 EQU 22
# 00FFFFFF                     58.  CNTSTRT EQU $00FFFFFF
# 00000001                     59   ENABLTM EQU 1
# 00000011                     60   OUTMESC EQU 17
# 00000100                     61.  PASSCNT EQU 256
# 00000008                     62.  DIV256  EQU 8
# 00000004                     63.  EOT     EQU 4
# 0000000D                     64.  CR      EQU $0D
# 0000000A                     65.  LF      EQU $0A
# 000000FF                     66.  BGEXP   EQU $FF
# 0000001F                     67.  SGNMSK  EQU 31
# 00000080                     68.  BIAS    EQU 128
# 00000009                     69.  ALSB    EQU 9
# 00000008                     70.  ANTLSB  EQU 8
# 00000020                     71.  ARNDDR  EQU $20
# 80000000                     72.  ABSGN   EQU $80000000
# 00000017                     73.  SIGBITS EQU 23
# 0000001D                     74.  NORMTST EQU 29
# 00000007                     75   MLSB    EQU 7
# 00000006                     76.  MNTLSB  EQU 6
# 00000080                     77.  MRNDDR  EQU $80
```

```
                                78.
                                79.  *
                                80.  ******************
                                81.  * MASTER PROGRAM *
                                82.  ******************
                                83.  *
00000FFE                        84.       ORG.L $1000-2
00000FFE  <2>                   85.  NSPEC    DS B $2 ;MUST HAVE N SPECIFIER HERE
00001000  <640>                 86.  BMATRIX DS B $640
00001640  <640>                 87.  AMATRIX DS B $640
00001C80  <640>                 88.  CMATRIX DS.B $640
000022C0  <4>                   89.  TEMPCNT DS.B $4      ;TEMP FOR SLAVE SELF TIME
000022C4  BD BB BD CB BD        90.  BRDCAST DC B $BD,$BB,$BD,$CB,$BD,$8C,$CB,$8B
          8C CB 8B
000022CC  B8 8B                 91.           DC.B $B8,$8B
                                92.  *
                                93.  * CONFIGURE NETWORK TO BROADCAST STATE, ALL
                                94.  * PROGRAMS EXACTLY THE SAME!
                                95.  * RESPOND TO ALL SLAVE'S RFP SRQ, AND LOAD
                                96.  * THE SLAVE PROGRAM DOWN
                                97.  *
000022CE  287C 00012FD0         98.  STRTMAT MOVEA.L #TCR,A4          ,POINTER TIMER
000022D4  4214                  99.          CLR.B (A4)               ,DISABLE F'SURE
000022D6  2F3C 00FFFFFF        100           MOVE.L #CNTSTRT,-(SP)    ,MIN TIME
000022DC  2F3C 00000000        101.          MOVE.L #$00,-(SP)        ;MAX TIME
000022E2  2F3C 00000000        102.          MOVE.L #$00,-(SP)        ,RUN'G SUM TIME
000022E8  2E3C 000000FF        103.          MOVE.L #PASSCNT-1,D7     ,COUNTER
000022EE  41FA FFD4            104.          LEA BRDCAST(PC),A0
000022F2  4E48                 105.          TRAP #NETCNF
000022F4  103C 000F            106.          MOVE.B #ALLSLV,D0
000022F8  4E49                 107.          TRAP #SRQSRVC
000022FA  41FA 046E            108.          LEA SLVPRO(PC),A0        ,STPT ADDR
000022FE  43FA 06DA            109           LEA SLVPP02(PC),A1       ,END ADDR+1
00002302  93C8                 110.          SUBA.L A0,A1
00002304  3009                 111.          MOVE W A1,D0
00002306  13FC 0020            112           MOVE.B #ENABLEB,PGCR
          00012FC0
0000230E  4E47                 113.          TRAP #ADBYTOT
00002310  4E46                 114           TRAP #OUTDATA
00002312  4239 00012FC0        115.          CLR.B PGCR
                               116.  *
                               117.  * ALL PROGRAMS LOADED, EACH ASSERTING READY FOR
                               118.  * DATA SRQ.  RESPOND TO ALL, THEN DOWNLOAD THE
                               119.  * BMATRIX DATA IN BROADCAST MODE  NOTE THAT THE
                               120.  * BYTE COUNT IS ACTUALLY LONGER TO ALLOW
                               121.  * CONSECUTIVE LOADING OF THE PROPER AMATRIX TO
                               122   * EACH SLAVE FOLLOWING ITS BMATRIX
                               123   *
                               124.  * LOAD TIMER PRE-LOAD REGISTER HERE
                               125.  *
00002318  41FA FFAA            126.  PEEXE    LEA BRDCAST(PC) A0
0000231C  4E48                 127.          TRAP #NETCNF             ,CONFIG BRDCAST
0000231E  297C 00FFFFFF        128.          MOVE.L #CNTSTRT,2(A4)
          0002
00002326  4281                 129.          CLR.L D1
00002328  103C 000F            130.          MOVE.B #ALLSLV,D0
0000232C  4E49                 131.          TRAP #SRQSRVC
0000232E  41FA ECCE            132.          LEA NSPEC(PC),A0         ,START OF DATA
00002332  3210                 133.          MOVE.W (A0),D1  ,D1 HOLDS N
00002334  2401                 134.          MOVE.L D1,D2
00002336  C4C2                 135.          MULU D2,D2       ;D2 HOLDS N**2
00002338  2602                 136.          MOVE.L D2,D3     ,D3=4N**2/4
0000233A  E582                 137.          ASL.L #2,D2      ,D2=4N**2
0000233C  2003                 138.          MOVE.L D3,D0
0000233E  D082                 139.          ADD.L D2,D0      ,D0=4N**2+N**2+2
00002340  5480                 140.          ADDQ.L #2,D0     ;BYTE COUNT TO BRDCAST
                               141.  *
                               142.  * SEND START ADDRESS AND BYTE COUNT TO SLAVES
                               143.  * ENABLE TIMER TOO!
                               144.  *
00002342  18BC 0001            145.  TMSTRT   MOVE.B #ENABLTM,(A4)
00002346  13FC 0020            146.          MOVE.B #ENABLEB,PGCR
          00012FC0
0000234E  4E47                 147.          TRAP #ADBYTOT   ,ADDRESS AND BYTE COUNT
                               148.  *
                               149.  * MODIFY D0 TO HAVE ONLY BMATRIX BYTE COUNT+2
                               150.  * SEND IT OUT...SLAVES WAIT FOR REST
                               151.  *
00002350  2002                 152.          MOVE.L D2,D0
00002352  5480                 153.          ADDQ.L #2,D0
00002354  4E46                 154.          TRAP #OUTDATA   ,SEND OFF THE B MATRIX
```

```
00002356  4239 00012FC0   155.          CLR.B PGCR      ;
                          156.   *
                          157.   * NOW (A0)+(D0) POINTS TO START OF THE AMATRIX,
                          158.   * THE FIRST (4*N**2)/4 ELEMENTS OF WHICH GO TO
                          159.   * SLAVE 0.  CONFIGURE NETWORK
                          160.   *
0000235C  13FC 00BB       161.          MOVE.B #$BB,PSR01       ;TO SLV0 ONLY
          00012FB0
00002364  D1C0            162.          ADDA.L D0,A0
00002366  3003            163.          MOVE.W D3,D0    ;ADDITIONAL BYTE COUNT
00002368  13FC 0020       164.          MOVE.B #ENABLEB,PGCR    ;ENABLE PORT
          00012FC0
00002370  4E46            165.          TRAP #OUTDATA   ;SEND SLAVE 0 PORTION
00002372  4239 00012FC0   166.          CLR.B PGCR      ;
                          167.   *
                          168.   * RESPOND TO SLAVE 0'S RTE SRQ, AND LET EXECUTE
                          169.   *
00002378  103C 0001       170.          MOVE.B #SLV0,D0
0000237C  4E49            171.          TRAP #SRQSRVC
                          172.   *
                          173.   * NEXT CHUNK OF DATA GOES TO SLAVE 1, STARTS
                          174.   * AT A0+D3, AND HAS BYTE COUNT D3
                          175.   * CONFIGURE NETWORK TO SLAVE 1
                          176.   *
0000237E  13FC 00BB       177.          MOVE.B #$BB,PSR45
          00012FB2
00002386  13FC 0004       178.          MOVE.B #$04,PSR01       ;TO SLV1 ONLY
          00012FB0
0000238E  D1C3            179.          ADDA.L D3,A0
00002390  3003            180.          MOVE.W D3,D0
00002392  13FC 0020       181.          MOVE.B #ENABLEB,PGCR
          00012FC0
0000239A  4E46            182.          TRAP #OUTDATA
0000239C  4239 00012FC0   183.          CLR.B PGCR
                          184.   *
                          185.   * RESPOND TO SLAVE 1'S RTE SRQ, AND LET EXECUTE
                          186.   *
000023A2  103C 0002       187.          MOVE.B #SLV1,D0
000023A6  4E49            188.          TRAP #SRQSRVC
                          189.   *
                          190.   * NEXT CHUNK OF DATA GOES TO SLAVE 2, STARTS
                          191.   * AT A0+D3, AND HAS BYTE COUNT D3
                          192.   * CONFIGURE NETWORK TO SLAVE 2
                          193.   *
000023A8  13FC 00BB       194           MOVE.B #$BB,PSR89
          00012FB4
000023B0  13FC 0004       195           MOVE.B #$04,PSR45       ;TO SLV2 ONLY
          00012FB2
000023B8  D1C3            196           ADDA.L D3,A0
000023BA  3003            197.          MOVE.W D3,D0
000023BC  13FC 0020       198.          MOVE.B #ENABLEB,PGCR
          00012FC0
000023C4  4E46            199           TRAP #OUTDATA
000023C6  4239 00012FC0   200.          CLR.B PGCR
                          201.   *
                          202.   * RESPOND TO SLAVE 2'S RTE SRQ, AND LET EXECUTE
                          203.   *
000023CC  103C 0004       204.          MOVE.B #SLV2,D0
000023D0  4E49            205.          TRAP #SRQSRVC
                          206.   *
                          207   * NEXT CHUNK OF DATA GOES TO SLAVE 3, STARTS
                          208.   * AT A0+D3, AND HAS BYTE COUNT D3
                          209.   * CONFIGURE NETWORK TO SLAVE 3
                          210.   *
000023D2  13FC 0004       211.          MOVE.B #$04,PSR89       ;TO SLV3 ONLY
          00012FB4
000023DA  D1C3            212.          ADDA.L D3,A0
000023DC  3003            213.          MOVE.W D3,D0
000023DE  13FC 0020       214.          MOVE.B #ENABLEB,PGCR
          00012FC0
000023E6  4E46            215.          TRAP #OUTDATA
000023E8  4239 00012FC0   216.          CLR.B PGCR
                          217.   *
                          218.   * RESPOND TO SLAVE 3'S RTE SRQ, AND LET EXECUTE
                          219.   *
000023EE  103C 0008       220.          MOVE.B #SLV3,D0
000023F2  4E49            221.          TRAP #SRQSRVC
                          222.   *
                          223.   * ALL DATA IS LOADED DOWN TO SLAVES, AND ARE
                          224.   * RESPOND TO INDIVIDUAL SLAVE SRQ'S AND UPLOAD
                          225.   * THE RESULTS TO THE CMATRIX ARRAY
```

```
                                226.   * BYTE COUNT FROM EACH SLAVE WILL BE
                                227.   * ((4*N**2)/4)=D3
                                228.   * USE FIRST COM FIRST SERVED STRATEGY
                                229.   *
                                230    * PUT (A5)=PSR01
                                231.   * AND (A6)=PGCR
                                232.   *
000023F4  2A7C 00012FB0         233            MOVEA.L #PSR01,A5
000023FA  2C7C 00012FC0         234.           MOVEA.L #PGCR,A6
00002400  700F                  235.           MOVEQ.L #ALLSLV,D0
00002402  0800 0000             236.   TSTB0   BTST.L #0,D0
00002406  67 44                 237.           BEQ S TSTB1
00002408  123C 0000             238.           MOVE.B #0,D1
0000240C  6100 0276             239.           BSR TSTSLV
00002410  0801 0007             240            BTST.L #7,D1     ;SERVICED?
00002414  67 36                 241            BEQ S TSTB1      ;IF NOT GOTO SLV1
                                242.   *
                                243.   * SLAVE 0 SENSED AND ACKNOWLEDGED, GET DATA
                                244    *
00002416  1B7C 0080 0003        245            MOVE.B #$80,3(A5)
0000241C  1B7C 0080 0005        246.           MOVE.B #$80,5(A5)
00002422  1B7C 0080 0007        247.           MOVE.B #$80,7(A5)
00002428  1B7C 0080 0009        248.           MOVE.B #$80,9(A5)
0000242E  3F00                  249.           MOVE.W D0,-(SP)
00002430  3003                  250            MOVE.W D3,D0        ,BYTE COUNT
00002432  41FA F84C             251.           LEA CMATRIX(PC),A0      ,DEST ADDRESS
00002436  1CBC 0010             252            MOVE.B #ENABLEA,(A6)
0000243A  4E45                  253            TRAP #INDATA
0000243C  4216                  254            CLR B (A6)
0000243E  301F                  255            MOVE W (SP)+,D0
00002440  0880 0000             256            BCLR L #0,D0       LEAR THE BIT
00002444  0C00 0000             257            CMPI.B #0,D0
00002448  6700 00DA             258.           BEQ DONEDAT
0000244C  0800 0001             259    TSTB1   BTST.L #1,D0
00002450  67 40                 260            BEQ S TSTB2
00002452  123C 0001             261            MOVE B #1,D1
00002456  6100 0220             262            BSR TSTSLV
0000245A  0801 0007             263            BTST.L #7,D1     ,SERVICED?
0000245E  67 32                 264.           BEQ S TSTB2      ,IF NOT GOTO SLV1
                                265    *
                                266    * SLAVE 1 SENSED AND ACKNOWLEDGED, GET DATA
                                267    *
00002460  1B7C 00B8 0005        268            MOVE B #$B8,5(A5)
00002466  1B7C 0080 0007        269            MOVE.B #$80,7(A5)
0000246C  1B7C 0080 0009        270            MOVE.B #$80,9(A5)
00002472  3F00                  271            MOVE.W D0,-(SP)
00002474  3003                  272            MOVE.W D3,D0        ,BYTE COUNT
00002476  41FA F808             273            LEA CMATRIX(PC),A0
0000247A  D1C3                  274            ADDA.L D3,A0     ,DEST ADDRESS
0000247C  1CBC 0010             275            MOVE.B #ENABLEA,(A6)
00002480  4E45                  276            TRAP #INDATA
00002482  4216                  277            CLR.B (A6)
00002484  301F                  278.           MOVE.W (SP)+,D0
00002486  0880 0001             279            BCLR.L #1,D0     ,CLEAR THE BIT
0000248A  0C00 0000             280.           CMPI.B #0,D0
0000248E  6700 0094             281            BEQ DONEDAT
00002492  0800 0002             282    TSTB2   BTST.L #2,D0
00002496  67 42                 283.           BEQ S T TB3
00002498  123C 0002             284            MOVE.B #2,D1
0000249C  6100 01E6             285            BSR TSTSLV
000024A0  0801 0007             286.           BTST.L #7,D1     ;SERVICED?
000024A4  67 34                 287.           BEQ S TSTB3      ;IF NOT GOTO SLV1
                                288.   *
                                289.   * SLAVE 2 SENSED AND ACKNOWLEDGED, GET DATA
                                290.   *
000024A6  1B7C 0080 0006        291            MOVE.B #$80,6(A5)
000024AC  1B7C 00BB 0007        292.           MOVE.B #$BB,7(A5)
000024B2  1B7C 0080 0009        293            MOVE.B #$80,9(A5)
000024B8  3F00                  294.           MOVE.W D0,-(SP)
000024BA  3003                  295.           MOVE.W D3,D0        ,BYTE COUNT
000024BC  41FA F7C2             296.           LEA CMATRIX(PC),A0
000024C0  D1C3                  297.           ADDA.L D3,A0
000024C2  D1C3                  298.           ADDA.L D3,A0     ,DEST ADDRESS
000024C4  1CBC 0010             299.           MOVE.B #ENABLEA,(A6)
000024C8  4E45                  300            TRAP #INDATA
000024CA  4216                  301.           CLR B (A6)
000024CC  301F                  302.           MOVE.W (SP)+,D0
000024CE  0880 0002             303.           BCLR.L #2,D0     ,CLEAR THE BIT
000024D2  0C00 0000             304.           CMPI.B #0,D0
000024D6  6700 004C             305.           BEQ DONEDAT
000024DA  0800 0003             306.   TSTB3   BTST.L #3,D0
```

```
000024DE  6700 FF22          307.          BEQ TSTB0
000024E2  123C 0003          308.          MOVE.B #3,D1
000024E6  6100 019C          309.          BSR TSTSLV
000024EA  0801 0007          310.          BTST.L #7,D1      ;SERVICED?
000024EE  6700 FF12          311.          BEQ TSTB0         ;IF NOT GOTO SLV1
                             312.  *
                             313.  * SLAVE 3 SENSED AND ACKNOWLEDGED, GET DATA
                             314.  *
000024F2  1B7C 00B8 0008     315.          MOVE.B #$B8,8(A5)
000024F8  1B7C 00BB 0009     316.          MOVE.B #$BB,9(A5)
000024FE  3F00               317.          MOVE.W D0,-(SP)
00002500  3003               318.          MOVE.W D3,D0          ;BYTE COUNT
00002502  41FA F77C          319.          LEA CMATRIX(PC),A0
00002506  D1C3               320.          ADDA.L D3,A0
00002508  D1C3               321.          ADDA.L D3,A0
0000250A  D1C3               322.          ADDA.L D3,A0      ;DEST ADDRESS
0000250C  1CBC 0010          323.          MOVE.B #ENABLEA,(A6)
00002510  4E45               324.          TRAP #INDATA
00002512  4216               325.          CLR.B (A6)
00002514  301F               326.          MOVE.W (SP)+,D0
00002516  0880 0003          327.          BCLR.L #3,D0      ,CLEAR THE BIT
0000251A  0C00 0000          328.          CMPI.B #0,D0
0000251E  67 04              329.          BEQ.S DONEDAT
00002520  6000 FEE0          330.          BRA TSTB0
                             331.  *
                             332.  * ALL DATA IS LOADED
                             333.  *
                             334.  *
                             335.  * PROGRAM IS DONE, CAN STOP TIMER, AND
                             336.  * OUTPUT NUMBER OF 32 CLOCK INTERVALS
                             337.  * PASSED. FIRST DISABLE TIMER
                             338.  *
00002524  4214               339.  DONEDAT CLR.B (A4)
00002526  222C 0006          340.          MOVE.L 6(A4),D1 ;GET NEW COUNT
0000252A  203C 00FFFFFF      341.          MOVE.L #CNTSTRT,D0
00002530  9081               342.          SUB.L D1,D0       ,GET ELAPSED COUNT
                             343.  *
                             344.  * OUTPUT CR, LF, THEN THE COUNT IN HEX
                             345.  *
00002532  4E4F               346.          TRAP #15
00002534  000A               347.          DC.W SENCLFC
00002536  4E4F               348.          TRAP #15
00002538  0018               349.          DC.W DISBUF8
                             350.  *
                             351.  * NOW KEEP TRACK OF STATISTICS
                             352.  *
0000253A  D197               353.          ADD.L D0,(SP)     ;ADD TO SUM
0000253C  B0AF 0004          354.          CMP.L 4(SP),D0    ;CHECK MAX
00002540  6F 04              355           BLE.S NOTMAX      ,D0 NOT MAX
00002542  2F40 0004          356.          MOVE.L D0,4(SP)   ;NEW MAX
00002546  B0AF 0008          357.  NOTMAX  CMP.L 8(SP),D0    ;CHECK MIN
0000254A  6C 04              358.          BGE.S NOTMIN      ;D0 NOT MIN
0000254C  2F40 0008          359.          MOVE.L D0,8(SP)   ;NEW MIN
00002550  51CF FDC6          360.  NOTMIN  DBRA D7,REEXE     ;DO D7 TIMES
                             361.  *
                             362.  * WHEN HERE, PROGRAM HAS BEEN EXECUTED 256 TIMES
                             363.  * SEND AN ILLEGAL ADDRESS FOR DATA, FORCING
                             364.  * SLAVES TO PROGRAM ACCEPT MODE
                             365.  *
00002554  41FA FD6E          366.          LEA BRDCAST(PC),A0
00002558  4E48               367.          TRAP #NETCNF
0000255A  103C 000F          368.          MOVE.B #ALLSLV,D0
0000255E  4E49               369.          TRAP #SRQSRVC
00002560  207C FFFF0000      370.          MOVEA.L #$FFFF0000,A0   ;ILLEGAL ADDR
00002566  13FC 0020          371.          MOVE.B #ENABLEB,PGCR
          00012FC0
0000256E  4E47               372.          TRAP #ADBYTOT     ;THAT'LL FIX 'EM
00002570  4239 00012FC0      373.          CLR.B PGCR
                             374.  *
                             375.  * OUTPUT STATISTICS
                             376.  *
00002576  4E4F               377.          TRAP #15
00002578  000A               378.          DC.W SENCLFC
0000257A  4E4F               379.          TRAP #15
0000257C  000A               380.          DC.W SENCLFC
0000257E  41FA 013E          381.          LEA AVEMES(PC),A0
00002582  4E4F               382.          TRAP #15
00002584  0011               383.          DC.W OUTMESC
00002586  201F               384.          MOVE.L (SP)+,D0 ;GET SUM
00002588  E080               385.          ASR.L #DIV256,D0        ;DO AVERAGE
0000258A  4E4F               386.          TRAP #15
```

```
0000258C  0018             387           DC.W DISBUF8     ;OUTPUT AVERAGE
0000258E  4E4F             388.          TRAP #15
00002590  000A             389           DC.W SENCLFC
00002592  41FA 0151        390.          LEA MAXMES(PC),A0
00002596  4E4F             391.          TRAP #15
00002598  0011             392           DC.W OUTMESC
0000259A  201F             393.          MOVE.L (SP)+,D0 ;GET MAX
0000259C  4E4F             394           TRAP #15
0000259E  0018             395.          DC.W DISBUF8
000025A0  4E4F             396           TRAP #15
000025A2  000A             397           DC.W SENCLFC
000025A4  41FA 0166        398.          LEA MINMES(PC),A0
000025A8  4E4F             399           TRAP #15
000025AA  0011             400.          DC.W OUTMESC
000025AC  201F             401           MOVE.L (SP)+,D0 ;GET MIN
000025AE  4E4F             402           TRAP #15
000025B0  0018             403           DC.W DISBUF8
000025B2  4E4F             404           TRAP #15
000025B4  000A             405           DC.W SENCLFC
                           406   *
                           407   * AFTER 256 RUNS, LOAD THE SECOND SLAVE
                           408   * PROGRAM TO ACQUIRE THE TIMING DATA
                           409   * IN THE SLAVES TIMER COUNT REGISTERS
                           410   *
000025B6  41FA FD0C        411           LEA BRDCAST(PC),A0       ;BROADCAST MODE
000025BA  4E48             412           TRAP #NETCNF
                           413   *
                           414.  * RESPOND TO SLAVES RFP
                           415   *
000025BC  103C 000F        416           MOVE.B #ALLSLV,D0
000025C0  4E49             417           TRAP #SRQSRVC
000025C2  41FA 0416        418           LEA SLVPRO2(PC),A0
000025C6  43FA 042E        419           LEA ENDSLV(PC),A1
000025CA  93C8             420           SUBA.L A0,A1
000025CC  3009             421           MOVE W A1,D0
000025CE  13FC 0020        422           MOVE B #ENABLEB,PGCR
          00012FC0
000025D6  4E47             423           TRAP #ADBYTOT
000025D8  4E46             424           TRAP #OUTDATA
000025DA  4239 00012FC0    425           CLR B PGCR       ,PROGRAM SENT
                           426   *
                           427   * RESPOND TO SLAVES RFD
                           428   *
000025E0  103C 000F        429           MOVE.B #ALLSLV,D0
000025E4  4E49             430           TRAP #SRQSRVC
                           431   *
                           432   * PROGRAM HAS NO INPUT DATA, SEND DUMMY
                           433   * ADDRESS AND ZERO BYTE COUNT
                           434   * AND THEN NO DATA
                           435   *
000025E6  4240             436.          CLR.W D0
000025E8  13FC 0020        437           MOVE.B #ENABLEB,PGCR
          00012FC0
000025F0  4E47             438           TRAP #ADBYTOT
000025F2  4239 00012FC0    439           CLR.B PGCR
                           440.  *
                           441   * RESPOND TO READY TO EXECUTE
                           442   *
000025F8  103C 000F        443           MOVE B #ALLSLV,D0
000025FC  4E49             444.          TRAP #SRQSRVC
                           445   *
                           446.  * SLAVES EXECUTE AND ASSERT AN SRQ WHEN
                           447   * READY TO REPORT. RESPOND ONE AT A TIME
                           448   * DETERMINE TIME ELAPSED AND REPORT TO
                           449   * CONSOLE
                           450   *
000025FE  103C 0001        451           MOVE.B #SLV0,D0 ;SLAVE 0
00002602  4281             452           CLR.L D1
00002604  61 32            453.          BSR.S GETSLTM
00002606  13FC 00BC        454.          MOVE.B #$BC,PSR1011 ;SLAVE 1
          00012FB5
0000260E  5281             455.          ADDQ.L #1,D1
00002610  103C 0002        456           MOVE.B #SLV1,D0
00002614  61 22            457.          BSR.S GETSLTM
00002616  13FC 00BB        458           MOVE.B #$BB,PSR1415 ;SLAVE 2
          00012FB7
0000261E  5281             459.          ADDQ.L #1,D1
00002620  103C 0004        460           MOVE.B #SLV2,D0
00002624  61 12            461           BSR S GETSLTM
00002626  13FC 00BB        462           MOVE.B #$BB,PSR1819 ,SLAVE3
          00012FB9
```

```
0000262E  5281              463.          ADDQ.L #1,D1
00002630  103C 0008         464.          MOVE.B #SLV3,D0
00002634  61 02             465.          BSR.S GETSLTM
                            466.  *
                            467.  * NOW FINALLY DONE!
                            468.  *
00002636  4E75              469.          RTS
                            470.  *
                            471.  * SUBROUTINE TO GET TIMING DATA FROM SLAVE
                            472.  * SPECIFIED IN D1.B, WITH MASK IN D0.B
                            473.  *
00002638  48E7 C0C0         474.  GETSLTM MOVEM.L D0-D1/A0-A1,-(SP) ;SAVE ENV
0000263C  4E49              475.          TRAP #SRQSRVC
0000263E  7004              476.          MOVEQ.L #$4,D0  ;BYTE COUNT
00002640  41FA FC7E         477.          LEA TEMPCNT(PC),A0
00002644  13FC 0010         478.          MOVE.B #ENABLEA,PGCR     ;ENABLE INPUT
          00012FC0
0000264C  4E45              479.          TRAP #INDATA
0000264E  4239 00012FC0     480.          CLR.B PGCR
00002654  2248              481.          MOVEA.L A0,A1
00002656  41FA 00DB         482.          LEA MESSA(PC),A0
0000265A  4E4F              483.          TRAP #15
0000265C  0011              484.          DC.W OUTMESC
0000265E  1001              485.          MOVE.B D1,D0
00002660  4E4F              486.          TRAP #15
00002662  0016              487.          DC.W DISBUF2
00002664  41FA 00E7         488.          LEA MESSB(PC),A0
00002668  4E4F              489.          TRAP #15
0000266A  0011              490.          DC.W OUTMESC
0000266C  2211              491.          MOVE.L (A1),D1
0000266E  203C 00FFFFFF     492.          MOVE.L #CNTSTRT,D0 ;CALC ELAPSED
00002674  9081              493.          SUB.L D1,D0
00002676  4E4F              494.          TRAP #15
00002678  0018              495.          DC.W DISBUF8
0000267A  4E4F              496.          TRAP #15
0000267C  000A              497.          DC.W SENCLFC
0000267E  4CDF 0303         498.          MOVEM.L (SP)+,D0-D1/A0-A1
00002682  4E75              499.          RTS
                            500
                            501.  * SUBROUTINE TSTSLV
                            502.  * SLAVE NUMBER TO TEST IS IN D1.B
                            503.  * IF ASSERTING AN SRQ, THEN ACKNOWLEDGED AND
                            504.  * HIGH BIT IN D1.B IS SET, ELSE CLEAR
                            505.  *
00002684  48E7 8080         506.  TSTSLV  MOVEM.L D0/A0,-(SP)
00002688  207C 00012FCC     507.          MOVEA.L #PCDR,A0
0000268E  0810 0000         508.          BTST.B #0,(A0)
00002692  66 22             509.          BNE.S NOINT
00002694  E509              510.          LSL.B #2,D1
00002696  08C1 0004         511.          BSET.L #4,D1
0000269A  1081              512.          MOVE.B D1,(A0)
0000269C  0890 0004         513.          BCLR.B #4,(A0)
000026A0  1010              514.          MOVE.B (A0),D0
000026A2  08D0 0004         515.          BSET.B #4,(A0)
000026A6  0800 0001         516.          BTST.L #1,D0
000026AA  66 0A             517.          BNE.S NOINT
000026AC  123C 00FF         518.          MOVE.B #$FF,D1
000026B0  4CDF 0101         519.          MOVEM.L (SP)+,D0/A0
000026B4  4E75              520.          RTS
000026B6  4201              521.  NOINT   CLR.B D1
000026B8  4CDF 0101         522.          MOVEM.L (SP)+,D0/A0
000026BC  4E75              523.          RTS
                            524.
                            525.  *
                            526.  *
000026BE  41 76 65 72 61    527.  AVEMES  DC.B 'Average time ( x 4 for microseco'
          67 65 20 74 69
          6D 65 20 28 20
          78 20 34 20 66
          6F 72 20 6D 69
          63 72 6F 73 65
          63 6F
000026DE  6E 64 73 29 3D    528.          DC.B 'nds)= ',EOT
          20 04
000026E5  4D 61 78 69 6D    529.  MAXMES  DC.B 'Maximum time ( x 4 for microseco'
          75 6D 20 74 69
          6D 65 20 28 20
          78 20 34 20 66
```

```
          6F 72 20 6D 69
          63 72 6F 73 65
          63 6F
00002705  6E 64 73 29 3D  530.          DC.B 'nds)= ',EOT
          20 04
0000270C  4D 69 6E 69 6D  531.  MINMES  DC.B 'Minimum time ( x 4 for microseco'
          75 6D 20 74 69
          6D 65 20 28 20
          78 20 34 20 66
          6F 72 20 6D 69
          63 72 6F 73 65
          63 6F
0000272C  6E 64 73 29 3D  532.          DC.B 'nds)= ',EOT
          20 04
00002733  45 78 65 63 75  533.  MESSA   DC.B 'Execution time by slave #',EOT
          74 69 6F 6E 20
          74 69 6D 65 20
          62 79 20 73 6C
          61 76 65 20 23
          04
0000274D  20 20 28 20 78  534.  MESSB   DC.B '   ( x 4 for microseconds) = ',EOT
          20 34 20 66 6F
          72 20 6D 69 63
          72 6F 73 65 63
          6F 6E 64 73 29
          20 3D 20 04
                          535.  ********************
                          536.  * ALL SLAVE'S PROGRAM *
                          537.  ********************
                          538.  *
                          539.  * HAVE LOADED N AT NSPEC, FOLLOWED BY BMATRIX
                          540.  * AND THE AMATRIX. BMATRIX IS N x N, AND WORD
                          541.  * LENGTH ELEMENTS. BMATRIX STARTS AT NSPEC+2
                          542.  * AND THE AMATRIX STARTS AT BMATRIX+(2*N**2)
                          543.  * ONCE DISCOVERED, CAN USE THE MATRIX
                          544.  * MULTIPLY SIMILAR TO THAT IN MESC
                          545.  *
0000276A  4239 00012FD0   546.  SLVPRO  CLR.B TCR        ;DISABLE TIMER
00002770  23FC 00FFFFFF   547.          MOVE.L #CNTSTRT,TCR+2 ,LOAD MAX COUNT
          00012FD2
0000277A  13FC 00 ..      548.          MOVE.B #ENABLTM,TCR     ,ENABLE TIMER
          00012FD0
00002782  4286            549.          CLR L D6
00002784  43FA E87A       550.          LEA BMATRIX(PC),A1
00002788  3C29 FFFE       551.          MOVE W -2(A1),D6        ,D6 HOLDS N
0000278C  3A06            552.          MOVE.W D6,D5
0000278E  CAC5            553.          MULU D5,D5        ;D5=N**2
00002790  2405            554.          MOVE L D5,D2      ,D2=N**2
00002792  E585            555.          ASL.L #2,D5       ;D5=4N**2
00002794  2A45            556.          MOVEA.L D5,A5     ;
00002796  9BFC 00000004   557.          SUBA.L #4,A5      ;A5=4N**2-4
0000279C  2049            558.          MOVE.L A1,A0
0000279E  D1C5            559.          ADDA.L D5,A0      ;A0 POINTS TO AMATRIX
000027A0  45FA F4DE       560.          LEA CMATRIX(PC),A2      ,CMATRIX P  NTEF
                          561.  *
                          562.  *
                          563.  * HERE, DO (N/4 x N) X (N x N) MATRIX MULTIPLY
                          564.  *
                          565.  *
000027A4  D5C2            566.  MATRIX  ADDA.L D2,A2     ;A2=END OF C=A2+N**2
000027A6  5982            567.          SUBQ L #4,D2      ;D2=N**2-4
000027A8  264D            568.          MOVEA.L A5,A3     ,A3=4N**2-4
000027AA  2606            569.          MOVE L D6,D3      ,D3=N
000027AC  E583            570.          ASL.L #2,D3       ,D3=4N
000027AE  5386            571.          SUBQ.L #1,D6      ;D6=N-1
000027B0  2E06            572.          MOVE.L D6,D7      ;D7=N-1
000027B2  2846            573.          MOVEA.L D6,A4     ;A4=N-1
000027B4  4285            574.  OUTRLP  CLR.L D5         ;HOLDS RESULT OF ADDS
000027B6  2802            575.          MOVE.L D2,D4
000027B8  2C4D            576.          MOVE.L A5,A6
000027BA  2030 4800       577.  INRLP   MOVE.L 0(A0,D4 L),D0    ;GET A(I,J)
000027BE  2231 E800       578.          MOVE.L 0(A1,A6.L),D1    ,DO MUL B(J,K)
000027C2  6100 012E       579.          BSR MULTFP              ,RESULT IN D0
000027C6  2205            580.          MOVE.L D5,D1
000027C8  6100 0046       581.          BSR ADDFP
000027CC  2A00            582.          MOVE.L D0,D5
000027CE  5984            583.          SUBQ.L #4,D4          ,D4=D4-4
000027D0  9DC3            584.          SUBA L D3,A6          ;D5=D5-D3
000027D2  51CE FFE6       585.          DBRA D6,INRLP         ;DO N TIMES
000027D6  2505            586.          MOVE.L D5,-(A2)       ,D0=N-1 AGAIN
000027D8  2C0C            587.          MOVE.L A4,D6
```

```
000027DA  9BFC 00000004   588.          SUBA.L #4,A5
000027E0  51CF FFD2       589.          DBRA D7,OUTRLP        ;DO N TIMES
000027E4  2A4B            590.          MOVE.L A3,A5          ;A5=2*N**2-2
000027E6  2E0C            591.          MOVE.L A4,D7          ;D7=N-1
000027E8  9483            592.          SUB.L D3,D2           ;D2=D2-D3
000027EA  6C C8           593.          BGE.S OUTRLP
                          594.  *
                          595.  * CALCULATION DONE, REST IS OVERHEAD FOR UPLOADING
                          596.  * STOP TIMER HERE
                          597.  *
000027EC  4239 00012FD0   598.          CLR.B TCR
                          599.  *
000027F2  41FA F48C       600.          LEA CMATRIX(PC),A0    ;POINT TO OUTPUT
000027F6  303A E806       601.          MOVE.W NSPEC(PC),D0   ;GET N AGAIN
000027FA  C1C0            602.          MULS D0,D0
000027FC  4E43            603.          TRAP #SRQASRT         ;ASSERT SRQ
000027FE  13FC 0020       604.          MOVE.B #ENABLEB,PGCR
          00012FC0
00002806  4E46            605.          TRAP #OUTDATA
00002808  4239 00012FC0   606.          CLR.B PGCR
0000280E  4E75            607.          RTS
                          608.  *
                          609.  * SUBROUTINE ADDFP
                          610.  *
                          611.  * FLOATING POINT ADDITION FOR 32-BIT
                          612.  * FLOATING POINT FORMAT:
                          613.  * SIGN (1)| FRACTION (23) | EXPONENT (8)
                          614.  * WHERE EXPONENT IS BIASED EXCESS 128;
                          615.  * FRACTION IS NORMALIZED
                          616.  * 0 = $00000000
                          617.  * (SIGN) INFINTY = $(8.OR.0)00000FF
                          618.  * NaN = $XXXXXXFF
                          619.  *       WHERE XXXXXX IS NOT AS INFINITY
                          620.  *
                          621.  * DOES A*B=C WHERE
                          622.  * A IS IN D0
                          623.  * B IS IN D1
                          624.  * C RETURNS IN D0
                          625.  * IF A OR B = NaN, SETS C=NaN = $FFFFFFFF
                          626.  * IF A OR B = +-INFINITY, SET C=NaN
                          627.  *
00002810  48E7 3F00       628.  ADDFP   MOVEM.L D2-D7,-(SP)    ;SAVE OFF
00002814  0C00 00FF       629.          CMPI.B #BGEXP,D0      ;A=NaN OR inf?
00002818  6700 00BC       630.          BEQ CNANAD            ;C=NaN IF SO
0000281C  0C01 00FF       631.          CMPI.B #BGEXP,D1      ;B=NaN OR inf?
00002820  6700 00B4       632           BEQ CNANAD            ;C=NaN IF SO
00002824  4A81            633           TST.L D1              ;B=0?
00002826  6700 0096       634.          BEQ UNSTCKA           ;C=A IF SO
0000282A  4A80            635.          TST.L D0              ;A=0?
0000282C  6700 00AE       636.          BEQ CISB              ;C=B IF SO
00002830  4243            637.          CLR.W D3
00002832  4244            638.          CLR.W D4
00002834  1600            639.          MOVE.B D0,D3          ;(D3.W)=EXPA
00002836  1801            640.          MOVE.B D1,D4          ;(D4.W)=EXPB
00002838  9644            641.          SUB.W D4,D3           ;(D3.W)=SHFTCNT
0000283A  0C43 0017       642.          CMPI.W #SIGBITS,D3    ;TOO BIG? C=A
0000283E  6C00 007E       643.          BGE UNSTCKA
00002842  0C43 FFE9       644.          CMPI.W #-SIGBITS,D3   ;TOO SMALL? C=B
00002846  6F00 0094       645.          BLE CISB
                          646.  *
                          647.  * HERE, MUST DO ADD
                          648.  * D5, D6 GET FRACTIONS; D7 CARRIES SIGN
                          649.  *
0000284A  7E00            650.          MOVEQ.L #0,D7
0000284C  0800 001F       651.          BTST.L #SGNMSK,D0     ;SIGN OF A
00002850  67 04           652.          BEQ.S POSTVE
00002852  08C7 001F       653.          BSET.L #SGNMSK,D7     ;C IS NEG
00002856  2A00            654.  POSTVE  MOVE.L D0,D5          ;(D5)=AF
00002858  2C01            655.          MOVE.L D1,D6          ;(D6)=BF
0000285A  4205            656.          CLR.B D5
0000285C  4206            657.          CLR.B D6              ;KILL EXPS
0000285E  E38D            658.          LSL.L #1,D5
00002860  E38E            659.          LSL.L #1,D6           ;ALIGN
00002862  7400            660.          MOVEQ.L #0,D2         ;
00002864  1400            661.          MOVE.B D0,D2          ;CEXP=AEXP
00002866  4A43            662.          TST.W D3              ;TEST SC
00002868  67 06           663.          BEQ.S NOADJ           ;IF=0 NO SHFTNG
0000286A  6B00 0074       664.          BMI SHFTA             ;NEG, SHFT A
0000286E  E6AE            665.  SHFTB   LSR.L D3,D6           ;SHIFTB
00002870  2600            666.  NOADJ   MOVE.L D0,D3          ;COPY OF A
00002872  B383            667           EOR.L D1,D3           ;SIGNS
```

```
00002874  0803 001F      668.          BTST.L #SGNMSK,D3          ;
00002878  6700 0052      669.          BEQ SAMSGN                 ;EQUAL SIGNS?
0000287C  4486           670.  DIFFSGN NEG.L D6                   ;NEG B
0000287E  DC85           671.          ADD.L D5,D6                ;D6 HAS THE SUM
00002880  6500 0042      672.          BCS NCMPC                  ,IF CAR=0, NOT=0
00002884  0847 001F      673.  COMPC   BCHG.L #SGNMSK,D7          ;COMPLEMENT SUM
00002888  4486           674.          NEG.L D6
0000288A  0806 001F      675.  NORMC   BTST.L #SGNMSK,D6          ;NORMALIZED?
0000288E  66 06          676.          BNE.S CNORMD
00002890  E38E           677.          LSL.L #1,D6                ;ADJUST FRACTION
00002892  5342           678.          SUBQ.W #1,D2               ;ADJUST EXP
00002894  60 F4          679.          BRA.S NORMC
                         680.  *
                         681.  * BOUND CHECK ON EXP
                         682.  *
00002896  4A42           683.  CNORMD  TST.W D2
00002898  6F00 002E      684.          BLE CZERO
0000289C  0C42 00FF      685.          CMPI.W #BGEXP,D2
000028A0  6700 0046      686.          BEQ CINFAD
                         687.  *
                         688.  * ROUND OFF FRACTION
                         689.  *
000028A4  0806 0009      690.  ROUNDC  BTST.L #ALSB,D6
000028A8  66 0A          691.          BNE.S NOROUN
000028AA  0806 0008      692.          BTST.L #ANTLSB,D6
000028AE  67 04          693.          BEQ.S NOROUN
000028B0  0046 0020      694.          ORI.W #ARNDDR,D6
                         695.  *
                         696.  * ALIGN RESULT
                         697.  *
000028B4  E28E           698.  NOROUN  LSR.L #1,D6                 ;ROOM FOR SIGN
000028B6  4206           699.          CLR.B D6                   ,ROOM FOR EXP
                         700.  *
                         701.  * WITH BOUND CHECKS ON EXP, HAS LEADING ZEROES
                         702.  *
000028B8  8486           703.          OR.L D6,D2
000028BA  8487           704.          OR.L D7,D2
000028BC  2002           705.          MOVE.L D2,D0
000028BE  4CDF 00FC      706.  UNSTCKA MOVEM L (SP)+,D2-D7
000028C2  4E75           707.          RTS
                         708.  *
000028C4  4A86           709.  NCMPC   TST.L D6                   ;C=0?
000028C6  66 C2          710.          BNE.S NORMC                ,NO, NORMALIZE
000028C8  4280           711.  CZERO   CLR.L D0
000028CA  60 F2          712.          BRA UNSTCKA
000028CC  DC85           713.  SAMSGN  ADD.L D5,D6                ;D6 = SUM FRACT
000028CE  64 D4          714.          BCC.S ROUNDC
000028D0  E296           715.          ROXR.L #1,D6               ,IF CARRY,
000028D2  5242           716.          ADDQ.W #1,D2               ;ADJUST RESULT
000028D4  60 C0          717.          BRA.S CNORMD               ,GO ROUND
000028D6  7000           718.  CNANAD  MOVEQ.L #0,D0
000028D8  5380           719.          SUBQ.L #1,D0               ;C=NaN
000028DA  60 E2          720.          BRA UNSTCKA
000028DC  2001           721.  CISB    MOVE.L D1,D0
000028DE  60 DE          722.          BRA UNSTCKA
000028E0  4443           723.  SHFTA   NEG.W D3                   ;POS SHIFT CNT
000028E2  E6AD           724.          LSR.L D3,D5                ;ADJUST A FRACT
000028E4  1401           725.          MOVE.B D1,D2               ,USE BEXP=CEXP
000028E6  60 88          726.          BRA NOADJ
000028E8  7000           727.  CINFAD  MOVEQ.L #0,D0
000028EA  103C 00FF      728.          MOVE.B #BGEXP,D0           ;C=inf
000028EE  8087           729.          OR.L D7,D0                 ;TAKE SIGN
000028F0  60 CC          730.          BRA UNSTCKA                ;QUIT
                         731.  *
                         732.  *
                         733.  * SUBROUTINE MULTFP
                         734.  *
                         735.  * FLOATING POINT MULTIPLICATION FOR 32-BIT
                         736.  * FLOATING POINT FORMAT:
                         737.  * SIGN (1)| FRACTION (23) | EXPONENT (8)
                         738.  * WHERE EXPONENT IS BIASED EXCESS 128,
                         739.  * FRACTION IS NORMALIZED
                         740.  * 0 = $00000000
                         741.  * (SIGN) INFINTY = $(8.OR.0)0000FF
                         742.  * NaN = $XXXXXXXF
                         743.  *       WHERE XXXXXX IS NOT AS INFINITY
                         744.  *
                         745.  * DOES A*B=C WHERE
                         746.  * A IS IN D0
                         747.  * B IS IN D1
                         748.  * C RETURNS IN D0
```

```
                                749.  * IF A OR B = NaN, SETS C=NaN = $FFFFFFFF
                                750.  * IF A OR B = +-INFINITY, SET C=+-INFINITY
                                751.  *
000028F2  48E7 3F00             752.  MULTFP  MOVEM.L D2-D7,-(SP)      ;SAVE OFF
000028F6  0C00 00FF             753.         CMPI.B #BGEXP,D0         ;A=NaN OR inf?
000028FA  6700 009A             754.         BEQ AFUNNY               ;SERVICE IF SO
000028FE  4A80                  755.         TST.L D0                 ;A=0?
00002900  6700 00C2             756.         BEQ AZERO
00002904  0C01 00FF             757.         CMPI.B #BGEXP,D1         ;B=NaN OR inf?
00002908  6700 00C4             758.         BEQ BFUNNY               ;SERVICE IF SO
0000290C  4A81                  759.         TST.L D1                 ;B=0?
0000290E  6700 00BA             760.         BEQ ZERES                ;A IS VALID, B=0
                                761.  *                                ;THEN C=0
00002912  2800                  762.         MOVE.L D0,D4             ;D4 HAS A
00002914  2601                  763.         MOVE.L D1,D3             ;D3 HAS B
00002916  0884 001F             764.         BCLR.L #SGNMSK,D4        ;KILL SIGN OF A
0000291A  0883 001F             765.         BCLR.L #SGNMSK,D3        ;KILL SIGN OF B
0000291E  E08C                  766.         LSR.L #8,D4              ;D4 HAS A(F)
00002920  E08B                  767.         LSR.L #8,D3              ;D3 HAS B(F)
00002922  4245                  768.         CLR.W D5                 ;EXP CALC
00002924  4246                  769.         CLR.W D6
00002926  1A00                  770.         MOVE.B D0,D5             ;A(E) IN D5.W
00002928  1C01                  771.         MOVE.B D1,D6             ;B(E) IN D6.W
0000292A  DA46                  772.         ADD.W D6,D5              ;C(E) IN D5.W,
                                773.  *                                ;DBL BIAS
0000292C  0445 0080             774.         SUBI.W #BIAS,D5          ;C(E) IN D5.W,
                                775.  *                                ;SNG BIAS
00002930  6F00 0098             776.         BLE ZERES                ;0 IF TOO SMALL
00002934  0C45 00FF             777.         CMPI.W #BGEXP,D5         ;TOO BIG?
00002938  6C00 0076             778.         BGE CINFM                ;C=inf IF SO
                                779.  *
                                780.  * CAN PROCEED WITH MULTIPLY OPERATIONS
                                781.  *
0000293C  2E04                  782.         MOVE.L D4,D7     ;(D7) = AhAl
0000293E  C8C3                  783.         MULU.W D3,D4     ;(D4) = Al*Bl
00002940  2407                  784.         MOVE.L D7,D2
00002942  4842                  785.         SWAP.W D2        ;(D2.W) = Ah
00002944  C4C3                  786.         MULU.W D3,D2     ;(D2) = Ah*Bl
00002946  4843                  787.         SWAP.W D3        ;(D3) = BlBh
00002948  3C07                  788.         MOVE.W D7,D6     ;(D6.W) = Al
0000294A  CCC3                  789.         MULU D3,D6       ;(D6) = Al*Bh
0000294C  DC82                  790.         ADD.L D2,D6      ;(D6) = Al*Bh+Ah*Bl
0000294E  2407                  791.         MOVE.L D7,D2
00002950  4842                  792.         SWAP.W D2        ;(D2.W) = Ah
00002952  C4C3                  793.         MULU.W D3,D2     ;(D2)=Ah*Bh WON'T OVFL
00002954  4842                  794.         SWAP.W D2
00002956  4242                  795.         CLR.W D2         ;CANT DO ASL #16,D2
00002958  D486                  796.         ADD.L D6,D2      ;(D2)=Al*Bh+AhBl
                                797.  *                       ;+(Ah*Bh)SHFT16
0000295A  4244                  798.         CLR.W D4         ;LSW NOT SIGNIFICANT
0000295C  4844                  799.         SWAP.W D4        ;ALIGN FOR ADD
0000295E  D484                  800.         ADD.L D4,D2      ;(D2)=C(F) W/ MORE BITS
                                801.  *
                                802.  * NORMALIZE FRACTION, THEN ROUND
                                803.  *
00002960  0802 001D             804.         BTST.L #NORMTST,D2       ;NORMED ALREADY?
00002964  66 08                 805.         BNE.S NOSHFT             ;
00002966  E382                  806.         ASL.L #1,D2              ;NORMALIZE
00002968  5345                  807.         SUBQ.W #1,D5             ;C(E) ADJUST
0000296A  4A45                  808.         TST.W D5                 ;TOO SMALL ?
0000296C  67 5C                 809.         BEQ.S ZERES              ;WON'T BE <0,
                                810.  *                               ;TESTED BEFORE
0000296E  0802 0007             811.  NOSHFT BTST.L #MLSB,D2          ;NEED RND UP?
00002972  66 0A                 812.         BNE.S NORNDUP
00002974  0802 0006             813.         BTST.L #MNTLSB,D2        ;NEED RND UP?
00002978  67 04                 814.         BEQ.S NORNDUP
0000297A  0002 0080             815.         ORI.B #MRNDDR,D2         ;RND UP,
                                816.  *                               ;SET BIT LSB
0000297E  E382                  817.  NORNDUP ASL.L #1,D2              ;ALIGN C(F)
00002980  1405                  818.         MOVE.B D5,D2             ;INS EXPON C(E)
00002982  2801                  819.  DOSIGN  MOVE.L D1,D4             ;(D4) = AhAl
00002984  B184                  820.         EOR.L D0,D4              ;EOR FOR SIGN
00002986  0284 80000000         821.         ANDI.L #ABSGN,D4         ;KEEP JUST MSBIT
0000298C  8484                  822.         OR.L D4,D2               ;FP RSLT IN D2.L
0000298E  2002                  823.         MOVE.L D2,D0             ;RESULT IN D0.L
00002990  4CDF 00FC             824.  UNSTCKM MOVEM.L (SP)+,D2-D7
00002994  4E75                  825.         RTS
                                826.  *
                                827.  * EXCEPTIONS
                                828.  *
00002996  2600                  829.  AFUNNY  MOVE.L D0,D3
```

```
00002998  0283 7FFFFF00   830.          ANDI.L #$7FFFFF00,D3     ;CLR SGN, EXP
0000299E  67 06           831.          BEQ.S AINF               ;IF ZERO A=inf
000029A0  7000            832.  CNANM   MOVEQ.L #0,D0            ;(A OR B)=NaN?
                          833.  *                                ;THEN FLAG C=NaN
000029A2  5380            834.          SUBQ.L #1,D0             ;C=NaN
000029A4  60 EA           835.          BRA.S UNSTCKM
000029A6  0C01 00FF       836.  AINF    CMPI.B #BGEXP,D1         ;B=inf OR NaN?
000029AA  67 0C           837.          BEQ.S BCHECK
000029AC  4A81            838.          TST.L D1                 ;B=0?
000029AE  67 F0           839.          BEQ.S CNANM              ;Y: inf*0=NaN,
                          840.  *                                ;N: inf
000029B0  7400            841.  CINFM   MOVEQ.L #0,D2            ;C=+-inf
000029B2  143C 00FF       842.          MOVE.B #BGEXP,D2         ;EXP=$FF
000029B6  60 CA           843.          BRA DOSIGN               ;SGN A AND B
000029B8  2801            844.  BCHECK  MOVE.L D1,D4
000029BA  0284 7FFFFF00   845.          ANDI.L #$7FFFFF00,D4     ;CLR SGN, EXP
000029C0  67 EE           846.          BEQ.S CINFM              ;IF 0, B=+-inf,
                          847.  *                                ;A=+-inf,C=+-inf
000029C2  60 DC           848.          BRA.S CNANM              ;ELSE B=NaN:
                          849.  *                                ;THEN C=NaN
000029C4  0C01 00FF       850.  AZERO   CMPI.B #BGEXP,D1         ;B=inf OR NaN?
000029C8  67 D6           851.          BEQ.S CNANM              ;Y:
                          852.  *                                ;0*(INF OR NaN)
                          853.  *                                ;=NaN
000029CA  7000            854.  ZERES   MOVEQ.L #0,D0            ;ELSE 0*VALID=0
000029CC  60 C2           855.          BRA UNSTCKM              ;
000029CE  2801            856.  BFUNNY  MOVE.L D1,D4
000029D0  0284 7FFFFF00   857.          ANDI.L #$7FFFFF00,D4     ;CLR SIGN, EXP
000029D6  67 D8           858.          BEQ.S CINFM              ;A VALID;
                          859.  *                                ;B=+-inf
                          860.  *                                ;C=+-inf
000029D8  60 C6           861.          BRA.S CNANM              ;B=NaN: C=NaN
                          862.  *
                          863.  * DONE EXCEPTIONS
                          864.  *
                          865.  * SLAVE PROGRAM #1 IS DONE!
                          866.  *
                          867.  *
                          868.  ****************************
                          869.  * ALL SLAVES PROG 2
                          870.  ****************************
                          871.  *
                          872.  * PROGRAM TO REPORT CONTENTS OF THE
                          873.  * COUNTER REGISTER
                          874.  *
000029DA  207C 00012FD6   875.  SLVPRO2 MOVEA.L #TCR+6,A0        ;POINTER
000029E0  7004            876.          MOVEQ.L #$4,D0           ;BYTE COUNT
000029E2  4E43            877.          TRAP #SRQASRT            ;ASSERT SRQ
000029E4  13FC 0020       878.          MOVE.B #ENABLEB,PGCR     ;OUTPUT ENABLED
          00012FC0
000029EC  4E46            879.          TRAP #OUTDATA            ;SEND BLOCK
000029EE  4239 00012FC0   880.          CLR.B PGCR
000029F4  4E4E            881.          TRAP #ABORT      ;GOTO PROG ACCEPT MODE
000029F6  <1>             882.  ENDSLV  DS.B 1
000029F7                  883.          END
    0 Errors
```

K.4 Input Data: FPINDAT1

```
                                    1.          TTL FPINDAT1
                                    2.  *
                                    3.  * INPUT DATA FILE FOR 20 x 20 MATRIX
                                    4.  * ALL ELEMENTS $00000000
                                    5.  *
     # 00000000                     6.  POS     EQU $00000000
                                    7.  *
00000000                            8.          ORG.L $0000
00000000  0000000(                  9.          DC.L POS,POS,POS,POS,POS,POS,POS,POS
          00000000
          00000000
          00000000
          00000000
          00000000
          00000000
          00000000
          00000000
00000020  00000000                 10.          DC.L POS,POS,POS,POS,POS,POS,POS,POS
          00000000
          00000000
          00000000
          00000000
          00000000
          00000000
          00000000
00000040  00000000                 11.          DC.L POS,POS,POS,POS,POS,POS,POS,POS
          00000000
          00000000
          00000000
          00000000
          00000000
          00000000
          00000000
00000060  00000000                 12.          DC.L POS,POS,POS,POS,POS,POS,POS,POS
          00000000
          00000000
          00000000
          00000000
          00000000
          00000000
          00000000
00000080  00000000                 13.          DC.L POS,POS,POS,POS,POS,POS,POS,POS
          00000000
          00000000
          00000000
          00000000
          00000000
          00000000
          00000000
000000A0  00000000                 14.          DC.L POS,POS,POS,POS,POS,POS,POS,POS
          00000000
          00000000
          00000000
          00000000
          00000000
          00000000
          00000000
000000C0  00000000                 15.          DC.L POS,POS,POS,POS,POS,POS,POS,POS
          00000000
          00000000
          00000000
          00000000
          00000000
          00000000
          00000000
000000E0  00000000                 16.          DC.L POS,POS,POS,POS,POS,POS,POS,POS
          00000000
          00000000
          00000000
          00000000
          00000000
          00000000
          00000000
00000100  00000000                 17.          DC.L POS,POS,POS,POS,POS,POS,POS,POS
          00000000
          00000000
          00000000
          00000000
          00000000
          00000000
```

```
                00000000
                00000000
00000120        00000000        18.     DC.L POS,POS,POS,POS,POS,POS,POS,POS
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
00000140        00000000        19.     DC.L POS,POS,POS,POS,POS,POS,POS,POS
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
00000160        00000000        20.     DC.L POS,POS,POS,POS,POS,POS,POS,POS
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
00000180        00000000        21.     DC.L POS,POS,POS,POS,POS,POS,POS,POS
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
000001A0        00000000        22.     DC.L POS,POS,POS,POS,POS,POS,POS,POS
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
000001C0        00000000        23.     DC.L POS,POS,POS,POS,POS,POS,POS,POS
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
000001E0        00000000        24.     DC.L POS,POS,POS,POS,POS,POS,POS,POS
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
00000200        00000000        25.     DC.L POS,POS,POS,POS,POS,POS,POS,POS
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
00000220        00000000        26.     DC.L POS,POS,POS,POS,POS,POS,POS,POS
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
00000240        00000000        27.     DC.L POS,POS,POS,POS,POS,POS,POS,POS
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
```

```
                00000000
00000260        00000000        28.             DC.L POS,POS,POS,POS,POS,POS,POS,POS
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
00000280        00000000        29.             DC.L POS,POS,POS,POS,POS,POS,POS,POS
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
000002A0        00000000        30.             DC.L POS,POS,POS,POS,POS,POS,POS,POS
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
000002C0        00000000        31.             DC.L POS,POS,POS,POS,POS,POS,POS,POS
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
000002E0        00000000        32.             DC.L POS,POS,POS,POS,POS,POS,POS,POS
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
00000300        00000000        33.             DC.L POS,POS,POS,POS,POS,POS,POS,POS
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
00000320        00000000        34.             DC.L POS,POS,POS,POS,POS,POS,POS,POS
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
00000340        00000000        35.             DC.L POS,POS,POS,POS,POS,POS,POS,POS
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
00000360        00000000        36.             DC.L POS,POS,POS,POS,POS,POS,POS,POS
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
00000380        00000000        37.             DC.L POS,POS,POS,POS,POS,POS,POS,POS
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
```

```
000003A0 00000000          38.          DC.L POS,POS,POS,POS,POS,POS,POS,POS
         00000000
         00000000
         00000000
         00000000
         00000000
         00000000
         00000000
         00000000
000003C0 00000000          39.          DC.L POS,POS,POS,POS,POS,POS,POS,POS
         00000000
         00000000
         00000000
         00000000
         00000000
         00000000
         00000000
         00000000
000003E0 00000000          40.          DC.L POS,POS,POS,POS,POS,POS,POS,POS
         00000000
         00000000
         00000000
         00000000
         00000000
         00000000
         00000000
         00000000
00000400 00000000          41.          DC.L POS,POS,POS,POS,POS,POS,POS,POS
         00000000
         00000000
         00000000
         00000000
         00000000
         00000000
         00000000
         00000000
00000420 00000000          42.          DC.L POS,POS,POS,POS,POS,POS,POS,POS
         00000000
         00000000
         00000000
         00000000
         00000000
         00000000
         00000000
         00000000
00000440 00000000          43.          DC.L POS,POS,POS,POS,POS,POS,POS,POS
         00000000
         00000000
         00000000
         00000000
         00000000
         00000000
         00000000
         00000000
00000460 00000000          44.          DC.L POS,POS,POS,POS,POS,POS,POS,POS
         00000000
         00000000
         00000000
         00000000
         00000000
         00000000
         00000000
         00000000
00000480 00000000          45.          DC.L POS,POS,POS,POS,POS,POS,POS,POS
         00000000
         00000000
         00000000
         00000000
         00000000
         00000000
         00000000
         00000000
000004A0 00000000          46.          DC.L POS,POS,POS,POS,POS,POS,POS,POS
         00000000
         00000000
         00000000
         00000000
         00000000
         00000000
         00000000
         00000000
000004C0 00000000          47.          DC.L POS,POS,POS,POS,POS,POS,POS,POS
         00000000
         00000000
         00000000
         00000000
         00000000
         00000000
         00000000
         00000000
000004E0 00000000          48.          DC.L POS,POS,POS,POS,POS,POS,POS,POS
```

```
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
00000500        00000000        49.             DC.L POS,POS,POS,POS,POS,POS,POS,POS
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
00000520        00000000        50.             DC.L POS,POS,POS,POS,POS,POS,POS,POS
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
00000540        00000000        51.             DC.L POS,POS,POS,POS,POS,POS,POS,POS
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
00000560        00000000        52.             DC.L POS,POS,POS,POS,POS,POS,POS,POS
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
00000580        00000000        53.             DC.L POS,POS,POS,POS,POS,POS,POS,POS
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
000005A0        00000000        54.             DC.L POS,POS,POS,POS,POS,POS,POS,POS
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
000005C0        00000000        55.             DC.L POS,POS,POS,POS,POS,POS,POS,POS
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
000005E0        00000000        56.             DC.L POS,POS,POS,POS,POS,POS,POS,POS
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
00000600        00000000        57.             DC.L POS,POS,POS,POS,POS,POS,POS,POS
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
00000620        00000000        58.             DC.L POS,POS,POS,POS,POS,POS,POS,POS
                00000000
```

```
           00000000
           00000000
           00000000
           00000000
           00000000
           00000000
00000640                    59.          END
    0 Errors
```

K.5 Input Data: FPINDAT2

```
                                    1.          TTL FPINDAT2
                                    2.  *
                                    3.  * INPUT DATA FILE FOR 20 x 20 MATRIX
                                    4.  * ALL ELEMENTS $7FFFFF80
                                    5.  *
     # 7FFFFF80                     6.  POS     EQU $7FFFFF80
                                    7.  *
00000000                            8.          ORG.L $0000
00000000  7FFFFF80                  9.          DC.L POS,POS,POS,POS,POS,POS,POS,POS
          7FFFFF80
          7FFFFF80
          7FFFFF80
          7FFFFF80
          7FFFFF80
          7FFFFF80
          7FFFFF80
          7FFFFF80
00000020  7FFFFF80                 10.          DC.L POS,POS,POS,POS,POS,POS,POS,POS
          7FFFFF80
          7FFFFF80
          7FFFFF80
          7FFFFF80
          7FFFFF80
          7FFFFF80
          7FFFFF80
00000040  7FFFFF80                 11.          DC.L POS,POS,POS,POS,POS,POS,POS,POS
          7FFFFF80
          7FFFFF80
          7FFFFF80
          7FFFFF80
          7FFFFF80
          7FFFFF80
          7FFFFF80
00000060  7FFFFF80                 12.          DC L POS,POS,POS,POS,POS,POS,POS,POS
          7FFFFF80
          7FFFFF80
          7FFFFF80
          7FFFFF80
          7FFFFF80
          7FFFFF80
          7FFFFF80
00000080  7FFFFF80                 13.          DC L POS,POS,POS,POS,POS,POS,POS,POS
          7FFFFF80
          7FFFFF80
          7FFFFF80
          7FFFFF80
          7FFFFF80
          7FFFFF80
          7FFFFF80
000000A0  7FFFFF80                 14.          DC.L POS,POS,POS,POS,POS,POS,POS,POS
          7FFFFF80
          7FFFFF80
          7FFFFF80
          7FFFFF80
          7FFFFF80
          7FFFFF80
          7FFFFF80
000000C0  7FFFFF80                 15.          DC L POS,POS,POS,POS,POS,POS,POS,POS
          7FFFFF80
          7FFFFF80
          7FFFFF80
          7FFFFF80
          7FFFFF80
          7FFFFF80
          7FFFFF80
000000E0  7FFFFF80                 16.          DC.L POS,POS,POS,POS,POS,POS,POS,POS
          7FFFFF80
          7FFFFF80
          7FFFFF80
          7FFFFF80
          7FFFFF80
          7FFFFF80
          7FFFFF80
00000100  7FFFFF80                 17.          DC.L POS,POS,POS,POS,POS,POS,POS,POS
          7FFFFF80
          7FFFFF80
          7FFFFF80
          7FFFFF80
          7FFFFF80
```

```
                7FFFFF80
                7FFFFF80
00000120        7FFFFF80        18.         DC.L POS,POS,POS,POS,POS,POS,POS,POS
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
00000140        7FFFFF80        19.         DC.L POS,POS,POS,POS,POS,POS,POS,POS
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
00000160        7FFFFF80        20.         DC.L POS,POS,POS,POS,POS,POS,POS,POS
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
00000180        7FFFFF80        21.         DC.L POS,POS,POS,POS,POS,POS,POS,POS
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
000001A0        7FFFFF80        22.         DC.L POS,POS,POS,POS,POS,POS,POS,POS
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
000001C0        7FFFFF80        23.         DC.L POS,POS,POS,POS,POS,POS,POS,POS
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
000001E0        7FFFFF80        24.         DC.L POS,POS,POS,POS,POS,POS,POS,POS
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
00000200        7FFFFF80        25.         DC.L POS,POS,POS,POS,POS,POS,POS,POS
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
00000220        7FFFFF80        26.         DC.L POS,POS,POS,POS,POS,POS,POS,POS
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
00000240        7FFFFF80        27.         DC.L POS,POS,POS,POS,POS,POS,POS,POS
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
```

```
                7FFFFF80
00000260        7FFFFF80        28.     DC.L POS,POS,POS,POS,POS,POS,POS,POS
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
00000280        7FFFFF80        29.     DC.L POS,POS,POS,POS,POS,POS,POS,POS
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
000002A0        7FFFFF80        30.     DC.L POS,POS,POS,POS,POS,POS,POS,POS
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
000002C0        7FFFFF80        31.     DC.L POS,POS,POS,POS,POS,POS,POS,POS
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
000002E0        7FFFFF80        32.     DC.L POS,POS,POS,POS,POS,POS,POS,POS
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
00000300        7FFFFF80        33.     DC L POS,POS,POS,POS,POS,POS,POS,POS
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
00000320        7FFFFF80        34.     DC.L POS,POS,POS,POS,POS,POS,POS,POS
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
00000340        7FFFFF80        35.     DC.L POS,POS,POS,POS,POS,POS,POS,POS
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
00000360        7FFFFF80        36.     DC.L POS,POS,POS,POS,POS,POS,POS,POS
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
00000380        7FFFFF80        37.     DC.L POS,POS,POS,POS,POS,POS,POS,POS
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
```

```
000003A0 7FFFFF80          38.          DC.L POS,POS,POS,POS,POS,POS,POS,POS
         7FFFFF80
         7FFFFF80
         7FFFFF80
         7FFFFF80
         7FFFFF80
         7FFFFF80
         7FFFFF80
000003C0 7FFFFF80          39.          DC.L POS,POS,POS,POS,POS,POS,POS,POS
         7FFFFF80
         7FFFFF80
         7FFFFF80
         7FFFFF80
         7FFFFF80
         7FFFFF80
         7FFFFF80
000003E0 7FFFFF80          40.          DC.L POS,POS,POS,POS,POS,POS,POS,POS
         7FFFFF80
         7FFFFF80
         7FFFFF80
         7FFFFF80
         7FFFFF80
         7FFFFF80
         7FFFFF80
00000400 7FFFFF80          41.          DC.L POS,POS,POS,POS,POS,POS,POS,POS
         7FFFFF80
         7FFFFF80
         7FFFFF80
         7FFFFF80
         7FFFFF80
         7FFFFF80
         7FFFFF80
00000420 7FFFFF80          42.          DC.L POS,POS,POS,POS,POS,POS,POS,POS
         7FFFFF80
         7FFFFF80
         7FFFFF80
         7FFFFF80
         7FFFFF80
         7FFFFF80
         7FFFFF80
00000440 7FFFFF80          43.          DC.L POS,POS,POS,POS,POS,POS,POS,POS
         7FFFFF80
         7FFFFF80
         7FFFFF80
         7FFFFF80
         7FFFFF80
         7FFFFF80
         7FFFFF80
00000460 7FFFFF80          44.          DC.L POS,POS,POS,POS,POS,POS,POS,POS
         7FFFFF80
         7FFFFF80
         7FFFFF80
         7FFFFF80
         7FFFFF80
         7FFFFF80
         7FFFFF80
00000480 7FFFFF80          45.          DC.L POS,POS,POS,POS,POS,POS,POS,POS
         7FFFFF80
         7FFFFF80
         7FFFFF80
         7FFFFF80
         7FFFFF80
         7FFFFF80
         7FFFFF80
000004A0 7FFFFF80          46.          DC.L POS,POS,POS,POS,POS,POS,POS,POS
         7FFFFF80
         7FFFFF80
         7FFFFF80
         7FFFFF80
         7FFFFF80
         7FFFFF80
         7FFFFF80
000004C0 7FFFFF80          47.          DC.L POS,POS,POS,POS,POS,POS,POS,POS
         7FFFFF80
         7FFFFF80
         7FFFFF80
         7FFFFF80
         7FFFFF80
         7FFFFF80
         7FFFFF80
000004E0 7FFFFF80          48.          DC.L POS,POS,POS,POS,POS,POS,POS,POS
```

```
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
00000500        7FFFFF80        49.         DC.L POS,POS,POS,POS,POS,POS,POS,POS
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
00000520        7FFFFF80        50.         DC L POS,POS,POS,POS,POS,POS,POS,POS
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
00000540        7FFFFF80        51.         DC.L POS,POS,POS,POS,POS,POS,POS,POS
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
00000560        7FFFFF80        52.         DC.L POS,POS,POS,POS,POS,POS,POS,POS
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
00000580        7FFFFF80        53.         DC.L POS,POS,POS,POS,POS,POS,POS,POS
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
000005A0        7FFFFF80        54.         DC.L POS,POS,POS,POS,POS,POS,POS,POS
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
000005C0        7FFFFF80        55.         DC.L POS,POS,POS,POS,POS,POS,POS,POS
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
000005E0        7FFFFF80        56.         DC.L POS,POS,POS,POS,POS,POS,POS,POS
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
00000600        7FFFFF80        57.         DC.L POS,POS,POS,POS,POS,POS,POS,POS
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
00000620        7FFFFF80        58.         DC.L POS,POS,POS,POS,POS,POS,POS,POS
                7FFFFF80
```

```
           7FFFFF80
           7FFFFF80
           7FFFFF80
           7FFFFF80
           7FFFFF80
           7FFFFF80
00000640                    59.          END
    0 Errors
```

K.6 Input Data: FPINDAT3

```
                                    1.          TTL FPINDAT3
                                    2.  *
                                    3.  * INPUT DATA FILE FOR 20 x 20 MATRIX
                                    4.  * ALL ELEMENTS $7FFFFF80 OR $FFFFFF80
                                    5.  * FOR WORST CASE MULTIPLY OF DATA
                                    6.  *
     # FFFFFF80                     7.  NEG     EQU $FFFFFF80
     # 7FFFFF80                     8.  POS     EQU $7FFFFF80
                                    9.  *
00000000                           10.          ORG.L $0000
00000000  7F FFF80                 11.          DC.L POS,POS,POS,POS,POS,POS,POS,POS
          7FFFFF80
          7FFFFF80
          7FFFFF80
          7FFFFF80
          7FFFFF30
          7FFFFF80
          7FFFFF80
          7FFFFF80
00000020  FFFFFF80                 12.          DC.L NEG,NEG,NEG,NEG,NEG,NEG,NEG,NEG
          FFFFFF80
          FFFFFF80
          FFFFFF80
          FFFFFF80
          FFFFFF80
          FFFFFF80
          FFFFFF80
00000040  7FFFFF80                 13.          DC.L POS,NEG,POS,NEG,POS,NEG,POS,NEG
          FFFFFF80
          7FFFFF80
          FFFFFF80
          7FFFFF80
          FFFFFF80
          7FFFFF80
          FFFFFF80
00000060  FFFFFF80                 14.          DC.L NEG,POS,NEG,POS,NEG,POS,NEG,POS
          7FFFFF80
          FFFFFF80
          7FFFFF80
          FFFFFF80
          7FFFFF80
          FFFFFF80
          7FFFFF80
00000080  7FFFFF80                 15.          DC.L POS,POS,NEG,NEG,POS,POS,NEG,NEG
          7FFFFF80
          FFFFFF80
          FFFFFF80
          7FFFFF80
          7FFFFF80
          FFFFFF80
          FFFFFF80
000000A0  FFFFFF80                 16.          DC.L NEG,NEG,POS,POS,NEG,NEG,POS,POS
          FFFFFF80
          7FFFFF80
          7FFFFF80
          FFFFFF80
          FFFFFF80
          7FFFFF80
          7FFFFF80
000000C0  7FFFFF80                 17.          DC.L POS,POS,POS,POS,NEG,NEG,NEG,NEG
          7FFFFF80
          7FFFFF80
          7FFFFF80
          FFFFFF80
          FFFFFF80
          FFFFFF80
          FFFFFF80
000000E0  FFFFFF80                 18.          DC.L NEG,NEG,NEG,NEG,POS,POS,POS,POS
          FFFFFF80
          FFFFFF80
          FFFFFF80
          7FFFFF80
          7FFFFF80
          7FFFFF80
          7FFFFF80
00000100  7FFFFF80                 19.          DC.L POS,POS,POS,POS,POS,POS,POS,POS
          7FFFFF80
          7FFFFF80
          7FFFFF80
```

```
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
00000120        FFFFFF80        20.     DC.L NEG,NEG,NEG,NEG,NEG,NEG,NEG,NEG
                FFFFFF80
                FFFFFF80
                FFFFFF80
                FFFFFF80
                FFFFFF80
                FFFFFF80
                FFFFFF80
                FFFFFF80
00000140        7FFFFF80        21.     DC.L POS,NEG,POS,NEG,POS,NEG,POS,NEG
                FFFFFF80
                7FFFFF80
                FFFFFF80
                7FFFFF80
                FFFFFF80
                7FFFFF80
                FFFFFF80
00000160        FFFFFF80        22.     DC.L NEG,POS,NEG,POS,NEG,POS,NEG,POS
                7FFFFF80
                FFFFFF80
                7FFFFF80
                FFFFFF80
                7FFFFF80
                FFFFFF80
                7FFFFF80
00000180        7FFFFF80        23.     DC.L POS,POS,NEG,NEG,POS,POS,NEG,NEG
                7FFFFF80
                FFFFFF80
                FFFFFF80
                7FFFFF80
                7FFFFF80
                FFFFFF80
                FFFFFF80
000001A0        FFFFFF80        24.     DC.L NEG,NEG,POS,POS,NEG,NEG,POS,POS
                FFFFFF80
                7FFFFF80
                7FFFFF80
                FFFFFF80
                FFFFFF80
                7FFFFF80
                7FFFFF80
000001C0        7FFFFF80        25.     DC.L POS,POS,POS,POS.NEG,NEG,NEG,NEG
                7FFFFF80
                7FFFFF80
                7FFFFF80
                FFFFFF80
                FFFFFF80
                FFFFFF80
                FFFFFF80
000001E0        FFFFFF80        26.     DC.L NEG,NEG,NEG,NEG,POS,POS,POS,POS
                FFFFFF80
                FFFFFF80
                FFFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
00000200        7FFFFF80        27.     DC.L POS,POS,POS,POS,POS,POS,POS,POS
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
00000220        FFFFFF80        28.     DC.L NEG,NEG,NEG,NEG,NEG,NEG,NEG,NEG
                FFFFFF80
                FFFFFF80
                FFFFFF80
                FFFFFF80
                FFFFFF80
                FFFFFF80
                FFFFFF80
00000240        7FFFFF80        29.     DC.L POS,NEG,POS,NEG,POS,NEG,POS,NEG
                FFFFFF80
                7FFFFF80
                FFFFFF80
                7FFFFF80
```

```
                FFFFFF80
                7FFFFF80
                FFFFFF80
00000260        FFFFFF80        30.     DC.L NEG,POS,NEG,POS,NEG,POS,NEG,POS
                7FFFFF80
                FFFFFF80
                7FFFFF80
                FFFFFF80
                7FFFFF80
                FFFFFF80
                7FFFFF80
00000280        7FFFFF80        31.     DC.L POS,POS,NEG,NEG,POS,POS,NEG,NEG
                7FFFFF80
                FFFFFF80
                FFFFFF80
                7FFFFF80
                7FFFFF80
                FFFFFF80
                FFFFFF80
000002A0        FFFFFF80        32.     DC.L NEG,NEG,POS,POS,NEG,NEG,POS,POS
                FFFFFF80
                7FFFFF80
                7FFFFF80
                FFFFFF80
                FFFFFF80
                7FFFFF80
                7FFFFF80
000002C0        7FFFFF80        33.     DC.L POS,POS,POS,POS,NEG,NEG,NEG,NEG
                7FFFFF80
                7FFFFF80
                7FFFFF80
                FFFFFF80
                FFFFFF80
                FFFFFF80
                FFFFFF80
000002E0        FFFFFF80        34.     DC.L NEG,NEG,NEG,NEG,POS,POS,POS,POS
                FFFFFF80
                FFFFFF80
                FFFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
00000300        7FFFFF80        35.     DC.L POS,POS,POS,POS,POS,POS,POS,POS
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
00000320        FFFFFF80        36      DC.L NEG,NEG,NEG,NEG,NEG,NEG,NEG,NEG
                FFFFFF80
                FFFFFF80
                FFFFFF80
                FFFFFF80
                FFFFFF80
                FFFFFF80
                FFFFFF80
00000340        7FFFFF80        37.     DC.L POS,NEG,POS,NEG,POS,NEG,POS,NEG
                FFFFFF80
                7FFFFF80
                FFFFFF80
                7FFFFF80
                FFFFFF80
                7FFFFF80
                FFFFFF80
00000360        FFFFFF80        38.     DC.L NEG,POS,NEG,POS,NEG,POS,NEG,POS
                7FFFFF80
                FFFFFF80
                7FFFFF80
                FFFFFF80
                7FFFFF80
                FFFFFF80
                7FFFFF80
00000380        7FFFFF80        39.     DC.L POS,POS,NEG,NEG,POS,POS,NEG,NEG
                7FFFFF80
                FFFFFF80
                FFFFFF80
                7FFFFF80
                7FFFFF80
```

```
                FFFFFF80
                FFFFFF80
000003A0        FFFFFF80        40.             DC.L NEG,NEG,POS,POS,NEG,NEG,POS,POS
                FFFFFF80
                7FFFFF80
                7FFFFF80
                FFFFFF80
                FFFFFF80
                7FFFFF80
                7FFFFF80
000003C0        7FFFFF80        41.             DC.L POS,POS,POS,POS,NEG,NEG,NEG,NEG
                7FFFFF80
                7FFFFF80
                7FFFFF80
                FFFFFF80
                FFFFFF80
                FFFFFF80
                FFFFFF80
000003E0        FFFFFF80        42.             DC.L NEG,NEG,NEG,NEG,POS,POS,POS,POS
                FFFFFF80
                FFFFFF80
                FFFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
00000400        7FFFFF80        43.             DC.L POS,POS,POS,POS,POS,POS,POS,POS
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
00000420        FFFFFF80        44.             DC.L NEG,NEG,NEG,NEG,NEG,NEG,NEG,NEG
                FFFFFF80
                FFFFFF80
                FFFFFF80
                FFFFFF80
                FFFFFF80
                FFFFFF80
                FFFFFF80
00000440        7FFFFF80        45.             DC.L POS,NEG,POS,NEG,POS,NEG,POS,NEG
                FFFFFF80
                7FFFFF80
                FFFFFF80
                7FFFFF80
                FFFFFF80
                7FFFFF80
                FFFFFF80
00000460        FFFFFF80        46.             DC.L NEG,POS,NEG,POS,NEG,POS,NEG,POS
                7FFFFF80
                FFFFFF80
                7FFFFF80
                FFFFFF80
                7FFFFF80
                FFFFFF80
                7FFFFF80
00000480        7FFFFF80        47.             DC.L POS,POS,NEG,NEG,POS,POS,NEG,NEG
                7FFFFF80
                FFFFFF80
                FFFFFF80
                7FFFFF80
                7FFFFF80
                FFFFFF80
                FFFFFF80
000004A0        FFFFFF80        48.             DC.L NEG,NEG,POS,POS,NEG,NEG,POS,POS
                FFFFFF80
                7FFFFF80
                7FFFFF80
                FFFFFF80
                FFFFFF80
                7FFFFF80
                7FFFFF80
000004C0        7FFFFF80        49.             DC.L POS,POS,POS,POS,NEG,NEG,NEG,NEG
                7FFFFF80
                7FFFFF80
                7FFFFF80
                FFFFFF80
                FFFFFF80
                FFFFFF80
```

```
                FFFFFF80
000004E0        FFFFFF80        50.     DC.L NEG,NEG,NEG,NEG,POS,POS,POS,POS
                FFFFFF80
                FFFFFF80
                FFFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
00000500        7FFFFF80        51.     DC.L POS,POS,POS,POS,POS,POS,POS,POS
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
00000520        FFFFFF80        52.     DC.L NEG,NEG,NEG.NEG,NEG,NEG,NEG,NEG
                FFFFFF80
                FFFFFF80
                FFFFFF80
                FFFFFF80
                FFFFFF80
                FFFFFF80
                FFFFFF80
00000540        7FFFFF80        53.     DC.L POS,NEG,POS,NEG,POS,NEG,POS,NEG
                FFFFFF80
                7FFFFF80
                FFFFFF80
                7FFFFF80
                FFFFFF80
                7FFFFF80
                FFFFFF80
00000560        FFFFFF80        54.     DC.L NEG,POS,NEG,POS,NEG,POS,NEG,POS
                7FFFFF80
                FFFFFF80
                7FFFFF80
                FFFFFF80
                7FFFFF80
                FFFFFF80
                7FFFFF80
00000580        7FFFFF80        55.     DC.L POS,POS,NEG,NEG,POS,POS,NEG,NEG
                7FFFFF80
                FFFFFF80
                FFFFFF80
                7FFFFF80
                7FFFFF80
                FFFFFF80
                FFFFFF80
000005A0        FFFFFF80        56.     DC.L NEG,NEG,POS,POS,NEG,NEG,POS,POS
                FFFFFF80
                7FFFFF80
                7FFFFF80
                FFFFFF80
                FFFFFF80
                7FFFFF80
                7FFFFF80
000005C0        7FFFFF80        57.     DC.L POS,POS,POS,POS,NEG,NEG,NEG,NEG
                7FFFFF80
                7FFFFF80
                7FFFFF80
                FFFFFF80
                FFFFFF80
                FFFFFF80
                FFFFFF80
000005E0        7FFFFF80        58.     DC.L POS,POS,POS,POS,POS,POS,POS,POS
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
                7FFFFF80
00000600        FFFFFF80        59.     DC.L NEG,NEG,NEG,NEG,NEG,NEG,NEG,NEG
                FFFFFF80
                FFFFFF80
                FFFFFF80
                FFFFFF80
                FFFFFF80
                FFFFFF80
                FFFFFF80
```

```
00000620  7FFFFF80          60.         DC.L POS,NEG,POS,NEG,POS,NEG,POS,NEG
          FFFFFF80
          7FFFFF80
          FFFFFF80
          7FFFFF80
          FFFFFF80
          7FFFFF80
          FFFFFF80
00000640                    61.         END
    0 Errors
```

Appendix L: FFT Performance Test Program Code and Data

This appendix presents programs and data for the FFT performance tests. For the sake of brevity, some of the test programs are not shown, in particular, those which measure intermediate phases of execution (sum-and-twiddle time, 64-pt. FFT time, etc.). The programs are similar to those presented herein, with differences only in placement of the timer enable/disable instructions.

The multicomputer master programs (PFFT1-PFFT4) utilize a common set of slave programs; only the master program differs. The master programs are linked to the slave program segment, SLVFFT.

L.1 Program OPT256 Uniprocessor FFT test program

```
                                1.          TTL OPT256
                                2.  * 256 POINT FFT TEST ROUTINE
                                3.  * version with improvements
                                4.  * on bit reversal, and implements Rodriguez's
                                5.  * optimization on element swapping, however
                                6.  * small it is
                                7.  *
                                8.  * JULY 18,1991 ADD CODE FOR PI/T TIMING
                                9.  * AND NEW SIN AND COS TABLES
                               10.  *
                               11.  * CONTROL PROGRAM FOR THE FFT TESTS
                               12.  * ASSUMES COMPLEX DATA AT $1000
                               13.  * COPIES DATA TO $2000 & FFT IS
                               14.  * PERFORMED ON COPIED DATA. RESULTS
                               15.  * RETURNED AT $2000
                               16.  *
                               17.  *
     # 0000FA00                 18.  USTACK EQU $FA00
     # 00000100                 19.  NUMPT EQU 256
     # 00000008                 20.  NUGAM EQU 8
     # 00012FD0                 21.  TCR      EQU $12FD0
     # 00000001                 22.  ENBLETM EQU 1
     # 00FFFFFF                 23.  CNTSTRT EQU $00FFFFFF
     # 0000000A                 24.  SENCLFC EQU 10
     # 00000018                 25.  DISBUF8 EQU $18
                               26.  *
                               27.  *
00001000                       28.          ORG.L $1000
00001000  <400>                29.  DATAIN  DS.B $400
00001400  <400>                30.  DATAOUT DS.B $400
                               31.  *
00001800  43FA F7FE            32.          LEA DATAIN(PC),A1    ;INPUT POINTER
00001804  45FA FBFA            33.          LEA DATAOUT(PC),A2   ;OUTPUT POINTER
00001808  203C 000000FF        34.          MOVE.L #NUMPT-1,D0 ;LOOP COUNTER
0000180E  24D9                 35.  MOVELP  MOVE.L (A1)+,(A2)+
00001810  51C8 FFFC            36.          DBRA D0,MOVELP
00001814  41FA FBEA            37.          LEA DATAOUT(PC),A0   ;POINT FOR FFT
00001818  4239 00012FD0        38.          CLR.B TCR
0000181E  23FC 00FFFFFF        39.          MOVE.L #CNTSTRT,TCR+2 ;LOAD INITIAL
          00012FD2
00001828  13FC 0001            40.          MOVE.B #ENBLETM,TCR   ;START TIME
          00012FD0
00001830  61 1E                41.          BSR.S FFT256          ;DO FFT
00001832  4239 00012FD0        42.          CLR.B TCR         ;END TIME
00001838  203C 00FFFFFF        43.          MOVE.L #CNTSTRT,D0
0000183E  2239 00012FD6        44.          MOVE.L TCR+6,D1
00001844  9081                 45.          SUB.L D1,D0
00001846  4E4F                 46.          TRAP #15
00001848  000A                 47.          DC.W SENCLFC
0000184A  4E4F                 48.          TRAP #15
0000184C  0018                 49.          DC.W DISBUF8
0000184E  4E75                 50.          RTS
                               51.  *
                               52.  *
                               53.  * THIS PROGRAM CALCULATES A 256 POINT
                               54.  * FFT FOR COMPLEX INPUT DATA.  ALL
```

```
                                    55.  * 256 POINTS OF THE SPECTRUM ARE CALCULATED
                                    56.  * INPUT PARAMETERS:
                                    57.  * A0.L POINTS TO BASE ADDRESS OF DATA BLOCK
                                    58.  *      COMPOSED OF WORD DATA, REAL(X) AT
                                    59.  *      A0.L, IMAG(X) AT 2+A0.L
                                    60.  * THE RESULTING COMPLEX SPECTRUM IS RETURNED
                                    61.  * AT THE SAME LOCATIONS, WITH SEPERATION OF
                                    62.  * REAL AND IMAGINARY PARTS AS ABOVE
                                    63.  *
                                    64.  * ORIGINAL ALGORITHM AS SHOWN P.161 OF
                                    65.  * "THE FAST FOURIER TRANSFORM", E.O. BRIGHAM
                                    66.  *  1974, PRENTICE-HALL INC.
                                    67.  *
                                    68.  * NOTE SIN AND COS VALUES ARE * 16384
                                    69.  * AND PROVISION EXISTS FOR CORRECTION FACTORS.
                                    70.  * AND WORD TRUNCATION
                                    71.  *
                                    72.  * RESULTS OF EACH CALCULATION ARRAY ARE /2 TO
                                    73.  * PREVENT WORD OVERFLOWS. PROVIDES A TRUE
                                    74.  * SPECTRAL VALUE ANYWAY!
                                    75.  * DATA MUST BE IN RANGE >=-16384, <=16383
                                    76.  * RODRIGUEZ'S SHUFFLE
                                    77.  *
00001850  283C 000000EF             78.  FFT256  MOVE.L #NUMPT-17,D4        ;COUNTER=K
00001856  3C04                      79.  SWAPLOP MOVE.W D4,D6          ;PREPARE BIT REV
                                    80.  *
                                    81.  * A FAST 8 BIT REVERSAL PROCEDURE
                                    82.  * 8 BIT NUMBER TO BE REVERSED IS IN D6.W
                                    83.  * RESULT RETURNED IN D6.W
                                    84.  * OTHER REG'S TRANSPARENT
                                    85.  *
00001858  3E06                      86.  BITREV2 MOVE.W D6,D7        ;SAVE DATA
0000185A  4246                      87.          CLR.W D6            ;
0000185C  7A07                      88.          MOVEQ.L #7,D5       ;BIT COUNTER
0000185E  E257                      89.  REVLOP2 ROXR.W #1,D7        ;SFT RIGHT,XTEND
00001860  E356                      90.          ROXL.W #1,D6
00001862  51CD FFFA                 91           DBRA D5,REVLOP2
                                    92.  *
                                    93.  * DONE REVERSAL CONTINUE WITH UNSHUFFLE
                                    94.  *
00001866  B846                      95.          CMP.W D6,D4            ;I=REV(K)<=K?
00001868  6C 14                     96.          BGE.S DECK             ;NO SWAP IF SO
0000186A  3204                      97.          MOVE.W D4,D1           ;COPY OF K
0000186C  E541                      98.          ASL.W #2,D1            ;K*4
0000186E  E546                      99.          ASL.W #2,D6            ;I*4
00001870  2430 1000                100.          MOVE.L 0(A0,D1.W),D2 ;DO SWAP
00001874  21B0 6000 1000           101.          MOVE.L 0(A0,D6.W),0(A0,D1.W)
0000187A  2182 6000                102.          MOVE.L D2,0(A0,D6.W) ;DONE SWAP
0000187E  51CC FFD6                103.  DECK    DBRA D4,SWAPLOP
00001882  7201                     104.  FFT     MOVEQ.L #1,D1    ;D1=N2=1
00001884  43FA 009A                105.          LEA SINBLK(PC),A1  ;SIN VALS PNTR A1
00001888  45FA 0296                106.          LEA COSBLK(PC),A2  ;COS VALS PNTR A2
                                   107.  *
                                   108.  * L IS DOWN COUNTED #NUGAM-1 TO 0
                                   109.  *
0000188C  3F3C 0001                110.          MOVE.W #1,-(SP)  ;(SP) IS L
00001890  7407                     111.          MOVEQ.L #NUGAM-1,D2        ;D2=NUGAM-1=NU1
00001892  4283                     112.          CLR.L D3             ;D3=K=0
00001894  700E                     113.          MOVEQ.L #14,D0
                                   114.  *
                                   115.  * INITIALIZE I=N2-1, TO 0 IN STEPS -1
                                   116.  *
00001896  3801                     117.  INITI   MOVE.W D1,D4     ;D4=I=N2
00001898  5344                     118.          SUBQ.W #1,D4     ;SET FOR DBRA
                                   119.  *
                                   120.  * DETERMINE P=MOD(K*2**(NUGAM-L),256)
                                   121.  *
0000189A  3C03                     122.  LOOPIN  MOVE.W D3,D6     ;D6=K
0000189C  3A17                     123.          MOVE.W (SP),D5   ;GET L
0000189E  5145                     124.          SUBQ.W #NUGAM,D5 ;L-NUGAM
000018A0  4445                     125.          NEG.W D5         ;NUGAM-L
000018A2  EB66                     126.          ASL.W D5,D6      ;K*2**(NUGAM-L)
000018A4  0246 00FF                127.          ANDI.W #$00FF,D6 ;MOD 256
000018A8  DC46                     128.          ADD.W D6,D6      ;D6*2 FOR WORD BOUND
000018AA  3A71 6000                129.          MOVEA.W 0(A1,D6.W),A5 ;A5 PNTR IM(W**P)
000018AE  2E03                     130.          MOVE.L D3,D7     ;D7=K
000018B0  2A03                     131.          MOVE.L D3,D5     ;D5 TOO
000018B2  E545                     132.          ASL.W #2,D5      ;K*4 FOR LWORD BOUND
000018B4  4DF0 5000                133.          LEA 0(A0,D5.W),A6   ;A6 IS INDEX X(K)
                                   134.  *                          ;A6 PNT TO RE(X(K))
                                   135.  *                          ;A6+2 TO IM(X(K))
```

```
000018B8  DE41            136.          ADD.W D1,D7      ;D7=K+N2
000018BA  E547            137.          ASL.W #2,D7      ;D7=4(K+N2) LWORD BND
000018BC  47F0 7000       138.          LEA 0(A0,D7.W),A3 ;A3 PNT RE(X(K+N2))
                          139.  *                        ;A3+2 PNT IM(X(K+N2)
                          140.  *
                          141.  * CALCULATE W**P*X(K+N2)
                          142.  * NOTE COMPLEX
                          143.  *
000018C0  3A32 6000       144.          MOVE.W 0(A2,D6.W),D5    ;GET RE(W**P)
000018C4  3E05            145.          MOVE.W D5,D7     ;PUT ALSO IN D7
000018C6  CBD3            146.          MULS (A3),D5     ;RE(W**P)*RE(X(K+N2))
000018C8  3C0D            147.          MOVE.W A5,D6  ;GET IM(W**P)
000018CA  CDEB 0002       148.          MULS 2(A3),D6    ;IM(W**P)*IM(X(K+N2))
000018CE  9A86            149.          SUB.L D6,D5      ;D5=RE(T1)
000018D0  E0A5            150.          ASR.L D0,D5      ;/16384 TO SCALE
000018D2  3C0D            151.          MOVE.W A5,D6  ;GET IM(W**P)
000018D4  CDD3            152.          MULS (A3),D6     ;IM(W**P)*RE(X(K+N2))
000018D6  CFEB 0002       153.          MULS 2(A3),D7    ;RE(W**P)*IM(X(K+N2))
000018DA  DC87            154.          ADD.L D7,D6      ;D6=IM(T1)
000018DC  E0A6            155.          ASR.L D0,D6      ;/16384 TO SCALE
                          156.  *
                          157.  * D6 = IM(T1)
                          158.  * D5 = RE(T1)
                          159.  * D7 IS FREE
                          160.  * DO X(K+N2) = X(K) - T1
                          161.  *    X(K)    = X(K) + T1
                          162.  * AND DIVIDE ANSWERS BY 2 FOR SCALING
                          163.  *
000018DE  3E16            164.          MOVE.W (A6),D7   ;RE(X(K))
000018E0  9E45            165.          SUB.W D5,D7      ;RE(X(K))-RE(T1)
000018E2  E247            166.          ASR.W #1,D7      ;/2
000018E4  3687            167.          MOVE.W D7,(A3)   ;STORE ANSWER
000018E6  3E2E 0002       168.          MOVE.W 2(A6),D7  ;IM(X(K))
000018EA  9E46            169.          SUB.W D6,D7      ;IM(X(K))-IM(T1)
000018EC  E247            170.          ASR.W #1,D7      ;/2
000018EE  3747 0002       171.          MOVE.W D7,2(A3)  ;STORE ANSWER
000018F2  DB56            172.          ADD.W D5,(A6)    ;RE(X(K))+RE(T1)
000018F4  E0D6            173.          ASR.W (A6)       ;/2
000018F6  DD6E 0002       174.          ADD.W D6,2(A6)   ;IM(X(K))+IM(T1)
000018FA  E0EE 0002       175.          ASR.W 2(A6)      ;/2
                          176.  *
                          177.  * MATH IS DONE, NOW THE LOOPS
                          178.  *
000018FE  5243            179.          ADDQ.W #1,D3     ;INCREMENT K
00001900  51CC FF98       180.          DBRA D4,LOOPIN   ;DO TILL I=-1
00001904  D641            181.          ADD.W D1,D3      ;K=K+N2
00001906  0C43 00FF       182.          CMPI.W #NUMPT-1,D3 ;K<N-1 ?
0000190A  6D 8A           183.          BLT INITI      ;DO AGAIN IF SO
                          184.  *
                          185.  *
0000190C  4243            186.          CLR.W D3         ;K=0
0000190E  5342            187.          SUBQ.W #1,D2     ;NU1=NU1-1
00001910  E341            188.          ASL.W #1,D1      ;N2=N2*2
00001912  5257            189.          ADDQ.W #1,(SP)
00001914  0C57 0009       190.          CMPI.W #NUGAM+1,(SP)
00001918  6600 FF7C       191.          BNE INITI
0000191C  301F            192.          MOVE.W (SP)+,D0
0000191E  4E75            193.          RTS
                          194.  *
                          195.  * DATA SECTION
                          196.  *
                          197.  * SIN (-2*PI*I/256)*16384 FOR I=0 TO 255
                          198.  *
00001920  0000 FE6E FCDC  199.  SINBLK  DC.W       0,  -402,  -804, -1205
          FB4B
00001928  F9BA F82A F69C  200.          DC.W   -1606, -2006, -2404, -2801
          F50F
00001930  F384 F1FA F073  201.          DC.W   -3196, -3590, -3981, -4370
          EEEE
00001938  ED6C EBED EA70  202.          DC.W   -4756, -5139, -5520, -5897
          E8F7
00001940  E782 E611 E4A3  203.          DC.W   -6270, -6639, -7005, -7366
          E33A
00001948  E1D5 E074 DF19  204.          DC.W   -7723, -8076, -8423, -8765
          DDC3
00001950  DC72 DB26 D9E0  205.          DC.W   -9102, -9434, -9760,-10080
          D8A0
00001958  D766 D632 D505  206.          DC.W  -10394,-10702,-11003,-11297
          D3DF
00001960  D2BF D1A6 D094  207.          DC.W  -11585,-11866,-12140,-12406
          CF8A
```

```
00001968  CE87 CD8C CC98  208.          DC.W  -12665,-12916,-13160,-13395
          CBAD
00001970  CAC9 C9EE C91B  209.          DC.W  -13623,-13842,-14053,-14256
          C850
00001978  C78F C6D5 C625  210.          DC.W  -14449,-14635,-14811,-14978
          C57E
00001980  C4DF C44A C3BE  211.          DC.W  -15137,-15286,-15426,-15557
          C33B
00001988  C2C1 C251 C1EB  212.          DC.W  -15679,-15791,-15893,-15986
          C18E
00001990  C13B C0F1 C0B1  213.          DC.W  -16069,-16143,-16207,-16261
          C07B
00001998  C04F C02C C014  214.          DC.W  -16305,-16340,-16364,-16379
          C005
000019A0  C000 C005 C014  215.          DC.W  -16384,-16379,-16364,-16340
          C02C
000019A8  C04F C07B C0B1  216.          DC.W  -16305,-16261,-16207,-16143
          C0F1
000019B0  C13B C18E C1EB  217.          DC.W  -16069,-15986,-15893,-15791
          C251
000019B8  C2C1 C33B C3BE  218.          DC.W  -15679,-15557,-15426,-15286
          C44A
000019C0  C4DF C57E C625  219.          DC.W  -15137,-14978,-14811,-14635
          C6D5
000019C8  C78F C850 C91B  220.          DC.W  -14449,-14256,-14053,-13842
          C9EE
000019D0  CAC9 CBAD CC98  221.          DC.W  -13623,-13395,-13160,-12916
          CD8C
000019D8  CE87 CF8A D094  222.          DC.W  -12665,-12406,-12140,-11866
          D1A6
000019E0  D2BF D3DF D505  223.          DC.W  -11585,-11297,-11003,-10702
          D632
000019E8  D766 D8A0 D9E0  224.          DC.W  -10394,-10080, -9760, -9434
          DB26
000019F0  DC72 DDC3 DF19  225.          DC.W   -9102, -8765, -8423, -8076
          E074
000019F8  E1D5 E33A E4A3  226.          DC.W   -7723, -7366, -7005, -6639
          E611
00001A00  E782 E8F7 EA70  227.          DC.W   -6270, -5897, -5520, -5139
          EBED
00001A08  ED6C EEEE F073  228.          DC.W   -4756, -4370, -3981, -3590
          F1FA
00001A10  F384 F50F F69C  229.          DC.W   -3196, -2801, -2404, -2006
          F82A
00001A18  F9BA FB4B FCDC  230.          DC.W   -1606, -1205,  -804,  -402
          FE6E
00001A20  0000 0192 0324  231.          DC.W       0,   402,   804,  1205
          04B5
00001A28  0646 07D6 0964  232.          DC.W    1606,  2006,  2404,  2801
          0AF1
00001A30  0C7C 0E06 0F8D  233.          DC.W    3196,  3590,  3981,  4370
          1112
00001A38  1294 1413 1590  234.          DC.W    4756,  5139,  5520,  5897
          1709
00001A40  187E 19EF 1B5D  235.          DC.W    6270,  6639,  7005,  7366
          1CC6
00001A48  1E2B 1F8C 20E7  236.          DC.W    7723,  8076,  8423,  8765
          223D
00001A50  238E 24DA 2620  237.          DC.W    9102,  9434,  9760, 10080
          2760
00001A58  289A 29CE 2AFB  238.          DC.W   10394, 10702, 11003, 11297
          2C21
00001A60  2D41 2E5A 2F6C  239.          DC.W   11585, 11866, 12140, 12406
          3076
00001A68  3179 3274 3368  240.          DC.W   12665, 12916, 13160, 13395
          3453
00001A70  3537 3612 36E5  241.          DC.W   13623, 13842, 14053, 14256
          37B0
00001A78  3871 392B 39DB  242.          DC.W   14449, 14635, 14811, 14978
          3A82
00001A80  3B21 3BB6 3C42  243.          DC.W   15137, 15286, 15426, 15557
          3CC5
00001A88  3D3F 3DAF 3E15  244.          DC.W   15679, 15791, 15893, 15986
          3E72
00001A90  3EC5 3F0F 3F4F  245.          DC.W   16069, 16143, 16207, 16261
          3F85
00001A98  3FB1 3FD4 3FEC  246.          DC.W   16305, 16340, 16364, 16379
          3FFB
00001AA0  4000 3FFB 3FEC  247.          DC.W   16384, 16379, 16364, 16340
          3FD4
00001AA8  3FB1 3F85 3F4F  248.          DC.W   16305, 16261, 16207, 16143
```

```
                3F0F
00001AB0  3EC5 3E72 3E15  249.          DC.W   16069, 15986, 15893, 15791
                3DAF
00001AB8  3D3F 3CC5 3C42  250.          DC.W   15679, 15557, 15426, 15286
                3BB6
00001AC0  3B21 3A82 39DB  251.          DC.W   15137, 14978, 14811, 14635
                392B
00001AC8  3871 37B0 36E5  252.          DC.W   14449, 14256, 14053, 13842
                3612
00001AD0  3537 3453 3368  253.          DC.W   13623, 13395, 13160, 12916
                3274
00001AD8  3179 3076 2F6C  254.          DC.W   12665, 12406, 12140, 11866
                2E5A
00001AE0  2D41 2C21 2AFB  255.          DC.W   11585, 11297, 11003, 10702
                29CE
00001AE8  289A 2760 2620  256.          DC.W   10394, 10080,  9760,  9434
                24DA
00001AF0  238E 223D 20E7  257.          DC.W    9102,  8765,  8423,  8076
                1F8C
00001AF8  1E2B 1CC6 1B5D  258.          DC.W    7723,  7366,  7005,  6639
                19EF
00001B00  187E 1709 1590  259.          DC.W    6270,  5897,  5520,  5139
                1413
00001B08  1294 1112 0F8D  260.          DC.W    4756,  4370,  3981,  3590
                0E06
00001B10  0C7C 0AF1 0964  261.          DC.W    3196,  2801,  2404,  2006
                07D6
00001B18  0646 04B5 0324  262.          DC.W    1606,  1205,   804,   402
                0192
                          263.  *
                          264.  * COS(-2*PI*I/256)*16384 FOR I=0 TO 255
                          265.  *
00001B20  4000 3FFB 3FEC  266.  COSBLK  DC.W   16384, 16379, 16364, 16340
                3FD4
00001B28  3FB1 3F85 3F4F  267.          DC.W   16305, 16261, 16207, 16143
                3F0F
00001B30  3EC5 3E72 3E15  268.          DC.W   16069, 15986, 15893, 15791
                3DAF
00001B38  3D3F 3CC5 3C42  269.          DC.W   15679, 15557, 15426, 15286
                3BB6
00001B40  3B21 3A82 39DB  270.          DC.W   15137, 14978, 14811, 14635
                392B
00001B48  3871 37B0 36E5  271.          DC.W   14449, 14256, 14053, 13842
                3612
00001B50  3537 3453 3368  272.          DC.W   13623, 13395, 13160, 12916
                3274
00001B58  3179 3076 2F6C  273.          DC.W   12665, 12406, 12140, 11866
                2E5A
00001B60  2D41 2C21 2AFB  274.          DC.W   11585, 11297, 11003, 10702
                29CE
00001B68  289A 2760 2620  275.          DC.W   10394, 10080,  9760,  9434
                24DA
00001B70  238E 223D 20E7  276.          DC.W    9102,  8765,  8423,  8076
                1F8C
00001B78  1E2B 1CC6 1B5D  277.          DC.W    7723,  7366,  7005,  6639
                19EF
00001B80  187E 1709 1590  278.          DC.W    6270,  5897,  5520,  5139
                1413
00001B88  1294 1112 0F8D  279.          DC.W    4756,  4370,  3981,  3590
                0E06
00001B90  0C7C 0AF1 0964  280.          DC.W    3196,  2801,  2404,  2006
                07D6
00001B98  0646 04B5 0324  281.          DC.W    1606,  1205,   804,   402
                0192
00001BA0  0000 FE6E FCDC  282.          DC.W       0,  -402,  -804, -1205
                FB4B
00001BA8  F9BA F82A F69C  283.          DC.W   -1606, -2006, -2404, -2801
                F50F
00001BB0  F384 F1FA F073  284.          DC.W   -3196, -3590, -3981, -4370
                EEEE
00001BB8  ED6C EBED EA70  285.          DC.W   -4756, -5139, -5520, -5897
                E8F7
00001BC0  E782 E611 E4A3  286.          DC.W   -6270, -6639, -7005, -7366
                E33A
00001BC8  E1D5 E074 DF19  287.          DC.W   -7723, -8076, -8423, -8765
                DDC3
00001BD0  DC72 DB26 D9E0  288.          DC.W   -9102, -9434, -9760,-10080
                D8A0
00001BD8  D766 D632 D505  289.          DC.W  -10394,-10702,-11003,-11297
                D3DF
00001BE0  D2BF D1A6 D094  290.          DC.W  -11585,-11866,-12140,-12406
```

```
                CF8A
00001BE8  CE87 CD8C CC98  291.          DC.W  -12665,-12916,-13160,-13395
                CBAD
00001BF0  CAC9 C9EE C91B  292.          DC.W  -13623,-13842,-14053,-14256
                C850
00001BF8  C78F C6D5 C625  293.          DC.W  -14449,-14635,-14811,-14978
                C57E
00001C00  C4DF C44A C3BE  294.          DC.W  -15137,-15286,-15426,-15557
                C33B
00001C08  C2C1 C251 C1EB  295.          DC.W  -15679,-15791,-15893,-15986
                C18E
00001C10  C13B C0F1 C0B1  296.          DC.W  -16069,-16143,-16207,-16261
                C07B
00001C18  C04F C02C C014  297.          DC.W  -16305,-16340,-16364,-16379
                C005
00001C20  C000 C005 C014  298.          DC.W  -16384,-16379,-16364,-16340
                C02C
00001C28  C04F C07B C0B1  299.          DC.W  -16305,-16261,-16207,-16143
                C0F1
00001C30  C13B C18E C1EB  300.          DC.W  -16069,-15986,-15893,-15791
                C251
00001C38  C2C1 C33B C3BE  301.          DC.W  -15679,-15557,-15426,-15286
                C44A
00001C40  C4DF C57E C625  302.          DC.W  -15137,-14978,-14811,-14635
                C6D5
00001C48  C78F C850 C91B  303.          DC.W  -14449,-14256,-14053,-13842
                C9EE
00001C50  CAC9 CBAD CC98  304.          DC.W  -13623,-13395,-13160,-12916
                CD8C
00001C58  CE87 CF8A D094  305.          DC.W  -12665,-12406,-12140,-11866
                D1A6
00001C60  D2BF D3DF D505  306.          DC.W  -11585,-11297,-11003,-10702
                D632
00001C68  D766 D8A0 D9E0  307.          DC.W  -10394,-10080, -9760, -9434
                DB26
00001C70  DC72 DDC3 DF19  308.          DC.W   -9102, -8765, -8423, -8076
                E074
00001C78  E1D5 E33A E4A3  309.          DC.W   -7723, -7366, -7005, -6639
                E611
00001C80  E782 E8F7 EA70  310.          DC.W   -6270, -5897, -5520, -5139
                EBED
00001C88  ED6C EEEE F073  311.          DC.W   -4756, -4370, -3981, -3590
                F1FA
00001C90  F384 F50F F69C  312.          DC.W   -3196, -2801, -2404, -2006
                F82A
00001C98  F9BA FB4B FCDC  313.          DC.W   -1606, -1205,  -804,  -402
                FE6E
00001CA0  0000 0192 0324  314.          DC.W       0,   402,   804,  1205
                04B5
00001CA8  0646 07D6 0964  315.          DC.W    1606,  2006,  2404,  2801
                0AF1
00001CB0  0C7C 0E06 0F8D  316.          DC.W    3196,  3590,  3981,  4370
                1112
00001CB8  1294 1413 1590  317.          DC.W    4756,  5139,  5520,  5897
                1709
00001CC0  187E 19EF 1B5D  318.          DC.W    6270,  6639,  7005,  7366
                1CC6
00001CC8  1E2B 1F8C 20E7  319.          DC.W    7723,  8076,  8423,  8765
                223D
00001CD0  238E 24DA 2620  320.          DC.W    9102,  9434,  9760, 10080
                2760
00001CD8  289A 29CE 2AFB  321.          DC.W   10394, 10702, 11003, 11297
                2C21
00001CE0  2D41 2E5A 2F6C  322.          DC.W   11585, 11866, 12140, 12406
                3076
00001CE8  3179 3274 3368  323.          DC.W   12665, 12916, 13160, 13395
                3453
00001CF0  3537 3612 36E5  324.          DC.W   13623, 13842, 14053, 14256
                37B0
00001CF8  3871 392B 39DB  325.          DC.W   14449, 14635, 14811, 14978
                3A82
00001D00  3B21 3BB6 3C42  326.          DC.W   15137, 15286, 15426, 15557
                3CC5
00001D08  3D3F 3DAF 3E15  327.          DC.W   15679, 15791, 15893, 15986
                3E72
00001D10  3EC5 3F0F 3F4F  328.          DC.W   16069, 16143, 16207, 16261
                3F85
00001D18  3FB1 3FD4 3FEC  329.          DC.W   16305, 16340, 16364, 16379
                3FFB
00001D20                  330.          END
    0 Errors
```

L.2 Program PFFT1 Multicomputer FFT test program #1.

```
                          1.          TTL PFFT1
                          2.  *
                          3.  * THIS PROGRAM PERFORMS A 256-COMPLEX POINT
                          4.  * FFT USING A PARALLEL MODE IN THE RMPS
                          5.  * input/output data is stored as real/imaginary
                          6.  * pair, each component 16-bits long.  Input data
                          7.  * is at DATAII, output data at OUTDATA.
                          8.  *
                          9.  * SLAVE PROGRAMS PUT SLAVE IN READY FOR DATA
                         10.  * MODE AFTER EACH EXECUTION. AFTER 256 PASSES
                         11.  * MASTER SENDS ILLEGAL DATA START ADDRESS TO
                         12.  * PUT SLAVES BACK IN PROGRAM ACCEPT MODE
                         13.  * NEW PROGRAM IS SENT TO UPLOAD INDIVIDUAL
                         14.  * TIMING RESULTS FROM SLAVES.  THAT PROGRAM
                         15.  * ENDS WITH
                         16.  *       TRAP #ABORT
                         17.  * TO PUT SLAVES INTO PROGRAM ACCEPT MODE ONCE
                         18.  * AGAIN
                         19.  ************************************************
                         20.  * THIS VERSION POLLS FOR RESULTS WITH
                         21.  * INDIVIDUAL TRAPS FOR EACH UPLOAD, IN THE
                         22.  * ORDER 0,1,2,3
                         23.  ************************************************
                         24.  *
                         25.  * EQUATES,XREFS AND XDEFS
                         26.  *
                         27.          XDEF DATAII,PGCR,ENABLEB,ABORT,NPT
                         28.          XDEF NPTE,NPE,NPTE4,SHIFT14,TCR
                         29.          XDEF CNTSTRT,ENABLTM
                         30.          XREF S0PRO,S1PRO,S2PRO,S3PRO,DONE
                         31.
# 00012FC0               32.  PGCR EQU $12FC0
# 00000010               33.  ENABLEA EQU $10
# 00000020               34.  ENABLEB EQU $20
# 00000003               35.  SRQASRT EQU 3
# 00000004               36.  ADBYTIN EQU 4
# 00000007               37.  ADBYTOT EQU 7
# 00000005               38.  INDATA  EQU 5
# 00000006               39.  OUTDATA EQU 6
# 00000008               40.  NETCNF  EQU 8
# 00000009               41.  SRQSRVC EQU 9
# 0000000E               42.  ABORT   EQU 14
# 00000100               43.  NPT     EQU $100
# 00000040               44.  NPTE    EQU $40
# 00000100               45.  NPTE4   EQU NPTE*4
# 00000004               46.  NPE     EQU 4
# 0000000E               47.  SHIFT14 EQU 14
# 00000001               48.  SLV0    EQU $01
# 00000002               49.  SLV1    EQU $02
# 00000004               50.  SLV2    EQU $04
# 00000008               51.  SLV3    EQU $08
# 0000000F               52.  ALLSLV  EQU $0F
# 00000006               53.  NUGAM   EQU 6
# 00012FB0               54.  PSR01   EQU $12FB0
# 00012FB1               55.  PSR23   EQU $12FB1
# 00012FB2               56.  PSR45   EQU $12FB2
# 00012FB3               57.  PSR67   EQU $12FB3
# 00012FB4               58.  PSR89   EQU $12FB4
# 00012FB5               59.  PSR1011 EQU $12FB5
# 00012FB6               60.  PSR1213 EQU $12FB6
# 00012FB7               61.  PSR1415 EQU $12FB7
# 00012FB8               62.  PSR1617 EQU $12FB8
# 00012FB9               63.  PSR1819 EQU $12FB9
# 00012FD0               64.  TCR     EQU $12FD0
# 00012FD2               65.  TPLR    EQU TCR+2
# 00012FD6               66.  TCNTR   EQU TCR+6
# 0000000A               67.  SENCLFC EQU 10
# 00000018               68.  DISBUF8 EQU 24
# 0000001A               69.  DISBUF2 EQU 26
# 00FFFFFF               70.  CNTSTRT EQU $00FFFFFF
# 00000001               71.  ENABLTM EQU 1
# 00000011               72.  OUTMESC EQU 17
# 00000008               73.  DIV256  EQU 8
# 00000100               74.  PASSCNT EQU 256
# 00000004               75.  EOT     EQU 4
                         76.  *
                         77.  ******************
                         78.  * MASTER PROGRAM *
```

```
                                    79.  ******************
                                    80.  *
0'000000                            81.          SECTION 0
0'000000  <400>                     82.  DATAIN   DS.B $400
0'000400  <400>                     83.  DATAII   DS.B $400
0'000800  <100>                     84.  XDATA0   DS.B $100
0'000900  <100>                     85.  XDATA1   DS.B $100
0'000A00  <100>                     86.  XDATA2   DS.B $100
0'000B00  <100>                     87.  XDATA3   DS.B $100
0'000C00  BB BB BB CB BB            88.  INITCNF DC.B $BB,$BB,$BB,$CB,$BB,$8C,$CB,$8B,$B8,$8B
          8C CB 8B B8 8B
0'000C0A  BD BB BD CB BD            89.  BRDCAST DC.B $BD,$BB,$BD,$CB,$BD,$8C,$CB,$8B,$B8,$8B
          8C CB 8B B8 8B
                                    90.  *
                                    91.  * CONFIGURE NETWORK TO INITIAL STATE, MASTER
                                    92.  * TO SLAVE 0
                                    93.  * RESPOND TO SLAVE 0'S RFP SRQ, AND LOAD
                                    94.  * ITS PROGRAM
                                    95.  *
0'000C14  287C 00012FD0             96.  STRTFFT MOVEA.L #TCR,A4          ;POINTER TIMER
0'000C1A  4214                      97.          CLR.B (A4)               ;DISABLE F'SURE
0'000C1C  2F3C 00FFFFFF             98.          MOVE.L #CNTSTRT,-(SP)    ;MIN TIME
0'000C22  2F3C 00000000             99.          MOVE.L #$00,-(SP)        ;MAX TIME
0'000C28  2F3C 00000000            100.          MOVE.L #$00,-(SP)        ;RUN'G SUM TIME
0'000C2E  2E3C 000000FF            101.          MOVE.L #PASSCNT-1,D7     ;COUNTER
0'000C34  41FA FFCA                102.          LEA INITCNF(PC),A0
0'000C38  4E48                     103.          TRAP #NETCNF
0'000C3A  103C 0001                104.          MOVE.B #SLV0,D0
0'000C3E  4E49                     105.          TRAP #SRQSRVC
0'000C40  41FA ****                106.          LEA S0PRO(PC),A0         ;STRT ADDR
0'000C44  43FA ****                107.          LEA S1PRO(PC),A1         ;END ADDR+1
0'000C48  6100 027A                108.          BSR SENDOUT              ;ADDR,COUNT,CODE
                                   109.  *
                                   110.  * CONFIGURE NETWORK TO MASTER TO SLAVE 1
                                   111.  * RESPOND TO SLAVE 1'S RFP SRQ, AND LOAD ITS
                                   112.  * PROGRAM
                                   113.  *
0'000C4C  13FC 00B4                114.          MOVE.B #$B4,PSR01
          00012FB0
0'000C54  103C 0002                115.          MOVE.B #SLV1,D0
0'000C58  4E49                     116.          TRAP #SRQSRVC
0'000C5A  41FA ****                117.          LEA S1PRO(PC),A0
0'000C5E  43FA ****                118.          LEA S2PRO(PC),A1
0'000C62  6100 0260                119.          BSR SENDOUT
                                   120.  *
                                   121.  * CONFIGURE NETWORK TO MASTER TO SLAVE 2
                                   122.  * RESPOND TO SLAVE 2'S RFP SRQ, AND LOAD ITS
                                   123.  * PROGRAM
                                   124.  *
0'000C66  13FC 00B4                125.          MOVE.B #$B4,PSR45
          00012FB2
0'000C6E  103C 0004                126.          MOVE.B #SLV2,D0
0'000C72  4E49                     127.          TRAP #SRQSRVC
0'000C74  41FA ****                128.          LEA S2PRO(PC),A0
0'000C78  43FA ****                129.          LEA S3PRO(PC),A1
0'000C7C  6100 0246                130.          BSR SENDOUT
                                   131.  *
                                   132.  * CONFIGURE NETWORK TO MASTER TO SLAVE 3
                                   133.  * RESPOND TO SLAVE 3'S RFP SRQ, AND LOAD ITS
                                   134.  * PROGRAM
                                   135.  *
0'000C80  13FC 00B4                136.          MOVE.B #$B4,PSR89
          00012FB4
0'000C88  103C 0008                137.          MOVE.B #SLV3,D0
0'000C8C  4E49                     138.          TRAP #SRQSRVC
0'000C8E  41FA ****                139.          LEA S3PRO(PC),A0
0'000C92  43FA ****                140.          LEA DONE(PC),A1
0'000C96  6100 022C                141.          BSR SENDOUT
                                   142.  *
                                   143.  * ALL PROGRAMS LOADED, EACH ASSERTING READY FOR
                                   144.  * DATA SRQ.  RESPOND TO ALL, THEN LOAD THE DATA
                                   145.  * IN BROADCAST MODE. FIRST COPY INPUT DATA TO
                                   146.  * THE XDATA0-3 ARRAY, THEN AT END DO UNSHUFFLE
                                   147.  * BACK INTO DATAII ARRAY
                                   148.  *
                                   149.  * LOAD TIMER PRE-LOAD REGISTER HERE
                                   150.  *
0'000C9A  41FA F364                151.  REEXE   LEA DATAIN(PC),A0         ;SOURCE
0'000C9E  43FA F760                152.          LEA DATAII(PC),A1        ;DEST
0'000CA2  303C 00FF                153.          MOVE.W #NPT-1,D0         ;LWORD COUNT
0'000CA6  22D8                     154.  MOVLP   MOVE.L (A0)+,(A1)+       ;LONG WORDS
```

```
0'000CA8  51C8 FFFC        155.          DBRA D0,MOVLP
0'000CAC  297C 00FFFFFF    156.          MOVE.L #CNTSTRT,2(A4)
          0002
0'000CB4  41FA FF54        157.          LEA BRDCAST(PC),A0
0'000CB8  4E48             158.          TRAP #NETCNF
0'000CBA  103C 000F        159.          MOVE.B #ALLSLV,D0
0'000CBE  4E49             160.          TRAP #SRQSRVC
0'000CC0  41FA F73E        161.          LEA DATAII(PC),A0        ;START OF DATA
0'000CC4  43FA FB3A        162.          LEA XDATA0(PC),A1        ;END OF DATA+1
                           163.  *
                           164.  * SET UP SOME REGISTER VALUES
                           165.  *
0'000CC8  3C3C 0100        166.          MOVE.W #NPTE*4,D6
0'000CCC  267C 000003FC    167.          MOVEA.L #$3FC,A3
0'000CD2  247C 00000010    168.          MOVEA.L #$10,A2
0'000CD8  2C7C 00012FC0    169.          MOVEA.L #PGCR,A6
0'000CDE  2A7C 00012FB0    170.          MOVEA.L #PSR01,A5
                           171.  *
                           172.  * START TIMER COUNTING HERE
                           173.  *
0'000CE4  18BC 0001        174.          MOVE.B #ENABLTM,(A4)
0'000CE8  6100 01DA        175.          BSR SENDOUT
                           176.  *
                           177.  * WHEN HERE, ALL SLAVES LOADED, WILL ASSERT
                           178.  * READY TO EXECUTE SRQ.  RESPOND TO LET THEM
                           179.  * CONTINUE.  NETWORK IS CONFIGURED TO ACCEPT
                           180.  * DATA FROM SLAVE 0.
                           181.  *
0'000CEC  700F             182.          MOVEQ.L #ALLSLV,D0
0'000CEE  4E49             183.          TRAP #SRQSRVC            ;SLAVES EXECUTE
                           184.  *
                           185.  * WHEN DONE, SLAVES ASSERT SRQ, THEN UPLOAD DATA
                           186.  * WHEN ACKNOWLEDGED.  ACKNOWLEDGE AND UPLOAD
                           187.  * INDIVIDUALLY
                           188.  *
0'000CF0  7001             189.          MOVEQ.L #SLV0,D0
0'000CF2  4E49             190.          TRAP #SRQSRVC
0'000CF4  3006             191.          MOVE.W D6,D0        ;BYTE COUNT
0'000CF6  41FA FB08        192.          LEA XDATA0(PC),A0        ;DEST ADDRESS
0'000CFA  1CBC 0010        193.          MOVE.B #ENABLEA,(A6)
0'000CFE  4E45             194.          TRAP #INDATA
0'000D00  4216             195.          CLR.B (A6)
                           196.  *
                           197.  * CONFIGURE FOR SLAVE #1 TO MASTER, ACKNOWLEDGE
                           198.  * AND GET DATA
                           199.  *
0'000D02  1B7C 00BC 0005   200.          MOVE.B #$BC,5(A5)
0'000D08  7002             201.          MOVEQ.L #SLV1,D0
0'000D0A  4E49             202.          TRAP #SRQSRVC
0'000D0C  3006             203.          MOVE.W D6,D0
0'000D0E  41FA FBF0        204.          LEA XDATA1(PC),A0
0'000D12  1CBC 0010        205.          MOVE.B #ENABLEA,(A6)
0'000D16  4E45             206.          TRAP #INDATA
0'000D18  4216             207.          CLR.B (A6)
                           208.  *
                           209.  * CONFIGURE FOR SLAVE #2 TO MASTER, ACKNOWLEDGE
                           210.  * AND GET DATA
                           211.  *
0'000D1A  1B7C 00BB 0007   212.          MOVE.B #$BB,7(A5)
0'000D20  7004             213.          MOVEQ.L #SLV2,D0
0'000D22  4E49             214.          TRAP #SRQSRVC
0'000D24  3006             215.          MOVE.W D6,D0
0'000D26  41FA FCD8        216.          LEA XDATA2(PC),A0
0'000D2A  1CBC 0010        217.          MOVE.B #ENABLEA,(A6)
0'000D2E  4E45             218.          TRAP #INDATA
0'000D30  4216             219.          CLR.B (A6)
                           220.  *
                           221.  * CONFIGURE FOR SLAVE #3 TO MASTER, ACKNOWLEDGE
                           222.  * AND GET DATA
                           223.  *
0'000D32  1B7C 00BB 0009   224.          MOVE.B #$BB,9(A5)
0'000D38  7008             225.          MOVEQ.L #SLV3,D0
0'000D3A  4E49             226.          TRAP #SRQSRVC
0'000D3C  3006             227.          MOVE.W D6,D0
0'000D3E  41FA FDC0        228.          LEA XDATA3(PC),A0
0'000D42  1CBC 0010        229.          MOVE.B #ENABLEA,(A6)
0'000D46  4E45             230.          TRAP #INDATA
0'000D48  4216             231.          CLR.B (A6)
                           232.  *
                           233.  * HAVE ALL DATA IN, NOW DO REARRANGE FROM THE
                           234.  * XDAT[illegible] ARRAY BACK TO DATAII ARRAY FOR CORRECT
```

```
                                235.  * ORDER. BEWARE, HEAVILY OPTIMIZED CODE HERE
                                236.  *
0'000D4A  41FA FAB4             237.  REORDER LEA XDATA0(PC),A0
0'000D4E  43FA F6B0             238.          LEA DATAII(PC),A1
0'000D52  7C03                  239.          MOVEQ.L #NPE-1,D6
0'000D54  7A3F                  240.  RLOOPT  MOVEQ.L #NPTE-1,D5
0'000D56  2298                  241.  RLOOPN  MOVE.L (A0)+,(A1)
0'000D58  D3CA                  242.          ADDA.L A2,A1
0'000D5A  51CD FFFA             243.          DBRA D5,RLOOPN
0'000D5E  93CB                  244.          SUBA.L A3,A1      ;USE REGS TO OPT
0'000D60  51CE FFF2             245.          DBRA D6,RLOOPT
                                246.  *
                                247.  * PROGRAM IS DONE, CAN STOP TIMER, AND
                                248.  * OUTPUT NUMBER OF 32 CLOCK INTERVALS
                                249.  * PASSED. FIRST DISABLE TIMER
                                250.  *
0'000D64  4214                  251.          CLR.B (A4)
0'000D66  222C 0006             252.          MOVE.L 6(A4),D1 ;GET NEW COUNT
0'000D6A  203C 00FFFFFF         253.          MOVE.L #CNTSTRT,D0
0'000D70  9081                  254.          SUB.L D1,D0     ;GET ELAPSED COUNT
                                255.  *
                                256.  * OUTPUT CR, LF, THEN THE COUNT IN HEX
                                257.  *
0'000D72  4E4F                  258.          TRAP #15
0'000D74  000A                  259.          DC.W SENCLFC
0'000D76  4E4F                  260.          TRAP #15
0'000D78  0018                  261.          DC.W DISBUF8
                                262.  *
                                263.  * NOW KEEP TRACK OF STATISTICS
                                264.  *
0'000D7A  D197                  265.          ADD.L D0,(SP)   ;ADD TO SUM
0'000D7C  B0AF 0004             266.          CMP.L 4(SP),D0  ;CHECK MAX
0'000D80  6F 04                 267.          BLE.S NOTMAX    ;D0 NOT MAX
0'000D82  2F40 0004             268.          MOVE.L D0,4(SP) ;NEW MAX
0'000D86  B0AF 0008             269.  NOTMAX  CMP.L 8(SP),D0  ;CHECK MIN
0'000D8A  6C 04                 270.          BGE.S NOTMIN    ;D0 NOT MIN
0'000D8C  2F40 0008             271.          MOVE.L D0,8(SP) ;NEW MIN
0'000D90  51CF FF08             272.  NOTMIN  DBRA D7,REEXE   ;DO D7 TIMES
                                273.
                                274.  *
                                275.  * WHEN HERE, PROGRAM HAS BEEN EXECUTED 256 TIMES
                                276.  * SEND AN ILLEGAL ADDRESS FOR DATA, FORCING
                                277.  * SLAVES TO PROGRAM ACCEPT MODE
                                278.  *
0'000D94  41FA FE74             279.          LEA BRDCAST(PC),A0
0'000D98  4E48                  280.          TRAP #NETCNF
0'000D9A  103C 000F             281.          MOVE.B #ALLSLV,D0
0'000D9E  4E49                  282.          TRAP #SRQSRVC
0'000DA0  207C FFFF0000         283.          MOVEA.L #$FFFF0000,A0   ;ILLEGAL ADDR
0'000DA6  13FC 0020             284.          MOVE.B #ENABLEB,PGCR
          00012FC0
0'000DAE  4E47                  285.          TRAP #ADBYTOT   ;THAT'LL FIX 'EM
0'000DB0  4239 00012FC0         286.          CLR.B PGCR
                                287.  *
                                288.  * OUTPUT STATISTICS
                                289.  *
0'000DB6  4E4F                  290.          TRAP #15
0'000DB8  000A                  291.          DC.W SENCLFC
0'000DBA  4E4F                  292.          TRAP #15
0'000DBC  000A                  293.          DC.W SENCLFC
0'000DBE  41FA 011C             294.          LEA AVEMES(PC),A0
0'000DC2  4E4F                  295.          TRAP #15
0'000DC4  0011                  296.          DC.W OUTMESC
0'000DC6  201F                  297.          MOVE.L (SP)+,D0 ;GET SUM
0'000DC8  E080                  298.          ASR.L #DIV256,D0        ;DO AVERAGE
0'000DCA  4E4F                  299.          TRAP #15
0'000DCC  0018                  300.          DC.W DISBUF8    ;OUTPUT AVERAGE
0'000DCE  4E4F                  301.          TRAP #15
0'000DD0  000A                  302.          DC.W SENCLFC
0'000DD2  41FA 012F             303.          LEA MAXMES(PC),A0
0'000DD6  4E4F                  304.          TRAP #15
0'000DD8  0011                  305.          DC.W OUTMESC
0'000DDA  201F                  306.          MOVE.L (SP)+,D0 ;GET MAX
0'000DDC  4E4F                  307.          TRAP #15
0'000DDE  0018                  308.          DC.W DISBUF8
0'000DE0  4E4F                  309.          TRAP #15
0'000DE2  000A                  310.          DC.W SENCLFC
0'000DE4  41FA 0144             311.          LEA MINMES(PC),A0
0'000DE8  4E4F                  312.          TRAP #15
0'000DEA  0011                  313.          DC.W OUTMESC
0'000DEC  201F                  314.          MOVE.L (SP)+,D0 ;GET MIN
```

```
0'000DEE  4E4F             315.          TRAP #15
0'000DF0  0018             316.          DC.W DISBUF8
0'000DF2  4E4F             317.          TRAP #15
0'000DF4  000A             318.          DC.W SENCLFC
                           319.  *
                           320.  * AFTER 256 RUNS, LOAD THE SECOND SLAVE
                           321.  * PROGRAM TO ACQUIRE THE TIMING DATA
                           322.  * IN THE SLAVES TIMER COUNT REGISTERS
                           323.  *
0'000DF6  41FA FE12        324.          LEA BRDCAST(PC),A0      ;BROADCAST MODE
0'000DFA  4E48             325.          TRAP #NETCNF
                           326.  *
                           327.  * RESPOND TO SLAVES RFP
                           328.  *
0'000DFC  103C 000F        329.          MOVE.B #ALLSLV,D0
0'000E00  4E49             330.          TRAP #SRQSRVC
0'000E02  41FA ****        331.          LEA SLVPRO2(PC),A0
0'000E06  43FA ****        332.          LEA ENDSLV2(PC),A1
0'000E0A  93C8             333.          SUBA.L A0,A1
0'000E0C  3009             334.          MOVE.W A1,D0
0'000E0E  13FC 0020        335.          MOVE.B #ENABLEB,PGCR
          00012FC0
0'000E16  4E47             336.          TRAP #ADBYTOT
0'000E18  4E46             337.          TRAP #OUTDATA
0'000E1A  4239 00012FC0    338.          CLR.B PGCR       ;PROGRAM SENT
                           339.  *
                           340.  * RESPOND TO SLAVES RFD
                           341.  *
0'000E20  103C 000F        342.          MOVE.B #ALLSLV,D0
0'000E24  4E49             343.          TRAP #SRQSRVC
                           344.  *
                           345.  * PROGRAM HAS NO INPUT DATA, SEND DUMMY
                           346.  * ADDRESS AND ZERO BYTE COUNT
                           347.  * AND THEN NO DATA
                           348.  *
0'000E26  4240             349.          CLR.W D0
0'000E28  13FC 0020        350.          MOVE.B #ENABLEB,PGCR
          00012FC0
0'000E30  4E47             351.          TRAP #ADBYTOT
0'000E32  4239 00012FC0    352.          CLR.B PGCR
                           353.  *
                           354.  * RESPOND TO READY TO EXECUTE
                           355.  *
0'000E38  103C 000F        356.          MOVE.B #ALLSLV,D0
0'000E3C  4E49             357.          TRAP #SRQSRVC
                           358.  *
                           359.  * SLAVES EXECUTE AND ASSERT AN SRQ WHEN
                           360.  * READY TO REPORT. RESPOND ONE AT A TIME
                           361.  * DETERMINE TIME ELAPSED AND REPORT TO
                           362.  * CONSOLE
                           363.  *
0'000E3E  103C 0001        364.          MOVE.B #SLV0,D0 ;SLAVE 0
0'000E42  4281             365.          CLR.L D1
0'000E44  61 32            366.          BSR.S GETSLTM
0'000E46  13FC 00BC        367.          MOVE.B #$BC,PSR1011 ;SLAVE 1
          00012FB5
0'000E4E  5281             368.          ADDQ.L #1,D1
0'000E50  103C 0002        369.          MOVE.B #SLV1,D0
0'000E54  61 22            370.          BSR.S GETSLTM
0'000E56  13FC 00BB        371.          MOVE.B #$BB,PSR1415 ;SLAVE 2
          00012FB7
0'000E5E  5281             372.          ADDQ.L #1,D1
0'000E60  103C 0004        373.          MOVE.B #SLV2,D0
0'000E64  61 12            374.          BSR.S GETSLTM
0'000E66  13FC 00BB        375.          MOVE.B #$BB,PSR1819 ;SLAVE3
          00012FB9
0'000E6E  5281             376.          ADDQ.L #1,D1
0'000E70  103C 0008        377.          MOVE.B #SLV3,D0
0'000E74  61 02            378.          BSR.S GETSLTM
                           379.  *
                           380.  * NOW FINALLY DONE!
                           381.  *
0'000E76  4E75             382.          RTS
                           383.  *
                           384.  * SUBROUTINE TO GET TIMING DATA FROM SLAVE
                           385.  * SPECIFIED IN D1.B, WITH MASK IN D0.B
                           386.  *
0'000E78  48E7 C0C0        387.  GETSLTM MOVEM.L D0-D1/A0-A1,-(SP) ;SAVE ENV
0'000E7C  4E49             388.          TRAP #SRQSRVC
0'000E7E  7004             389.          MOVEQ.L #$4,D0  ;BYTE COUNT
0'000E80  41FA 0106        390.          LEA TEMPCNT(PC),A0
```

```
0'000E84  13FC 0010 00012FC0    391.          MOVE.B #ENABLEA,PGCR    ;ENABLE INPUT
0'000E8C  4E45                  392.          TRAP #INDATA
0'000E8E  4239 00012FC0         393.          CLR.B PGCR
0'000E94  2248                  394.          MOVEA.L A0,A1
0'000E96  41FA 00B9             395.          LEA MESSA(PC),A0
0'000E9A  4E4F                  396.          TRAP #15
0'000E9C  0011                  397.          DC.W OUTMESC
0'000E9E  1001                  398.          MOVE.B D1,D0
0'000EA0  4E4F                  399.          TRAP #15
0'000EA2  001A                  400.          DC.W DISBUF2
0'000EA4  41FA 00C5             401.          LEA MESSB(PC),A0
0'000EA8  4E4F                  402.          TRAP #15
0'000EAA  0011                  403.          DC.W OUTMESC
0'000EAC  2211                  404.          MOVE.L (A1),D1
0'000EAE  203C 00FFFFFF         405.          MOVE.L #CNTSTRT,D0 ;CALC ELAPSED
0'000EB4  9081                  406.          SUB.L D1,D0
0'000EB6  4E4F                  407.          TRAP #15
0'000EB8  0018                  408.          DC.W DISBUF8
0'000EBA  4E4F                  409.          TRAP #15
0'000EBC  000A                  410.          DC.W SENCLFC
0'000EBE  4CDF 0303             411.          MOVEM.L (SP)+,D0-D1/A0-A1
0'000EC2  4E75                  412.          RTS
                                413.  *
                                414.  * SUBROUTINE SENDOUT
                                415.  *
0'000EC4  93C8                  416.  SENDOUT SUBA.L A0,A1
0'000EC6  3009                  417.          MOVE.W A1,D0
0'000EC8  13FC 0020 00012FC0    418.          MOVE.B #ENABLEB,PGCR
0'000ED0  4E47                  419.          TRAP #ADBYTOT
0'000ED2  4E46                  420.          TRAP #OUTDATA
0'000ED4  4239 00012FC0         421.          CLR.B PGCR
0'000EDA  4E75                  422.          RTS
0'000EDC  41 76 65 72 61 67 65  423.  AVEMES  DC.B 'Average time ( x 4 for microseconds)= ',EOT
          20 74 69 6D 65 20 28
          20 78 20 34 20 66 6F
          72 20 6D 69 63 72 6F
          73 65 63 6F 6E 64 73
          29 3D 20 04
0'000F03  4D 61 78 69 6D 75 6D  424.  MAXMES  DC.B 'Maximum time ( x 4 for microseconds)= ',EOT
          20 74 69 6D 65 20 28
          20 78 20 34 20 66 6F
          72 20 6D 69 63 72 6F
          73 65 63 6F 6E 64 73
          29 3D 20 04
0'000F2A  4D 69 6E 69 6D 75 6D  425.  MINMES  DC.B 'Minimum time ( x 4 for microseconds)= ',EOT
          20 74 69 6D 65 20 28
          20 78 20 34 20 66 6F
          72 20 6D 69 63 72 6F
          73 65 63 6F 6E 64 73
          29 3D 20 04
0'000F51  45 78 65 63 75 74 69  426.  MESSA   DC.B 'Execution time by slave #',EOT
          6F 6E 20 74 69 6D 65
          20 62 79 20 73 6C 61
          76 65 20 23 04
0'000F6B  20 20 28 20 78 20 34  427.  MESSB   DC.B '  ( x 4 for microseconds) = ',EOT
          20 66 6F 72 20 6D 69
          63 72 6F 73 65 63 6F
          6E 64 73 29 20 3D 20
          04
0'000F88  <4>                   428.  TEMPCNT DS.B $4     ;TEMPORARY FOR SLAVE SELF TIME
                                429.  *
                                430.  *
1'000000                        431.          SECTION 1
                                432.  *
                                433.  ****************************
                                434.  * ALL SLAVES PROG 2
                                435.  ****************************
                                436.  *
                                437.  * PROGRAM TO REPORT CONTENTS OF THE
                                438.  * COUNTER REGISTER
```

```
                                439.  *
1'000000  207C 00012FD6         440.  SLVPRO2 MOVEA.L #TCR+6,A0        ;POINTER
1'000006  7004                  441.          MOVEQ.L #$4,D0           ;BYTE COUNT
1'000008  4E43                  442.          TRAP #SRQASRT            ;ASSERT SRQ
1'00000A  13FC 0020             443.          MOVE.B #ENABLEB,PGCR     ;OUTPUT ENABLED
          00012FC0
1'000012  4E46                  444.          TRAP #OUTDATA            ;SEND BLOCK
1'000014  4239 00012FC0         445.          CLR.B PGCR
1'00001A  4E4E                  446.          TRAP #ABORT      ;GOTO PROG ACCEPT MODE
1'00001C  <1>                   447.  ENDSLV2 DS.B 1
1'00001D                        448.          END
    0 Errors
*
* LINK SPEC FOR PFFT1
*
      LINK PFFT1
      LINK SLVFFT
      ORG $1000
      SECTION 0,15,1
      END
```

L.3 Program PFFT2 Multicomputer FFT test program #2.

```
                                1.          TTL PFFT2
                                2.  *
                                3.  * THIS PROGRAM PERFORMS A 256-COMPLEX POINT
                                4.  * FFT USING A PARALLEL MODE IN THE RMPS
                                5.  * input/output data is stored as real/imaginary
                                6.  * pair, each component 16-bits long.  Input data
                                7.  * is at DATAII, output data at OUTDATA.
                                8.  *
                                9.  * SLAVE PROGRAMS PUT SLAVE IN READY FOR DATA
                               10.  * MODE AFTER EACH EXECUTION. AFTER 256 PASSES
                               11.  * MASTER SENDS ILLEGAL DATA START ADDRESS TO
                               12.  * PUT SLAVES BACK IN PROGRAM ACCEPT MODE
                               13.  * NEW PROGRAM IS SENT TO UPLOAD INDIVIDUAL
                               14.  * TIMING RESULTS FROM SLAVES.  THAT PROGRAM
                               15.  * ENDS WITH
                               16.  *      TRAP #ABORT
                               17.  * TO PUT SLAVES INTO PROGRAM ACCEPT MODE ONCE
                               18.  * AGAIN
                               19.  ***********************************************
                               20.  * THIS VERSION AWAITS ALL SLAVES TO BE COMPLETE
                               21.  * BEFORE UPLOADING ANY RESULTS, THEN LOADS IN
                               22.  * THE ORDER 0,1,2,3
                               23.  ***********************************************
                               24.  *
                               25.  * EQUATES,XREFS AND XDEFS
                               26.  *
                               27.          XDEF DATAII,PGCR,ENABLEB,ABORT,NPT
                               28.          XDEF NPTE,NPE,NPTE4,SHIFT14,TCR
                               29.          XDEF CNTSTRT,ENABLTM
                               30.          XREF S0PRO,S1PRO,S2PRO,S3PRO,DONE
                               31.
# 00012FC0                     32.  PGCR EQU $12FC0
# 00000010                     33.  ENABLEA EQU $10
# 00000020                     34.  ENABLEB EQU $20
# 00000003                     35.  SRQASRT EQU 3
# 00000004                     36.  ADBYTIN EQU 4
# 00000007                     37.  ADBYTOT EQU 7
# 00000005                     38.  INDATA  EQU 5
# 00000006                     39.  OUTDATA EQU 6
# 00000008                     40.  NETCNF  EQU 8
# 00000009                     41.  SRQSRVC EQU 9
# 0000000E                     42.  ABORT   EQU 14
# 00000100                     43.  NPT     EQU $100
# 00000040                     44.  NPTE    EQU $40
# 00000100                     45.  NPTE4   EQU NPTE*4
# 00000004                     46.  NPE     EQU 4
# 0000000E                     47.  SHIFT14 EQU 14
# 00000001                     48.  SLV0    EQU $01
# 00000002                     49.  SLV1    EQU $02
# 00000004                     50.  SLV2    EQU $04
# 00000008                     51.  SLV3    EQU $08
# 0000000F                     52.  ALLSLV  EQU $0F
# 00000006                     53.  NUGAM   EQU 6
# 00012FB0                     54.  PSR01   EQU $12FB0
# 00012FB1                     55.  PSR23   EQU $12FB1
# 00012FB2                     56.  PSR45   EQU $12FB2
# 00012FB3                     57.  PSR67   EQU $12FB3
# 00012FB4                     58.  PSR89   EQU $12FB4
# 00012FB5                     59.  PSR1011 EQU $12FB5
# 00012FB6                     60.  PSR1213 EQU $12FB6
# 00012FB7                     61.  PSR1415 EQU $12FB7
# 00012FB8                     62.  PSR1617 EQU $12FB8
# 00012FB9                     63.  PSR1819 EQU $12FB9
# 00012FD0                     64.  TCR     EQU $12FD0
# 00012FD2                     65.  TPLR    EQU TCR+2
# 00012FD6                     66.  TCNTR   EQU TCR+6
# 0000000A                     67.  SENCLFC EQU 10
# 00000018                     68.  DISBUF8 EQU 24
# 0000001A                     69.  DISBUF2 EQU 26
# 00FFFFFF                     70.  CNTSTRT EQU $00FFFFFF
# 00000001                     71.  ENABLTM EQU 1
# 00000011                     72.  OUTMESC EQU 17
# 00000008                     73.  DIV256  EQU 8
# 00000100                     74.  PASSCNT EQU 256
# 00000004                     75.  EOT     EQU 4
                               76.  *
                               77.  ******************
                               78.  * MASTER PROGRAM *
```

```
                                    79.  ******************
                                    80.  *
0'000000                            81.          SECTION 0
0'000000  <400>                     82.  DATAIN   DS.B $400
0'000400  <400>                     83.  DATAII   DS.B $400
0'000800  <100>                     84.  XDATA0   DS.B $100
0'000900  <100>                     85.  XDATA1   DS.B $100
0'000A00  <100>                     86.  XDATA2   DS.B $100
0'000B00  <100>                     87.  XDATA3   DS.B $100
0'000C00  BB BB BB CB BB            88.  INITCNF DC.B $BB,$BB,$BB,$CB,$BB,$8C,$CB,$8B,$B8,$8B
          8C CB 8B B8 8B
0'000C0A  BD BB BD CB BD            89.  BRDCAST DC.B $BD,$BB,$BD,$CB,$BD,$8C,$CB,$8B,$B8,$8B
          8C CB 8B B8 8B
                                    90.  *
                                    91.  * CONFIGURE NETWORK TO INITIAL STATE, MASTER
                                    92.  * TO SLAVE 0
                                    93.  * RESPOND TO SLAVE 0'S RFP SRQ, AND LOAD
                                    94.  * ITS PROGRAM
                                    95.  *
0'000C14  287C 00012FD0             96.  STRTFFT MOVEA.L #TCR,A4          ;POINTER TIMER
0'000C1A  4214                      97.          CLR.B (A4)               ;DISABLE F'SURE
0'000C1C  2F3C 00FFFFFF             98.          MOVE.L #CNTSTRT,-(SP)    ;MIN TIME
0'000C22  2F3C 00000000             99.          MOVE.L #$00,-(SP)        ;MAX TIME
0'000C28  2F3C 00000000            100.          MOVE.L #$00,-(SP)        ;RUN'G SUM TIME
0'000C2E  2E3C 000000FF            101.          MOVE.L #PASSCNT-1,D7     ;COUNTER
0'000C34  41FA FFCA                102.          LEA INITCNF(PC),A0
0'000C38  4E48                     103.          TRAP #NETCNF
0'000C3A  103C 0001                104.          MOVE.B #SLV0,D0
0'000C3E  4E49                     105.          TRAP #SRQSRVC
0'000C40  41FA ****                106.          LEA S0PRO(PC),A0         ;STRT ADDR
0'000C44  43FA ****                107.          LEA S1PRO(PC),A1         ;END ADDR+1
0'000C48  6100 026E                108.          BSR SENDOUT              ;ADDR,COUNT,CODE
                                   109.  *
                                   110.  * CONFIGURE NETWORK TO MASTER TO SLAVE 1
                                   111.  * RESPOND TO SLAVE 1'S RFP SRQ, AND LOAD ITS
                                   112.  * PROGRAM
                                   113.  *
0'000C4C  13FC 00B4                114.          MOVE.B #$B4,PSR01
          00012FB0
0'000C54  103C 0002                115.          MOVE.B #SLV1,D0
0'000C58  4E49                     116.          TRAP #SRQSRVC
0'000C5A  41FA ****                117.          LEA S1PRO(PC),A0
0'000C5E  43FA ****                118.          LEA S2PRO(PC),A1
0'000C62  6100 0254                119.          BSR SENDOUT
                                   120.  *
                                   121.  * CONFIGURE NETWORK TO MASTER TO SLAVE 2
                                   122.  * RESPOND TO SLAVE 2'S RFP SRQ, AND LOAD ITS
                                   123.  * PROGRAM
                                   124.  *
0'000C66  13FC 00B4                125.          MOVE.B #$B4,PSR45
          00012FB2
0'000C6E  103C 0004                126.          MOVE.B #SLV2,D0
0'000C72  4E49                     127.          TRAP #SRQSRVC
0'000C74  41FA ****                128.          LEA S2PRO(PC),A0
0'000C78  43FA ****                129.          LEA S3PRO(PC),A1
0'000C7C  6100 023A                130.          BSR SENDOUT
                                   131.  *
                                   132.  * CONFIGURE NETWORK TO MASTER TO SLAVE 3
                                   133.  * RESPOND TO SLAVE 3'S RFP SRQ, AND LOAD ITS
                                   134.  * PROGRAM
                                   135.  *
0'000C80  13FC 00B4                136.          MOVE.B #$B4,PSR89
          00012FB4
0'000C88  103C 0008                137.          MOVE.B #SLV3,D0
0'000C8C  4E49                     138.          TRAP #SRQSRVC
0'000C8E  41FA ****                139.          LEA S3PRO(PC),A0
0'000C92  43FA ****                140.          LEA DONE(PC),A1
0'000C96  6100 0220                141.          BSR SENDOUT
                                   142.  *
                                   143.  * ALL PROGRAMS LOADED, EACH ASSERTING READY FOR
                                   144.  * DATA SRQ.  RESPOND TO ALL, THEN LOAD THE DATA
                                   145.  * IN BROADCAST MODE. FIRST COPY INPUT DATA TO
                                   146.  * THE XDATA0-3 ARRAY, THEN AT END DO UNSHUFFLE
                                   147.  * BACK INTO DATAII ARRAY
                                   148.  *
                                   149.  * LOAD TIMER PRE-LOAD REGISTER HERE
                                   150.  *
0'000C9A  41FA F364                151.  REEXE    LEA DATAIN(PC),A0        ;SOURCE
0'000C9E  43FA F760                152.          LEA DATAII(PC),A1        ;DEST
0'000CA2  303C 00FF                153.          MOVE.W #NPT-1,D0         ;LWORD COUNT
0'000CA6  22D8                     154.  MOVLP   MOVE.L (A0)+,(A1)+       ;LONG WORDS
```

```
0'000CA8  51C8 FFFC       155.          DBRA D0,MOVLP
0'000CAC  297C 00FFFFFF   156.          MOVE.L #CNTSTRT,2(A4)
          0002
0'000CB4  41FA FF54       157.          LEA BRDCAST(PC),A0
0'000CB8  4E48            158.          TRAP #NETCNF
0'000CBA  103C 000F       159.          MOVE.B #ALLSLV,D0
0'000CBE  4E49            160.          TRAP #SRQSRVC
0'000CC0  41FA F73E       161.          LEA DATAII(PC),A0        ;START OF DATA
0'000CC4  43FA FB3A       162.          LEA XDATA0(PC),A1        ;END OF DATA+1
                          163.  *
                          164.  * SET UP SOME REGISTER VALUES
                          165.  *
0'000CC8  3C3C 0100       166.          MOVE.W #NPTE*4,D6
0'000CCC  267C 000003FC   167.          MOVEA.L #$3FC,A3
0'000CD2  247C 00000010   168.          MOVEA.L #$10,A2
0'000CD8  2C7C 00012FC0   169.          MOVEA.L #PGCR,A6
0'000CDE  2A7C 00012FB0   170.          MOVEA.L #PSR01,A5
                          171.  *
                          172.  * START TIMER COUNTING HERE
                          173.  *
0'000CE4  18BC 0001       174.          MOVE.B #ENABLTM,(A4)
0'000CE8  6100 01CE       175.          BSR SENDOUT
                          176.  *
                          177.  * WHEN HERE, ALL SLAVES LOADED, WILL ASSERT
                          178.  * READY TO EXECUTE SRQ.  RESPOND TO LET THEM
                          179.  * CONTINUE.  NETWORK IS CONFIGURED TO ACCEPT
                          180.  * DATA FROM SLAVE 0.
                          181.  *
0'000CEC  700F            182.          MOVEQ.L #ALLSLV,D0
0'000CEE  4E49            183.          TRAP #SRQSRVC            ;SLAVES EXECUTE
                          184.  *
                          185.  * WHEN DONE, SLAVES ASSERT SRQ, THEN UPLOAD DATA
                          186.  * WHEN ACKNOWLEDGED.  ACKNOWLEDGE ALL SLAVES, THEN
                          187.  * AND UPLOAD INDIVIDUAL RESULTS
                          188.  *
0'000CF0  700F            189.          MOVEQ.L #ALLSLV,D0
0'000CF2  4E49            190.          TRAP #SRQSRVC
0'000CF4  3006            191.          MOVE.W D6,D0        ;BYTE COUNT
0'000CF6  41FA FB08       192.          LEA XDATA0(PC),A0        ;DEST ADDRESS
0'000CFA  1CBC 0010       193.          MOVE.B #ENABLEA,(A6)
0'000CFE  4E45            194.          TRAP #INDATA
0'000D00  4216            195.          CLR.B (A6)
                          196.  *
                          197.  * CONFIGURE FOR SLAVE #1 TO MASTER,
                          198.  * AND GET DATA
                          199.  *
0'000D02  1B7C 00BC 0005  200.          MOVE.B #$BC,5(A5)
0'000D08  3005            201.          MOVE.W D6,D0
0'000D0A  41FA FBF4       202.          LEA XDATA1(PC),A0
0'000D0E  1CBC 0010       203.          MOVE.B #ENABLEA,(A6)
0'000D12  4E45            204.          TRAP #INDATA
0'000D14  4216            205.          CLR.B (A6)
                          206.  *
                          207.  * CONFIGURE FOR SLAVE #2 TO MASTER,
                          208.  * AND GET DATA
                          209.  *
0'000D16  1B7C 00BB 0007  210.          MOVE.B #$BB,7(A5)
0'000D1C  3006            211.          MOVE.W D6,D0
0'000D1E  41FA FCE0       212.          LEA XDATA2(PC),A0
0'000D22  1CBC 0010       213.          MOVE.B #ENABLEA,(A6)
0'000D26  4E45            214.          TRAP #INDATA
0'000D28  4216            215.          CLR.B (A6)
                          216.  *
                          217.  * CONFIGURE FOR SLAVE #3 TO MASTER,
                          218.  * AND GET DATA
                          219.  *
0'000D2A  1B7C 00BB 0009  220.          MOVE.B #$BB,9(A5)
0'000D30  3006            221.          MOVE.W D6,D0
0'000D32  41FA FDCC       222.          LEA XDATA3(PC),A0
0'000D36  1CBC 0010       223.          MOVE.B #ENABLEA,(A6)
0'000D3A  4E45            224.          TRAP #INDATA
0'000D3C  4216            225.          CLR.B (A6)
                          226.  *
                          227.  * HAVE ALL DATA IN, NOW DO REARRANGE FROM THE
                          228.  * XDATAi ARRAY BACK TO DATAII ARRAY FOR CORRECT
                          229.  * ORDER. BEWARE, HEAVILY OPTIMIZED CODE HERE
                          230.  *
0'000D3E  41FA FAC0       231.  REORDER LEA XDATA0(PC),A0
0'000D42  43FA F6BC       232.          LEA DATAII(PC),A1
0'000D46  7C03            233.          MOVEQ.L #NPE-1,D6
0'000D48  7A3F            234.  RLOOPT  MOVEQ.L #NPTE-1,D5
```

```
0'000D4A  2298              235.  RLOOPN  MOVE.L (A0)+,(A1)
0'000D4C  D3CA              236.          ADDA.L A2,A1
0'000D4E  51CD FFFA         237.          DBRA D5,RLOOPN
0'000D52  93CB              238.          SUBA.L A3,A1       ;USE REGS TO OPT
0'000D54  51CE FFF2         239.          DBRA D6,RLOOPT
                            240.  *
                            241.  * PROGRAM IS DONE, CAN STOP TIMER, AND
                            242.  * OUTPUT NUMBER OF 32 CLOCK INTERVALS
                            243.  * PASSED. FIRST DISABLE TIMER
                            244.  *
0'000D58  4214              245.          CLR.B (A4)
0'000D5A  222C 0006         246.          MOVE.L 6(A4),D1 ;GET NEW COUNT
0'000D5E  203C 00FFFFFF     247.          MOVE.L #CNTSTRT,D0
0'00CD64  9081              248.          SUB.L D1,D0     ;GET ELAPSED COUNT
                            249.  *
                            250.  * OUTPUT CR, LF, THEN THE COUNT IN HEX
                            251.  *
0'000D66  4E4F              252.          TRAP #15
0'000D68  000A              253.          DC.W SENCLFC
0'000D6A  4E4F              254.          TRAP #15
0'000D6C  0018              255.          DC.W DISBUF8
                            256.  *
                            257.  * NOW KEEP TRACK OF STATISTICS
                            258.  *
0'000D6E  D197              259.          ADD.L D0,(SP)   ;ADD TO SUM
0'000D70  B0AF 0004         260.          CMP.L 4(SP),D0  ;CHECK MAX
0'000D74  6F 04             261.          BLE.S NOTMAX    ;D0 NOT MAX
0'000D76  2F40 0004         262.          MOVE.L D0,4(SP) ;NEW MAX
0'000D7A  B0AF 0008         263.  NOTMAX  CMP.L 8(SP),D0  ;CHECK MIN
0'000D7E  6C 04             264.          BGE.S NOTMIN    ;D0 NOT MIN
0'000D80  2F40 0008         265.          MOVE.L D0,8(SP) ;NEW MIN
0'000D84  51CF FF14         266.  NOTMIN  DBRA D7,REEXE   ;DO D7 TIMES
                            267.
                            268   *
                            269.  * WHEN HERE, PROGRAM HAS BEEN EXECUTED 256 TIMES
                            270.  * SEND AN ILLEGAL ADDRESS FOR DATA, FORCING
                            271.  * SLAVES TO PROGRAM ACCEPT MODE
                            272.  *
0'000D88  41FA FE80         273.          LEA BRDCAST(PC),A0
0'000D8C  4E48              274           TRAP #NETCMF
0'000D8E  103C 000F         275.          MOVE.B #ALLSLV,D0
0'000D92  4E49              276.          TRAP #SRQSRVC
0'000D94  207C FFFF0000     277.          MOVEA.L #$FFFF0000,A0   ;ILLEGAL ADDR
0'000D9A  13FC 0020         278.          MOVE.B #ENABLEB,PGCR
          00012FC0
0'000DA2  4E47              279.          TRAP #ADBYTOT   ;THAT'LL FIX 'EM
0'000DA4  4239 00012FC0     280.          CLR.B PGCR
                            281.  *
                            282.  * OUTPUT STATISTICS
                            283.  *
0'000DAA  4E4F              284.          TRAP #15
0'000DAC  000A              285.          DC.W SENCLFC
0'000DAE  4E4F              286.          TRAP #15
0'000DB0  000A              287.          DC.W SENCLFC
'000DB2   41FA 011C         288.          LEA AVEMES(PC),A0
 '000DB6  4E4F              289.          TRAP #15
0'000DB8  0011              290.          DC.W OUTMESC
0'000DBA  201F              291.          MOVE.L (SP)+,D0 ;GET SUM
0'000DBC  E080              292.          ASR.L #DIV256,D0        ;DO AVERAGE
0'000DBE  4E4F              293.          TRAP #15
0'000DC0  0018              294.          DC.W DISBUF8    ;OUTPUT AVERAGE
0'000DC2  4E4F              295.          TRAP #15
0'000DC4  000A              296.          DC.W SENCLFC
0'000DC6  41FA 012F         297.          LEA MAXMES(PC),A0
0'000DCA  4E4F              298.          TRAP #15
0'000DCC  0011              299.          DC.W OUTMESC
0'000DCE  201F              300.          MOVE.L (SP)+,D0 ;GET MAX
0'000DD0  4E4F              301.          TRAP #15
0'000DD2  0018              302           DC.W DISBUF8
0'000DD4  4E4F              303.          TRAP #15
0'000DD6  000A              304.          DC.W SENCLFC
0'000DD8  41FA 0144         305.          LEA MINMES(PC),A0
0'000DDC  4E4F              306.          TRAP #15
0'000DDE  0011              307.          DC.W OUTMESC
0'000DE0  201F              308.          MOVE.L (SP)+,D0 ;GET MIN
0'000DE2  4E4F              309.          TRAP #15
0'000DE4  0018              310.          DC.W DISBUF8
0'000DE6  4E4F              311.          TRAP #15
0'000DE8  000A              312.          DC.W SENCLFC
                            313.  *
                            314.  * AFTER 256 RUNS, LOAD THE SECOND SLAVE
```

```
                                315.  * PROGRAM TO ACQUIRE THE TIMING DATA
                                316.  * IN THE SLAVES TIMER COUNT REGISTERS
                                317.  *
0'000DEA  41FA FE1E             318.          LEA BRDCAST(PC),A0      ;BROADCAST MODE
0'000DEE  4E48                  319.          TRAP #NETCNF
                                320.  *
                                321.  * RESPOND TO SLAVES RFP
                                322.  *
0'000DF0  103C 000F             323.          MOVE.B #ALLSLV,D0
0'000DF4  4E49                  324.          TRAP #SRQSRVC
0'000DF6  41FA ****             325.          LEA SLVPRO2(PC),A0
0'000DFA  43FA ****             326.          LEA ENDSLV2(PC),A1
0'000DFE  93C8                  327.          SUBA.L A0,A1
0'000E00  3009                  328.          MOVE.W A1,D0
0'000E02  13FC 0020             329.          MOVE.B #ENABLEB,PGCR
          00012FC0
0'000E0A  4E47                  330.          TRAP #ADBYTOT
0'000E0C  4E46                  331.          TRAP #OUTDATA
0'000E0E  4239 00012FC0         332.          CLR.B PGCR       ;PROGRAM SENT
                                333.  *
                                334.  * RESPOND TO SLAVES RFD
                                335.  *
0'000E14  103C 000F             336.          MOVE.B #ALLSLV,D0
0'000E18  4E49                  337.          TRAP #SRQSRVC
                                338.  *
                                339.  * PROGRAM HAS NO INPUT DATA, SEND DUMMY
                                340.  * ADDRESS AND ZERO BYTE COUNT
                                341.  * AND THEN NO DATA
                                342.  *
0'000E1A  4240                  343.          CLR.W D0
0'000E1C  13FC 0020             344.          MOVE.B #ENABLEB,PGCR
          00012FC0
0'000E24  4E47                  345.          TRAP #ADBYTOT
0'000E26  4239 00012FC0         346.          CLR.B PGCR
                                347.  *
                                348.  * RESPOND TO READY TO EXECUTE
                                349.  *
0'000E2C  103C 000F             350.          MOVE.B #ALLSLV,D0
0'000E30  4E49                  351.          TRAP #SRQSRVC
                                352.  *
                                353.  * SLAVES EXECUTE AND ASSERT AN SRQ WHEN
                                354.  * READY TO REPORT. RESPOND ONE AT A TIME
                                355.  * DETERMINE TIME ELAPSED AND REPORT TO
                                356.  * CONSOLE
                                357.  *
0'000E32  103C 0001             358.          MOVE.B #SLV0,D0 ;SLAVE 0
0'000E36  4281                  359.          CLR.L D1
0'000E38  61 32                 360.          BSR.S GETSLTM
0'000E3A  13FC 00BC             361.          MOVE.B #$BC,PSR1011 ;SLAVE 1
          00012FB5
0'000E42  5281                  362.          ADDQ.L #1,D1
0'000E44  103C 0002             363.          MOVE.B #SLV1,D0
0'000E48  61 22                 364.          BSR.S GETSLTM
0'000E4A  13FC 00BB             365.          MOVE.B #$BB,PSR1415 ;SLAVE 2
          00012FB7
0'000E52  5281                  366.          ADDQ.L #1,D1
0'000E54  103C 0004             367.          MOVE.B #SLV2,D0
0'000E58  61 12                 368.          BSR.S GETSLTM
0'000E5A  13FC 00BB             369.          MOVE.B #$BB,PSR1819 ;SLAVE3
          00012FB9
0'000E62  5281                  370.          ADDQ.L #1,D1
0'000E64  103C 0008             371.          MOVE.B #SLV3,D0
0'000E68  61 02                 372.          BSR.S GETSLTM
                                373.  *
                                374.  * NOW FINALLY DONE!
                                375.  *
0'000E6A  4E75                  376.          RTS
                                377.  *
                                378.  * SUBROUTINE TO GET TIMING DATA FROM SLAVE
                                379.  * SPECIFIED IN D1.B, WITH MASK IN D0.B
                                380.  *
0'000E6C  48E7 C0C0             381.  GETSLTM MOVEM.L D0-D1/A0-A1,-(SP) ;SAVE ENV
0'000E70  4E49                  382.          TRAP #SRQSRVC
0'000E72  7004                  383.          MOVEQ.L #$4,D0  ;BYTE COUNT
0'000E74  41FA 0106             384.          LEA TEMPCNT(PC),A0
0'000E78  13FC 0010             385.          MOVE.B #ENABLEA,PGCR    ;ENABLE INPUT
          00012FC0
0'000E80  4E45                  386.          TRAP #INDATA
0'000E82  4239 00012FC0         387.          CLR.B PGCR
0'000E88  2248                  388.          MOVEA.L A0,A1
0'000E8A  41FA 00B9             389.          LEA MESSA(PC),A0
```

```
0'000E8E  4E4F              390.          TRAP #15
0'000E90  0011              391.          DC.W OUTMESC
0'000E92  1001              392.          MOVE.B D1,D0
0'000E94  4E4F              393.          TRAP #15
0'000E96  001A              394.          DC.W DISBUF2
0'000E98  41FA 00C5         395.          LEA MESSB(PC),A0
0'000E9C  4E4F              396.          TRAP #15
0'000E9E  0011              397.          DC.W OUTMESC
0'000EA0  2211              398.          MOVE.L (A1),D1
0'000EA2  203C 00FFFFFF     399.          MOVE.L #CNTSTRT,D0 ;CALC ELAPSED
0'000EA8  9081              400.          SUB.L D1,D0
0'000EAA  4E4F              401.          TRAP #15
0'000EAC  0018              402.          DC.W DISBUF8
0'000EAE  4E4F              403.          TRAP #15
0'000EB0  000A              404.          DC.W SENCLFC
0'000EB2  4CDF 0303         405.          MOVEM.L (SP)+,D0-D1/A0-A1
0'000EB6  4E75              406.          RTS
                            407.  *
                            408.  * SUBROUTINE SENDOUT
                            409.  *
0'000EB8  93C8              410.  SENDOUT SUBA.L A0,A1
0'000EBA  3009              411.          MOVE.W A1,D0
0'000EBC  13FC 0020         412.          MOVE.B #ENABLEB,PGCR
          00012FC0
0'000EC4  4E47              413.          TRAP #ADBYTOT
0'000EC6  4E46              414.          TRAP #OUTDATA
0'000EC8  4239 00012FC0     415.          CLR.B PGCR
0'000ECE  4E75              416.          RTS
0'000ED0  41 76 65 72 61    417.  AVEMES  DC.B 'Average time ( x 4 for microseconds)= ',EOT
          67 65 20 74 69
          6D 65 20 28 20
          78 20 34 20 66
          6F 72 20 6D 69
          63 72 6F 73 65
          63 6F 6E 64 73
          29 3D 20 04
0'000EF7  4D 61 78 69 6D    418.  MAXMES  DC.B 'Maximum time ( x 4 for microseconds)= ',EOT
          75 6D 20 74 69
          6D 65 20 28 20
          78 20 34 20 66
          6F 72 20 6D 69
          63 72 6F 73 65
          63 6F 6E 64 73
          29 3D 20 04
0'000F1E  4D 69 6E 69 6D    419.  MINMES  DC.B 'Minimum time ( x 4 for microseconds)= ',EOT
          75 6D 20 74 69
          6D 65 20 28 20
          78 20 34 20 66
          6F 72 20 6D 69
          63 72 6F 73 65
          63 6F 6E 64 73
          29 3D 20 04
0'000F45  45 78 65 63 75    420.  MESSA   DC.B 'Execution time by slave #',EOT
          74 69 6F 6E 20
          74 69 6D 65 20
          62 79 20 73 6C
          61 76 65 20 23
          04
0'000F5F  20 20 28 20 78    421.  MESSB   DC.B '   ( x 4 for microseconds) = ',EOT
          20 34 20 66 6F
          72 20 6D 69 63
          72 6F 73 65 63
          6F 6E 64 73 29
          20 3D 20 04
0'000F7C  <4>               422.  TEMPCNT DS.B $4     ;TEMPORARY FOR SLAVE SELF TIME
                            423.  *
                            424.  *
1'000000                    425.          SECTION 1
                            426.  *
                            427.  ******************************
                            428.  * ALL SLAVES PROG 2
                            429.  ******************************
                            430.  *
                            431.  * PROGRAM TO REPORT CONTENTS OF THE
                            432.  * COUNTER REGISTER
                            433.  *
1'000000  207C 00012FD6     434.  SLVPRO2 MOVEA.L #TCR+6,A0       ;POINTER
1'000006  7004              435.          MOVEQ  #$4,D0           ;BYTE COUNT
1'000008  4E43              436.          TRAP #SRQASRT           ;ASSERT SRQ
1'00000A  13FC 0020         437.          MOVE.B #ENABLEB,PGCR    ;OUTPUT ENABLED
          00012FC0
```

```
1'000012  4E46            438.          TRAP #OUTDATA            ;SEND BLOCK
1'000014  4239 00012FC0   439.          CLR.B PGCR
1'00001A  4E4E            440.          TRAP #ABORT      ;GOTO PROG ACCEPT MODE
1'00001C  <1>             441.  ENDSLV2 DS.B 1
1'00001D                  442.          END
    0 Errors
*
* LINK SPEC FOR PFFT1
*
      LINK PFFT2
      LINK SLVFFT
      ORG $1000
      SECTION 0,15,1
      END
```

L.4 Program PFFT3 Multicomputer FFT test program #3.

```
                                1.          TTL PFFT3
                                2.  *
                                3.  * THIS PROGRAM PERFORMS A 256-COMPLEX POINT
                                4.  * FFT USING A PARALLEL MODE IN THE RMPS
                                5.  * input/output data is stored as real/imaginary
                                6.  * pair, each component 16-bits long.  Input data
                                7.  * is at DATAII, output data at OUTDATA.
                                8.  *
                                9.  * SLAVE PROGRAMS PUT SLAVE IN READY FOR DATA
                               10.  * MODE AFTER EACH EXECUTION. AFTER 256 PASSES
                               11.  * MASTER SENDS ILLEGAL DATA START ADDRESS TO
                               12.  * PUT SLAVES BACK IN PROGRAM ACCEPT MODE
                               13.  * NEW PROGRAM IS SENT TO UPLOAD INDIVIDUAL
                               14.  * TIMING RESULTS FROM SLAVES.  THAT PROGRAM
                               15.  * ENDS WITH
                               16.  *      TRAP #ABORT
                               17.  * TO PUT SLAVES INTO PROGRAM ACCEPT MODE ONCE
                               18.  * AGAIN
                               19.  ***********************************************
                               20   * THIS VERSION AWAITS SLAVE 0 TO COMPLETE FIRST,
                               21.  * UPLOADS ITS RESULTS, THEN AWAITS ALL OF SLAVES
                               22.  * 1,2,3 THEN UPLOADS THEIR RESULTS IN ORDER
                               23.  * 1,2,3
                               24.  ***********************************************
                               25.  *
                               26.  * EQUATES,XREFS AND XDEFS
                               27.  *
                               28.          XDEF DATAII,PGCR,ENABLEB,ABORT,NPT
                               29.          XDEF NPTE,NPE,NPTE4,SHIFT14,TCR
                               30.          XDEF CNTSTRT,ENABLTM
                               31.          XREF S0PRO,S1PRO,S2PRO,S3PRO,DONE
                               32.
# 00012FC0                     33.  PGCR EQU $12FC0
# 00000010                     34.  ENABLEA EQU $10
# 00000020                     35.  ENABLEB EQU $20
# 00000003                     36.  SRQASRT EQU 3
# 00000004                     37.  ADBYTIN EQU 4
# 00000007                     38.  ADBYTOT EQU 7
# 00000005                     39.  INDATA  EQU 5
# 00000006                     40.  OUTDATA EQU 6
# 00000008                     41.  NETCNF  EQU 8
# 00000009                     42.  SRQSRVC EQU 9
# 0000000E                     43.  ABORT   EQU 14
# 00000100                     44.  NPT     EQU $100
# 00000040                     45.  NPTE    EQU $40
# 00000100                     46.  NPTE4   EQU NPTE*4
# 00000004                     47.  NPE     EQU 4
# 0000000E                     48.  SHIFT14 EQU 14
# 00000001                     49.  SLV0    EQU $01
# 00000002                     50.  SLV1    EQU $02
# 00000004                     51.  SLV2    EQU $04
# 00000008                     52.  SLV3    EQU $08
# 0000000F                     53.  ALLSLV  EQU $0F
# 00000006                     54.  NUGAM   EQU 6
# 00012FB0                     55.  PSR01   EQU $12FB0
# 00012FB1                     56.  PSR23   EQU $12FB1
# 00012FB2                     57.  PSR45   EQU $12FB2
# 00012FB3                     58.  PSR67   EQU $12FB3
# 00012FB4                     59.  PSR89   EQU $12FB4
# 00012FB5                     60.  PSR1011 EQU $12FB5
# 00012FB6                     61.  PSR1213 EQU $12FB6
# 00012FB7                     62.  PSR1415 EQU $12FB7
# 00012FB8                     63.  PSR1617 EQU $12FB8
# 00012FB9                     64.  PSR1819 EQU $12FB9
# 00012FD0                     65.  TCR     EQU $12FD0
# 00012FD2                     66.  TPLR    EQU TCR+2
# 00012FD6                     67.  TCNTR   EQU TCR+6
# 0000000A                     68.  SENCLFC EQU 10
# 00000018                     69.  DISBUF8 EQU 24
# 0000001A                     70.  DISBUF2 EQU 26
# 00FFFFFF                     71.  CNTSTRT EQU $00FFFFFF
# 00000001                     72.  ENABLTM EQU 1
# 00000011                     73.  OUTMESC EQU 17
# 00000008                     74.  DIV256  EQU 8
# 00000100                     75.  PASSCNT EQU 256
# 00000004                     76.  EOT     EQU 4
                               77.  *
                               78.  ******************
```

```
                                    79.  * MASTER PROGRAM *
                                    80.  ******************
                                    81.  *
0'000000                            82.          SECTION 0
0'000000  <400>                     83.  DATAIN   DS.B $400
0'000400  <400>                     84.  DATAII   DS.B $400
0'000800  <100>                     85.  XDATA0   DS.B $100
0'000900  <100>                     86.  XDATA1   DS.B $100
0'000A00  <100>                     87.  XDATA2   DS.B $100
0'000B00  <100>                     88.  XDATA3   DS.B $100
0'000C00  BB BB BB CB BB            89.  INITCNF  DC.B $BB,$BB,$BB,$CB,$BB,$8C,$CB,$8B.$B8,$8B
          8C CB 8B B8 8B
0'000C0A  BD BB BD CB BD            90.  BRDCAST  DC.B $BD,$BB,$BD,$CB,$BD,$8C,$CB,$8B,$B8,$8B
          8C CB 8B B8 8B
                                    91.  *
                                    92.  * CONFIGURE NETWORK TO INITIAL STATE, MASTER
                                    93.  * TO SLAVE 0
                                    94.  * RESPOND TO SLAVE 0'S RFP SRQ, AND LOAD
                                    95.  * ITS PROGRAM
                                    96.  *
0'000C14  287C 00012FD0             97.  STRTFFT  MOVEA.L #TCR,A4          ;POINTER TIMER
0'000C1A  4214                      98.           CLR.B (A4)               ;DISABLE F'SURE
0'000C1C  2F3C 00FFFFFF             99.           MOVE.L #CNTSTRT,-(SP)    ;MIN TIME
0'000C22  2F3C 00000000            100.           MOVE.L #$00,-(SP)        ;MAX TIME
0'000C28  2F3C 00000000            101.           MOVE.L #$00,-(SP)        ;RUN'G SUM TIME
0'000C2E  2E3C 000000FF            102.           MOVE.L #PASSCNT-1,D7     ;COUNTER
0'000C34  41FA FFCA                103.           LEA INITCNF(PC),A0
0'000C38  4E48                     104.           TRAP #NETCNF
0'000C3A  103C 0001                105.           MOVE.B #SLV0,D0
0'000C3E  4E49                     106.           TRAP #SRQSRVC
0'000C40  41FA ****                107.           LEA S0PRO(PC),A0         ;STRT ADDR
0'000C44  43FA ****                108.           LEA S1PRO(PC),A1         ;END ADDR+1
0'000C48  6100 0274                109.           BSR SENDOUT              ;ADDR,COUNT,CODE
                                   110.  *
                                   111.  * CONFIGURE NETWORK TO MASTER TO SLAVE 1
                                   112.  * RESPOND TO SLAVE 1'S RFP SRQ, AND LOAD ITS
                                   113.  * PROGRAM
                                   114.  *
0'000C4C  13FC 00B4                115.           MOVE.B #$B4,PSR01
          00012FB0
0'000C54  103C 0002                116.           MOVE.B #SLV1,D0
0'000C58  4E49                     117.           TRAP #SRQSRVC
0'000C5A  41FA ****                118.           LEA S1PRO(PC),A0
0'000C5E  43FA ****                119.           LEA S2PRO(PC),A1
0'000C62  6100 025A                120.           BSR SENDOUT
                                   121.  *
                                   122.  * CONFIGURE NETWORK TO MASTER TO SLAVE 2
                                   123.  * RESPOND TO SLAVE 2'S RFP SRQ, AND LOAD ITS
                                   124.  * PROGRAM
                                   125.  *
0'000C66  13FC 00B4                126.           MOVE.B #$B4,PSR45
          00012FB2
0'000C6E  103C 0004                127.           MOVE.B #SLV2,D0
0'000C72  4E49                     128.           TRAP #SRQSRVC
0'000C74  41FA ****                129.           LEA S2PRO(PC),A0
0'000C78  43FA ****                130.           LEA S3PRO(PC),A1
0'000C7C  6100 0240                131.           BSR SENDOUT
                                   132.  *
                                   133.  * CONFIGURE NETWORK TO MASTER TO SLAVE 3
                                   134.  * RESPOND TO SLAVE 3'S RFP SRQ, AND LOAD ITS
                                   135.  * PROGRAM
                                   136.  *
0'000C80  13FC 00B4                137.           MOVE.B #$B4,PSR89
          00012FB4
0'000C88  103C 0008                138.           MOVE.B #SLV3,D0
0'000C8C  4E49                     139.           TRAP #SRQSRVC
0'000C8E  41FA ****                140.           LEA S3PRO(PC),A0
0'000C92  43FA ****                141.           LEA DONE(PC),A1
0'000C96  6100 0226                142.           BSR SENDOUT
                                   143.  *
                                   144.  * ALL PROGRAMS LOADED, EACH ASSERTING READY FOR
                                   145.  * DATA SRQ.  RESPOND TO ALL, THEN LOAD THE DATA
                                   146.  * IN BROADCAST MODE. FIRST COPY INPUT DATA TO
                                   147.  * THE XDATA0-3 ARRAY, THEN AT END DO UNSHUFFLE
                                   148.  * BACK INTO DATAII ARRAY
                                   149.  *
                                   150.  * LOAD TIMER PRE-LOAD REGISTER HERE
                                   151.  *
0'000C9A  41FA F364                152.  PEEXE    LEA DATAIN(PC),A0        ;SOURCE
0'000C9E  43FA F760                153.           LEA DATAII(PC),A1        ,DEST
0'000CA2  303C 00FF                154.           MOVE.W #NPT-1,D0         ;LWORD COUNT
```

```
0'000CA6  22D8              155.  MOVLP   MOVE.L (A0)+,(A1)+       ;LONG WORDS
0'000CA8  51C8 FFFC         156.          DBRA D0,MOVLP
0'000CAC  297C 00FFFFFF     157.          MOVE.L #CNTSTRT,2(A4)
          0002
0'000CB4  41FA FF54         158.          LEA BRDCAST(PC),A0
0'000CB8  4E48              159.          TRAP #NETCNF
0'000CBA  103C 000F         160.          MOVE.B #ALLSLV,D0
0'000CBE  4E49              161.          TRAP #SRQSRVC
0'000CC0  41FA F73E         162.          LEA DATAII(PC),A0        ;START OF DATA
0'000CC4  43FA FB3A         163.          LEA XDATA0(PC),A1        ;END OF DATA+1
                            164.  *
                            165.  * SET UP SOME REGISTER VALUES
                            166.  *
0'000CC8  3C3C 0100         167.          MOVE.W #NPTE*4,D6
0'000CCC  267C 000003FC     168.          MOVEA.L #$3FC,A3
0'000CD2  247C 00000010     169.          MOVEA.L #$10,A2
0'000CD8  2C7C 00012FC0     170.          MOVEA.L #PGCR,A6
0'000CDE  2A7C 00012FB0     171.          MOVEA.L #PSR01,A5
                            172.  *
                            173.  * START TIMER COUNTING HERE
                            174.  *
0'000CE4  18BC 0001         175.          MOVE.B #ENABLTM,(A4)
0'000CE8  6100 01D4         176.          BSR SENDOUT
                            177.  *
                            178.  * WHEN HERE, ALL SLAVES LOADED, WILL ASSERT
                            179.  * READY TO EXECUTE SRQ.  RESPOND TO LET THEM
                            180.  * CONTINUE.  NETWORK IS CONFIGURED TO ACCEPT
                            181.  * DATA FROM SLAVE 0.
                            182.  *
0'000CEC  700F              183.          MOVEQ.L #ALLSLV,D0
0'000CEE  4E49              184.          TRAP #SRQSRVC            ;SLAVES EXECUTE
                            185.  *
                            186.  * WHEN DONE, SLAVES ASSERT SRQ, THEN UPLOAD DATA
                            187.  * WHEN ACKNOWLEDGED.  ACKNOWLEDGE SLAVE 0, THEN
                            188.  * AND UPLOAD ITS RESULTS
                            189.  *
0'000CF0  7001              190.          MOVEQ.L #SLV0,D0
0'000CF2  4E49              191.          TRAP #SRQSRVC
0'000CF4  3006              192.          MOVE.W D6,D0         ;BYTE COUNT
0'000CF6  41FA FB08         193.          LEA XDATA0(PC),A0        ;DEST ADDRESS
0'000CFA  1CBC 0010         194.          MOVE.B #ENABLEA,(A6)
0'000CFE  4E45              195.          TRAP #INDATA
0'000D00  4216              196.          CLR.B (A6)
                            197   *
                            198.  * CONFIGURE FOR SLAVE #1 TO MASTER, WAIT FOR ALL
                            199.  * SLAVES 1,2,3 REQUESTS, ACKNOWLEDGE
                            200.  * AND GET DATA
                            201.  *
0'000D02  103C 000E         202.          MOVE.B #$0E,D0
0'000D06  4E49              203.          TRAP #SRQSRVC
0'000D08  1B7C 00BC 0005    204.          MOVE.B #$BC,5(A5)
0'000D0E  3006              205.          MOVE.W D6,D0
0'000D10  41FA FBEE         206.          LEA XDATA1(PC),A0
0'000D14  1CBC 0010         207.          MOVE.B #ENABLEA,(A6)
0'000D18  4E45              208.          TRAP #INDATA
0'000D1A  4216              209.          CLR.B (A6)
                            210.  *
                            211.  * CONFIGURE FOR SLAVE #2 TO MASTER,
                            212.  * AND GET DATA
                            213.  *
0'000D1C  1B7C 00BB 0007    214.          MOVE.B #$BB,7(A5)
0'000D22  3006              215.          MOVE.W D6,D0
0'000D24  41FA FCDA         216.          LEA XDATA2(PC),A0
0'000D28  1CBC 0010         217.          MOVE.B #ENABLEA,(A6)
0'000D2C  4E45              218.          TRAP #INDATA
0'000D2E  4216              219.          CLR.B (A6)
                            220.  *
                            221.  * CONFIGURE FOR SLAVE #3 TO MASTER,
                            222.  * AND GET DATA
                            223.  *
0'000D30  1B7C 00BB 0009    224.          MOVE.B #$BB,9(A5)
0'000D36  3006              225.          MOVE.W D6,D0
0'000D38  41FA FDC6         226.          LEA XDATA3(PC),A0
0'000D3C  1CBC 0010         227.          MOVE.B #ENABLEA,(A6)
0'000D40  4E45              228.          TRAP #INDATA
0'000D42  4216              229.          CLR.B (A6)
                            230.  *
                            231.  * HAVE ALL DATA IN, NOW DO REARRANGE FROM THE
                            232.  * XDATAi ARRAY BACK TO DATAII ARRAY FOR CORRECT
                            233.  * ORDER. BEWARE, HEAVILY OPTIMIZED CODE HERE
                            234.  *
```

```
0'000D44  41FA FABA       235.  REORDER LEA XDATA0(PC),A0
0'000D48  43FA F6B6       236.           LEA DATAII(PC),A1
0'000D4C  7C03            237.           MOVEQ.L #NPE-1,D6
0'000D4E  7A3F            238.  RLOOPT   MOVEQ.L #NPTE-1,D5
0'000D50  2298            239.  RLOOPN   MOVE.L (A0)+,(A1)
0'000D52  D3CA            240.           ADDA.L A2,A1
0'000D54  51CD FFFA       241.           DBRA D5,RLOOPN
0'000D58  93CB            242.           SUBA.L A3,A1       ;USE REGS TO OPT
0'000D5A  51CE FFF2       243.           DBRA D6,RLOOPT
                          244.  *
                          245.  * PROGRAM IS DONE, CAN STOP TIMER, AND
                          246.  * OUTPUT NUMBER OF 32 CLOCK INTERVALS
                          247.  * PASSED. FIRST DISABLE TIMER
                          248.  *
0'000D5E  4214            249.           CLR.B (A4)
0'000D60  222C 0006       250.           MOVE.L 6(A4),D1 ;GET NEW COUNT
0'000D64  203C 00FFFFFF   251.           MOVE.L #CNTSTRT,D0
0'000D6A  9081            252.           SUB.L D1,D0      ;GET ELAPSED COUNT
                          253.  *
                          254.  * OUTPUT CR, LF, THEN THE COUNT IN HEX
                          255.  *
0'000D6C  4E4F            256.           TRAP #15
0'000D6E  000A            257.           DC.W SENCLFC
0'000D70  4E4F            258.           TRAP #15
0'000D72  0018            259.           DC.W DISBUF8
                          260.  *
                          261.  * NOW KEEP TRACK OF STATISTICS
                          262.  *
0'000D74  D197            263.           ADD.L D0,(SP)    ;ADD TO SUM
0'000D76  B0AF 0004       264.           CMP.L 4(SP),D0   ;CHECK MAX
0'000D7A  6F 04           265.           BLE.S NOTMAX     ;D0 NOT MAX
0'000D7C  2F40 0004       266.           MOVE.L D0,4(SP)  ;NEW MAX
0'000D80  B0AF 0008       267.  NOTMAX   CMP.L 8(SP),D0   ;CHECK MIN
0'000D84  6C 04           268.           BGE.S NOTMIN     ;D0 NOT MIN
0'000D86  2F40 0008       269.           MOVE.L D0,8(SP)  ;NEW MIN
0'000D8A  51CF FF0E       270.  NOTMIN   DBRA D7,REEXE    ;DO D7 TIMES
                          271.
                          272.  *
                          273.  * WHEN HERE, PROGRAM HAS BEEN EXECUTED 256 TIMES
                          274.  * SEND AN ILLEGAL ADDRESS FOR DATA, FORCING
                          275.  * SLAVES TO PROGRAM ACCEPT MODE
                          276.  *
0'000D8E  41FA FE7A       277.           LEA BRDCAST(PC),A0
0'000D92  4E48            278.           TRAP #NETCNF
0'000D94  103C 000F       279.           MOVE.B #ALLSLV,D0
0'000D98  4E49            280.           TRAP #SRQSRVC
0'000D9A  207C FFFF0000   281.           MOVEA.L #$FFFF0000,A0    ;ILLEGAL ADDR
0'000DA0  13FC 0020       282.           MOVE.B #ENABLEB,PGCR
          00012FC0
0'000DA8  4E47            283.           TRAP #ADBYTOT    ;THAT'LL FIX 'EM
0'000DAA  4239 00012FC0   284.           CLR.B PGCR
                          285.  *
                          286.  * OUTPUT STATISTICS
                          287.  *
0'000DB0  4E4F            288.           TRAP #15
0'000DB2  000A            289.           DC.W SENCLFC
0'000DB4  4E4F            290.           TRAP #15
0'000DB6  000A            291.           DC.W SENCLFC
0'000DB8  41FA 011C       292.           LEA AVEMES(PC),A0
0'000DBC  4E4F            293.           TRAP #15
0'000DBE  0011            294.           DC.W OUTMESC
0'000DC0  201F            295.           MOVE.L (SP)+,D0 ;GET SUM
0'000DC2  E080            296.           ASR.L #DIV256,D0         ;DO AVERAGE
0'000DC4  4E4F            297.           TRAP #15
0'000DC6  0018            298.           DC.W DISBUF8     ;OUTPUT AVERAGE
0'000DC8  4E4F            299.           TRAP #15
0'000DCA  000A            300.           DC.W SENCLFC
0'000DCC  41FA 012F       301.           LEA MAXMES(PC),A0
0'000DD0  4E4F            302.           TRAP #15
0'000DD2  0011            303.           DC.W OUTMESC
0'000DD4  201F            304.           MOVE.L (SP)+,D0 ;GET MAX
0'000DD6  4E4F            305.           TRAP #15
0'000DD8  0018            306.           DC.W DISBUF8
0'000DDA  4E4F            307.           TRAP #15
0'000DDC  000A            308.           DC.W SENCLFC
0'000DDE  41FA 0144       309.           LEA MINMES(PC),A0
0'000DE2  4E4F            310.           TRAP #15
0'000DE4  0011            311.           DC.W OUTMESC
0'000DE6  201F            312.           MOVE.L (SP)+,D0 ;GET MIN
0'000DE8  4E4F            313.           TRAP #15
0'000DEA  0018            314.           DC.W DISBUF8
```

```
0'000DEC  4E4F             315.          TRAP #15
0'000DEE  000A             316.          DC.W SENCLFC
                           317.  *
                           318.  * AFTER 256 RUNS, LOAD THE SECOND SLAVE
                           319.  * PROGRAM TO ACQUIRE THE TIMING DATA
                           320.  * IN THE SLAVES TIMER COUNT REGISTERS
                           321.  *
0'000DF0  41FA FE18        322.          LEA BRDCAST(PC),A0      ;BROADCAST MODE
0'000DF4  4E48             323.          TRAP #NETCNF
                           324.  *
                           325.  * RESPOND TO SLAVES RFP
                           326.  *
0'000DF6  103C 000F        327.          MOVE.B #ALLSLV,D0
0'000DFA  4E49             328.          TRAP #SRQSRVC
0'000DFC  41FA ****        329.          LEA SLVPRO2(PC),A0
0'000E00  43FA ****        330.          LEA ENDSLV2(PC),A1
0'000E04  93C8             331.          SUBA.L A0,A1
0'000E06  3009             332.          MOVE.W A1,D0
0'000E08  13FC 0020        333.          MOVE.B #ENABLEB,PGCR
          00012FC0
0'000E10  4E47             334.          TRAP #ADBYTOT
0'000E12  4E46             335.          TRAP #OUTDATA
0'000E14  4239 00012FC0    336.          CLR.B PGCR       ;PROGRAM SENT
                           337.  *
                           338.  * RESPOND TO SLAVES RFD
                           339.  *
0'000E1A  103C 000F        340.          MOVE.B #ALLSLV,D0
0'000E1E  4E49             341.          TRAP #SRQSRVC
                           342.  *
                           343.  * PROGRAM HAS NO INPUT DATA, SEND DUMMY
                           344.  * ADDRESS AND ZERO BYTE COUNT
                           345.  * AND THEN NO DATA
                           346.  *
0'000E20  4240             347.          CLR.W D0
0'000E22  13FC 0020        348.          MOVE.B #ENABLEB,PGCR
          00012FC0
0'000E2A  4E47             349.          TRAP #ADBYTOT
0'000E2C  4239 00012FC0    350.          CLR.B PGCR
                           351.  *
                           352.  * RESPOND TO READY TO EXECUTE
                           353.  *
0'000E32  103C 000F        354.          MOVE.B #ALLSLV,D0
0'000E36  4E49             355.          TRAP #SRQSRVC
                           356.  *
                           357.  * SLAVES EXECUTE AND ASSERT AN SRQ WHEN
                           358.  * READY TO REPORT  RESPOND ONE AT A TIME
                           359.  * DETERMINE TIME ELAPSED AND REPORT TO
                           360.  * CONSOLE
                           361.  *
0'000E38  103C 0001        362.          MOVE.B #SLV0,D0 ;SLAVE 0
0'000E3C  4281             363.          CLR.L D1
0'000E3E  61 32            364.          BSR.S GETSLTM
0'000E40  13FC 00BC        365.          MOVE.B #$BC,PSR1011 ;SLAVE 1
          00012FB5
0'000E48  5281             366.          ADDQ.L #1,D1
0'000E4A  103C 0002        367.          MOVE.B #SLV1,D0
0'000E4E  61 22            368.          BSR.S GETSLTM
0'000E50  13FC 00BB        369.          MOVE.B #$BB,PSR1415 ;SLAVE 2
          00012FB7
0'000E58  5281             370.          ADDQ.L #1,D1
0'000E5A  103C 0004        371.          MOVE.B #SLV2,D0
0'000E5E  61 12            372.          BSR.S GETSLTM
0'000E60  13FC 00BB        373.          MOVE.B #$BB,PSR1819 ;SLAVE3
          00012FB9
0'000E68  5281             374.          ADDQ.L #1,D1
0'000E6A  103C 0008        375.          MOVE.B #SLV3,D0
0'000E6E  61 02            376.          BSR.S GETSLTM
                           377.  *
                           378.  * NOW FINALLY DONE!
                           379.  *
0'000E70  4E75             380.          RTS
                           381.  *
                           382.  * SUBROUTINE TO GET TIMING DATA FROM SLAVE
                           383.  * SPECIFIED IN D1.B, WITH MASK IN D0.B
                           384.  *
0'000E72  48E7 C0C0        385.  GETSLTM MOVEM.L D0-D1/A0-A1,-(SP) ;SAVE ENV
0'000E76  4E49             386.          TRAP #SRQSRVC
0'000E78  7004             387.          MOVEQ.L #$4,D0  ;BYTE COUNT
0'000E7A  41FA 0106        388.          LEA TEMPCNT(PC),A0
0'000E7E  13FC 0010        389.          MOVE.B #ENABLEA,PGCR    ;ENABLE INPUT
          00012FC0
```

```
0'000E86  4E45            390.          TRAP #INDATA
0'000E88  4239 00012FC0   391.          CLR.B PGCR
0'000E8E  2248            392.          MOVEA.L A0,A1
0'000E90  41FA 00B9       393.          LEA MESSA(PC),A0
0'000E94  4E4F            394.          TRAP #15
0'000E96  0011            395.          DC.W OUTMESC
0'000E98  1001            396.          MOVE.B D1,D0
0'000E9A  4E4F            397.          TRAP #15
0'000E9C  001A            398.          DC.W DISBUF2
0'000E9E  41FA 00C5       399.          LEA MESSB(PC),A0
0'000EA2  4E4F            400.          TRAP #15
0'000EA4  0011            401.          DC.W OUTMESC
0'000EA6  2211            402.          MOVE.L (A1),D1
0'000EA8  203C 00FFFFFF   403.          MOVE.L #CNTSTRT,D0 ;CALC ELAPSED
0'000EAE  9081            404.          SUB.L D1,D0
0'000EB0  4E4F            405.          TRAP #15
0'000EB2  0018            406.          DC.W DISBUF8
0'000EB4  4E4F            407.          TRAP #15
0'000EB6  000A            408.          DC.W SENCLFC
0'000EB8  4CDF 0303       409.          MOVEM.L (SP)+,D0-D1/A0-A1
0'000EBC  4E75            410.          RTS
                          411.  *
                          412.  * SUBROUTINE SENDOUT
                          413.  *
0'000EBE  93C8            414.  SENDOUT SUBA.L A0,A1
0'000EC0  3009            415.          MOVE.W A1,D0
0'000EC2  13FC 0020       416.          MOVE.B #ENABLEB,PGCR
          00012FC0
0'000ECA  4E47            417.          TRAP #ADBYTOT
0'000ECC  4E46            418.          TRAP #OUTDATA
0'000ECE  4239 00012FC0   419.          CLR.B PGCR
0'000ED4  4E75            420.          RTS
0'000ED6  41 76 65 72 61  421.  AVEMES  DC.B 'Average time ( x 4 for microseconds)= ',EOT
          67 65 20 74 69
          6D 65 20 28 20
          78 20 34 20 66
          6F 72 20 6D 69
          63 72 6F 73 65
          63 6F 6E 64 73
          29 3D 20 04
0'000EFD  4D 61 78 69 6D  422.  MAXMES  DC.B 'Maximum time ( x 4 for microseconds)= ',EOT
          75 6D 20 74 69
          6D 65 20 28 20
          78 20 34 20 66
          6F 72 20 6D 69
          63 72 6F 73 65
          63 6F 6E 64 73
          29 3D 20 04
0'000F24  4D 69 6E 69 6D  423.  MINMES  DC.B 'Minimum time ( x 4 for microseconds)= ',EOT
          75 6D 20 74 69
          6D 65 20 28 20
          78 20 34 20 66
          6F 72 20 6D 69
          63 72 6F 73 65
          63 6F 6E 64 73
          29 3D 20 04
0'000F4B  45 78 65 63 75  424.  MESSA   DC.B 'Execution time by slave #',EOT
          74 69 6F 6E 20
          74 69 6D 65 20
          62 79 20 73 6C
          61 76 65 20 23
          04
0'000F65  20 20 28 20 78  425.  MESSB   DC.B '   ( x 4 for microseconds) = ',EOT
          20 34 20 66 6F
          72 20 6D 69 63
          72 6F 73 65 63
          6F 6E 64 73 29
          20 3D 20 04
0'000F82  <4>             426.  TEMPCNT DS.B $4     ;TEMPORARY FOR SLAVE SELF TIME
                          427.  *
                          428.  *
1'000000                  429.          SECTION 1
                          430.  *
                          431.  ******************************
                          432.  * ALL SLAVES PROG 2
                          433.  ******************************
                          434.  *
                          435.  * PROGRAM TO REPORT CONTENTS OF THE
                          436.  * COUNTER REGISTER
                          437.  *
1'000000  207C 00012FD6   438.  SLVPRO2 MOVEA.L #TCR+6,A0       ;POINTER
```

```
1'000006  7004             439.          MOVEQ.L #$4,D0           ;BYTE COUNT
1'000008  4E43             440.          TRAP #SRQASRT            ;ASSERT SRQ
1'00000A  13FC 0020        441.          MOVE.B #ENABLEB,PGCR     ;OUTPUT ENABLED
          00012FC0
1'000012  4E46             442.          TRAP #OUTDATA            ;SEND BLOCK
1'000014  4239 00012FC0    443.          CLR.B PGCR
1'00001A  4E4E             444.          TRAP #ABORT      ;GOTO PROG ACCEPT MODE
1'00001C  <1>              445.  ENDSLV2 DS.B 1
1'00001D                   446.          END
    0 Errors
*
* LINK SPEC FOR PFFT1
*
      LINK PFFT3
        LINK SLVFFT
      ORG $1000
      SECTION 0,15,1
      END
```

L.5 Program PFFT4 Multicomputer FFT test program #4.

```
Quelo  ...A68K D6.2Ja Dec 23 1986   ...Run on Aug 26, 1991  10:58:25 ...Page  1
PFFT4.LTX , PFFT4.PRN , PFFT4.SYM = PFFT4.A68
   ...PFFT4
                                1.           TTL PFFT4
                                2.   *
                                3.   * THIS PROGRAM PERFORMS A 256-COMPLEX POINT
                                4.   * FFT USING A PARALLEL MODE IN THE RMPS
                                5.   * input/output data is stored as real/imaginary
                                6.   * pair, each component 16-bits long.  Input data
                                7.   * is at DATAII, output data at OUTDATA.
                                8.   *
                                9.   * SLAVE PROGRAMS PUT SLAVE IN READY FOR DATA
                               10.   * MODE AFTER EACH EXECUTION. AFTER 256 PASSES
                               11.   * MASTER SENDS ILLEGAL DATA START ADDRESS TO
                               12.   * PUT SLAVES BACK IN PROGRAM ACCEPT MODE
                               13.   * NEW PROGRAM IS SENT TO UPLOAD INDIVIDUAL
                               14.   * TIMING RESULTS FROM SLAVES.  THAT PROGRAM
                               15.   * ENDS WITH
                               16.   *       TRAP #ABORT
                               17.   * TO PUT SLAVES INTO PROGRAM ACCEPT MODE ONCE
                               18.   * AGAIN
                               19.   ************************************************
                               20.   * THIS VERSION DOES A FIRST COME FIRST SERVE
                               21.   * METHOD FOR SENSING WHEN TO UPLOAD DATA FROM
                               22.   * SLAVES
                               23.   ************************************************
                               24.   *
                               25.   * EQUATES,XREFS AND XDEFS
                               26.   *
                               27.           XDEF DATAII,PGCR,ENABLEB,ABORT,NPT
                               28.           XDEF NPTE,NPE,NPTE4,SHIFT14,TCR
                               29.           XDEF CNTSTRT,ENABLTM
                               30.           XREF S0PRO,S1PRO,S2PRO,S3PRO,DONE
# 00012FCC                     31.   PCDR EQU $12FCC
# 00012FC0                     32.   PGCR EQU $12FC0
# 00000010                     33.   ENABLEA EQU $10
# 00000020                     34.   ENABLEB EQU $20
# 00000003                     35.   SRQASRT EQU 3
# 00000004                     36.   ADBYTIN EQU 4
# 00000007                     37.   ADBYTOT EQU 7
# 00000005                     38.   INDATA  EQU 5
# 00000006                     39.   OUTDATA EQU 6
# 00000008                     40.   NETCNF  EQU 8
# 00000009                     41.   SRQSRVC EQU 9
# 0000000E                     42.   ABORT   EQU 14
# 00000100                     43.   NPT     EQU $100
# 00000040                     44.   NPTE    EQU $40
# 00000100                     45.   NPTE4   EQU NPTE*4
# 00000004                     46.   NPE     EQU 4
# 0000000E                     47.   SHIFT14 EQU 14
# 00000001                     48.   SLV0    EQU $01
# 00000002                     49.   SLV1    EQU $02
# 00000004                     50.   SLV2    EQU $04
# 00000008                     51.   SLV3    EQU $08
# 0000000F                     52.   ALLSLV  EQU $0F
# 00000006                     53.   NUGAM   EQU 6
# 00012FB0                     54.   PSR01   EQU $12FB0
# 00012FB1                     55.   PSR23   EQU $12FB1
# 00012FB2                     56.   PSR45   EQU $12FB2
# 00012FB3                     57.   PSR67   EQU $12FB3
# 00012FB4                     58.   PSR89   EQU $12FB4
# 00012FB5                     59.   PSR1011 EQU $12FB5
# 00012FB6                     60.   PSR1213 EQU $12FB6
# 00012FB7                     61.   PSR1415 EQU $12FB7
# 00012FB8                     62.   PSR1617 EQU $12FB8
# 00012FB9                     63.   PSR1819 EQU $12FB9
# 00012FD0                     64.   TCR     EQU $12FD0
# 00012FD2                     65.   TPLR    EQU TCR+2
# 00012FD6                     66.   TCNTR   EQU TCR+6
# 0000000A                     67.   SENCLFC EQU 10
# 00000018                     68.   DISBUF8 EQU 24
# 0000001A                     69.   DISBUF2 EQU 26
# 00FFFFFF                     70.   CNTSTRT EQU $00FFFFFF
# 00000001                     71.   ENABLTM EQU 1
# 00000011                     72.   OUTMESC EQU 17
# 00000008                     73.   DIV256  EQU 8
# 00000100                     74.   PASSCNT EQU 256
# 00000004                     75.   EOT     EQU 4
```

```
                                    76.  *
                                    77.  ******************
                                    78.  * MASTER PROGRAM *
                                    79.  ******************
                                    80.  *
0'000000                            81.          SECTION 0
0'000000  <400>                     82.  DATAIN   DS.B $400
0'000400  <400>                     83.  DATAII   DS.B $400
0'000800  <100>                     84.  XDATA0   DS.B $100
0'000900  <100>                     85.  XDATA1   DS.B $100
0'000A00  <100>                     86.  XDATA2   DS.B $100
0'000B00  <100>                     87.  XDATA3   DS.B $100
0'000C00  BB BB BB CB BB            88.  INITCNF DC.B $BB,$BB,$BB,$CB,$BB,$8C,$CB,$8B,$B8,$8B
          8C CB 8B B8 8B
0'000C0A  BD BB BD CB BD            89.  BRDCAST DC.B $BD,$BB,$BD,$CB,$BD,$8C,$CB,$8B,$B8,$8B
          8C CB 8B B8 8B
                                    90.  *
                                    91.  * CONFIGURE NETWORK TO INITIAL STATE, MASTER
                                    92.  * TO SLAVE 0
                                    93.  * RESPOND TO SLAVE 0'S RFP SRQ, AND LOAD
                                    94.  * ITS PROGRAM
                                    95.  *
0'000C14  287C 00012FD0             96.  STRTFFT MOVEA.L #TCR,A4          ;POINTER TIMER
0'000C1A  4214                      97.          CLR.B (A4)               ;DISABLE F'SURE
0'000C1C  2F3C 00FFFFFF             98.          MOVE.L #CNTSTRT,-(SP)    ;MIN TIME
0'000C22  2F3C 00000000             99.          MOVE.L #$00,-(SP)        ;MAX TIME
0'000C28  2F3C 00000000            100.          MOVE.L #$00,-(SP)        ;RUN'G SUM TIME
0'000C2E  2E3C 000000FF            101.          MOVE.L #PASSCNT-1,D7     ;COUNTER
0'000C34  41FA FFCA                102.          LEA INITCNF(PC),A0
0'000C38  4E48                     103.          TRAP #NETCNF
0'000C3A  103C 0001                104.          MOVE.B #SLV0,D0
0'000C3E  4E49                     105.          TRAP #SRQSRVC
0'000C40  41FA ****                106.          LEA S0PRO(PC),A0         ;STRT ADDR
0'000C44  43FA ****                107.          LEA S1PRO(PC),A1         ;END ADDR+1
0'000C48  6100 0338                108.          BSR SENDOUT              ;ADDR,COUNT,CODE
                                   109.  *
                                   110.  * CONFIGURE NETWORK TO MASTER TO SLAVE 1
                                   111.  * RESPOND TO SLAVE 1'S RFP SRQ, AND LOAD ITS
                                   112.  * PROGRAM
                                   113.  *
0'000C4C  13FC 00B4                114.          MOVE.? #$B4,PSR01
          00012FB0
0'000C54  103C 0002                115.          MOVE.B #SLV1,D0
0'000C58  4E49                     116.          TRAP #SRQSRVC
0'000C5A  41FA ****                117.          LEA S1PRO(PC),A0
0'000C5E  43FA ****                118.          LEA S2PRO(PC),A1
0'000C62  6100 031E                119.          BSR SENDOUT
                                   120.  *
                                   121.  * CONFIGURE NETWORK TO MASTER TO SLAVE 2
                                   122.  * RESPOND TO SLAVE 2'S RFP SRQ, AND LOAD ITS
                                   123.  * PROGRAM
                                   124.  *
0'000C66  13FC 00B4                125.          MOVE.B #$B4,PSR45
          00012FB2
0'000C6E  103C 0004                126.          MOVE.B #SLV2,D0
0'000C72  4E49                     127.          TRAP #SRQSRVC
0'000C74  41FA ****                128.          LEA S2PRO(PC),A0
0'000C78  43FA ****                129.          LEA S3PRO(PC),A1
0'000C7C  6100 0304                130.          BSR SENDOUT
                                   131.  *
                                   132.  * CONFIGURE NETWORK TO MASTER TO SLAVE 3
                                   133.  * RESPOND TO SLAVE 3'S RFP SRQ, AND LOAD ITS
                                   134.  * PROGRAM
                                   135.  *
0'000C80  13FC 00B4                136.          MOVE.B #$B4,PSR89
          00012FB4
0'000C88  103C 0008                137.          MOVE.B #SLV3,D0
0'000C8C  4E49                     138.          TRAP #SRQSRVC
0'000C8E  41FA ****                139.          LEA S3PRO(PC),A0
0'000C92  43FA ****                140.          LEA DONE(PC),A1
0'000C96  6100 02EA                141.          BSR SENDOUT
                                   142.  *
                                   143.  * ALL PROGRAMS LOADED, EACH ASSERTING READY FOR
                                   144.  * DATA SRQ.  RESPOND TO ALL, THEN LOAD THE DATA
                                   145.  * IN BROADCAST MODE. FIRST COPY INPUT DATA TO
                                   146.  * THE XDATA0-3 ARRAY, THEN AT END DO UNSHUFFLE
                                   147.  * BACK INTO DATAII ARRAY
                                   148.  *
                                   149.  * LOAD TIMER PRE-LOAD REGISTER HERE
                                   150.  *
0'000C9A  41FA F364                151.  REEXE    LEA DATAIN(PC),A0        ;SOURCE
```

```
0'000C9E  43FA F760        152.          LEA DATAII(PC),A1        ;DEST
0'000CA2  303C 00FF        153.          MOVE.W #NPT-1,D0         ;LWORD COUNT
0'000CA6  22D8             154.  MOVLP   MOVE.L (A0)+,(A1)+       ;LONG WORDS
0'000CA8  51C8 FFFC        155.          DBRA D0,MOVLP
0'000CAC  297C 00FFFFFF    156.          MOVE.L #CNTSTRT,2(A4)
          0002
0'000CB4  41FA FF54        157.          LEA BRDCAST(PC),A0
0'000CB8  4E48             158.          TRAP #NETCNF
0'000CBA  103C 000F        159.          MOVE.B #ALLSLV,D0
0'000CBE  4E49             160.          TRAP #SRQSRVC
0'000CC0  41FA F73E        161.          LEA DATAII(PC),A0        ;START OF DATA
0'000CC4  43FA FB3A        162.          LEA XDATA0(PC),A1        ;END OF DATA+1
                           163.  *
                           164.  * SET UP SOME REGISTER VALUES
                           165.  *
0'000CC8  3C3C 0100        166.          MOVE.W #NPTE*4,D6
0'000CCC  267C 000003FC    167.          MOVEA.L #$3FC,A3
0'000CD2  247C 00000010    168.          MOVEA.L #$10,A2
0'000CD8  2C7C 00012FC0    169.          MOVEA.L #PGCR,A6
0'000CDE  2A7C 00012FB0    170.          MOVEA.L #PSR01,A5
                           171.  *
                           172.  * START TIMER COUNTING HERE
                           173.  *
0'000CE4  18BC 0001        174.          MOVE.B #ENABLTM,(A4)
0'000CE8  6100 0298        175.          BSR SENDOUT
                           176.  *
                           177.  * WHEN HERE, ALL SLAVES LOADED, WILL ASSERT
                           178.  * READY TO EXECUTE SRQ.  RESPOND TO LET THEM
                           179.  * CONTINUE.  NETWORK IS CONFIGURED TO ACCEPT
                           180.  * DATA FROM SLAVE 0.
                           181.  *
0'000CEC  700F             182.          MOVEQ.L #ALLSLV,D0
0'000CEE  4E49             183.          TRAP #SRQSRVC            ;SLAVES EXECUTE
                           184.  *
                           185.  * WHEN DONE, SLAVES ASSERT SRQ, THEN UPLOAD DATA
                           186.  * WHEN ACKNOWLEDGED.  ACKNOWLEDGE FCFS, AND UPLOAD
                           187.  * FCFS
                           188.  *
0'000CF0  700F             189.          MOVEQ.L #ALLSLV,D0
0'000CF2  0800 0000        190.  TSTB0   BTST.L #0,D0
0'000CF6  67 44            191.          BEQ.S TSTB1
0'000CF8  123C 0000        192.          MOVE.B #0.D1
0'000CFC  6100 029C        193.          BSR TSTSLV
0'000D00  0801 0007        194.          BTST.L #7,D1     ;SERVICED?
0'000D04  67 36            195.          BEQ.S TSTB1      ;IF NOT GOTO SLV1
                           196.  *
                           197.  * SLAVE 0 SENSED AND ACKNOWLEDGED, GET DATA
                           198.  *
0'000D06  1B7C 0080 0003   199.          MOVE.B #$80,3(A5)
0'000D0C  1B7C 0080 0005   200.          MOVE.B #$80,5(A5)
0'000D12  1B7C 0080 0007   201.          MOVE.B #$80,7(A5)
0'000D18  1B7C 0080 0009   202.          MOVE.B #$80,9(A5)
0'000D1E  3F00             203.          MOVE.W D0,-(SP)
0'000D20  3006             204.          MOVE.W D6,D0         ;BYTE COUNT
0'000D22  41FA FADC        205.          LEA XDATA0(PC),A0        ;DEST ADDRESS
0'000D26  1CBC 0010        206.          MOVE.B #ENABLEA,(A6)
0'000D2A  4E45             207.          TRAP #INDATA
0'000D2C  4216             208.          CLR.B (A6)
0'000D2E  301F             209.          MOVE.W (SP)+,D0
0'000D30  0880 0000        210.          BCLR.L #0,D0     ;CLEAR THE BIT
0'000D34  0C00 0000        211.          CMPI.B #0,D0
0'000D38  6700 00CE        212.          BEQ REORDER
0'000D3C  0800 0001        213.  TSTB1   BTST.L #1,D0
0'000D40  67 3E            214.          BEQ.S TSTB2
0'000D42  123C 0001        215.          MOVE.B #1,D1
0'000D46  6100 0252        216.          BSR TSTSLV
0'000D4A  0801 0007        217.          BTST.L #7,D1     ;SERVICED?
0'000D4E  67 30            218.          BEQ.S TSTB2      ;IF NOT GOTO SLV1
                           219.  *
                           220.  * SLAVE 1 SENSED AND ACKNOWLEDGED, GET DATA
                           221.  *
0'000D50  1B7C 00B8 0005   222.          MOVE.B #$B8,5(A5)
0'000D56  1B7C 0080 0007   223.          MOVE.B #$80,7(A5)
0'000D5C  1B7C 0080 0009   224.          MOVE.B #$80,9(A5)
0'000D62  3F00             225.          MOVE.W D0,-(SP)
0'000D64  3006             226.          MOVE.W D6,D0         ;BYTE COUNT
0'000D66  41FA FB98        227.          LEA XDATA1(PC),A0        ,DEST ADDRESS
0'000D6A  1CBC 0010        228.          MOVE.B #ENABLEA,(A6)
0'000D6E  4E45             229.          TRAP #INDATA
0'000D70  4216             230.          CLR.B (A6)
0'000D72  301F             231.          MOVE.W (SP)+,D0
```

```
0'000D74  0880 0001        232.          BCLR.L #1,D0     ;CLEAR THE BIT
0'000D78  0C00 0000        233.          CMPI.B #0,D0
0'000D7C  6700 008A        234.          BEQ REORDER
0'000D80  0800 0002        235.  TSTB2    BTST.L #2,D0
0'000D84  67 3E            236.          BEQ.S TSTB3
0'000D86  123C 0002        237.          MOVE.B #2,D1
0'000D8A  6100 020E        238.          BSR TSTSLV
0'000D8E  0801 0007        239.          BTST.L #7,D1     ;SERVICED?
0'000D92  67 30            240.          BEQ.S TSTB3      ;IF NOT GOTO SLV1
                           241.  *
                           242.  * SLAVE 2 SENSED AND ACKNOWLEDGED, GET DATA
                           243.  *
0'000D94  1B7C 0080 0006   244.          MOVE.B #$80,6(A5)
0'000D9A  1B7C 00BB 0007   245.          MOVE.B #$BB,7(A5)
0'000DA0  1B7C 0080 0009   246.          MOVE.B #$80,9(A5)
0'000DA6  3F00             247.          MOVE.W D0,-(SP)
0'000DA8  3006             248.          MOVE.W D6,D0         ;BYTE COUNT
0'000DAA  41FA FC54        249.          LEA XDATA2(PC),A0        ;DEST ADDRESS
0'000DAE  1CBC 0010        250.          MOVE.B #ENABLEA,(A6)
0'000DB2  4E45             251.          TRAP #INDATA
0'000DB4  4216             252.          CLR.B (A6)
0'000DB6  301F             253.          MOVE.W (SP)+,D0
0'000DB8  0880 0002        254.          BCLR.L #2,D0     ;CLEAR THE BIT
0'000DBC  0C00 0000        255.          CMPI.B #0,D0
0'000DC0  6700 0046        256.          BEQ REORDER
0'000DC4  0800 0003        257.  TSTB3    BTST.L #3,D0
0'000DC8  6700 FF28        258.          BEQ TSTB0
0'000DCC  123C 0003        259.          MOVE.B #3,D1
0'000DD0  6100 01C8        260.          BSR TSTSLV
0'000DD4  0801 0007        261.          BTST.L #7,D1     ;SERVICED?
0'000DD8  6700 FF18        262.          BEQ TSTB0        ;IF NOT GOTO SLV1
                           263.  *
                           264.  * SLAVE 3 SENSED AND ACKNOWLEDGED, GET DATA
                           265.  *
0'000DDC  1B7C 00B8 0008   266.          MOVE.B #$B8,8(A5)
0'000DE2  1B7C 00BB 0009   267.          MOVE.B #$BB,9(A5)
0'000DE8  3F00             268.          MOVE.W D0,-(SP)
0'000DEA  3006             269.          MOVE.W D6,D0         ;BYTE COUNT
0'000DEC  41FA FD12        270.          LEA XDATA3(PC),A0        ;DEST ADDRESS
0'000DF0  1CBC 0010        271.          MOVE.B #ENABLEA,(A6)
0'000DF4  4E45             272.          TRAP #INDATA
0'000DF6  4216             273.          CLR.B (A6)
0'000DF8  301F             274.          MOVE.W (SP)+,D0
0'000DFA  0880 0003        275.          BCLR.L #3,D0     ;CLEAR THE BIT
0'000DFE  0C00 0000        276.          CMPI.B #0,D0
0'000E02  67 04            277.          BEQ.S REORDER
0'000E04  6000 FEEC        278.          BRA TSTB0
                           279.  *
                           280.  * ALL DATA IS LOADED
                           281.  * HAVE ALL DATA IN, NOW DO REARRANGE FROM THE
                           282.  * XDATAi ARRAY BACK TO DATAII ARRAY FOR CORRECT
                           283.  * ORDER. BEWARE, HEAVILY OPTIMIZED CODE HERE
                           284.  *
0'000E08  41FA F9F6        285.  REORDER LEA XDATA0(PC),A0
0'000E0C  43FA F5F2        286.          LEA DATAII(PC),A1
0'000E10  7C03             287.          MOVEQ.L #NPE-1,D6
0'000E12  7A3F             288.  RLOOPT  MOVEQ.L #NPTE-1,D5
0'000E14  2298             289.  RLOOPW  MOVE.L (A0)+,(A1)
0'000E16  D3CA             290.          ADDA.L A2,A1
0'000E18  51CD FFFA        291.          DBRA D5,RLOOPW
0'000E1C  93CB             292.          SUBA.L A3,A1         ;USE REGS TO OPT
0'000E1E  51CE FFF2        293.          DBRA D6,RLOOPT
                           294.  *
                           295.  * PROGRAM IS DONE, CAN STOP TIMER, AND
                           296.  * OUTPUT NUMBER OF 32 CLOCK INTERVALS
                           297.  * PASSED. FIRST DISABLE TIMER
                           298.  *
0'000E22  4214             299.          CLR.B (A4)
0'000E24  222C 0006        300.          MOVE.L 6(A4),D1 ;GET NEW COUNT
0'000E28  203C 00FFFFFF    301.          MOVE.L #CNTSTRT,D0
0'000E2E  9081             302.          SUB.L D1,D0      ;GET ELAPSED COUNT
                           303.  *
                           304.  * OUTPUT CR, LF, THEN THE COUNT IN HEX
                           305.  *
0'000E30  4E4F             306.          TRAP #15
0'000E32  000A             307.          DC.W SENCLFC
0'000E34  4E4F             308.          TRAP #15
0'000E36  0018             309.          DC.W DISBUF8
                           310.  *
                           311.  * NOW KEEP TRACK OF STATISTICS
                           312.  *
```

```
0'000E38  D197             313.          ADD.L D0,(SP)    ;ADD TO SUM
0'000E3A  B0AF 0004        314.          CMP.L 4(SP),D0   ;CHECK MAX
0'000E3E  6F 04            315.          BLE.S NOTMAX     ;D0 NOT MAX
0'000E40  2F40 0004        316.          MOVE.L D0,4(SP)  ;NEW MAX
0'000E44  B0AF 0008        317.  NOTMAX  CMP.L 8(SP),D0   ;CHECK MIN
0'000E48  6C 04            318.          BGE.S NOTMIN     ;D0 NOT MIN
0'000E4A  2F40 0008        319.          MOVE.L D0,8(SP)  ;NEW MIN
0'000E4E  51CF FE4A        320.  NOTMIN  DBRA D7,REEXE    ;DO D7 TIMES
                           321.
                           322.  *
                           323.  * WHEN HERE, PROGRAM HAS BEEN EXECUTED 256 TIMES
                           324.  * SEND AN ILLEGAL ADDRESS FOR DATA, FORCING
                           325.  * SLAVES TO PROGRAM ACCEPT MODE
                           326.  *
0'000E52  41FA FDB6        327.          LEA BRDCAST(PC),A0
0'000E56  4E48             328.          TRAP #NETCNF
0'000E58  103C 000F        329.          MOVE.B #ALLSLV,D0
0'000E5C  4E49             330.          TRAP #SRQSRVC
0'000E5E  207C FFFF0000    331.          MOVEA.L #$FFFF0000,A0   ;ILLEGAL ADDR
0'000E64  13FC 0020        332.          MOVE.B #ENABLEB,PGCR
          00012FC0
0'000E6C  4E47             333.          TRAP #ADBYTOT    ;THAT'LL FIX 'EM
0'000E6E  4239 00012FC0    334.          CLR.B PGCR
                           335.  *
                           336.  * OUTPUT STATISTICS
                           337.  *
0'000E74  4E4F             338.          TRAP #15
0'000E76  000A             339.          DC.W SENCLFC
0'000E78  4E4F             340.          TRAP #15
0'000E7A  000A             341.          DC.W SENCLFC
0'000E7C  41FA 0156        342.          LEA AVEMES(PC),A0
0'000E80  4E4F             343.          TRAP #15
0'000E82  0011             344.          DC.W OUTMESC
0'000E84  201F             345.          MOVE.L (SP)+,D0 ;GET SUM
0'000E86  E080             346.          ASR.L #DIV256,D0         ,DO AVERAGE
0'000E88  4E4F             347.          TRAP #15
0'000E8A  0018             348.          DC.W DISBUF8     ;OUTPUT AVERAGE
0'000E8C  4E4F             349.          TRAP #15
0'000E8E  000A             350.          DC.W SENCLFC
0'000E90  41FA 0169        351.          LEA MAXMES(PC),A0
0'000E94  4E4F             352.          TRAP #15
0'000E96  0011             353.          DC.W OUTMESC
0'000E98  201F             354.          MOVE.L (SP)+,D0 ;GET MAX
0'000E9A  4E4F             355.          TRAP #15
0'000E9C  0018             356.          DC.W DISBUF8
0'000E9E  4E4F             357.          TRAP #15
0'000EA0  000A             358.          DC.W SENCLFC
0'000EA2  41FA 017E        359.          LEA MINMES(PC),A0
0'000EA6  4E4F             360.          TRAP #15
0'000EA8  0011             361.          DC.W OUTMESC
0'000EAA  201F             362.          MOVE.L (SP)+,D0 ;GET MIN
0'000EAC  4E4F             363.          TRAP #15
0'000EAE  0018             364.          DC.W DISBUF8
0'000EB0  4E4F             365.          TRAP #15
0'000EB2  000A             366.          DC.W SENCLFC
                           367.  *
                           368.  * AFTER 256 RUNS, LOAD THE SECOND SLAVE
                           369.  * PROGRAM TO ACQUIRE THE TIMING DATA
                           370.  * IN THE SLAVES TIMER COUNT REGISTERS
                           371.  *
0'000EB4  41FA FD54        372.          LEA BRDCAST(PC),A0       ;BROADCAST MODE
0'000EB8  4E48             373.          TRAP #NETCNF
                           374.  *
                           375.  * RESPOND TO SLAVES RFP
                           376.  *
0'000EBA  103C 000F        377.          MOVE.B #ALLSLV,D0
0'000EBE  4E49             378.          TRAP #SRQSRVC
0'000EC0  41FA ****        379.          LEA SLVPRO2(PC),A0
0'000EC4  43FA ****        380.          LEA ENDSLV2(PC),A1
0'000EC8  93C8             381.          SUBA.L A0,A1
0'000ECA  3009             382.          MOVE.W A1,D0
0'000ECC  13FC 0020        383.          MOVE.B #ENABLEB,PGCR
          00012FC0
0'000ED4  4E47             384.          TRAP #ADBYTOT
```

```
0'000ED6  4E46             385.          TRAP #OUTDATA
0'000ED8  4239 00012FC0    386.          CLR.B PGCR       ;PROGRAM SENT
                           387.  *
                           388.  * RESPOND TO SLAVES RFD
                           389.  *
0'000EDE  103C 000F        390.          MOVE.B #ALLSLV,D0
0'000EE2  4E49             391.          TRAP #SRQSRVC
                           392.  *
                           393.  * PROGRAM HAS NO INPUT DATA, SEND DUMMY
                           394.  * ADDRESS AND ZERO BYTE COUNT
                           395.  * AND THEN NO DATA
                           396.  *
0'000EE4  4240             397.          CLR.W D0
0'000EE6  13FC 0020        398.          MOVE.B #ENABLEB,PGCR
          00012FC0
0'000EEE  4E47             399.          TRAP #ADBYTOT
0'000EF0  4239 00012FC0    400.          CLR.B PGCR
                           401.  *
                           402.  * RESPOND TO READY TO EXECUTE
                           403.  *
0'000EF6  103C 000F        404.          MOVE.B #ALLSLV,D0
0'000EFA  4E49             405.          TRAP #SRQSRVC
                           406.  *
                           407.  * SLAVES EXECUTE AND ASSERT AN SRQ WHEN
                           408.  * READY TO REPORT. RESPOND ONE AT A TIME
                           409.  * DETERMINE TIME ELAPSED AND REPORT TO
                           410.  * CONSOLE
                           411.  *
0'000EFC  103C 0001        412.          MOVE.B #SLV0,D0 ;SLAVE 0
0'000F00  4281             413.          CLR.L D1
0'000F02  61 32            414.          BSR.S GETSLTM
0'000F04  13FC 00BC        415.          MOVE.B #$BC,PSR1011 ;SLAVE 1
          00012FB5
0'000F0C  5281             416.          ADDQ.L #1,D1
0'000F0E  103C 0002        417.          MOVE.B #SLV1,D0
0'000F12  61 22            418.          BSR.S GETSLTM
0'000F14  13FC 00BB        419.          MOVE.B #$BB,PSR1415 ;SLAVE 2
          00012FB7
0'000F1C  5281             420.          ADDQ.L #1,D1
0'000F1E  103C 0004        421.          MOVE.B #SLV2,D0
0'000F22  61 12            422.          BSR.S GETSLTM
0'000F24  13FC 00BB        423.          MOVE.B #$BB,PSR1819 ;SLAVE3
          00012FB9
0'000F2C  5281             424.          ADDQ.L #1,D1
0'000F2E  103C 0008        425.          MOVE.B #SLV3,D0
0'000F32  61 02            426.          BSR.S GETSLTM
                           427.  *
                           428.  * NOW FINALLY DONE!
                           429.  *
0'000F34  4E75             430.          RTS
                           431.  *
                           432.  * SUBROUTINE TO GET TIMING DATA FROM SLAVE
                           433.  * SPECIFIED IN D1.B, WITH MASK IN D0.B
                           434.  *
0'000F36  48E7 C0C0        435.  GETSLTM MOVEM.L D0-D1/A0-A1,-(SP) ;SAVE ENV
0'000F3A  4E49             436.          TRAP #SRQSRVC
0'000F3C  7004             437.          MOVEQ.L #$4,D0  ;BYTE COUNT
0'000F3E  41FA 0140        438.          LEA TEMPCNT(PC),A0
0'000F42  13FC 0010        439.          MOVE.B #ENABLEA,PGCR    ;ENABLE INPUT
          00012FC0
0'000F4A  4E45             440.          TRAP #INDATA
0'000F4C  4239 00012FC0    441.          CLR.B PGCR
0'000F52  2248             442.          MOVEA.L A0,A1
0'000F54  41FA 00F3        443.          LEA MESSA(PC),A0
0'000F58  4E4F             444.          TRAP #15
0'000F5A  0011             445.          DC.W OUTMESC
0'000F5C  1001             446.          MOVE.B D1,D0
0'000F5E  4E4F             447.          TRAP #15
0'000F60  001A             448.          DC.W DISBUF2
0'000F62  41FA 00FF        449.          LEA MESSB(PC),A0
0'000F66  4E4F             450.          TRAP #15
0'000F68  0011             451.          DC.W OUTMESC
0'000F6A  2211             452.          MOVE.L (A1),D1
0'000F6C  203C 00FFFFFF    453.          MOVE.L #CNTSTRT,D0 ;CALC ELAPSED
0'000F72  9081             454.          SUB.L D1,D0
0'000F74  4E4F             455.          TRAP #15
0'000F76  0018             456.          DC.W DISBUF8
0'000F78  4E4F             457.          TRAP #15
0'000F7A  000A             458.          DC.W SENCLFC
0'000F7C  4CDF 0303        459.          MOVEM.L (SP)+,D0-D1/A0-A1
0'000F80  4E75             460.          RTS
```

```
                                    461.  *
                                    462.  * SUBROUTINE SENDOUT
                                    463.  *
0'000F82  93C8                      464.  SENDOUT SUBA.L A0,A1
0'000F84  3009                      465.          MOVE.W A1,D0
0'000F86  13FC 0020                 466.          MOVE.B #ENABLEB,PGCR
          00012FC0
0'000F8E  4E47                      467.          TRAP #ADBYTOT
0'000F90  4E46                      468.          TRAP #OUTDATA
0'000F92  4239 00012FC0             469.          CLR.B PGCR
0'000F98  4E75                      470.          RTS
                                    471.  *
                                    472.  * SUBROUTINE TSTSLV
                                    473.  * SLAVE NUMBER TO TEST IS IN D1.B
                                    474.  * IF ASSERTING AN SRQ, THEN ACKNOWLEDGED AND
                                    475.  * HIGH BIT IN D1.B IS SET, ELSE CLEAR
                                    476.  *
0'000F9A  48E7 8080                 477.  TSTSLV  MOVEM.L D0/A0,-(SP)
0'000F9E  207C 00012FCC             478.          MOVEA.L #PCDR,A0
0'000FA4  0810 0000                 479.          BTST.B #0,(A0)
0'000FA8  66 22                     480.          BNE.S NOINT
0'000FAA  E509                      481.          LSL.B #2,D1
0'000FAC  08C1 0004                 482.          BSET.L #4,D1
0'000FB0  1081                      483.          MOVE.B D1,(A0)
0'000FB2  0890 0004                 484.          BCLR.B #4,(A0)
0'000FB6  1010                      485.          MOVE.B (A0),D0
0'000FB8  08D0 0004                 486.          BSET.B #4,(A0)
0'000FBC  0800 0001                 487.          BTST.L #1,D0
0'000FC0  66 0A                     488.          BNE.S NOINT
0'000FC2  123C 00FF                 489.          MOVE.B #$FF,D1
0'000FC6  4CDF 0101                 490.          MOVEM.L (SP)+,D0/A0
0'000FCA  4E75                      491.          RTS
0'000FCC  4201                      492.  NOINT   CLR.B D1
0'000FCE  4CDF 0101                 493.          MOVEM.L (SP)+,D0/A0
0'000FD2  4E75                      494.          RTS
0'000FD4  41 76 65 72 61            495.  AVEMES  DC.B 'Average time ( x 4 for microseconds)= ',EOT
          67 65 20 74 69
          6D 65 20 28 20
          78 20 34 20 66
          6F 72 20 6D 69
          63 72 6F 73 65
          63 6F 6E 64 73
          29 3D 20 04
0'000FFB  4D 61 78 69 6D            496.  MAXMES  DC.B 'Maximum time ( x 4 for microseconds)= ',EOT
          75 6D 20 74 69
          6D 65 20 28 20
          78 20 34 20 66
          6F 72 20 6D 69
          63 72 6F 73 65
          63 6F 6E 64 73
          29 3D 20 04
0'001022  4D 69 6E 69 6D            497.  MINMES  DC.B 'Minimum time ( x 4 for microseconds)= ',EOT
          75 6D 20 74 69
          6D 65 20 28 20
          78 20 34 20 66
          6F 72 20 6D 69
          63 72 6F 73 65
          63 6F 6E 64 73
          29 3D 20 04
0'001049  45 78 65 63 75            498.  MESSA   DC.B 'Execution time by slave #',EOT
          74 69 6F 6E 20
          74 69 6D 65 20
          62 79 20 73 6C
          61 76 65 20 23
          04
0'001063  20 20 28 20 78            499.  MESSB   DC.B '   ( x 4 for microseconds) = ',EOT
          20 34 20 66 6F
          72 20 6D 69 63
          72 6F 73 65 63
          6F 6E 64 73 29
          20 3D 20 04
0'001080  <4>                       500.  TEMPCNT DS.B $4      ;TEMPORARY FOR SLAVE SELF TIME
                                    501.  *
                                    502.  *
1'000000                            503.          SECTION 1
                                    504.  *
                                    505.  ****************************
                                    506.  * ALL SLAVES PROG 2
                                    507.  ****************************
                                    508.  *
                                    509.  * PROGRAM TO REPORT CONTENTS OF THE
```

```
                                510.  * COUNTER REGISTER
                                511.  *
1'000000  207C 00012FD6         512.  SLVPRO2 MOVEA.L #TCR+6,A0        ;POINTER
1'000006  7004                  513.          MOVEQ.L #$4,D0           ;BYTE COUNT
1'000008  4E43                  514.          TRAP #SRQASRT            ;ASSERT SRQ
1'00000A  13FC 0020             515.          MOVE.B #ENABLEB,PGCR     ;OUTPUT ENABLED
          00012FC0
1'000012  4E46                  516.          TRAP #OUTDATA            ;SEND BLOCK
1'000014  4239 00012FC0         517.          CLR.B PGCR
1'00001A  4E4E                  518.          TRAP #ABORT      ;GOTO PROG ACCEPT MODE
1'00001C  <1>                   519.  ENDSLV2 DS.B 1
1'00001D                        520.          END
    0 Errors
*
* LINK SPEC FOR PFFT1
*
      LINK PFFT4
        LINK SLVFFT
      ORG $1000
      SECTION 0,15,1
      END
```

L.6 Program SLVFFT Multicomputer FFT test program: Slave Programs.

```
                                    1.          TTL SLVFFT
                                    2.  *
                                    3.  * THIS CODE IS A SET OF SLAVE PROGRAMS ONLY
                                    4.  * USED IN THE PARALLEL FFT PROGRAMS.
                                    5.  * ASSEMBLE SEPARATELY, BUT LINK TO THE MASTER
                                    6.  * PROGRAMS TO FORM THE CORRECT .HEX FILES FOR
                                    7.  * EXECUTION, BUT SAVING LOTSA DISK SPACE!
                                    8.  *
                                    9.  * EQUS, XREFS, AND XDEFS
                                   10.  *
F'000000                           11.          SECTION 15
                                   12.          XREF PGCR,ENABLEB
                                   13.          XREF ABORT NPT,NPTE,NPE
                                   14.          XREF SHIFT_4
                                   15.          XREF TCR,CNTSTRT,ENABLTM
                                   16.          XREF DATAII
                                   17.          XDEF S0PRO,S1PRO,S2PRO,S3PRO,DONE
                                   18.  *
       # 00000003                 19.  SRQASRT EQU 3
       # 00000006                 20.  OUTDATA EQU 6
       # 00000006                 21.  NUGAM   EQU 6
                                   22.  *
                                   23.  ********************
                                   24.  * SLAVE 0 PROGRAM *
                                   25.  ********************
                                   26.  *
                                   27.  * INPUT DATA AT DATAII
                                   28.  * FILTER DATA AT FILTER 64 POINTS
                                   29.  * NOTE THAT SLAVE 0 HAS NO TWIDDLE FACTOR
                                   30.  * MULTIPLIES
                                   31.  *
                                   32.  * add self timing code Aug 22,1991
                                   33.  *
F'000000  4239'00000000            34.  S0PRO   CLR.B TCR
F'000006  23FC'00000000            35.          MOVE.L #CNTSTRT,TCR+2
          '00000002
F'000010  13FC 00'00               36.          MOVE.B #ENABLTM,TCR
          '00000000
F'000018  2A7C'00000000            37.          MOVEA.L #NPE,A5          ;ADDR INC
F'00001E  7E'00                    38.          MOVEQ.L #SHIFT14,D7      ;/16384 SHFT
F'000020  303C'FFFF                39.          MOVE.W #NPTE-1,D0        ;MAJ LOOP CNTR
F'000024  2C3C'00000000            40.          MOVE.L #NPT,D6           ;INCREMENT
F'00002A  287C'FFFFFFFC            41.          MOVEA.L #NPT-4,A4        ;INDX TO ARRAYS
F'000030  41FA ****                42.          LEA DATAII(PC),A0        ;XDATA POINTER
F'000034  4281                     43.  TWDL0   CLR.L D1                 ;INIT SUMS
F'000036  2401                     44.          MOVE.L D1,D2
F'000038  2648                     45.          MOVEA.L A0,A3
F'00003A  7A'FF                    46.          MOVEQ.L #NPE-1,D5
F'00003C  D7CC                     47.          ADDA.L A4,A3
F'00003E  3613                     48.  ADDL0   MOVE.W (A3),D3           ;GET REAL PART
F'000040  48C3                     49.          EXT.L D3
F'000042  D283                     50.          ADD.L D3,D1              ;REAL SUM D1
F'000044  362B 0002                51.          MOVE.W 2(A3),D3          ;GET IMAG PART
F'000048  48C3                     52.          EXT.L D3
F'00004A  D483                     53.          ADD.L D3,D2              ;IMAG SUM D2
F'00004C  D7C6                     54.          ADDA.L D6,A3             ;NEXT TERM
F'00004E  51CD FFEE                55.          DBRA D5,ADDL0
F'000052  E481                     56.          ASR.L #2,D1              ;ADJUST SUMS
F'000054  E482                     57.          ASR.L #2,D2              ;ADJUST SUMS
                                   58.  *
                                   59.  * NO TWIDDLE FACTORS FOR SLAVE 0
                                   60.  * STORE SUMMED AND TWIDDLED DATA AT DATAII IN
                                   61.  * PLACE
                                   62.  *
F'000056  3181 C800                63.          MOVE.W D1,0(A0,A4.L)
F'00005A  3182 C802                64.          MOVE.W D2,2(A0,A4.L)
F'00005E  99CD                     65.          SUBA.L A5,A4
F'000060  51C8 FFD2                66.          DBRA D0,TWDL0
                                   67.  *
                                   68.  * 64-POINT FFT
                                   69.  * INPUT DATA AT DATAII (0-255)
                                   70.  *
F'000064  78'F7                    71.  USHFL0  MOVEQ.L #NPTE-9,D4        ;COUNTER=#
F'000066  3C04                     72.  SWAPL0  MOVE.W D4,D6          ;PREPARE BIT REV
F'000068  3E06                     73.  BREV20  MOVE.W D6,D7        ;SAVE DATA
F'00006A  4246                     74.          CLR.W D6            ;
F'00006C  7A05                     75.          MOVEQ.L #5,D5       ;BIT COUNTER
F'00006E  E257                     76.  REV20   ROXR.W #1,D7        ;SFT RIGHT,XTEND
```

```
F'000070  E356              77.          ROXL.W #1,D6
F'000072  51CD FFFA         78.          DBRA D5,REV20
F'000076  B846              79.          CMP.W D6,D4          ;I=REV(K)<=K?
F'000078  6C 14             80.          BGE.S DECK0           ;NO SWAP IF SO
F'00007A  3204              81.          MOVE.W D4,D1         ;COPY OF K
F'00007C  E541              82.          ASL.W #2,D1          ;K*4
F'00007E  E546              83.          ASL.W #2,D6          ;I*4
F'000080  2430 1000         84.          MOVE.L 0(A0,D1.W),D2 ;DO SWAP
F'000084  21B0 6000 1000    85.          MOVE.L 0(A0,D6.W),0(A0,D1.W)
F'00008A  2182 6000         86.          MOVE.L D2,0(A0,D6.W) ;DONE SWAP
F'00008E  51CC FFD6         87.  DECK0   DBRA D4,SWAPL0
F'000092  7201              88.  FFT0    MOVEQ.L #1,D1   ;D1=N2=1
F'000094  43FA 00B6         89.          LEA SINBLK0(PC),A1  ;SIN VALS PNTR A1
F'000098  45FA 0132         90.          LEA COSBLK0(PC),A2  ;COS VALS PNTR A2
                            91.  *
                            92.  * L IS DOWN COUNTED #NUGAM-1 TO 0
                            93.  *
F'00009C  3F3C 0001         94.          MOVE.W #1,-(SP)  ;(SP) IS L
F'0000A0  7405              95.          MOVEQ.L #NUGAM-1,D2       ;D2=NUGAM-1=NU1
F'0000A2  4283              96.          CLR.L D3             ;D3=K=0
F'0000A4  700E              97.          MOVEQ.L #14,D0
                            98.  *
                            99.  * INITIALIZE I=N2-1, TO 0 IN STEPS -1
                           100.  *
F'0000A6  3801             101.  INITI0  MOVE.W D1,D4    ;D4=I=N2
F'0000A8  5344             102.          SUBQ.W #1,D4    ;SET FOR DBRA
                           103.  *
                           104.  * DETERMINE P=MOD(K*2**(NUGAM-L),64)
                           105.  *
                           106.  *
F'0000AA  3C03             107.  LOOPIN0 MOVE.W D3,D6    ;D6=K
F'0000AC  3A17             108.          MOVE.W (SP),D5  ;GET L
F'0000AE  5D45             109.          SUBQ.W #NUGAM,D5 ;L-NUGAM
F'0000B0  4445             110.          NEG.W D5        ;NUGAM-L
F'0000B2  EB66             111.          ASL.W D5,D6     ;K*2**(NUGAM-L)
F'0000B4  0246 003F        112.          ANDI.W #$003F,D6 ;MOD 64
                           113.  *
                           114.  * D6 HAS INDEX REQUIRED
F'0000B8  DC46             115.          ADD.W D6,D6     ;D6*2 FOR WORD BOUND
F'0000BA  3A71 6000        116.          MOVEA.W 0(A1,D6.W),A5 ;A5 PNTR IM(W**P)
F'0000BE  2E03             117.          MOVE.L D3,D7    ;D7=K
F'0000C0  2A03             118.          MOVE.L D3,D5    ;D5 TOO
F'0000C2  E545             119.          ASL.W #2,D5     ;K*4 FOR LWORD BOUND
F'0000C4  4DF0 5000        120.          LEA 0(A0,D5.W),A6   ;A6 IS INDEX X(K)
F'0000C8  DE41             121.          ADD.W D1,D7     ;D7=K+N2
F'0000CA  E547             122.          ASL.W #2,D7     ;D7=4(K+N2) LWORD BND
F'0000CC  47F0 7000        123.          LEA 0(A0,D7.W),A3 ;A3 PNT RE(X(K+N2))
F'0000D0  3A32 6000        124.          MOVE.W 0(A2,D6.W),D5    ;GET RE(W**P)
F'0000D4  3E05             125.          MOVE.W D5,D7    ;PUT ALSO IN D7
F'0000D6  CBD3             126.          MULS (A3),D5    ;RE(W**P)*RE(X(K+N2))
F'0000D8  3C0D             127.          MOVE.W A5,D6  ;GET IM(W**P)*
F'0000DA  CDEB 0002        128.          MULS 2(A3),D6   ;IM(W**P)*IM(X(K+N2))
F'0000DE  9A86             129.          SUB.L D6,D5     ;D5=RE(T1)*
F'0000E0  E0A5             130.          ASR.L D0,D5     ;/16384 TO SCALE
F'0000E2  3C0D             131.          MOVE.W A5,D6  ;GET IM(W**P)
F'0000E4  CDD3             132.          MULS (A3),D6    ;IM(W**P)*RE(X(K+N2))
F'0000E6  CFEB 0002        133.          MULS 2(A3),D7   ;RE(W**P)*IM(X(K+N2))
F'0000EA  DC87             134.          ADD.L D7,D6     ;D6=IM(T1)
F'0000EC  E0A6             135.          ASR.L D0,D6     ;/16384 TO SCALE
F'0000EE  3E16             136.          MOVE.W (A6),D7  ;RE(X(K))
F'0000F0  9E45             137.          SUB.W D5,D7     ;RE(X(K))-RE(T1)
F'0000F2  E247             138.          ASR.W #1,D7     ;/2
F'0000F4  3687             139.          MOVE.W D7,(A3)  ;STORE ANSWER
F'0000F6  3E2E 0002        140.          MOVE.W 2(A6),D7 ;IM(X(K))
F'0000FA  9E46             141.          SUB.W D6,D7     ;IM(X(K))-IM(T1)
F'0000FC  E247             142.          ASR.W #1,D7     ;/2
F'0000FE  3747 0002        143.          MOVE.W D7,2(A3) ;STORE ANSWER
F'000102  DB56             144.          ADD.W D5,(A6)   ;RE(X(K))+RE(T1)
F'000104  E0D6             145.          ASR.W (A6)      ;/2
F'000106  DD6E 0002        146.          ADD.W D6,2(A6)  ;IM(X(K))+IM(T1)
F'00010A  E0EE 0002        147.          ASR.W 2(A6)     ;/2
F'00010E  5243             148.          ADDQ.W #1,D3    ;INCREMENT K
F'000110  51CC FF98        149.          DBRA D4,LOOPIN0  ;DO TILL I=-1
F'000114  D641             150.          ADD.W D1,D3     ;K=K+N2
F'000116  0C43'FFFF        151.          CMPI.W #NPTE-1,D3 ;K<N-1 ?
F'00011A  6D 8A            152.          BLT INITI0      ;DO AGAIN IF SO
F'00011C  4243             153.          CLR.W D3        ;K=0
F'00011E  5342             154.          SUBQ.W #1,D2    ;NU1=NU1-1
F'000120  E341             155.          ASL.W #1,D1     ;N2=N2*2
F'000122  5257             156.          ADDQ.W #1,(SP)
F'000124  0C57 0007        157.          CMPI.W #NUGAM+1,(SP)
```

```
F'000128  6600 FF7C        158.          BNE INITI0
F'00012C  301F             159.          MOVE.W (SP)+,D0 ;CLEAN STACK
                           160.  *
                           161.  * STOP TIMER
                           162.  *
F'00012E  4239'00000000    163.          CLR.B TCR
                           164.  *
                           165.  * ASSERT AN SRQ AND UPLOAD THE DATA
                           166.  *
F'000134  4E43             167.          TRAP #SRQASRT
>>> WARNING line[168] Expr. type   [168] SLVFFT.A68
F'000136  303C ****        168.          MOVE.W #NPTE*4,D0       ;BYTE COUNT
F'00013A  13FC 00'00       169.          MOVE.B #ENABLEB,PGCR
          '00000000
F'000142  4E46             170.          TRAP #OUTDATA
F'000144  4239'00000000    171.          CLR.B PGCR
                           172.  *
                           173.  * RETURN TO READY FOR DATA
                           174.  * STATE TO DO OVER!
                           175.  *
F'00014A  4E75             176.          RTS
                           177.  *
                           178.  * DONE!
                           179.  *
                           180.  * SIN AND COS TABLES HERE
                           181.  *
                           182.  *
                           183.  * SIN (-2*PI*I/64)*16384 FOR I=0 TO 63
                           184.  *
F'00014C  0000 F9BA F384   185.  SINBLK0 DC.W       0, -1606, -3196, -4756
          ED6C
F'000154  E782 E1D5 DC72   186.          DC.W   -6270, -7723, -9102,-10394
          D766
F'00015C  D2BF CE87 CAC9   187.          DC.W  -11585,-12665,-13623,-14449
          C78F
F'000164  C4DF C2C1 C13B   188.          DC.W  -15137,-15679,-16069,-16305
          C04F
F'00016C  C000 C04F C13B   189.          DC.W  -16384,-16305,-16069,-15679
          C2C1
F'000174  C4DF C78F CAC9   190.          DC.W  -15137,-14449,-13623,-12665
          CE87
F'00017C  D2BF D766 DC72   191.          DC.W  -11585,-10394, -9102, -7723
          E1D5
F'000184  E782 ED6C F384   192.          DC.W   -6270, -4756, -3196, -1606
          F9BA
F'00018C  0000 0646 0C7C   193.          DC.W       0,  1606,  3196,  4756
          1294
F'000194  187E 1E2B 238E   194.          DC.W    6270,  7723,  9102, 10394
          289A
F'00019C  2D41 3179 3537   195.          DC.W   11585, 12665, 13623, 14449
          3871
F'0001A4  3B21 3D3F 3EC5   196.          DC.W   15137, 15679, 16069, 16305
          3FB1
F'0001AC  4000 3FB1 3EC5   197.          DC.W   16384, 16305, 16069, 15679
          3D3F
F'0001B4  3B21 3871 3537   198.          DC.W   15137, 14449, 13623, 12665
          3179
F'0001BC  2D41 289A 238E   199.          DC.W   11585, 10394,  9102,  7723
          1E2B
F'0001C4  187E 1294 0C7C   200.          DC.W    6270,  4756,  3196,  1606
          0646
                           201.  *
                           202.  * COS(-2*PI*I/64)*16384 FOR I=0 TO 63
                           203.  *
F'0001CC  4000 3FB1 3EC5   204.  COSBLK0 DC.W   16384, 16305, 16069, 15679
          3D3F
F'0001D4  3B21 3871 3537   205.          DC.W   15137, 14449, 13623, 12665
          3179
F'0001DC  2D41 289A 238E   206.          DC.W   11585, 10394,  9102,  7723
          1E2B
F'0001E4  187E 1294 0C7C   207.          DC.W    6270,  4756,  3196,  1606
          0646
F'0001EC  0000 F9BA F384   208.          DC.W       0, -1606, -3196, -4756
          ED6C
F'0001F4  E782 E1D5 DC72   209.          DC.W   -6270, -7723, -9102,-10394
          D766
F'0001FC  D2BF CE87 CAC9   210.          DC.W  -11585,-12665,-13623,-14449
          C78F
F'000204  C4DF C2C1 C13B   211.          DC.W  -15137,-15679,-16069,-16305
          C04F
F'00020C  C000 C04F C13B   212.          DC.W  -16384,-16305,-16069,-15679
```

```
                C2C1
F'000214        C4DF C78F CAC9          213.            DC.W  -15137,-14449,-13623,-12665
                CE87
F'00021C        D2BF D766 DC72          214.            DC.W  -11585,-10394, -9102, -7723
                E1D5
F'000224        E782 ED6C F384          215.            DC.W   -6270, -4756, -3196, -1606
                F9BA
F'00022C        0000 0646 0C7C          216.            DC.W       0,  1606,  3196,  4756
                1294
F'000234        187E 1E2B 238E          217.            DC.W    6270,  7723,  9102, 10394
                289A
F'00023C        2D41 3179 3537          218.            DC.W   11585, 12665, 13623, 14449
                3871
F'000244        3B21 3D3F 3EC5          219.            DC.W   15137, 15679, 16069, 16305
                3FB1
                                        220.
                                        221.    ********************
                                        222.    * SLAVE 1 PROGRAM *
                                        223.    ********************
                                        224.    *
                                        225.    * INPUT DATA AT DATAII, TWIDDLE FACTORS AT
                                        226.    * TWD1
                                        227.    *
                                        228.    * SELF TIME CODE AUG 22,1991
                                        229.    *
F'00024C        4239'00000000           230.    S1PRO   CLR.B TCR
F'000252        23FC'00000000           231.            MOVE.L #CNTSTRT,TCR+2
                '00000002
F'00025C        13FC 00'00              232.            MOVE.B #ENABLTM,TCR
                '00000000
F'000264        2A7C'00000000           233.            MOVEA.L #NPE,A5         ;ADDR INC
F'00026A        7E'00                   234.            MOVEQ.L #SHIF-14,D7     ;/16384 SHFT
F'00026C        303C'FFFF               235.            MOVE.W #NPTE-1,D0       ;MAJ LOOP CNTR
F'000270        287C'FFFFFFFC           236.            MOVEA.L #NPT-4,A4       ;INDX TO ARRAYS
F'000276        43FA 025A               237.            LEA TWD1(PC),A1         ;TWID FACT PNTR
F'00027A        41FA ****               238.            LEA DATAII(PC),A0       ;XDATA POINTER
F'00027E        2648                    239.    TWDL1   MOVEA.L A0,A3
F'000280        D7CC                    240.            ADDA.L A4,A3
F'000282        3213                    241.    ADDL1   MOVE.W (A3),D1          ;GET REAL PART
F'000284        48C1                    242.            EXT.L D1                ;REAL SUM IN D1
F'000286        362B 0102               243.            MOVE.W 258(A3),D3
F'00028A        48C3                    244.            EXT.L D3
F'00028C        D283                    245.            ADD.L D3,D1
F'00028E        362B 0200               246.            MOVE.W 512(A3),D3
F'000292        48C3                    247.            EXT.L D3
F'000294        9283                    248.            SUB.L D3,D1
F'000296        362B 0302               249.            MOVE.W 770(A3),D3
F'00029A        48C3                    250.            EXT.L D3
F'00029C        9283                    251.            SUB.L D3,D1             ;REAL SUM DONE
F'00029E        342B 0002               252.            MOVE.W 2(A3),D2         ;IMAG SUM IN D2
F'0002A2        48C2                    253.            EXT.L D2
F'0002A4        362B 0100               254.            MOVE.W 256(A3),D3
F'0002A8        48C3                    255.            EXT.L D3
F'0002AA        9483                    256.            SUB.L D3,D2
F'0002AC        362B 0202               257.            MOVE.W 514(A3),D3
F'0002B0        48C3                    258.            EXT.L D3
F'0002B2        9483                    259.            SUB.L D3,D2
F'0002B4        362B 0300               260.            MOVE.W 768(A3),D3
F'0002B8        48C3                    261.            EXT.L D3
F'0002BA        D483                    262.            ADD.L D3,D2             ;IMAG SUM DONE
F'0002BC        E481                    263.            ASR.L #2,D1
F'0002BE        E482                    264.            ASR.L #2,D2             ;ADJUST SUMS
                                        265.    *
                                        266.    * NOW DO TWIDDLE FACTOR MULTIPLY
                                        267.    *
F'0002C0        3A01                    268.            MOVE.W D1,D5
F'0002C2        3831 C800               269.            MOVE.W 0(A1,A4.L),D4
F'0002C6        3631 C802               270.            MOVE.W 2(A1,A4.L),D3
F'0002CA        CBC4                    271.            MULS D4,D5
F'0002CC        3C02                    272.            MOVE.W D2,D6
F'0002CE        CDC3                    273.            MULS D3,D6
F'0002D0        9A86                    274.            SUB.L D6,D5             ;REAL IN D5
F'0002D2        C3C3                    275.            MULS D3,D1
F'0002D4        C5C4                    276.            MULS D4,D2
F'0002D6        D282                    277.            ADD.L D2,D1             ;IMAG IN D1
F'0002D8        EEA5                    278.            ASR.L D7,D5
F'0002DA        EEA1                    279.            ASR.L D7,D1
                                        280.    *
                                        281.    * STORE SUMMED AND TWIDDLED DATA AT DATAII IN
                                        282.    * PLACE
                                        283.    *
```

```
F'0002DC  3185 C800        284.          MOVE.W D5,0(A0,A4.L)
F'0002E0  3181 C802        285.          MOVE.W D1,2(A0,A4.L)
F'0002E4  99CD             286.          SUBA.L A5,A4
F'0002E6  51C8 FF96        287.          DBRA D0,TWDL1
                           288.  *
                           289.  * 64-POINT FFT
                           290.  * INPUT DATA AT DATAII (0-255)
                           291.  *
F'0002EA  78'F7            292.  USHFL1  MOVEQ.L #NPTE-9,D4        ;COUNTER=K
F'0002EC  3C04             293.  SWAPL1  MOVE.W D4,D6          ;PREPARE BIT REV
F'0002EE  3E06             294.  BREV21  MOVE.W D6,D7        ;SAVE DATA
F'0002F0  4246             295.          CLR.W D6            ;
F'0002F2  7A05             296.          MOVEQ.L #5,D5       ;BIT COUNTER
F'0002F4  E257             297.  REV21   ROXR.W #1 D7        ;SFT RIGHT,XTEND
F'0002F6  E356             298.          ROXL.W #1,D6
F'0002F8  51CD FFFA        299.          DBRA D5,REV21
F'0002FC  B846             300.          CMP.W D6,D4           ;I=REV(K)<=K?
F'0002FE  6C 14            301.          BGE.S DECK1           ;NO SWAP IF SO
F'000300  3204             302.          MOVE.W D4,D1          ;COPY OF K
F'000302  E541             303.          ASL.W #2,D1           ;K*4
F'000304  E546             304.          ASL.W #2,D6           ;I*4
F'000306  2430 1000        305.          MOVE.L 0(A0,D1.W),D2 ;DO SWAP
F'00030A  21B0 6000 1000   306.          MOVE.L 0(A0.D6.W),0(A0,D1.W)
F'000310  2182 6000        307.          MOVE.L D2,0(A0,D6.W) ;DONE SWAP
F'000314  51CC FFD6        308.  DECK1   DBRA D4,SWAPL1
F'000318  7201             309.  FFT641  MOVEQ.L #1,D1    ;D1=N2=1
F'00031A  43FA 00B6        310.          LEA SINBLK1(PC),A1  ;SIN VALS PNTR A1
F'00031E  45FA 0132        311.          LEA COSBLK1(PC),A2  ;COS VALS PNTR A2
                           312.  *
                           313.  * L IS DOWN COUNTED #NUGAM-1 TO 0
                           314.  *
F'000322  3F3C 0001        315.          MOVE.W #1,-(SP)  ;(SP) IS L
F'000326  7405             316.          MOVEQ.L #NUGAM-1,D2         ;D2=NUGAM-1=NU1
F'000328  4283             317.          CLR.L D3              ;D3=K=0
F'00032A  700E             318.          MOVEQ.L #14,D0
                           319.  *
                           320.  * INITIALIZE I=N2-1, TO 0 IN STEPS -1
                           321.  *
F'00032C  3801             322.  INITI1  MOVE.W D1,D4     ;D4=I=N2
F'00032E  5344             323.          SUBQ.W #1,D4     ;SET FOR DBRA
                           324.  *
                           325.  * DETERMINE P=MOD(K*2**(NUGAM-L),64)
                           326.  *
                           327.  *
F'000330  3C03             328.  LOOPIN1 MOVE.W D3,D6     ;D6=K
F'000332  3A17             329.          MOVE.W (SP),D5   ;GET L
F'000334  5D45             330.          SUBQ.W #NUGAM,D5 ;L-NUGAM
F'000336  4445             331.          NEG.W D5         ;NUGAM-L
F'000338  EB66             332.          ASL.W D5,D6      ;K*2**(NUGAM-L)
F'00033A  0246 003F        333.          ANDI.W #$003F,D6 ;MOD 64
                           334.  *
                           335.  * D6 HAS INDEX REQUIRED
                           336.
F'00033E  DC46             337.          ADD.W D6,D6      ;D6*2 FOR WORD BOUND
F'000340  3A71 6000        338.          MOVEA.W 0(A1,D6.W),A5 ;A5 PNTR IM(W**P)
F'000344  2E03             339.          MOVE.L D3,D7     ;D7=K
F'000346  2A03             340.          MOVE.L D3,D5     ;D5 TOO
F'000348  E545             341.          ASL.W #2,D5      ;K*4 FOR LWORD BOUND
F'00034A  4DF0 5000        342.          LEA 0(A0,D5.W),A6   ;A6 IS INDEX X(K)
F'00034E  DE41             343.          ADD.W D1,D7      ;D7=K+N2
F'000350  E547             344.          ASL.W #2,D7      ;D7=4(K+N2) LWORD BND
F'000352  47F0 7000        345.          LEA 0(A0,D7.W),A3 ;A3 PNT RE(X(K+N2))
F'000356  3A32 6000        346.          MOVE.W 0(A2,D6.W),D5    ;GET RE(W**P)
F'00035A  3E05             347.          MOVE.W D5,D7     ;PUT ALSO IN D7
F'00035C  CBD3             348.          MULS (A3),D5     ;RE(W**P)*RE(X(K+N2))
F'00035E  3C0D             349.          MOVE.W A5,D6  ;GET IM(W**P)
F'000360  CDEB 0002        350.          MULS 2(A3),D6    ;IM(W**P)*IM(X(K+N2))
F'000364  9A86             351.          SUB.L D6,D5      ;D5=RE(T1)
F'000366  E0A5             352.          ASR.L D0,D5      ;/16384 TO SCALE
F'000368  3C0D             353.          MOVE.W A5,D6  ;GET IM(W**P)
F'00036A  CDD3             354.          MULS (A3),D6     ;IM(W**P)*RE(X(K+N2))
F'00036C  CFEB 0002        355.          MULS 2(A3),D7    ;RE(W**P)*IM(X(K+N2))
F'000370  DC87             356.          ADD.L D7,D6      ;D6=IM(T1)
F'000372  E0A6             357.          ASR.L D0,D6      ;/16384 TO SCALE
F'000374  3E16             358.          MOVE.W (A6),D7   ;RE(X(K))
F'000376  9E45             359.          SUB.W D5,D7      ;RE(X(K))-RE(T1)
F'000378  E247             360.          ASR.W #1,D7      ;/2
F'00037A  3687             361.          MOVE.W D7,(A3)   ;STORE ANSWER
F'00037C  3E2E 0002        362.          MOVE.W 2(A6),D7 ;IM(X(K))
F'000380  9E46             363.          SUB.W D6,D7      ;IM(X(K))-IM(T1)
F'000382  E247             364.          ASR.W #1,D7      ;/2
```

```
F'000384  3747 0002        365.          MOVE.W D7,2(A3) ;STORE ANSWER
F'000388  DB56             366.          ADD.W D5,(A6)   ;RE(X(K))+RE(T1)
F'00038A  E0D6             367.          ASR.W (A6)      ;/2
F'00038C  DD6E 0002        368.          ADD.W D6,2(A6)  ;IM(X(K))+IM(T1)
F'000390  E0EE 0002        369.          ASR.W 2(A6)     ;/2
F'000394  5243             370.          ADDQ.W #1,D3    ;INCREMENT K
F'000396  51CC FF98        371.          DBRA D4,LOOPIN1  ;DO TILL I=-1
F'00039A  D641             372.          ADD.W D1,D3     ;K=K+N2
F'00039C  0C43'FFFF        373.          CMPI.W #NPTE-1,D3 ;K<N-1 ?
F'0003A0  6D 8A            374.          BLT INITI1      ;DO AGAIN IF SO
F'0003A2  4243             375.          CLR.W D3        ;K=0
F'0003A4  5342             376.          SUBQ.W #1,D2    ;NU1=NU1-1
F'0003A6  E341             377.          ASL.W #1,D1     ;N2=N2*2
F'0003A8  5257             378.          ADDQ.W #1,(SP)
F'0003AA  0C57 0007        379.          CMPI.W #NUGAM+1,(SP)
F'0003AE  6600 FF7C        380.          BNE INITI1
F'0003B2  301F             381.          MOVE.W (SP)+,D0
                           382.  *
                           383.  * STOP TIMER
                           384.  *
F'0003B4  4239'00000000    385.          CLR.B TCR
                           386.  *
                           387.  * ASSERT SRQ AND WAIT FOR ACKNOWLEDGE
                           388.  *
F'0003BA  4E43             389.          TRAP #SRQASRT
                           390.  * SEND OUT 64 COMPLEX POINTS CALCULATED
>>> WARNING line[391] Expr. type   [391] SLVFFT.A68
F'0003BC  303C ****        391.          MOVE.W #NPTE*4,D0       ;BYTE COUNT
F'0003C0  13FC 00'00       392.          MOVE.B #ENABLEB,PGCR
          '00000000
F'0003C8  4E46             393.          TRAP #OUTDATA
F'0003CA  4239'00000000    394.          CLR.B PGCR
                           395.  *
                           396.  * DONE!
                           397.  *
F'0003D0  4E75             398.          RTS
                           399.  *
                           400.  * SIN (-2*PI*I/64)*16384 FOR I=0 TO 63
                           401.  *
F'0003D2  0000 F9BA F384   402.  SINBLK1 DC.W       0, -1606, -3196, -4756
          ED6C
F'0003DA  E782 E1D5 DC72   403.          DC.W   -6270, -7723, -9102,-10394
          D766
F'0003E2  D2BF CE87 CAC9   404.          DC.W  -11585,-12665,-13623,-14449
          C78F
F'0003EA  C4DF C2C1 C13B   405.          DC.W  -15137,-15679,-16069,-16305
          C04F
F'0003F2  C000 C04F C13B   406.          DC.W  -16384,-16305,-16069,-15679
          C2C1
F'0003FA  C4DF C78F CAC9   407.          DC.W  -15137,-14449,-13623,-12665
          CE87
F'000402  D2BF D766 DC72   408.          DC.W  -11585,-10394, -9102, -7723
          E1D5
F'00040A  E782 ED6C F384   409.          DC.W   -6270, -4756, -3196, -1606
          F9BA
F'000412  0000 0646 0C7C   410.          DC.W       0,  1606,  3196,  4756
          1294
F'00041A  187E 1E2B 238E   411.          DC.W    6270,  7723,  9102, 10394
          289A
F'000422  2D41 3179 3537   412.          DC.W   11585, 12665, 13623, 14449
          3871
F'00042A  3B21 3D3F 3EC5   413.          DC.W   15137, 15679, 16069, 16305
          3FB1
F'000432  4000 3FB1 3EC5   414.          DC.W   16384, 16305, 16069, 15679
          3D3F
F'00043A  3B21 3871 3537   415.          DC.W   15137, 14449, 13623, 12665
          3179
F'000442  2D41 289A 238E   416.          DC.W   11585, 10394,  9102,  7723
          1E2B
F'00044A  187E 1294 0C7C   417.          DC.W    6270,  4756,  3196,  1606
          0646
                           418.  *
                           419.  * COS(-2*PI*I/64)*16384 FOR I=0 TO 63
                           420.  *
F'000452  4000 3FB1 3EC5   421.  COSBLK1 DC.W   16384, 16305, 16069, 15679
          3D3F
F'00045A  3B21 3871 3537   422.          DC.W   15137, 14449, 13623, 12665
          3179
F'000462  2D41 289A 238E   423.          DC.W   11585, 10394,  9102,  7723
          1E2B
F'00046A  187E 1294 0C7C   424.          DC.W    6270,  4756,  3196,  1606
```

```
                0646
F'000472        0000 F9BA F384   425.            DC.W       0, -1606, -3196, -4756
                ED6C
F'00047A        E782 E1D5 DC72   426.            DC.W   -6270, -7723, -9102,-10394
                D766
F'000482        D2BF CE87 CAC9   427.            DC.W  -11585,-12665,-13623,-14449
                C78F
F'00048A        C4DF C2C1 C13B   428.            DC.W  -15137,-15679,-16069,-16305
                C04F
F'000492        C000 C04F C13B   429.            DC.W  -16384,-16305,-16069,-15679
                C2C1
F'00049A        C4DF C78F CAC9   430.            DC.W  -15137,-14449,-13623,-12665
                CE87
F'0004A2        D2BF D766 DC72   431.            DC.W  -11585,-10394, -9102, -7723
                E1D5
F'0004AA        E782 ED6C F384   432.            DC.W   -6270, -4756, -3196, -1606
                F9BA
F'0004B2        0000 0646 0C7C   433.            DC.W       0,  1606,  3196,  4756
                1294
F'0004BA        187E 1E2B 238E   434.            DC.W    6270,  7723,  9102, 10394
                289A
F'0004C2        2D41 3179 3537   435.            DC.W   11585, 12665, 13623, 14449
                3871
F'0004CA        3B21 3D3F 3EC5   436.            DC.W   15137, 15679, 16069, 16305
                3FB1
                                 437.  *
                                 438.  * TWIDDLE FACTOR TABLE 1 TWD1
                                 439.  *
F'0004D2        4000 0000 3FFB   440.  TWD1      DC.W   16384,     0, 16379,  -402
                FE6E
F'0004DA        3FEC FCDC 3FD4   441.            DC.W   16364,  -804, 16340, -1205
                FB4B
F'0004E2        3FB1 F9BA 3F85   442.            DC.W   16305, -1606, 16261, -2006
                F82A
F'0004EA        3F4F F69C 3F0F   443.            DC.W   16207, -2404, 16143, -2801
                F50F
F'0004F2        3EC5 F384 3E72   444.            DC.W   16069, -3196, 15986, -3590
                F1FA
F'0004FA        3E15 F073 3DAF   445.            DC.W   15893, -3981, 15791, -4370
                EEEE
F'000502        3D3F ED6C 3CC5   446.            DC.W   15679, -4756, 15557, -5139
                EBED
F'00050A        3C42 EA70 3BB6   447.            DC.W   15426, -5520, 15286, -5897
                E8F7
F'000512        3B21 E782 3A82   448.            DC.W   15137, -6270, 14978, -6639
                E611
F'00051A        39DB E4A3 392B   449.            DC.W   14811, -7005, 14635, -7366
                E33A
F'000522        3871 E1D5 37B0   450.            DC.W   14449, -7723, 14256, -8076
                E074
F'00052A        36E5 DF19 3612   451.            DC.W   14053, -8423, 13842, -8765
                DDC3
F'000532        3537 DC72 3453   452.            DC.W   13623, -9102, 13395, -9434
                DB26
F'00053A        3368 D9E0 3274   453.            DC.W   13160, -9760, 12916,-10080
                D8A0
F'000542        3179 D766 3076   454.            DC.W   12665,-10394, 12406,-10702
                D632
F'00054A        2F6C D505 2E5A   455.            DC.W   12140,-11003, 11866,-11297
                D3DF
F'000552        2D41 D2BF 2C21   456.            DC.W   11585,-11585, 11297,-11866
                D1A6
F'00055A        2AFB D094 29CE   457.            DC.W   11003,-12140, 10702,-12406
                CF8A
F'000562        289A CE87 2760   458.            DC.W   10294,-12665, 10080,-12916
                CD8C
F'00056A        2620 CC98 24DA   459.            DC.W    9760,-13160,  9434,-13395
                CBAD
F'000572        238E CAC9 223D   460.            DC.W    9102,-13623,  8765,-13842
                C9EE
F'00057A        20E7 C91B 1F8C   461.            DC.W    8423,-14053,  8076,-14256
                C850
F'000582        1E2B C78F 1CC6   462.            DC.W    7723,-14449,  7366,-14635
                C6D5
F'00058A        1B5D C625 19EF   463.            DC.W    7005,-14811,  6639,-14978
                C57E
F'000592        187E C4DF 1709   464.            DC.W    6270,-15137,  5897,-15286
```

```
                C44A
F'00059A        1590 C3BE 1413  465.          DC.W    5520,-15426,  5139,-15557
                C33B
F'0005A2        1294 C2C1 1112  466.          DC.W    4756,-15679,  4370,-15791
                C251
F'0005AA        0F8D C1EB 0E06  467.          DC.W    3981,-15893,  3590,-15986
                C18E
F'0005B2        0C7C C13B 0AF1  468.          DC.W    3196,-16069,  2801,-16143
                C0F1
F'0005BA        0964 C0B1 07D6  469.          DC.W    2404,-16207,  2006,-16261
                C07B
F'0005C2        0646 C04F 04B5  470.          DC.W    1606,-16305,  1205,-16340
                C02C
F'0005CA        0324 C014 0192  471.          DC.W     804,-16364,   402,-16379
                C005
                                472.
                                473.  ********************
                                474.  * SLAVE 2 PROGRAM *
                                475.  ********************
                                476.  *
                                477.  * INPUT DATA AT DATAII, TWIDDLE FACTORS AT
                                478.  * TWD2
                                479.  *
                                480.  * SELF TIME CODE AUG 22,1991
                                481.  *
F'0005D2        4239'00000000   482.  S2PRO   CLR.B TCR
F'0005D8        23FC'00000000   483.          MOVE.L #CNTSTRT,TCR+2
                '00000002
F'0005E2        13FC 00'00      484.          MOVE.B #ENABLTM,TCR
                '00000000
F'0005EA        2A7C'00000000   485.          MOVEA.L #NPE,A5          ;ADDR INC
F'0005F0        7E'00           486.          MOVEQ.L #SHIFT14,D7      ;/16384 SHFT
F'0005F2        303C'FFFF       487.          MOVE.W #NPTE-1,D0        ;MAJ LOOP CNTR
F'0005F6        287C'FFFFFFFC   488.          MOVEA.L #NPT-4,A4        ;INDX TO ARRAYS
F'0005FC        43FA 025A       489.          LEA TWD2(PC),A1          ;TWID FACT PNTR
F'000600        41FA ****       490.          LEA DATAII(PC),A0        ;XDATA POINTER
F'000604        2648            491.  TWDL2   MOVEA.L A0,A3
F'000606        D7CC            492.          ADDA.L A4,A3
F'000608        3213            493.  ADDL2   MOVE.W (A3),D1           ;GET REAL PART
F'00060A        48C1            494.          EXT.L D1                 ;REAL SUM IN D1
F'00060C        362B 0100       495.          MOVE.W 256(A3),D3
F'000610        48C3            496.          EXT.L D3
F'000612        9283            497.          SUB.L D3,D1
F'000614        362B 0200       498.          MOVE.W 512(A3),D3
F'000618        48C3            499.          EXT.L D3
F'00061A        D283            500.          ADD.L D3,D1
F'00061C        362B 0300       501.          MOVE.W 768(A3),D3
F'000620        48C3            502.          EXT.L D3
F'000622        9283            503.          SUB.L D3,D1              ;REAL SUM DONE
F'000624        342B 0002       504.          MOVE.W 2(A3),D2          ;IMAG SUM IN D2
F'000628        48C2            505.          EXT.L D2
F'00062A        362B 0102       506.          MOVE.W 258(A3),D3
F'00062E        48C3            507.          EXT.L D3
F'000630        9483            508.          SUB.L D3,D2
F'000632        362B 0202       509.          MOVE.W 514(A3),D3
F'000636        48C3            510.          EXT.L D3
F'000638        D483            511.          ADD.L D3,D2
F'00063A        362B 0302       512.          MOVE.W 770(A3),D3
F'00063E        48C3            513.          EXT.L D3
F'000640        9483            514.          SUB.L D3,D2              ;IMAG SUM DONE
F'000642        E481            515.          ASR.L #2,D1
F'000644        E482            516.          ASR.L #2,D2              ;ADJUST SUMS
                                517.  *
                                518.  * NOW DO TWIDDLE FACTOR MULTIPLY
                                519.  *
F'000646        3A01            520.          MOVE.W D1,D5
F'000648        3831 C800       521.          MOVE.W 0(A1,A4.L),D4
F'00064C        3631 C802       522.          MOVE.W 2(A1,A4.L),D3
F'000650        CBC4            523.          MULS D4,D5
F'000652        3C02            524.          MOVE.W D2,D6
F'000654        CDC3            525.          MULS D3,D6
F'000656        9A86            526.          SUB.L D6,D5              ;REAL IN D5
F'000658        C3C3            527.          MULS D3,D1
F'00065A        C5C4            528.          MULS D4,D2
F'00065C        D282            529.          ADD.L D2,D1              ;IMAG IN D1
F'00065E        EEA5            530.          ASR.L D7,D5
F'000660        EEA1            531.          ASR.L D7,D1
                                532.  *
                                533.  * STORE SUMMED AND TWIDDLED DATA AT XDATA IN
                                534.  * PLACE
                                535.  *
```

```
F'000662  3185 C800          536.          MOVE.W D5,0(A0,A4.L)
F'000666  3181 C802          537.          MOVE.W D1,2(A0,A4.L)
F'00066A  99CD               538.          SUBA.L A5,A4
F'00066C  51C8 FF96          539.          DBRA D0,TWDL2
                             540.  *
                             541.  * 64-POINT FFT
                             542.  * INPUT DATA AT XDATA 0-255
                             543.  *
F'000670  78'F7              544.  USHFL2  MOVEQ.L $NPTE-9,D4        ;COUNTER=K
F'000672  3C04               545.  SWAPL2  MOVE.W D4,D6          ;PREPARE BIT REV
F'000674  3E06               546.  BREV22  MOVE.W D6,D7        ;SAVE DATA
F'000676  4246               547.          CLR.W D6            ;
F'000678  7A05               548.          MOVEQ.L #5,D5       ;BIT COUNTER
F'00067A  E257               549.  REV22   ROXR.W #1,D7        ;SFT RIGHT,XTEND
F'00067C  E356               550.          ROXL.W #1,D6
F'00067E  51CD FFFA          551.          DBRA D5,REV22
F'000682  B846               552.          CMP.W D6,D4           ;I=REV(K)<=K?
F'000684  6C 14              553.          BGE.S DECK2            ;NO SWAP IF SO
F'000686  3204               554.          MOVE.W D4,D1          ;COPY OF K
F'000688  E541               555.          ASL.W #2,D1           ;K*4
F'00068A  E546               556.          ASL.W #2,D6           ;I*4
F'00068C  2430 1000          557.          MOVE.L 0(A0,D1.W),D2 ;DO SWAP
F'000690  21B0 6000 1000     558.          MOVE.L 0(A0,D6.W),0(A0,D1.W)
F'000696  2182 6000          559.          MOVE.L D2,0(A0,D6.W) ;DONE SWAP
F'00069A  51CC FFD6          560.  DECK2   DBRA D4,SWAPL2
F'00069E  7201               561.  FFT642  MOVEQ.L #1,D1   ;D1=N2=1
F'0006A0  43FA 00B6          562.          LEA SINBLK2(PC),A1  ;SIN VALS PNTR A1
F'0006A4  45FA 0132          563.          LEA COSBLK2(PC),A2  ;COS VALS PNTR A2
                             564.  *
                             565.  * L IS DOWN COUNTED #NUGAM-1 TO 0
                             566.  *
F'0006A8  3F3C 0001          567.          MOVE.W #1,-(SP)  ;(SP) IS L
F'0006AC  7405               568.          MOVEQ.L #NUGAM-1,D2         ;D2=NUGAM-1=NU1
F'0006AE  4283               569.          CLR.L D3              ;D3=K=0
F'0006B0  700E               570.          MOVEQ.L #14,D0
                             571.  *
                             572.  * INITIALIZE I=N2-1, TO 0 IN STEPS -1
                             573.  *
F'0006B2  3801               574.  INITI2  MOVE.W D1,D4     ;D4=I=N2
F'0006B4  5344               575.          SUBQ.W #1,D4     ;SET FOR DBRA
                             576.  *
                             577.  * DETERMINE P=MOD(K*2**(NUGAM-L),64)
                             578.  *
                             579.  *
F'0006B6  3C03               580.  LOOPIN2 MOVE.W D3,D6     ;D6=K
F'0006B8  3A17               581.          MOVE.W (SP),D5  ;GET L
F'0006BA  5D45               582.          SUBQ.W #NUGAM,D5 ;L-NUGAM
F'0006BC  4445               583.          NEG.W D5         ;NUGAM-L
F'0006BE  EB66               584.          ASL.W D5,D6      ;K*2**(NUGAM-L)
F'0006C0  0246 003F          585.          ANDI.W #$003F,D6 ;MOD 64
                             586.  *
                             587.  * D6 HAS INDEX REQUIRED
                             588.  *
F'0006C4  DC46               589.          ADD.W D6,D6      ;D6*2 FOR WORD BOUND
F'0006C6  3A71 6000          590.          MOVEA.W 0(A1,D6.W),A5 ;A5 PNTR IM(W**P)
F'0006CA  2E03               591.          MOVE.L D3,D7     ;D7=K
F'0006CC  2A03               592.          MOVE.L D3,D5     ;D5 TOO
F'0006CE  E545               593.          ASL.W #2,D5      ;K*4 FOR LWORD BOUND
F'0006D0  4DF0 5000          594.          LEA 0(A0,D5.W),A6    ;A6 IS INDEX X(K)
F'0006D4  DE41               595.          ADD.W D1,D7      ;D7=K+N2
F'0006D6  E547               596.          ASL.W #2,D7      ;D7=4(K+N2) LWORD BND
F'0006D8  47F0 7000          597.          LEA 0(A0,D7.W),A3 ;A3 PNT RE(X(K+N2))
F'0006DC  3A32 6000          598.          MOVE.W 0(A2,D6.W),D5    ;GET RE(W**P)
F'0006E0  3E05               599.          MOVE.W D5,D7     ;PUT ALSO IN D7
F'0006E2  CBD3               600.          MULS (A3),D5     ;RE(W**P)*RE(X(K+N2))
F'0006E4  3C0D               601.          MOVE.W A5,D6  ;GET IM(W**P)
F'0006E6  CDEB 0002          602.          MULS 2(A3),D6    ;IM(W**P)*IM(X(K+N2))
F'0006EA  9A86               603.          SUB.L D6,D5      ;D5=RE(T1)
F'0006EC  E0A5               604.          ASR.L D0,D5      ;/16384 TO SCALE
F'0006EE  3C0D               605.          MOVE.W A5,D6  ;GET IM(W**P)
F'0006F0  CDD3               606.          MULS (A3),D6     ;IM(W**P)*RE(X(K+N2))
F'0006F2  CFEB 0002          607.          MULS 2(A3),D7    ;RE(W**P)*IM(X(K+N2))
F'0006F6  DC87               608.          ADD.L D7,D6      ,D6=IM(T1)
F'0006F8  E0A6               609.          ASR.L D0,D6      ;/16384 TO SCALE
F'0006FA  3E16               610.          MOVE.W (A6),D7   ;RE(X(K))
F'0006FC  9E45               611.          SUB.W D5,D7      ;RE(X(K))-RE(T1)
F'0006FE  E247               612.          ASR.W #1,D7      ;/2
F'000700  3687               613.          MOVE.W D7,(A3)   ;STORE ANSWER
F'000702  3E2E 0002          614.          MOVE.W 2(A6),D7 ;IM(X(K))
F'000706  9E46               615.          SUB.W D6,D7      ;IM(X(K))-IM(T1)
F'000708  E247               616.          ASR.W #1,D7      ;/2
```

```
F'00070A  3747 0002        617.          MOVE.W D7,2(A3) ;STORE ANSWER
F'00070E  DB56             618.          ADD.W D5,(A6)   ;RE(X(K))+RE(T1)
F'000710  E0D6             619.          ASR.W (A6)      ;/2
F'000712  DD6E 0002        620.          ADD.W D6,2(A6)  ;IM(X(K))+IM(T1)
F'000716  E0EE 0002        621.          ASR.W 2(A6)     ;/2
F'00071A  5243             622.          ADDQ.W #1,D3    ;INCREMENT K
F'00071C  51CC FF98        623.          DBRA D4,LOOPIN2  ;DO TILL I=-1
F'000720  D641             624.          ADD.W D1,D3     ;K=K+N2
F'000722  0C43'FFFF        625.          CMPI.W #NPTE-1,D3 ;K<N-1 ?
F'000726  6D 8A            626.          BLT INITI2      ;DO AGAIN IF SO
F'000728  4243             627.          CLR.W D3        ;K=0
F'00072A  5342             628.          SUBQ.W #1,D2    ;NU1=NU1-1
F'00072C  E341             629.          ASL.W #1,D1     ;N2=N2*2
F'00072E  5257             630.          ADDQ.W #1,(SP)
F'000730  0C57 0007        631.          CMPI.W #NUGAM+1,(SP)
F'000734  6600 FF7C        632.          BNE INITI2   ;DO TILL L=-1
F'000738  301F             633.          MOVE.W (SP)+,D0
                           634.  *
                           635.  * STOP TIMER
                           636.  *
F'00073A  4239'00000000    637.          CLR.B TCR
                           638.  *
                           639.  * ASSERT SRQ AND WAIT FOR ACKNOWLEDGE
                           640.  *
F'000740  4E43             641.          TRAP #SRQASRT
                           642.  * SEND OUT 64 COMPLEX POINTS CALCULATED
>>> WARNING line[643] Expr. type   [643] SLVFFT.A68
F'000742  303C ****        643.          MOVE.W #NPTE*4,D0
F'000746  13FC 00'00       644.          MOVE.B #ENABLEB,PGCR
          '00000000
F'00074E  4E46             645.          TRAP #OUTDATA
F'000750  4239'00000000    646.          CLR.B PGCR
                           647.  *
                           648.  * NOW DO RTS AND RETURN TO READY FOR DATA
                           649.  * STATE TO DO OVER!
F'000756  4E75             650.          RTS
                           651.  *
                           652.  * DONE!
                           653.  *
                           654.  * SIN (-2*PI*I/64)*16384 FOR I=0 TO 63
                           655.  *
F'000758  0000 F9BA F384   656.  SINBLK2 DC.W       0, -1606, -3196, -4756
          ED6C
F'000760  E782 E1D5 DC72   657.          DC.W   -6270, -7723, -9102,-10394
          D766
F'000768  D2BF CE87 CAC9   658.          DC.W  -11585,-12665,-13623,-14449
          C78F
F'000770  C4DF C2C1 C13B   659.          DC.W  -15137,-15679,-16069,-16305
          C04F
F'000778  C000 C04F C13B   660.          DC.W  -16384,-16305,-16069,-15679
          C2C1
F'000780  C4DF C78F CAC9   661.          DC.W  -15137,-14449,-13623,-12665
          CE87
F'000788  D2BF D766 DC72   662.          DC.W  -11585,-10394, -9102, -7723
          E1D5
F'000790  E782 ED6C F384   663.          DC.W   -6270, -4756, -3196, -1606
          F9BA
F'000798  0000 0646 0C7C   664.          DC.W       0,  1606,  3196,  4756
          1294
F'0007A0  187E 1E2B 238E   665.          DC.W    6270,  7723,  9102, 10394
          289A
F'0007A8  2D41 3179 3537   666.          DC.W   11585, 12665, 13623, 14449
          3871
F'0007B0  3B21 3D3F 3E C5  667.          DC.W   15137, 15679, 16069, 16305
          3FB1
F'0007B8  4000 3FB1 3EC5   668.          DC.W   16384, 16305, 16069, 15679
          3D3F
F'0007C0  3B21 3871 3537   669.          DC.W   15137, 14449, 13623, 12665
          3179
F'0007C8  2D41 289A 238E   670.          DC.W   11585, 10394,  9102,  7723
          1E2B
F'0007D0  187E 1294 0C7C   671.          DC.W    6270,  4756,  3196,  1606
          0646
                           672.  *
                           673.  * COS(-2*PI*I/64)*16384 FOR I=0 TO 63
                           674.  *
F'0007D8  4000 3FB1 3EC5   675.  COSBLK2 DC.W   16384, 16305, 16069, 15679
          3D3F
F'0007E0  3B21 3871 3537   676.          DC.W   15137, 14449, 13623, 12665
          3179
F'0007E8  2D41 289A 238E   677.          DC.W   11585, 10394,  9102,  7723
```

```
                1E2B
F'0007F0        187E 1294 0C7C  678.            DC.W    6270,  4756,  3196,  1606
                0646
F'0007F8        0000 F9BA F384  679.            DC.W       0, -1606, -3196, -4756
                ED6C
F'000800        E782 E1D5 DC72  680.            DC.W   -6270, -7723, -9102,-10394
                D766
F'000808        D2BF CE87 CAC9  681.            DC.W  -11585,-12665,-13623,-14449
                C78F
F'000810        C4DF C2C1 C13B  682.            DC.W  -15137,-15679,-16069,-16305
                C04F
F'000818        C000 C04F C13B  683.            DC.W  -16384,-16305,-16069,-15679
                C2C1
F'000820        C4DF C78F CAC9  684.            DC.W  -15137,-14449,-13623,-12665
                CE87
F'000828        D2BF D766 DC72  685.            DC.W  -11585,-10394, -9102, -7723
                E1D5
F'000830        E782 ED6C F384  686.            DC.W   -6270, -4756, -3196, -1606
                F9BA
F'000838        0000 0646 0C7C  687.            DC.W       0,  1606,  3196,  4756
                1294
F'000840        187E 1E2B 238E  688.            DC.W    6270,  7723,  9102, 10394
                289A
F'000848        2D41 3179 3537  689.            DC.W   11585, 12665, 13623, 14449
                3871
F'000850        3B21 3D3F 3EC5  690.            DC.W   15137, 15679, 16069, 16305
                3FB1
                                691.  *
                                692.  * TWIDDLE FACTOR TABLE TWD2
                                693.  *
F'000858        4000 0000 3FEC  694.  TWD2      DC.W   16384,     0, 16364,  -804
                FCDC
F'000860        3FB1 F9BA 3F4F  695.            DC.W   16305, -1606, 16207, -2404
                F69C
F'000868        3EC5 F384 3E15  696.            DC.W   16069, -3196, 15893, -3981
                F073
F'000870        3D3F ED6C 3C42  697.            DC.W   15679, -4756, 15426, -5520
                EA70
F'000878        3B21 E782 39DB  698.            DC.W   15137, -6270, 14811, -7005
                E4A3
F'000880        3871 E1D5 36E5  699.            DC.W   14449, -7723, 14053, -8423
                DF19
F'000888        3537 DC72 3368  700.            DC.W   13623, -9102, 13160, -9760
                D9E0
F'000890        3179 D766 2F6C  701.            DC.W   12665,-10394, 12140,-11003
                D505
F'000898        2D41 D2BF 2AFB  702.            DC.W   11585,-11585, 11003,-12140
                D094
F'0008A0        289A CE87 2620  703.            DC.W   10394,-12665,  9760,-13160
                CC98
F'0008A8        238E CAC9 20E7  704.            DC.W    9102,-13623,  8423,-14053
                C91B
F'0008B0        1E2B C78F 1B5D  705.            DC.W    7723,-14449,  7005,-14811
                C625
F'0008B8        187E C4DF 1590  706.            DC.W    6270,-15137,  5520,-15426
                C3BE
F'0008C0        1294 C2C1 0F8D  707.            DC.W    4756,-15679,  3981,-15893
                C1EB
F'0008C8        0C7C C13B 0964  708.            DC.W    3196,-16069,  2404,-16207
                C0B1
F'0008D0        0646 C04F 0324  709.            DC.W    1606,-16305,   804,-16364
                C014
F'0008D8        0000 C000 FCDC  710.            DC.W       0,-16384,  -804,-16364
                C014
F'0008E0        F9BA C04F F69C  711.            DC.W   -1606,-16305, -2404,-16207
                C0B1
F'0008E8        F384 C13B F073  712.            DC.W   -3196,-16069, -3981,-15893
                C1EB
F'0008F0        ED6C C2C1 EA70  713.            DC.W   -4756,-15679, -5520,-15426
                C3BE
F'0008F8        E782 C4DF E4A3  714.            DC.W   -6270,-15137, -7005,-14811
                C625
F'000900        E1D5 C78F DF19  715.            DC.W   -7723,-14449, -8423,-14053
                C91B
F'000908        DC72 CAC9 D9E0  716.            DC.W   -9102,-13623, -9760,-13160
                CC98
F'000910        D766 CE87 D505  717.            DC.W  -10394,-12665,-11003,-12140
                D094
F'000918        D2BF D2BF D094  718.            DC.W  -11585,-11585,-12140,-11003
                D505
F'000920        CE87 D766 CC98  719.            DC.W  -12665,-10394,-13160, -9760
```

```
                D9E0
F'000928  CAC9 DC72 C91B  720.          DC.W  -13623, -9102,-14053, -8423
                DF19
F'000930  C78F E1D5 C625  721.          DC.W  -14449, -7723,-14811, -7005
                E4A3
F'000938  C4DF E782 C3BE  722.          DC.W  -15137, -6270,-15426, -5520
                EA70
F'000940  C2C1 ED6C C1EB  723.          DC.W  -15679, -4756,-15893, -3981
                F073
F'000948  C13B F384 C0B1  724.          DC.W  -16069, -3196,-16207, -2404
                F69C
F'000950  C04F F9BA C014  725.          DC.W  -16305, -1606,-16364,  -804
                FCDC
                          726.
                          727.  ********************
                          728.  * SLAVE 3 PROGRAM *
                          729.  ********************
                          730.  *
                          731.  * INPUT DATA AT DATAII, TWIDDLE FACTORS AT
                          732.  * TWD3
                          733.  *
                          734.  * SELF TIME CODE AUG 22,1991
                          735.  *
F'000958  4239'00000000   736.  S3PRO   CLR.B TCR
F'00095E  23FC'00000000   737.          MOVE.L #CNTSTRT,TCR+2
                '00000002
F'000968  13FC 00'00      738.          MOVE.B #ENABLTM,TCR
                '00000000
F'000970  2A7C'00000000   739.          MOVEA.L #NPE,A5          ;ADDR INC
F'000976  7E'00           740.          MOVEQ.L #SHIFT14,D7      ;/16384 SHFT
F'000978  303C'FFFF       741.          MOVE.W #NPTE-1,D0        ;MAJ LOOP CNTR
F'00097C  287C'FFFFFFFC   742.          MOVEA.L #NPT-4,A4        ;INDX TO ARRAYS
F'000982  43FA 025A       743.          LEA TWD3(PC),A1          ;TWID FACT PNTR
F'000986  41FA ****       744.          LEA DATAII(PC),A0        ;XDATA POINTER
F'00098A  2648            745.  TWDL3   MOVEA.L A0,A3
F'00098C  D7CC            746.          ADDA.L A4,A3
F'00098E  3213            747.  ADDL3   MOVE.W (A3),D1           ;GET REAL PART
F'000990  48C1            748.          EXT.L D1                 ;REAL SUM IN D1
F'000992  362B 0102       749.          MOVE.W 258(A3),D3
F'000996  48C3            750.          EXT.L D3
F'000998  9283            751.          SUB.L D3,D1
F'00099A  362B 0200       752.          MOVE.W 512(A3),D3
F'00099E  48C3            753.          EXT.L D3
F'0009A0  9283            754.          SUB.L D3,D1
F'0009A2  362B 0302       755.          MOVE.W 770(A3),D3
F'0009A6  48C3            756.          EXT.L D3
F'0009A8  D283            757.          ADD.L D3,D1              ;REAL SUM DONE
F'0009AA  342B 0002       758.          MOVE.W 2(A3),D2          ;IMAG SUM IN D2
F'0009AE  48C2            759.          EXT.L D2
F'0009B0  362B 0100       760.          MOVE.W 256(A3),D3
F'0009B4  48C3            761.          EXT.L D3
F'0009B6  D483            762.          ADD.L D3,D2
F'0009B8  362B 0202       763.          MOVE.W 514(A3),D3
F'0009BC  48C3            764.          EXT.L D3
F'0009BE  9483            765.          SUB.L D3,D2
F'0009C0  362B 0300       766.          MOVE.W 768(A3),D3
F'0009C4  48C3            767.          EXT.L D3
F'0009C6  9483            768.          SUB.L D3,D2              ;IMAG SUM DONE
F'0009C8  E482            769.          ASR.L #2,D2              ;ADJUST SUMS
F'0009CA  E481            770.          ASR.L #2,D1              ;ADJUST SUMS
                          771.  *
                          772.  * NOW DO TWIDDLE FACTOR MULTIPLY
                          773.  *
F'0009CC  3A01            774.          MOVE.W D1,D5
F'0009CE  3831 C800       775.          MOVE.W 0(A1,A4.L),D4
F'0009D2  3631 C802       776.          MOVE.W 2(A1,A4.L),D3
F'0009D6  CBC4            777.          MULS D4,D5
F'0009D8  3C02            778.          MOVE.W D2,D6
F'0009DA  CDC3            779.          MULS D3,D6
F'0009DC  9A86            780.          SUB.L D6,D5              ;REAL IN D5
F'0009DE  C3C3            781.          MULS D3,D1
F'0009E0  C5C4            782.          MULS D4,D2
F'0009E2  D282            783.          ADD.L D2,D1              ;IMAG IN D1
F'0009E4  EEA5            784.          ASR.L D7,D5
F'0009E6  EEA1            785.          ASR.L D7,D1
                          786.  *
                          787.  * STORE SUMMED AND TWIDDLED DATA AT XDATA IN
                          788.  * PLACE
                          789.  *
F'0009E8  3185 C800       790.          MOVE.W D5,0(A0,A4.L)
F'0009EC  3181 C802       791.          MOVE.W D1,2(A0,A4.L)
```

```
F'0009F0  99CD             792.          SUBA.L A5,A4
F'0009F2  51C8 FF96        793.          DBRA D0,TWDL3
                           794.  *
                           795.  * 64-POINT FFT
                           796.  * INPUT DATA AT XDATA (0-255)
                           797.  *
F'0009F6  78'F7            798.  USHFL3  MOVEQ.L #NOTE-9,D4        ;COUNTER=K
F'0009F8  3C04             799.  SWAPL3  MOVE.W D4,D6             ;PREPARE BIT REV
F'0009FA  3E06             800.  BREV23  MOVE.W D6,D7            ;SAVE DATA
F'0009FC  4246             801.          CLR.W D6                ;
F'0009FE  7A05             802.          MOVEQ.L #5,D5           ;BIT COUNTER
F'000A00  E257             803.  REV23   ROXR.W #1,D7            ;SFT RIGHT,XTEND
F'000A02  E356             804.          ROXL.W #1,D6
F'000A04  51CD FFFA        805.          DBRA D5,REV23
F'000A08  B846             806.          CMP.W D6,D4             ;I=REV(K)<=K?
F'000A0A  6C 14            807.          BGE.S DECK3              ,NO SWAP IF SO
F'000A0C  3204             808.          MOVE.W D4,D1            ;COPY OF K
F'000A0E  E541             809.          ASL.W #2,D1             ;K*4
F'000A10  E546             810.          ASL.W #2,D6             ;I*4
F'000A12  2430 1000        811.          MOVE.L 0(A0,D1.W),D7 ;DO SWAP
F'000A16  21B0 6000 1000   812.          MOVE.L 0(A0,D6.W),0  ),D1.W)
F'000A1C  2182 6000        813.          MOVE L D2,0(A0,D6.W) ;DONE SWAP
F'000A20  51CC FFD6        814.  DECK3   DBRA D4,SWAPL3
F'000A24  7201             815.  FFT643  MOVEQ.L #1,D1    ;D1=N2=1
F'000A26  43FA 00B6        816.          LEA SINBLK3(PC),A1  ;SIN VALS PNTR A1
F'000A2A  45FA 0132        817.          LEA COSBLK3(PC),A2  ;COS VALS PNTR A2
                           818.  *
                           819.  * L IS DOWN COUNTED #NUGAM-1 TO 0
                           820.  *
F'000A2E  3F3C 0001        821.          MOVE.W #1,-(SP)  ;(SP) IS L
F'000A32  7405             822.          MOVEQ.L #NUGAM-1,D2          ;D2=NUGAM-1=NU1
F'000A34  4283             823.          CLR.L D3                ;D3=K=0
F'000A36  700E             824.          MOVEQ.L #14,D0
                           825.  *
                           826.  * INITIALIZE I=N2-1, TO 0 IN STEPS -1
                           827.  *
F'000A38  3801             828.  INITI3  MOVE.W D1,D4     ;D4=I=N2
F'000A3A  5344             829.          SUBQ.W #1,D4     ;SET FOR DBRA
                           830.  *
                           831.  * DETERMINE P=MOD(K*2**(NUGAM-L),64)
                           832.  *
                           833.  *
F'000A3C  3C03             834.  LOOPIN3 MOVE.W D3,D6     ;D6=K
F'000A3E  3A17             835.          MOVE.W (SP),D5   ;GET L
F'000A40  5D45             836.          SUBQ.W #NUGAM,D5 ;L-NUGAM
F'000A42  4445             837.          NEG.W D5         ;NUGAM-L
F'000A44  EB66             838.          ASL.W D5,D6      ;K*2**(NUGAM-L)
F'000A46  0246 003F        839.          ANDI.W #$003F,D6 ;MOD 64
                           840.  *
                           841.  * D6 HAS INDEX REQUIRED
F'000A4A  DC46             842.          ADD.W D6,D6      ;D6*2 FOR WORD BOUND
F'000A4C  3A71 6000        843.          MOVEA.W 0(A1,D6.W),A5 ;A5 PNTR IM(W**P)
F'000A50  2E03             844.          MOVE.L D3,D7     ;D7=K
F'000A52  2A03             845.          MOVE.L D3,D5     ;D5 TOO
F'000A54  E545             846.          ASL.W #2,D5      ;K*4 FOR LWORD BOUND
F'000A56  4DF0 5000        847.          LEA 0(A0,D5.W),A6    ;A6 IS INDEX X(K)
F'000A5A  DE41             848.          ADD.W D1,D7      ;D7=K+N2
F'000A5C  E547             849.          ASL.W #2,D7      ;D7=4(K+N2) LWORD BND
F'000A5E  47F0 7000        850.          LEA 0(A0,D7.W),A3 ;A3 PNT RE(X(K+N2))
F'000A62  3A32 6000        851.          MOVE.W 0(A2,D6.W),D5     ;GET RE(W**P)
F'000A66  3E05             852.          MOVE.W D5,D7     ;PUT ALSO IN D7
F'000A68  CBD3             853.          MULS (A3),D5     ;RE(W**P)*RE(X(K+N2))
F'000A6A  3C0D             854.          MOVE.W A5,D6  ;GET IM(W**P)
F'000A6C  CDEB 0002        855.          MULS 2(A3),D6    ;IM(W**P)*IM(X(K+N2))
F'000A70  9A86             856.          SUB.L D6,D5      ;D5=RE(T1)
F'000A72  E0A5             857.          ASR.L D0,D5      ;/16384 TO SCALE
F'000A74  3C0D             858.          MOVE.W A5,D6  ;GET IM(W**P)
F'000A76  CDD3             859.          MULS (A3),D6     ;IM(W**P)*RE(X(K+N2))
F'000A78  CFEB 0002        860.          MULS 2(A3),D7    ;RE(W**P)*IM(X(K+N2))
F'000A7C  DC87             861.          ADD.L D7,D6      ;D6=IM(T1)
F'000A7E  E0A6             862.          ASR.L D0,D6      ;/16384 TO SCALE
F'000A80  3E16             863.          MOVE.W (A6),D7   ;RE(X(K))
F'000A82  9E45             864.          SUB.W D5,D7      ;RE(X(K))-RE(T1)
F'000A84  E247             865.          ASR.W #1,D7      ;/2
F'000A86  3687             866.          MOVE.W D7,(A3)   ;STORE ANSWER
F'000A88  3E2E 0002        867.          MOVE.W 2(A6),D7 ;IM(X(K))
F'000A8C  9E46             868.          SUB.W D6,D7      ;IM(X(K))-IM(T1)
F'000A8E  E247             869.          ASR.W #1,D7      ;/2
F'000A90  3747 0002        870.          MOVE.W D7,2(A3) ;STORE ANSWER
F'000A94  DB56             871.          ADD.W D5,(A6)    ;RE(X(K))+RE(T1)
F'000A96  E0D6             872.          ASR.W (A6)       ;/2
```

```
F'000A98  DD6E 0002        873.          ADD.W D6,2(A6)   ;IM(X(K))+IM(T1)
F'000A9C  E0EE 0002        874.          ASR.W 2(A6)      ;/2
F'000AA0  5243             875.          ADDQ.W #1,D3     ;INCREMENT K
F'000AA2  51CC FF98        876.          DBRA D4,LOOPIN3  ;DO TILL I=-1
F'000AA6  D641             877.          ADD.W D1,D3      ;K=K+N2
F'000AA8  0C43'FFFF        878.          CMPI.W #NPTE-1,D3 ;K<N-1 ?
F'000AAC  6D 8A            879.          BLT INITI3       ;DO AGAIN IF SO
F'000AAE  4243             880.          CLR.W D3         ;K=0
F'000AB0  5342             881.          SUBQ.W #1,D2     ;NU1=NU1-1
F'000AB2  E341             882.          ASL.W #1,D1      ;N2=N2*2
F'000AB4  5257             883.          ADDQ.W #1,(SP)
F'000AB6  0C57 0007        884.          CMPI.W #NUGAM+1,(SP)
F'000ABA  6600 FF7C        885.          BNE INITI3
F'000ABE  301F             886.          MOVE.W (SP)+,D0
                           887.  *
                           888.  * STOP TIMER
                           889.  *
F'000AC0  4239'00000000    890.          CLR.B TCR
                           891.  *
                           892.  * ASSERT SRQ AND WAIT FOR ACKNOWLEDGE
                           893.  *
F'000AC6  4E43             894.          TRAP #SRQASRT
                           895.  * SEND OUT 64 COMPLEX POINTS CALCULATED
>>> WARNING line[896] Expr. type   [896] SLVFFT.A68
F'000AC8  303C ****        896.          MOVE.W #NPTE*4,D0
F'000ACC  13FC 00'00       897.          MOVE.B #ENABLEB,PGCR
          '00000000
F'000AD4  4E46             898.          TRAP #OUTDATA
F'000AD6  4239'00000000    899.          CLR.B PGCR
                           900.  *
                           901.  * NOW DO RTS AND RETURN TO READY FOR DATA
                           902.  * STATE TO DO OVER!
                           903.  *
F'000ADC  4E75             904.          RTS
                           905.  *
                           906.  * DONE!
                           907.  *
                           908.  * SIN (-2*PI*I/64)*16384 FOR I=0 TO 63
                           909.  *
F'000ADE  0000 F9BA F384   910.  SINBLK3 DC.W       0, -1606, -3196, -4756
          ED6C
F'000AE6  E782 E1D5 DC72   911.          DC.W   -6270, -7723, -9102,-10394
          D766
F'000AEE  D2BF CE87 CAC9   912.          DC.W  -11585,-12665,-13623,-14449
          C78F
F'000AF6  C4DF C2C1 C13B   913.          DC.W  -15137,-15679,-16069,-16305
          C04F
F'000AFE  C000 C04F C13B   914.          DC.W  -16384,-16305,-16069,-15679
          C2C1
F'000B06  C4DF C78F CAC9   915.          DC.W  -15137,-14449,-13623,-12665
          CE87
F'000B0E  D2BF D766 DC72   916.          DC.W  -11585,-10394, -9102, -7723
          E1D5
F'000B16  E782 ED6C F384   917.          DC.W   -6270, -4756, -3196, -1606
          F9BA
F'000B1E  0000 0646 0C7C   918.          DC.W       0,  1606,  3196,  4756
          1294
F'000B26  187E 1E2B 238E   919.          DC.W    6270,  7723,  9102, 10394
          289A
F'000B2E  2D41 3179 3537   920.          DC.W   11585, 12665, 13623, 14449
          3871
F'000B36  3B21 3D3F 3EC5   921.          DC.W   15137, 15679, 16069, 16305
          3FB1
F'000B3E  4000 3FB1 3EC5   922.          DC.W   16384, 16305, 16069, 15679
          3D3F
F'000B46  3B21 3871 3537   923.          DC.W   15137, 14449, 13623, 12665
          3179
F'000B4E  2D41 289A 238E   924.          DC.W   11585, 10394,  9102,  7723
          1E2B
F'000B56  187E 1294 0C7C   925.          DC.W    6270,  4756,  3196,  1606
          0646
                           926.  *
                           927.  * COS(-2*PI*I/64)*16384 FOR I=0 TO 63
                           928.  *
```

```
F'000B5E  4000 3FB1 3EC5  929.  COSBLK3 DC.W   16384,  6305, 16069, 15679
          3D3F
F'000B66  3B21 3871 3537  930.          DC.W   15137, 14449, 13623, 12665
          3179
F'000B6E  2D41 289A 238E  931.          DC.W   11585, 10394,  9102,  7723
          1E2B
F'000B76  187E 1294 0C7C  932.          DC.W    6270,  4756,  3196,  1606
          0646
F'000B7E  0000 F9BA F384  933.          DC.W       0, -1606, -3196, -4756
          ED6C
F'000B86  E782 E1D5 DC72  934.          DC.W   -6270, -7723, -9102,-10394
          D766
F'000B8E  D2BF CE87 CAC9  935.          DC.W  -11585,-12665,-13623,-14449
          C78F
F'000B96  C4DF C2C1 C13B  936.          DC.W  -15137,-15679,-16069,-16305
          C04F
F'000B9E  C000 C04F C13B  937.          DC.W  -16384,-16305,-16069,-15679
          C2C1
F'000BA6  C4DF C78F CAC9  938           DC.W  -15137,-14449,-13623,-12665
          CE87
F'000BAE  D2BF D766 DC72  939.          DC.W  -11585,-10394, -9102, -7723
          E1D5
F'000BB6  E782 ED6C F384  940.          DC.W   -6270, -4756, -3196, -1606
          F9BA
F'000BBE  0000 0646 0C7C  941.          DC.W       0,  1606,  3196,  4756
          1294
F'000BC6  187E 1E2B 238E  942.          DC.W    6270,  7723,  9102, 10394
          289A
F'000BCE  2D41 3179 3537  943.          DC.W   11585, 12665, 13623, 14449
          3871
F'000BD6  3B21 3D3F 3EC5  944.          DC.W   15137, 15679, 16069, 16305
          3FB1
                          945.  *
                          946.  * TWIDDLE FACTOR TABLE 3 TWD3
                          947.  *
F'000BDE  4000 0000 3FD4  948.  TWD3    DC.W   16384,     0, 16340, -1205
          FB4B
F'000BE6  3F4F F69C 3E72  949.          DC.W   16207, -2404, 15986, -3590
          F1FA
F'000BEE  3D3F ED6C 3BB6  950.          DC.W   15679, -4756, 15286, -5897
          E8F7
F'000BF6  39DB E4A3 37B0  951.          DC.W   14811, -7005, 14256, -8076
          E074
F'000BFE  3537 DC72 3274  952.          DC.W   13623, -9102, 12916,-10080
          D8A0
F'000C06  2F6C D505 2C21  953.          DC.W   12140,-11003, 11297,-11866
          D1A6
F'000C0E  289A CE87 24DA  954.          DC.W   10394,-12665,  9434,-13395
          CBAD
F'000C16  20E7 C91B 1CC6  955.          DC.W    8423,-14053,  7366,-14635
          C6D5
F'000C1E  187E C4DF 1413  956.          DC.W    6270,-15137,  5139,-15557
          C33B
F'000C26  0F8D C1EB 0AF1  957.          DC.W    3981,-15893,  2801,-16143
          C0F1
F'000C2E  0646 C04F 0192  958.          DC.W    1606,-16305,   402,-16379
          C005
F'000C36  FCDC C014 F82A  959.          DC.W    -804,-16364, -2006,-16261
          C07B
F'000C3E  F384 C13B EEEE  960.          DC.W   -3196,-16069, -4370,-15791
          C251
F'000C46  EA70 C3BE E611  961.          DC.W   -5520,-15426, -6639,-14978
          C57E
F'000C4E  E1D5 C78F DDC3  962.          DC.W   -7723,-14449, -8765,   842
          C9EE
F'000C56  D9E0 CC98 D632  963.          DC.W   -9760,-13160,-10702,-12406
          CF8A
F'000C5E  D2BF D2BF CF8A  964.          DC.W  -11585,-11585,-12406,-10702
          D632
F'000C66  CC98 D9E0 C9EE  965.          DC.W  -13160, -9760,-13842, -8765
          DDC3
F'000C6E  C78F E1D5 C57E  966.          DC.W  -14449, -7723,-14978, -6639
          E611
F'000C76  C3BE EA70 C251  967           DC.W  -15426, -5520,-15791, -4370
          EEEE
F'000C7E  C13B F384 C07B  968.          DC.W  -16069, -3196,-16261, -2006
          F82A
F'000C86  C014 FCDC C005  969.          DC.W  -16364,  -804,-16379,   402
          0192
F'000C8E  C04F 0646 C0F1  970.          DC.W  -16305,  1606,-16143,  2801
          0AF1
```

```
F'000C96  C1EB 0F8D C33B  971.          DC.W  -15893,  3981,-15557,  5139
          1413
F'000C9E  C4DF 187E C6D5  972.          DC.W  -15137,  6270,-14635,  7366
          1CC6
F'000CA6  C91B 20E7 CBAD  973.          DC.W  -14053,  8423,-13395,  9434
          24DA
F'000CAE  CE87 289A D1A6  974.          DC.W  -12665, 10394,-11866, 11297
          2C21
F'000CB6  D505 2F6C D8A0  975.          DC.W  -11003, 12140,-10080, 12916
          3274
F'000CBE  DC72 3537 E074  976.          DC.W   -9102, 13623, -8076, 14256
          37B0
F'000CC6  E4A3 39DB E8F7  977.          DC.W   -7005, 14811, -5897, 15286
          3BB6
F'000CCE  ED6C 3D3F F1FA  978.          DC.W   -4756, 15679, -3590, 15986
          3E72
F'000CD6  F69C 3F4F FB4B  979.          DC.W   -2404, 16207, -1205, 16340
          3FD4
F'000CDE  <1>             980.  DONE    DS.B 1
F'000CDF                  981.          END
    4 Warnings
    0 Errors
```

L.7 Input data: NULLDC

```
00000000                          1.          ORG.L $0000
00000000  0000 0000 0000          2.          DC.W $0000,$0000,$0000,$0000
          0000
00000008  0000 0000 0000          3.          DC.W $0000,$0000,$0000,$0000
          0000
00000010  0000 0000 0000          4.          DC.W $0000,$0000,$0000,$0000
          0000
00000018  0000 0000 0000          5.          DC.W $0000,$0000,$0000,$0000
          0000
00000020  0000 0000 0000          6.          DC.W $0000,$0000,$0000,$0000
          0000
00000028  0000 0000 0000          7.          DC.W $0000,$0000,$0000,$0000
          0000
00000030  0000 0000 0000          8.          DC.W $0000,$0000,$0000,$0000
          0000
00000038  0000 0000 0000          9.          DC.W $0000,$0000,$0000,$0000
          0000
00000040  0000 0000 0000         10.          DC.W $0000,$0000,$0000,$0000
          0000
00000048  0000 0000 0000         11.          DC.W $0000,$0000,$0000,$0000
          0000
00000050  0000 0000 0000         12.          DC.W $0000,$0000,$0000,$0000
          0000
00000058  0000 0000 0000         13.          DC.W $0000,$0000,$0000,$0000
          0000
00000060  0000 0000 0000         14.          DC.W $0000,$0000,$0000,$0000
          0000
00000068  0000 0000 0000         15.          DC.W $0000,$0000,$0000,$0000
          0000
0000007C  0000 0000 0000         16.          DC.W $0000,$0000,$0000,$0000
          0000
00000078  0000 0000 0000         17.          DC.W $0000,$0000,$0000,$0000
          0000
00000080  0000 0000 0000         18.          DC.W $0000,$0000,$0000,$0000
          0000
00000088  0000 0000 0000         19.          DC.W $0000,$0000,$0000,$0000
          0000
00000090  0000 0000 0000         20.          DC.W $0000,$0000,$0000,$0000
          0000
00000098  0000 0000 0000         21.          DC.W $0000,$0000,$0000,$0000
          0000
000000A0  0000 0000 0000         22.          DC.W $0000,$0000,$0000,$0000
          0000
000000A8  0000 0000 0000         23.          DC.W $0000,$0000,$0000,$0000
          0000
000000B0  0000 0000 0000         24.          DC.W $0000,$0000,$0000,$0000
          0000
000000B8  0000 0000 0000         25.          DC.W $0000,$0000,$0000,$0000
          0000
000000C0  0000 0000 0000         26.          DC.W $0000,$0000,$0000,$0000
          0000
000000C8  0000 0000 0000         27.          DC.W $0000,$0000,$0000,$0000
          0000
000000D0  0000 0000 0000         28.          DC.W $0000,$0000,$0000,$0000
          0000
000000D8  0000 0000 0000         29.          DC.W $0000,$0000,$0000,$0000
          0000
000000E0  0000 0000 0000         30.          DC.W $0000,$0000,$0000,$0000
          0000
000000E8  0000 0000 0000         31.          DC.W $0000,$0000,$0000,$0000
          0000
000000F0  0000 0000 0000         32.          DC.W $0000,$0000,$0000,$0000
          0000
000000F8  0000 0000 0000         33.          DC.W $0000,$0000,$0000,$0000
          0000
00000100  0000 0000 0000         34.          DC.W $0000,$0000,$0000,$0000
          0000
00000108  0000 0000 0000         35.          DC.W $0000,$0000,$0000,$0000
          0000
00000110  0000 0000 0000         36.          DC.W $0000,$0000,$0000,$0000
          0000
00000118  0000 0000 0000         37.          DC.W $0000,$0000,$0000,$0000
          0000
00000120  0000 0000 0000         38           DC.W $0000,$0000,$0000,$0000
          0000
00000128  0000 0000 0000         39.          DC.W $0000,$0000,$0000,$0000
          0000
00000130  0000 0000 0000         40.          DC.W $0000,$0000,$0000,$0000
```

```
                0000
00000138        0000 0000 0000    41.           DC.W $0000,$0000,$0000,$0000
                0000
00000140        0000 0000 0000    42.           DC.W $0000,$0000,$0000,$0000
                0000
00000148        0000 0000 0000    43.           DC.W $0000,$0000,$0000,$0000
                0000
00000150        0000 0000 0000    44.           DC.W $0000,$0000,$0000,$0000
                0000
00000158        0000 0000 0000    45.           DC.W $0000,$0000,$0000,$0000
                0000
00000160        0000 0000 0000    46.           DC.W $0000,$0000,$0000,$0000
                0000
00000168        0000 0000 0000    47.           DC.W $0000,$0000,$0000,$0000
                0000
00000170        0000 0000 0000    48.           DC.W $0000,$0000,$0000,$0000
                0000
00000178        0000 0000 0000    49.           DC.W $0000,$0000,$0000,$0000
                0000
00000180        0000 0000 0000    50.           DC.W $0000,$0000,$0000,$0000
                0000
00000188        0000 0000 0000    51.           DC.W $0000,$0000,$0000,$0000
                0000
00000190        0000 0000 0000    52.           DC.W $0000,$0000,$0000,$0000
                0000
00000198        0000 0000 0000    53.           DC.W $0000,$0000,$0000,$0000
                0000
000001A0        0000 0000 0000    54.           DC.W $0000,$0000,$0000,$0000
                0000
000001A8        0000 0000 0000    55.           DC.W $0000,$0000,$0000,$0000
                0000
000001B0        0000 0000 0000    56.           DC.W $0000,$0000,$0000,$0000
                0000
000001B8        0000 0000 0000    57.           DC.W $0000,$0000,$0000,$0000
                0000
000001C0        0000 0000 0000    58.           DC W $0000,$0000,$0000,$0000
                0000
000001C8        0000 0000 0000    59.           DC.W $0000,$0000,$0000,$0000
                0000
000001D0        0000 0000 0000    60.           DC.W $0000,$0000,$0000,$0000
                0000
000001D8        0000 0000 0000    61.           DC.W $0000,$0000,$0000,$0000
                0000
000001E0        0000 0000 0000    62.           DC.W $0000,$0000,$0000,$0000
                0000
000001E8        0000 0000 0000    63.           DC.W $0000,$0000,$0000,$0000
                0000
000001F0        0000 0000 0000    64.           DC.W $0000,$0000,$0000,$0000
                0000
000001F8        0000 0000 0000    65.           DC.W $0000,$0000,$0000,$0000
                0000
00000200        0000 0000 0000    66.           DC.W $0000,$0000,$0000,$0000
                0000
00000208        0000 0000 0000    67.           DC.W $0000,$0000,$0000,$0000
                0000
00000210        0000 0000 0000    68.           DC.W $0000,$0000,$0000,$0000
                0000
00000218        0000 0000 0000    69.           DC.W $0000,$0000,$0000,$0000
                0000
00000220        0000 0000 0000    70.           DC.W $0000,$0000,$0000,$0000
                0000
00000228        0000 0000 0000    71.           DC.W $0000,$0000,$0000,$0000
                0000
00000230        0000 0000 0000    72            DC.W $0000,$0000,$0000,$0000
                0000
00000238        0000 0000 0000    73.           DC.W $0000,$0000,$0000,$0000
                0000
00000240        0000 0000 0000    74.           DC.W $0000,$0000,$0000,$0000
                0000
00000248        0000 0000 0000    75.           DC.W $0000,$0000,$0000,$0000
                0000
00000250        0000 0000 0000    76.           DC.W $0000,$0000,$0000,$0000
                0000
00000258        0000 0000 0000    77.           DC.W $0000,$0000,$0000,$0000
                0000
00000260        0000 0000 0000    78.           DC.W $0000,$0000,$0000,$0000
                0000
00000268        0000 0000 0000    79.           DC.W $0000,$0000,$0000,$0000
                0000
00000270        0000 0000 0000    80.           DC.W $0000,$0000,$0000,$0000
                0000
```

```
00000278  0000 0000 0000 0000   81.          DC.W $0000,$0000,$0000,$0000
00000280  0000 0000 0000 0000   82.          DC.W $0000,$0000,$0000,$0000
00000288  0000 0000 0000 0000   83.          DC.W $0000,$0000,$0000,$0000
00000290  0000 0000 0000 0000   84.          DC.W $0000,$0000,$0000,$0000
00000298  0000 0000 0000 0000   85.          DC.W $0000,$0000,$0000,$0000
000002A0  0000 0000 0000 0000   86.          DC.W $0000,$0000,$0000,$0000
000002A8  0000 0000 0000 0000   87.          DC.W $0000,$0000,$0000,$0000
000002B0  0000 0000 0000 0000   88.          DC.W $0000,$0000,$0000,$0000
000002B8  0000 0000 0000 0000   89           DC.W $0000,$0000,$0000,$0000
000002C0  0000 0000 0000 0000   90.          DC.W $0000,$0000,$0000,$0000
000002C8  0000 0000 0000 0000   91.          DC.W $0000,$0000,$0000,$0000
000002D0  0000 0000 0000 0000   92           DC.W $0000,$0000,$0000,$0000
000002D8  0000 0000 0000 0000   93.          DC.W $0000,$0000,$0000,$0000
000002E0  0000 0000 0000 0000   94.          DC.W $0000,$0000,$0000,$0000
000002E8  0000 0000 0000 0000   95.          DC.W $0000,$0000,$0000,$0000
000002F0  0000 0000 0000 0000   96.          DC.W $0000,$0000,$0000,$0000
000002F8  0000 0000 0000 0000   97.          DC.W $0000,$0000,$0000,$0000
00000300  0000 0000 0000 0000   98.          DC.W $0000,$0000,$0000,$0000
00000308  0000 0000 0000 0000   99.          DC.W $0000,$0000,$0000,$0000
00000310  0000 0000 0000 0000  100.          DC.W $0000,$0000,$0000,$0000
00000318  0000 0000 0000 0000  101.          DC.W $0000,$0000,$0000,$0000
00000320  0000 0000 0000 0000  102.          DC.W $0000,$0000,$0000,$0000
00000328  0000 0000 0000 0000  103.          DC.W $0000,$0000,$0000,$0000
00000330  0000 0000 0000 0000  104           DC.W $0000,$0000,$0000,$0000
00000338  0000 0000 0000 0000  105.          DC.W $0000,$0000,$0000,$0000
00000340  0000 0000 0000 0000  106.          DC.W $0000,$0000,$0000,$0000
00000348  0000 0000 0000 0000  107.          DC.W $0000,$0000,$0000,$0000
00000350  0000 0000 0000 0000  108.          DC.W $0000,$0000,$0000,$0000
00000358  0000 0000 0000 0000  109.          DC.W $0000,$0000,$0000,$0000
00000360  0000 0000 0000 0000  110.          DC.W $0000,$0000,$0000,$0000
00000368  0000 0000 0000 0000  111.          DC.W $0000,$0000,$0000,$0000
00000370  0000 0000 0000 0000  112.          DC.W $0000,$0000,$0000,$0000
00000378  0000 0000 0000 0000  113.          DC.W $0000,$0000,$0000,$0000
00000380  0000 0000 0000 0000  114.          DC.W $0000,$0000,$0000,$0000
00000388  0000 0000 0000 0000  115.          DC.W $0000,$0000,$0000,$0000
00000390  0000 0000 0000 0000  116.          DC.W $0000,$0000,$0000,$0000
00000398  0000 0000 0000 0000  117.          DC.W $0000,$0000,$0000,$0000
000003A0  0000 0000 0000 0000  118           DC.W $0000,$0000,$0000,$0000
000003A8  0000 0000 0000 0000  119.          DC.W $0000,$0000,$0000,$0000
000003B0  0000 0000 0000 0000  120           DC.W $0000,$0000,$0000,$0000
000003B8  0000 0000 0000 0000  121.          DC.W $0000,$0000,$0000,$0000
```

```
           0000
000003C0   0000 0000 0000  122.          DC.W $0000,$0000,$0000,$0000
           0000
000003C8   0000 0000 0000  123.          DC.W $0000,$0000,$0000,$0000
           0000
000003D0   0000 0000 0000  124.          DC.W $0000,$0000,$0000,$0000
           0000
000003D8   0000 0000 0000  125.          DC.W $0000,$0000,$0000,$0000
           0000
000003E0   0000 0000 0000  126.          DC.W $0000,$0000,$0000,$0000
           0000
000003E8   0000 0000 0000  127.          DC.W $0000,$0000,$0000,$0000
           0000
000003F0   0000 0000 0000  128.          DC.W $0000,$0000,$0000,$0000
           0000
000003F8   0000 0000 0000  129.          DC.W $0000,$0000,$0000,$0000
           0000
00000400                   130.        END
    0 Errors
```

L.8 Input data: FULLDC

```
00000000                          1.          ORG.L $0000
00000000 3FFF 0000 3FFF 0000      2.          DC.W $3FFF,$0000,$3FFF,$0000
00000008 3FFF 0000 3FFF 0000      3.          DC.W $3FFF,$0000,$3FFF,$0000
00000010 3FFF 0000 3FFF 0000      4.          DC.W $3FFF,$0000,$3FFF,$0000
00000018 3FFF 0000 3FFF 0000      5.          DC.W $3FFF,$0000,$3FFF,$0000
00000020 3FFF 0000 3FFF 0000      6.          DC.W $3FFF,$0000,$3FFF,$0000
00000028 3FFF 0000 3FFF 0000      7.          DC.W $3FFF,$0000,$3FFF,$0000
00000030 3FFF 0000 3FFF 0000      8.          DC.W $3FFF,$0000,$3FFF,$0000
00000038 3FFF 0000 3FFF 0000      9.          DC.W $3FFF,$0000,$3FFF,$0000
00000040 3FFF 0000 3FFF 0000     10.          DC.W $3FFF,$0000,$3FFF,$0000
00000048 3FFF 0000 3FFF 0000     11.          DC.W $3FFF,$0000,$3FFF,$0000
00000050 3FFF 0000 3FFF 0000     12.          DC.W $3FFF,$0000,$3FFF,$0000
00000058 3FFF 0000 3FFF 0000     13.          DC.W $3FFF,$0000,$3FFF,$0000
00000060 3FFF 0000 3FFF 0000     14.          DC.W $3FFF,$0000,$3FFF,$0000
00000068 3FFF 0000 3FFF 0000     15.          DC.W $3FFF,$0000,$3FFF,$0000
00000070 3FFF 0000 3FFF 0000     16.          DC.W $3FFF,$0000,$3FFF,$0000
00000078 3FFF 0000 3FFF 0000     17.          DC.W $3FFF,$0000,$3FFF,$0000
00000080 3FFF 0000 3FFF 0000     18.          DC.W $3FFF,$0000,$3FFF,$0000
00000088 3FFF 0000 3FFF 0000     19.          DC.W $3FFF,$0000,$3FFF,$0000
00000090 3FFF 0000 3FFF 0000     20.          DC.W $3FFF,$0000,$3FFF,$0000
00000098 3FFF 0000 3FFF 0000     21.          DC.W $3FFF,$0000,$3FFF,$0000
000000A0 3FFF 0000 3FFF 0000     22.          DC.W $3FFF,$0000,$3FFF,$0000
000000A8 3FFF 0000 3FFF 0000     23.          DC.W $3FFF,$0000,$3FFF,$0000
000000B0 3FFF 0000 3FFF 0000     24.          DC.W $3FFF,$0000,$3FFF,$0000
000000B8 3FFF 0000 3FFF 0000     25.          DC.W $3FFF,$0000,$3FFF,$0000
000000C0 3FFF 0000 3FFF 0000     26.          DC.W $3FFF,$0000,$3FFF,$0000
000000C8 3FFF 0000 3FFF 0000     27.          DC.W $3FFF,$0000,$3FFF,$0000
000000D0 3FFF 0000 3FFF 0000     28.          DC.W $3FFF,$0000,$3FFF,$0000
000000D8 3FFF 0000 3FFF 0000     29.          DC.W $3FFF,$0000,$3FFF,$0000
000000E0 3FFF 0000 3FFF 0000     30.          DC.W $3FFF,$0000,$3FFF,$0000
000000E8 3FFF 0000 3FFF 0000     31.          DC.W $3FFF,$0000,$3FFF,$0000
000000F0 3FFF 0000 3FFF 0000     32.          DC.W $3FFF,$0000,$3FFF,$0000
000000F8 3FFF 0000 3FFF 0000     33.          DC.W $3FFF,$0000,$3FFF,$0000
00000100 3FFF 0000 3FFF 0000     34.          DC.W $3FFF,$0000,$3FFF,$0000
00000108 3FFF 0000 3FFF 0000     35.          DC.W $3FFF,$0000,$3FFF,$0000
00000110 3FFF 0000 3FFF 0000     36.          DC.W $3FFF,$0000,$3FFF,$0000
00000118 3FFF 0000 3FFF 0000     37.          DC.W $3FFF,$0000,$3FFF,$0000
00000120 3FFF 0000 3FFF 0000     38.          DC.W $3FFF,$0000,$3FFF,$0000
00000128 3FFF 0000 3FFF 0000     39.          DC.W $3FFF,$0000,$3FFF,$0000
00000130 3FFF 0000 3FFF 0000     40.          DC.W $3FFF,$0000,$3FFF,$0000
```

```
                0000
00000138  3FFF 0000 3FFF    41.          DC.W $3FFF,$0000,$3FFF,$0000
                0000
00000140  3FFF 0000 3FFF    42.          DC.W $3FFF,$0000,$3FFF,$0000
                0000
00000148  3FFF 0000 3FFF    43.          DC.W $3FFF,$0000,$3FFF,$0000
                0000
00000150  3FFF 0000 3FFF    44.          DC.W $3FFF,$0000,$3FFF,$0000
                0000
00000158  3FFF 0000 3FFF    45.          DC.W $3FFF,$0000,$3FFF,$0000
                0000
00000160  3FFF 0000 3FFF    46.          DC.W $3FFF,$0000,$3FFF,$0000
                0000
00000168  3FFF 0000 3FFF    47.          DC.W $3FFF,$0000,$3FFF,$0000
                0000
00000170  3FFF 0000 3FFF    48.          DC.W $3FFF,$0000,$3FFF,$0000
                0000
00000178  3FFF 0000 3FFF    49.          DC.W $3FFF,$0000,$3FFF,$0000
                0000
00000180  3FFF 0000 3FFF    50.          DC.W $3FFF,$0000,$3FFF,$0000
                0000
00000188  3FFF 0000 3FFF    51.          DC.W $3FFF,$0000,$3FFF,$0000
                0000
00000190  3FFF 0000 3FFF    52.          DC.W $3FFF,$0000,$3FFF,$0000
                0000
00000198  3FFF 0000 3FFF    53.          DC.W $3FFF,$0000,$3FFF,$0000
                0000
000001A0  3FFF 0000 3FFF    54.          DC.W $3FFF,$0000,$3FFF,$0000
                0000
000001A8  3FFF 0000 3FFF    55.          DC.W $3FFF,$0000,$3FFF,$0000
                0000
000001B0  3FFF 0000 3FFF    56.          DC.W $3FFF,$0000,$3FFF,$0000
                0000
000001B8  3FFF 0000 3FFF    57.          DC.W $3FFF,$0000,$3FFF,$0000
                0000
000001C0  3FFF 0000 3FFF    58.          DC.W $3FFF,$0000,$3FFF,$0000
                0000
000001C8  3FFF 0000 3FFF    59.          DC.W $3FFF,$0000,$3FFF,$0000
                0000
000001D0  3FFF 0000 3FFF    60.          DC.W $3FFF,$0000,$3FFF,$0000
                0000
000001D8  3FFF 0000 3FFF    61.          DC.W $3FFF,$0000,$3FFF,$0000
                0000
000001E0  3FFF 0000 3FFF    62.          DC.W $3FFF,$0000,$3FFF,$0000
                0000
000001E8  3FFF 0000 3FFF    63.          DC.W $3FFF,$0000,$3FFF,$0000
                0000
000001F0  3FFF 0000 3FFF    64.          DC.W $3FFF,$0000,$3FFF,$0000
                0000
000001F8  3FFF 0000 3FFF    65.          DC.W $3FFF,$0000,$3FFF,$0000
                0000
00000200  3FFF 0000 3FFF    66.          DC.W $3FFF,$0000,$3FFF,$0000
                0000
00000208  3FFF 0000 3FFF    67.          DC.W $3FFF,$0000,$3FFF,$0000
                0000
00000210  3FFF 0000 3FFF    68.          DC.W $3FFF,$0000,$3FFF,$0000
                0000
00000218  3FFF 0000 3FFF    69.          DC.W $3FFF,$0000,$3FFF,$0000
                0000
00000220  3FFF 0000 3FFF    70.          DC.W $3FFF,$0000,$3FFF,$0000
                0000
00000228  3FFF 0000 3FFF    71.          DC.W $3FFF,$0000,$3FFF,$0000
                0000
00000230  3FFF 0000 3FFF    72.          DC.W $3FFF,$0000,$3FFF,$0000
                0000
00000238  3FFF 0000 3FFF    73.          DC.W $3FFF,$0000,$3FFF,$0000
                0000
00000240  3FFF 0000 3FFF    74.          DC.W $3FFF,$0000,$3FFF,$0000
                0000
00000248  3FFF 0000 3FFF    75.          DC.W $3FFF,$0000,$3FFF,$0000
                0000
00000250  3FFF 0000 3FFF    76.          DC.W $3FFF,$0000,$3FFF,$0000
                0000
00000258  3FFF 0000 3FFF    77.          DC.W $3FFF,$0000,$3FFF,$0000
                0000
00000260  3FFF 0000 3FFF    78.          DC.W $3FFF,$0000,$3FFF,$0000
                0000
00000268  3FFF 0000 3FFF    79.          DC.W $3FFF,$0000,$3FFF,$0000
                0000
00000270  3FFF 0000 3FFF    80.          DC.W $3FFF,$0000,$3FFF,$0000
                0000
```

```
00000278 3FFF 0000 3FFF   81.          DC.W $3FFF,$0000,$3FFF,$0000
         0000
00000280 3FFF 0000 3FFF   82.          DC.W $3FFF,$0000,$3FFF,$0000
         0000
00000288 3FFF 0000 3FFF   83.          DC.W $3FFF,$0000,$3FFF,$0000
         0000
00000290 3FFF 0000 3FFF   84.          DC.W $3FFF,$0000,$3FFF,$0000
         0000
00000298 3FFF 0000 3FFF   85.          DC.W $3FFF,$0000,$3FFF,$0000
         0000
000002A0 3FFF 0000 3FFF   86.          DC.W $3FFF,$0000,$3FFF,$0000
         0000
000002A8 3FFF 0000 3FFF   87.          DC.W $3FFF,$0000,$3FFF,$0000
         0000
000002B0 3FFF 0000 3FFF   88.          DC.W $3FFF,$0000,$3FFF,$0000
         0000
000002B8 3FFF 0000 3FFF   89.          DC.W $3FFF,$0000,$3FFF $0000
         0000
000002C0 3FFF 0000 3FFF   90.          DC.W $3FFF,$0000,$3FFF,$0000
         0000
000002C8 3FFF 0000 3FFF   91.          DC.W $3FFF,$0000,$3FFF,$0000
         0000
000002D0 3FFF 0000 3FFF   92.          DC.W $3FFF,$0000,$3FFF,$0000
         0000
000002D8 3FFF 0000 3FFF   93.          DC.W $3FFF,$0000,$3FFF,$0000
         0000
000002E0 3FFF 0000 3FFF   94.          DC.W $3FFF,$0000,$3FFF,$0000
         0000
000002E8 3FFF 0000 3FFF   95.          DC.W $3FFF,$0000,$3FFF,$0000
         0000
000002F0 3FFF 0000 3FFF   96.          DC.W $3FFF,$0000,$3FFF,$0000
         0000
000002F8 3FFF 0000 3FFF   97.          DC.W $3FFF,$0000,$3FFF,$0000
         0000
00000300 3FFF 0000 3FFF   98.          DC.W $3FFF,$0000,$3FFF,$0000
         0000
00000308 3FFF 0000 3FFF   99.          DC.W $3FFF,$0000,$3FFF,$0000
         0000
00000310 3FFF 0000 3FFF  100.          DC.W $3FFF,$0000,$3FFF,$0000
         0000
00000318 3FFF 0000 3FFF  101.          DC.W $3FFF,$0000,$3FFF,$0000
         0000
00000320 3FFF 0000 3FFF  102.          DC.W $3FFF,$0000,$3FFF,$0000
         0000
00000328 3FFF 0000 3FFF  103.          DC.W $3FFF,$0000,$3FFF,$0000
         0000
00000330 3FFF 0000 3FFF  104.          DC.W $3FFF,$0000,$3FFF,$0000
         0000
00000338 3FFF 0000 3FFF  105.          DC.W $3FFF,$0000,$3FFF,$0000
         0000
00000340 3FFF 0000 3FFF  106.          DC.W $3FFF,$0000,$3FFF,$0000
         0000
00000348 3FFF 0000 3FFF  107.          DC.W $3FFF,$0000,$3FFF,$0000
         0000
00000350 3FFF 0000 3FFF  108.          DC.W $3FFF,$0000,$3FFF,$0000
         0000
00000358 3FFF 0000 3FFF  109.          DC.W $3FFF,$0000,$3FFF,$0000
         0000
00000360 3FFF 0000 3FFF  110.          DC.W $3FFF,$0000,$3FFF,$0000
         0000
00000368 3FFF 0000 3FFF  111.          DC.W $3FFF,$0000,$3FFF,$0000
         0000
00000370 3FFF 0000 3FFF  112.          DC.W $3FFF,$0000,$3FFF,$0000
         0000
00000378 3FFF 0000 3FFF  113.          DC.W $3FFF,$0000,$3FFF,$0000
         0000
00000380 3FFF 0000 3FFF  114.          DC.W $3FFF,$0000,$3FFF,$0000
         0000
00000388 3FFF 0000 3FFF  115.          DC.W $3FFF,$0000,$3FFF,$0000
         0000
00000390 3FFF 0000 3FFF  116.          DC.W $3FFF,$0000,$3FFF,$0000
         0000
00000398 3FFF 0000 3FFF  117.          DC.W $3FFF,$0000,$3FFF,$0000
         0000
000003A0 3FFF 0000 3FFF  118.          DC.W $3FFF,$0000,$3FFF,$0000
         0000
000003A8 3FFF 0000 3FFF  119.          DC.W $3FFF,$0000,$3FFF,$0000
         0000
000003B0 3FFF 0000 3FFF  120.          DC.W $3FFF,$0000,$3FFF,$0000
         0000
000003B8 3FFF 0000 3FFF  121.          DC.W $3FFF,$0000,$3FFF,$0000
```

```
                0000
000003C0        3FFF 0000 3FFF  122.            DC.W $3FFF,$0000,$3FFF,$0000
                0000
000003C8        3FFF 0000 3FFF  123.            DC.W $3FFF,$0000,$3FFF,$0000
                0000
000003D0        3FFF 0000 3FFF  124.            DC.W $3FFF,$0000,$3FFF,$C000
                0000
000003D8        3FFF C000 3FFF  125.            DC.W $3FFF,$0000,$3FFF,$0000
                0000
000003E0        3FFF 0000 3FFF  126.            DC.W $3FFF,$0000,$3FFF,$0000
                0000
000003E8        3FFF 0000 3FFF  127.            DC.W $3FFF,$0000,$3FFF,$0000
                0000
000003F0        3FFF 0000 3FFF  128.            DC.W $3FFF,$0000,$3FFF,$0000
                0000
000003F8        3FFF 0000 3FFF  129.            DC.W $3FFF,$0000,$3FFF,$0000
                0000
00000400                        130.          END
    0 Errors
```

L.9 Input data: MAXDC

```
00000000                          1.          ORG.L $0000
00000000 D555 0000 D555           2.          DC.W $D555,$0000,$D555,$0000
         0000
00000008 D555 0000 D555           3.          DC.W $D555,$0000,$D555,$0000
         0000
00000010 D555 0000 D555           4.          DC.W $D555,$0000,$D555,$0000
         0000
00000018 D555 0000 D555           5.          DC.W $D555,$0000,$D555,$0000
         0000
00000020 D555 0000 D555           6.          DC.W $D555,$0000,$D555,$0000
         0000
00000028 D555 0000 D555           7.          DC.W $D555,$0000,$D555,$0000
         0000
00000030 D555 0000 D555           8.          DC.W $D555,$0000,$D555,$0000
         0000
00000038 D555 0000 D555           9.          DC.W $D555,$0000,$D555,$0000
         0000
00000040 D555 0000 D555          10.          DC.W $D555,$0000,$D555,$0000
         0000
00000048 D555 0000 D555          11.          DC.W $D555,$0000,$D555,$0000
         0000
00000050 D555 0000 D555          12.          DC.W $D555,$0000,$D555,$0000
         0000
00000058 D555 0000 D555          13.          DC.W $D555,$0000,$D555,$0000
         0000
00000060 D555 0000 D555          14.          DC.W $D555,$0000,$D555,$0000
         0000
00000068 D555 0000 D555          15.          DC.W $D555,$0000,$D555,$0000
         0000
00000070 D555 0000 D555          16.          DC.W $D555,$0000,$D555,$0000
         0000
00000078 D555 0000 D555          17.          DC.W $D555,$0000,$D555,$0000
         0000
00000080 D555 0000 D555          18.          DC.W $D555,$0000,$D555,$0000
         0000
00000088 D555 0000 D555          19.          DC.W $D555,$0000,$D555,$0000
         0000
00000090 D555 0000 D555          20.          DC.W $D555,$0000,$D555,$0000
         0000
00000098 D555 0000 D555          21.          DC.W $D555,$0000,$D555,$0000
         0000
000000A0 D555 0000 D555          22.          DC.W $D555,$0000,$D555,$0000
         0000
000000A8 D555 0000 D555          23.          DC.W $D555,$0000,$D555,$0000
         0000
000000B0 D555 0000 D555          24.          DC.W $D555,$0000,$D555,$0000
         0000
000000B8 D555 0000 D555          25.          DC.W $D555,$0000,$D555,$0000
         0000
000000C0 D555 0000 D555          26.          DC.W $D555,$0000,$D555,$0000
         0000
000000C8 D555 0000 D555          27.          DC.W $D555,$0000,$D555,$0000
         0000
000000D0 D555 0000 D555          28.          DC.W $D555,$0000,$D555,$0000
         0000
000000D8 D555 0000 D555          29.          DC.W $D555,$0000,$D555,$0000
         0000
000000E0 D555 0000 D555          30.          DC.W $D555,$0000,$D555,$0000
         0000
000000E8 D555 0000 D555          31.          DC.W $D555,$0000,$D555,$0000
         0000
000000F0 D555 0000 D555          32.          DC.W $D555,$0000,$D555,$0000
         0000
000000F8 D555 0000 D555          33.          DC.W $D555,$0000,$D555,$0000
         0000
00000100 D555 0000 D555          34.          DC.W $D555,$0000,$D555,$0000
         0000
00000108 D555 0000 D555          35.          DC.W $D555,$0000,$D555,$0000
         0000
00000110 D555 0000 D555          36.          DC.W $D555,$0000,$D555,$0000
         0000
00000118 D555 0000 D555          37.          DC.W $D555,$0000,$D555,$0000
         0000
00000120 D555 0000 D555          38.          DC.W $D555,$0000,$D555,$0000
         0000
00000128 D555 0000 D555          39.          DC.W $D555,$0000,$D555,$0000
         0000
00000130 D555 0000 D555          40.          DC.W $D555,$0000,$D555,$0000
```

```
                0000
00000138  D555 0000 D555    41.          DC.W $D555,$0000,$D555,$0000
                0000
00000140  D555 0000 D555    42.          DC.W $D555,$0000,$D555,$0000
                0000
00000148  D555 0000 D555    43.          DC.W $D555,$0000,$D555,$0000
                0000
00000150  D555 0000 D555    44.          DC.W $D555,$0000,$D555,$0000
                0000
00000158  D555 0000 D555    45.          DC.W $D555,$0000,$D555,$0000
                0000
00000160  D555 0000 D555    46.          DC.W $D555,$0000,$D555,$0000
                0000
00000168  D555 0000 D555    47.          DC.W $D555,$0000,$D555,$0000
                0000
00000170  D555 0000 D555    48.          DC.W $D555,$0000,$D555,$0000
                0000
00000178  D555 0000 D555    49.          DC.W $D555,$0000,$D555,$0000
                0000
00000180  D555 0000 D555    50.          DC.W $D555,$0000,$D555,$0000
                0000
00000188  D555 0000 D555    51           DC.W $D555,$0000,$D555,$0000
                0000
00000190  D555 0000 D555    52.          DC.W $D555,$0000,$D555,$0000
                0000
00000198  D555 0000 D555    53.          DC.W $D555,$0000,$D555,$0000
                0000
000001A0  D555 0000 D555    54.          DC.W $D555,$0000,$D555,$0000
                0000
000001A8  D555 0000 D555    55.          DC.W $D555,$0000,$D555,$0000
                0000
000001B0  D555 0000 D555    56.          DC.W $D555,$0000,$D555,$0000
                0000
000001B8  D555 0000 D555    57.          DC.W $D555,$0000,$D555,$0000
                0000
000001C0  D555 0000 D555    58.          DC.W $D555,$0000,$D555,$0000
                0000
000001C8  D555 0000 D555    59.          DC.W $D555,$0000,$D555,$0000
                0000
000001D0  D555 0000 D555    60.          DC.W $D555,$0000,$D555,$0000
                0000
000001D8  D555 0000 D555    61.          DC.W $D555,$0000,$D555,$0000
                0000
000001E0  D555 0000 D555    62.          DC.W $D555,$0000,$D555,$0000
                0000
000001E8  D555 0000 D555    63.          DC.W $D555,$0000,$D555,$0000
                0000
000001F0  D555 0000 D555    64.          DC.W $D555,$0000,$D555,$0000
                0000
000001F8  D555 0000 D555    65.          DC.W $D555,$0000,$D555,$0000
                0000
00000200  D555 0000 D555    66.          DC.W $D555,$0000,$D555,$0000
                0000
00000208  D555 0000 D555    67.          DC.W $D555,$0000,$D555,$0000
                0000
00000210  D555 0000 D555    68.          DC.W $D555,$0000,$D555,$0000
                0000
00000218  D555 0000 D555    69.          DC.W $D555,$0000,$D555,$0000
                0000
00000220  D555 0000 D555    70.          DC.W $D555,$0000,$D555,$0000
                0000
00000228  D555 0000 D555    71.          DC.W $D555,$0000,$D555,$0000
                0000
00000230  D555 0000 D555    72.          DC.W $D555,$0000,$D555,$0000
                0000
00000238  D555 0000 D555    73.          DC.W $D555,$0000,$D555,$0000
                0000
00000240  D555 0000 D555    74.          DC.W $D555,$0000,$D555,$0000
                0000
00000248  D555 0000 D555    75.          DC.W $D555,$0000,$D555,$0000
                0000
00000250  D555 0000 D555    76.          DC.W $D555,$0000,$D555,$0000
                0000
00000258  D555 0000 D555    77.          DC.W $D555,$0000,$D555,$0000
                0000
00000260  D555 0000 D555    78.          DC.W $D555,$0000,$D555,$0000
                0000
00000268  D555 0000 D555    79.          DC.W $D555,$0000,$D555,$0000
                0000
00000270  D555 0000 D555    80           DC.W $D555,$0000,$D555,$0000
                0000
```

```
00000278  D555 0000 D555    81.          DC.W $D555,$0000,$D555,$0000
          0000
00000280  D555 0000 D555    82.          DC.W $D555,$0000,$D555,$0000
          0000
00000288  D555 0000 D555    83.          DC.W $D555,$0000,$D555,$0000
          0000
00000290  D555 0000 D555    84.          DC.W $D555,$0000,$D555,$0000
          0000
00000298  D555 0000 D555    85.          DC.W $D555,$0000,$D555,$000(
          0000
000002A0  D555 0000 D555    86.          DC.W $D555,$0000,$D555,$0000
          0000
000002A8  D555 0000 D555    87.          DC.W $D555,$0000,$D555,$0000
          0000
000002B0  D555 0000 D555    88.          DC.W $D555,$0000,$D555,$0000
          0000
000002B8  D555 0000 D555    89.          DC.W $D555,$0000,$D555,$0000
          0000
000002C0  D555 0000 D555    90.          DC.W $D555,$0000,$D555,$0000
          0000
000002C8  D555 0000 D555    91.          DC.W $D555,$0000,$D555,$0000
          0000
000002D0  D555 0000 D555    92.          DC.W $D555,$0000,$D555,$0000
          0000
000002D8  D555 0000 D555    93.          DC.W $D555,$0000,$D555,$0000
          0000
000002E0  D555 0000 D555    94.          DC.W $D555,$0000,$D555,$0000
          0000
000002E8  D555 0000 D555    95.          DC.W $D555,$0000,$D555,$0000
          0000
000002F0  D555 0000 D555    96.          DC.W $D555,$0000,$D555,$0000
          0000
000002F8  D555 0000 D555    97.          DC.W $D555,$0000,$D555,$0000
          0000
00000300  D555 0000 D555    98.          DC.W $D555,$0000,$D555,$0000
          0000
00000308  D555 0000 D555    99.          DC.W $D555,$0000,$D555,$0000
          0000
00000310  D555 0000 D555   100.          DC.W $D555,$0000,$D555,$0000
          0000
00000318  D555 0000 D555   101.          DC.W $D555,$0000,$D555,$0000
          0000
00000320  D555 0000 D555   102.          DC.W $D555,$0000,$D555,$0000
          0000
00000328  D555 0000 D555   103.          DC W $D555,$0000,$D555,$0000
          0000
00000330  D555 0000 D555   104.          DC.W $D555,$0000,$D555,$0000
          0000
00000338  D555 0000 D555   105.          DC W $D555,$0000,$D555,$0000
          0000
00000340  D555 0[illegible] D555   106.          DC.W $D555,$0000,$D555,$0000
          0000
00000348  D555 0000 D555   107.          DC.W $D555,$0000,$D555,$0000
          0000
00000350  D555 0000 D555   108.          DC.W $D555,$0000,$D555,$0000
          0000
00000358  D555 0000 D555   109.          DC.W $D555,$0000,$D555,$0000
          0000
00000360  D555 0000 D555   110.          DC.W $D555,$0000,$D555,$0000
          0000
00000368  D555 0000 D555   111.          DC.W $D555,$0000,$D555,$0000
          0000
00000370  D555 0000 D555   112.          DC.W $D555,$0000,$D555,$0000
          0000
00000378  D555 0000 D555   113.          DC.W $D555,$0000,$D555,$0000
          0000
00000380  D555 0000 D555   114.          DC.W $D555,$0000,$D555,$0000
          0000
00000388  D555 0000 D555   115.          DC.W $D555,$0000,$D555,$0000
          0000
00000390  D555 0000 D555   116.          DC.W $D555,$0000,$D555,$0000
          0000
00000398  D555 0000 D555   117.          DC.W $D555,$0000,$D555,$0000
          0000
000003A0  D555 0000 D555   118.          DC.W $D555,$0000,$D555,$0000
          0000
000003A8  D555 0000 D555   119.          DC.W $D555,$0000,$D555,$0000
          0000
000003B0  D555 0000 D555   120.          DC.W $D555,$0000,$D555,$0000
          0000
000003B8  D555 0000 D555   121.          DC.W $D555,$0000,$D555,$0000
```

```
          0000
000003C0  D555 0000 D555  122.          DC.W $D555,$0000,$D555,$0000
          0000
000003C8  D555 0000 D555  123.          DC W $D555,$0000,$D555,$0000
          0000
000003D0  D555 0000 D555  124.          DC.W $D555,$0000,$D555,$0000
          0000
000003D8  D555 0000 D555  125           DC W $D555,$0000,$D555,$0000
          0000
000003E0  D555 0000 D555  126           DC W $D555,$0000,$D555,$0000
          0000
000003E8  D555 0000 D555  127.          DC W $D555,$0000,$D555,$0000
          0000
000003F0  D555 0000 D555  128           DC.W $D555,$0000,$D555,$0000
          0000
000003F8  D555 0000 D555  129           DC W $D555,$0000,$D555,$0000
          0000
00000400                  130           END
    0 Errors
```

L.10 Input data: COS1

```
00000000                          1.          ORG.L $0000
00000000 3FFF 0000 3E14           2.          DC.W $3FFF,$0000,$3E14,$0000
         0000
00000008 3870 0000 2F6B           3.          DC.W $3870,$0000,$2F6B,$0000
         0000
00000010 238D 0000 158F           4.          DC.W $238D,$0000,$158F,$0000
         0000
00000018 0645 0000 F69C           5.          DC.W $0645,$0000,$F69C,$0000
         0000
00000020 E782 0000 D9E0           6.          DC.W $E782,$0000,$D9E0,$0000
         0000
00000028 CE87 0000 C625           7.          DC.W $CE87,$0000,$C625,$0000
         0000
00000030 C13B 0000 C014           8.          DC.W $C13B,$0000,$C014,$0000
         0000
00000038 C2C1 0000 C91B           9.          DC.W $C2C1,$0000,$C91B,$0000
         0000
00000040 D2BF 0000 DF19          10.          DC.W $D2BF,$0000,$DF19,$0000
         0000
00000048 ED6C 0000 FCDC          11.          DC.W $ED6C,$0000,$FCDC,$0000
         0000
00000050 0C7B 0000 1B5C          12.          DC.W $0C7B,$0000,$1B5C,$0000
         0000
00000058 2899 0000 3367          13.          DC.W $2899,$0000,$3367,$0000
         0000
00000060 3B20 0000 3F4E          14           DC.W $3B20,$0000,$3F4E,$0000
         0000
00000068 3FB0 0000 3C41          15.          DC.W $3FB0,$0000,$3C41,$0000
         0000
00000070 3536 0000 2AFA          16           DC.W $3536,$0000,$2AFA,$0000
         0000
00000078 1E2A 0000 0F8C          17           DC.W $1E2A,$0000,$0F8C,$0000
         0000
00000080 C000 0000 F073          18           DC.W $C000,$0000,$F073,$0000
         0000
00000088 E1D5 0000 D505          19.          DC.W $E1D5,$0000,$D505,$0000
         0000
00000090 CAC9 0000 C3BE          20           DC.W $CAC9,$0000,$C3BE,$0000
         0000
00000098 C04F 0000 C0B1          21           DC.W $C04F,$0000,$C0B1,$0000
         0000
000000A0 C4DF 0000 CC98          22           DC.W $C4DF,$0000,$CC98,$0000
         0000
000000A8 D766 0000 E4A3          23           DC.W $D766,$0000,$E4A3,$0000
         0000
000000B0 F384 0000 0323          24.          DC.W $F384,$0000,$0323,$0000
         0000
000000B8 1293 0000 20E6          25           DC.W $1293,$0000,$20E6,$0000
         0000
000000C0 2D40 0000 36E4          26           DC.W $2D40,$0000,$36E4,$0000
         0000
000000C8 3D3E 0000 3FEB          27.          DC.W $3D3E,$0000,$3FEB,$0000
         0000
000000D0 3EC4 0000 39DA          28           DC.W $3EC4,$0000,$39DA,$0000
         0000
000000D8 3178 0000 261F          29.          DC.W $3178,$0000,$261F,$0000
         0000
000000E0 187D 0000 0963          30.          DC.W $187D,$0000,$0963,$0000
         0000
000000E8 F9BA 0000 EA70          31           DC.W $F9BA,$0000,$EA70,$0000
         0000
000000F0 DC72 0000 D094          32.          DC.W $DC72,$0000,$D094,$0000
         0000
000000F8 C78F 0000 C1EB          33           DC.W $C78F,$0000,$C1EB,$0000
         0000
00000100 C000 0000 C1EB          34.          DC.W $C000,$0000,$C1EB,$0000
         0000
00000108 C78F 0000 D094          35           DC.W $C78F,$0000,$D094,$0000
         0000
00000110 DC72 0000 EA70          36.          DC.W $DC72,$0000,$EA70,$0000
         0000
00000118 F9BA 0000 0963          37.          DC.W $F9BA,$0000,$0963,$0000
         0000
00000120 187D 0000 261F          38.          DC.W $187D,$0000,$261F,$0000
         0000
00000128 3178 0000 39DA          39.          DC.W $3178,$0000,$39DA,$0000
         0000
00000130 3EC4 0000 3FEB          40.          DC.W $3EC4,$0000,$3FEB,$0000
```

```
                0000
00000138  3D3E 0000 36E4   41            DC.W $3D3E,$0000,$36E4,$0000
                0000
00000140  2D40 0000 20E6   42.           DC.W $2D40,$0000,$20E6,$0000
                0000
00000148  1293 0000 0323   43.           DC.W $1293,$0000,$0323,$0000
                0000
00000150  F384 0000 E4A3   44.           DC.W $F384,$0000,$E4A3,$0000
                0000
00000158  D766 0000 CC98   45.           DC.W $D766,$0000,$CC98,$0000
                0000
00000160  C4DF 0000 C0B1   46.           DC.W $C4DF,$0000,$C0B1,$0000
                0000
00000168  C04F 0000 C3BE   47.           DC.W $C04F,$0000,$C3BE,$0000
                000(
00000170  CAC9 0000 D505   48.           DC.W $CAC9,$0000,$D505,$0000
                0000
00000178  E1D5 0000 F073   49.           DC.W $E1D5,$0000,$F073,$0000
                0000
00000180  0000 0000 0F8C   50.           DC.W $0000,$0000,$0F8C,$0000
                0000
00000188  1E2A 0000 2AFA   51.           DC.W $1E2A,$0000,$2AFA,$0000
                0000
00000190  3536 0000 3C41   52.           DC.W $3536,$0000,$3C41,$0000
                0000
00000198  3FB0 0000 3F4E   53.           DC.W $3FB0,$0000,$3F4E,$0000
                0000
000001A0  3B20 0000 3367   54.           DC.W $3B20,$0000,$3367,$0000
                0000
000001A8  2899 0000 1B5C   55.           DC.W $2899,$0000,$1B5C,$0000
                0000
000001B0  0C7B 0000 FCDC   56.           DC.W $0C7B,$0000,$FCDC,$0000
                0000
000001B8  ED6C 0000 DF19   57.           DC.W $ED6C,$0000,$DF19,$0000
                0000
000001C0  D2BF 0000 C91B   58.           DC.W $D2BF,$0000,$C91B,$0000
                0000
000001C8  C2C1 0000 C014   59.           DC.W $C2C1,$0000,$C014,$0000
                0000
000001D0  C13B 0000 C625   60.           DC.W $C13B,$0000,$C625,$0000
                0000
000001D8  CE87 0000 D9E0   61.           DC.W $CE87,$0000,$D9E0,$0000
                0000
000001E0  E782 0000 F69C   62.           DC.W $E782,$0000,$F69C,$0000
                0000
000001E8  0645 0000 158F   63.           DC.W $0645,$0000,$158F,$0000
                0000
000001F0  238D 0000 2F6B   64            DC.W $238D,$0000,$2F6B,$0000
                0000
000001F8  3870 0000 3E14   65.           DC.W $3870,$0000,$3E14,$0000
                0000
00000200  3FFF 0000 3E14   66            DC.W $3FFF,$0000,$3E14,$0000
                0000
00000208  3870 0000 2F6B   67.           DC.W $3870,$0000,$2F6B,$0000
                0000
00000210  238D 0000 158F   68.           DC.W $238D,$0000,$158F,$0000
                0000
00000218  0645 0000 F69C   69.           DC.W $0645,$0000,$F69C,$0000
                0000
00000220  E782 0000 D9E0   70            DC.W $E782,$0000,$D9E0,$0000
                0000
00000228  CE87 0000 C625   71            DC.W $CE87,$0000,$C625,$0000
                0000
00000230  C13B 0000 C014   72.           DC.W $C13B,$0000,$C014,$0000
                0000
00000238  C2C1 0000 C91B   73.           DC.W $C2C1,$0000,$C91B,$0000
                0000
00000240  D2BF 0000 DF19   74.           DC.W $D2BF,$0000,$DF19,$0000
                0000
00000248  ED6C 0000 FCDC   75            DC.W $ED6C,$0000,$FCDC,$0000
                0000
00000250  0C7B 0000 1B5C   76.           DC.W $0C7B,$0000,$1B5C,$0000
                0000
00000258  2899 0000 3367   77.           DC.W $2899,$0000,$3367,$0000
                0000
00000260  3B20 0000 3F4E   78.           DC.W $3B20,$0000,$3F4E,$0000
                0000
00000268  3FB0 0000 3C41   79            DC.W $3FB0,$0000,$3C41,$0000
                0000
00000270  3536 0000 2AFA   80.           DC.W $3536,$0000,$2AFA,$0000
                0000
```

```
00000278  1E2A 0000 0F8C   81.          DC.W $1E2A,$0000,$0F8C,$0000
          0000
00000280  0000 0000 F073   82.          DC.W $0000,$0000,$F073,$0000
          0000
00000288  E1D5 0000 D505   83.          DC.W $E1D5,$0000,$D505,$0000
          0000
00000290  CAC9 0000 C3BE   84.          DC.W $CAC9,$0000,$C3BE,$0000
          0000
00000298  C04F 0000 C0B1   85.          DC.W $C04F,$0000,$C0B1,$0000
          0000
000002A0  C4DF 0000 CC98   86.          DC.W $C4DF,$0000,$CC98,$0000
          0000
000002A8  D766 0000 E4A3   87.          DC.W $D766,$0000,$E4A3,$0000
          0000
000002B0  F384 0000 0323   88.          DC.W $F384,$0000,$0323,$0000
          0000
000002B8  1293 0000 20E6   89.          DC.W $1293,$0000,$20E6,$0000
          0000
000002C0  2D40 0000 36E4   90.          DC W $2D40,$0000,$36E4,$0000
          0000
000002C8  3D3E 0000 3FEB   91.          DC.W $3D3E,$0000,$3FEB,$0000
          0000
000002D0  3EC4 0000 39DA   92.          DC W $3EC4,$0000,$39DA,$0000
          0000
000002D8  3178 0000 261F   93.          DC.W $3178,$0000,$261F,$0000
          0000
000002E0  187D 0000 0963   94.          DC W $187D,$0000,$0963,$0000
          0000
000002E8  F9BA 0000 EA70   95.          DC.W $F9BA,$0000,$EA70,$0000
          0000
000002F0  DC72 0000 D094   96.          DC W $DC72,$0000,$D094,$0000
          0000
000002F8  C78F 0000 C1EB   97           DC.W $C78F,$0000,$C1EB,$0000
          0000
00000300  C000 0000 C1EB   98.          DC.W $C000,$0000,$C1EB,$0000
          0000
00000308  C78F 0000 D094   99           DC W $C78F,$0000,$D094,$0000
          0000
00000310  DC72 0000 EA70  100           DC.W $DC72,$0000,$EA70,$0000
          0000
00000318  F9BA 0000 0963  101.          DC.W $F9BA,$0000,$0963,$0000
          0000
00000320  187D 0000 261F  102.          DC W $187D,$0000,$261F,$0000
          0000
00000328  3178 0000 39DA  103           DC W $3178,$0000,$39DA,$0000
          0000
00000330  3EC4 0000 3FEB  104           DC W $3EC4,$0000,$3FEB,$0000
          0000
00000338  3D3E 0000 36E4  105           DC.W $3D3E,$0000,$36E4,$0000
          0000
00000340  2D40 0000 20E6  106.          DC W $2D40,$0000,$20E6,$0000
          0000
00000348  1293 0000 0323  107           DC W $1293,$0000,$0323,$0000
          0000
00000350  F384 0000 E4A3  108.          DC W $F384,$0000,$E4A3,$0000
          0000
00000358  D766 0000 CC98  109.          DC.W $D766,$0000,$CC98,$0000
          0000
00000360  C4DF 0000 C0B1  110           DC W $C4DF,$0000,$C0B1,$0000
          0000
00000368  C04F 0000 C3BE  111.          DC W $C04F,$0000,$C3BE,$0000
          0000
00000370  CAC9 0000 D505  112.          DC W $CAC9,$0000,$D505,$0000
          0000
00000378  E1D5 0000 F073  113.          DC W $E1D5,$0000,$F073,$0000
          0000
00000380  0000 0000 0F8C  114.          DC W $0000,$0000,$0F8C,$0000
          0000
00000388  1E2A 0000 2AFA  115           DC W $1E2A,$0000,$2AFA,$0000
          0000
00000390  3536 0000 3C41  11 .          DC.W $3536,$0000,$3C41,$0000
          0000
00000398  3FB0 0000 3F4E  117.          DC W $3FB0,$0000,$3F4E,$0000
          0000
000003A0  3B20 0000 3367  118           DC.W $3B20,$0000,$3367,$0000
          0000
000003A8  2899 0000 1B5C  119.          DC.W $2899,$0000,$1B5C,$0000
          0000
000003B0  0C7B 0000 FCDC  120           DC.W $0C7B,$0000,$FCDC,$0000
          0000
000003B8  ED6C 0000 DF19  121.          DC W $ED6C,$0000,$DF19,$0000
```

```
                0000
000003C0        D2BF 0000 C91B  122.            DC.W $D2BF,$0000,$C91B,$0000
                0000
000003C8        C2C1 0000 C014  123.            DC.W $C2C1,$0000,$C014,$0000
                0000
000003D0        C13B 0000 C625  124.            DC.W $C13B,$0000,$C625,$0000
                0000
000003D8        CE87 0000 D9E0  125.            DC.W $CE87,$0000,$D9E0,$0000
                0000
000003E0        E782 0000 F69C  126.            DC.W $E782,$0000,$F69C,$0000
                0000
000003E8        0645 0000 158F  127.            DC.W $0645,$0000,$158F,$0000
                0000
000003F0        238D 0000 2F6B  128.            DC.W $238D,$0000,$2F6B,$0000
                0000
000003F8        3870 0000 3E14  129.            DC.W $3870,$0000,$3E14,$0000
                0000
    0 Errors
```

L.11 Input data: COS8

```
00000000                          1.          ORG.L $0000
00000000 FFFF 0000 0963           2.          DC.W $FFFF,$0000,$0963,$0000
         0000
00000008 00FE 0000 F574           3.          DC.W $00FE,$0000,$F574,$0000
         0000
00000010 0C64 0000 FC4B           4.          DC.W $0C64,$0000,$FC4B,$0000
         0000
00000018 0418 0000 F79A           5.          DC.W $0418,$0000,$F79A,$0000
         0000
00000020 0690 0000 FF76           6.          DC.W $0690,$0000,$FF76,$0000
         0000
00000028 FDCF 0000 FA92           7.          DC.W $FDCF,$0000,$FA92,$0000
         0000
00000030 FEB6 0000 F64B           8.          DC.W $FEB6,$0000,$F64B,$0000
         0000
00000038 F5E6 0000 FF44           9.          DC.W $F5E6,$0000,$FF44,$0000
         0000
00000040 0837 0000 F02A          10.          DC.W $0837,$0000,$F02A,$0000
         0000
00000048 FECA 0000 0EFF          11.          DC.W $FECA,$0000,$0EFF,$0000
         0000
00000050 0C28 0000 0E03          12.          DC.W $0C28,$0000,$0E03,$0000
         0000
00000058 0D80 0000 0D72          13.          DC.W $0D80,$0000,$0D72,$0000
         0000
00000060 0D59 0000 022B          14.          DC.W $0D59,$0000,$022B,$0000
         0000
00000068 0737 0000 0F30          15.          DC.W $0737,$0000,$0F30,$0000
         0000
00000070 F4FD 0000 F7F1          16.          DC.W $F4FD,$0000,$F7F1,$0000
         0000
00000078 F3B1 0000 073F          17.          DC.W $F3B1,$0000,$073F,$0000
         0000
00000080 0000 0000 0FE5          18.          DC.W $0000,$0000,$0FE5,$0000
         0000
00000088 F3B1 0000 0CD7          19.          DC.W $F3B1,$0000,$0CD7,$0000
         0000
00000090 0089 0000 FAE0          20.          DC.W $0089,$0000,$FAE0,$0000
         0000
00000098 0737 0000 FF84          21.          DC.W $0737,$0000,$FF84,$0000
         0000
000000A0 0A94 0000 F558          22.          DC.W $0A94,$0000,$F558,$0000
         0000
000000A8 0D80 0000 F921          23           DC.W $0D80,$0000,$F921,$0000
         0000
000000B0 0D8B 0000 F3BF          24.          DC.W $0D8B,$0000,$F3BF,$0000
         0000
000000B8 FECA 0000 FE7B          25.          DC.W $FECA,$0000,$FE7B,$0000
         0000
000000C0 F407 0000 0E6A          26.          DC.W $F407,$0000,$0E6A,$0000
         0000
000000C8 F5E6 0000 0236          27.          DC.W $F5E6,$0000,$0236,$0000
         0000
000000D0 06A8 0000 01F9          28.          DC.W $06A8,$0000,$01F9,$0000
         0000
000000D8 FDCF 0000 00CF          29.          DC.W $FDCF,$0000,$00CF,$0000
         0000
000000E0 06E4 0000 0854          30           DC.W $06E4,$0000,$0854,$0000
         0000
000000E8 0418 0000 FFC9          31.          DC.W $0418,$0000,$FFC9,$0000
         0000
000000F0 FF54 0000 00D5          32.          DC.W $FF54,$0000,$00D5,$0000
         0000
000000F8 00FE 0000 FFFD          33.          DC.W $00FE,$0000,$FFFD,$0000
         0000
00000100 0000 0000 00FD          34.          DC.W $0000,$0000,$00FD,$0000
         0000
00000108 00FE 0000 0006          35.          DC.W $00FE,$0000,$0006,$0000
         0000
00000110 008D 0000 0315          36.          DC.W $008D,$0000,$0315,$0000
         0000
00000118 0418 0000 009A          37.          DC.W $0418,$0000,$009A,$0000
         0000
00000120 FB5C 0000 FA27          38.          DC.W $FB5C,$0000,$FA27,$0000
         0000
00000128 FDCF 0000 0033          39.          DC.W $FDCF,$0000,$0033,$0000
         0000
00000130 FC56 0000 F60B          40.          DC.W $FC56,$0000,$F60B,$0000
```

```
                0000
00000138  F5E6  0000  FDC1    41.          DC.W $F5E6,$0000,$FDC1,$0000
                0000
00000140  0558  0000  04A8    42.          DC.W $0558,$0000,$04A8,$0000
                0000
00000148  FECA  0000  FE35    43.          DC.W $FECA,$0000,$FE35,$0000
                0000
00000150  06A0  0000  0F31    44           DC.W $06A0,$0000,$0F31,$0000
                0000
00000158  0D80  0000  0367    45.          DC.W $0D80,$0000,$0367,$0000
                0000
00000160  FCFA  0000  00CA    46.          DC.W $FCFA,$0000,$00CA,$0000
                0000
00000168  0737  0000  0482    47.          DC.W $0737,$0000,$0482,$0000
                0000
00000170  F8B9  0000  F02C    48.          DC.W $F8B9,$0000,$F02C,$0000
                0000
00000178  F3B1  0000  FD3D    49           DC.W $F3B1,$0000,$FD3D,$0000
                0000
00000180  0000  0000  F93D    50.          DC.W $0000,$0000,$F93D,$0000
                0000
00000188  F3B1  0000  F92E    51           DC.W $F3B1,$0000,$F92E,$0000
                0000
00000190  0581  0000  0BD8    52.          DC.W $0581,$0000,$0BD8,$0000
                0000
00000198  0737  0000  005A    53.          DC.W $0737,$0000,$005A,$0000
                0000
000001A0  01B7  0000  09C6    54.          DC.W $01B7,$0000,$09C6,$0000
                0000
000001A8  0D80  0000  0711    55.          DC.W $0D80,$0000,$0711,$0000
                0000
000001B0  FD3B  0000  FA38    56.          DC.W $FD3B,$0000,$FA38,$0000
                0000
000001B8  FECA  0000  0239    57.          DC.W $FECA,$0000,$0239,$0000
                0000
000001C0  FE69  0000  F784    58           DC.W $FE69,$0000,$F784,$0000
                0000
000001C8  F5E6  0000  FB26    59.          DC.W $F5E6,$0000,$FB26,$0000
                0000
000001D0  00BB  0000  00F7    60.          DC.W $00BB,$0000,$00F7,$0000
                0000
000001D8  FDCF  0000  FD1D    61.          DC.W $FDCF,$0000,$FD1D,$0000
                0000
000001E0  008E  0000  0421    62.          DC.W $008E,$0000,$0421,$0000
                0000
000001E8  0418  0000  0189    63           DC.W $0418,$0000,$0189,$0000
                0000
000001F0  FFF8  0000  006A    64           DC.W $FFF8,$0000,$006A,$0000
                0000
000001F8  00FE  0000  007E    65           DC.W $00FE,$0000,$007E,$0000
                0000
00000200  0000  0000  007E    66           DC.W $0000,$0000,$007E,$0000
                0000
00000208  00FE  0000  006A    67           DC.W $00FE,$0000,$006A,$0000
                0000
00000210  FFF8  0000  0188    68           DC.W $FFF8,$0000,$0188,$0000
                0000
00000218  0418  0000  0421    69.          DC.W $0418,$0000,$0421,$0000
                0000
00000220  008E  0000  FD1D    70.          DC.W $008E,$0000,$FD1D,$0000
                0000
00000228  FDCF  0000  00F7    71           DC.W $FDCF,$0000,$00F7,$0000
                0000
00000230  00BB  0000  FB26    72           DC.W $00BB,$0000,$FB26,$0000
                0000
00000238  F5E6  0000  F784    73           DC.W $F5E6,$0000,$F784,$0000
                0000
00000240  FE69  0000  0239    74           DC.W $FE69,$0000,$0239,$0000
                0000
00000248  FECA  0000  FA38    75           DC.W $FECA,$0000,$FA38,$0000
                0000
00000250  FD3B  0000  0711    76.          DC.W $FD3B,$0000,$0711,$0000
                0000
00000258  0D80  0000  09C5    77           DC.W $0D80,$0000,$09C5,$0000
                0000
00000260  01B7  0000  005A    78.          DC.W $01B7,$0000,$005A,$0000
                0000
00000268  0737  0000  0BD8    79.          DC.W $0737,$0000,$0BD8,$0000
                0000
00000270  0581  0000  F92E    80           DC.W $0581,$0000,$F92E,$0000
                0000
```

```
00000278  F3B1 0000 F93D    81.          DC.W $F3B1,$0000,$F93D,$0000
          0000
00000280  0000 0000 FD3D    82.          DC.W $0000,$0000,$FD3D,$0000
          0000
00000288  F3B1 0000 F02C    83.          DC.W $F3B1,$0000,$F02C,$0000
          0000
00000290  F8B9 0000 0482    84.          DC.W $F8B9,$0000,$0482,$0000
          0000
00000298  0737 0000 00CA    85.          DC.W $0737,$0000,$00CA,$0000
          0000
000002A0  FCFA 0000 0367    86.          DC.W $FCFA,$0000,$0367,$0000
          0000
000002A8  0D80 0000 0F31    87.          DC.W $0D80,$0000,$0F31,$0000
          0000
000002B0  06A0 0000 FE35    88.          DC.W $06A0,$0000,$FE35,$0000
          0000
000002B8  FECA 0000 04A8    89.          DC.W $FECA,$0000,$04A8,$0000
          0000
000002C0  0558 0000 FDC1    90.          DC.W $0558,$0000,$FDC1,$0000
          0000
000002C8  F5E6 0000 F60B    91.          DC.W $F5E6,$0000,$F60B,$0000
          0000
000002D0  FC56 0000 0033    92.          DC.W $FC56,$0000,$0033,$0000
          0000
000002D8  FDD0 0000 FA27    93.          DC.W $FDD0,$0000,$FA27,$0000
          0000
000002E0  FB5C 0000 009A    94.          DC.W $FB5C,$0000,$009A,$0000
          0000
000002E8  0418 0000 0315    95.          DC.W $0418,$0000,$0315,$0000
          0000
000002F0  008C 0000 0006    96.          DC.W $008C,$0000,$0006,$0000
          0000
000002F8  00FE 0000 00FD    97           DC.W $00FE,$0000,$00FD,$0000
          0000
00000300  0000 0000 FFFD    98           DC.W $0000,$0000,$FFFD,$0000
          0000
00000308  00FE 0000 00D5    99           DC.W $00FE,$0000,$00D5,$0000
          0000
00000310  FF54 0000 FFC9   100.          DC.W $FF54,$0000,$FFC9,$0000
          0000
00000318  0418 0000 0854   101.          DC.W $0418,$0000,$0854,$0000
          0000
00000320  06E4 0000 00CE   102           DC.W $06E4,$0000,$00CE,$0000
          0000
00000328  FDCF 0000 01F9   103           DC.W $FDCF,$0000,$01F9,$0000
          0000
00000330  06A8 0000 0236   104           DC.W $06A8,$0000,$0236,$0000
          0000
00000338  F5E6 0000 0E6A   105           DC.W $F5E6,$0000,$0E6A,$0000
          0000
00000340  F407 0000 FE7C   106           DC.W $F407,$0000,$FE7C,$0000
          0000
00000348  FEC9 0000 F3BF   107.          DC.W $FEC9,$0000,$F3BF,$0000
          0000
00000350  0D8B 0000 F921   108.          DC.W $0D8B,$0000,$F921,$0000
          0000
00000358  0D80 0000 F558   109.          DC.W $0D80,$0000,$F558,$0000
          0000
00000360  0A94 0000 FF84   110.          DC.W $0A94,$0000,$FF84,$0000
          0000
00000368  0737 0000 FAE0   111.          DC.W $0737,$0000,$FAE0,$0000
          0000
00000370  0089 0000 0CD7   112.          DC.W $0089,$0000,$0CD7,$0000
          0000
00000378  F3B1 0000 0FE5   113           DC.W $F3B1,$0000,$0FE5,$0000
          0000
00000380  0000 0000 073F   114           DC.W $0000,$0000,$073F,$0000
          0000
00000388  F3B1 0000 F7F1   115.          DC.W $F3B1,$0000,$F7F1,$0000
          0000
00000390  F4FD 0000 0F30   116.          DC.W $F4FD,$0000,$0F30,$0000
          0000
00000398  0737 0000 022B   117.          DC.W $0737,$0000,$022B,$0000
          0000
000003A0  0D59 0000 0D72   118.          DC.W $0D59,$0000,$0D72,$0000
          0000
000003A8  0D80 0000 0E03   119.          DC.W $0D80,$0000,$0E03,$0000
          0000
000003B0  0C28 0000 0EFF   120           DC.W $0C28,$0000,$0EFF,$0000
          0000
000003B8  FECA 0000 F02A   121.          DC.W $FECA,$0000,$F02A,$0000
```

```
          0000
000003C0  0837 0000 FF44  122.          DC.W $0837,$0000,$FF44,$0000
          0000
000003C8  F5E6 0000 F64B  123.          DC.W $F5E6,$0000,$F64B,$0000
          0000
000003D0  FEB6 0000 FA92  124.          DC.W $FEB6,$0000,$FA92,$0000
          0000
000003D8  FDCF 0000 FF76  125.          DC.W $FDCF,$0000,$FF76,$0000
          0000
000003E0  0690 0000 F79A  126.          DC.W $0690,$0000,$F79A,$0000
          0000
000003E8  0418 0000 FC4B  127.          DC.W $0418,$0000,$FC4B,$0000
          0000
000003F0  0C64 0000 F574  128.          DC.W $0C64,$0000,$F574,$0000
          0000
000003F8  00FE 0000 0963  129.          DC.W $00FE,$0000,$0963,$0000
          0000
    0 Errors
```

L.12 Input data: NOISE

```
00000000                          1.          ORG.L $0000
00000000  C36A 0000 DA8F          2.          DC.W $C36A,$0000,$DA8F,$0000
          0000
00000008  DB22 0000 C8D5          3.          DC.W $DB22,$0000,$C8D5,$0000
          0000
00000010  2D2A 0000 C4D0          4           DC.W $2D2A,$0000,$C4D0,$0000
          0000
00000018  0921 0000 CED7          5.          DC.W $0921,$0000,$CED7,$0000
          0000
00000020  E869 0000 3383          6.          DC.W $E869,$0000,$3383,$0000
          0000
00000028  C2F5 0000 D9D0          7.          DC.W $C2F5,$0000,$D9D0,$0000
          0000
00000030  3F57 0000 E36C          8.          DC.W $3F57,$0000,$E36C,$0000
          0000
00000038  C063 0000 3848          9.          DC.W $C063,$0000,$3848,$0000
          0000
00000040  24EF 0000 1FE6         10.          DC.W $24EF,$0000,$1FE6,$0000
          0000
00000048  F26D 0000 DE03         11.          DC.W $F26D,$0000,$DE03,$0000
          0000
00000050  2971 0000 3583         12.          DC.W $2971,$0000,$3583,$0000
          0000
00000058  3B6B 0000 0BCB         13.          DC.W $3B6B,$0000,$0BCB,$0000
          0000
00000060  1313 0000 F7F3         14.          DC.W $1313,$0000,$F7F3,$0000
          0000
00000068  ED5D 0000 2470         15.          DC.W $ED5D,$0000,$2470,$0000
          0000
00000070  EEF3 0000 DF27         16.          DC W $EEF3,$0000,$DF27,$0000
          0000
00000078  0280 0000 D30C         17           DC.W $0280,$0000,$D30C,$0000
          0000
00000080  2E51 0000 FFB5         18.          DC.W $2E51,$0000,$FFB5,$0000
          0000
00000088  C41F 0000 E68B         19.          DC.W $C41F,$0000,$E68B,$0000
          0000
00000090  23FB 0000 E27F         20.          DC.W $23FB,$0000,$E27F,$0000
          0000
00000098  D3AC 0000 066B         21.          DC.W $D3AC,$0000,$066B,$0000
          0000
000000A0  F608 0000 FF90         22.          DC.W $F608,$0000,$FF90,$0000
          0000
000000A8  CA0F 0000 24C1         23.          DC.W $CA0F,$0000,$24C1,$0000
          0000
000000B0  D7C9 0000 E046         24.          DC.W $D7C9,$0000,$E046,$0000
          0000
000000B8  E13E 0000 D17D         25.          DC.W $E13E,$0000,$D17D,$0000
          0000
000000C0  1FFD 0000 2CB6         26.          DC.W $1FFD,$0000,$2CB6,$0000
          0000
000000C8  DC0E 0000 D2EE         27.          DC.W $DC0E,$0000,$D2EE,$0000
          0000
000000D0  0F44 0000 284C         28.          DC.W $0F44,$0000,$284C,$0000
          0000
000000D8  2049 0000 2789         29.          DC.W $2049,$0000,$2789,$0000
          0000
000000E0  DAD7 0000 E1B2         30.          DC W $DAD7,$0000,$E1B2,$0000
          0000
000000E8  3700 0000 D4E6         31.          DC.W $3700,$0000,$D4E6,$0000
          0000
000000F0  CF75 0000 19F3         32.          DC.W $CF75,$0000,$19F3,$0000
          0000
000000F8  2722 0000 280B         33.          DC.W $2722,$0000,$280B,$0000
          0000
00000100  20A1 0000 26E1         34.          DC.W $20A1,$0000,$26E1,$0000
          0000
00000108  DE4E 0000 CAED         35.          DC.W $DE4E,$0000,$CAED,$0000
          0000
00000110  380A 0000 D4B4         36.          DC.W $380A,$0000,$D4B4,$0000
          0000
00000118  1BE6 0000 D334         37.          DC.W $1BE6,$0000,$D334,$0000
          0000
00000120  194B 0000 0AF3         38.          DC.W $194B,$0000,$0AF3,$0000
          0000
00000128  1265 0000 2E3F         39.          DC.W $1265,$0000,$2E3F,$0000
          0000
00000130  2DD4 0000 3898         40.          DC W $2DD4,$0000,$3898,$0000
```

```
                0000
00000138  32F2  0000 0E68    41.          DC.W $32F2,$0000,$0E68,$0000
                0000
00000140  0D28  0000 CB6F    42.          DC.W $0D28,$0000,$CB6F,$0000
                0000
00000148  D735  0000 3D89    43.          DC.W $D735,$0000,$3D89,$0000
                0000
00000150  154A  0000 36BF    44.          DC.W $154A,$0000,$36BF,$0000
                0000
00000158  3D69  0000 F8BF    45.          DC.W $3D69,$0000,$F8BF,$0000
                0000
00000160  E771  0000 CD2C    46.          DC.W $E771,$0000,$CD2C,$0000
                0000
00000168  0AB2  0000 396D    47.          DC.W $0AB2,$0000,$396D,$0000
                0000
00000170  1D03  0000 F1DC    48.          DC.W $1D03,$0000,$F1DC,$0000
                0000
00000178  C7B1  0000 0F84    49.          DC.W $C7B1,$0000,$0F84,$0000
                0000
00000180  C8C1  0000 E4D9    50.          DC.W $C8C1,$0000,$E4D9,$0000
                0000
00000188  0B56  0000 F102    51.          DC.W $0B56,$0000,$F102,$0000
                0000
00000190  183F  0000 2EB6    52.          DC.W $183F,$0000,$2EB6,$0000
                0000
00000198  C7F6  0000 22B9    53.          DC.W $C7F6,$0000,$22B9,$0000
                0000
000001A0  2998  0000 EF76    54.          DC.W $2998,$0000,$EF76,$0000
                0000
000001A8  EE9E  0000 1E52    55.          DC.W $EE9E,$0000,$1E52,$0000
                0000
000001B0  E95F  0000 14A9    56.          DC.W $E95F,$0000,$14A9,$0000
                0000
000001B8  DCA6  0000 1991    57           DC.W $DCA6,$0000,$1991,$0000
                0000
000001C0  0CD9  0000 3AD8    58.          DC.W $0CD9,$0000,$3AD8,$0000
                0000
000001C8  C789  0000 12DB    59.          DC.W $C789,$0000,$12DB,$0000
                0000
000001D0  FC69  0000 3E23    60.          DC.W $FC69,$0000,$3E23,$0000
                0000
000001D8  1AF4  0000 06F5    61.          DC.W $1AF4,$0000,$06F5,$0000
                0000
000001E0  C24B  0000 3E29    62.          DC.W $C24B,$0000,$3E29,$0000
                0000
000001E8  FD1C  0000 340E    63.          DC.W $FD1C,$0000,$340E,$0000
                0000
000001F0  D185  0000 1928    64.          DC.W $D185,$0000,$1928,$0000
                0000
000001F8  ED55  0000 0E00    65.          DC.W $ED55,$0000,$0E00,$0000
                0000
00000200  391B  0000 2265    66.          DC.W $391B,$0000,$2265,$0000
                0000
00000208  30E1  0000 C7D3    67.          DC.W $30E1,$0000,$C7D3,$0000
                0000
00000210  1783  0000 17CF    68.          DC.W $1783,$0000,$17CF,$0000
                0000
00000218  0205  0000 1682    69.          DC.W $0205,$0000,$1682,$0000
                0000
00000220  E258  0000 1ADF    70.          DC.W $E258,$0000,$1ADF,$0000
                0000
00000228  3562  0000 1102    71           DC.W $3562,$0000,$1102,$0000
                0000
00000230  D653  0000 30C2    72.          DC.W $D653,$0000,$30C2,$0000
                0000
00000238  C983  0000 D181    73.          DC.W $C983,$0000,$D181,$0000
                0000
00000240  2376  0000 0DBA    74.          DC.W $2376,$0000,$0DBA,$0000
                0000
00000248  14B3  0000 3BEC    75.          DC.W $14B3,$0000,$3BEC,$0000
                0000
00000250  298A  0000 2FC2    76.          DC.W $298A,$0000,$2FC2,$0000
                0000
00000258  0511  0000 0DB3    77.          DC.W $0511,$0000,$0DB3,$0000
                0000
00000260  D8CC  0000 0C71    78.          DC.W $D8CC,$0000,$0C71,$0000
                0000
00000268  26E6  0000 1AD0    79.          DC.W $26E6,$0000,$1AD0,$0000
                0000
00000270  0AE4  0000 D620    80.          DC.W $0AE4,$0000,$D620,$0000
                0000
```

```
00000278  E535 0000 DC06   81.          DC.W $E535,$0000,$DC06,$0000
          0000
00000280  E816 0000 1C4F   82.          DC.W $E816,$0000,$1C4F,$0000
          0000
00000288  3894 0000 3266   83.          DC.W $3894,$0000,$3266,$0000
          0000
00000290  2CBC 0000 CB45   84.          DC.W $2CBC,$0000,$CB45,$0000
          0000
00000298  383B 0000 0418   85.          DC.W $383B,$0000,$0418,$0000
          0000
000002A0  E2F4 0000 CEF7   86.          DC.W $E2F4,$0000,$CEF7,$0000
          0000
000002A8  C15B 0000 1658   87.          DC.W $C15B,$0000,$1658,$0000
          0000
000002B0  E496 0000 DD2B   88.          DC.W $E496,$0000,$DD2B,$0000
          0000
000002B8  28B1 0000 C8BF   89.          DC.W $28B1,$0000,$C8BF,$0000
          0000
000002C0  3969 0000 2ADD   90.          DC.W $3969,$0000,$2ADD,$0000
          0000
000002C8  2A5A 0000 15C4   91.          DC.W $2A5A,$0000,$15C4,$0000
          0000
000002D0  2596 0000 10E2   92.          DC.W $2596,$0000,$10E2,$0000
          0000
000002D8  0BE9 0000 FC82   93.          DC.W $0BE9,$0000,$FC82,$0000
          0000
000002E0  FC5C 0000 E3CB   94.          DC.W $FC5C,$0000,$E3CB,$0000
          0000
000002E8  24B7 0000 37BC   95.          DC.W $24B7,$0000,$37BC,$0000
          0000
000002F0  0B06 0000 030C   96.          DC.W $0B06,$0000,$030C,$0000
          0000
000002F8  C07A 0000 E61E   97.          DC.W $C07A,$0000,$E61E,$0000
          0000
00000300  301A 0000 E35E   98.          DC.W $301A,$0000,$E35E,$0000
          0000
00000308  1019 0000 07C5   99.          DC.W $1019,$0000,$07C5,$0000
          0000
00000310  F2D4 0000 1861  100           DC.W $F2D4,$0000,$1861,$0000
          0000
00000318  1CBF 0000 F502  101           DC.W $1CBF,$0000,$F502,$0000
          0000
00000320  2ED2 0000 2186  102.          DC.W $2ED2,$0000,$2186,$0000
          0000
00000328  F12F 0000 325B  103.          DC.W $F12F,$0000,$325B,$0000
          0000
00000330  2811 0000 FE2C  104           DC.W $2811,$0000,$FE2C,$0000
          0000
00000338  ED57 0000 C5D3  105.          DC.W $ED57,$0000,$C5D3,$0000
          0000
00000340  1B18 0000 CD09  106.          DC.W $1B18,$0000,$CD09,$0000
          0000
00000348  F827 0000 F16C  107.          DC.W $F827,$0000,$F16C,$0000
          0000
00000350  1D73 0000 FACC  108           DC.W $1D73,$0000,$FACC,$0000
          0000
00000358  03A3 0000 F6E9  109.          DC.W $03A3,$0000,$F6E9,$0000
          0000
00000360  E263 0000 C400  110           DC.W $E263,$0000,$C400,$0000
          0000
00000368  1737 0000 C8D8  111.          DC.W $1737,$0000,$C8D8,$0000
          0000
00000370  37D4 0000 0E34  112.          DC.W $37D4,$0000,$0E34,$0000
          0000
00000378  D44B 0000 CCD0  113.          DC.W $D44B,$0000,$CCD0,$0000
          0000
00000380  CF8F 0000 DC59  114.          DC.W $CF8F,$0000,$DC59,$0000
          0000
00000388  2918 0000 12F8  115.          DC.W $2918,$0000,$12F8,$0000
          0000
00000390  28B1 0000 E26B  116.          DC.W $28B1,$0000,$E26B,$0000
          0000
00000398  25BA 0000 26C8  117.          DC.W $25BA,$0000,$26C8,$0000
          0000
000003A0  2D5C 0000 FC54  118.          DC.W $2D5C,$0000,$FC54,$0000
          0000
000003A8  2786 0000 DD14  119           DC.W $2786,$0000,$DD14,$0000
          0000
000003B0  D8AC 0000 0C0D  120.          DC.W $D8AC,$0000,$0C0D,$0000
          0000
000003B8  CE9D 0000 C346  121.          DC.W $CE9D,$0000,$C346,$0000
```

```
                0000
000003C0   F8ED 0000 0306  122.          DC.W $F8ED,$0000,$0306,$0000
                0000
000003C8   F1C1 0000 13EB  123.          DC.W $F1C1,$0000,$13EB,$0000
                0000
000003D0   E20B 0000 1AC8  124.          DC.W $E20B,$0000,$1AC8,$0000
                0000
000003D8   0468 0000 D471  125.          DC.W $0468,$0000,$D471,$0000
                0000
000003E0   2449 0000 00D9  126.          DC.W $2449,$0000,$00D9,$0000
                0000
000003E8   230F 0000 002E  127.          DC.W $230F,$0000,$002E,$0000
                0000
000003F0   1B36 0000 F9DF  128.          DC.W $1B36,$0000,$F9DF,$0000
                0000
000003F8   39D1 0000 E4A5  129.          DC.W $39D1,$0000,$E4A5,$0000
                0000
00000400                   130.         END
    0 Errors
```

Appendix M: Frequency Domain Filtering Performance Test Program Code and Data

This appendix presents programs and data for the frequency domain filtering performance tests. For the sake of brevity, some of the test programs are not shown, in particular, those which measure intermediate phases of execution (Forward FFT, Filter time, Inverse FFT, etc.). The programs are similar to those presented herein, with differences only in placement of the timer enable/disable instructions. Input signal data for the tests are shown in Appendix K: NOISE and NULLDC.

The multicomputer master programs (PFILNOM1-PFILNOM4) utilize a common set of slave programs; only the master program differs. The master programs are linked to the slave program segment, SLVNOMES.

The multicomputer master programs (FSTFL1NM-FSTFL4NM) utilize a common set of slave programs; only the master program differs. The master programs are linked to the slave program segment, FSTSLVNM.

Filter data is shown. Prior to all test programs, the program FILTER must be executed to distribute the filter coefficients to the slave memories.

M.1 Program SNGFNOMS: Uniprocessor Frequency Domain Filtering Program

```
                                1.          TTL SNGFNOMS
                                2.  * 256 POINT FILTER TEST ROUTINE
                                3.  *
                                4.  ** AUG13,1991
                                5.  *
                                6.  * INPUT DATA AT $1000 TO $13FF
                                7.  * OUTPUT DATA AT $1400 $17FF
                                8.  * COMPLEX FILTER DATA AT  $1800 TO $1AFF
                                9.  *
                               10.  * COPIES DATA TO $1400 & FILTERING
                               11.  * IS PERFORMED ON COPIED DATA. RESULTS
                               12.  * RETURNED AT $1400
                               13.  * AUGUST 19,1991 ( USES NEGATE IN INVFFT
                               14.  * INSTEAD OF NEW SIN TABLE; FFT HAS SMALL
                               15.  * FIX IN REGISTER USAGE;
                               16.  *
                               17.  * SEPT 6 1991 REMOVE PHASE MEASURING CODE
                               18.  * JUST MEASURE OVERALL TIME.
                               19.  *
     # 00000100                 20.  NUMPT EQU 256
     # 00000008                 21   NUGAM EQU 8
     # 00012FD0                 22.  TCR     EQU $12FD0
     # 00000001                 23.  ENBLETM EQU 1
     # 00FFFFFF                 24.  CNTSTRT EQU $00FFFFFF
     # 0000000A                 25.  SENCLFC EQU 10
     # 00000018                 26.  DISBUF8 EQU $18
     # 0000FA00                 27.  USTACK  EQU $FA00
     # 00000001                 28.  USST8   EQU 1
     # 00000002                 29.  SUPST8  EQU 2
     # 00000004                 30.  EOT     EQU 04
     # 00000011                 31.  OUTMESC      EQU 17
     # 0000000E                 32.  SHIFT14 EQU 14
                               33.  *
                               34.  *
00001000                       35.          ORG.L $1000
00001000  <400>                36.  INDATA  DS.B $400
00001400  <400>                37.  OUTDATA DS.B $400
00001800  <400>                38.  FILTER  DS.B $400
00001CC0  43FA F3FE            39.          LEA INDATA(PC),A1    ;INPUT POINTER
00001CC4  45FA F7FA            40.          LEA OUTDATA(PC),A2   ;OUTPUT POINTER
00001CC8  203C 000000FF        41.          MOVE.L #NUMPT-1,D0 ;LOOP COUNTER
00001CCE  24D9                 42.  MOVELP  MOVE.L (A1)+,(A2)+
00001CD0  51C8 FFFC            43.          DBRA D0,MOVELP
00001CD4  41FA F7EA            44.          LEA OUTDATA(PC),A0   ;POINT FOR FFT
00001CD8  6100 0068            45.          BSR FILPRG           ;DO FILTERING
                               46.  *
                               47.  * PUT USER STACK POINTER INTO A2 TO GET
                               48.  * THE DATA OFF THE USER STACK
                               49.  *
00001C1C  4E6A                 50.          MOVE.L USP,A2
00001C1E  241A                 51.          MOVE.L (A2)+,D2
                               52.  *
                               53.  * D2 HAS FILTERING TIME
```

```
                                54.  * CLEAN UP USER STACK POINTER
                                55.  *
00001C20  4E62                  56.          MOVE.L A2,USP
00001C22  2202                  57.          MOVE.L D2,D1     ;OVERALL TIME
00001C24  61 08                 58.          BSR.S TIMCALC
00001C26  41FA 0026             59.          LEA OVRMES(PC),A0
00001C2A  61 12                 60.          BSR.S OUTWRIT
00001C2C  4E75                  61.          RTS
                                62.  *
                                63.  * SUBROUTINE TIMCALC
                                64.  * D1 HAS TIMER REGISTER COUNT, RETURNS
                                65.  * TOTAL COUNT FROM TIMER START TO FINISH
                                66.  * IN D1.L, OTHER REG'S TRANSPARENT
                                67.  *
00001C2E  2F00                  68.  TIMCALC MOVE.L D0,-(SP)
00001C30  2001                  69.          MOVE.L D1,D0
00001C32  223C 00FFFFFF         70.          MOVE.L #CNTSTRT,D1
00001C38  9280                  71.          SUB.L D0,D1
00001C3A  201F                  72.          MOVE.L (SP)+,D0
00001C3C  4E75                  73.          RTS
                                74.  *
                                75.  * SUBROUTINE OUTWRIT
                                76.  * A0 POINTS TO A MESSAGE, D1 CONTAINS A
                                77.  * NUMBER TO BE OUTPUT AT END
                                78.  *
00001C3E  4E4F                  79.  OUTWRIT TRAP #15
00001C40  0011                  80.          DC.W OUTMESC
00001C42  2001                  81.          MOVE.L D1,D0
00001C44  4E4F                  82.          TRAP #15
00001C46  0018                  83.          DC.W DISBUF8
00001C48  4E4F                  84.          TRAP #15
00001C4A  000A                  85.          DC.W SENCLFC
00001C4C  4E75                  86.          RTS              ;RETURN
                                87.  *
00001C4E  4F 76 65 72 61  88.  OVRMES  DC.B 'Overall filtering time   ( x 4 for microseconds)=
',EOT
          6C 6C 20 66 69
          6C 74 65 72 69
          6E 67 20 74 69
          6D 65 20 20 20
          28 20 78 20 34
          20 66 6F 72 20
          6D 69 63 72 6F
          73 65 63 6F 6E
          64 73 29 3D 20
          04
                                89.  *
                                90.  * DO FORWARD FFT
                                91.  * FIRST GOTO user STATE
                                92.  *
          -- boundary align
00001C82  4E41                  93.  FILPRG  TRAP #USST8      ;USER STATE
00001C84  2E7C 0000FA00         94.          MOVEA.L #USTACK,SP  ;USER STACK PNTER
                                95.  *
                                96.  * START TIMING FORWARD FFT
                                97.  *
00001C8A  4239 00012FD0         98.          CLR.B TCR
00001C90  23FC 00FFFFFF         99.          MOVE.L #CNTSTRT,TCR+2 ;LOAD INITIAL
          00012FD2
00001C9A  13FC 0001            100.          MOVE.B #ENBLETM,TCR   ;START TIME
          00012FD0
00001CA2  283C 000000EF        101.          MOVE.L #NUMPT-17,D4        ;COUNTER=K
00001CA8  3C04                 102.  SWAPLPF MOVE.W D4,D6          ;PREPARE BIT REV
00001CAA  3E06                 103.          MOVE.W D6,D7        ;SAVE DATA
00001CAC  4246                 104.          CLR.W D6            ;
00001CAE  7A07                 105.          MOVEQ.L #7,D5       ;BIT COUNTER
00001CB0  E257                 106.  REVLPF  ROXR.W #1,D7        ;SFT RIGHT,XTEND
00001CB2  E356                 107.          ROXL.W #1,D6
00001CB4  51CD FFFA            108.          DBRA D5,REVLPF
                               109.  *
                               110.  * DONE REVERSAL CONTINUE WITH UNSHUFFLE
                               111.  *
00001CB8  B846                 112.          CMP.W D6,D4           ;I=REV(K)<=K?
00001CBA  6C 14                113.          BGE.S DECKF            ;NO SWAP IF SO
00001CBC  3204                 114.          MOVE.W D4,D1          ;COPY OF K
00001CBE  E541                 115.          ASL.W #2,D1           ;K*4
00001CC0  E546                 116.          ASL.W #2,D6           ;I*4
00001CC2  2430 1000            117.          MOVE.L 0(A0,D1.W),D2 ;DO SWAP
00001CC6  21B0 6000 1000       118.          MOVE.L 0(A0,D6.W),0(A0,D1.W)
00001CCC  2182 6000            119.          MOVE.L D2,0(A0,D6.W) ;DONE SWAP
00001CD0  51CC FFD6            120.  DECKF   DBRA D4,SWAPLPF
```

```
00001CD4  7201          121.  FFTF    MOVEQ.L #1,D1   ;D1=N2=1
00001CD6  43FA 019C     122.          LEA SINBLK(PC),A1  ;SIN VALS PNTR A1
00001CDA  45FA 0398     123.          LEA COSBLK(PC),A2  ;COS VALS PNTR A2
                        124.  *
                        125.  * L IS DOWN COUNTED #NUGAM-1 TO 0
                        126.  *
00001CDE  3F3C 0001     127.          MOVE.W #1,-(SP)  ;(SP) IS L
00001CE2  7407          128           MOVEQ.L #NUGAM-1,D2        ;D2=NUGAM-1=NU1
00001CE4  4[illegible]  129.          CLR.L D3              ;D3=K=0
00001C[illegible]       130.          MOVEQ.L #SHIFT14,D0
                        131.  *
                        132.  * INITIALIZE I=N2-1, TO 0 IN STEPS -1
                        133.  *
00001CE[illegible]      134.  INITIF  MOVE.W D1,D4     ;D4=I=N2
00001CEA [illegible]    135.          SUBQ.W #1,D4     ;SET FOR DBRA
                        136.  *
                        137.  * DETERMINE P=MOD(K*2**(NUGAM-L),256)
                        138.  *
00001CEC  3C03          139.  LOOPINF MOVE.W D3,D6     ;D6=K
00001CEE  3A17          140.          MOVE.W (SP),D5   ;GET L
00001CF0  5145          141.          SUBQ.W #NUGAM,D5 ;L-NUGAM
00001CF2  4445          142.          NEG.W D5         ;NUGAM-L
00001CF4  ED66          143.          ASL.W D5,D6      ;K*2**(NUGAM-L)
00001CF6  0246 00FF     144.          ANDI.W #$00FF,D6 ;MOD 256
00001CFA  DC46          145.          ADD.W D6,D6      ;D6*2 FOR WORD BOUND
00001CFC  3A71 6000     146.          MOVEA.W 0(A1,D6.W),A5 ;A5 PNTR IM(W**P)
00001D00  2E03          147.          MOVE.L D3,D7     ;D7=K
00001D02  2A03          148.          MOVE.L D3,D5     ;D5 TOO
00001D04  E545          149.          ASL.W #2,D5      ;K*4 FOR LWORD BOUND
00001D06  4DF0 5000     150.          LEA 0(A0,D5.W),A6   ;A6 IS INDEX X(K)
                        151.  *                          ;A6 PNT TO RE(X(K))
                        152.  *                          ;A6+2 TO IM(X(K))
00001D0A  DE41          153.          ADD.W D1,D7      ;D7=K+N2
00001D0C  E547          154.          ASL.W #2,D7      ;D7=4(K+N2) LWORD BND
00001D0E  47F0 7000     155.          LEA 0(A0,D7.W),A3 ;A3 PNT RE(X(K+N2))
                        156.  *                          ;A3+2 PNT IM(X(K+N2)
                        157.  *
                        158.  * CALCULATE W**P*X(K+N2)
                        159.  * NOTE COMPLEX
                        160.  *
00001D12  3A32 6000     161.          MOVE.W 0(A2,D6.W),D5    ;GET RE(W**P)
00001D16  3E05          162.          MOVE.W D5,D7     ;PUT ALSO IN D7
00001D18  CBD3          163.          MULS (A3),D5     ;RE(W**P)*RE(X(K+N2))
00001D1A  3C0D          164.          MOVE.W A5,D6  ;GET IM(W**P)
00001D1C  CDEB 0002     165.          MULS 2(A3),D6    ;IM(W**P)*IM(X(K+N2))
00001D20  9A86          166.          SUB.L D6,D5      ;D5=RE(T1)
00001D22  E0A5          167.          ASR.L D0,D5      ;/16384 TO SCALE
00001D24  3C0D          168.          MOVE.W A5,D6  ;GET IM(W**P)
00001D26  CDD3          169.          MULS (A3),D6     ;IM(W**P)*RE(X(K+N2))
00001D28  CFEB 0002     170.          MULS 2(A3),D7    ;RE(W**P)*IM(X(K+N2))
00001D2C  DC87          171.          ADD.L D7,D6      ;D6=IM(T1)
00001D2E  E0A6          172.          ASR.L D0,D6      ;/16384 TO SCALE
                        173.  *
                        174.  * D6 = IM(T1)
                        175.  * D5 = RE(T1)
                        176.  * D7 IS FREE
                        177.  * DO X(K+N2) = X(K) - T1
                        [illegible]   *    X(K)    = X(K) + T1
                        [illegible]   * AND DIVIDE ANSWERS BY 2 FOR SCALING
                        180.  *
00001D30  3E16          181.          MOVE.W (A6),D7   ;RE(X(K))
00001D32  9E45          182.          SUB.W D5,D7      ;RE(X(K))-RE(T1)
00001D34  E247          183.          ASR.W #1,D7      ;/2
00001D36  3687          184.          MOVE.W D7,(A3)   ;STORE ANSWER
00001D38  3E2E 0002     185.          MOVE.W 2(A6),D7  ;IM(X(K))
00001D3C  9E46          186.          SUB.W D6,D7      ;IM(X(K))-IM(T1)
00001D3E  E247          187.          ASR.W #1,D7      ;/2
00001D40  3747 0002     188.          MOVE.W D7,2(A3)  ;STORE ANSWER
00001D44  DB56          189.          ADD.W D5,(A6)    ;RE(X(K))+RE(T1)
00001D46  E0D6          190.          ASR.W (A6)       ;/2
00001D48  DD6E 0002     191.          ADD.W D6,2(A6)   ;IM(X(K))+IM(T1)
00001D4C  E0EE 0002     192.          ASR.W 2(A6)      ;/2
                        193.  *
                        194.  * MATH IS DONE, NOW THE LOOPS
                        195.  *
00001D50  5243          196.          ADDQ.W #1,D3     ;INCREMENT K
00001D52  51CC FF98     197.          DBRA D4,LOOPINF  ;DO TILL I=-1
00001D56  D641          198.          ADD.W D1,D3      ;K=K+N2
00001D58  0C43 00FF     199.          CMPI.W #NUMPT-1,D3 ;K<N-1 ?
00001D5C  6D 8A         200.          BLT INITIF       ;DO AGAIN IF SO
                        201.  *
```

```
                          202.  *
00001D5E  4243            203.          CLR.W D3          ;K=0
00001D60  5342            204.          SUBQ.W #1,D2      ;NU1=NU1-1
00001D62  E341            205.          ASL.W #1,D1       ;N2=N2*2
00001D64  5257            206.          ADDQ.W #1,(SP)
00001D66  0C57 0009       207.          CMPI.W #NUGAM+1,(SP)
00001D6A  6600 FF7C       208.          BNE INITIF
00001D6E  301F            209.          MOVE.W (SP)+,D0
                          210.  *
                          211.  * DO COMPLEX MULTIPLY BY THE FILTER.
                          212.  * FILTER CAN BE +16384 TO -16384 WITH 16384
                          213.  * BEING THE 0 db GAIN VALUE.
                          214.  * (A+iB)*(C+iD)
                          215.  * A0 STILL POINTS TO OUTDATA
                          216.  *
00001D70  43FA FA8E       217.          LEA FILTER(PC),A1        ;A1 PNT FILTER
00001D74  303C 00FF       218.          MOVE.W #NUMPT-1,D0       ;LOOP COUNTER
00001D78  7A0E            219.          MOVEQ.L #SHIFT14,D5              ;SHIFT COUNT
00001D7A  3219            220.  FILOOP  MOVE.W (A1)+,D1          ;GET C
00001D7C  3601            221.          MOVE.W D1,D3
00001D7E  C3D8            222.          MULS (A0)+,D1            ;DO AC
00001D80  3419            223.          MOVE.W (A1)+,D2          ;GET D
00001D82  3802            224.          MOVE.W D2,D4
00001D84  C5D0            225.          MULS (A0),D2             ;DO BD
00001D86  9282            226.          SUB.L D2,D1              ;AC-BD
00001D88  EAA1            227.          ASR.L D5,D1              ;ADJUST /16384
00001D8A  C7D0            228.          MULS (A0),D3             ;DO BC
00001D8C  C9E0            229.          MULS -(A0),D4            ;DO AD
00001D8E  D883            230.          ADD.L D3,D4              ;BC+AD
00001D90  EAA4            231.          ASR.L D5,D4              ;ADJUST /16384
00001D92  30C1            232.          MOVE.W D1,(A0)+ ;STORE REAL
00001D94  30C4            233.          MOVE.W D4,(A0)+ ;STORE IMAG
00001D96  51C8 FFE2       234.          DBRA D0,FILOOP
                          235.  *
                          236.  * NOW DO INVERSE FFT ON THE NEW DATA
                          237.  * IN PLACE. HERE, THE ASR.L #2 FOR
                          238.  * OVERALL DIVISION BY NUMPT IS ELIMINATED
                          239.  * USES NEGATED SINBLK AND COSBLK DATA
00001D9A  41FA F664       240.  FFTINV  LEA OUTDATA(PC),A0
00001D9E  283C 000000EF   241.          MOVE.L #NUMPT-17,D4          ;COUNTER=K
00001DA4  3C04            242.  SWAPLPR MOVE.W D4,D6           ;PREPARE BIT RE"
00001DA6  3E06            243.          MOVE.W D6,D7         ;SAVE DATA
00001DA8  4246            244.          CLR.W D6             ;
00001DAA  7A07            245.          MOVEQ.L #7,D5        ;BIT COUNTER
00001DAC  E257            246.  REVLPR  ROXR.W #1,D7         ;SFT RIGHT,XTEND
00001DAE  E356            247.          ROXL.W #1,D6
00001DB0  51CD FFFA       248.          DBRA D5,REVLPR
                          249.  *
                          250.  * DONE REVERSAL CONTINUE WITH UNSHUFFLE
                          251.  *
00001DB4  B846            252.          CMP.W D6,D4              ;I=REV(K)<=K?
00001DB6  6C 14           253.          BGE.S DECKR               ;NO SWAP IF SO
00001DB8  3204            254.          MOVE.W D4,D1             ;COPY OF K
00001DBA  E541            255.          ASL.W #2,D1              ;K*4
00001DBC  E546            256.          ASL.W #2,D6              ;I*4
00001DBE  2430 1000       257.          MOVE.L 0(A0,D1.W),D2 ;DO SWAP
00001DC2  21B0 6000 1000  258.          MOVE.L 0(A0,D6.W),0(A0,D1.W)
00001DC8  2182 6000       259.          MOVE.L D2,0(A0,D6.W) ;DONE SWAP
00001DCC  51CC FFD6       260.  DECKR   DBRA D4,SWAPLPR
00001DD0  7201            261.  FFTR    MOVEQ.L #1,D1    ;D1=N2=1
00001DD2  43FA 00A0       262.          LEA SINBLK(PC),A1  ;SIN VALS PNTR A1
00001DD6  45FA 029C       263.          LEA COSBLK(PC),A2  ;COS VALS PNTR A2
                          264.  *
                          265.  * L IS DOWN COUNTED #NUGAM-1 TO 0
                          266.  *
00001DDA  3F3C 0001       267.          MOVE.W #1,-(SP)  ;(SP) IS L
00001DDE  7407            268.          MOVEQ.L #NUGAM-1,D2        ;D2=NUGAM-1=NU1
00001DE0  4283            269.          CLR.L D3                ;D3=K=0
00001DE2  700E            270.          MOVEQ.L #SHIFT14,D0
                          271.  *
                          272.  * INITIALIZE I=N2-1, TO 0 IN STEPS -1
                          273.  *
00001DE4  3801            274.  INITIR  MOVE.W D1,D4     ;D4=I=N2
00001DE6  5344            275.          SUBQ.W #1,D4     ;SET FOR DBRA
                          276.  *
                          277.  * DETERMINE P=MOD(K*2**(NUGAM-L),256)
                          278.  *
00001DE8  3C03            279.  LOOPINR MOVE.W D3,D6     ;D6=K
00001DEA  3A17            280.          MOVE.W (SP),D5   ;GET L
00001DEC  5145            281.          SUBQ.W #NUGAM,D5 ;L-NUGAM
00001DEE  4445            282.          NEG.W D5         ;NUGAM-L
```

```
00001DF0  EB66            283.          ASL.W D5,D6      ;K*2**(NUGAM-L)
00001DF2  0246 00FF       284.          ANDI.W #$00FF,D6 ;MOD 256
00001DF6  DC46            285.          ADD.W D6,D6      ;D6*2 FOR WORD BOUND
00001DF8  3A71 6000       286.          MOVEA.W 0(A1,D6.W),A5 ;A5 PNTR IM(W**P)
00001DFC  2E03            287.          MOVE.L D3,D7     ;D7=K
00001DFE  2A03            288.          MOVE.L D3,D5     ;D5 TOO
00001E00  E545            289.          ASL.W #2,D5      ;K*4 FOR LWORD BOUND
00001E02  4DF0 5000       290.          LEA 0(A0,D5.W),A6   ;A6 IS INDEX X(K)
                          291.  *                          ;A6 PNT TO RE(X(K))
                          292.  *                          ;A6+2 TO IM(X(K))
00001E06  DE41            293.          ADD.W D1,D7      ;D7=K+N2
00001E08  E547            294.          ASL.W #2,D7      ;D7=4(K+N2) LWORD BND
00001E0A  47F0 7000       295.          LEA 0(A0,D7.W),A3 ;A3 PNT RE(X(K+N2))
                          296.  *                          ;A3+2 PNT IM(X(K+N2)
                          297.  *
                          298.  * CALCULATE W**P*X(K+N2)
                          299.  * NOTE COMPLEX
                          300.  *
00001E0E  3A32 6000       301.          MOVE.W 0(A2,D6.W),D5    ;GET RE(W**P)
00001E12  3E05            302.          MOVE.W D5,D7     ;PUT ALSO IN D7
00001E14  CBD3            303.          MULS (A3),D5     ;RE(W**P)*RE(X(K+N2))
00001E16  3C0D            304.          MOVE.W A5,D6  ;GET IM(W**P)
                          305.  *
                          306.  * NEGATE FOR INVERSE
                          307.  *
00001E18  4446            308.          NEG.W D6
00001E1A  CDEB 0002       309.          MULS 2(A3),D6    ;IM(W**P)*IM(X(K+N2))
00001E1E  9A86            310.          SUB.L D6,D5      ;D5=RE(T1)
00001E20  E0A5            311.          ASR.L D0,D5      ;/16384 TO SCALE
00001E22  3C0D            312.          MOVE.W A5,D6  ;GET IM(W**P)
                          313.  *
                          314.  * NEGATE FOR INVERSE
                          315.  *
00001E24  4446            316.          NEG.W D6
00001E26  CDD3            317.          MULS (A3),D6     ;IM(W**P)*RE(X(K+N2))
00001E28  CFEB 0002       318.          MULS 2(A3),D7    ;RE(W**P)*IM(X(K+N2))
00001E2C  DC87            319.          ADD.L D7,D6      ;D6=IM(T1)
00001E2E  E0A6            320.          ASR.L D0,D6      ;/16384 TO SCALE
                          321.  *
                          322.  * D6 = IM(T1)
                          323.  * D5 = RE(T1)
                          324.  * D7 IS FREE
                          325.  * DO X(K+N2) = X(K) - T1
                          326.  *    X(K)    = X(K) + T1
                          327.  *
00001E30  3E16            328.          MOVE.W (A6),D7   ;RE(X(K))
00001E32  9E45            329.          SUB.W D5,D7      ;RE(X(K))-RE(T1)
00001E34  3687            330.          MOVE.W D7,(A3)   ;STORE ANSWER
00001E36  3E2E 0002       331.          MOVE.W 2(A6),D7  ;IM(X(K))
00001E3A  9E46            332.          SUB.W D6,D7      ;IM(X(K))-IM(T1)
00001E3C  3747 0002       333.          MOVE.W D7,2(A3)  ;STORE ANSWER
00001E40  DB56            334.          ADD.W D5,(A6)    ;RE(X(K))+RE(T1)
00001E42  DD6E 0002       335.          ADD.W D6,2(A6)   ;IM(X(K))+IM(T1)
                          336.  *
                          337.  * MATH IS DONE, NOW THE LOOPS
                          338.  *
00001E46  5243            339.          ADDQ.W #1,D3     ;INCREMENT K
00001E48  51CC FF9E       340.          DBRA D4,LOOPINR  ;DO TILL I=-1
00001E4C  D641            341.          ADD.W D1,D3      ;K=K+N2
00001E4E  0C43 00FF       342.          CMPI.W #NUMPT-1,D3 ;K<N-1 ?
00001E52  6D 90           343.          BLT INITIR       ;DO AGAIN IF SO
                          344.  *
                          345.  *
00001E54  4243            346.          CLR.W D3         ;K=0
00001E56  5342            347.          SUBQ.W #1,D2     ;NU1=NU1-1
00001E58  E341            348.          ASL.W #1,D1      ;N2=N2*2
00001E5A  5257            349.          ADDQ.W #1,(SP)
00001E5C  0C57 0009       350.          CMPI.W #NUGAM+1,(SP)
00001E60  66 82           351.          BNE INITIR
00001E62  301F            352.          MOVE.W (SP)+,D0
                          353.  *
                          354.  * NOW overall filter ROUTINE IS ALL DONE!
                          355.  * GET TIME ONTO USER STACK
                          356.  *
00001E64  4239 00012FD0   357.          CLR.B TCR
00001E6A  2F39 00012FD6   358.          MOVE.L TCR+6,-(SP)
                          359.  *
                          360.  * NOW GO BACK TO THE SUPERVISOR STATE
                          361.  * TO GIVE CORRECT RETURN ADDRESS
                          362.  *
00001E70  4E42            363.          TRAP #SUPST8
```

```
                                    364.  *
                                    365.  * RETURN TO CALLING ROUTINE
                                    366.  *
00001E72  4E75                      367.          RTS
                                    368.  *
                                    369.  * DATA SECTION
                                    370.  *
                                    371.  * SIN (-2*PI*I/256)*16384 FOR I=0 TO 255
                                    372.  *
00001E74  0000 FE6E FCDC FB4B       373.  SINBLK  DC.W       0,  -402,  -804, -1205
00001E7C  F9BA F82A F69C F50F       374.          DC.W   -1606, -2006, -2404, -2801
00001E84  F384 F1FA F073 EEEE       375.          DC.W   -3196, -3590, -3981, -4370
00001E8C  ED6C EBED EA70 E8F7       376.          DC.W   -4756, -5139, -5520, -5897
00001E94  E782 E611 E4A3 E33A       377.          DC.W   -6270, -6639, -7005, -7366
00001E9C  E1D5 E074 DF19 DDC3       378.          DC.W   -7723, -8076, -8423, -8765
00001EA4  DC72 DB26 D9E0 D8A0       379.          DC.W   -9102, -9434, -9760,-10080
00001EAC  D766 D632 D505 D3DF       380.          DC.W  -10394,-10702,-11003,-11297
00001EB4  D2BF D1A6 D094 CF8A       381.          DC.W  -11585,-11866,-12140,-12406
00001EBC  CE87 CD8C CC98 CBAD       382.          DC.W  -12665,-12916,-13160,-13395
00001EC4  CAC9 C9EE C91B C850       383.          DC.W  -13623,-13842,-14053,-14256
00001ECC  C78F C6D5 C625 C57E       384.          DC.W  -14449,-14635,-14811,-14978
00001ED4  C4DF C44A C3BE C33B       385.          DC.W  -15137,-15286,-15426,-15557
00001EDC  C2C1 C251 C1EB C18E       386.          DC.W  -15679,-15791,-15893,-15986
00001EE4  C13B C0F1 C0B1 C07B       387.          DC.W  -16069,-16143,-16207,-16261
00001EEC  C04F C02C C014 C005       388.          DC.W  -16305,-16340,-16364,-16379
00001EF4  C000 C005 C014 C02C       389.          DC.W  -16384,-16379,-16364,-16340
00001EFC  C04F C07B C0B1 C0F1       390.          DC.W  -16305,-16261,-16207,-16143
00001F04  C13B C18E C1EB C251       391.          DC.W  -16069,-15986,-15893,-15791
00001F0C  C2C1 C33B C3BE C44A       392.          DC.W  -15679,-15557,-15426,-15286
00001F14  C4DF C57E C625 C6D5       393.          DC.W  -15137,-14978,-14811,-14635
00001F1C  C78F C850 C91B C9EE       394.          DC.W  -14449,-14256,-14053,-13842
00001F24  CAC9 CBAD CC98 CD8C       395.          DC.W  -13623,-13395,-13160,-12916
00001F2C  CE87 CF8A D094 D1A6       396.          DC.W  -12665,-12406,-12140,-11866
00001F34  D2BF D3DF D505 D632       397.          DC.W  -11585,-11297,-11003,-10702
00001F3C  D766 D8A0 D9E0 DB26       398.          DC.W  -10394,-10080, -9760, -9434
00001F44  DC72 DDC3 DF19 E074       399.          DC.W   -9102, -8765, -8423, -8076
00001F4C  E1D5 E33A E4A3 E611       400.          DC.W   -7723, -7366, -7005, -6639
00001F54  E782 E8F7 EA70 EBED       401.          DC.W   -6270, -5897, -5520, -5139
00001F5C  ED6C EEEE F073 F1FA       402.          DC.W   -4756, -4370, -3981, -3590
00001F64  F384 F50F F69C F82A       403.          DC.W   -3196, -2801, -2404, -2006
00001F6C  F9BA FB4B FCDC FE6E       404.          DC.W   -1606, -1205,  -804,  -402
00001F74  0000 0192 0324 04B5       405.          DC.W       0,   402,   804,  1205
00001F7C  0646 07D6 0964 0AF1       406.          DC.W    1606,  2006,  2404,  2801
00001F84  0C7C 0E06 0F8D 1112       407.          DC.W    3196,  3590,  3981,  4370
00001F8C  1294 1413 1590 1709       408.          DC.W    4756,  5139,  5520,  5897
```

```
00001F94 187E 19EF 1B5D 1CC6  409.          DC.W    6270,  6639,  7005,  7366
00001F9C 1E2B 1F8C 20E7 223D  410.          DC.W    7723,  8076,  8423,  8765
00001FA4 238E 24DA 2620 2760  411.          DC.W    9102,  9434,  9760, 10080
00001FAC 289A 29CE 2AFB 2C21  412.          DC.W   10394, 10702, 11003, 11297
00001FB4 2D41 2E5A 2F6C 3076  413.          DC.W   11585, 11866, 12140, 12406
00001FBC 3179 3274 3368 3453  414.          DC.W   12665, 12916, 13160, 13395
00001FC4 3537 3612 36E5 37B0  415.          DC.W   13623, 13842, 14053, 14256
00001FCC 3871 392B 39DB 3A82  416.          DC.W   14449, 14635, 14811, 14978
00001FD4 3B21 3BB6 3C42 3CC5  417.          DC.W   15137, 15286, 15426, 15557
00001FDC 3D3F 3DAF 3E15 3E72  418.          DC.W   15679, 15791, 15893, 15986
00001FE4 3EC5 3F0F 3F4F 3F85  419.          DC.W   16069, 16143, 16207, 16261
00001FEC 3FB1 3FD4 3FEC 3FFB  420.          DC.W   16305, 16340, 16364, 16379
00001FF4 4000 3FFB 3FEC 3FD4  421.          DC.W   16384, 16379, 16364, 16340
00001FFC 3FB1 3F85 3F4F 3F0F  422.          DC.W   16305, 16261, 16207, 16143
00002004 3EC5 3E72 3E15 3DAF  423.          DC.W   16069, 15986, 15893, 15791
0000200C 3D3F 3CC5 3C42 3BB6  424.          DC.W   15679, 15557, 15426, 15286
00002014 3B21 3A82 39DB 392B  425.          DC.W   15137, 14978, 14811, 14635
0000201C 3871 37B0 36E5 3612  426.          DC.W   14449, 14256, 14053, 13842
00002024 3537 3453 3368 3274  427.          DC.W   13623, 13395, 13160, 12916
0000202C 3179 3076 2F6C 2E5A  428.          DC.W   12665, 12406, 12140, 11866
00002034 2D41 2C21 2AFB 29CE  429.          DC.W   11585, 11297, 11003, 10702
0000203C 289A 2760 2620 24DA  430.          DC.W   10394, 10080,  9760,  9434
00002044 238E 223D 20E7 1F8C  431.          DC.W    9102,  8765,  8423,  8076
0000204C 1E2B 1CC6 1B5D 19EF  432.          DC.W    7723,  7366,  7005,  6639
00002054 187E 1709 1590 1413  433.          DC.W    6270,  5897,  5520,  5139
0000205C 1294 1112 0F8D 0E06  434.          DC.W    4756,  4370,  3981,  3590
00002064 0C7C 0AF1 0964 07D6  435.          DC.W    3196,  2801,  2404,  2006
0000206C 0646 04B5 0324 0192  436.          DC.W    1606,  1205,   804,   402
                              437.  *
                              438.  * COS(-2*PI*I/256)*16384 FOR I=0 TO 255
                              439.  *
00002074 4000 3FFB 3FEC 3FD4  440.  COSBLK  DC.W   16384, 16379, 16364, 16340
0000207C 3FB1 3F85 3F4F 3F0F  441.          DC.W   16305, 16261, 16207, 16143
00002084 3EC5 3E72 3E15 3DAF  442.          DC.W   16069, 15986, 15893, 15791
0000208C 3D3F 3CC5 3C42 3BB6  443.          DC.W   15679, 15557, 15426, 15286
00002094 3B21 3A82 39DB 392B  444.          DC.W   15137, 14978, 14811, 14635
0000209C 3871 37B0 36E5 3612  445.          DC.W   14449, 14256, 14053, 13842
000020A4 3537 3453 3368 3274  446.          DC.W   13623, 13395, 13160, 12916
000020AC 3179 3076 2F6C 2E5A  447.          DC.W   12665, 12406, 12140, 11866
000020B4 2D41 2C21 2AFB 29CE  448.          DC.W   11585, 11297, 11003, 10702
000020BC 289A 2760 2620 24DA  449.          DC.W   10394, 10080,  9760,  9434
000020C4 238E 223D 20E7 1F8C  450.          DC.W    9102,  8765,  8423,  8076
```

```
000020CC 1E2B 1CC6 1B5D 451.          DC.W    7723,  7366,  7005,  6639
         19EF
000020D4 187E 1709 1590 452.          DC.W    6270,  5897,  5520,  5139
         1413
000020DC 1294 1112 0F8D 453.          DC.W    4756,  4370,  3981,  3590
         0E06
000020E4 0C7C 0AF1 0964 454.          DC.W    3196,  2801,  2404,  2006
         07D6
000020EC 0646 04B5 0324 455.          DC.W    1606,  1205,   804,   402
         0192
000020F4 0000 FE6E FCDC 456.          DC.W       0,  -402,  -804, -1205
         FB4B
000020FC F9BA F82A F69C 457.          DC.W   -1606, -2006, -2404, -2801
         F50F
00002104 F384 F1FA F073 458.          DC.W   -3196, -3590, -3981, -4370
         EEEE
0000210C ED6C EBED EA70 459.          DC.W   -4756, -5139, -5520, -5897
         E8F7
00002114 E782 E611 E4A3 460.          DC.W   -6270, -6639, -7005, -7366
         E33A
0000211C E1D5 E074 DF19 461.          DC.W   -7723, -8076, -8423, -8765
         DDC3
00002124 DC72 DB26 D9E0 462.          DC.W   -9102, -9434, -9760,-10080
         D8A0
0000212C D766 D632 D505 463.          DC.W  -10394,-10702,-11003,-11297
         D3DF
00002134 D2BF D1A6 D094 464.          DC.W  -11585,-11866,-12140,-12406
         CF8A
0000213C CE87 CD8C CC98 465.          DC.W  -12665,-12916,-13160,-13395
         CBAD
00002144 CAC9 C9EE C91B 466.          DC.W  -13623,-13842,-14053,-14256
         C850
0000214C C78F C6D5 C625 467.          DC.W  -14449,-14635,-14811,-14978
         C57E
00002154 C4DF C44A C3BE 468.          DC.W  -15137,-15286,-15426,-15557
         C33B
0000215C C2C1 C251 C1EB 469.          DC.W  -15679,-15791,-15893,-15986
         C18E
00002164 C13B C0F1 C0B1 470.          DC.W  -16069,-16143,-16207,-16261
         C07B
0000216C C04F C02C C014 471.          DC.W  -16305,-16340,-16364,-16379
         C005
00002174 C000 C005 C014 472.          DC.W  -16384,-16379,-16364,-16340
         C02C
0000217C C04F C07B C0B1 473.          DC.W  -16305,-16261,-16207,-16143
         C0F1
00002184 C13B C18E C1EB 474.          DC.W  -16069,-15986,-15893,-15791
         C251
0000218C C2C1 C33B C3BE 475.          DC.W  -15679,-15557,-15426,-15286
         C44A
00002194 C4DF C57E C625 476.          DC.W  -15137,-14978,-14811,-14635
         C6D5
0000219C C78F C850 C91B 477.          DC.W  -14449,-14256,-14053,-13842
         C9EE
000021A4 CAC9 CBAD CC98 478.          DC.W  -13623,-13395,-13160,-12916
         CD8C
000021AC CE87 CF8A D094 479.          DC.W  -12665,-12406,-12140,-11866
         D1A6
000021B4 D2BF D3DF D505 480.          DC.W  -11585,-11297,-11003,-10702
         D632
000021BC D766 D8A0 D9E0 481.          DC.W  -10394,-10080, -9760, -9434
         DB26
000021C4 DC72 DDC3 DF19 482.          DC W   -9102, -8765, -8423, -8076
         E074
000021CC E1D5 E33A E4A3 483.          DC.W   -7723, -7366, -7005, -6639
         E611
000021D4 E782 E8F7 EA70 484.          DC.W   -6270, -5897, -5520, -5139
         EBED
000021DC ED6C EEEE F073 485.          DC.W   -4756, -4370, -3981, -3590
         F1FA
000021E4 F384 F50F F69C 486.          DC.W   -3196, -2801, -2404, -2006
         F82A
000021EC F9BA FB4B FCDC 487.          DC.W   -1606, -1205.  -804,  -402
         FE6E
000021F4 0000 0192 0324 488.          DC.W       0,   402,   804,  1205
         04B5
000021FC 0646 07D6 0964 489.          DC.W    1606,  2006,  2404,  2801
         0AF1
```

```
00002204  0C7C 0E06 0F8D  490.          DC.W    3196,  3590,  3981,  4370
          1112
0000220C  1294 1413 1590  491.          DC.W    4756,  5139,  5520,  5897
          1709
00002214  187E 19EF 1B5D  492.          DC.W    6270,  6639,  7005,  7366
          1CC6
0000221C  1E2B 1F8C 20E7  493.          DC.W    7723,  8076,  8423,  8765
          223D
00002224  238E 24DA 2620  494.          DC.W    9102,  9434,  9760, 10080
          2760
0000222C  289A 29CE 2AFB  495.          DC.W   10394, 10702, 11003, 11297
          2C21
00002234  2D41 2E5A 2F6C  496.          DC.W   11585, 11866, 12140, 12406
          3076
0000223C  3179 3274 3368  497.          DC.W   12665, 12916, 13160, 13395
          3453
00002244  3537 3612 36E5  498.          DC.W   13623, 13842, 14053, 14256
          37B0
0000224C  3871 392B 39DB  499.          DC.W   14449, 14635, 14811, 14978
          3A82
00002254  3B21 3BB6 3C42  500.          DC.W   15137, 15286, 15426, 15557
          3CC5
0000225C  3D3F 3DAF 3E15  501.          DC.W   15679, 15791, 15893, 15986
          3E72
00002264  3EC5 3F0F 3F4F  502.          DC.W   16069, 16143, 16207, 16261
          3F85
0000226C  3FB1 3FD4 3FEC  503.          DC.W   16305, 16340, 16364, 16379
          3FFB
00002274                  504.          END
    0 Errors
```

M.2 Program FILTER: Multicomputer Frequency Domain Filtering Program: distributes filter coefficients to slave memory

```
                                    1.          TTL filter
                                    2.  *
                                    3.  * This program simply takes a 256-complex point
                                    4.  * frequency domain filter in natural order and
                                    5.  * performs an inverse 4-shuffle, then downloads
                                    6.  * 64 points to each slave processor, to be used
                                    7.  * in the digital filtering application program
                                    8.  *
                                    9.  * Aug 13,1991 Paul A. Smeulders
                                   10.  *
                                   11.  * EQUATES
                                   12.  *
# 00012FC0                         13.  PGCR EQU $12FC0
# 00000020                         14.  ENABLEB EQU $20
# 00000007                         15.  ADBYTOT EQU 7
# 00000006                         16.  OUTDATA EQU 6
# 00000008                         17.  NETCNF  EQU 8
# 00000009                         18.  SRQSRVC EQU 9
# 0000000E                         19.  ABORT   EQU 14
# 00000100                         20.  NPT     EQU $100
# 00000040                         21.  NPTE    EQU $40
# 00000004                         22.  NPE     EQU 4
# 00000001                         23.  SLV0    EQU $01
# 00000002                         24.  SLV1    EQU $02
# 00000004                         25.  SLV2    EQU $04
# 00000008                         26.  SLV3    EQU $08
# 0000000F                         27.  ALLSLV  EQU $0F
# 00012FB0                         28.  PSR01   EQU $12FB0
# 00012FB1                         29.  PSR23   EQU $12FB1
# 00012FB2                         30.  PSR45   EQU $12FB2
# 00012FB3                         31.  PSR67   EQU $12FB3
# 00012FB4                         32.  PSR89   EQU $12FB4
# 00012FB5                         33.  PSR1011 EQU $12FB5
# 00012FB6                         34.  PSR1213 EQU $12FB6
# 00012FB7                         35.  PSR1415 EQU $12FB7
# 00012FB8                         36.  PSR1617 EQU $12FB8
# 00012FB9                         37.  PSR1819 EQU $12FB9
# 0000000A                         38.  SENCLFC EQU 10
# 00000010                         39.  INCA2   EQU $10
# 000003FC                         40.  DECA3         EQU $3FC
                                   41.  *
                                   42.  ******************
                                   43.  * MASTER PROGRAM *
                                   44.  ******************
                                   45.  *
00001800                           46.          ORG.L $1800
00001800  <400>                    47.  FILSTRT DS B $400
00001C00  <100>                    48.  FILSCR0 DS.B $100
00001D00  <100>                    49.  FILSCR1 DS.B $100
00001E00  <100>                    50.  FILSCR2 DS.B $100
00001F00  <100>                    51.  FILSCR3 DS.B $100
00002000  BB BB BB CB BB           52.  INITCNF DC.B $BB,$BB,$BB,$CB,$BB,$8C,$CB,$8B,$B8,$8B
          8C CB 8B B8 8B
0000200A  BD BB BD CB BD           53.  BRDCAST DC.B $BD,$BB,$BD,$CB,$BD,$8C,$CB,$8B,$B8,$8B
          8C CB 8B B8 8B
00002014  00 00 00 00 00           54.  RESNET  DC.B $00,$00,$00,$00,$00,$00,$00,$00,$00,$00
          00 00 00 00 00
                                   55.  *
                                   56.  * FIRST PERFORM THE INVERSE 4-SHUFFLE OF 256
                                   57.  * LONGWORDS AT FILSTRT AND PUT IN FILSCR0-3
                                   58.  *
0000201E  41FA F7E0                59.  FILTER  LEA FILSTRT(PC),A0      ;A0 SOURCE
00002022  43FA FBDC                60.          LEA FILSCR0(PC),A1      ;A1 DEST
00002026  247C 00000010            61.          MOVEA.L #INCA2,A2
0000202C  267C 000003FC            62.          MOVEA.L #DECA3,A3
00002032  7003                     63.          MOVEQ.L #NPE-1,D0       ;OUTER CNTR
00002034  723F                     64.  OUTR    MOVEQ.L #NPTE-1,D1      ;INNER CNTR
00002036  22D0                     65.  INNER   MOVE.L (A0),(A1)+
00002038  D1CA                     66.          ADDA.L A2,A0
0000203A  51C9 FFFA                67.          DBRA D1,INNER
0000203E  91CB                     68.          SUBA.L A3,A0
00002040  51C8 FFF2                69.          DBRA D0,OUTR
                                   70.  *
                                   71.  * REV SHUFFLE DONE
                                   72.  *
```

```
                                73.  *
                                74.  * CONFIGURE NETWORK TO INITIAL STATE, MASTER BRDCAST
                                75.  * TO ALL SLAVES
                                76.  * RESPOND TO ALL SLAVE'S RFP SRQ, AND LOAD
                                77.  * THE PROGRAM.  SINCE ONLY WANT DATA LOADED
                                78.  * SLAVE PROGRAM IS A ONE-LINER
                                79.  *
                                80.  *
00002044  41FA FFC4              81.          LEA BRDCAST(PC),A0
00002048  4E48                  82.          TRAP #NETCNF
0000204A  103C 000F             83.          MOVE.B #ALL_SLV,D0
0000204E  4E49                  84.          TRAP #SRQSRVC
00002050  41FA 00A4             85.          LEA SLVPRO(PC),A0        ;STRT ADDR
00002054  43FA 00A2             86.          LEA ENDSLV(PC),A1        ;END ADDR+1
00002058  6100 007C             87.          BSR SENDOUT              ;ADDR,COUNT,CODE
                                88.  *
                                89.  * RESPOND TO ALL SLAVE'S RFD,
                                90.  * BROADCAST THE ADDRESS AND BYTE COUNT OF
                                91.  * FILTER DATA TO ALL, BUT SEND UP DATA
                                92.  * SEPARATELY
                                93.  *
0000205C  103C 000F             94.          MOVE.B #ALLSLV,D0
00002060  4E49                  95.          TRAP #SRQSRVC
00002062  41FA F79C             96.          LEA FILSTRT(PC),A0
00002066  303C 0100             97.          MOVE.W #$100,D0
0000206A  13FC 0020             98.          MOVE.B #ENABLEB,PGCR
          00012FC0
00002072  4E47                  99.          TRAP #ADBYTOT
                               100.  *
                               101.  * NOW SEND UP INDIVIDUAL FILTER DATA
                               102.  *
00002074  41FA FF8A             103.          LEA INITCNF(PC),A0
00002078  4E48                  104.          TRAP #NETCNF
0000207A  41FA FB84             105.          LEA FILSCR0(PC),A0
0000207E  43FA FC80             106.          LEA FILSCR1(PC),A1
00002082  6100 006A             107.          BSR SNDOUT2
                               108.  *
                               109.  * RESPOND TO SLAVE 1'S RFD, CONFIGURE
                               110.  * NETWORK, AND LOAD DATA
                               111.  *
00002086  13FC 00B4             112.          MOVE.B #$B4,PSR01
          00012FB0
0000208E  41FA FC70             113.          LEA FILSCR1(PC),A0
00002092  43FA FD6C             114.          LEA FILSCR2(PC),A1
00002096  6100 0056             115.          BSR SNDOUT2
                               116.  *
                               117.  * RESPOND TO SLAVE 2'S RFD, CONFIGURE
                               118.  * NETWORK, AND LOAD DATA
                               119.  *
0000209A  13FC 00B4             120.          MOVE.B #$B4,PSR45
          00012FB2
000020A2  41FA FD5C             121.          LEA FILSCR2(PC),A0
000020A6  43FA FE58             122.          LEA FILSCR3(PC),A1
000020AA  6100 0042             123.          BSR SNDOUT2
                               124.  *
                               125.  * RESPOND TO SLAVE 3'S RFD, CONFIGURE
                               126.  * NETWORK, AND LOAD DATA
                               127.  *
000020AE  13FC 00B4             128.          MOVE.B #$B4,PSR89
          00012FB4
000020B6  41FA FE48             129.          LEA FILSCR3(PC),A0
000020BA  43FA FF44             130.        LEA INITCNF(PC),A1
000020BE  6100 002E             131.          BSR SNDOUT2
000020C2  4239 00012FC0         132.          CLR.B PGCR
                               133.  *
                               134.  * WHEN HERE, ALL SLAVES LOADED, WILL ASSERT
                               135.  * READY TO EXECUTE SRQ.  RESPOND TO LET THEM
                               136.  * CONTINUE. THEY DO NOTHING, AND THE
                               137.  * PROGRAM IS DONE.
                               138.  *
000020C8  41FA FF4A             139.          LEA RESNET(PC),A0
000020CC  4E48                  140.          TRAP #NETCNF
000020CE  103C 000F             141.          MOVE.B #ALLSLV,D0
000020D2  4E49                  142.          TRAP #SRQSRVC
000020D4  4E75                  143.          RTS
                               144.  *
                               145.  *
                               146.  * SUBROUTINE SENDOUT
                               147.  *
000020D6  93C8                  148.  SENDOUT SUBA.L A0,A1
000020D8  3009                  149.          MOVE.W A1,D0
```

```
000020DA  13FC 0020          150.          MOVE.B #ENABLEB,PGCR
          00012FC0
000020E2  4E47               151.          TRAP #ADBYTOT
000020E4  4E46               152.          TRAP #OUTDATA
000020E6  4239 00012FC0      153.          CLR.B PGCR
000020EC  4E75               154.          RTS
                             155.  *
                             156.  * SUBROUTINE SNDOUT2
                             157.  *
000020EE  93C8               158.  SNDOUT2 SUBA.L A0,A1
000020F0  3009               159.          MOVE.W A1,D0
000020F2  4E46               160.          TRAP #OUTDATA
000020F4  4E75               161.          RTS
                             162.  *
                             163.  ********************
                             164.  * SLAVE PROGRAM    *
                             165.  ********************
                             166.  *
000020F6  4E4E               167.  SLVPRO  TRAP #ABORT
000020F8  <1>                168.  ENDSLV  DS.B 1
000020F9                     169.          END
    0 Errors
```

M.3 Program PFILNOM1: Multicomputer Frequency Domain Filtering Program #1A

```
                                1.          TIL PARFIL1
                                2.  *
                                3.  * This program performs a 256 complex point
                                4.  * frequency domain filtering operation on an
                                5.  * input signal at DATAII.  The filter
                                6.  * coefficients are at FILTER, and are
                                7.  * distributed among slaves using another
                                8.  * program (filter.a68).
                                9.  *
                               10.  * input/output data is stored as real/imaginary
                               11.  * pair, each component 16-bits long.
                               12.  * twiddle factors within slave programs are
                               13.  * stored similarly
                               14.  *
                               15.  *
                               16.  * self timed operation, 256 runs
                               17.  *
                               18.  * EQUATES
C'000000                       19.          SECTION 0
                               20.          XDEF PGCR,ENABLEA,ENABLEB
                               21.          XDEF NPT,NPTE,NPE,SHIFT14
                               22.          XDEF TCR,CNTSTRT,ENABLTM
                               23.          XREF S0PRO,S1PRO,S2PRO,S3PRO,DONE
                               24.          XDEF DATAII,FILTER,BRDCAST
                               25.  *
     # 00012FC0                26.  PGCR     EQU $12FC0
     # 00000010                27.  ENABLEA  EQU $10
     # 00000020                28.  ENABLEB  EQU $20
     # 00000100                29.  NPT      EQU $100
     # 00000040                30.  NPTE     EQU $40
     # 00000004                31.  NPE      EQU 4
     # 0000000E                32.  SHIFT14  EQU 14
     # 00000003                33.  SRQASRT  EQU 3
     # 00000004                34.  ADBYTIN  EQU 4
     # 00000007                35.  ADBYTOT  EQU 7
     # 00000005                36.  INDATA   EQU 5
     # 00000006                37.  OUTDATA  EQU 6
     # 00000008                38.  NETCNF   EQU 8
     # 00000009                39.  SRQSRVC  EQU 9
     # 0000000E                40.  ABORT    EQU 14
     # 00000001                41.  SLV0     EQU $01
     # 00000002                42.  SLV1     EQU $02
     # 00000004                43.  SLV2     EQU $04
     # 00000008                44.  SLV3     EQU $08
     # 0000000F                45.  ALLSLV   EQU $0F
     # 00000006                46.  NUGAM    EQU 6
     # 00012FB0                47.  PSR01    EQU $12FB0
     # 00012FB1                48.  PSR23    EQU $12FB1
     # 00012FB2                49.  PSR45    EQU $12FB2
     # 00012FB3                50.  PSR67    EQU $12FB3
     # 00012FB4                51.  PSR89    EQU $12FB4
     # 00012FB5                52.  PSR1011  EQU $12FB5
     # 00012FB6                53.  PSR1213  EQU $12FB6
     # 00012FB7                54.  PSR1415  EQU $12FB7
     # 00012FB8                55.  PSR1617  EQU $12FB8
     # 00012FB9                56.  PSR1819  EQU $12FB9
     # 00012FD0                57.  TCR      EQU $12FD0
     # 00012FD2                58.  TPLR     EQU TCR+2
     # 00012FD6                59.  TCNTR    EQU TCR+6
     # 0000000A                60.  SENCLFC  EQU 10
     # 00000018                61.  DISBUF8  EQU 24
     # 0000001A                62.  DISBUF2  EQU 26
     # 00FFFFFF                63.  CNTSTRT  EQU $00FFFFFF
     # 00000001                64.  ENABLTM  EQU 1
     # 00000011                65.  OUTMESC  EQU 17
     # 00000008                66.  DIV256   EQU 8
     # 00000100                67.  PASSCNT  EQU 256
     # 00000004                68.  EOT      EQU 4
                               69.  ****************************************
                               70.  * THIS VERSION DOES INDIVIDUAL SYNCS FOR
                               71.  * EACH UPLOAD OF SLAVE DATA
                               72.  * COLLECTS IN ORDER 0,1,2,3
                               73.  ****************************************
                               74.  ******************
                               75.  * MASTER PROGRAM *
                               76.  ******************
```

```
                                77.  *
0'000000   <400>                78.  DATAIN  DS.B $400
0'000400   <400>                79.  DATAII  DS.B $400
0'000800   <400>                80.  FILTER  DS.B $400
0'000C00   <100>                81.  XDATA0  DS.B $100
0'000D00   <100>                82.  XDATA1  DS.B $100
0'000E00   <100>                83.  XDATA2  DS.B $100
0'000F00   <100>                84.  XDATA3  DS.B $100
0'001000   BB BB BB CB BB       85.  INITCNF DC.B $BB,$BB,$BB,$CB,$BB,$8C,$CB,$8B,$B8,$8B
           8C CB 8B B8 8B
0'00100A   BD BB BD CB BB       86.  BRDCAST DC.B $BD,$BB,$BD,$CB,$BD,$8C,$CB,$8B,$B8,$8B
           8C CB 8B B8 8B
                                87.  *
                                88.  * CONFIGURE NETWORK TO INITIAL STATE, MASTER
                                89.  * TO SLAVE 0
                                90.  * RESPOND TO SLAVE 0'S RFP SRQ, AND LOAD
                                91.  * ITS PROGRAM
                                92.  *
0'001014   287C 00012FD0        93.  STRTFFT MOVEA.L #TCR,A4          ;POINTER TIMER
0'00101A   4214                 94.          CLR.B (A4)               ;DISABLE F'SURE
0'00101C   2F3C 00FFFFFF        95.          MOVE.L #CNTSTRT,-(SP)    ;MIN TIME
0'001022   2F3C 00000000        96.          MOVE.L #$00,-(SP)        ;MAX TIME
0'001028   2F3C 00000000        97.          MOVE.L #$00,-(SP)        ;RUN'G SUM TIME
0'00102E   2E3C 000000FF        98.          MOVE.L #PASSCNT-1,D7     ;COUNTER
0'001034   41FA FFCA            99.          LEA INITCNF(PC),A0
0'001038   4E48                100.          TRAP #NETCNF
0'00103A   103C 0001           101.          MOVE.B #SLV0,D0
0'00103E   4E49                102.          TRAP #SRQSRVC
0'001040   41FA ****           103.          LEA S0PRO(PC),A0         ;STRT ADDR
0'001044   43FA ****           104.          LEA S1PRO(PC),A1         ;END ADDR+1
0'001048   6100 01F6           105.          BSR SENDOUT              ;ADDR,COUNT,CODE
                               106.  *
                               107.  * CONFIGURE NETWORK TO MASTER TO SLAVE 1
                               108.  * RESPOND TO SLAVE 1'S RFP SRQ, AND LOAD ITS
                               109.  * PROGRAM
                               110.  *
0'00104C   13FC 00B4           111.          MOVE.B #$B4,PSR01
           00012FB0
0'001054   103C 0002           112.          MOVE.B #SLV1,D0
0'001058   4E49                113.          TRAP #SRQSRVC
0'00105A   41FA ****           114.          LEA S1PRO(PC),A0
0'00105E   43FA ****           115.          LEA S2PRO(PC),A1
0'001062   6100 01DC           116.          BSR SENDOUT
                               117.  *
                               118.  * CONFIGURE NETWORK TO MASTER TO SLAVE 2
                               119.  * RESPOND TO SLAVE 2'S RFP SRQ, AND LOAD ITS
                               120.  * PROGRAM
                               121.  *
0'001066   13FC 00B4           122.          MOVE.B #$B4,PSR45
           00012FB2
0'00106E   103C 0004           123.          MOVE.B #SLV2,D0
0'001072   4E49                124.          TRAP #SRQSRVC
0'001074   41FA ****           125.          LEA S2PRO(PC),A0
0'001078   43FA ****           126.          LEA S3PRO(PC),A1
0'00107C   6100 01C2           127.          BSR SENDOUT
                               128.  *
                               129.  * CONFIGURE NETWORK TO MASTER TO SLAVE 3
                               130.  * RESPOND TO SLAVE 3'S RFP SRQ, AND LOAD ITS
                               131.  * PROGRAM
                               132.  *
0'001080   13FC 00B4           133.          MOVE.B #$B4,PSR89
           00012FB4
0'001088   103C 0008           134.          MOVE.B #SLV3,D0
0'00108C   4E49                135.          TRAP #SRQSRVC
0'00108E   41FA ****           136.          LEA S3PRO(PC),A0
0'001092   43FA ****           137.          LEA DONE(PC),A1
0'001096   6100 01A8           138.          BSR SENDOUT
                               139.  *
                               140.  * COPY INPUT DATA ARRAY TO DATAII SO EACH RUN
                               141.  * HAS SAME INPUT DATA
                               142.  *
0'00109A   41FA EF64           143.  REEXE   LEA DATAIN(PC),A0    ;SOURCE
0'00109E   43FA F360           144.          LEA DATAII(PC),A1    ;DEST
0'0010A2   323C 00FF           145.          MOVE.W #$FF,D1
0'0010A6   22D8                146.  MVLP    MOVE.L (A0)+,(A1)+
0'0010A8   51C9 FFFC           147.          DBRA D1,MVLP
                               148.  *
                               149.  * ALL PROGRAMS LOADED, EACH ASSERTING READY FOR
                               150.  * DATA SRQ.  RESPOND TO ALL, THEN LOAD THE DATA
                               151.  * IN BROADCAST MODE.
                               152.  *
```

```
                                153.  * LOAD TIMER PRE-LOAD REGISTER HERE
                                154.  *
0'0010AC  297C 00FFFFFF         155.          MOVE.L #CNTSTRT,2(A4)
          0002
                                156.  *
                                157.  * SET UP SOME REGISTER VALUES
                                158.  *
0'0010B4  3C3C 0100             159.          MOVE.W #NPTE*4,D6
0'0010B8  267C 000003FC         160.          MOVEA.L #$3FC,A3
0'0010BE  247C 00000010         161.          MOVEA.L #$10,A2
0'0010C4  2C7C 00012FC0         162.          MOVEA.L #PGCR,A6
0'0010CA  2A7C 00012FB0         163.          MOVEA.L #PSR01,A5
0'0010D0  4243                  164.          CLR.W D3
0'0010D2  41FA FF36             165.          LEA BRDCAST(PC),A0
0'0010D6  4E48                  166.          TRAP #NETCNF
0'0010D8  103C 000F             167.          MOVE.B #ALLSLV,D0
0'0010DC  4E49                  168.          TRAP #SRQSRVC
0'0010DE  41FA F320             169.          LEA DATA1(PC),A0        ;START OF DATA
0'0010E2  43FA F71C             170.          LEA FILTER(PC),A1       ;END OF DATA+1
                                171.  *
                                172.  * START TIMER COUNTING HERE
                                173.  *
0'0010E6  18BC 0001             174.          MOVE.B #ENABLTM,(A4)
0'0010EA  6100 0154             175.          BSR SENDOUT
                                176.  *
                                177.  * WHEN HERE, ALL SLAVES LOADED, WILL ASSERT
                                178.  * READY TO EXECUTE SRQ.  RESPOND TO LET THEM
                                179.  * CONTINUE.  NETWORK IS CONFIGURED TO ACCEPT
                                180.  * DATA FROM SLAVE 0.
                                181.  *
0'0010EE  700F                  182.          MOVEQ.L #ALLSLV,D0
0'0010F0  4E49                  183.          TRAP #SRQSRVC           ;SLAVES EXECUTE
                                184.  *
                                185.  * SLAVES ASSERT SRQ, THEN UPLOAD DATA
                                186.  * WHEN ACKNOWLEDGED. ACKNOWLEDGE INDIVIDUALLY
                                187.  * THEN UPLOAD FROM EACH
                                188.  *
0'0010F2  7001                  189.  REDIST  MOVEQ.L #SLV0,D0
0'0010F4  4E49                  190.          TRAP #SRQSRVC
0'0010F6  3006                  191.          MOVE.W D6,D0        ;BYTE COUNT
0'0010F8  41FA FB06             192.          LEA XDATA0(PC),A0       ;DEST ADDRESS
0'0010FC  1CBC 0010             193.          MOVE.B #ENABLEA,(A6)
0'001100  4E45                  194.          TRAP #INDATA
0'001102  4216                  195.          CLR.B (A6)
                                196.  *
                                197.  * CONFIGURE FOR SLAVE #1 TO MASTER,
                                198.  * AND GET DATA
                                199.  *
0'001104  7002                  200.          MOVEQ.L #SLV1,D0
0'001106  4E49                  201.          TRAP #SRQSRVC
0'001108  1B7C 00BC 0005        202.          MOVE.B #$BC,5(A5)
0'00110E  3006                  203.          MOVE.W D6,D0
0'001110  41FA FBEE             204.          LEA XDATA1(PC),A0
0'001114  1CBC 0010             205.          MOVE.B #ENABLEA,(A6)
0'001118  4E45                  206.          TRAP #INDATA
0'00111A  4216                  207.          CLR.B (A6)
                                208.  *
                                209.  * CONFIGURE FOR SLAVE #2 TO MASTER,
                                210.  * AND GET DATA
                                211.  *
0'00111C  7004                  212.          MOVEQ.L #SLV2,D0
0'00111E  4E49                  213.          TRAP #SRQSRVC
0'001120  1B7C 00BB 0007        214.          MOVE.B #$BB,7(A5)
0'001126  3006                  215.          MOVE.W D6,D0
0'001128  41FA FCD6             216.          LEA XDATA2(PC),A0
0'00112C  1CBC 0010             217.          MOVE.B #ENABLEA,(A6)
0'001130  4E45                  218.          TRAP #INDATA
0'001132  4216                  219.          CLR.B (A6)
                                220.  *
                                221.  * CONFIGURE FOR SLAVE #3 TO MASTER,
                                222.  * AND GET DATA
                                223.  *
0'001134  7008                  224.          MOVEQ.L #SLV3,D0
0'001136  4E49                  225.          TRAP #SRQSRVC
0'001138  1B7C 00BB 0009        226.          MOVE.B #$BB,9(A5)
0'00113E  3006                  227.          MOVE.W D6,D0
0'001140  41FA FDBE             228.          LEA XDATA3(PC),A0
0'001144  1CBC 0010             229.          MOVE.B #ENABLEA,(A6)
0'001148  4E45                  230.          TRAP #INDATA
0'00114A  4216                  231.          CLR.B (A6)
                                232.  *
```

```
                                233.  * HAVE ALL DATA IN, NOW DO REARRANGE FROM THE
                                234.  * XDATAi ARRAY TO THE DATAII ARRAY FOR CORRECT
                                235.  * ORDER.
                                236.  *
0'00114C  41FA FAB2             237.  REORDER LEA XDATA0(PC),A0
0'001150  43FA F2AE             238.          LEA DATAII(PC),A1
0'001154  7203                  239.          MOVEQ.L #NPE-1,D1
0'001156  7A3F                  240.  RLOOPT  MOVEQ.L #NPTE-1,D5
0'001158  2298                  241.  RLOOPN  MOVE.L (A0)+,(A1)
0'00115A  D3CA                  242.          ADDA.L A2,A1
0'00115C  51CD FFFA             243.          DBRA D5,RLOOPN
0'001160  93CB                  244.          SUBA.L A3,A1
0'001162  51C9 FFF2             245.          DBRA D1,RLOOPT
                                246.  *
                                247.  * forward fft and filtering is done
                                248.  * now redistribute the frequency domain data
                                249.  * to the slaves for the inverse transform.
                                250.  * simply do again!
                                251.  *
0'001166  5243                  252.          ADDQ.W #1,D3
0'001168  0C43 0002             253.          CMPI.W #2,D3
0'00116C  67 2C                 254.          BEQ.S SKIP
0'00116E  41FA FE9A             255.          LEA BRDCAST(PC),A0
0'001172  4E48                  256.          TRAP #NETCNF
0'001174  103C 000F             257.          MOVE.B #ALLSLV,D0
0'001178  4E49                  258.          TRAP #SRQSRVC
0'00117A  41FA F284             259.          LEA DATAII(PC),A0       ;START OF DATA
0'00117E  43FA F680             260.          LEA FILTER(PC),A1
0'001182  93C8                  261.          SUBA.L A0,A1
0'001184  3009                  262.          MOVE.W A1,D0
0'001186  13FC 0020             263.          MOVE.B #ENABLEB,PGCR
          00012FC0
0'00118E  4E46                  264.          TRAP #OUTDATA
0'001190  4239 00012FC0         265.          CLR.B PGCR
0'001196  6000 FF5A             266.          BRA REDIST
                                267.  *
                                268.  * PROGRAM IS DONE, CAN STOP TIMER, AND
                                269.  * OUTPUT NUMBER OF 32 CLOCK INTERVALS
                                270.  * PASSED. FIRST DISABLE TIMER
                                271.  *
0'00119A  4214                  272.  SKIP    CLR.B (A4)
0'00119C  222C 0006             273.          MOVE.L 6(A4),D1 ;GET NEW COUNT
0'0011A0  203C 00FFFFFF         274.          MOVE.L #CNTSTRT,D0
0'0011A6  9081                  275.          SUB.L D1,D0     ;GET ELAPSED COUNT
                                276.  *
                                277.  * OUTPUT CR, LF, THEN THE COUNT IN HEX
                                278.  *
0'0011A8  4E4F                  279.          TRAP #15
0'0011AA  000A                  280.          DC.W SENCLFC
0'0011AC  4E4F                  281.          TRAP #15
0'0011AE  0018                  282.          DC.W DISBUF8
0'0011B0  2F00                  283.          MOVE.L D0,-(SP)
0'0011B2  41FA 0119             284.          LEA SPC(PC),A0
0'0011B6  4E4F                  285.          TRAP #15
0'0011B8  0011                  286.          DC.W OUTMESC
0'0011BA  2007                  287.          MOVE.L D7,D0
0'0011BC  4E4F                  288.          TRAP #15
0'0011BE  0018                  289.          DC.W DISBUF8
0'0011C0  201F                  290.          MOVE.L (SP)+,D0
                                291.  *
                                292.  * NOW KEEP TRACK OF STATISTICS
                                293.  *
0'0011C2  D197                  294.          ADD.L D0,(SP)    ;ADD TO SUM
0'0011C4  B0AF 0004             295.          CMP.L 4(SP),D0   ;CHECK MAX
0'0011C8  6F 04                 296.          BLE.S NOTMAX     ;D0 NOT MAX
0'0011CA  2F40 0004             297.          MOVE.L D0,4(SP)  ;NEW MAX
0'0011CE  B0AF 0008             298.  NOTMAX  CMP.L 8(SP),D0   ;CHECK MIN
0'0011D2  6C 04                 299.          BGE.S NOTMIN     ;D0 NOT MIN
0'0011D4  2F40 0008             300.          MOVE.L D0,8(SP)  ;NEW MIN
0'0011D8  51CF FEC0             301.  NOTMIN  DBRA D7,REEXE    ;DO D7 TIMES
                                302.
                                303.  *
                                304.  * WHEN HERE, PROGRAM HAS BEEN EXECUTED 256 TIMES
                                305.  * SEND AN ILLEGAL ADDRESS FOR DATA, FORCING
                                306.  * SLAVES TO PROGRAM ACCEPT MODE
                                307.  *
0'0011DC  41FA FE2C             308.          LEA BRDCAST(PC),A0
0'0011E0  4E48                  309.          TRAP #NETCNF
0'0011E2  103C 000F             310.          MOVE.B #ALLSLV,D0
0'0011E6  4E49                  311.          TRAP #SRQSRVC
0'0011E8  207C FFFF0000         312.          MOVEA.L #$FFFF0000,A0   ;ILLEGAL ADDR
```

```
0'0011EE  13FC 0020          313.          MOVE.B #ENABLEB,PGCR
          00012FC0
0'0011F6  4E47               314.          TRAP #ADBYTOT   ;THAT'LL FIX 'EM
0'0011F8  4239 00012FC0      315.          CLR.B PGCR
                             316.  *
                             317.  * OUTPUT STATISTICS
                             318.  *
0'0011FE  4E4F               319.          TRAP #15
0'001200  000A               320.          DC.W SENCLFC
0'001202  4E4F               321.          TRAP #15
0'001204  000A               322.          DC.W SENCLFC
0'001206  41FA 0050          323.          LEA AVEMES(PC),A0
0'00120A  4E4F               324.          TRAP #15
0'00120C  0011               325.          DC.W OUTMESC
0'00120E  201F               326.          MOVE.L (SP)+,D0 ;GET SUM
0'001210  E080               327.          ASR.L #DIV256,D0        ;DO AVERAGE
0'001212  4E4F               328.          TRAP #15
0'001214  0018               329.          DC.W DISBUF8    ;OUTPUT AVERAGE
0'001216  4E4F               330.          TRAP #15
0'001218  000A               331.          DC.W SENCLFC
0'00121A  41FA 0063          332.          LEA MAXMES(PC),A0
0'00121E  4E4F               333.          TRAP #15
0'001220  0011               334.          DC.W OUTMESC
0'001222  201F               335.          MOVE.L (SP)+,D0 ;GET MAX
0'001224  4E4F               336.          TRAP #15
0'001226  0018               337.          DC.W DISBUF8
0'001228  4E4F               338.          TRAP #15
0'00122A  000A               339.          DC.W SENCLFC
0'00122C  41FA 0078          340.          LEA MINMES(PC),A0
0'001230  4E4F               341.          TRAP #15
0'001232  0011               342.          DC.W OUTMESC
0'001234  201F               343.          MOVE.L (SP)+,D0 ;GET MIN
0'001236  4E4F               344.          TRAP #15
0'001238  0018               345.          DC.W DISBUF8
0'00123A  4E4F               346.          TRAP #15
0'00123C  000A               347.          DC.W SENCLFC
                             348.  *
                             349.  * AFTER 256 RUNS, LOAD THE SECOND SLAVE
                             350.  * PROGRAM TO ACQUIRE THE TIMING DATA
                             351.  * IN THE SLAVES TIMER COUNT REGISTERS
                             352.  *
                             353.  * THAT'S SECTION 1, LINK IN CODE!
                             354.  *
0'00123E  4E75               355.          RTS
                             356.  *
                             357.  * SUBROUTINE SENDOUT
                             358.  *
0'001240  93C8               359.  SENDOUT SUBA.L A0,A1
0'001242  3009               360.          MOVE.W A1,D0
0'001244  13FC 0020          361.          MOVE.B #ENABLEB,PGCR
          00012FC0
0'00124C  4E47               362.          TRAP #ADBYTOT
0'00124E  4E46               363.          TRAP #OUTDATA
0'001250  4239 00012FC0      364.          CLR.B PGCR
0'001256  4E75               365.          RTS
                             366.  *
0'001258  41 76 65 72 61     367.  AVEMES  DC.B 'Average time ( x 4 for microseconds)= ',EOT
          67 65 20 74 69
          6D 65 20 28 20
          78 20 34 20 66
          6F 72 20 6D 69
          63 72 6F 73 65
          63 6F 6E 64 73
          29 3D 20 04
0'00127F  4D 61 78 69 6D     368.  MAXMES  DC.B 'Maximum time ( x 4 for microseconds)= ',EOT
          75 6D 20 74 69
          6D 65 20 28 20
          78 20 34 20 66
          6F 72 20 6D 69
          63 72 6F 73 65
          63 6F 6E 64 73
          29 3D 20 04
0'0012A6  4D 69 6E 69 6D     369.  MINMES  DC.B 'Minimum time ( x 4 for microseconds)= ' EOT
          75 6D 20 74 69
          6D 65 20 28 20
          78 20 34 20 66
          6F 72 20 6D 69
          63 72 6F 73 65
          63 6F 6E 64 73
          29 3D 20 04
```

```
0'0012CD  20 20 20 20 04  370.  SPC     DC.B '    ',EOT
0'0012D2                  371.          END
    0 Errors
*
* LINK SPEC FOR PFILNOM1
*
      LINK PFILNOM1
      LINK SLVNOMES
      ORG $1000
      SECTION 0,15
      END
```

M.4 Program PFILNOM2: Multicomputer Frequency Domain Filtering Program #2A

```
                                    1.          TTL PARFIL2
                                    2.  *
                                    3.  * This program performs a 256 complex point
                                    4.  * frequency domain filtering operation on an
                                    5.  * input signal at DATAII.  The filter
                                    6.  * coefficients are at FILTER, and are
                                    7.  * distributed among slaves using another
                                    8.  * program (filter.a68).
                                    9.  *
                                   10.  * input/output data is stored as real/imaginary
                                   11.  * pair, each component 16-bits long.
                                   12.  * twiddle factors within slave programs are
                                   13.  * stored similarly
                                   14.  *
                                   15.  *
                                   16.  * self timed operation, 256 runs
                                   17.  *
                                   18.  * EQUATES
0'000000                           19.          SECTION 0
                                   20.          XDEF PGCR,ENABLEA,ENABLEB
                                   21.          XDEF NPT,NPTE,NPE,SHIFT14
                                   22.          XDEF TCR,CNTSTRT,ENABLTM
                                   23.          XREF S0PRO,S1PRO,S2PRO,S3PRO,DONE
                                   24.          XDEF DATAII,FILTER,BRDCAST
                                   25.  *
     # 00012FC0                    26.  PGCR     EQU $12FC0
     # 00000010                    27.  ENABLEA EQU $10
     # 00000020                    28.  ENABLEB EQU $20
     # 00000100                    29.  NPT      EQU $100
     # 00000040                    30.  NPTE     EQU $40
     # 00000004                    31.  NPE      EQU 4
     # 0000000E                    32.  SHIFT14 EQU 14
     # 00000003                    33.  SRQASRT EQU 3
     # 00000004                    34.  ADBYTIN EQU 4
     # 00000007                    35.  ADBYTOT EQU 7
     # 00000005                    36.  INDATA  EQU 5
     # 00000006                    37.  OUTDATA EQU 6
     # 00000008                    38.  NETCNF  EQU 8
     # 00000009                    39.  SRQSRVC EQU 9
     # 0000000E                    40.  ABORT   EQU 14
     # 00000001                    41.  SLV0     EQU $01
     # 00000002                    42.  SLV1     EQU $02
     # 00000004                    43.  SLV2     EQU $04
     # 00000008                    44.  SLV3     EQU $08
     # 0000000F                    45.  ALLSLV  EQU $0F
     # 00000006                    46.  NUGAM   EQU 6
     # 00012FB0                    47.  PSR01   EQU $12FB0
     # 00012FB1                    48.  PSR23   EQU $12FB1
     # 00012FB2                    49.  PSR45   EQU $12FB2
     # 00012FB3                    50.  PSR67   EQU $12FB3
     # 00012FB4                    51.  PSR89   EQU $12FB4
     # 00012FB5                    52.  PSR1011 EQU $12FB5
     # 00012FB6                    53.  PSR1213 EQU $12FB6
     # 00012FB7                    54.  PSR1415 EQU $12FB7
     # 00012FB8                    55.  PSR1617 EQU $12FB8
     # 00012FB9                    56.  PSR1819 EQU $12FB9
     # 00012FD0                    57.  TCR      EQU $12FD0
     # 00012FD2                    58.  TPLR     EQU TCR+2
     # 00012FD6                    59.  TCNTR    EQU TCR+6
     # 0000000A                    60.  SENCLFC EQU 10
     # 00000018                    61.  DISBUF8 EQU 24
     # 0000001A                    62.  DISBUF2 EQU 26
     # 00FFFFFF                    63.  CNTSTRT EQU $00FFFFFF
     # 00000001                    64.  ENABLTM EQU 1
     # 00000011                    65.  OUTMESC EQU 17
     # 00000008                    66.  DIV256  EQU 8
     # 00000100                    67.  PASSCNT EQU 256
     # 00000004                    68.  EOT      EQU 4
                                   69.  ****************************************
                                   70.  * THIS VERSION WAITS FOR ALL SLAVES TO
                                   71.  * ASSERT SYNC BEFORE COLLECTING SLAVE
                                   72.  * DATA, THEN
                                   73.  * COLLECTS IN ORDER 0,1,2,3
                                   74.  ****************************************
                                   75.  ******************
                                   76.  * MASTER PROGRAM *
```

```
                                77.  ******************
                                78.  *
0'000000  <400>                 79.  DATAIN  DS.B $400
0'000400  <400>                 80.  DATAII  DS.B $400
0'000800  <400>                 81.  FILTER  DS.B $400
0'000C00  <100>                 82.  XDATA0  DS.B $100
0'000D00  <100>                 83.  XDATA1  DS.B $100
0'000E00  <100>                 84.  XDATA2  DS.B $100
0'000F00  <100>                 85.  XDATA3  DS.B $100
0'001000  BB BB BB CB BB        86.  INITCNF DC.B $BB,$BB,$BB,$CB,$BB,$8C,$CB,$8B,$B8,$8B
          8C CB 8B B8 8B
0'00100A  BD BB BD CB BD        87.  BRDCAST DC.B $BD,$BB,$BD,$CB,$BD,$8C,$CB,$8B,$B8,$8B
          8C CB 8B B8 8B
                                88.  *
                                89.  * CONFIGURE NETWORK TO INITIAL STATE, MASTER
                                90.  * TO SLAVE 0
                                91.  * RESPOND TO SLAVE 0'S RFP SRQ, AND LOAD
                                92.  * ITS PROGRAM
                                93.  *
0'001014  287C 00012FD0         94.  STRTFFT MOVEA.L #TCR,A4          ;POINTER TIMER
0'00101A  4214                  95.          CLR.B (A4)               ;DISABLE F'SURE
0'00101C  2F3C 00FFFFFF         96.          MOVE.L #CNTSTRT,-(SP)    ;MIN TIME
0'001022  2F3C 00000000         97.          MOVE.L #$00,-(SP)        ;MAX TIME
0'001028  2F3C 00000000         98.          MOVE.L #$00,-(SP)        ;RUN'G SUM TIME
0'00102E  2E3C 000000FF         99.          MOVE.L #PASSCNT-1,D7     ;COUNTER
0'001034  41FA FFCA            100.          LEA INITCNF(PC),A0
0'001038  4E48                 101.          TRAP #NETCNF
0'00103A  103C 0001            102.          MOVE.B #SLV0,D0
0'00103E  4E49                 103.          TRAP #SRQSRVC
0'001040  41FA ****            104.          LEA S0PRO(PC),A0         ;STRT ADDR
0'001044  43FA ****            105.          LEA S1PRO(PC),A1         ;END ADDR+1
0'001048  6100 01EA            106.          BSR SENDOUT              ;ADDR,COUNT,CODE
                               107.  *
                               108.  * CONFIGURE NETWORK TO MASTER TO SLAVE 1
                               109.  * RESPOND TO SLAVE 1'S RFP SRQ, AND LOAD ITS
                               110.  * PROGRAM
                               111.  *
0'00104C  13FC 00B4            112.          MOVE.B #$B4,PSR01
          00012FB0
0'001054  103C 0002            113.          MOVE.B #SLV1,D0
0'001058  4E49                 114.          TRAP #SRQSRVC
0'00105A  41FA ****            115.          LEA S1PRO(PC),A0
0'00105E  43FA ****            116.          LEA S2PRO(PC),A1
0'001062  6100 01D0            117.          BSR SENDOUT
                               118.  *
                               119.  * CONFIGURE NETWORK TO MASTER TO SLAVE 2
                               120.  * RESPOND TO SLAVE 2'S RFP SRQ, AND LOAD ITS
                               121.  * PROGRAM
                               122.  *
0'001066  13FC 00B4            123.          MOVE.B #$B4,PSR45
          00012FB2
0'00106E  103C 0004            124.          MOVE.B #SLV2,D0
0'001072  4E49                 125.          TRAP #SRQSRVC
0'001074  41FA ****            126.          LEA S2PRO(PC),A0
0'001078  43FA ****            127.          LEA S3PRO(PC),A1
0'00107C  6100 01B6            128.          BSR SENDOUT
                               129.  *
                               130.  * CONFIGURE NETWORK TO MASTER TO SLAVE 3
                               131.  * RESPOND TO SLAVE 3'S RFP SRQ, AND LOAD ITS
                               132.  * PROGRAM
                               133.  *
0'001080  13FC 00B4            134.          MOVE.B #$B4,PSR89
          00012FB4
0'001088  103C 0008            135.          MOVE.B #SLV3,D0
0'00108C  4E49                 136.          TRAP #SRQSRVC
0'00108E  41FA ****            137.          LEA S3PRO(PC),A0
0'001092  43FA ****            138.          LEA DONE(PC),A1
0'001096  6100 019C            139.          BSR SENDOUT
                               140.  *
                               141.  * COPY INPUT DATA ARRAY TO DATAII SO EACH RUN
                               142.  * HAS SAME INPUT DATA
                               143.  *
0'00109A  41FA EF64            144.  REEXE    LEA DATAIN(PC),A0    ;SOURCE
0'00109E  43FA F360            145.          LEA DATAII(PC),A1    ;DEST
0'0010A2  323C 00FF            146.          MOVE.W #$FF,D1
0'0010A6  22D8                 147.  MVLP     MOVE.L (A0)+,(A1)+
0'0010A8  51C9 FFFC            148.          DBRA D1,MVLP
                               149.  *
                               150.  * ALL PROGRAMS LOADED, EACH ASSERTING READY FOR
                               151.  * DATA SRQ.  RESPOND TO ALL, THEN LOAD THE DATA
                               152.  * IN BROADCAST MODE.
```

```
                          153.  *
                          154.  * LOAD TIMER PRE-LOAD REGISTER HERE
                          155.  *
0'0010AC  297C 00FFFFFF   156.          MOVE.L #CNTSTRT,2(A4)
          0002
                          157.  *
                          158.  * SET UP SOME REGISTER VALUES
                          159.  *
0'0010B4  3C3C 0100       160.          MOVE.W #NPTE*4,D6
0'0010B8  267C 000003FC   161.          MOVEA.L #$3FC,A3
0'0010BE  247C 00000010   162.          MOVEA.L #$10,A2
0'0010C4  2C7C 00012FC0   163.          MOVEA.L #PGCR,A6
0'0010CA  2A7C 00012FB0   164.          MOVEA.L #PSR01,A5
0'0010D0  4243            165.          CLR.W D3
0'0010D2  41FA FF36       166.          LEA BRDCAST(PC),A0
0'0010D6  4E48            167.          TRAP #NETCNF
0'0010D8  103C 000F       168.          MOVE.B #ALLSLV,D0
0'0010DC  4E49            169.          TRAP #SRQSRVC
0'0010DE  41FA F320       170.          LEA DATAII(PC),A0        ;START OF DATA
0'0010E2  43FA F71C       171.          LEA FILTER(PC),A1        ;END OF DATA+1
                          172.  *
                          173.  * START TIMER COUNTING HERE
                          174.  *
0'0010E6  18BC 0001       175.          MOVE.B #ENABLTM,(A4)
0'0010EA  6100 0148       176.          BSR SENDOUT
                          177.  *
                          178.  * WHEN HERE, ALL SLAVES LOADED, WILL ASSERT
                          179.  * READY TO EXECUTE SRQ.  RESPOND TO LET THEM
                          180.  * CONTINUE.  NETWORK IS CONFIGURED TO ACCEPT
                          181.  * DATA FROM SLAVE 0.
                          182.  *
0'0010EE  700F            183.          MOVEQ.L #ALLSLV,D0
0'0010F0  4E49            184.          TRAP #SRQSRVC            ,SLAVES EXECUTE
                          185.  *
                          186.  * SLAVES ASSERT SRQ, THEN UPLOAD DATA
                          187.  * WHEN ACKNOWLEDGED. ACKNOWLEDGE ALL THEN
                          188.  * UPLOAD FROM EACH
                          189.  *
0'0010F2  700F            190.  REDIST  MOVEQ.L #ALLSLV,D0
0'0010F4  4E49            191.          TRAP #SRQSRVC
0'0010F6  3006            192.          MOVE.W D6,D0        ;BYTE COUNT
0'0010F8  41FA FB06       193.          LEA XDATA0(PC),A0       ;DEST ADDRESS
0'0010FC  1CBC 0010       194.          MOVE.B #ENABLEA,(A6)
0'001100  4E45            195.          TRAP #INDATA
0'001102  4216            196.          CLR.B (A6)
                          197.  *
                          198.  * CONFIGURE FOR SLAVE #1 TO MASTER,
                          199.  * AND GET DATA
                          200.  *
0'001104  1B7C 00BC 0005  201.          MOVE.B #$BC,5(A5)
0'00110A  3006            202.          MOVE.W D6,D0
0'00110C  41FA FBF2       203.          LEA XDATA1(PC),A0
0'001110  1CBC 0010       204.          MOVE.B #ENABLEA,(A6)
0'001114  4E45            205.          TRAP #INDATA
0'001116  4216            206.          CLR.B (A6)
                          207.  *
                          208.  * CONFIGURE FOR SLAVE #2 TO MASTER,
                          209.  * AND GET DATA
                          210.  *
0'001118  1B7C 00BB 0007  211.          MOVE.B #$BB,7(A5)
0'00111E  3006            212.          MOVE.W D6,D0
0'001120  41FA FCDE       213.          LEA XDATA2(PC),A0
0'001124  1CBC 0010       214.          MOVE.B #ENABLEA,(A6)
0'001128  4E45            215.          TRAP #INDATA
0'00112A  4216            216.          CLR.B (A6)
                          217.  *
                          218.  * CONFIGURE FOR SLAVE #3 TO MASTER,
                          219.  * AND GET DATA
                          220.  *
0'00112C  1B7C 00BB 0009  221.          MOVE.B #$BB,9(A5)
0'001132  3006            222.          MOVE.W D6,D0
0'001134  41FA FDCA       223.          LEA XDATA3(PC),A0
0'001138  1CBC 0010       224.          MOVE.B #ENABLEA,(A6)
0'00113C  4E45            225.          TRAP #INDATA
0'00113E  4216            226.          CLR.B (A6)
                          227.  *
                          228.  * HAVE ALL DATA IN, NOW DO REARRANGE FROM THE
                          229.  * XDATAi ARRAY TO THE DATAII ARRAY FOR CORRECT
                          230.  * ORDER.
                          231.  *
0'001140  41FA FABE       232.  REORDER LEA XDATA0(PC),A0
```

```
0'001144  43FA F2BA       233.          LEA DATAII(PC),A1
0'001148  7203            234.          MOVEQ.L #NPE-1,D1
0'00114A  7A3F            235.  RLOOPT  MOVEQ.L #NPTE-1,D5
0'00114C  2298            236.  RLOOPN  MOVE.L (A0)+,(A1)
0'00114E  D3CA            237.          ADDA.L A2,A1
0'001150  51CD FFFA       238.          DBRA D5,RLOOPN
0'001154  93CB            239.          SUBA.L A3,A1
0'001156  51C9 FFF2       240.          DBRA D1,RLOOPT
                          241.  *
                          242.  * forward fft and filtering is done
                          243.  * now redistribute the frequency domain data
                          244.  * to the slaves for the inverse transform.
                          245.  * simply do again!
                          246.  *
0'00115A  5243            247.          ADDQ.W #1,D3
0'00115C  0C43 0002       248.          CMPI.W #2,D3
0'001160  67 2C           249.          BEQ.S SKIP
0'001162  41FA FEA6       250.          LEA BRDCAST(PC),A0
0'001166  4E48            251.          TRAP #NETCNF
0'001168  103C 000F       252.          MOVE.B #ALLSLV,D0
0'00116C  4E49            253.          TRAP #SRQSRVC
0'00116E  41FA F290       254.          LEA DATAII(PC),A0       ;START OF DATA
0'001172  43FA F68C       255.          LEA FILTER(PC),A1
0'001176  93C8            256.          SUBA.L A0,A1
0'001178  3009            257.          MOVE.W A1,D0
0'00117A  13FC 0020       258.          MOVE.B #ENABLEB,PGCR
          00012FC0
0'001182  4E46            259.          TRAP #OUTDATA
0'001184  4239 00012FC0   260.          CLR.B PGCR
0'00118A  6000 FF66       261.          BRA REDIST
                          262.  *
                          263.  * PROGRAM IS DONE, CAN STOP TIMER, AND
                          264.  * OUTPUT NUMBER OF 32 CLOCK INTERVALS
                          265.  * PASSED. FIRST DISABLE TIMER
                          266.  *
0'00118E  4214            267.  SKIP    CLR.B (A4)
0'001190  222C 0006       268.          MOVE.L 6(A4),D1 ;GET NEW COUNT
0'001194  203C 00FFFFFF   269.          MOVE.L #CNTSTRT,D0
0'00119A  9081            270.          SUB.L D1,D0     ;GET ELAPSED COUNT
                          271.  *
                          272.  * OUTPUT CR, LF, THEN THE COUNT IN HEX
                          273.  *
0'00119C  4E4F            274.          TRAP #15
0'00119E  000A            275.          DC.W SENCLFC
0'0011A0  4E4F            276.          TRAP #15
0'0011A2  0018            277.          DC.W DISBUF8
0'0011A4  2F00            278.          MOVE.L D0,-(SP)
0'0011A6  41FA 0119       279.          LEA SPC(PC),A0
0'0011AA  4E4F            280.          TRAP #15
0'0011AC  0011            281.          DC.W OUTMESC
0'0011AE  2007            282.          MOVE.L D7,D0
0'0011B0  4E4F            283.          TRAP #15
0'0011B2  0018            284.          DC.W DISBUF8
0'0011B4  201F            285.          MOVE.L (SP)+,D0
                          286.  *
                          287.  * NOW KEEP TRACK OF STATISTICS
                          288.  *
0'0011B6  D197            289.          ADD.L D0,(SP)   ;ADD TO SUM
0'0011B8  B0AF 0004       290.          CMP.L 4(SP),D0  ;CHECK MAX
0'0011BC  6F 04           291.          BLE.S NOTMAX    ;D0 NOT MAX
0'0011BE  2F40 0004       292.          MOVE.L D0,4(SP) ;NEW MAX
0'0011C2  B0AF 0008       293.  NOTMAX  CMP.L 8(SP),D0  ;CHECK MIN
0'0011C6  6C 04           294.          BGE.S NOTMIN    ;D0 NOT MIN
0'0011C8  2F40 0008       295.          MOVE.L D0,8(SP) ;NEW MIN
0'0011CC  51CF FECC       296.  NOTMIN  DBRA D7,REEXE   ;DO D7 TIMES
                          297.
                          298.  *
                          299.  * WHEN HERE, PROGRAM HAS BEEN EXECUTED 256 TIMES
                          300.  * SEND AN ILLEGAL ADDRESS FOR DATA, FORCING
                          301.  * SLAVES TO PROGRAM ACCEPT MODE
                          302.  *
0'0011D0  41FA FE38       303.          LEA BRDCAST(PC),A0
0'0011D4  4E48            304.          TRAP #NETCNF
0'0011D6  103C 000F       305.          MOVE.B #ALLSLV,D0
0'0011DA  4E49            306.          TRAP #SRQSRVC
0'0011DC  207C FFFF0000   307.          MOVEA.L #$FFFF0000,A0   ;ILLEGAL ADDR
0'0011E2  13FC 0020       308.          MOVE.B #ENABLEB,PGCR
          00012FC0
0'0011EA  4E47            309.          TRAP #ADBYTOT   ;THAT'LL FIX 'EM
0'0011EC  4239 00012FC0   310.          CLR.B PGCR
                          311.  *
```

```
                                312.  * OUTPUT STATISTICS
                                313.  *
0'0011F2  4E4F                  314.          TRAP #15
0'0011F4  000A                  315.          DC.W SENCLFC
0'0011F6  4E4F                  316.          TRAP #15
0'0011F8  000A                  317.          DC.W SENCLFC
0'0011FA  41FA 0050             318.          LEA AVEMES(PC),A0
0'0011FE  4E4F                  319.          TRAP #15
0'001200  0011                  320.          DC.W OUTMESC
0'001202  201F                  321.          MOVE.L (SP)+,D0 ;GET SUM
0'001204  E080                  322.          ASR.L #DIV256,D0        ;DO AVERAGE
0'001206  4E4F                  323.          TRAP #15
0'001208  0018                  324.          DC.W DISBUF8    ;OUTPUT AVERAGE
0'00120A  4E4F                  325.          TRAP #15
0'00120C  000A                  326.          DC.W SENCLFC
0'00120E  41FA 0063             327.          LEA MAXMES(PC),A0
0'001212  4E4F                  328.          TRAP #15
0'001214  0011                  329.          DC.W OUTMESC
0'001216  201F                  330.          MOVE.L (SP)+,D0 ;GET MAX
0'001218  4E4F                  331.          TRAP #15
0'00121A  0018                  332.          DC.W DISBUF8
0'00121C  4E4F                  333.          TRAP #15
0'00121E  000A                  334.          DC.W SENCLFC
0'001220  41FA 0078             335.          LEA MINMES(PC),A0
0'001224  4E4F                  336.          TRAP #15
0'001226  0011                  337.          DC.W OUTMESC
0'001228  201F                  338.          MOVE.L (SP)+,D0 ;GET MIN
0'00122A  4E4F                  339.          TRAP #15
0'00122C  0018                  340.          DC.W DISBUF8
0'00122E  4E4F                  341.          TRAP #15
0'001230  000A                  342.          DC.W SENCLFC
                                343.  *
                                344.  * AFTER 256 RUNS, LOAD THE SECOND SLAVE
                                345.  * PROGRAM TO ACQUIRE THE TIMING DATA
                                346.  * IN THE SLAVES TIMER COUNT REGISTERS
                                347.  *
0'001232  4E75                  348.          RTS
                                349.  *
                                350.  * SUBROUTINE SENDOUT
                                351.  *
0'001234  93C8                  352.  SENDOUT SUBA.L A0,A1
0'001236  3009                  353.          MOVE.W A1,D0
0'001238  13FC 0020             354.          MOVE.B #ENABLEB,PGCR
          00012FC0
0'001240  4E47                  355.          TRAP #ADBYTOT
0'001242  4E46                  356.          TRAP #OUTDATA
0'001244  4239 00012FC0         357.          CLR.B PGCR
0'00124A  4E75                  358.          RTS
0'00124C  41 76 65 72 61        359.  AVEMES  DC.B 'Average time ( x 4 for microseconds)= ',EOT
          67 65 20 74 69
          6D 65 20 28 20
          78 20 34 20 66
          6F 72 20 6D 69
          63 72 6F 73 65
          63 6F 6E 64 73
          29 3D 20 04
0'001273  4D 61 78 69 6D        360.  MAXMES  DC.B 'Maximum time ( x 4 for microseconds)= ',EOT
          75 6D 20 74 69
          6D 65 20 28 20
          78 20 34 20 66
          6F 72 20 6D 69
          63 72 6F 73 65
          63 6F 6E 64 73
          29 3D 20 04
0'00129A  4D 69 6E 69 6D        361.  MINMES  DC.B 'Minimum time ( x 4 for microseconds)= ',EOT
          75 6D 20 74 69
          6D 65 20 28 20
          78 20 34 20 66
          6F 72 20 6D 69
          63 72 6F 73 65
          63 6F 6E 64 73
          29 3D 20 04
0'0012C1  20 20 20 20 04        362.  SPC     DC.B '    ',EOT
0'0012C6                        363.          END
    0 Errors
```

```
*
* LINK SPEC FOR PFILNIM2
*
      LINK PFILNOM2
      LINK SLVNOMES
      ORG $1000
      SECTION 0,15
      END
```

M.5 Program PFILNOM3: Multicomputer Frequency Domain Filtering Program #3A

```
                                1.          TTL PARFIL3
                                2.  *
                                3.  * This program performs a 256 complex point
                                4.  * frequency domain filtering operation on an
                                5.  * input signal at DATAII.  The filter
                                6.  * coefficients are at FILTER, and are
                                7.  * distributed among slaves using another
                                8.  * program (filter.a68).
                                9.  *
                               10.  * input/output data is stored as real/imaginary
                               11.  * pair, each component 16-bits long.
                               12.  * twiddle factors within slave programs are
                               13.  * stored similarly
                               14.  *
                               15.  *
                               16.  * self timed operation, 256 runs
                               17.  *
                               18.  * EQUATES
0'000000                       19.          SECTION 0
                               20.          XDEF PGCR,ENABLEA,ENABLEB
                               21.          XDEF NPT,NPTE,NPE,SHIFT14
                               22.          XDEF TCR,CNTSTRT,ENABLTM
                               23.          XREF S0PRO,S1PRO,S2PRO,S3PRO,DONE
                               24.          XDEF DATAII,FILTER,BRDCAST
                               25.  *
     # 00012FC0                26.  PGCR     EQU $12FC0
     # 00000010                27.  ENABLEA  EQU $10
     # 00000020                28.  ENABLEB  EQU $20
     # 00000100                29.  NPT      EQU $100
     # 00000040                30.  NPTE     EQU $40
     # 00000004                31.  NPE      EQU 4
     # 0000000E                32.  SHIFT14  EQU 14
     # 00000003                33.  SRQASRT  EQU 3
     # 00000004                34.  ADBYTIN  EQU 4
     # 00000007                35.  ADBYTOT  EQU 7
     # 00000005                36.  INDATA   EQU 5
     # 00000006                37.  OUTDATA  EQU 6
     # 00000008                38.  NETCNF   EQU 8
     # 00000009                39.  SRQSRVC  EQU 9
     # 0000000E                40.  ABORT    EQU 14
     # 00000001                41.  SLV0     EQU $01
     # 00000002                42.  SLV1     EQU $02
     # 00000004                43.  SLV2     EQU $04
     # 00000008                44.  SLV3     EQU $08
     # 0000000F                45.  ALLSLV   EQU $0F
     # 00000006                46.  NUGAM    EQU 6
     # 00012FB0                47.  PSR01    EQU $12FB0
     # 00012FB1                48.  PSR23    EQU $12FB1
     # 00012FB2                49.  PSR45    EQU $12FB2
     # 00012FB3                50.  PSR67    EQU $12FB3
     # 00012FB4                51.  PSR89    EQU $12FB4
     # 00012FB5                52.  PSR1011  EQU $12FB5
     # 00012FB6                53.  PSR1213  EQU $12FB6
     # 00012FB7                54.  PSR1415  EQU $12FB7
     # 00012FB8                55.  PSR1617  EQU $12FB8
     # 00012FB9                56.  PSR1819  EQU $12FB9
     # 00012FD0                57.  TCR      EQU $12FD0
     # 00012FD2                58.  TPLR     EQU TCR+2
     # 00012FD6                59.  TCNTR    EQU TCR+6
     # 0000000A                60.  SENCLFC  EQU 10
     # 00000018                61.  DISBUF8  EQU 24
     # 0000001A                62.  DISBUF2  EQU 26
     # 00FFFFFF                63.  CNTSTRT  EQU $00FFFFFF
     # 00000001                64.  ENABLTM  EQU 1
     # 00000011                65.  OUTMESC  EQU 17
     # 00000008                66.  DIV256   EQU 8
     # 00000100                67.  PASSCNT  EQU 256
     # 00000004                68.  EOT      EQU 4
                               69.  ****************************************
                               70.  * THIS VERSION WIATS FOR SLAVE 0 SYNC,
                               71.  * THEN UPLOADS, THEN WAITS FOR ALL OF
                               72.  * SLAVE 1,2,3 SYNCS THEN UPLOADS
                               73.  * INDIVIDUALLY IN ORDER 1,2,3
                               74.  ****************************************
                               75.  ******************
                               76.  * MASTER PROGRAM *
```

```
                                77.  ********************
                                78.  *
0'000000  <400>                 79.  DATAIN  DS.B $400
0'000400  <400>                 80.  DATAII  DS.B $400
0'000800  <400>                 81.  FILTER  DS.B $400
0'000C00  <100>                 82.  XDATA0  DS.B $100
0'000D00  <100>                 83.  XDATA1  DS.B $100
0'000E00  <100>                 84.  XDATA2  DS.B $100
0'000F00  <100>                 85.  XDATA3  DS.B $100
0'001000  BB BB BB CB BB        86.  INITCNF DC.B $BB,$BB,$BB,$CB,$BB,$8C,$CB,$8B,$B8,$8B
          8C CB 8B B8 8B
0'00100A  BD BB BD CB BD        87.  BRDCAST DC.B $BD,$BB,$BD,$CB,$BD,$8C,$CB,$8B,$B8,$8B
          8C CB 8B B8 8B
                                88.  *
                                89.  * CONFIGURE NETWORK TO INITIAL STATE, MASTER
                                90.  * TO SLAVE 0
                                91.  * RESPOND TO SLAVE 0'S RFP SRQ, AND LOAD
                                92.  * ITS PROGRAM
                                93.  *
0'001014  287C 00012FD0         94.  STRTFFT MOVEA.L #TCR,A4          ;POINTER TIMER
0'00101A  4214                  95.          CLR.B (A4)               ;DISABLE F'SURE
0'00101C  2F3C 00FFFFFF         96.          MOVE.L #CNTSTRT,-(SP)    ;MIN TIME
0'001022  2F3C 00000000         97.          MOVE.L #$00,-(SP)        ;MAX TIME
0'001028  2F3C 00000000         98.          MOVE.L #$00,-(SP)        ;RUN'G SUM TIME
0'00102E  2E3C 000000FF         99.          MOVE.L #PASSCNT-1,D7     ;COUNTER
0'001034  41FA FFCA            100.          LEA INITCNF(PC),A0
0'001038  4E48                 101.          TRAP #NETCNF
0'00103A  103C 0001            102.          MOVE.B #SLV0,D0
0'00103E  4E49                 103.          TRAP #SRQSRVC
0'001040  41FA ****            104.          LEA S0PRO(PC),A0         ;STRT ADDR
0'001044  43FA ****            105.          LEA S1PRO(PC),A1         ;END ADDR+1
0'001048  6100 01EE            106.          BSR SENDOUT              ;ADDR,COUNT,CODE
                               107.  *
                               108.  * CONFIGURE NETWORK TO MASTER TO SLAVE 1
                               109.  * RESPOND TO SLAVE 1'S RFP SRQ, AND LOAD ITS
                               110.  * PROGRAM
                               111.  *
0'00104C  13FC 00B4            112.          MOVE.B #$B4,PSR01
          00012FB0
0'001054  103C 0002            113.          MOVE.B #SLV1,D0
0'001058  4E49                 114.          TRAP #SRQSRVC
0'00105A  41FA ****            115.          LEA S1PRO(PC),A0
0'00105E  43FA ****            116.          LEA S2PRO(PC),A1
0'001062  6100 01D4            117.          BSR SENDOUT
                               118.  *
                               119.  * CONFIGURE NETWORK TO MASTER TO SLAVE 2
                               120.  * RESPOND TO SLAVE 2'S RFP SRQ, AND LOAD ITS
                               121.  * PROGRAM
                               122.  *
0'001066  13FC 00B4            123.          MOVE.B #$B4,PSR45
          00012FB2
0'00106E  103C 0004            124.          MOVE.B #SLV2,D0
0'001072  4E49                 125.          TRAP #SRQSRVC
0'001074  41FA ****            126.          LEA S2PRO(PC),A0
0'001078  43FA ****            127.          LEA S3PRO(PC),A1
0'00107C  6100 01BA            128.          BSR SENDOUT
                               129.  *
                               130.  * CONFIGURE NETWORK TO MASTER TO SLAVE 3
                               131.  * RESPOND TO SLAVE 3'S RFP SRQ, AND LOAD ITS
                               132.  * PROGRAM
                               133.  *
0'001080  13FC 00B4            134.          MOVE.B #$B4,PSR89
          00012FB4
0'001088  103C 0008            135.          MOVE.B #SLV3,D0
0'00108C  4E49                 136.          TRAP #SRQSRVC
0'00108E  41FA ****            137.          LEA S3PRO(PC),A0
0'001092  43FA ****            138.          LEA DONE(PC),A1
0'001096  6100 01A0            139.          BSR SENDOUT
                               140.  *
                               141.  * COPY INPUT DATA ARRAY TO DATAII SO EACH RUN
                               142.  * HAS SAME INPUT DATA
                               143.  *
0'00109A  41FA EF64            144.  REEXE   LEA DATAIN(PC),A0    ;SOURCE
0'00109E  43FA F360            145.          LEA DATAII(PC),A1    ;DEST
0'0010A2  323C 00FF            146.          MOVE.W #$FF,D1
0'0010A6  22D8                 147.  MVLP    MOVE.L (A0)+,(A1)+
0'0010A8  51C9 FFFC            148.          DBRA D1,MVLP
                               149.  *
                               150.  * ALL PROGRAMS LOADED, EACH ASSERTING READY FOR
                               151.  * DATA SRQ.  RESPOND TO ALL, THEN LOAD THE DATA
                               152.  * IN BROADCAST MODE.
```

```
                                153.  *
                                154.  * LOAD TIMER PRE-LOAD REGISTER HERE
                                155.  *
0'0010AC  297C 00FFFFFF         156.          MOVE.L #CNTSTRT,2(A4)
          0002
                                157.  *
                                158.  * SET UP SOME REGISTER VALUES
                                159.  *
0'0010B4  3C3C 0100             160.          MOVE.W #NPTE*4,D6
0'0010B8  267C 000003FC         161.          MOVEA.L #$3FC,A3
0'0010BE  247C 00000010         162.          MOVEA.L #$10,A2
0'0010C4  2C7C 00012FC0         163.          MOVEA.L #PGCR,A6
0'0010CA  2A7C 00012FB0         164.          MOVEA.L #PSR01,A5
0'0010D0  4243                  165.          CLR.W D3
0'0010D2  41FA FF36             166.          LEA BRDCAST(PC),A0
0'0010D6  4E48                  167.          TRAP #NETCNF
0'0010D8  103C 000F             168.          MOVE.B #ALLSLV,D0
0'0010DC  4E49                  169.          TRAP #SRQSRVC
0'0010DE  41FA F320             170.          LEA DATAII(PC),A0       ;START OF DATA
0'0010E2  43FA F71C             171.          LEA FILTER(PC),A1       ;END OF DATA+1
                                172.  *
                                173.  * START TIMER COUNTING HERE
                                174.  *
0'0010E6  18BC 0001             175.          MOVE.B #ENABLTM,(A4)
0'0010EA  6100 014C             176.          BSR SENDOUT
                                177.  *
                                178.  * WHEN HERE, ALL SLAVES LOADED, WILL ASSERT
                                179.  * READY TO EXECUTE SRQ.  RESPOND TO LET THEM
                                180.  * CONTINUE.  NETWORK IS CONFIGURED TO ACCEPT
                                181.  * DATA FROM SLAVE 0.
                                182.  *
0'0010EE  700F                  183.          MOVEQ.L #ALLSLV,D0
0'0010F0  4E49                  184.          TRAP #SRQSRVC           ;SLAVES EXECUTE
                                185.  *
                                186.  * SLAVES ASSERT SRQ, THEN UPLOAD DATA
                                187.  * WHEN ACKNOWLEDGED. ACKNOWLEDGE AND LOAD
                                188.  * SLAVE 0, LOAD, THEN SLAVES 1,2,3
                                189.  * THEN UPLOAD FROM EACH
                                190.  *
0'0010F2  7001                  191.  REDIST  MOVEQ.L #SLV0,D0
0'0010F4  4E49                  192.          TRAP #SRQSRVC
0'0010F6  3006                  193.          MOVE.W D6,D0        ;BYTE COUNT
0'0010F8  41FA FB06             194.          LEA XDATA0(PC),A0       ;DEST ADDRESS
0'0010FC  1CBC 0010             195.          MOVE.B #ENABLEA,(A6)
0'001100  4E45                  196.          TRAP #INDATA
0'001102  4216                  197.          CLR.B (A6)
                                198.  *
                                199.  * CONFIGURE FOR SLAVE #1 TO MASTER,
                                200.  * AND GET DATA
                                201.  *
0'001104  700E                  202.          MOVEQ.L #$0E,D0
0'001106  4E49                  203.          TRAP #SRQSRVC
0'001108  1B7C 00BC 0005        204.          MOVE.B #$BC,5(A5)
0'00110E  3006                  205.          MOVE.W D6,D0
0'001110  41FA FBEE             206.          LEA XDATA1(PC),A0
0'001114  1CBC 0010             207.          MOVE.B #ENABLEA,(A6)
0'001118  4E45                  208.          TRAP #INDATA
0'00111A  4216                  209.          CLR.B (A6)
                                210.  *
                                211.  * CONFIGURE FOR SLAVE #2 TO MASTER,
                                212.  * AND GET DATA
                                213.  *
0'00111C  1B7C 00BB 0007        214.          MOVE.B #$BB,7(A5)
0'001122  3006                  215.          MOVE.W D6,D0
0'001124  41FA FCDA             216.          LEA XDATA2(PC),A0
0'001128  1CBC 0010             217.          MOVE.B #ENABLEA,(A6)
0'00112C  4E45                  218.          TRAP #INDATA
0'00112E  4216                  219.          CLR.B (A6)
                                220.  *
                                221.  * CONFIGURE FOR SLAVE #3 TO MASTER,
                                222.  * AND GET DATA
                                223.  *
0'001130  1B7C 00BB 0009        224.          MOVE.B #$BB,9(A5)
0'001136  3006                  225.          MOVE.W D6,D0
0'001138  41FA FDC6             226.          LEA XDATA3(PC),A0
0'00113C  1CBC 0010             227.          MOVE.B #ENABLEA,(A6)
0'001140  4E45                  228.          TRAP #INDATA
0'001142  4216                  229.          CLR.B (A6)
                                230.  *
                                231.  * HAVE ALL DATA IN, NOW DO REARRANGE FROM THE
                                232.  * XDATAi ARRAY TO THE DATAII ARRAY FOR CORRECT
```

```
                                    233.  * ORDER.
                                    234.  *
0'001144  41FA FABA                 235.  REORDER LEA XDATA0(PC),A0
0'001148  43FA F2B6                 236.         LEA DATAII(PC),A1
0'00114C  7203                      237.         MOVEQ.L #NPE-1,D1
0'00114E  7A3F                      238.  RLOOPT  MOVEQ.L #NPTE-1,D5
0'001150  2298                      239.  RLOOPN  MOVE.L (A0)+,(A1)
0'001152  D3CA                      240.         ADDA.L A2,A1
0'001154  51CD FFFA                 241.         DBRA D5,RLOOPN
0'001158  93CB                      242.         SUBA.L A3,A1
0'00115A  51C9 FFF2                 243.         DBRA D1,RLOOPT
                                    244.  *
                                    245.  * forward fft and filtering is done
                                    246.  * now redistribute the frequency domain data
                                    247.  * to the slaves for the inverse transform.
                                    248.  * simply do again!
                                    249.  *
0'00115E  5243                      250.         ADDQ.W #1,D3
0'001160  0C43 0002                 251.         CMPI.W #2,D3
0'001164  67 2C                     252.         BEQ.S SKIP
0'001166  41FA FEA2                 253.         LEA BRDCAST(PC),A0
0'00116A  4E48                      254.         TRAP #NETCNF
0'00116C  103C 000F                 255.         MOVE.B #ALLSLV,D0
0'001170  4E49                      256.         TRAP #SRQSRVC
0'001172  41FA F28C                 257.         LEA DATAII(PC),A0       ;START OF DATA
0'001176  43FA F688                 258.         LEA FILTER(PC),A1
0'00117A  93C8                      259.         SUBA.L A0,A1
0'00117C  3009                      260.         MOVE.W A1,D0
0'00117E  13FC 0020                 261.         MOVE.B #ENABLEB,PGCR
          00012FC0
0'001186  4E46                      262.         TRAP #OUTDATA
0'001186  4239 00012FC0             263.         CLR.B PGCR
0'00118E  6000 FF62                 264.         BRA REDIST
                                    265.  *
                                    266.  * PROGRAM IS DONE, CAN STOP TIMER, AND
                                    267.  * OUTPUT NUMBER OF 32 CLOCK INTERVALS
                                    268.  * PASSED. FIRST DISABLE TIMER
                                    269.  *
0'001192  4214                      270.  SKIP    CLR.B (A4)
0'001194  222C 0006                 271.         MOVE.L 6(A4),D1 ;GET NEW COUNT
0'001198  203C 00FFFFFF             272.         MOVE.L #CNTSTRT,D0
0'00119E  9081                      273.         SUB.L D1,D0     ;GET ELAPSED COUNT
                                    274.  *
                                    275.  * OUTPUT CR, LF, THEN THE COUNT IN HEX
                                    276.  *
0'0011A0  4E4F                      277.         TRAP #15
0'0011A2  000A                      278.         DC.W SENCLFC
0'0011A4  4E4F                      279.         TRAP #15
0'0011A6  0018                      280.         DC.W DISBUF8
0'0011A8  2F00                      281.         MOVE.L D0,-(SP)
0'0011AA  41FA 0119                 282.         LEA SPC(PC),A0
0'0011AE  4E4F                      283.         TRAP #15
0'0011B0  0011                      284.         DC.W OUTMESC
0'0011B2  2007                      285.         MOVE.L D7,D0
0'0011B4  4E4F                      286.         TRAP #15
0'0011B6  0018                      287.         DC.W DISBUF8
0'0011B8  201F                      288.         MOVE.L (SP)+,D0
                                    289.  *
                                    290.  * NOW KEEP TRACK OF STATISTICS
                                    291.  *
0'0011BA  D197                      292.         ADD.L D0,(SP)    ;ADD TO SUM
0'0011BC  B0AF 0004                 293.         CMP.L 4(SP),D0   ;CHECK MAX
0'0011C0  6F 04                     294.         BLE.S NOTMAX     ;D0 NOT MAX
0'0011C2  2F40 0004                 295.         MOVE.L D0,4(SP)  ;NEW MAX
0'0011C6  B0AF 0008                 296.  NOTMAX  CMP.L 8(SP),D0   ;CHECK MIN
0'0011CA  6C 04                     297.         BGE.S NOTMIN     ;D0 NOT MIN
0'0011CC  2F40 0008                 298.         MOVE.L D0,8(SP)  ;NEW MIN
0'0011D0  51CF FEC8                 299.  NOTMIN  DBRA D7,REEXE    ;DO D7 TIMES
                                    300.
                                    301.  *
                                    302.  * WHEN HERE, PROGRAM HAS BEEN EXECUTED 256 TIMES
                                    303.  * SEND AN ILLEGAL ADDRESS FOR DATA, FORCING
                                    304.  * SLAVES TO PROGRAM ACCEPT MODE
                                    305.  *
0'0011D4  41FA FE34                 306.         LEA BRDCAST(PC),A0
0'0011D8  4E48                      307.         TRAP #NETCNF
0'0011DA  103C 000F                 308.         MOVE.B #ALLSLV,D0
0'0011DE  4E49                      309.         TRAP #SRQSRVC
0'0011E0  207C FFFF0000             310.         MOVEA.L #$FFFF0000,A0   ;ILLEGAL ADDR
0'0011E6  13FC 0020                 311.         MOVE.B #ENABLEB,PGCR
          00012FC0
```

```
0'0011EE  4E47             312.          TRAP #ADBYTOT   ;THAT'LL FIX 'EM
0'0011F0  4239 00012FC0    313.          CLR.B PGCR
                           314.  *
                           315.  * OUTPUT STATISTICS
                           316.  *
0'0011F6  4E4F             317.          TRAP #15
0'0011F8  000A             318.          DC.W SENCLFC
0'0011FA  4E4F             319.          TRAP #15
0'0011FC  000A             320.          DC.W SENCLFC
0'0011FE  41FA 0050        321.          LEA AVEMES(PC),A0
0'001202  4E4F             322.          TRAP #15
0'001204  0011             323.          DC.W OUTMESC
0'001206  201F             324.          MOVE.L (SP)+,D0 ;GET SUM
0'001208  E080             325.          ASR.L #DIV256,D0        ;DO AVERAGE
0'00120A  4E4F             326.          TRAP #15
0'00120C  0018             327.          DC.W DISBUF8    ;OUTPUT AVERAGE
0'00120E  4E4F             328.          TRAP #15
0'001210  000A             329.          DC.W SENCLFC
0'001212  41FA 0063        330.          LEA MAXMES(PC),A0
0'001216  4E4F             331.          TRAP #15
0'001218  0011             332.          DC.W OUTMESC
0'00121A  201F             333.          MOVE.L (SP)+,D0 ;GET MAX
0'00121C  4E4F             334.          TRAP #15
0'00121E  0018             335.          DC.W DISBUF8
0'001220  4E4F             336.          TRAP #15
0'001222  000A             337.          DC.W SENCLFC
0'001224  41FA 0078        338.          LEA MINMES(PC),A0
0'001228  4E4F             339.          TRAP #15
0'00122A  0011             340.          DC.W OUTMESC
0'00122C  201F             341.          MOVE.L (SP)+,D0 ;GET MIN
0'00122E  4E4F             342.          TRAP #15
0'001230  0018             343.          DC.W DISBUF8
0'001232  4E4F             344.          TRAP #15
0'001234  000A             345.          DC.W SENCLFC
                           346.  *
                           347.  * AFTER 256 RUNS, LOAD THE SECOND SLAVE
                           348.  * PROGRAM TO ACQUIRE THE TIMING DATA
                           349.  * IN THE SLAVES TIMER COUNT REGISTERS
                           350.  *
0'001236  4E75             351.          RTS
                           352.  *
                           353.  * SUBROUTINE SENDOUT
                           354.  *
0'001238  93C8             355.  SENDOUT SUBA.L A0,A1
0'00123A  3009             356.          MOVE.W A1,D0
0'00123C  13FC 0020        357.          MOVE.B #ENABLEB,PGCR
          00012FC0
0'001244  4E47             358.          TRAP #ADBYTOT
0'001246  4E46             359.          TRAP #OUTDATA
0'001248  4239 00012FC0    360.          CLR.B PGCR
0'00124E  4E75             361.          RTS
                           362.  *
0'001250  41 76 65 72 61   363.  AVEMES   DC.B 'Average time ( x 4 for microseconds)= ',EOT
          67 65 20 74 69
          6D 65 20 28 20
          78 20 34 20 66
          6F 72 20 6D 69
          63 72 6F 73 65
          63 6F 6E 64 73
          29 3D 20 04
0'001277  4D 61 78 69 6D   364.  MAXMES   DC.B 'Maximum time ( x 4 for microseconds)= ',EOT
          75 6D 20 74 69
          6D 65 20 28 20
          78 20 34 20 66
          6F 72 20 6D 69
          63 72 6F 73 65
          63 6F 6E 64 73
          29 3D 20 04
0'00129E  4D 69 6E 69 6D   365.  MINMES   DC.B 'Minimum time ( x 4 for microseconds)= ',EOT
          75 6D 20 74 69
          6D 65 20 28 20
          78 20 34 20 66
          6F 72 20 6D 69
          63 72 6F 73 65
          63 6F 6E 64 73
          29 3D 20 04
0'0012C5  20 20 20 20 04   366.  SPC      DC B '     ',EOT
                           367.  *
                           368.  *
0'0012CA                   369.          END
    0 Errors
```

```
*
* LINK SPEC FOR PFILNOM3
*
      LINK PFILNOM3
      LINK SLVNOMES
      ORG $1000
      SECTION 0,15
      END
```

M.6 Program PFILNOM4: Multicomputer Frequency Domain Filtering Program #4A

```
                                1.          TTL PARFIL4
                                2.  *
                                3.  * This program performs a 256 complex point
                                4.  * frequency domain filtering operation on an
                                5.  * input signal at DATAII.  The filter
                                6.  * coefficients are at FILTER, and are
                                7.  * distributed among slaves using another
                                8.  * program (filter.a68).
                                9.  *
                               10.  * input/output data is stored as real/imaginary
                               11.  * pair, each component 16-bits long.
                               12.  * twiddle factors within slave programs are
                               13.  * stored similarly
                               14.  *
                               15.  *
                               16.  * self timed operation, 256 runs
                               17.  *
                               18.  * EQUATES
0'000000                       19.          SECTION 0
                               20.          XDEF PGCR,ENABLEA,ENABLEB
                               21.          XDEF NPT,NPTE,NPE,SHIFT14
                               22.          XDEF TCR,CNTSTRT,ENABLTM
                               23.          XREF S0PRO,S1PRO,S2PRO,S3PRO,DONE
                               24.          XDEF DATAII,FILTER,BRDCAST
                               25.  *
# 00012FC0                     26.  PGCR     EQU $12FC0
# 00012FCC                     27.  PCDR     EQU $12FCC
# 00000010                     28.  ENABLEA  EQU $10
# 00000020                     29.  ENABLEB  EQU $20
# 00000100                     30.  NPT      EQU $100
# 00000040                     31.  NPTE     EQU $40
# 00000004                     32.  NPE      EQU 4
# 0000000E                     33.  SHIFT14  EQU 14
# 00000003                     34.  SRQASRT  EQU 3
# 00000004                     35.  ADBYTIN  EQU 4
# 00000007                     36.  ADBYTOT  EQU 7
# 00000005                     37.  INDATA   EQU 5
# 00000006                     38.  OUTDATA  EQU 6
# 00000008                     39.  NETCNF   EQU 8
# 00000009                     40.  SRQSRVC  EQU 9
# 0000000E                     41.  ABORT    EQU 14
# 00000001                     42.  SLV0     EQU $01
# 00000002                     43.  SLV1     EQU $02
# 00000004                     44.  SLV2     EQU $04
# 00000008                     45.  SLV3     EQU $08
# 0000000F                     46.  ALLSLV   EQU $0F
# 00000006                     47.  NUGAM    EQU 6
# 00012FB0                     48.  PSR01    EQU $12FB0
# 00012FB1                     49.  PSR23    EQU $12FB1
# 00012FB2                     50.  PSR45    EQU $12FB2
# 00012FB3                     51.  PSR67    EQU $12FB3
# 00012FB4                     52.  PSR89    EQU $12FB4
# 00012FB5                     53.  PSR1011  EQU $12FB5
# 00012FB6                     54.  PSR1213  EQU $12FB6
# 00012FB7                     55.  PSR1415  EQU $12FB7
# 00012FB8                     56.  PSR1617  EQU $12FB8
# 00012FB9                     57.  PSR1819  EQU $12FB9
# 00012FD0                     58.  TCR      EQU $12FD0
# 00012FD2                     59.  TPLR     EQU TCR+2
# 00012FD6                     60.  TCNTR    EQU TCR+6
# 0000000A                     61.  SENCLFC  EQU 10
# 00000018                     62.  DISBUF8  EQU 24
# 0000001A                     63.  DISBUF2  EQU 26
# 00FFFFFF                     64.  CNTSTRT  EQU $00FFFFFF
# 00000001                     65.  ENABLTM  EQU 1
# 00000011                     66.  OUTMESC  EQU 17
# 00000008                     67.  DIV256   EQU 8
# 00000100                     68.  PASSCNT  EQU 256
# 00000004                     69.  EOT      EQU 4
                               70.  ****************************************
                               71.  * THIS VERSION USES FIRST COME FIRST
                               72.  * SERVED STRATEGY FOR SYNC'S AND UPLOADS
                               73.  ****************************************
                               74.  ******************
                               75.  * MASTER PROGRAM *
                               76.  ******************
```

```
                                    77.  *
0'000000  <400>                     78.  DATAIN  DS.B $400
0'000400  <400>                     79.  DATAII  DS.B $400
0'000800  <400>                     80.  FILTER  DS.B $400
0'000C00  <100>                     81.  XDATA0  DS.B $100
0'000D00  <100>                     82.  XDATA1  DS.B $100
0'000E00  <100>                     83.  XDATA2  DS.B $100
0'000F00  <100>                     84.  XDATA3  DS.B $100
0'001000  BB BB BB CB BB            85.  INITCNF DC.B $BB,$BB,$BB,$CB,$BB,$8C,$CB,$8B,$B8,$8B
          8C CB 8B B8 8B
0'00100A  BD BB BD CB BD            86.  BRDCAST DC.B $BD,$BB,$BD,$CB,$BD,$8C,$CB,$8B,$B8,$8B
          8C CB 8B B8 8B
                                    87.  *
                                    88.  * CONFIGURE NETWORK TO INITIAL STATE, MASTER
                                    89.  * TO SLAVE 0
                                    90.  * RESPOND TO SLAVE 0'S RFP SRQ, AND LOAD
                                    91.  * ITS PROGRAM
                                    92.  *
0'001014  287C 00012FD0             93.  STRTFFT MOVEA.L #TCR,A4          ;POINTER TIMER
0'00101A  4214                      94.          CLR.B (A4)               ;DISABLE F'SURE
0'00101C  2F3C 00FFFFFF             95.          MOVE.L #CNTSTRT,-(SP)    ;MIN TIME
0'001022  2F3C 00000000             96.          MOVE.L #$00,-(SP)        ;MAX TIME
0'001028  2F3C 00000000             97.          MOVE.L #$00,-(SP)        ;RUN'G SUM TIME
0'00102E  2E3C 000000FF             98.          MOVE.L #PASSCNT-1,D7     ;COUNTER
0'001034  41FA FFCA                 99.          LEA INITCNF(PC),A0
0'001038  4E48                     100.          TRAP #NETCNF
0'00103A  103C 0001                101.          MOVE.B #SLV0,D0
0'00103E  4E49                     102.          TRAP #SRQSRVC
0'001040  41FA ****                103.          LEA S0PRO(PC),A0         ;STRT ADDR
0'001044  43FA ****                104.          LEA S1PRO(PC),A1         ;END ADDR+1
0'001048  6100 02B4                105.          BSR SENDOUT              ;ADDR,COUNT,CODE
                                   106.  *
                                   107.  * CONFIGURE NETWORK TO MASTER TO SLAVE 1
                                   108.  * RESPOND TO SLAVE 1'S RFP SRQ, AND LOAD ITS
                                   109.  * PROGRAM
                                   110.  *
0'00104C  13FC 00B4                111.          MOVE.B #$B4,PSR01
          00012FB0
0'001054  103C 0002                112.          MOVE.B #SLV1,D0
0'001058  4E49                     113.          TRAP #SRQSRVC
0'00105A  41FA ****                114.          LEA S1PRO(PC),A0
0'00105E  43FA ****                115.          LEA S2PRO(PC),A1
0'001062  6100 029A                116.          BSR SENDOUT
                                   117.  *
                                   118.  * CONFIGURE NETWORK TO MASTER TO SLAVE 2
                                   119.  * RESPOND TO SLAVE 2'S RFP SRQ, AND LOAD ITS
                                   120.  * PROGRAM
                                   121.  *
0'001066  13FC 00B4                122.          MOVE.B #$B4,PSR45
          00012FB2
0'00106E  103C 0004                123.          MOVE.B #SLV2,D0
0'001072  4E49                     124.          TRAP #SRQSRVC
0'001074  41FA ****                125.          LEA S2PRO(PC),A0
0'001078  43FA ****                126.          LEA S3PRO(PC),A1
0'00107C  6100 0280                127.          BSR SENDOUT
                                   128.  *
                                   129.  * CONFIGURE NETWORK TO MASTER TO SLAVE 3
                                   130.  * RESPOND TO SLAVE 3'S RFP SRQ, AND LOAD ITS
                                   131.  * PROGRAM
                                   132.  *
0'001080  13FC 00B4                133.          MOVE.B #$B4,PSR89
          00012FB4
0'001088  103C 0008                134.          MOVE.B #SLV3,D0
0'00108C  4E49                     135.          TRAP #SRQSRVC
0'00108E  41FA ****                136.          LEA S3PRO(PC),A0
0'001092  43FA ****                137.          LEA DONE(PC),A1
0'001096  6100 0266                138.          BSR SENDOUT
                                   139.  *
                                   140.  * COPY INPUT DATA ARRAY TO DATAII SO EACH RUN
                                   141.  * HAS SAME INPUT DATA
                                   142.  *
0'00109A  41FA EF64                143.  REEXE   LEA DATAIN(PC),A0    ;SOURCE
0'00109E  43FA F360                144.          LEA DATAII(PC),A1    ;DEST
0'0010A2  323C 00FF                145.          MOVE.W #$FF,D1
0'0010A6  22D8                     146.  MVLP    MOVE.L (A0)+,(A1)+
0'0010A8  51C9 FFFC                147.          DBRA D1,MVLP
                                   148.  *
                                   149.  * ALL PROGRAMS LOADED, EACH ASSERTING READY FOR
                                   150.  * DATA SRQ.  RESPOND TO ALL, THEN LOAD THE DATA
                                   151.  * IN BROADCAST MODE.
                                   152.  *
```

```
                                153.  * LOAD TIMER PRE-LOAD REGISTER HERE
                                154.  *
0'0010AC  297C 00FFFFFF         155.          MOVE.L #CNTSTRT,2(A4)
          0002
                                156.  *
                                157.  * SET UP SOME REGISTER VALUES
                                158.  *
0'0010B4  3C3C 0100             159.          MOVE.W #NPTE*4,D6
0'0010B8  267C 000003FC         160.          MOVEA.L #$3FC,A3
0'0010BE  247C 00000010         161.          MOVEA.L #$10,A2
0'0010C4  2C7C 00012FC0         162.          MOVEA.L #PGCR,A6
0'0010CA  2A7C 00012FB0         163.          MOVEA.L #PSR01,A5
0'0010D0  4243                  164.          CLR.W D3
0'0010D2  41FA FF36             165.          LEA BRDCAST(PC),A0
0'0010D6  4E48                  166.          TRAP #NETCNF
0'0010D8  103C 000F             167.          MOVE.B #ALLSLV,D0
0'0010DC  4E49                  168.          TRAP #SRQSRVC
0'0010DE  41FA F320             169.          LEA DATAII(PC),A0       ;START OF DATA
0'0010E2  43FA F71C             170.          LEA FILTER(PC),A1       ;END OF DATA+1
                                171.  *
                                172.  * START TIMER COUNTING HERE
                                173.  *
0'0010E6  18BC 0001             174.          MOVE.B #ENABLTM,(A4)
0'0010EA  6100 0212             175.          BSR SENDOUT
                                176.  *
                                177.  * WHEN HERE, ALL SLAVES LOADED, WILL ASSERT
                                178.  * READY TO EXECUTE SRQ.  RESPOND TO LET THEM
                                179.  * CONTINUE.  NETWORK IS CONFIGURED TO ACCEPT
                                180.  * DATA FROM SLAVE 0.
                                181.  *
0'0010EE  700F                  182.          MOVEQ.L #ALLSLV,D0
0'0010F0  4E49                  183.          TRAP #SRQSRVC           ;SLAVES EXECUTE
                                184.  *
                                185.  * SLAVES ASSERT SRQ, THEN UPLOAD DATA
                                186.  * WHEN ACKNOWLEDGED. ACKNOWLEDGE INDIVIDUALLY
                                187.  * THEN UPLOAD FROM EACH
                                188.  *
                                189.  *
                                190.  * WHEN DONE, SLAVES ASSERT SRQ, THEN UPLOAD DATA
                                191.  * WHEN ACKNOWLEDGED.  ACKNOWLEDGE FCFS, AND UPLOAD
                                192.  * FCFS
                                193.  *
0'0010F2  700F                  194.  REDIST  MOVEQ.L #ALLSLV,D0
0'0010F4  0800 0000             195.  TSTB0   BTST.L #0,D0
0'0010F8  67 44                 196.          BEQ.S TSTB1
0'0010FA  123C 0000             197.          MOVE.B #0,D1
0'0010FE  6100 0216             198.          BSR TSTSLV
0'001102  0801 0007             199.          BTST.L #7,D1    ;SERVICED?
0'001106  67 36                 200.          BEQ.S TSTB1     ;IF NOT GOTO SLV1
                                201.  *
                                202.  * SLAVE 0 SENSED AND ACKNOWLEDGED, GET DATA
                                203.  *
0'001108  1B7C 0080 0003        204.          MOVE.B #$80,3(A5)
0'00110E  1B7C 0080 0005        205.          MOVE.B #$80,5(A5)
0'001114  1B7C 0080 0007        206.          MOVE.B #$80,7(A5)
0'00111A  1B7C 0080 0009        207.          MOVE.B #$80,9(A5)
0'001120  3F00                  208.          MOVE.W D0,-(SP)
0'001122  3006                  209.          MOVE.W D6,D0        ;BYTE COUNT
0'001124  41FA FADA             210.          LEA XDATA0(PC),A0       ;DEST ADDRESS
0'001128  1CBC 0010             211.          MOVE.B #ENABLEA,(A6)
0'00112C  4E45                  212.          TRAP #INDATA
0'00112E  4216                  213.          CLR.B (A6)
0'001130  301F                  214.          MOVE.W (SP)+,D0
0'001132  0880 0000             215.          BCLR.L #0,D0    ;CLEAR THE BIT
0'001136  0C00 0000             216.          CMPI.B #0,D0
0'00113A  6700 00CE             217.          BEQ REORDER
0'00113E  0800 0001             218.  TSTB1   BTST.L #1,D0
0'001142  67 3E                 219.          BEQ.S TSTB2
0'001144  123C 0001             220.          MOVE.B #1,D1
0'001148  6100 01CC             221.          BSR TSTSLV
0'00114C  0801 0007             222.          BTST.L #7,D1    ;SERVICED?
0'001150  67 30                 223.          BEQ.S TSTB2     ;IF NOT GOTO SLV1
                                224.  *
                                225.  * SLAVE 1 SENSED AND ACKNOWLEDGED, GET DATA
                                226.  *
0'001152  1B7C 00B8 0005        227.          MOVE.B #$B8,5(A5)
0'001158  1B7C 0080 0007        228.          MOVE.B #$80,7(A5)
0'00115E  1B7C 0080 0009        229.          MOVE.B #$80,9(A5)
0'001164  3F00                  230.          MOVE.W D0,-(SP)
0'001166  3006                  231.          MOVE.W D6,D0        ;BYTE COUNT
0'001168  41FA FB96             232.          LEA XDATA1(PC),A0       ;DEST ADDRESS
```

```
0'00116C  1CBC 0010        233.          MOVE.B #ENABLEA,(A6)
0'001170  4E45             234.          TRAP #INDATA
0'001172  4216             235.          CLR.B (A6)
0'001174  301F             236.          MOVE.W (SP)+,D0
0'001176  0880 0001        237.          BCLR.L #1,D0     ;CLEAR THE BIT
0'00117A  0C00 0000        238.          CMPI.B #0,D0
0'00117E  6700 008A        239.          BEQ REORDER
0'001182  0800 0002        240.  TSTB2    BTST.L #2,D0
0'001186  67 3E            241.          BEQ.S TSTB3
0'001188  123C 0002        242.          MOVE.B #2,D1
0'00118C  6100 0188        243.          BSR TSTSLV
0'001190  0801 0007        244.          BTST.L #7,D1     ;SERVICED?
0'001194  67 30            245.          BEQ.S TSTB3      ;IF NOT GOTO SLV1
                           246.  *
                           247.  * SLAVE 2 SENSED AND ACKNOWLEDGED, GET DATA
                           248.  *
0'001196  1B7C 0080 0006   249.          MOVE.B #$80,6(A5)
0'00119C  1B7C 00BB 0007   250.          MOVE.B #$BB,7(A5)
0'0011A2  1B7C 0080 0009   251.          MOVE.B #$80,9(A5)
0'0011A8  3F00             252.          MOVE.W D0,-(SP)
0'0011AA  3006             253.          MOVE.W D6,D0        ;BYTE COUNT
0'0011AC  41FA FC52        254.          LEA XDATA2(PC),A0       ;DEST ADDRESS
0'0011B0  1CBC 0010        255.          MOVE.B #ENABLEA,(A6)
0'0011B4  4E45             256.          TRAP #INDATA
0'0011B6  4216             257.          CLR.B (A6)
0'0011B8  301F             258.          MOVE.W (SP)+,D0
0'0011BA  0880 0002        259.          BCLR.L #2,D0     ;CLEAR THE BIT
0'0011BE  0C00 0000        260.          CMPI.B #0,D0
0'0011C2  6700 0046        261.          BEQ REORDER
0'0011C6  0800 0003        262.  TSTB3    BTST.L #3,D0
0'0011CA  6700 FF28        263.          BEQ TSTB0
0'0011CE  123C 0003        264.          MOVE.B #3,D1
0'0011D2  6100 0142        265.          BSR TSTSLV
0'0011D6  0801 0007        266.          BTST.L #7,D1     ;SERVICED?
0'0011DA  6700 FF18        267.          BEQ TSTB0        ;IF NOT GOTO SLV1
                           268.  *
                           269.  * SLAVE 3 SENSED AND ACKNOWLEDGED, GET DATA
                           270.  *
0'0011DE  1B7C 00B8 0008   271.          MOVE.B #$B8,8(A5)
0'0011E4  1B7C 00BB 0009   272.          MOVE.B #$BB,9(A5)
0'0011EA  3F00             273.          MOVE.W D0,-(SP)
0'0011EC  3006             274.          MOVE.W D6,D0        ;BYTE COUNT
0'0011EE  41FA FD10        275.          LEA XDATA3(PC),A0       ;DEST ADDRESS
0'0011F2  1CBC 0010        276.          MOVE.B #ENABLEA,(A6)
0'0011F6  4E45             277.          TRAP #INDATA
0'0011F8  4216             278.          CLR.B (A6)
0'0011FA  301F             279.          MOVE.W (SP)+,D0
0'0011FC  0880 0003        280.          BCLR.L #3,D0     ;CLEAR THE BIT
0'001200  0C00 0000        281.          CMPI.B #0,D0
0'001204  67 04            282.          BEQ.S REORDER
0'001206  6000 FEEC        283.          BRA TSTB0
                           284.  *
                           285.  * HAVE ALL DATA IN, NOW DO REARRANGE FROM THE
                           286.  * XDATAi ARRAY TO THE DATAII ARRAY FOR CORRECT
                           287.  * ORDER.
                           288.  *
0'00120A  41FA F9F4        289.  REORDER LEA XDATA0(PC),A0
0'00120E  43FA F1F0        290.          LEA DATAII(PC),A1
0'001212  7203             291.          MOVEQ.L #NPE-1,D1
0'001214  7A3F             292.  RLOOPT  MOVEQ.L #NPTE-1,D5
0'001216  2298             293.  RLOOPN  MOVE.L (A0)+,(A1)
0'001218  D3CA             294.          ADDA.L A2,A1
0'00121A  51CD FFFA        295.          DBRA D5,RLOOPN
0'00121E  93CB             296.          SUBA.L A3,A1
0'001220  51C9 FFF2        297.          DBRA D1,RLOOPT
                           298.  *
                           299.  * forward fft and filtering is done
                           300.  * now redistribute the frequency domain data
                           301.  * to the slaves for the inverse transform.
                           302.  * simply do again!
                           303.  *
0'001224  5243             304.          ADDQ.W #1,D3
0'001226  0C43 0002        305.          CMPI.W #2,D3
0'00122A  67 2C            306.          BEQ.S SKIP
0'00122C  41FA FDDC        307.          LEA BRDCAST(PC),A0
0'001230  4E48             308.          TRAP #NETCNF
0'001232  103C 000F        309.          MOVE.B #ALLSLV,D0
0'001236  4E49             310.          TRAP #SRQSRVC
0'001238  41FA F1C6        311.          LEA DATAII(PC),A0       ;START OF DATA
0'00123C  43FA F5C2        312.          LEA FILTER(PC),A1
0'001240  93C8             313.          SUBA.L A0,A1
```

```
0'001242  3009            314.          MOVE.W A1,D0
0'001244  13FC 0020       315.          MOVE.B #ENABLEB,PGCR
          00012FC0
0'00124C  4E46            316.          TRAP #OUTDATA
0'00124E  4239 00012FC0   317.          CLR.B PGCR
0'001254  6000 FE9C       318.          BRA REDIST
                          319.  *
                          320.  * PROGRAM IS DONE, CAN STOP TIMER, AND
                          321.  * OUTPUT NUMBER OF 32 CLOCK INTERVALS
                          322.  * PASSED. FIRST DISABLE TIMER
                          323.  *
0'001258  4214            324.  SKIP    CLR.B (A4)
0'00125A  222C 0006       325.          MOVE.L 6(A4),D1 ;GET NEW COUNT
0'00125E  203C 00FFFFFF   326.          MOVE.L #CNTSTRT,D0
0'001264  9081            327.          SUB.L D1,D0     ;GET ELAPSED COUNT
                          328.  *
                          329.  * OUTPUT CR, LF, THEN THE COUNT IN HEX
                          330.  *
0'001266  4E4F            331.          TRAP #15
0'001268  000A            332.          DC.W SENCLFC
0'00126A  4E4F            333.          TRAP #15
0'00126C  0018            334.          DC.W DISBUF8
0'00126E  2F00            335.          MOVE.L D0,-(SP)
0'001270  41FA 0153       336.          LEA SPC(PC),A0
0'001274  4E4F            337.          TRAP #15
0'001276  0011            338.          DC.W OUTMESC
0'001278  2007            339.          MOVE.L D7,D0
0'00127A  4E4F            340.          TRAP #15
0'00127C  0018            341.          DC.W DISBUF8
0'00127E  201F            342.          MOVE.L (SP)+,D0
                          343.  *
                          344.  * NOW KEEP TRACK OF STATISTICS
                          345.  *
0'001280  D197            346.          ADD.L D0,(SP)    ;ADD TO SUM
0'001282  B0AF 0004       347.          CMP.L 4(SP),D0   ;CHECK MAX
0'001286  6F 04           348.          BLE.S NOTMAX     ;D0 NOT MAX
0'001288  2F40 0004       349.          MOVE.L D0,4(SP)  ;NEW MAX
0'00128C  B0AF 0008       350.  NOTMAX  CMP.L 8(SP),D0   ;CHECK MIN
0'001290  6C 04           351.          BGE.S NOTMIN     ;D0 NOT MIN
0'001292  2F40 0008       352.          MOVE.L D0,8(SP)  ;NEW MIN
0'001296  51CF FE02       353.  NOTMIN  DBRA D7,REEXE    ;DO D7 TIMES
                          354.
                          355.  *
                          356.  * WHEN HERE, PROGRAM HAS BEEN EXECUTED 256 TIMES
                          357.  * SEND AN ILLEGAL ADDRESS FOR DATA, FORCING
                          358.  * SLAVES TO PROGRAM ACCEPT MODE
                          359.  *
0'00129A  41FA FD6E       360.          LEA BRDCAST(PC),A0
0'00129E  4E48            361.          TRAP #NETCNF
0'0012A0  103C 000F       362.          MOVE.B #ALLSLV,D0
0'0012A4  4E49            363.          TRAP #SRQSRVC
0'0012A6  207C FFFF0000   364.          MOVEA.L #$FFFF0000,A0   ;ILLEGAL ADDR
0'0012AC  13FC 0020       365.          MOVE.B #ENABLEB,PGCR
          00012FC0
0'0012B4  4E47            366.          TRAP #ADBYTOT   ;THAT'LL FIX 'EM
0'0012B6  4239 00012FC0   367.          CLR.B PGCR
                          368.  *
                          369.  * OUTPUT STATISTICS
                          370.  *
0'0012BC  4E4F            371.          TRAP #15
0'0012BE  000A            372.          DC.W SENCLFC
0'0012C0  4E4F            373.          TRAP #15
0'0012C2  000A            374.          DC.W SENCLFC
0'0012C4  41FA 008A       375.          LEA AVEMES(PC),A0
0'0012C8  4E4F            376.          TRAP #15
0'0012CA  0011            377.          DC.W OUTMESC
0'0012CC  201F            378.          MOVE.L (SP)+,D0 ;GET SUM
0'0012CE  E080            379.          ASR.L #DIV256,D0        ;DO AVERAGE
0'0012D0  4E4F            380.          TRAP #15
0'0012D2  0018            381.          DC.W DISBUF8    ;OUTPUT AVERAGE
0'0012D4  4E4F            382.          TRAP #15
0'0012D6  000A            383.          DC.W SENCLFC
0'0012D8  41FA 009D       384.          LEA MAXMES(PC),A0
0'0012DC  4E4F            385.          TRAP #15
0'0012DE  0011            386.          DC.W OUTMESC
0'0012E0  201F            387.          MOVE.L (SP)+,D0 ;GET MAX
0'0012E2  4E4F            388.          TRAP #15
0'0012E4  0018            389.          DC.W DISBUF8
0'0012E6  4E4F            390.          TRAP #15
0'0012E8  000A            391.          DC.W SENCLFC
0'0012EA  41FA 00B2       392.          LEA MINMES(PC),A0
```

```
0'0012EE  4E4F            393.          TRAP #15
0'0012F0  0011            394.          DC.W OUTMESC
0'0012F2  201F            395.          MOVE.L (SP)+,D0 ;GET MIN
0'0012F4  4E4F            396.          TRAP #15
0'0012F6  0018            397.          DC.W DISBUF8
0'0012F8  4E4F            398.          TRAP #15
0'0012FA  000A            399.          DC.W SENCLFC
                          400.  *
                          401.  * AFTER 256 RUNS, LOAD THE SECOND SLAVE
                          402.  * PROGRAM TO ACQUIRE THE TIMING DATA
                          403.  * IN THE SLAVES TIMER COUNT REGISTERS
                          404.  *
0'0012FC  4E75            405.          RTS
                          406.  * NOW FINALLY DONE!
                          407.  *
                          408.  *
                          409.  * SUBROUTINE SENDOUT
                          410.  *
0'0012FE  93C8            411.  SENDOUT SUBA.L A0,A1
0'001300  3009            412.          MOVE.W A1,D0
0'001302  13FC 0020       413.          MOVE.B #ENABLEB,PGCR
          00012FC0
0'00130A  4E47            414.          TRAP #ADBYTOT
0'00130C  4E46            415.          TRAP #OUTDATA
0'00130E  4239 00012FC0   416.          CLR.B PGCR
0'001314  4E75            417.          RTS
                          418.  *
                          419.  * SUBROUTINE TSTSLV
                          420.  * SLAVE NUMBER TO TEST IS IN D1.B
                          421.  * IF ASSERTING AN SRQ, THEN ACKNOWLEDGED AND
                          422.  * HIGH BIT IN D1.B IS SET, ELSE CLEAR
                          423.  *
0'001316  48E7 8080       424.  TSTSLV  MOVEM.L D0/A0,-(SP)
0'00131A  207C 00012FCC   425.          MOVEA.L #PCDR,A0
0'001320  0810 0000       426.          BTST.B #0,(A0)
0'001324  66 22           427.          BNE.S NOINT
0'001326  E509            428.          LSL.B #2,D1
0'001328  08C1 0004       429.          BSET.L #4,D1
0'00132C  1081            430.          MOVE.B D1,(A0)
0'00132E  0890 0004       431.          BCLR.B #4,(A0)
0'001332  1010            432.          MOVE.B (A0),D0
0'001334  08D0 0004       433.          BSET.B #4,(A0)
0'001338  0800 0001       434.          BTST.L #1,D0
0'00133C  66 0A           435.          BNE.S NOINT
0'00133E  123C 00FF       436.          MOVE.B #$FF,D1
0'001342  4CDF 0101       437.          MOVEM.L (SP)+,D0/A0
0'001346  4E75            438.          RTS
0'001348  4201            439.  NOINT   CLR.B D1
0'00134A  4CDF 0101       440.          MOVEM.L (SP)+,D0/A0
0'00134E  4E75            441.          RTS
                          442.  *
0'001350  41 76 65 72 61  443.  AVEMES  DC.B 'Average time ( x 4 for microseconds)= ',EOT
          67 65 20 74 69
          6D 65 20 28 20
          78 20 34 20 66
          6F 72 20 6D 69
          63 72 6F 73 65
          63 6F 6E 64 73
          29 3D 20 04
0'001377  4D 61 78 69 6D  444.  MAXMES  DC.B 'Maximum time ( x 4 for microseconds)= ',EOT
          75 6D 20 74 69
          6D 65 20 28 20
          78 20 34 20 66
          6F 72 20 6D 69
          63 72 6F 73 65
          63 6F 6E 64 73
          29 3D 20 04
0'00139E  4D 69 6E 69 6D  445.  MINMES  DC.B 'Minimum time ( x 4 for microseconds)= ',EOT
          75 6D 20 74 69
          6D 65 20 28 20
          78 20 34 20 66
          6F 72 20 6D 69
          63 72 6F 73 65
          63 6F 6E 64 73
          29 3D 20 04
0'0013C5  20 20 20 20 04  446.  SPC     DC.B '    ',EOT
0'0013CA                  447.          END
    0 Errors
```

```
*
* LINK SPEC FOR PFILNOM4
*
      LINK PFILNOM4
      LINK SLVNOMES
      ORG $1000
      SECTION 0,15
      END
```

M.7 Program SLVNOMES: Multicomputer Frequency Domain Filtering Program: Slave programs for PFILNOM?

```
                                1.          TTL SLVNOMES
                                2.  *
                                3.  * THIS CODE IS USED AS THE COMMON SLAVE
                                4.  * CODE FOR ALL PARALLEL DIGITAL FILTERING
                                5.  * TEST PROGRAMS.  LINK WITH THE MASTER
                                6.  * CODE.
                                7.  * HAS NO CODE FOR SELF MEASURMENT
                                8.  * BUT LINK WITH PFILNOM? TO GET OVERALL
                                9.  * TIMING
                               10.  *
                               11.  * XDEF'S,XREF'S,EQU'S
                               12.  *
F'000000                       13.          SECTION 15
                               14.          XREF DATAII,FILTER
                               15.          XREF PGCR,ENABLEA,ENABLEB
                               16.          XREF NPT,NPTE,NPE
                               17.          XREF SHIFT14,TCR,CNTSTRT,ENABLTM
                               18.          XDEF S0PRO,S1PRO,S2PRO,S3PRO,DONE
                               19.  *
        # 00000003             20.  SRQASRT EQU 3
        # 00000005             21.  INDATA  EQU 5
        # 00000006             22.  OUTDATA EQU 6
        # 0000000E             23.  ABORT   EQU 14
        # 00000006             24.  NUGAM   EQU 6
                               25.  *
                               26.  ********************
                               27.  * SLAVE 0 PROGRAM *
                               28.  ********************
                               29.  *
                               30.  * INPUT DATA AT DATAII
                               31.  * FILTER DATA AT FILTER 64 POINTS
                               32.  * NOTE THAT SLAVE 0 HAS NO TWIDDLE FACTOR
                               33.  * MULTIPLIES
                               34.  *
                               35.  *
F'000000  2A7C'00000000        36.  S0PRO   MOVEA.L #NPE,A5          ;ADDR INC
F'000006  2E3C'00000000        37.          MOVE.L #SHIFT14,D7       ;/16384 SHFT
F'00000C  303C'FFFF            38.          MOVE.W #NPTE-1,D0        ;MAJ LOOP CNTR
F'000010  2C3C'00000000        39.          MOVE.L #NPT,D6           ;INCREMENT
F'000016  287C'FFFFFFFC        40.          MOVEA.L #NPT-4,A4        ;INDX TO ARRAYS
F'00001C  41FA ****            41.          LEA DATAII(PC),A0        ;XDATA POINTER
F'000020  4281                 42.  TWDL0   CLR.L D1                 ;INIT SUMS
F'000022  2401                 43.          MOVE.L D1,D2
F'000024  2648                 44.          MOVEA.L A0,A3
F'000026  7A'FF                45.          MOVEQ.L #NPE-1,D5
F'000028  D7CC                 46.          ADDA.L A4,A3
F'00002A  3613                 47.  ADDL0   MOVE.W (A3),D3           ;GET REAL PART
F'00002C  48C3                 48.          EXT.L D3
F'00002E  D283                 49.          ADD.L D3,D1              ;REAL SUM D1
F'000030  362B 0002            50.          MOVE.W 2(A3),D3          ;GET IMAG PART
F'000034  48C3                 51.          EXT.L D3
F'000036  D483                 52.          ADD.L D3,D2              ;IMAG SUM D2
F'000038  D7C6                 53.          ADDA.L D6,A3             ;NEXT TERM
F'00003A  51CD FFEE            54.          DBRA D5,ADDL0
F'00003E  E481                 55.          ASR.L #2,D1              ;ADJUST SUMS
F'000040  E482                 56.          ASR.L #2,D2              ;ADJUST SUMS
                               57.  *
                               58.  * NO TWIDDLE FACTORS FOR SLAVE 0
                               59.  * STORE SUMMED AND TWIDDLED DATA AT DATAII IN
                               60.  * PLACE
                               61.  *
F'000042  3181 C800            62.          MOVE.W D1,0(A0,A4.L)
F'000046  3182 C802            63.          MOVE.W D2,2(A0,A4.L)
F'00004A  99CD                 64.          SUBA.L A5,A4
F'00004C  51C8 FFD2            65.          DBRA D0,TWDL0
                               66.  *
                               67.  * INSERT FAVORITE 64-POINT FFT HERE
                               68.  * INPUT DATA AT DATAII (0-255)
                               69.  *
F'000050  78'F7                70.  USHFL0  MOVEQ.L #NPTE-9,D4        ;COUNTER=K
F'000052  3C04                 71.  SWAPL0  MOVE.W D4,D6          ;PREPARE BIT REV
F'000054  3E06                 72.  BREV20  MOVE.W D6,D7        ;SAVE DATA
F'000056  4246                 73.          CLR.W D6            ;
F'000058  7A05                 74.          MOVEQ.L #5,D5       ;BIT COUNTER
F'00005A  E257                 75.  REV20   ROXR.W #1,D7        ;SFT RIGHT,XTEND
F'00005C  E356                 76.          ROXL.W #1,D6
```

```
F'00005E  51CD FFFA          77.          DBRA D5,REV20
F'000062  B846               78.          CMP.W D6,D4          ;I=REV(K)<=K?
F'000064  6C 14              79.          BGE.S DECK0           ;NO SWAP IF SO
F'000066  3204               80.          MOVE.W D4,D1         ;COPY OF K
F'000068  E541               81.          ASL.W #2,D1          ;K*4
F'00006A  E546               82.          ASL.W #2,D6          ;I*4
F'00006C  2430 1000          83.          MOVE.L 0(A0,D1.W),D2 ;DO SWAP
F'000070  21B0 6000 1000     84.          MOVE.L 0(A0,D6.W),0(A0,D1.W)
F'000076  2182 6000          85.          MOVE.L D2,0(A0,D6.W) ;DONE SWAP
F'00007A  51CC FFD6          86.  DECK0   DBRA D4,SWAPL0
F'00007E  7201               87.  FFT0    MOVEQ.L #1,D1    ;D1=N2=1
F'000080  43FA 0218          88.          LEA SINBLK0(PC),A1   ;SIN VALS PNTR A1
F'000084  45FA 0294          89.          LEA COSBLK0(PC),A2   ;COS VALS PNTR A2
                             90.  *
                             91.  * L IS DOWN COUNTED #NUGAM-1 TO 0
                             92.  *
F'000088  3F3C 0001          93.          MOVE.W #1,-(SP)  ;(SP) IS L
F'00008C  7405               94.          MOVEQ.L #NUGAM-1,D2         ;D2=NUGAM-1=NU1
F'00008E  4283               95.          CLR.L D3                ;D3=K=0
F'000090  70'00              96.          MOVEQ.L #SHIFT14,D0
                             97.  *
                             98.  * INITIALIZE I=N2-1, TO 0 IN STEPS -1
                             99.  *
F'000092  3801              100.  INITI0  MOVE.W D1,D4     ;D4=I=N2
F'000094  5344              101.          SUBQ.W #1,D4     ;SET FOR DBRA
                            102.  *
                            103.  * DETERMINE P=MOD(K*2**(NUGAM-L),64)
                            104.  *
                            105.  *
F'000096  3C03              106.  LOOPIN0 MOVE.W D3,D6     ;D6=K
F'000098  3A17              107.          MOVE.W (SP),D5   ;GET L
F'00009A  5D45              108.          SUBQ.W #NUGAM,D5 ;L-NUGAM
F'00009C  4445              109.          NEG.W D5         ;NUGAM-L
F'00009E  EB66              110.          ASL.W D5,D6      ;K*2**(NUGAM-L)
F'0000A0  0246 003F         111.          ANDI.W #$003F,D6 ;MOD 64
                            112.  *
                            113.  * D6 HAS INDEX REQUIRED
F'0000A4  DC46              114.          ADD.W D6,D6      ;D6*2 FOR WORD BOUND
F'0000A6  3A71 6000         115.          MOVEA.W 0(A1,D6.W),A5 ;A5 PNTR IM(W**P)
F'0000AA  2E03              116.          MOVE.L D3,D7     ;D7=K
F'0000AC  2A03              117.          MOVE.L D3,D5     ;D5 TOO
F'0000AE  E545              118.          ASL.W #2,D5      ;K*4 FOR LWORD BOUND
F'0000B0  4DF0 5000         119.          LEA 0(A0,D5.W),A6    ;A6 IS INDEX X(K)
F'0000B4  DE41              120.          ADD.W D1,D7      ;D7=K+N2
F'0000B6  E547              121.          ASL.W #2,D7      ;D7=4(K+N2) LWORD BND
F'0000B8  47F0 7000         122.          LEA 0(A0,D7.W),A3 ;A3 PNT RE(X(K+N2))
F'0000BC  3A32 6000         123.          MOVE.W 0(A2,D6.W),D5     ;GET RE(W**P)
F'0000C0  3E05              124.          MOVE.W D5,D7     ;PUT ALSO IN D7
F'0000C2  CBD3              125.          MULS (A3),D5     ;RE(W**P)*RE(X(K+N2))
F'0000C4  3C0D              126.          MOVE.W A5,D6  ;GET IM(W**P)*
F'0000C6  CDEB 0002         127.          MULS 2(A3),D6    ;IM(W**P)*IM(X(K+N2))
F'0000CA  9A86              128.          SUB.L D6,D5      ;D5=RE(T1)*
F'0000CC  E0A5              129.          ASR.L D0,D5      ;/16384 TO SCALE
F'0000CE  3C0D              130.          MOVE.W A5,D6  ;GET IM(W**P)
F'0000D0  CDD3              131.          MULS (A3),D6     ;IM(W**P)*RE(X(K+N2))
F'0000D2  CFEB 0002         132.          MULS 2(A3),D7    ;RE(W**P)*IM(X(K+N2))
F'0000D6  DC87              133.          ADD.L D7,D6      ;D6=IM(T1)
F'0000D8  E0A6              134.          ASR.L D0,D6      ;/16384 TO SCALE
F'0000DA  3E16              135.          MOVE.W (A6),D7   ;RE(X(K))
F'0000DC  9E45              136.          SUB.W D5,D7      ;RE(X(K))-RE(T1)
F'0000DE  E247              137.          ASR.W #1,D7      ;/2
F'0000E0  3687              138.          MOVE.W D7,(A3)   ;STORE ANSWER
F'0000E2  3E2E 0002         139.          MOVE.W 2(A6),D7  ;IM(X(K))
F'0000E6  9E46              140.          SUB.W D6,D7      ;IM(X(K))-IM(T1)
F'0000E8  E247              141.          ASR.W #1,D7      ;/2
F'0000EA  3747 0002         142.          MOVE.W D7,2(A3)  ;STORE ANSWER
F'0000EE  DB56              143.          ADD.W D5,(A6)    ;RE(X(K))+RE(T1)
F'0000F0  E0D6              144.          ASR.W (A6)       ;/2
F'0000F2  DD6E 0002         145.          ADD.W D6,2(A6)   ;IM(X(K))+IM(T1)
F'0000F6  E0EE 0002         146.          ASR.W 2(A6)      ;/2
F'0000FA  5243              147.          ADDQ.W #1,D3     ;INCREMENT K
F'0000FC  51CC FF98         148.          DBRA D4,LOOPIN0   ;DO TILL I=-1
F'000100  D641              149.          ADD.W D1,D3      ;K=K+N2
F'000102  0C43'FFFF         150.          CMPI.W #NPTE-1,D3 ;K<N-1 ?
F'000106  6D 8A             151.          BLT INITI0       ;GO AGAIN IF SO
F'000108  4243              152.          CLR.W D3         ;K=0
F'00010A  5342              153.          SUBQ.W #1,D2     ;NU1=NU1-1
F'00010C  E341              154.          ASL.W #1,D1      ;N2=N2*2
F'00010E  5257              155.          ADDQ.W #1,(SP)
F'000110  0C57 0007         156.          CMPI.W #NUGAM+1,(SP)
F'000114  6600 FF7C         157.          BNE INITI0
```

```
F'000118  301F                158.          MOVE.W (SP)+,D0 ;CLEAN STACK
                              159.    *
                              160.    * NOW DO FILTERING, FILTER DATA IS AT "FILTER"
                              161.    * 64 POINTS
                              162.    *
                              163.    * DO COMPLEX MULTIPLY BY THE FILTER.
                              164.    * FILTER CAN BE +16384 TO -16384 WITH 16384
                              165.    * BEING THE 0 db GAIN VALUE.
                              166.    * (A+iB)*(C+iD)
                              167.    *
F'00011A  43FA ****           168.          LEA FILTER(PC),A1        ;A1 PNT FILTER
F'00011E  303C'FFFF           169.          MOVE.W #NPTE-1,D0        ;LOOP COUNTER
F'000122  7A'00               170.          MOVEQ.L #SHIFT14,D5              ;SHIFT COUNT
F'000124  3219                171.    FILOOP0 MOVE.W (A1)+,D1        ;GET C
F'000126  3601                172.          MOVE.W D1,D3
F'000128  C3D8                173.          MULS (A0)+,D1            ;DO AC
F'00012A  3419                174.          MOVE.W (A1)+,D2          ;GET D
F'00012C  3802                175.          MOVE.W D2,D4
F'00012E  C5D0                176.          MULS (A0),D2             ;DO BD
F'000130  9282                177.          SUB.L D2,D1              ;AC-BD
F'000132  EAA1                178.          ASR.L D5,D1              ;ADJUST /16384
F'000134  C7D0                179.          MULS (A0),D3             ;DO BC
F'000136  C9E0                180.          MULS -(A0),D4            ;DO AD
F'000138  D883                181.          ADD.L D3,D4              ;BC+AD
F'00013A  EAA4                182.          ASR.L D5,D4              ;ADJUST /16384
F'00013C  30C1                183.          MOVE.W D1,(A0)+ ;STORE REAL
F'00013E  30C4                184.          MOVE.W D4,(A0)+ ;STORE IMAG
F'000140  51C8 FFE2           185.          DBRA D0,FILOOP0
                              186.    *
                              187.    * ASSERT SRQ AND WAIT FOR ACKNOWLEDGE
                              188.    *
F'000144  4E43                189.          TRAP #SRQASRT
                              190.    * SEND OUT 64 COMPLEX POINTS CALCULATED
>>> WARNING line[191] Expr. type   [191] SLVNOMES.A68
F'000146  303C ****           191.          MOVE.W #NPTE*4,D0        ;BYTE COUNT
F'00014A  41FA ****           192.          LEA DATAII(PC),A0
F'00014E  13FC 00'00          193.          MOVE.B #ENABLEB,PGCR
        '00000000
F'000156  4E46                194.          TRAP #OUTDATA
F'000158  4239'00000000       195.          CLR.B PGCR
                              196.    *
                              197.    * ASSERT AN SRQ TO SIGNAL READY FOR NEW DATA
                              198.    * AND RECEIVE NEW FILTERED DATA TO BE INVERSE
                              199.    * TRANSFORMED
                              200.    *
F'00015E  4E43                201.          TRAP #SRQASRT
>>> WARNING line[202] Expr. type   [202] SLVNOMES.A68
F'000160  303C ****           202.          MOVE.W #NPT*4,D0         ;BYTE COUNT
F'000164  41FA ****           203.          LEA DATAII(PC),A0
F'000168  13FC 00'00          204.          MOVE.B #ENABLEA,PGCR
        '00000000
F'000170  4E45                205.          TRAP #INDATA
F'000172  4239'00000000       206.          CLR.B PGCR
                              207.    *
                              208.    * NOW DO THE INVERSE TRANSFORM
                              209.    * USE SAME SIN AND COS TABLES, NEGATE THE SIN
                              210.    * ENTRY, AND USE NO ASR #2 FOR DIVISION BY NPTE
                              211.    * NOTE SIGN CHANGE ON 4-ADDITIONS TOO
                              212.    *
F'000178  2A7C'00000000       213.          MOVEA.L #NPE,A5          ;ADDR INC
F'00017E  2E3C'00000000       214.          MOVE.L #SHIFT14,D7       ;/16384 SHFT
F'000184  303C'FFFF           215.          MOVE.W #NPTE-1,D0        ;MAJ LOOP CNTR
F'000188  2C3C'00000000       216.          MOVE.L #NPT,D6           ;INCREMENT
F'00018E  287C'FFFFFFFC       217.          MOVEA.L #NPT-4,A4        ;INDX TO ARRAYS
F'000194  41FA ****           218.          LEA DATAII(PC),A0        ;XDATA POINTER
F'000198  4281                219.    TWDL0R  CLR.L D1               ;INIT SUMS
F'00019A  2401                220.          MOVE.L D1,D2
F'00019C  2648                221.          MOVEA.L A0,A3
F'00019E  7A'FF               222.          MOVEQ.L #NPE-1,D5
F'0001A0  D7CC                223.          ADDA.L A4,A3
F'0001A2  3613                224.    ADDL0R    OVE.W (A3),D3          ;GET REAL PART
F'0001A4  D243                225.              DD.W D3,D1             ;REAL SUM D1
F'0001A6  362B 0002           226.          MOVE.W 2(A3),D3          ;GET IMAG PART
F'0001AA  D443                227.          ADD.W D3,D2              ;IMAG SUM D2
F'0001AC  D7C6                228.          ADDA.L D6,A3             ;NEXT TERM
F'0001AE  51CD FFF2           229.          DBRA D5,ADDL0R
                              230.    *
                              231.    * NO TWIDDLE FACTORS FOR SLAVE 0
                              232.    * STORE SUMMED AND TWIDDLED DATA AT DATAII IN
                              233.    * PLACE
                              234.    *
```

```
F'0001B2  3181 C800          235.          MOVE.W D1,0(A0,A4.L)
F'0001B6  3182 C802          236.          MOVE.W D2,2(A0,A4.L)
F'0001BA  99CD               237.          SUBA.L A5,A4
F'0001BC  51C8 FFDA          238.          DBRA D0,TWDL0R
                             239.  *
                             240.  * INSERT FAVORITE 64-POINT FFT HERE
                             241.  * INPUT DATA AT DATAII (0-255)
                             242.  *
F'0001C0  78'F7              243.  USHFL0R MOVEQ.L #NPTE-9,D4        ;COUNTER=K
F'0001C2  3C04               244.  SWAPL0R MOVE.W D4,D6          ;PREPARE BIT REV
F'0001C4  3E06               245.  BREV20R MOVE.W D6,D7        ;SAVE DATA
F'0001C6  4246               246.          CLR.W D6            ;
F'0001C8  7A05               247.          MOVEQ.L #5,D5       ;BIT COUNTER
F'0001CA  E257               248.  REV20R  ROXR.W #1,D7        ;SFT RIGHT,XTEND
F'0001CC  E356               249.          ROXL.W #1,D6
F'0001CE  51CD FFFA          250.          DBRA D5,REV20R
F'0001D2  B846               251.          CMP.W D6,D4           ;I=REV(K)<=K?
F'0001D4  6C 14              252.          BGE.S DECK0R            ;NO SWAP IF SO
F'0001D6  3204               253.          MOVE.W D4,D1          ;COPY OF K
F'0001D8  E541               254.          ASL.W #2,D1           ;K*4
F'0001DA  E546               255.          ASL.W #2,D6           ;I*4
F'0001DC  2430 1000          256.          MOVE.L 0(A0,D1.W),D2 ;DO SWAP
F'0001E0  21B0 6000 1000     257.          MOVE.L 0(A0,D6.W),0(A0,D1.W)
F'0001E6  2182 6000          258.          MOVE.L D2,0(A0,D6.W) ;DONE SWAP
F'0001EA  51CC FFD6          259.  DECK0R   DBRA D4,SWAPL0R
F'0001EE  7201               260.  FFT0R   MOVEQ.L #1,D1    ;D1=N2=1
F'0001F0  43FA 00A8          261.          LEA SINBLK0(PC),A1  ;SIN VALS PNTR A1
F'0001F4  45FA 0124          262.          LEA COSBLK0(PC),A2  ;COS VALS PNTR A2
                             263.  *
                             264.  * L IS DOWN COUNTED #NUGAM-1 TO 0
                             265.  *
F'0001F8  3F3C 0001          266.          MOVE.W #1,-(SP)  ;(SP) IS L
F'0001FC  7405               267.          MOVEQ.L #NUGAM-1,D2         ;D2=NUGAM-1=NU1
F'0001FE  4283               268.          CLR.L D3                ;D3=K=0
F'000200  70'00              269.          MOVEQ.L #SHIFT14,D0
                             270.  *
                             271.  * INITIALIZE I=N2-1, TO 0 IN STEPS -1
                             272.  *
F'000202  3801               273.  INITI0R MOVE.W D1,D4     ;D4=I=N2
F'000204  5344               274.          SUBQ.W #1,D4     ;SET FOR DBRA
                             275.  *
                             276.  * DETERMINE P=MOD(K*2**(NUGAM-L),64)
                             277.  *
                             278.  *
F'000206  3C03               279.  LPIN0R  MOVE.W D3,D6     ;D6=K
F'000208  3A17               280.          MOVE.W (SP),D5  ;GET L
F'00020A  5D45               281.          SUBQ.W #NUGAM,D5 ;L-NUGAM
F'00020C  4445               282.          NEG.W D5         ;NUGAM-L
F'00020E  EB66               283.          ASL.W D5,D6      ;K*2**(NUGAM-L)
F'000210  0246 003F          284.          ANDI.W #$003F,D6 ;MOD 64
                             285.  *
                             286.  * D6 HAS INDEX REQUIRED
F'000214  DC46               287.          ADD.W D6,D6      ;D6*2 FOR WORD BOUND
F'000216  3A71 6000          288.          MOVEA.W 0(A1,D6.W),A5 ;A5 PNTR IM(W**P)
F'00021A  2E03               289.          MOVE.L D3,D7     ;D7=K
F'00021C  2A03               290.          MOVE.L D3,D5     ;D5 TOO
F'00021E  E545               291.          ASL.W #2,D5      ;K*4 FOR LWORD BOUND
F'000220  4DF0 5000          292.          LEA 0(A0,D5.W),A6    ;A6 IS INDEX X(K)
F'000224  DE41               293.          ADD.W D1,D7      ,D7=K+N2
F'000226  E547               294.          ASL.W #2,D7      ;D7=4(K+N2) LWORD BND
F'000228  47F0 7000          295.          LEA 0(A0,D7.W),A3 ;A3 PNT RE(X(K+N2))
F'00022C  3A32 6000          296.          MOVE.W 0(A2,D6.W),D5     ;GET RE(W**P)
F'000230  3E05               297.          MOVE.W D5,D7     ;PUT ALSO IN D7
F'000232  CBD3               298.          MULS (A3),D5     ;RE(W**P)*RE(X(K+N2))
F'000234  3C0D               299.          MOVE.W A5,D6  ;GET IM(W**P)*
F'000236  4446               300.          NEG.W D6
F'000238  CDEB 0002          301.          MULS 2(A3),D6    ;IM(W**P)*IM(X(K+N2))
F'00023C  9A86               302.          SUB.L D6,D5      ;D5=RE(T1)*
F'00023E  E0A5               303.          ASR.L D0,D5      ;/16384 TO SCALE
F'000240  3C0D               304.          MOVE.W A5,D6  ;GET IM(W**P)
F'000242  4446               305.          NEG.W D6
F'000244  CDD3               306.          MULS (A3),D6     ;IM(W**P)*RE(X(K+N2))
F'000246  CFEB 0002          307.          MULS 2(A3),D7    ;RE(W**P)*IM(X(K+N2))
F'00024A  DC87               308.          ADD.L D7,D6      ;D6=IM(T1)
F'00024C  E0A6               309.          ASR.L D0,D6      ;/16384 TO SCALE
F'00024E  3E16               310.          MOVE.W (A6),D7   ;RE(X(K))
F'000250  9E45               311.          SUB.W D5,D7      ;RE(X(K))-RE(T1)
F'000252  3687               312.          MOVE.W D7,(A3)   ;STORE ANSWER
F'000254  3E2E 0002          313.          MOVE.W 2(A6),D7 ;IM(X(K))
F'000258  9E46               314.          SUB.W D6,D7      ;IM(X(K))-IM(T1)
F'00025A  3747 0002          315.          MOVE.W D7,2(A3) ;STORE ANSWER
```

```
F'00025E  DB56             316.          ADD.W D5,(A6)    ;RE(X(K))+RE(T1)
F'000260  DD6E 0002        317.          ADD.W D6,2(A6)   ;IM(X(K))+IM(T1)
F'000264  5243             318.          ADDQ.W #1,D3     ;INCREMENT K
F'000266  51CC FF9E        319.          DBRA D4,LPINOR   ;DO TILL I=-1
F'00026A  D641             320.          ADD.W D1,D3      ;K=K+N2
F'00026C  0C43'FFFF        321.          CMPI.W #NPTE-1,D3 ;K<N-1 ?
F'000270  6D 90            322.          BLT INITIOR      ;DO AGAIN IF SO
F'000272  4243             323.          CLR.W D3         ;K=0
F'000274  5342             324.          SUBQ.W #1,D2     ;NU1=NU1-1
F'000276  E341             325.          ASL.W #1,D1      ;N2=N2*2
F'000278  5257             326.          ADDQ.W #1,(SP)
F'00027A  0C57 0007        327.          CMPI.W #NUGAM+1,(SP)
F'00027E  66 82            328.          BNE INITIOR
F'000280  301F             329.          MOVE.W (SP)+,D0 ;CLEAN STACK
                           330.  *
                           331.  * ASSERT AN SRQ AND UPLOAD THE DATA
                           332.  *
F'000282  4E43             333.          TRAP #SRQASRT
>>> WARNING line[334] Expr. type   [334] SLVNOMES.A68
F'000284  303C ***'        334.          MOVE.W #NPTE*4,D0       ;BYTE COUNT
F'000288  13FC 00'00       335.          MOVE.B #ENABLEB,PGCR
          '00000000
F'000290  4E46             336.          TRAP #OUTDATA
F'000292  4239'00000000    337.          CLR.B PGCR
                           338.  *
                           339.  * NOW DO RTS AND RETURN TO READY FOR DATA
                           340.  * STATE TO DO OVER!
                           341.  *
F'000298  4E75             342.          RTS
                           343.  *
                           344.  * DONE!
                           345.  *
                           346.  * SIN AND COS TABLES HERE
                           347.  *
                           348.  *
                           349.  * SIN (-2*PI*I/64)*16384 FOR I=0 TO 63
                           350.  *
F'00029A  0000 F9BA F384   351.  SINBLK0 DC.W       0, -1606, -3196, -4756
          ED6C
F'0002A2  E782 E1D5 DC72   352.          DC.W   -6270, -7723, -9102,-10394
          D766
F'0002AA  D2BF CE87 CAC9   353.          DC.W  -11585,-12665,-13623,-14449
          C78F
F'0002B2  C4DF C2C1 C13B   354.          DC.W  -15137,-15679,-16069,-16305
          C04F
F'0002BA  C000 C04F C13B   355.          DC.W  -16384,-16305,-16069,-15679
          C2C1
F'0002C2  C4DF C78F CAC9   356.          DC.W  -15137,-14449,-13623,-12665
          CE87
F'0002CA  D2BF D766 DC72   357.          DC.W  -11585,-10394, -9102, -7723
          E1D5
F'0002D2  E782 ED6C F384   358.          DC.W   -6270, -4756, -3196, -1606
          F9BA
F'0002DA  0000 0646 0C7C   359.          DC.W       0,  1606,  3196,  4756
          1294
F'0002E2  187E 1E2B 238E   360.          DC.W    6270,  7723,  9102, 10394
          289A
F'0002EA  2D41 3179 3537   361.          DC.W   11585, 12665, 13623, 14449
          3871
F'0002F2  3B21 3D3F 3EC5   362.          DC.W   15137, 15679, 16069, 16305
          3FB1
F'0002FA  4000 3FB1 3EC5   363.          DC.W   16384, 16305, 16069, 15679
          3D3F
F'000302  3B21 3871 3537   364.          DC.W   15137, 14449, 13623, 12665
          3179
F'00030A  2D41 289A 238E   365.          DC.W   11585, 10394,  9102,  7723
          1E2B
F'000312  187E 1294 0C7C   366.          DC.W    6270,  4756,  3196,  1606
          0646
                           367.  *
                           368.  * COS(-2*PI*I/64)*16384 FOR I=0 TO 63
                           369.  *
F'00031A  4000 3FB1 3EC5   370.  COSBLK0 DC.W   16384, 16305, 16069, 15679
          3D3F
F'000322  3B21 3871 3537   371.          DC.W   15137, 14449, 13623, 12665
          3179
F'00032A  2D41 289A 238E   372.          DC.W   11585, 10394,  9102,  7723
          1E2B
F'000332  187E 1294 0C7C   373.          DC.W    6270,  4756,  3196,  1606
          0646
F'00033A  0000 F9BA F384   374.          DC.W       0, -1606, -3196, -4756
```

```
                ED6C
F'000342        E782 E1D5 DC72      375.          DC.W   -6270, -7723, -9102,-10394
                D766
F'00034A        D2BF CE87 CAC9      376.          DC.W  -11585,-12665,-13623,-14449
                C78F
F'000352        C4DF C2C1 C13B      377.          DC.W  -15137,-15679,-16069,-16305
                C04F
F'00035A        C000 C04F C13B      378.          DC.W  -16384,-16305,-16069,-15679
                C2C1
F'000362        C4DF C78F CAC9      379.          DC.W  -15137,-14449,-13623,-12665
                CE87
F'00036A        D2BF D766 DC72      380.          DC.W  -11585,-10394, -9102, -7723
                E1D5
F'000372        E782 ED6C F384      381.          DC.W   -6270, -4756, -3196, -1606
                F9BA
F'00037A        0000 0646 0C7C      382.          DC.W       0,  1606,  3196,  4756
                1294
F'000382        187E 1E2B 238E      383.          DC.W    6270,  7723,  9102, 10394
                289A
F'00038A        2D41 3179 3537      384.          DC.W   11585, 12665, 13623, 14449
                3871
F'000392        3B21 3D3F 3EC5      385.          DC.W   15137, 15679, 16069, 16305
                3FB1
                                    386.
                                    387.  ********************
                                    388.  * SLAVE 1 PROGRAM *
                                    389.  ********************
                                    390.  *
                                    391.  * INPUT DATA AT DATAII, TWIDDLE FACTORS AT
                                    392.  * TWD1
                                    393.  *
                                    394.  *
F'00039A  2A7C'00000000             395.  S1PRO   MOVEA.L #NPE,A5          ;ADDR INC
F'0003A0  2E3C'00000000             396.          MOVE.L #SHIFT14,D7       ;/16384 SHFT
F'0003A6  303C'FFFF                 397.          MOVE.W #NPTE-1,D0        ;MAJ LOOP CNTR
F'0003AA  287C'FFFFFFFC             398.          MOVEA.L #NPT-4,A4        ;INDX TO ARRAYS
F'0003B0  43FA 03EC                 399.          LEA TWD1(PC),A1          ;TWID FACT PNTR
F'0003B4  41FA ****                 400.          LEA DATAII(PC),A0        ;XDATA POINTER
F'0003B8  2648                      401.  TWDL1   MOVEA.L A0,A3
F'0003BA  D7CC                      402.          ADDA.L A4,A3
F'0003BC  3213                      403.  ADDL1   MOVE.W (A3),D1           ;GET REAL PART
F'0003BE  48C1                      404.          EXT.L D1                 ;REAL SUM IN D1
F'0003C0  362B 0102                 405.          MOVE.W 258(A3),D3
F'0003C4  48C3                      406.          EXT.L D3
F'0003C6  D283                      407.          ADD.L D3,D1
F'0003C8  362B 0200                 408.          MOVE.W 512(A3),D3
F'0003CC  48C3                      409.          EXT.L D3
F'0003CE  9283                      410.          SUB.L D3,D1
F'0003D0  362B 0302                 411.          MOVE.W 770(A3),D3
F'0003D4  48C3                      412.          EXT.L D3
F'0003D6  9283                      413.          SUB.L D3,D1              ;REAL SUM DONE
F'0003D8  342B 0002                 414.          MOVE.W 2(A3),D2          ;IMAG SUM IN D2
F'0003DC  48C2                      415.          EXT.L D2
F'0003DE  362B 0100                 416.          MOVE.W 256(A3),D3
F'0003E2  48C3                      417.          EXT.L D3
F'0003E4  9483                      418.          SUB.L D3,D2
F'0003E6  362B 0202                 419.          MOVE.W 514(A3),D3
F'0003EA  48C3                      420.          EXT.L D3
F'0003EC  9483                      421.          SUB.L D3,D2
F'0003EE  362B 0300                 422.          MOVE.W 768(A3),D3
F'0003F2  48C3                      423.          EXT.L D3
F'0003F4  D483                      424.          ADD.L D3,D2              ;IMAG SUM DONE
F'0003F6  E481                      425.          ASR.L #2,D1
F'0003F8  E482                      426.          ASR.L #2,D2              ;ADJUST SUMS
                                    427.  *
                                    428.  * NOW DO TWIDDLE FACTOR MULTIPLY
                                    429.  *
F'0003FA  3A01                      430.          MOVE.W D1,D5
F'0003FC  3831 C800                 431.          MOVE.W 0(A1,A4.L),D4
F'000400  3631 C802                 432.          MOVE.W 2(A1,A4.L),D3
F'000404  CBC4                      433.          MULS D4,D5
F'000406  3C02                      434.          MOVE.W D2,D6
F'000408  CDC3                      435.          MULS D3,D6
F'00040A  9A86                      436.          SUB.L D6,D5              ;REAL IN D5
F'00040C  C3C3                      437.          MULS D3,D1
F'00040E  C5C4                      438.          MULS D4,D2
F'000410  D282                      439.          ADD.L D2,D1              ;IMAG IN D1
F'000412  EEA5                      440.          ASR.L D7,D5
F'000414  EEA1                      441.          ASR.L D7,D1
                                    442.  *
                                    443.  * STORE SUMMED AND TWIDDLED DATA AT DATAII IN
```

```
                                444.  * PLACE
                                445.  *
F'000416  3185 C800             446.          MOVE.W D5,0(A0,A4.L)
F'00041A  3181 C802             447.          MOVE.W D1,2(A0,A4.L)
F'00041E  99CD                  448.          SUBA.L A5,A4
F'000420  51C8 FF96             449.          DBRA D0,TWDL1
                                450.  *
                                451.  * INSERT FAVORITE 64-POINT FFT HERE
                                452.  * INPUT DATA AT DATAII (0-255)
                                453.  *
F'000424  78'F7                 454.  USHFL1  MOVEQ.L #NPTE-9,D4        ;COUNTER=K
F'000426  3C04                  455.  SWAPL1  MOVE.W D4,D6          ;PREPARE BIT REV
F'000428  3E06                  456.  BREV21  MOVE.W D6,D7        ;SAVE DATA
F'00042A  4246                  457.          CLR.W D6            ;
F'00042C  7A05                  458.          MOVEQ.L #5,D5       ;BIT COUNTER
F'00042E  E257                  459.  REV21   ROXR.W #1,D7        ;SFT RIGHT,XTEND
F'000430  E356                  460.          ROXL.W #1,D6
F'000432  51CD FFFA             461.          DBRA D5,REV21
F'000436  B846                  462.          CMP.W D6,D4           ;I=REV(K)<=K?
F'000438  6C 14                 463.          BGE.S DECK1           ;NO SWAP IF SO
F'00043A  3204                  464.          MOVE.W D4,D1          ;COPY OF K
F'00043C  E541                  465.          ASL.W #2,D1           ;K*4
F'00043E  E546                  466.          ASL.W #2,D6           ;I*4
F'000440  2430 1000             467.          MOVE.L 0(A0,D1.W),D2 ;DO SWAP
F'000444  21B0 6000 1000        468.          MOVE.L 0(A0,D6.W),0(A0,D1.W)
F'00044A  2182 6000             469.          MOVE.L D2,0(A0,D6.W) ;DONE SWAP
F'00044E  51CC FFD6             470.  DECK1   DBRA D4,SWAPL1
F'000452  7201                  471.  FFT641  MOVEQ.L #1,D1    ;D1=N2=1
F'000454  43FA 0248             472.          LEA SINBLK1(PC),A1  ;SIN VALS PNTR A1
F'000458  45FA 02C4             473.          LEA COSBLK1(PC),A2  ;COS VALS PNTR A2
                                474.  *
                                475.  * L IS DOWN COUNTED #NUGAM-1 TO 0
                                476.  *
F'00045C  3F3C 0001             477.          MOVE.W #1,-(SP)  ;(SP) IS L
F'000460  7405                  478.          MOVEQ.L #NUGAM-1,D2         ;D2=NUGAM-1=NU1
F'000462  4283                  479.          CLR.L D3                ;D3=K=0
F'000464  70'00                 480.          MOVEQ.L #SHIFT14,D0
                                481.  *
                                482.  * INITIALIZE I=N2-1, TO 0 IN STEPS -1
                                483.  *
F'000466  3801                  484.  INITI1  MOVE.W D1,D4     ;D4=I=N2
F'000468  5344                  485.          SUBQ.W #1,D4     ;SET FOR DBRA
                                486.  *
                                487.  * DETERMINE P=MOD(K*2**(NUGAM-L),64)
                                488.  *
                                489.  *
F'00046A  3C03                  490.  LOOPIN1 MOVE.W D3,D6     ;D6=K
F'00046C  3A17                  491.          MOVE.W (SP),D5  ;GET L
F'00046E  5D45                  492.          SUBQ.W #NUGAM,D5 ;L-NUGAM
F'000470  4445                  493.          NEG.W D5          ;NUGAM-L
F'000472  EB66                  494.          ASL.W D5,D6       ;K*2**(NUGAM-L)
F'000474  0246 003F             495.          ANDI.W #$003F,D6 ;MOD 64
                                496.  *
                                497.  * D6 HAS INDEX REQUIRED
                                498.
F'000478  DC46                  499.          ADD.W D6,D6       ;D6*2 FOR WORD BOUND
F'00047A  3A71 6000             500.          MOVEA.W 0(A1,D6.W),A5 ;A5 PNTR IM(W**P)
F'00047E  2E03                  501.          MOVE.L D3,D7      ;D7=K
F'000480  2A03                  502.          MOVE.L D3,D5      ;D5 TOO
F'000482  E545                  503.          ASL.W #2,D5       ;K*4 FOR LWORD BOUND
F'000484  4DF0 5000             504.          LEA 0(A0,D5.W),A6   ;A6 IS INDEX X(K)
F'000488  DE41                  505.          ADD.W D1,D7       ;D7=K+N2
F'00048A  E547                  506.          ASL.W #2,D7       ;D7=4(K+N2) LWORD BND
F'00048C  47F0 7000             507.          LEA 0(A0,D7.W),A3 ;A3 PNT RE(X(K+N2))
F'000490  3A32 6000             508.          MOVE.W 0(A2,D6.W),D5     ;GET RE(W**P)
F'000494  3E05                  509.          MOVE.W D5,D7      ;PUT ALSO IN D7
F'000496  CBD3                  510.          MULS (A3),D5      ;RE(W**P)*RE(X(K+N2))
F'000498  3C0D                  511.          MOVE.W A5,D6  ;GET IM(W**P)
F'00049A  CDEB 0002             512.          MULS 2(A3),D6     ;IM(W**P)*IM(X(K+N2))
F'00049E  9A86                  513.          SUB.L D6,D5       ;D5=RE(T1)
F'0004A0  E0A5                  514.          ASR.L D0,D5       ;/16384 TO SCALE
F'0004A2  3C0D                  515.          MOVE.W A5,D6  ;GET IM(W**P)
F'0004A4  CDD3                  516.          MULS (A3),D6      ;IM(W**P)*RE(X(K+N2))
F'0004A6  CFEB 0002             517.          MULS 2(A3),D7     ;RE(W**P)*IM(X(K+N2))
F'0004AA  DC87                  518.          ADD.L D7,D6       ;D6=IM(T1)
F'0004AC  E0A6                  519.          ASR.L D0,D6       ;/16384 TO SCALE
F'0004AE  3E16                  520.          MOVE.W (A6),D7    ;RE(X(K))
F'0004B0  9E45                  521.          SUB.W D5,D7       ;RE(X(K))-RE(T1)
F'0004B2  E247                  522.          ASR.W #1,D7       ;/2
```

```
F'0004B4  3687            523.          MOVE.W D7,(A3)   ;STORE ANSWER
F'0004B6  3E2E 0002       524.          MOVE.W 2(A6),D7 ;IM(X(K))
F'0004BA  9E46            525.          SUB.W D6,D7      ;IM(X(K))-IM(T1)
F'0004BC  E247            526.          ASR.W #1,D7      ;/2
F'0004BE  3747 0002       527.          MOVE.W D7,2(A3) ;STORE ANSWER
F'0004C2  DB56            528.          ADD.W D5,(A6)    ;RE(X(K))+RE(T1)
F'0004C4  E0D6            529.          ASR.W (A6)       ;/2
F'0004C6  DD6E 0002       530.          ADD.W D6,2(A6)   ;IM(X(K))+IM(T1)
F'0004CA  E0EE 0002       531.          ASR.W 2(A6)      ;/2
F'0004CE  5243            532.          ADDQ.W #1,D3     ;INCREMENT K
F'0004D0  51CC FF98       533.          DBRA D4,LOOPIN1  ;DO TILL I=-1
F'0004D4  D641            534.          ADD.W D1,D3      ;K=K+N2
F'0004D6  0C43'FFFF       535.          CMPI.W #NPTE-1,D3 ;K<N-1 ?
F'0004DA  6D 8A           536.          BLT INITI1       ;DO AGAIN IF SO
F'0004DC  4243            537.          CLR.W D3         ;K=0
F'0004DE  5342            538.          SUBQ.W #1,D2     ;NU1=NU1-1
F'0004E0  E341            539.          ASL.W #1,D1      ;N2=N2*2
F'0004E2  5257            540.          ADDQ.W #1,(SP)
F'0004E4  0C57 0007       541.          CMPI.W #NUGAM+1,(SP)
F'0004E8  6600 FF7C       542.          BNE INITI1
F'0004EC  301F            543.          MOVE.W (SP)+,D0
                          544.  *
                          545.  * NOW DO FILTERING, FILTER DATA IS AT "FILTER"
                          546.  * 64 POINTS
                          547.  *
                          548.  * DO COMPLEX MULTIPLY BY THE FILTER.
                          549.  * FILTER CAN BE +16384 TO -16384 WITH 16384
                          550.  * BEING THE 0 db GAIN VALUE.
                          551.  * (A+iB)*(C+iD)
                          552.  *
F'0004EE  43FA ****       553.          LEA FILTER(PC),A1        ;A1 PNT FILTER
F'0004F2  303C'FFFF       554.          MOVE.W #NPTE-1,D0        ;LOOP COUNTER
F'0004F6  7A'00           555.          MOVEQ.L #SHIFT14,D5              ;SHIFT COUNT
F'0004F8  3219            556.  FILOOP1 MOVE.W (A1)+,D1          ;GET C
F'0004FA  3601            557.          MOVE.W D1,D3
F'0004FC  C3D8            558.          MULS (A0)+,D1            ;DO AC
F'0004FE  3419            559.          MOVE.W (A1)+,D2          ;GET D
F'000500  3802            560.          MOVE.W D2,D4
F'000502  C5D0            561.          MULS (A0),D2             ;DO BD
F'000504  9282            562.          SUB.L D2,D1              ;AC-BD
F'000506  EAA1            563.          ASR.L D5,D1              ;ADJUST /16384
F'000508  C7D0            564.          MULS (A0),D3             ;DO BC
F'00050A  C9E0            565.          MULS -(A0),D4            ;DO AD
F'00050C  D883            566.          ADD.L D3,D4              ;BC+AD
F'00050E  EAA4            567.          ASR.L D5,D4              ;ADJUST /16384
F'000510  30C1            568.          MOVE.W D1,(A0)+ ;STORE REAL
F'000512  30C4            569.          MOVE.W D4,(A0)+ ;STORE IMAG
F'000514  51C8 FFE2       570.          DBRA D0,FILOOP1
                          571.  *
                          572.  * ASSERT SRQ AND WAIT FOR ACKNOWLEDGE
                          573.  *
F'000518  4E43            574.          TRAP #SRQASRT
                          575.  * SEND OUT 64 COMPLEX POINTS CALCULATED
>>> WARNING line[576] Expr. type   [576] SLVNOMES.A68
F'00051A  303C ****       576.          MOVE.W #NPTE*4,D0        ;BYTE COUNT
F'00051E  41FA ****       577.          LEA DATAII(PC),A0
F'000522  13FC 00'00      578.          MOVE.B #ENABLEB,PGCR
          '00000000
F'00052A  4E46            579.          TRAP #OUTDATA
F'00052C  4239'00000000   580.          CLR.B PGCR
                          581.  *
                          582.  * ASSERT AN SRQ TO SIGNAL READY FOR NEW DATA
                          583.  * AND RECEIVE NEW FILTERED DATA TO BE INVERSE
                          584.  * TRANSFORMED
                          585.  *
F'000532  4E43            586.          TRAP #SRQASRT
>>> WARNING line[587] Expr. type   [587] SLVNOMES.A68
F'000534  303C ****       587.          MOVE.W #NPT*4,D0         ;BYTE COUNT
F'000538  41FA ****       588.          LEA DATAII(PC),A0
F'00053C  13FC 00'00      589.          MOVE.B #ENABLEA,PGCR
          '00000000
F'000544  4E45            590.          TRAP #INDATA
F'000546  4239'00000000   591.          CLR.B PGCR
                          592.  *
                          593.  * NOW DO THE INVERSE TRANSFORM
                          594.  * USE SAME SIN AND COS TABLES, NEGATE THE SIN
                          595.  * ENTRY, AND USE NO ASR #2 FOR DIVISION BY NPTE
                          596.  * NOTE SIGN CHANGE ON 4-ADDITIONS TOO
F'00054C  2A7C'00000000   597.  SIPROR  MOVEA.L #NPE,A5          ;ADDR INC
F'000552  2E3C'00000000   598.          MOVE.L #SHIFT14,D7       ;/16384 SHFT
F'000558  303C'FFFF       599.          MOVE.W #NPTE-1,D0        ;MAJ LOOP CNTR
```

```
F'00055C  287C'FFFFFFFC  600.          MOVEA.L #NPT-4,A4       ;INDX TO ARRAYS
F'000562  43FA 023A      601.          LEA TWD1(PC),A1         ;TWID FACT PNTR
F'000566  41FA ****      602.          LEA DATAII(PC),A0       ;XDATA POINTER
F'00056A  2648           603.  TWDL1R  MOVEA.L A0,A3
F'00056C  D7CC           604.          ADDA.L A4,A3
F'00056E  3213           605.  ADDL1R  MOVE.W (A3),D1          ;GET REAL PART
F'000570  362B 0102      606.          MOVE.W 258(A3),D3
F'000574  9243           607.          SUB.W D3,D1
F'000576  362B 0200      608.          MOVE.W 512(A3),D3
F'00057A  9243           609.          SUB.W D3,D1
F'00057C  362B 0302      610.          MOVE.W 770(A3),D3
F'000580  D243           611.          ADD.W D3,D1             ;REAL SUM DONE
F'000582  342B 0002      612.          MOVE.W 2(A3),D2         ;IMAG SUM IN D2
F'000586  362B 0100      613.          MOVE.W 256(A3),D3
F'00058A  D443           614.          ADD.W D3,D2
F'00058C  362B 0202      615.          MOVE.W 514(A3),D3
F'000590  9443           616.          SUB.W D3,D2
F'000592  362B 0300      617.          MOVE.W 768(A3),D3
F'000596  9443           618.          SUB.W D3,D2             ;IMAG SUM DONE
                         619.  *
                         620.  * NOW DO TWIDDLE FACTOR MULTIPLY
                         621.  * MUST NEGATE THE IMAGINARY PART OF TWDF
                         622.  *
F'000598  3A01           623.          MOVE.W D1,D5
F'00059A  3831 C800      624.          MOVE.W 0(A1,A4.L),D4
F'00059E  3631 C802      625.          MOVE.W 2(A1,A4.L),D3
F'0005A2  4443           626.          NEG.W D3
F'0005A4  CBC4           627.          MULS D4,D5
F'0005A6  3C02           628.          MOVE.W D2,D6
F'0005A8  CDC3           629.          MULS D3,D6
F'0005AA  9A86           630.          SUB.L D6,D5             ;REAL IN D5
F'0005AC  C3C3           631.          MULS D3,D1
F'0005AE  C5C4           632.          MULS D4,D2
F'0005B0  D282           633.          ADD.L D2,D1             ;IMAG IN D1
F'0005B2  EEA5           634.          ASR.L D7,D5
F'0005B4  EEA1           635.          ASR.L D7,D1
                         636.  *
                         637.  * STORE SUMMED AND TWIDDLED DATA AT DATAII IN
                         638.  * PLACE
                         639.  *
F'0005B6  3185 C800      640.          MOVE.W D5,0(A0,A4.L)
F'0005BA  3181 C802      641.          MOVE.W D1,2(A0,A4.L)
F'0005BE  99CD           642.          SUBA.L A5,A4
F'0005C0  51C8 FFA8      643.          DBRA D0,TWDL1R
                         644.  *
                         645.  * INSERT FAVORITE 64-POINT FFT HERE
                         646.  * INPUT DATA AT DATAII (0-255)
                         647.  *
F'0005C4  78'F7          648.  USHFL1R MOVEQ.L #NPTE-9,D4       ;COUNTER=K
F'0005C6  3C04           649.  SWAPL1R MOVE.W D4,D6          ;PREPARE BIT REV
F'0005C8  3E06           650.  BREV21R MOVE.W D6,D7        ;SAVE DATA
F'0005CA  4246           651.          CLR.W D6            ;
F'0005CC  7A05           652.          MOVEQ.L #5,D5       ;BIT COUNTER
F'0005CE  E257           653.  REV21R  ROXR.W #1,D7        ;SFT RIGHT,XTEND
F'0005D0  E356           654.          ROXL.W #1,D6
F'0005D2  51CD FFFA      655.          DBRA D5,REV21R
F'0005D6  B846           656.          CMP.W D6,D4           ;I=REV(K)<=K?
F'0005D8  6C 14          657.          BGE.S DECK1R            ;NO SWAP IF SO
F'0005DA  3204           658.          MOVE.W D4,D1          ;COPY OF K
F'0005DC  E541           659.          ASL.W #2,D1           ;K*4
F'0005DE  E546           660.          ASL.W #2,D6           ;I*4
F'0005E0  2430 1000      661.          MOVE.L 0(A0,D1.W),D2 ;DO SWAP
F'0005E4  21B0 6000 1000 662.          MOVE.L 0(A0,D6.W),0(A0,D1.W)
F'0005EA  2182 6000      663.          MOVE.L D2,0(A0,D6.W) ;DONE SWAP
F'0005EE  51CC FFD6      664.  DECK1R   DBRA D4,SWAPL1R
F'0005F2  7201           665.  FFT641R MOVEQ.L #1,D1    ;D1=N2=1
F'0005F4  43FA 00A8      666.          LEA SINBLK1(PC),A1  ;SIN VALS PNTR A1
F'0005F8  45FA 0124      667.          LEA COSBLK1(PC),A2  ;COS VALS PNTR A2
                         668.  *
                         669.  * L IS DOWN COUNTED #NUGAM-1 TO 0
                         670.  *
F'0005FC  3F3C 0001      671.          MOVE.W #1,-(SP)  ;(SP) IS L
F'000600  7405           672.          MOVEQ.L #NUGAM-1,D2         ;D2=NUGAM-1=NU1
F'000602  4283           673.          CLR.L D3                ;D3=K=0
F'000604  70'00          674.          MOVEQ.L #SHIFT14,D0
                         675.  *
                         676.  * INITIALIZE I=N2-1, TO 0 IN STEPS -1
                         677.  *
F'000606  3801           678.  INITI1R MOVE.W D1,D4     ;D4=I=N2
F'000608  5344           679.          SUBQ.W #1,D4     ;SET FOR DBRA
                         680.  *
```

```
                                    681.   * DETERMINE P=MOD(K*2**(NUGAM-L),64)
                                    682.   *
                                    683.   *
F'00060A  3C03                      684.   LPIN1R  MOVE.W D3,D6     ;D6=K
F'00060C  3A17                      685.           MOVE.W (SP),D5   ;GET L
F'00060E  5D45                      686.           SUBQ.W #NUGAM,D5 ;L-NUGAM
F'000610  4445                      687.           NEG.W D5         ;NUGAM-L
F'000612  EB66                      688.           ASL.W D5,D6      ;K*2**(NUGAM-L)
F'000614  0246 003F                 689.           ANDI.W #$003F,D6 ;MOD 64
                                    690.   *
                                    691.   * D6 HAS INDEX REQUIRED
                                    692.
F'000618  DC46                      693.           ADD.W D6,D6      ;D6*2 FOR WORD BOUND
F'00061A  3A71 6000                 694.           MOVEA.W 0(A1,D6.W),A5 ;A5 PNTR IM(W**P)
F'00061E  2E03                      695.           MOVE.L D3,D7     ;D7=K
F'000620  2A03                      696.           MOVE.L D3,D5     ;D5 TOO
F'000622  E545                      697.           ASL.W #2,D5      ;K*4 FOR LWORD BOUND
F'000624  4DF0 5000                 698.           LEA 0(A0,D5.W),A6    ;A6 IS INDEX X(K)
F'000628  DE41                      699.           ADD.W D1,D7      ;D7=K+N2
F'00062A  E547                      700.           ASL.W #2,D7      ;D7=4(K+N2) LWORD BND
F'00062C  47F0 7000                 701.           LEA 0(A0,D7.W),A3 ;A3 PNT RE(X(K+N2))
F'000630  3A32 6000                 702.           MOVE.W 0(A2,D6.W),D5     ;GET RE(W**P)
F'000634  3E05                      703.           MOVE.W D5,D7     ;PUT ALSO IN D7
F'000636  CBD3                      704.           MULS (A3),D5     ;RE(W**P)*RE(X(K+N2))
F'000638  3C0D                      705.           MOVE.W A5,D6  ;GET IM(W**P)
F'00063A  4446                      706.           NEG.W D6
F'00063C  CDEB 0002                 707.           MULS 2(A3),D6    ;IM(W**P)*IM(X(K+N2))
F'000640  9A86                      708.           SUB.L D6,D5      ;D5=RE(T1)
F'000642  E0A5                      709.           ASR.L D0,D5      ;/16384 TO SCALE
F'000644  3C0D                      710.           MOVE.W A5,D6  ;GET IM(W**P)
F'000646  4446                      711.           NEG.W D6
F'000648  CDD3                      712.           MULS (A3),D6     ;IM(W**P)*RE(X(K+N2))
F'00064A  CFEB 0002                 713.           MULS 2(A3),D7    ;RE(W**P)*IM(X(K+N2))
F'00064E  DC87                      714.           ADD.L D7,D6      ;D6=IM(T1)
F'000650  E0A6                      715.           ASR.L D0,D6      ;/16384 TO SCALE
F'000652  3E16                      716.           MOVE.W (A6),D7   ;RE(X(K))
F'000654  9E45                      717.           SUB.W D5,D7      ;RE(X(K))-RE(T1)
F'000656  3687                      718.           MOVE.W D7,(A3)   ;STORE ANSWER
F'000658  3E2E 0002                 719.           MOVE.W 2(A6),D7 ;IM(X(K))
F'00065C  9E46                      720.           SUB.W D6,D7      ;IM(X(K))-IM(T1)
F'00065E  3747 0002                 721.           MOVE.W D7,2(A3)  ;STORE ANSWER
F'000662  DB56                      722.           ADD.W D5,(A6)    ;RE(X(K))+RE(T1)
F'000664  DD6E 0002                 723.           ADD.W D6,2(A6)   ;IM(X(K))+IM(T1)
F'000668  5243                      724.           ADDQ.W #1,D3     ;INCREMENT K
F'00066A  51CC FF9E                 725.           DBRA D4,LPIN1R   ;DO TILL I=-1
F'00066E  D641                      726.           ADD.W D1,D3      ;K=K+N2
F'000670  0C43'FFFF                 727.           CMPI.W #NPTE-1,D3 ;K<N-1 ?
F'000674  6D 90                     728.           BLT INITI1R      ;DO AGAIN IF SO
F'000676  4243                      729.           CLR.W D3         ;K=0
F'000678  5342                      730.           SUBQ.W #1,D2     ;NU1=NU1-1
F'00067A  E341                      731.           ASL.W #1,D1      ;N2=N2*2
F'00067C  5257                      732.           ADDQ.W #1,(SP)
F'00067E  0C57 0007                 733.           CMPI.W #NUGAM+1,(SP)
F'000682  66 82                     734.           BNE INITI1R
F'000684  301F                      735.           MOVE.W (SP)+,D0
                                    736.   *
                                    737.   * ASSERT SRQ AND WAIT FOR ACKNOWLEDGE
                                    738.   *
F'000686  4E43                      739.           TRAP #SRQASRT
                                    740.   * SEND OUT 64 COMPLEX POINTS CALCULATED
>>> WARNING line[741] Expr. type    [741] SLVNOMES.A68
F'000688  303C ****                 741.           MOVE.W #NPTE*4,D0        ;BYTE COUNT
F'00068C  13FC 00'00                742.           MOVE.B #ENABLEB,PGCR
          '00000000
F'000694  4E46                      743.           TRAP #OUTDATA
F'000696  4239'00000000             744.           CLR.B PGCR
                                    745.   *
                                    746.   * DONE!
                                    747.   *
F'00069C  4E75                      748.           RTS
                                    749.   *
                                    750.   * SIN (-2*PI*I/64)*16384 FOR I=0 TO 63
                                    751.   *
F'00069E  0000 F9BA F384            752.   SINBLK1 DC.W        0, -1606, -3196, -47
          ED6C
F'0006A6  E782 E1D5 DC72            753.           DC.W   -6270, -7723, -9102,-10
          D766
F'0006AE  D2BF CE87 CAC9            754.           DC.W  -11585,-12665,-13623,-14449
          C78F
F'0006B6  C4DF C2C1 C13B            755.           DC.W  -15137,-15679,-16069,-16305
          C04F
```

```
F'0006BE  C000 C04F C13B  756.          DC.W  -16384,-16305,-16069,-15679
          C2C1
F'0006C6  C4DF C78F CAC9  757.          DC.W  -15137,-14449,-13623,-12665
          CE87
F'0006CE  D2BF D766 DC72  758.          DC.W  -11585,-10394, -9102, -7723
          E1D5
F'0006D6  E782 ED6C F384  759.          DC.W   -6270, -4756, -3196, -1606
          F9BA
F'0006DE  0000 0646 0C7C  760.          DC.W       0,  1606,  3196,  4756
          1294
F'0006E6  187E 1E2B 238E  761.          DC.W    6270,  7723,  9102, 10394
          289A
F'0006EE  2D41 3179 3537  762.          DC.W   11585, 12665, 13623, 14449
          3871
F'0006F6  3B21 3D3F 3EC5  763.          DC.W   15137, 15679, 16069, 16305
          3FB1
F'0006FE  4000 3FB1 3EC5  764.          DC.W   16384, 16305, 16069, 15679
          3D3F
F'000706  3B21 3871 3537  765.          DC.W   15137, 14449, 13623, 12665
          3179
F'00070E  2D41 289A 238E  766.          DC.W   11585, 10394,  9102,  7723
          1E2B
F'000716  187E 1294 0C7C  767.          DC.W    6270,  4756,  3196,  1606
          0646
                          768.  *
                          769.  * COS(-2*PI*I/64)*16384 FOR I=0 TO 63
                          770.  *
F'00071E  4000 3FB1 3EC5  771.  COSBLK1 DC.W   16384, 16305, 16069, 15679
          3D3F
F'000726  3B21 3871 3537  772.          DC.W   15137, 14449, 13623, 12665
          3179
F'00072E  2D41 289A 238E  773.          DC.W   11585, 10394,  9102,  7723
          1E2B
F'000736  187E 1294 0C7C  774.          DC.W    6270,  4756,  3196,  1606
          0646
F'00073E  0000 F9BA F384  775.          DC.W       0, -1606, -3196, -4756
          ED6C
F'000746  E782 E1D5 DC72  776.          DC.W   -6270, -7723, -9102,-10394
          D766
F'00074E  D2BF CE87 CAC9  777.          DC.W  -11585,-12665,-13623,-14449
          C78F
F'000756  C4DF C2C1 C13B  778.          DC.W  -15137,-15679,-16069,-16305
          C04F
F'00075E  C000 C04F C13B  779.          DC.W  -16384,-16305,-16069,-15679
          C2C1
F'000766  C4DF C78F CAC9  780.          DC.W  -15137,-14449,-13623,-12665
          CE87
F'00076E  D2BF D766 DC72  781.          DC.W  -11585,-10394, -9102, -7723
          E1D5
F'000776  E782 ED6C F384  782.          DC.W   -6270, -4756, -3196, -1606
          F9BA
F'00077E  0000 0646 0C7C  783.          DC.W       0,  1606,  3196,  4756
          1294
F'000786  187E 1E2B 238E  784.          DC.W    6270,  7723,  9102, 10394
          289A
F'00078E  2D41 3179 3537  785.          DC.W   11585, 12665, 13623, 14449
          3871
F'000796  3B21 3D3F 3EC5  786.          DC.W   15137, 15679, 16069, 16305
          3FB1
                          787.  *
                          788.  * TWIDDLE FACTOR TABLE 1 TWD1
                          789.  *
F'00079E  4000 0000 3FFB  790.  TWD1    DC.W   16384,     0, 16379,  -402
          FE6E
F'0007A6  3FEC FCDC 3FD4  791.          DC.W   16364,  -804, 16340, -1205
          FB4B
F'0007AE  3FB1 F9BA 3F85  792.          DC.W   16305, -1606, 16261, -2006
          F82A
F'0007B6  3F4F F69C 3F0F  793.          DC.W   16207, -2404, 16143, -2801
          F50F
F'0007BE  3EC5 F384 3E72  794.          DC.W   16069, -3196, 15986, -3590
          F1FA
F'0007C6  3E15 F073 3DAF  795.          DC.W   15893, -3981, 15791, -4370
          EEEE
F'0007CE  3D3F ED6C 3CC5  796.          DC.W   15679, -4756, 15557, -5139
          EBED
F'0007D6  3C42 EA70 3BB6  797.          DC.W   15426, -5520, 15286, -5897
          E8F7
F'0007DE  3B21 E782 3A82  798.          DC.W   15137, -6270, 14978, -6639
          E611
F'0007E6  39DB E4A3 392B  799.          DC.W   14811, -7005, 14635, -7366
```

```
                E33A
F'0007EE  3871 E1D5 37B0  800.          DC.W   14449, -7723, 14256, -8076
                E074
F'0007F6  36E5 DF19 3612  801.          DC.W   14053, -8423, 13842, -8765
                DDC3
F'0007FE  3537 DC72 3453  802.          DC.W   13623, -9102, 13395, -9434
                DB26
F'000806  3368 D9E0 3274  803.          DC.W   13160, -9760, 12916,-10080
                D8A0
F'00080E  3179 D766 3076  804.          DC.W   12665,-10394, 12406,-10702
                D632
F'000816  2F6C D505 2E5A  805.          DC.W   12140,-11003, 11866,-11297
                D3DF
F'00081E  2D41 D2BF 2C21  806.          DC.W   11585,-11585, 11297,-11866
                D1A6
F'000826  2AFB D094 29CE  807.          DC.W   11003,-12140, 10702,-12406
                CF8A
F'00082E  289A CE87 2760  808.          DC.W   10394,-12665, 10080,-12916
                CD8C
F'000836  2620 CC98 24DA  809.          DC.W    9760,-13160,  9434,-13395
                CBAD
F'00083E  238E CAC9 223D  810.          DC.W    9102,-13623,  8765,-13842
                C9EE
F'000846  20E7 C91B 1F8C  811.          DC.W    8423,-14053,  8076,-14256
                C850
F'00084E  1E2B C78F 1CC6  812.          DC.W    7723,-14449,  7366,-14635
                C6D5
F'000856  1B5D C625 19EF  813.          DC.W    7005,-14811,  6639,-14978
                C57E
F'00085E  187E C4DF 1709  814.          DC.W    6270,-15137,  5897,-15286
                C44A
F'000866  1590 C3BE 1413  815.          DC.W    5520,-15426,  5139,-15557
                C33B
F'00086E  1294 C2C1 1112  816.          DC.W    4756,-15679,  4370,-15791
                C251
F'000876  0F8D C1EB 0E06  817.          DC.W    3981,-15893,  3590,-15986
                C18E
F'00087E  0C7C C13B 0AF1  818.          DC.W    3196,-16069,  2801,-16143
                C0F1
F'000886  0964 C0B1 07D6  819.          DC.W    2404,-16207,  2006,-16261
                C07B
F'00088E  0646 C04F 04B5  820.          DC.W    1606,-16305,  1205,-16340
                C02C
F'000896  0324 C014 0192  821.          DC.W     804,-16364,   402,-16379
                C005
                          822.
                          823.  ********************
                          824.  * SLAVE 2 PROGRAM *
                          825.  ********************
                          826.  *
                          827.  * INPUT DATA AT DATAII, TWIDDLE FACTORS AT
                          828.  * TWD2
                          829.  *
F'00089E  2A7C'00000000   830.  S2PRO   MOVEA.L #NPE,A5          ;ADDR INC
F'0008A4  2E3C'00000000   831.          MOVE.L #SHIFT14,D7      ;/16384 SHFT
F'0008AA  303C'FFFF       832.          MOVE.W #NPTE-1,D0       ;MAJ LOOP CNTR
F'0008AE  287C'FFFFFFFC   833.          MOVEA.L #NPT-4,A4       ;INDX TO ARRAYS
F'0008B4  43FA 03EC       834.          LEA TWD2(PC),A1         ;TWID FACT PNTR
F'0008B8  41FA ****       835.          LEA DATAII(PC),A0       ;XDATA POINTER
F'0008BC  2648            836.  TWDL2   MOVEA.L A0,A3
F'0008BE  D7CC            837.          ADDA.L A4,A3
F'0008C0  3213            838.  ADDL2   MOVE.W (A3),D1          ;GET REAL PART
F'0008C2  48C1            839.          EXT.L D1                ;REAL SUM IN D1
F'0008C4  362B 0100       840.          MOVE.W 256(A3),D3
F'0008C8  48C3            841.          EXT.L D3
F'0008CA  9283            842.          SUB.L D3,D1
F'0008CC  362B 0200       843.          MOVE.W 512(A3),D3
F'0008D0  48C3            844.          EXT.L D3
F'0008D2  D283            845.          ADD.L D3,D1
F'0008D4  362B 0300       846.          MOVE.W 768(A3),D3
F'0008D8  48C3            847.          EXT.L D3
F'0008DA  9283            848.          SUB.L D3,D1             ;REAL SUM DONE
F'0008DC  342B 0002       849.          MOVE.W 2(A3),D2         ;IMAG SUM IN D2
F'0008E0  48C2            850.          EXT.L D2
F'0008E2  362B 0102       851.          MOVE.W 258(A3),D3
F'0008E6  48C3            852.          EXT.L D3
F'0008E8  9483            853.          SUB.L D3,D2
F'0008EA  362B 0202       854.          MOVE.W 514(A3),D3
F'0008EE  48C3            855.          EXT.L D3
F'0008F0  D483            856.          ADD.L D3,D2
F'0008F2  362B 0302       857.          MOVE.W 770(A3),D3
```

```
F'0008F6  48C3                858.          EXT.L D3
F'0008F8  9483                859.          SUB.L D3,D2             ;IMAG SUM DONE
F'0008FA  E481                860.          ASR.L #2,D1
F'0008FC  E482                861.          ASR.L #2,D2             ;ADJUST SUMS
                              862.  *
                              863.  * NOW DO TWIDDLE FACTOR MULTIPLY
                              864.  *
F'0008FE  3A01                865.          MOVE.W D1,D5
F'000900  3831 C800           866.          MOVE.W 0(A1,A4.L),D4
F'000904  3631 C802           867.          MOVE.W 2(A1,A4.L),D3
F'000908  CBC4                868.          MULS D4,D5
F'00090A  3C02                869.          MOVE.W D2,D6
F'00090C  CDC3                870.          MULS D3,D6
F'00090E  9A86                871.          SUB.L D6,D5             ;REAL IN D5
F'000910  C3C3                872.          MULS D3,D1
F'000912  C5C4                873.          MULS D4,D2
F'000914  D282                874.          ADD.L D2,D1             ;IMAG IN D1
F'000916  EEA5                875.          ASR.L D7,D5
F'000918  EEA1                876.          ASR.L D7,D1
                              877.  *
                              878.  * STORE SUMMED AND TWIDDLED DATA AT XDATA IN
                              879.  * PLACE
                              880.  *
F'00091A  3185 C800           881.          MOVE.W D5,0(A0,A4.L)
F'00091E  3181 C802           882.          MOVE.W D1,2(A0,A4.L)
F'000922  99CD                883.          SUBA.L A5,A4
F'000924  51C8 FF96           884.          DBRA D0,TWDL2
                              885.  *
                              886.  * INSERT FAVORITE 64-POINT FFT HERE
                              887.  * INPUT DATA AT XDATA 0-255
                              888.  *
F'000928  78'F7               889.  USHFL2  MOVEQ.L #NPTE-9,D4       ;COUNTER=K
F'00092A  3C04                890.  SWAPL2  MOVE.W D4,D6          ;PREPARE BIT REV
F'00092C  3E06                891.  BREV22  MOVE.W D6,D7        ;SAVE DATA
F'00092E  4246                892.          CLR.W D6            ;
F'000930  7A05                893.          MOVEQ.L #5,D5       ;BIT COUNTER
F'000932  E257                894.  REV22   ROXR.W #1,D7        ;SFT RIGHT,XTEND
F'000934  E356                895.          ROXL.W #1,D6
F'000936  51CD FFFA           896.          DBRA D5,REV22
F'00093A  B846                897.          CMP.W D6,D4           ;I=REV(K)<=K?
F'00093C  6C 14               898.          BGE.S DECK2            ;NO SWAP IF SO
F'00093E  3204                899.          MOVE.W D4,D1          ;COPY OF K
F'000940  E541                900.          ASL.W #2,D1           ;K*4
F'000942  E546                901.          ASL.W #2,D6           ;I*4
F'000944  2430 1000           902.          MOVE.L 0(A0,D1.W),D2 ;DO SWAP
F'000948  21B0 6000 1000      903.          MOVE.L 0(A0,D6.W),0(A0,D1.W)
F'00094E  2182 6000           904.          MOVE.L D2,0(A0,D6.W) ;DONE SWAP
F'000952  51CC FFD6           905.  DECK2   DBRA D4,SWAPL2
F'000956  7201                906.  FFT642  MOVEQ.L #1,D1   ;D1=N2=1
F'000958  43FA 0248           907.          LEA SINBLK2(PC),A1  ;SIN VALS PNTR A1
F'00095C  45FA 02C4           908.          LEA COSBLK2(PC),A2  ;COS VALS PNTR A2
                              909.  *
                              910.  * L IS DOWN COUNTED #NUGAM-1 TO 0
                              911.  *
F'000960  3F3C 0001           912.          MOVE.W #1,-(SP)  ;(SP) IS L
F'000964  7405                913.          MOVEQ.L #NUGAM-1,D2        ;D2=NUGAM-1=NU1
F'000966  4283                914.          CLR.L D3             ;D3=K=0
F'000968  70'00               915.          MOVEQ.L #SHIFT14,D0
                              916.  *
                              917.  * INITIALIZE I=N2-1, TO 0 IN STEPS -1
                              918.  *
F'00096A  3801                919.  INITI2  MOVE.W D1,D4    ;D4=I=N2
F'00096C  5344                920.          SUBQ.W #1,D4    ;SET FOR DBRA
                              921.  *
                              922.  * DETERMINE P=MOD(K*2**(NUGAM-L),64)
                              923.  *
                              924.  *
F'00096E  3C03                925.  LOOPIN2 MOVE.W D3,D6    ;D6=K
F'000970  3A17                926.          MOVE.W (SP),D5  ;GET L
F'000972  5D45                927.          SUBQ.W #NUGAM,D5 ;L-NUGAM
F'000974  4445                928.          NEG.W D5        ;NUGAM-L
F'000976  EB66                929.          ASL.W D5,D6     ;K*2**(NUGAM-L)
F'000978  0246 003F           930.          ANDI.W #$003F,D6 ;MOD 64
                              931.  *
                              932.  * D6 HAS INDEX REQUIRED
                              933.  *
F'00097C  DC46                934.          ADD.W D6,D6     ;D6*2 FOR WORD BOUND
F'00097E  3A71 6000           935.          MOVEA.W 0(A1,D6.W),A5 ;A5 PNTR IM(W**P)
F'000982  2E03                936.          MOVE.L D3,D7    ;D7=K
F'000984  2A03                937.          MOVE.L D3,D5    ;D5 TOO
F'000986  E545                938.          ASL.W #2,D5     ;K*4 FOR LWORD BOUND
```

```
F'000988  4DF0 5000        939.          LEA 0(A0,D5.W),A6   ;A6 IS INDEX X(K)
F'00098C  DE41             940.          ADD.W D1,D7         ;D7=K+N2
F'00098E  E547             941.          ASL.W #2,D7         ;D7=4(K+N2) LWORD BND
F'000990  47F0 7000        942.          LEA 0(A0,D7.W),A3 ;A3 PNT RE(X(K+N2))
F'000994  3A32 6000        943.          MOVE.W 0(A2,D6.W),D5     ;GET RE(W**P)
F'000998  3E05             944.          MOVE.W D5,D7        ;PUT ALSO IN D7
F'00099A  CBD3             945.          MULS (A3),D5        ;RE(W**P)*RE(X(K+N2))
F'00099C  3C0D             946.          MOVE.W A5,D6  ;GET IM(W**P)
F'00099E  CDEB 0002        947.          MULS 2(A3),D6       ;IM(W**P)*IM(X(K+N2))
F'0009A2  9A86             948.          SUB.L D6,D5         ;D5=RE(T1)
F'0009A4  E0A5             949.          ASR.L D0,D5         ;/16384 TO SCALE
F'0009A6  3C0D             950.          MOVE.W A5,D6  ;GET IM(W**P)
F'0009A8  CDD3             951.          MULS (A3),D6        ;IM(W**P)*RE(X(K+N2))
F'0009AA  CFEB 0002        952.          MULS 2(A3),D7       ;RE(W**P)*IM(X(K+N2))
F'0009AE  DC87             953.          ADD.L D7,D6         ;D6=IM(T1)
F'0009B0  E0A6             954.          ASR.L D0,D6         ;/16384 TO SCALE
F'0009B2  3E16             955.          MOVE.W (A6),D7  ;RE(X(K))
F'0009B4  9E45             956.          SUB.W D5,D7         ;RE(X(K))-RE(T1)
F'0009B6  E247             957.          ASR.W #1,D7         ;/2
F'0009B8  3687             958.          MOVE.W D7,(A3)  ;STORE ANSWER
F'0009BA  3E2E 0002        959.          MOVE.W 2(A6),D7 ;IM(X(K))
F'0009BE  9E46             960.          SUB.W D6,D7         ;IM(X(K))-IM(T1)
F'0009C0  E247             961.          ASR.W #1,D7         ;/2
F'0009C2  3747 0002        962.          MOVE.W D7,2(A3) ;STORE ANSWER
F'0009C6  DB56             963.          ADD.W D5,(A6)   ;RE(X(K))+RE(T1)
F'0009C8  E0D6             964.          ASR.W (A6)          ;/2
F'0009CA  DD6E 0002        965.          ADD.W D6,2(A6)  ;IM(X(K))+IM(T1)
F'0009CE  E0EE 0002        966.          ASR.W 2(A6)         ;/2
F'0009D2  5243             967.          ADDQ.W #1,D3        ;INCREMENT K
F'0009D4  51CC FF98        968.          DBRA D4,LOOPIN2  ;DO TILL I=-1
F'0009D8  D641             969.          ADD.W D1,D3         ;K=K+N2
F'0009DA  0C43'FFFF        970.          CMPI.W #NPTE-1,D3 ;K<N-1 ?
F'0009DE  6D 8A            971.          BLT INITI2       ;DO AGAIN IF SO
F'0009E0  4243             972.          CLR.W D3            ;K=0
F'0009E2  5342             973.          SUBQ.W #1,D2        ;NU1=NU1-1
F'0009E4  E341             974.          ASL.W #1,D1         ;N2=N2*2
F'0009E6  5257             975.          ADDQ.W #1,(SP)
F'0009E8  0C57 0007        976.          CMPI.W #NUGAM+1,(SP)
F'0009EC  6600 FF7C        977.          BNE INITI2    ;DO TILL L=-1
F'0009F0  301F             978.          MOVE.W (SP)+,D0
                           979.  *
                           980.  * NOW DO FILTERING, FILTER DATA IS AT "FILTER"
                           981.  * 64 POINTS
                           982.  *
                           983.  * DO COMPLEX MULTIPLY BY THE FILTER.
                           984.  * FILTER CAN BE +16384 TO -16384 WITH 16384
                           985.  * BEING THE 0 db GAIN VALUE.
                           986.  * (A+iB)*(C+iD)
                           987.  *
F'0009F2  43FA ****        988.          LEA FILTER(PC),A1        ;A1 PNT FILTER
F'0009F6  303C'FFFF        989.          MOVE.W #NPTE-1,D0        ;LOOP COUNTER
F'0009FA  7A'00            990.          MOVEQ.L #SHIFT14,D5              ;SHIFT COUNT
F'0009FC  3219             991.  FILOOP2 MOVE.W (A1)+,D1          ;GET C
F'0009FE  3601             992.          MOVE.W D1,D3
F'000A00  C3D8             993.          MULS (A0)+,D1            ;DO AC
F'000A02  3419             994.          MOVE.W (A1)+,D2          ;GET D
F'000A04  3802             995.          MOVE.W D2,D4
F'000A06  C5D0             996.          MULS (A0),D2             ;DO BD
F'000A08  9282             997.          SUB.L D2,D1              ;AC-BD
F'000A0A  EAA1             998.          ASR.L D5,D1              ;ADJUST /16384
F'000A0C  C7D0             999.          MULS (A0),D3             ;DO BC
F'000A0E  C9E0            1000.          MULS -(A0),D4            ;DO AD
F'000A10  D883            1001.          ADD.L D3,D4              ;BC+AD
F'000A12  EAA4            1002.          ASR.L D5,D4              ;ADJUST /16384
F'000A14  30C1            1003.          MOVE.W D1,(A0)+ ;STORE REAL
F'000A16  30C4            1004.          MOVE.W D4,(A0)+ ;STORE IMAG
F'000A18  51C8 FFE2       1005.          DBRA D0,FILOOP2
                          1006.  *
                          1007.  * ASSERT SRQ AND WAIT FOR ACKNOWLEDGE
                          1008.  *
F'000A1C  4E43            1009.          TRAP #SRQASRT
                          1010.  * SEND OUT 64 COMPLEX POINTS CALCULATED
>>> WARNING line[1011] Expr. type   [1011] SLVNOMES.A68
F'000A1E  303C ****       1011.          MOVE.W #NPTE*4,D0        ;BYTE COUNT
```

```
F'000A22  41FA ****       1012.          LEA DATAII(PC),A0
F'000A26  13FC 00'00      1013.          MOVE.B #ENABLEB,PGCR
          '00000000
F'000A2E  4E46            1014.          TRAP #OUTDATA
F'000A30  4239'00000000   1015.          CLR.B PGCR
                          1016.  *
                          1017.  * ASSERT AN SRQ TO SIGNAL READY FOR NEW DATA
                          1018.  * AND RECEIVE NEW FILTERED DATA TO BE INVERSE
                          1019.  * TRANSFORMED
                          1020.  *
F'000A36  4E43            1021.          TRAP #SRQASRT
>>> WARNING line[1022] Expr. type   [1022] SLVNOMES.A68
F'000A38  303C ****       1022.          MOVE.W #NPT*4,D0          ;BYTE COUNT
F'000A3C  41FA ****       1023.          LEA DATAII(PC),A0
F'000A40  13FC 00'00      1024.          MOVE.B #ENABLEA,PGCR
          '00000000
F'000A48  4E45            1025.          TRAP #INDATA
F'000A4A  4239'00000000   1026.          CLR.B PGCR
                          1027.  * NOW DO THE INVERSE TRANSFORM
                          1028.  * USE SAME SIN AND COS TABLES, NEGATE THE SIN
                          1029.  * ENTRY, AND USE NO ASR #2 FOR DIVISION BY NPTE
                          1030.  * NOTE SIGN CHANGE ON 4-ADDITIONS TOO
                          1031.  *
F'000A50  2A7C'00000000   1032.  S2PROR  MOVEA.L #NPE,A5           ;ADDR INC
F'000A56  2E3C'00000000   1033.          MOVE.L #SHIFT14,D7        ;/16384 SHFT
F'000A5C  303C'FFFF       1034.          MOVE.W #NPTE-1,D0         ;MAJ LOOP CNTR
F'000A60  287C'FFFFFFFC   1035.          MOVEA.L #NPT-4,A4         ;INDX TO ARRAYS
F'000A66  43FA 023A       1036.          LEA TWD2(PC),A1           ;TWID FACT PNTR
F'000A6A  41FA ****       1037.          LEA DATAII(PC),A0         ;XDATA POINTER
F'000A6E  2648            1038.  TWDL2R  MOVEA.L A0,A3
F'000A70  D7CC            1039.          ADDA.L A4,A3
F'000A72  3613            1040.  ADDL2R  MOVE.W (A3),D1            ;GET REAL PART
F'000A74  362B 0100       1041.          MOVE.W 256(A3),D3
F'000A78  9243            1042.          SUB.W D3,D1
F'000A7A  362B 0200       1043.          MOVE.W 512(A3),D3
F'000A7E  D243            1044.          ADD.W D3,D1
F'000A80  362B 0300       1045.          MOVE.W 768(A3),D3
F'000A84  9243            1046.          SUB.W D3,D1               ;REAL SUM DONE
F'000A86  342B 0002       1047.          MOVE.W 2(A3),D2           ;IMAG SUM IN D2
F'000A8A  362B 0102       1048.          MOVE.W 258(A3),D3
F'000A8E  9443            1049.          SUB.W D3,D2
F'000A90  362B 0202       1050.          MOVE.W 514(A3),D3
F'000A94  D443            1051.          ADD.W D3,D2
F'000A96  362B 0302       1052.          MOVE.W 770(A3),D3
F'000A9A  9443            1053.          SUB.W D3,D2               ;IMAG SUM DONE
                          1054.  *
                          1055.  * NOW DO TWIDDLE FACTOR MULTIPLY
                          1056.  *
F'000A9C  3A01            1057.          MOVE.W D1,D5
F'000A9E  3831 C800       1058.          MOVE.W 0(A1,A4.L),D4
F'000AA2  3631 C802       1059.          MOVE.W 2(A1,A4.L),D3
F'000AA6  4443            1060.          NEG.W D3
F'000AA8  CBC4            1061.          MULS D4,D5
F'000AAA  3C02            1062.          MOVE.W D2,D6
F'000AAC  CDC3            1063.          MULS D3,D6
F'000AAE  9A86            1064.          SUB.L D6,D5               ;REAL IN D5
F'000AB0  C3C3            1065.          MULS D3,D1
F'000AB2  C5C4            1066.          MULS D4,D2
F'000AB4  D282            1067.          ADD.L D2,D1               ;IMAG IN D1
F'000AB6  EEA5            1068.          ASR.L D7,D5
F'000AB8  EEA1            1069.          ASR.L D7,D1
                          1070.  *
                          1071.  * STORE SUMMED AND TWIDDLED DATA AT XDATA IN
                          1072.  * PLACE
                          1073.  *
F'000ABA  3185 C800       1074.          MOVE.W D5,0(A0,A4.L)
F'000ABE  3181 C802       1075.          MOVE.W D1,2(A0,A4.L)
F'000AC2  99CD            1076.          SUBA.L A5,A4
F'000AC4  51C8 FFA8       1077.          DBRA D0,TWDL2R
                          1078.  *
                          1079.  * INSERT FAVORITE 64-POINT FFT HERE
                          1080.  * INPUT DATA AT XDATA 0-255
                          1081.  *
                          1082.  *
F'000AC8  78'F7           1083.  USHFL2R MOVEQ.L #NPTE-9,D4          ;COUNTER=K
F'000ACA  3C04            1084.  SWAPL2R MOVE.W D4,D6            ;PREPARE BIT REV
F'000ACC  3E06            1085.  BREV22R MOVE.W D6,D7          ;SAVE DATA
F'000ACE  4246            1086.          CLR.W D6              ;
F'000AD0  7A05            1087.          MOVEQ.L #5,D5         ;BIT COUNTER
F'000AD2  E257            1088.  REV22R  ROXR.W #1,D7          ;SFT RIGHT,XTEND
F'000AD4  E356            1089.          ROXL.W #1,D6
```

```
F'000AD6  51CD FFFA       1090.          DBRA D5,REV22R
F'000ADA  B846            1091.          CMP.W D6,D4           ;I=REV(K)<=K?
F'000ADC  6C 14           1092.          BGE.S DECK2R           ;NO SWAP IF SO
F'000ADE  3204            1093.          MOVE.W D4,D1          ;COPY OF K
F'000AE0  E541            1094.          ASL.W #2,D1           ;K*4
F'000AE2  E546            1095.          ASL.W #2,D6           ;I*4
F'000AE4  2430 1000       1096.          MOVE.L 0(A0,D1.W),D2 ;DO SWAP
F'000AE8  21B0 6000 1000  1097.          MOVE.L 0(A0,D6.W),0(A0,D1.W)
F'000AEE  2182 6000       1098.          MOVE.L D2,0(A0,D6.W) ;DONE SWAP
F'000AF2  51CC FFD6       1099.  DECK2R   DBRA D4,SWAPL2R
F'000AF6  7201            1100.  FFT642R MOVEQ.L #1,D1    ;D1=N2=1
F'000AF8  43FA 00A8       1101.          LEA SINBLK2(PC),A1  ;SIN VALS PNTR A1
F'000AFC  45FA 0124       1102.          LEA COSBLK2(PC),A2  ;COS VALS PNTR A2
                          1103.  *
                          1104.  * L IS DOWN COUNTED #NUGAM-1 TO 0
                          1105.  *
F'000B00  3F3C 0001       1106.          MOVE.W #1,-(SP)  ;(SP) IS L
F'000B04  7405            1107.          MOVEQ.L #NUGAM-1,D2        ;D2=NUGAM-1=NU1
F'000B06  4283            1108.          CLR.L D3                ;D3=K=0
F'000B08  70'00           1109.          MOVEQ.L #SHIFT14,D0
                          1110.  *
                          1111.  * INITIALIZE I=N2-1, TO 0 IN STEPS -1
                          1112.  *
F'000B0A  3801            1113.  INITI2R MOVE.W D1,D4    ;D4=I=N2
F'000B0C  5344            1114.          SUBQ.W #1,D4    ;SET FOR DBRA
                          1115.  *
                          1116.  * DETERMINE P=MOD(K*2**(NUGAM-L),64)
                          1117.  *
                          1118.  *
F'000B0E  3C03            1119.  LPIN2R  MOVE.W D3,D6    ;D6=K
F'000B10  3A17            1120.          MOVE.W (SP),D5  ;GET L
F'000B12  5D45            1121.          SUBQ.W #NUGAM,D5 ;L-NUGAM
F'000B14  4445            1122.          NEG.W D5        ;NUGAM-L
F'000B16  EB66            1123.          ASL.W D5,D6     ;K*2**(NUGAM-L)
F'000B18  0246 003F       1124.          ANDI.W #$003F,D6 ;MOD 64
                          1125.  *
                          1126.  * D6 HAS INDEX REQUIRED
                          1127.  *
F'000B1C  DC46            1128.          ADD.W D6,D6     ;D6*2 FOR WORD BOUND
F'000B1E  3A71 6000       1129.          MOVEA.W 0(A1,D6.W),A5 ;A5 PNTR IM(W**P)
F'000B22  2E03            1130.          MOVE.L D3,D7    ;D7=K
F'000B24  2A03            1131.          MOVE.L D3,D5    ;D5 TOO
F'000B26  E545            1132.          ASL.W #2,D5     ;K*4 FOR LWORD BOUND
F'000B28  4DF0 5000       1133.          LEA 0(A0,D5.W),A6   ;A6 IS INDEX X(K)
F'000B2C  DE41            1134.          ADD.W D1,D7     ;D7=K+N2
F'000B2E  E547            1135.          ASL.W #2,D7     ;D7=4(K+N2) LWORD BND
F'000B30  47F0 7000       1136.          LEA 0(A0,D7.W),A3 ;A3 PNT RE(X(K+N2))
F'000B34  3A32 6000       1137.          MOVE.W 0(A2,D6.W),D5    ;GET RE(W**P)
F'000B38  3E05            1138.          MOVE.W D5,D7    ;PUT ALSO IN D7
F'000B3A  CBD3            1139.          MULS (A3),D5    ;RE(W**P)*RE(X(K+N2))
F'000B3C  3C0D            1140.          MOVE.W A5,D6  ;GET IM(W**P)
F'000B3E  4446            1141.          NEG.W D6
F'000B40  CDEB 0002       1142.          MULS 2(A3),D6   ;IM(W**P)*IM(X(K+N2))
F'000B44  9A86            1143.          SUB.L D6,D5     ;D5=RE(T1)
F'000B46  E0A5            1144.          ASR.L D0,D5     ;/16384 TO SCALE
F'000B48  3C0D            1145.          MOVE.W A5,D6  ;GET IM(W**P)
F'000B4A  4446            1146.          NEG.W D6
F'000B4C  CDD3            1147.          MULS (A3),D6    ;IM(W**P)*RE(X(K+N2))
F'000B4E  CFEB 0002       1148.          MULS 2(A3),D7   ;RE(W**P)*IM(X(K+N2))
F'000B52  DC87            1149.          ADD.L D7,D6     ;D6=IM(T1)
F'000B54  E0A6            1150.          ASR.L D0,D6     ;/16384 TO SCALE
F'000B56  3E16            1151.          MOVE.W (A6),D7  ;RE(X(K))
F'000B58  9E45            1152.          SUB.W D5,D7     ;RE(X(K))-RE(T1)
F'000B5A  3687            1153.          MOVE.W D7,(A3)  ;STORE ANSWER
F'000B5C  3E2E 0002       1154.          MOVE.W 2(A6),D7 ;IM(X(K))
F'000B60  9E46            1155.          SUB.W D6,D7     ;IM(X(K))-IM(T1)
F'000B62  3747 0002       1156.          MOVE.W D7,2(A3) ;STORE ANSWER
F'000B66  DB56            1157.          ADD.W D5,(A6)   ;RE(X(K))+RE(T1)
F'000B68  DD6E 0002       1158.          ADD.W D6,2(A6)  ;IM(X(K))+IM(T1)
F'000B6C  5243            1159.          ADDQ.W #1,D3    ;INCREMENT K
F'000B6E  51CC FF9E       1160.          DBRA D4,LPIN2R  ;DO TILL I=-1
F'000B72  D641            1161.          ADD.W D1,D3     ;K=K+N2
F'000B74  0C43'FFFF       1162.          CMPI.W #NPTE-1,D3 ;K<N-1 ?
F'000B78  6D 90           1163.          BLT INITI2R     ;DO AGAIN IF SO
F'000B7A  4243            1164.          CLR.W D3        ;K=0
F'000B7C  5342            1165.          SUBQ.W #1,D2    ;NU1=NU1-1
F'000B7E  E341            1166.          ASL.W #1,D1     ;N2=N2*2
F'000B80  5257            1167.          ADDQ.W #1,(SP)
F'000B82  0C57 0007       1168.          CMPI.W #NUGAM+1,(SP)
F'000B86  66 82           1169.          BNE INITI2R   ;DO TILL L=-1
F'000B88  301F            1170.          MOVE.W (SP)+,D0
```

```
                                1171.  *
                                1172.  * ASSERT SRQ AND WAIT FOR ACKNOWLEDGE
                                1173.  *
F'000B8A  4E43                  1174.          TRAP #SRQASRT
                                1175.  * SEND OUT 64 COMPLEX POINTS CALCULATED
>>> WARNING line[1176] Expr. type    [1176] SLVNOMES.A68
F'000B8C  303C ****             1176.          MOVE.W #NPTE*4,D0
F'000B90  13FC 00'00            1177.          MOVE.B #ENABLEB,PGCR
          '00000000
F'000B98  4E46                  1178.          TRAP #OUTDATA
F'000B9A  4239'00000000         1179.          CLR.B PGCR
                                1180.  *
                                1181.  * NOW GO RTS AND RETURN TO READY FOR DATA
                                1182.  * STATE TO DO OVER!
F'000BA0  4E75                  1183.          RTS
                                1184.  *
                                1185.  * DONE!
                                1186.  *
                                1187.  * SIN (-2*PI*I/64)*16384 FOR I=0 TO 63
                                1188.  *
F'000BA2  0000 F9BA F384        1189.  SINBLK2 DC.W       0, -1606, -3196, -4756
          ED6C
F'000BAA  E782 E1D5 DC72        1190.          DC.W   -6270, -7723, -9102,-10394
          D766
F'000BB2  D2BF CE87 CAC9        1191.          DC.W  -11585,-12665,-13623,-14449
          C78F
F'000BBA  C4DF C2C1 C13B        1192.          DC.W  -15137,-15679,-16069,-16305
          C04F
F'000BC2  C000 C04F C13B        1193.          DC.W  -16384,-16305,-16069,-15679
          C2C1
F'000BCA  C4DF C78F CAC9        1194.          DC.W  -15137,-14449,-13623,-12665
          CE87
F'000BD2  D2BF D766 DC72        1195.          DC.W  -11585,-10394, -9102, -7723
          E1D5
F'000BDA  E782 ED6C F384        1196.          DC.W   -6270, -4756, -3196, -1606
          F9BA
F'000BE2  0000 0646 0C7C        1197.          DC.W       0,  1606,  3196,  4756
          1294
F'000BEA  187E 1E2B 238E        1198.          DC.W    6270,  7723,  9102, 10394
          289A
F'000BF2  2D41 3179 3537        1199.          DC.W   11585, 12665, 13623, 14449
          3871
F'000BFA  3B21 3D3F 3EC5        1200.          DC.W   15137, 15679, 16069, 16305
          3FB1
F'000C02  4000 3FB1 3EC5        1201.          DC.W   16384, 16305, 16069, 15679
          3D3F
F'000C0A  3B21 3871 3537        1202.          DC.W   15137, 14449, 13623, 12665
          3179
F'000C12  2D41 289A 238E        1203.          DC.W   11585, 10394,  9102,  7723
          1E2B
F'000C1A  187E 1294 0C7C        1204.          DC.W    6270,  4756,  3196,  1606
          0646
                                1205.  *
                                1206.  * COS(-2*PI*I/64)*16384 FOR I=0 TO 63
                                1207.  *
F'000C22  4000 3FB1 3EC5        1208.  COSBLK2 DC.W   16384, 16305, 16069, 15679
          3D3F
F'000C2A  3B21 3871 3537        1209.          DC.W   15137, 14449, 13623, 12665
          3179
F'000C32  2D41 289A 238E        1210.          DC.W   11585, 10394,  9102,  7723
          1E2B
F'000C3A  187E 1294 0C7C        1211.          DC.W    6270,  4756,  3196,  1606
          0646
F'000C42  0000 F9BA F384        1212.          DC.W       0, -1606, -3196, -4756
          ED6C
F'000C4A  E782 E1D5 DC72        1213.          DC.W   -6270, -7723, -9102,-10394
          D766
F'000C52  D2BF CE87 CAC9        1214.          DC.W  -11585,-12665,-13623,-14449
          C78F
F'000C5A  C4DF C2C1 C13B        1215           DC.W  -15137,-15679,-16069,-16305
          C04F
F'000C62  C000 C04F C13B        1216.          DC.W  -16384,-16305,-16069,-15679
          C2C1
F'000C6A  C4DF C78F CAC9        1217.          DC.W  -15137,-14449,-13623,-12665
          CE87
F'000C72  D2BF D766 DC72        1218.          DC.W  -11585,-10394, -9102, -7723
          E1D5
F'000C7A  E782 ED6C F384        1219.          DC.W   -6270, -4756, -3196, -1606
          F9BA
F'000C82  0000 0646 0C7C        1220.          DC.W       0,  1606,  3196,  4756
          1294
```

```
F'000C8A  187E 1E2B 238E 1221.          DC.W    6270,  7723,  9102, 10394
          289A
F'000C92  2D41 3179 3537 1222.          DC.W   11585, 12665, 13623, 14449
          3871
F'000C9A  3B21 3D3F 3EC5 1223.          DC.W   15137, 15679, 16069, 16305
          3FB1
                       1224.  *
                       1225.  * TWIDDLE FACTOR TABLE TWD2
                       1226.  *
F'000CA2  4000 0000 3FEC 1227.  TWD2    DC.W   16384,     0, 16364,  -804
          FCDC
F'000CAA  3FB1 F9BA 3F4F 1228.          DC.W   16305, -1606, 16207, -2404
          F69C
F'000CB2  3EC5 F384 3E15 1229.          DC.W   16069, -3196, 15893, -3981
          F073
F'000CBA  3D3F ED6C 3C42 1230.          DC.W   15679, -4756, 15426, -5520
          EA70
F'000CC2  3B21 E782 39DB 1231.          DC.W   15137, -6270, 14811, -7005
          E4A3
F'000CCA  3871 E1D5 36E5 1232.          DC.W   14449, -7723, 14053, -8423
          DF19
F'000CD2  3537 DC72 3368 1233.          DC.W   13623, -9102, 13160, -9760
          D9E0
F'000CDA  3179 D766 2F6C 1234.          DC.W   12665,-10394, 12140,-11003
          D505
F'000CE2  2D41 D2BF 2AFB 1235.          DC.W   11585,-11585, 11003,-12140
          D094
F'000CEA  289A CE87 2620 1236.          DC.W   10394,-12665,  9760,-13160
          CC98
F'000CF2  238E CAC9 20E7 1237.          DC.W    9102,-13623,  8423,-14053
          C91B
F'000CFA  1E2B C78F 1B5D 1238.          DC.W    7723,-14449,  7005,-14811
          C625
F'000D02  187E C4DF 1590 1239.          DC.W    6270,-15137,  5520,-15426
          C3BE
F'000D0A  1294 C2C1 0F8D 1240.          DC.W    4756,-15679,  3981,-15893
          C1EB
F'000D12  0C7C C13B 0964 1241.          DC.W    3196,-16069,  2404,-16207
          C0B1
F'000D1A  0646 C04F 0324 1242.          DC.W    1606,-16305,   804,-16364
          C014
F'000D22  0000 C000 FCDC 1243.          DC.W       0,-16384,  -804,-16364
          C014
F'000D2A  F9BA C04F F69C 1244.          DC.W   -1606,-16305, -2404,-16207
          C0B1
F'000D32  F384 C13B F073 1245.          DC.W   -3196,-16069, -3981,-15893
          C1EB
F'000D3A  ED6C C2C1 EA70 1246.          DC.W   -4756,-15679, -5520,-15426
          C3BE
F'000D42  E782 C4DF E4A? 1247.          DC.W   -6270,-15137, -7005,-14811
          C625
F'000D4A  E1D5 C78F DF19 1248.          DC.W   -7723,-14449, -8423,-14053
          C91B
F'000D52  DC72 CAC9 D9E0 1249.          DC.W   -9102,-13623, -9760,-13160
          CC98
F'000D5A  D766 CE87 D505 1250.          DC.W  -10394,-12665,-11003,-12140
          D094
F'000D62  D2BF D2BF D094 1251.          DC.W  -11585,-11585,-12140,-11003
          D505
F'000D6A  CE87 D766 CC98 1252.          DC.W  -12665,-10394,-13160, -9760
          D9E0
F'000D72  CAC9 DC72 C91B 1253.          DC.W  -13623, -9102,-14053, -8423
          DF19
F'000D7A  C78F E1D5 C625 1254.          DC.W  -14449, -7723,-14811, -7005
          E4A3
F'000D82  C4DF E782 C3BE 1255.          DC.W  -15137, -6270,-15426, -5520
          EA70
F'000D8A  C2C1 ED6C C1EB 1256.          DC.W  -15679, -4756,-15893, -3981
          F073
F'000D92  C13B F384 C0B1 1257.          DC.W  -16069, -3196,-16207, -2404
          F69C
F'000D9A  C04F F9BA C014 1258.          DC.W  -16305, -1606,-16364,  -804
          FCDC
                       1259.
                       1260.  ********************
                       1261.  * SLAVE 3 PROGRAM *
                       1262.  ********************
                       1263.  *
                       1264.  * INPUT DATA AT DATAII, TWIDDLE FACTORS AT
                       1265.  * TWD3
                       1266.  *
```

```
F'000DA2  2A7C'00000000  1267.  S3PRO   MOVEA.L #NPE,A5          ;ADDR INC
F'000DA8  2E3C'00000000  1268.          MOVE.L #SHIFT14,D7       ;/16384 SHFT
F'000DAE  303C'FFFF      1269.          MOVE.W #NPTE-1,D0        ;MAJ LOOP CNTR
F'000DB2  287C'FFFFFFFC  1270.          MOVEA.L #NPT-4,A4        ;INDX TO ARRAYS
F'000DB8  43FA 03EC      1271.          LEA TWD3(PC),A1          ;TWID FACT PNTR
F'000DBC  41FA ****      1272.          LEA DATAII(PC),A0        ;XDATA POINTER
F'000DC0  2648           1273.  TWDL3   MOVEA.L A0,A3
F'000DC2  D7CC           1274.          ADDA.L A4,A3
F'000DC4  3213           1275.  ADDL3   MOVE.W (A3),D1           ;GET REAL PART
F'000DC6  48C1           1276.          EXT.L D1                 ;REAL SUM IN D1
F'000DC8  362B 0102      1277.          MOVE.W 258(A3),D3
F'000DCC  48C3           1278.          EXT.L D3
F'000DCE  9283           1279.          SUB.L D3,D1
F'000DD0  362B 0200      1280.          MOVE.W 512(A3),D3
F'000DD4  48C3           1281.          EXT.L D3
F'000DD6  9283           1282.          SUB.L D3,D1
F'000DD8  362B 0302      1283.          MOVE.W 770(A3),D3
F'000DDC  48C3           1284.          EXT.L D3
F'000DDE  D283           1285.          ADD.L D3,D1              ;REAL SUM DONE
F'000DE0  342B 0002      1286.          MOVE.W 2(A3),D2          ;IMAG SUM IN D2
F'000DE4  48C2           1287.          EXT.L D2
F'000DE6  362B 0100      1288.          MOVE.W 256(A3),D3
F'000DEA  48C3           1289.          EXT.L D3
F'000DEC  D483           1290.          ADD.L D3,D2
F'000DEE  362B 0202      1291.          MOVE.W 514(A3),D3
F'000DF2  48C3           1292.          EXT.L D3
F'000DF4  9483           1293.          SUB.L D3,D2
F'000DF6  362B 0300      1294.          MOVE.W 768(A3),D3
F'000DFA  48C3           1295.          EXT.L D3
F'000DFC  9483           1296.          SUB.L D3,D2              ;IMAG SUM DONE
F'000DFE  E482           1297.          ASR.L #2,D2              ;ADJUST SUMS
F'000E00  E481           1298.          ASR.L #2,D1              ;ADJUST SUMS
                         1299.  *
                         1300.  * NOW DO TWIDDLE FACTOR MULTIPLY
                         1301.  *
F'000E02  3A01           1302.          MOVE.W D1,D5
F'000E04  3831 C800      1303.          MOVE.W 0(A1,A4.L),D4
F'000E08  3631 C802      1304.          MOVE.W 2(A1,A4.L),D3
F'000E0C  CBC4           1305.          MULS D4,D5
F'000E0E  3C02           1306.          MOVE.W D2,D6
F'000E10  CDC3           1307.          MULS D3,D6
F'000E12  9A86           1308.          SUB.L D6,D5              ;REAL IN D5
F'000E14  C3C3           1309.          MULS D3,D1
F'000E16  C5C4           1310.          MULS D4,D2
F'000E18  D282           1311.          ADD.L D2,D1              ;IMAG IN D1
F'000E1A  EEA5           1312.          ASR.L D7,D5
F'000E1C  EEA1           1313.          ASR.L D7,D1
                         1314.  *
                         1315.  * STORE SUMMED AND TWIDDLED DATA AT XDATA IN
                         1316.  * PLACE
                         1317.  *
F'000E1E  3185 C800      1318.          MOVE.W D5,0(A0,A4.L)
F'000E22  3181 C802      1319.          MOVE.W D1,2(A0,A4.L)
F'000E26  99CD           1320.          SUBA.L A5,A4
F'000E28  51C8 FF96      1321.          DBRA D0,TWDL3
                         1322.  *
                         1323.  * INSERT FAVORITE 64-POINT FFT HERE
                         1324.  * INPUT DATA AT XDATA (0-255)
                         1325.  *
                         1326.  *
F'000E2C  78'F7          1327.  USHFL3  MOVEQ.L #NPTE-9,D4         ;COUNTER=K
F'000E2E  3C04           1328.  SWAPL3  MOVE.W D4,D6          ;PREPARE BIT REV
F'000E30  3E06           1329.  BREV23  MOVE.W D6,D7        ;SAVE DATA
F'000E32  4246           1330.          CLR.W D6            ;
F'000E34  7A05           1331.          MOVEQ.L #5,D5       ;BIT COUNTER
F'000E36  E257           1332.  REV23   ROXR.W #1,D7        ;SFT RIGHT,XTEND
F'000E38  E356           1333.          ROXL.W #1,D6
F'000E3A  51CD FFFA      1334.          DBRA D5,REV23
F'000E3E  B846           1335.          CMP.W D6,D4           ;I=REV(K)<=K?
F'000E40  6C 14          1336.          BGE.S DECK3            ;NO SWAP IF SO
F'000E42  3204           1337.          MOVE.W D4,D1          ;COPY OF K
F'000E44  E541           1338.          ASL.W #2,D1           ;K*4
F'000E46  E546           1339.          ASL.W #2,D6           ;I*4
F'000E48  2430 1000      1340.          MOVE.L 0(A0,D1.W),D2 ;DO SWAP
F'000E4C  21B0 6000 1000 1341.          MOVE.L 0(A0,D6.W),0(A0,D1.W)
F'000E52  2182 6000      1342.          MOVE.L D2,0(A0,D6.W) ;DONE SWAP
F'000E56  51CC FFD6      1343.  DECK3   DBRA D4,SWAPL3
F'000E5A  7201           1344.  FFT643  MOVEQ.L #1,D1    ;D1=N2=1
F'000E5C  43FA 0248      1345.          LEA SINBLK3(PC),A1  ;SIN VALS PNTR A1
F'000E60  45FA 02C4      1346.          LEA COSBLK3(PC),A2  ;COS VALS PNTR A2
                         1347.  *
```

```
                                1348.  * L IS DOWN COUNTED #NUGAM-1 TO 0
                                1349.  *
F'000E64  3F3C 0001             1350.          MOVE.W #1,-(SP)  ;(SP) IS L
F'000E68  7405                  1351.          MOVEQ.L #NUGAM-1,D2        ;D2=NUGAM-1=NU1
F'000E6A  4283                  1352.          CLR.L D3              ;D3=K=0
F'000E6C  70'00                 1353.          MOVEQ.L #SHIFT14,D0
                                1354.  *
                                1355.  * INITIALIZE I=N2-1, TO 0 IN STEPS -1
                                1356.  *
F'000E6E  3801                  1357.  INITI3  MOVE.W D1,D4     ;D4=I=N2
F'000E70  5344                  1358.          SUBQ.W #1,D4     ;SET FOR DBRA
                                1359.  *
                                1360.  * DETERMINE P=MOD(K*2**(NUGAM-L),64)
                                1361.  *
                                1362.  *
F'000E72  3C03                  1363.  LOOPIN3 MOVE.W D3,D6     ;D6=K
F'000E74  3A17                  1364.          MOVE.W (SP),D5   ;GET L
F'000E76  5D45                  1365.          SUBQ.W #NUGAM,D5 ;L-NUGAM
F'000E78  4445                  1366.          NEG.W D5         ;NUGAM-L
F'000E7A  EB66                  1367.          ASL.W D5,D6      ;K*2**(NUGAM-L)
F'000E7C  0246 003F             1368.          ANDI W #$003F,D6 ;MOD 64
                                1369.  *
                                1370.  * D6 HAS INDEX REQUIRED
F'000E80  DC46                  1371.          ADD.W D6,D6      ;D6*2 FOR WORD BOUND
F'000E82  3A71 6000             1372.          MOVEA.W 0(A1,D6.W),A5 ;A5 PNTR IM(W**P)
F'000E86  2E03                  1373.          MOVE.L D3,D7     ;D7=K
F'000E88  2A03                  1374.          MOVE.L D3,D5     ;D5 TOO
F'000E8A  E545                  1375.          ASL.W #2,D5      ;K*4 FOR LWORD BOUND
F'000E8C  4DF0 5000             1376.          LEA 0(A0,D5.W),A6    ;A6 IS INDEX X(K)
F'000E90  DE41                  1377.          ADD.W D1,D7      ;D7=K+N2
F'000E92  E547                  1378.          ASL.W #2,D7      ;D7=4(K+N2) LWORD BND
F'000E94  47F0 7000             1379.          LEA 0(A0,D7.W),A3 ;A3 PNT RE(X(K+N2))
F'000E98  3A32 6000             1380.          MOVE.W 0(A2,D6.W),D5     ;GET RE(W**P)
F'000E9C  3E05                  1381.          MOVE.W D5,D7     ;PUT ALSO IN D7
F'000E9E  CBD3                  1382.          MULS (A3),D5     ;RE(W**P)*RE(X(K+N2))
F'000EA0  3C0D                  1383.          MOVE.W A5,D6  ;GET IM(W**P)
F'000EA2  CDEB 0002             1384.          MULS 2(A3),D6    ;IM(W**P)*IM(X(K+N2))
F'000EA6  9A86                  1385.          SUB.L D6,D5      ;D5=RE(T1)
F'000EA8  E0A5                  1386.          ASR.L D0,D5      ;/16384 TO SCALE
F'000EAA  3C0D                  1387.          MOVE.W A5,D6  ;GET IM(W**P)
F'000EAC  CDD3                  1388.          MULS (A3),D6     ;IM(W**P)*RE(X(K+N2))
F'000EAE  CFEB 0002             1389.          MULS 2(A3),D7    ;R   **P)*IM(X(K+N2))
F'000EB2  DC87                  1390.          ADD.L D7,D6      ;D   M(T1)
F'000EB4  E0A6                  1391.          ASR.L D0,D6      ;/16384 TO SCALE
F'000EB6  3E16                  1392.          MOVE.W (A6),D7   ;RE(X(K))
F'000EB8  9E45                  1393.          SUB.W D5,D7      ;RE(X(K))-RE(T1)
F'000EBA  E247                  1394.          ASR.W #1,D7      ;/2
F'000EBC  3687                  1395.          MOVE.W D7,(A3)   ;STORE ANSWER
F'000EBE  3E2E 0002             1396.          MOVE.W 2(A6),D7  ;IM(X(K))
F'000EC2  9E46                  1397.          SUB.W D6,D7      ;IM(X(K))-IM(T1)
F'000EC4  E247                  1398.          ASR.W #1,D7      ;/2
F'000EC6  3747 0002             1399.          MOVE.W D7,2(A3)  ;STORE ANSWER
F'000ECA  DB56                  1400.          ADD.W D5,(A6)    ;RE(X(K))+RE(T1)
F'000ECC  E0D6                  1401.          ASR.W (A6)       ;/2
F'000ECE  DD6E 0002             1402.          ADD.W D6,2(A6)   ;IM(X(K))+IM(T1)
F'000ED2  E0EE 0002             1403.          ASR.W 2(A6)      ;/2
F'000ED6  5243                  1404.          ADDQ.W #1,D3     ;INCREMENT K
F'000ED8  51CC FF98             1405.          DBRA D4,LOOPIN3  ;DO TILL I=-1
F'000EDC  D641                  1406.          ADD.W D1,D3      ;K=K+N2
F'000EDE  0C43'FFFF             1407.          CMPI.W #NPTE-1,D3 ;K<N-1 ?
F'000EE2  6D 8A                 1408.          BLT INITI3       ;DO AGAIN IF SO
F'000EE4  4243                  1409.          CLR.W D3         ;K=0
F'000EE6  5342                  1410.          SUBQ.W #1,D2     ;NU1=NU1-1
F'000EE8  E341                  1411.          ASL.W #1,D1      ;N2=N2*2
F'000EEA  5257                  1412.          ADDQ.W #1,(SP)
F'000EEC  0C57 0007             1413.          CMPI.W #NUGAM+1,(SP)
F'000EF0  6600 FF7C             1414.          BNE INITI3
F'000EF4  301F                  1415.          MOVE.W (SP)+,D0
                                1416.  *
                                1417.  * NOW DO FILTERING, FILTER DATA IS AT "FILTER"
                                1418.  * 64 POINTS
                                1419.  *
                                1420.  * DO COMPLEX MULTIPLY BY THE FILTER.
                                1421.  * FILTER CAN BE +16384 TO -16384 WITH 16384
                                1422.  * BEING THE 0 db GAIN VALUE.
                                1423   * (A+iB)*(C+iD)
                                1424.  *
F'000EF6  43FA ****             1425.          LEA FILTER(PC),A1        ;A1 PNT FILTER
F'000EFA  303C'FFFF             1426.          MOVE.W #NPTE-1,D0        ;LOOP COUNTER
F'000EFE  7A'00                 1427.          MOVEQ.L #SHIFT14,D5              ,SHIFT COUNT
F'000F00  3219                  1428.  FILOOP3 MOVE.W (A1)+,D1          ;GET C
```

```
F'000F02  3601           1429.          MOVE.W D1,D3
F'000F04  C3D8           1430.          MULS (A0)+,D1           ;DO AC
F'000F06  3419           1431.          MOVE.W (A1)+,D2         ;GET D
F'000F08  3802           1432.          MOVE.W D2,D4
F'000F0A  C5D0           1433.          MULS (A0),D2            ;DO BD
F'000F0C  9282           1434.          SUB.L D2,D1             ;AC-BD
F'000F0E  EAA1           1435.          ASR.L D5,D1             ;ADJUST /16384
F'000F10  C7D0           1436.          MULS (A0),D3            ;DO BC
F'000F12  C9E0           1437.          MULS -(A0),D4           ;DO AD
F'000F14  D883           1438.          ADD.L D3,D4             ;BC+AD
F'000F16  EAA4           1439.          ASR.L D5,D4             ;ADJUST /16384
F'000F18  30C1           1440.          MOVE.W D1,(A0)+ ;STORE REAL
F'000F1A  30C4           1441.          MOVE.W D4,(A0)+ ;STORE IMAG
F'000F1C  51C8 FFE2      1442.          DBRA D0,FILOOP3
                         1443.  *
                         1444.  * FILTERING DONE
                         1445.  *
                         1446.  *
                         1447.  * ASSERT SRQ AND WAIT FOR ACKNOWLEDGE
                         1448.  *
F'000F20  4E43           1449.          TRAP #SRQASRT
                         1450.  * SEND OUT 64 COMPLEX POINTS CALCULATED
>>> WARNING line[1451] Expr. type   [1451] SLVNOMES.A68
F'000F22  303C ****      1451.          MOVE.W #NPTE*4,D0       ;BYTE COUNT
F'000F26  41FA ****      1452.          LEA DATAII(PC),A0
F'000F2A  13FC 00'00     1453.          MOVE.B #ENABLEB,PGCR
          '00000000
F'000F32  4E46           1454.          TRAP #OUTDATA
F'000F34  4239'00000000  1455.          CLR.B PGCR
                         1456.  *
                         1457.  * ASSERT AN SRQ TO SIGNAL READY FOR NEW DATA
                         1458.  * AND RECEIVE NEW FILTERED DATA TO BE INVERSE
                         1459.  * TRANSFORMED
                         1460.  *
F'000F3A  4E43           1461.          TRAP #SRQASRT
>>> WARNING line[1462] Expr. type   [1462] SLVNOMES.A68
F'000F3C  303C ****      1462.          MOVE.W #NPT*4,D0        ;BYTE COUNT
F'000F40  41FA ****      1463.          LEA DATAII(PC),A0
F'000F44  13FC 00'00     1464           MOVE.B #ENABLEA,PGCR
          '00000000
F'000F4C  4E45           1465.          TRAP #INDATA
F'000F4E  4239'00000000  1466.          CLR.B PGCR
                         1467.  *
                         1468.  * NOW DO THE INVERSE TRANSFORM
                         1469.  * USE SAME SIN AND COS TABLES, NEGATE THE SIN
                         1470.  * ENTRY, AND USE NO ASR #2 FOR DIVISION BY NPTE
                         1471.  * NOTE SIGN CHANGE ON 4-ADDITIONS TOO
                         1472.  *
F'000F54  2A7C'00000000  1473.  S3PROR  MOVEA.L #NPE,A5          ;ADDR INC
F'000F5A  2E3C'00000000  1474.          MOVE.L #SHIFT14,D7      ;/16384 SHFT
F'000F60  303C'FFFF      1475.          MOVE.W #NPTE-1,D0       ;MAJ LOOP CNTR
F'000F64  287C'FFFFFFFC  1476.          MOVEA.L #NPT-4,A4       ;INDX TO ARRAYS
F'000F6A  43FA 023A      1477.          LEA TWD3(PC),A1         ;TWID FACT PNTR
F'000F6E  41FA ****      1478.          LEA DATAII(PC),A0       ;XDATA POINTER
F'000F72  2648           1479.  TWDL3R  MOVEA.L A0,A3
F'000F74  D7CC           1480.          ADDA.L A4,A3
F'000F76  3213           1481.  ADDL3R  MOVE.W (A3),D1          ;GET REAL PART
F'000F78  362B 0102      1482.          MOVE.W 258(A3),D3
F'000F7C  D243           1483.          ADD.W D3,D1
F'000F7E  362B 0200      1484.          MOVE.W 512(A3),D3
F'000F82  9243           1485.          SUB.W D3,D1
F'000F84  362B 0302      1486.          MOVE.W 770(A3),D3
F'000F88  9243           1487.          SUB.W D3,D1             ;REAL SUM DONE
F'000F8A  342B 0002      1488.          MOVE.W 2(A3),D2         ;IMAG SUM IN D2
F'000F8E  362B 0100      1489.          MOVE.W 256(A3),D3
F'000F92  9443           1490.          SUB.W D3,D2
F'000F94  362B 0202      1491.          MOVE.W 514(A3),D3
F'000F98  9443           1492.          SUB.W D3,D2
F'000F9A  362B 0300      1493.          MOVE.W 768(A3),D3
F'000F9E  D443           1494.          ADD.W D3,D2             ;IMAG SUM DONE
                         1495.  *
                         1496.  * NOW DO TWIDDLE FACTOR MULTIPLY
                         1497.  *
F'000FA0  3A01           1498.          MOVE.W D1,D5
```

```
F'000FA2  3831 C800      1499.          MOVE.W 0(A1,A4.L),D4
F'000FA6  3631 C802      1500.          MOVE.W 2(A1,A4.L),D3
F'000FAA  4443           1501.          NEG.W D3
F'000FAC  CBC4           1502.          MULS D4,D5
F'000FAE  3C02           1503.          MOVE.W D2,D6
F'000FB0  CDC3           1504.          MULS D3,D6
F'000FB2  9A86           1505.          SUB.L D6,D5              ;REAL IN D5
F'000FB4  C3C3           1506.          MULS D3,D1
F'000FB6  C5C4           1507.          MULS D4,D2
F'000FB8  D282           1508           ADD.L D2,D1              ;IMAG IN D1
F'000FBA  EEA5           1509.          ASR.L D7,D5
F'000FBC  EEA1           1510.          ASR.L D7,D1
                         1511.  *
                         1512.  * STORE SUMMED AND TWIDDLED DATA AT XDATA IN
                         1513.  * PLACE
                         1514.  *
F'000FBE  3185 C800      1515.          MOVE.W D5,0(A0,A4.L)
F'000FC2  3181 C802      1516.          MOVE.W D1,2(A0,A4.L)
F'000FC6  99CD           1517.          SUBA.L A5,A4
F'000FC8  51C8 FFA8      1518.          DBRA D0,TWDL3R
                         1519.  *
                         1520.  * INSERT FAVORITE 64-POINT FFT HERE
                         1521.  * INPUT DATA AT XDATA (0-255)
                         1522.  *
                         1523.  *
F'000FCC  78'F7          1524.  USHFL3R MOVEQ.L #NPTE-9,D4        ,COUNTER=K
F'000FCE  3C04           1525.  SWAPL3R MOVE.W D4,D6          ;PREPARE BIT REV
F'000FD0  3E06           1526.  BREV23R MOVE.W D6,D7        ;SAVE DATA
F'000FD2  4246           1527.          CLR.W D6            ;
F'000FD4  7A05           1528.          MOVEQ.L #5,D5       ;BIT COUNTER
F'000FD6  E257           1529.  REV23R  ROXR.W #1,D7        ;SFT RIGHT,XTEND
F'000FD8  E356           1530.          ROXL.W #1,D6
F'000FDA  51CD FFFA      1531.          DBRA D5,REV23R
F'000FDE  B846           1532.          CMP.W D6,D4           ;I=REV(K)<=K?
F'000FE0  6C 14          1533.          BGE.S DECK3R            ;NO SWAP IF SO
F'000FE2  3204           1534.          MOVE.W D4,D1          ;COPY OF K
F'000FE4  E541           1535.          ASL.W #2,D1           ;K*4
F'000FE6  E546           1536.          ASL.W #2,D6           ;I*4
F'000FE8  2430 1000      1537.          MOVE.L 0(A0,D1.W),D2 ;DO SWAP
F'000FEC  21B0 6000 100G 1538.          MOVE.L 0(A0,D6.W),0(A0,D1.W)
F'000FF2  2182 6000      1539.          MOVE.L D2,0(A0,D6.W) ;DONE SWAP
F'000FF6  51CC FFD6      1540.  DECK3R  DBRA D4,SWAPL3R
F'000FFA  7201           1541.  FFT643R MOVEQ.L #1,D1   ;D1=N2=1
F'000FFC  43FA 00A8      1542.          LEA SINBLK3(PC),A1  ;SIN VALS PNTR A1
F'001000  45FA 0124      1543.          LEA COSBLK3(PC),A2  ;COS VALS PNTR A2
                         1544.  *
                         1545.  * L IS DOWN COUNTED #NUGAM-1 TO 0
                         1546.  *
F'001004  3F3C 0001      1547.          MOVE.W #1,-(SP)  ;(SP) IS L
F'001008  7405           1548.          MOVEQ.L #NUGAM-1,D2        ;D2=NUGAM-1=NU1
F'00100A  4283           1549.          CLR.L D3                ;D3=K=0
F'00100C  70'00          1550.          MOVEQ.L #SHIFT14,D0
                         1551.  *
                         1552.  * INITIALIZE I=N2-1, TO 0 IN STEPS -1
                         1553.  *
F'00100E  3801           1554.  INITI3R MOVE.W D1,D4     ;D4=I=N2
F'001010  5344           1555.          SUBQ.W #1,D4     ;SET FOR DBRA
                         1556.  *
                         1557.  * DETERMINE P=MOD(K*2**(NUGAM-L),64)
                         1558.  *
                         1559.  *
F'001012  3C03           1560.  LPIN3R  MOVE.W D3,D6     ;D6=K
F'001014  3A17           1561.          MOVE.W (SP),D5  ;GET L
F'001016  5D45           1562.          SUBQ.W #NUGAM,D5 ;L-NUGAM
F'001018  4445           1563.          NEG.W D5         ;NUGAM-L
F'00101A  EB66           1564.          ASL.W D5,D6      ;K*2**(NUGAM-L)
F'00101C  0246 003F      1565.          ANDI.W #$003F,D6 ;MOD 64
                         1566.  *
                         1567.  * D6 HAS INDEX REQUIRED
F'001020  DC46           1568.          ADD.W D6,D6      ;D6*2 FOR WORD BOUND
F'001022  3A71 6000      1569.          MOVEA.W 0(A1,D6.W),A5 ;A5 PNTR IM(W**P)
F'001026  2E03           1570.          MOVE.L D3,D7     ;D7=K
F'001028  2A03           1571.          MOVE.L D3,D5     ;D5 TOO
F'00102A  E545           1572.          ASL.W #2,D5      ;K*4 FOR LWORD BOUND
F'00102C  4DF0 5000      1573.          LEA 0(A0,D5.W),A6   ;A6 IS INDEX X(K)
F'001030  DE41           1574.          ADD.W D1,D7      ;D7=K+N2
F'001032  E547           1575.          ASL.W #2,D7      ;D7=4(K+N2) LWORD BND
F'001034  47F0 7000      1576.          LEA 0(A0,D7.W),A3 ;A3 PNT RE(X(K+N2))
F'001038  3A32 6000      1577.          MOVE.W 0(A2,D6.W),D5     ;GET RE(W**P)
F'00103C  3E05           1578.          MOVE.W D5,D7     ;PUT ALSO IN D7
F'00103E  CBD3           1579.          MULS (A3),D5     ;RE(W**P)*RE(X(K+N2))
```

```
F'001040  3C0D           1580.          MOVE.W A5,D6  ;GET IM(W**P)
F'001042  4446           1581.          NEG.W D6
F'001044  CDEB 0002      1582.          MULS 2(A3),D6    ;IM(W**P)*IM(X(K+N2))
F'001048  9A86           1583.          SUB.L D6,D5      ;D5=RE(T1)
F'00104A  E0A5           1584.          ASR.L D0,D5      ;/16384 TO SCALE
F'00104C  3C0D           1585.          MOVE.W A5,D6  ;GET IM(W**P)
F'00104E  4446           1586.          NEG.W D6
F'001050  CDD3           1587.          MULS (A3),D6     ;IM(W**P)*RE(X(K+N2))
F'001052  CFEB 0002      1588.          MULS 2(A3),D7    ;RE(W**P)*IM(X(K+N2))
F'001056  DC87           1589.          ADD.L D7,D6      ;D6=IM(T1)
F'001058  E0A6           1590.          ASR.L D0,D6      ;/16384 TO SCALE
F'00105A  3E16           1591.          MOVE.W (A6),D7   ;RE(X(K))
F'00105C  9E45           1592.          SUB.W D5,D7      ;RE(X(K))-RE(T1)
F'00105E  3687           1593.          MOVE.W D7,(A3)   ;STORE ANSWER
F'001060  3E2E 0002      1594.          MOVE.W 2(A6),D7  ;IM(X(K))
F'001064  9E46           1595.          SUB.W D6,D7      ;IM(X(K))-IM(T1)
F'001066  3747 0002      1596.          MOVE.W D7,2(A3)  ;STORE ANSWER
F'00106A  DB56           1597.          ADD.W D5,(A6)    ;RE(X(K))+RE(T1)
F'00106C  DD6E 0002      1598.          ADD.W D6,2(A6)   ;IM(X(K))+IM(T1)
F'001070  5243           1599.          ADDQ.W #1,D3     ;INCREMENT K
F'001072  51CC FF9E      1600.          DBRA D4,LPIN3R   ;DO TILL I=-1
F'001076  D641           1601.          ADD.W D1,D3      ;K=K+N2
F'001078  0C43'FFFF      1602.          CMPI.W #NPTE-1,D3 ;K<N-1 ?
F'00107C  6D 90          1603.          BLT INITI3R      ;DO AGAIN IF SO
F'00107E  4243           1604.          CLR.W D3         ;K=0
F'001080  5342           1605.          SUBQ.W #1,D2     ;NU1=NU1-1
F'001082  E341           1606.          ASL.W #1,D1      ;N2=N2*2
F'001084  5257           1607.          ADDQ.W #1,(SP)
F'001086  0C57 0007      1608.          CMPI.W #NUGAM+1,(SP)
F'00108A  66 82          1609.          BNE INITI3R
F'00108C  301F           1610.          MOVE.W (SP)+,D0
                         1611.  *
                         1612.  * ASSERT SRQ AND WAIT FOR ACKNOWLEDGE
                         1613.  *
F'00108E  4E43           1614.          TRAP #SRQASRT
                         1615.  * SEND OUT 64 COMPLEX POINTS CALCULATED
>>> WARNING line[1616] Expr. type   [1616] SLVNOMES.A68
F'001090  303C ****      1616.          MOVE.W #NPTE*4,D0
F'001094  13FC 00'00     1617.          MOVE.B #ENABLEB,PGCR
          '00000000
F'00109C  4E46           1618.          TRAP #OUTDATA
F'00109E  4239'00000000  1619.          CLR.B PGCR
                         1620.  *
                         1621.  * NOW DO RTS AND RETURN TO READY FOR DATA
                         1622.  * STATE TO DO OVER!
                         1623.  *
F'0010A4  4E75           1624.          RTS
                         1625.  *
                         1626.  * DONE!
                         1627.  *
                         1628.  * SIN (-2*PI*I/64)*16384 FOR I=0 TO 63
                         1629.  *
F'0010A6  0000 F9BA F384 1630.  SINBLK3 DC.W       0, -1606, -3196, -4756
          ED6C
F'0010AE  E782 E1D5 DC72 1631.          DC.W   -6270, -7723, -9102,-10394
          D766
F'0010B6  D2BF CE87 CAC9 1632.          DC.W  -11585,-12665,-13623,-14449
          C78F
F'0010BE  C4DF C2C1 C13B 1633.          DC.W  -15137,-15679,-16069,-16305
          C04F
F'0010C6  C000 C04F C13B 1634.          DC.W  -16384,-16305,-16069,-15679
          C2C1
F'0010CE  C4DF C78F CAC9 1635.          DC.W  -15137,-14449,-13623,-12665
          CE87
F'0010D6  D2BF D766 DC72 1636.          DC.W  -11585,-10394, -9102, -7723
          E1D5
F'0010DE  E782 ED6C F384 1637.          DC.W   -6270, -4756, -3196, -1606
          F9BA
F'0010E6  0000 0646 0C7C 1638.          DC.W       0,  1606,  3196,  4756
          1294
F'0010EE  187E 1E2B 238E 1639.          DC.W    6270,  7723,  9102, 10394
          289A
F'0010F6  2D41 3179 3537 1640.          DC.W   11585, 12665, 13623, 14449
          3871
F'0010FE  3B21 3D3F 3EC5 1641.          DC.W   15137, 15679, 16069, 16305
          3FB1
F'001106  4000 3FB1 3EC5 1642.          DC.W   16384, 16305, 16069, 15679
          3D3F
F'00110E  3B21 3871 3537 1643.          DC.W   15137, 14449, 13623, 12665
          3179
F'001116  2D41 289A 238E 1644.          DC.W   11585, 10394,  9102,  7723
```

```
                1E2B
F'00111E  187E 1294 0C7C 1645.          DC.W    6270,  4756,  3196,  1606
                0646
                                1646.   *
                                1647.   * COS(-2*PI*I/64)*16384 FOR I=0 TO 63
                                1648.   *
F'001126  4000 3FB1 3EC5 1649.  COSBLK3 DC.W    16384, 16305, 16069, 15679
                3D3F
F'00112E  3B21 3871 3537 1650.          DC.W    15137, 14449, 13623, 12665
                3179
F'001136  2D41 289A 238E 1651.          DC.W    11585, 10394,  9102,  7723
                1E2B
F'00113E  187E 1294 0C7C 1652.          DC.W     6270,  4756,  3196,  1606
                0646
F'001146  0000 F9BA F384 1653.          DC.W        0, -1606, -3196, -4756
                ED6C
F'00114E  E782 E1D5 DC72 1654.          DC.W    -6270, -7723, -9102,-10394
                D766
F'001156  D2BF CE87 CAC9 1655.          DC.W   -11585,-12665,-13623,-14449
                C78F
F'00115E  C4DF C2C1 C13B 1656.          DC.W   -15137,-15679,-16069,-16305
                C04F
F'001166  C000 C04F C13B 1657.          DC.W   -16384,-16305,-16069,-15679
                C2C1
F'00116E  C4DF C78F CAC9 1658.          DC.W   -15137,-14449,-13623,-12665
                CE87
F'001176  D2BF D766 DC72 1659.          DC.W   -11585,-10394, -9102, -7723
                E1D5
F'00117E  E782 ED6C F384 1660.          DC.W    -6270, -4756, -3196, -1606
                F9BA
F'001186  0000 0646 0C7C 1661.          DC.W        0,  1606,  3196,  4756
                1294
F'00118E  187E 1E2B 238E 1662.          DC.W     6270,  7723,  9102, 10394
                289A
F'001196  2D41 3179 3537 1663.          DC.W    11585, 12665, 13623, 14449
                3871
F'00119E  3B21 3D3F 3EC5 1664.          DC.W    15137, 15679, 16069, 16305
                3FB1
                                1665.   *
                                1666.   * TWIDDLE FACTOR TABLE 3 TWD3
                                1667.   *
F'0011A6  4000 0000 3FD4 1668.  TWD3    DC.W    16384,     0, 16340, -1205
                FB4B
F'0011AE  3F4F F69C 3E72 1669.          DC.W    16207, -2404, 15986, -3590
                F1FA
F'0011B6  3D3F ED6C 3BB6 1670.          DC.W    15679, -4756, 15286, -5897
                E8F7
F'0011BE  39DB E4A3 37B0 1671.          DC.W    14811, -7005, 14256, -8076
                E074
F'0011C6  3537 DC72 3274 1672.          DC.W    13623, -9102, 12916,-10080
                D8A0
F'0011CE  2F6C D505 2C21 1673.          DC.W    12140,-11003, 11297,-11866
                D1A6
F'0011D6  289A CE87 24DA 1674.          DC.W    10394,-12665,  9434,-13395
                CBAD
F'0011DE  20E7 C91B 1CC6 1675.          DC.W     8423,-14053,  7366,-14635
                C6D5
F'0011E6  187E C4DF 1413 1676.          DC.W     6270,-15137,  5139,-15557
                C33B
F'0011EE  0F8D C1EB 0AF1 1677.          DC.W     3981,-15893,  2801,-16143
                C0F1
F'0011F6  0646 C04F 0192 1678.          DC W     1606,-16305,   402,-16379
                C005
F'0011FE  FCDC C014 F82A 1679.          DC.W     -804,-16364, -2006,-16261
                C07B
F'001206  F384 C13B EEEE 1680.          DC.W    -3196,-16069, -4370,-15791
                C251
F'00120E  EA70 C3BE E611 1681.          DC.W    -5520,-15426, -6639,-14978
                C57E
F'001216  E1D5 C78F DDC3 1682.          DC.W    -7723,-14449, -8765,-13842
                C9EE
F'00121E  D9E0 CC98 D632 1683.          DC.W    -9760,-13160,-10702,-12406
                CF8A
F'001226  D2BF D2BF CF8A 1684.          DC.W   -11585,-11585,-12406,-10702
                D632
F'00122E  CC98 D9E0 C9EE 1685.          DC.W   -13160, -9760,-13842, -8765
                DDC3
F'001236  C78F E1D5 C57E 1686.          DC.W   -14449, -7723,-14978, -6639
                E611
F'00123E  C3BE EA70 C251 1687.          DC.W   -15426, -5520,-15791, -4370
                EEEE
```

```
F'001246  C13B F384 C07B 1688.          DC.W  -16069, -3196,-16261, -2006
          F82A
F'00124E  C014 FCDC C005 1689.          DC.W  -16364,  -804,-16379,   402
          0192
F'001256  C04F 0646 C0F1 1690.          DC.W  -16305,  1606,-16143,  2801
          0AF1
F'00125E  C1EB 0F8D C33B 1691.          DC.W  -15893,  3981,-15557,  5139
          1413
F'001266  C4DF 187E C6D5 1692.          DC.W  -15137,  6270,-14635,  7366
          1CC6
F'00126E  C91B 20E7 CBAD 1693.          DC.W  -14053,  8423,-13395,  9434
          24DA
F'001276  CE87 289A D1A6 1694.          DC.W  -12665, 10394,-11866, 11297
          2C21
F'00127E  D505 2F6C D8A0 1695.          DC.W  -11003, 12140,-10080, 12916
          3274
F'001286  DC72 3537 E074 1696.          DC.W   -9102, 13623, -8076, 14256
          37B0
F'00128E  E4A3 39DB E8F7 1697.          DC.W   -7005, 14811, -5897, 15286
          3BB6
F'001296  ED6C 3D3F F1FA 1698.          DC.W   -4756, 15679, -3590, 15986
          3E72
F'00129E  F69C 3F4F FB4B 1699.          DC.W   -2404, 16207, -1205, 16340
          3FD4
F'0012A6  <1>            1700.  DONE    DS.B 1
F'0012A7                 1701.          END
   12 Warnings
    0 Errors
```

M.8 Program FSTFL1NM: Multicomputer Frequency Domain Filtering Program #1B

```
                                1.          TTL FSTFLNM1
                                2.  *
                                3.  * This program performs a 256 complex point
                                4.  * frequency domain filtering operation on an
                                5.  * input signal at DATAII.  The filter
                                6.  * coefficients are at FILTER, and are
                                7.  * distributed among slaves using another
                                8.  * program (filter.a68).
                                9.  *
                               10.  * input/output data is stored as real/imaginary
                               11.  * pair, each component 16-bits long.
                               12.  * twiddle factors wit..in slave programs are
                               13.  * stored similarly
                               14.  *
                               15.  *
                               16.  * self timed operation, 256 runs
                               17.  *
                               18.  * EQUATES
0'000000                       19.          SECTION 0
                               20.          XDEF PGCR,ENABLEA,ENABLEB
                               21.          XDEF NPT,NPTE,NPE,SHIFT14
                               22.          XDEF TCR,CNTSTRT,ENABLTM
                               23.          XREF S0PRO,S1PRO,S2PRO,S3PRO,DONE
                               24.          XDEF DATAII,FILTER,MBRDCST
                               25.          XDEF XDATA0,XDATA1,XDATA2,XDATA3
                               26.  *
     # 00012FC0                27.  PGCR     EQU $12FC0
     # 00000010                28.  ENABLEA  EQU $10
     # 00000020                29.  ENABLEB  EQU $20
     # 00000100                30.  NPT      EQU $100
     # 00000040                31.  NPTE     EQU $40
     # 00000004                32.  NPE      EQU 4
     # 0000000E                33.  SHIFT14  EQU 14
     # 00000003                34.  SRQASRT  EQU 3
     # 00000004                35.  ADBYTIN  EQU 4
     # 00000007                36.  ADBYTOT  EQU 7
     # 00000005                37.  INDATA   EQU 5
     # 00000006                38.  OUTDATA  EQU 6
     # 00000008                39.  NETCNF   EQU 8
     # 00000009                40.  SRQSRVC  EQU 9
     # 0000000E                41.  ABORT    EQU 14
     # 00000001                42.  SLV0     EQU $01
     # 00000002                43.  SLV1     EQU $02
     # 00000004                44.  SLV2     EQU $04
     # 00000008                45.  SLV3     EQU $08
     # 0000000F                46.  ALLSLV   EQU $0F
     # 00000006                47.  NUGAM    EQU 6
     # 00012FB0                48.  PSR01    EQU $12FB0
     # 00012FB1                49.  PSR23    EQU $12FB1
     # 00012FB2                50.  PSR45    EQU $12FB2
     # 00012FB3                51.  PSR67    EQU $12FB3
     # 00012FB4                52.  PSR89    EQU $12FB4
     # 00012FB5                53.  PSR1011  EQU $12FB5
     # 00012FB6                54.  PSR1213  EQU $12FB6
     # 00012FB7                55.  PSR1415  EQU $12FB7
     # 00012FB8                56.  PSR1617  EQU $12FB8
     # 00012FB9                57.  PSR1819  EQU $12FB9
     # 00012FD0                58.  TCR      EQU $12FD0
     # 00012FD2                59.  TPLR     EQU TCR+2
     # 00012FD6                60.  TCNTR    EQU TCR+6
     # 0000000A                61.  SENCLFC  EQU 10
     # 00000018                62.  DISBUF8  EQU 24
     # 0000001A                63.  DISBUF2  EQU 26
     # 00FFFFFF                64.  CNTSTRT  EQU $00FFFFFF
     # 00000001                65.  ENABLTM  EQU 1
     # 00000011                66.  OUTMESC  EQU 17
     # 00000008                67.  DIV256   EQU 8
     # 00000100                68.  PASSCNT  EQU 256
     # 00000004                69.  EOT      EQU 4
                               70.  ****************************************
                               71.  * THIS VERSION DOES INDIVIDUAL SYNCS FOR
                               72.  * EACH UPLOAD OF SLAVE DATA
                               73.  * COLLECTS IN ORDER 0,1,2,3
                               74.  ****************************************
                               75.  ******************
                               76.  * MASTER PROGRAM *
```

```
                                77.  ******************
                                78.  *
0'000000  <400>                 79.  DATAIN  DS.B $400
0'000400  <400>                 80.  DATAII  DS.B $400
0'000800  <400>                 81.  FILTER  DS.B $400
0'000C00  <100>                 82.  XDATA0  DS.B $100
0'000D00  <100>                 83.  XDATA1  DS.B $100
0'000E00  <100>                 84.  XDATA2  DS.B $100
0'000F00  <100>                 85.  XDATA3  DS.B $100
0'001000  BB BB BB CB BB        86.  INITCNF DC.B $BB,$BB,$BB,$CB,$BB,$8C,$CB,$8B,$B8,$8B
          8C CB 8B B8 8B
0'00100A  BB DB 00 CE E0        87.  MBRDCST DC.B $BB,$DB,$00,$CE,$E0,$7C,$C8,$07,$88,$4B
          7C C8 07 86 4B
                                88.  *
                                89.  * CONFIGURE NETWORK TO INITIAL STATE, MASTER
                                90.  * TO SLAVE 0
                                91.  * RESPOND TO SLAVE 0'S RFP SRQ, AND LOAD
                                92.  * ITS PROGRAM
                                93.  *
0'001014  287C 00012FD0         94.  STRTFFT MOVEA.L #TCR,A4          ;POINTER TIMER
0'00101A  4214                  95.          CLR.B (A4)               ;DISABLE F'SURE
0'00101C  2F3C 00FFFFFF         96.          MOVE.L #CNTSTRT,-(SP)    ;MIN TIME
0'001022  2F3C 00000000         97.          MOVE.L #$00,-(SP)        ;MAX TIME
0'001028  2F3C 00000000         98.          MOVE.L #$00,-(SP)        ;RUN'G SUM TIME
0'00102E  2E3C 000000FF         99.          MOVE.L #PASSCNT-1,D7     ;COUNTER
0'001034  41FA FFCA            100.          LEA INITCNF(PC),A0
0'001038  4E48                 101.          TRAP #NETCNF
0'00103A  103C 0001            102.          MOVE.B #SLV0,D0
0'00103E  4E49                 103.          TRAP #SRQSRVC
0'001040  41FA ****            104.          LEA S0PRO(PC),A0         ;STRT ADDR
0'001044  43FA ****            105.          LEA S1PRO(PC),A1         ;END ADDR+1
0'001048  6100 021E            106.          BSR SENDOUT              ;ADDR,COUNT,CODE
                               107.  *
                               108.  * CONFIGURE NETWORK TO MASTER TO SLAVE 1
                               109.  * RESPOND TO SLAVE 1'S RFP SRQ, AND LOAD ITS
                               110.  * PROGRAM
                               111.  *
0'00104C  13FC 00B4            112.          MOVE.B #$B4,PSR01
          00012FB0
0'001054  103C 0002            113.          MOVE.B #SLV1,D0
0'001058  4E49                 114.          TRAP #SRQSRVC
0'00105A  41FA ****            115.          LEA S1PRO(PC),A0
0'00105E  43FA ****            116.          LEA S2PRO(PC),A1
0'001062  6100 0204            117.          BSR SENDOUT
                               118.  *
                               119.  * CONFIGURE NETWORK TO MASTER TO SLAVE 2
                               120.  * RESPOND TO SLAVE 2'S RFP SRQ, AND LOAD ITS
                               121.  * PROGRAM
                               122.  *
0'001066  13FC 00B4            123.          MOVE.B #$B4,PSR45
          00012FB2
0'00106E  103C 0004            124.          MOVE.B #SLV2,D0
0'001072  4E49                 125.          TRAP #SRQSRVC
0'001074  41FA ****            126.          LEA S2PRO(PC),A0
0'001078  43FA ****            127.          LEA S3PRO(PC),A1
0'00107C  6100 01EA            128.          BSR SENDOUT
                               129.  *
                               130.  * CONFIGURE NETWORK TO MASTER TO SLAVE 3
                               131.  * RESPOND TO SLAVE 3'S RFP SRQ, AND LOAD ITS
                               132.  * PROGRAM
                               133.  *
0'001080  13FC 00B4            134.          MOVE.B #$B4,PSR89
          00012FB4
0'001088  103C 0008            135.          MOVE.B #SLV3,D0
0'00108C  4E49                 136.          TRAP #SRQSRVC
0'00108E  41FA ****            137.          LEA S3PRO(PC),A0
0'001092  43FA ****            138.          LEA DONE(PC),A1
0'001096  6100 01D0            139.          BSR SENDOUT
                               140.  *
                               141.  * COPY INPUT DATA ARRAY TO DATAII SO EACH RUN
                               142.  * HAS SAME INPUT DATA
                               143.  *
0'00109A  41FA EF64            144.  REEXE   LEA DATAIN(PC),A0    ;SOURCE
0'00109E  43FA F360            145.          LEA DATAII(PC),A1    ;DEST
0'0010A2  323C 00FF            146.          MOVE.W #$FF,D1
0'0010A6  22D8                 147.  MVLP    MOVE.L (A0)+,(A1)+
0'0010A8  51C9 FFFC            148.          DBRA D1,MVLP
                               149.  *
                               150.  * ALL PROGRAMS LOADED, EACH ASSERTING READY FOR
                               151.  * DATA SRQ.  RESPOND TO ALL, THEN LOAD THE DATA
                               152.  * IN BROADCAST MODE.
```

```
                                153.  *
                                154.  * LOAD TIMER PRE-LOAD REGISTER HERE
                                155.  *
0'0010AC  297C 00FFFFFF         156.          MOVE.L #CNTSTRT,2(A4)
          0002
                                157.  *
                                158.  * SET UP SOME REGISTER VALUES
                                159.  *
0'0010B4  3C3C 0100             160.          MOVE.W #NPTE*4,D6
0'0010B8  267C 000003FC         161.          MOVEA.L #$3FC,A3
0'0010BE  247C 00000010         162.          MOVEA.L #$10,A2
0'0010C4  2C7C 00012FC0         163.          MOVEA.L #PGCR,A6
0'0010CA  2A7C 00012FB0         164.          MOVEA.L #PSR01,A5
0'0010D0  4243                  165.          CLR.W D3
0'0010D2  41FA FF36             166.          LEA MBRDCST(PC),A0
0'0010D6  4E48                  167.          TRAP #NETCNF
0'0010D8  103C 000F             168.          MOVE.B #ALLSLV,D0
0'0010DC  4E49                  169.          TRAP #SRQSRVC
0'0010DE  41FA F320             170.          LEA DATAII(PC),A0       ;START OF DATA
0'0010E2  43FA F71C             171.          LEA FILTER(PC),A1       ;END OF DATA+1
                                172.  *
                                173.  * START TIMER COUNTING HERE
                                174.  *
0'0010E6  18BC 0001             175.          MOVE.B #ENABLTM,(A4)
0'0010EA  93C8                  176.          SUBA.L A0,A1
0'0010EC  3009                  177.          MOVE.W A1,D0
0'0010EE  13FC 0020             178.          MOVE.B #ENABLEB,PGCR
          00012FC0
0'0010F6  4E47                  179.          TRAP #ADBYTOT
0'0010F8  4E46                  180.          TRAP #OUTDATA
0'0010FA  4239 00012FC0         181.          CLR.B PGCR
                                182.  *
                                183.  * WHEN HERE, ALL SLAVES LOADED, WILL ASSERT
                                184.  * READY TO EXECUTE SRQ.  RESPOND TO LET THEM
                                185.  * CONTINUE.
                                186.  *
0'001100  700F                  187.          MOVEQ.L #ALLSLV,D0
0'001102  4E49                  188.          TRAP #SRQSRVC           ;SLAVES EXECUTE
                                189.  *
                                190.  * SLAVES ASSERT SRQ WHEN ALL DONE FILTERING
                                191.  * ACKNOWLEDGE, THEN CONFIGURE TO ALLOW
                                192.  * THE TRANSFER SLAVE#0 TO REMAINING SLAVES
                                193.  *
0'001104  4E49                  194.          TRAP #SRQSRVC
0'001106  1ABC 0000             195.          MOVE.B #$00,(A5)
0'00110A  1B7C 0070 0001        196.          MOVE.B #$70,1(A5)
                                197.  *
                                198.  * SLV0 TO OTHERS COMPLETES, FOLLOWED BY SRQ
                                199.  * CYCLE. ACKNOWLEDGE, THEN CONFIGURE FOR SLV1
                                200.  * TO REMAINDER OF SLAVES.
                                201.  * ALLSLV IS STILL IN D0.B
                                202.  *
0'001110  4E49                  203.          TRAP #SRQSRVC
0'001112  1B7C 00E0 0001        204.          MOVE.B #$E0,1(A5)
0'001118  1B7C 00C7 0003        205.          MOVE.B #$C7,3(A5)
0'00111E  1B7C 004C 0005        206.          MOVE.B #$4C,5(A5)
                                207.
                                208.  *
                                209.  * SLV1 TO OTHERS COMPLETES, FOLLOWED BY SRQ
                                210.  * CYCLE. ACKNOWLEDGE, THEN CONFIGURE FOR SLV2
                                211.  * TO REMAINDER OF SLAVES
                                212.  * ALLSLV IS STILL IN D0
                                213.  *
0'001124  4E49                  214.          TRAP #SRQSRVC
0'001126  1B7C 00CE 0003        215.          MOVE.B #$CE,3(A5)
0'00112C  1B7C 0070 0004        216.          MOVE.B #$70,4(A5)
0'001132  1B7C 004C 0007        217.          MOVE.B #$4C,7(A5)
                                218.  *
                                219.  * SLV2 TO OTHERS COMPLETES, FOLLOWED BY SRQ
                                220.  * CYCLE. ACKNOWLEDGE, THEN CONFIGURE FOR SLV3
                                221.  * TO REMAINDER OF SLAVES
                                222.  * ALLSLV IS STILL IN D0
                                223.  *
0'001138  4E49                  224.          TRAP #SRQSRVC
0'00113A  1B7C 0080 0004        225.          MOVE.B #$80,4(A5)
0'001140  1B7C 00B8 0008        226.          MOVE.B #$B8,8(A5)
                                227.  *
                                228.  * WHEN DONE, WILL HAVE AN SRQ CYCLE FROM
                                229.  * SLAVES TO BEGIN THE INVERSE FFT CALCULATION
                                230.  * ACKNOWLEDGE; THEN IT IS SAFE TO CONFIGURE
                                231.  * NETWORK FROM SLAVE #0 TO MASTER.
```

```
                                232.  *
0'001146  4E49                  233.          TRAP #SRQSRVC
0'001148  41FA FEB6             234.          LEA INITCNF(PC),A0
0'00114C  4E48                  235.          TRAP #NETCNF
                                236.  *
                                237.  * SLAVES ASSERT SRQ, THEN UPLOAD DATA
                                238.  * WHEN ACKNOWLEDGED.
                                239.  *****
                                240.  * THIS VERSION ACKNOWLEDGES INDIVIDUALLY
                                241.  * THEN UPLOADS FROM EACH
                                242.  *****
                                243.  *
0'00114E  7001                  244.  REDIST  MOVEQ.L #SLV0,D0
0'001150  4E49                  245.          TRAP #SRQSRVC
0'001152  3006                  246.          MOVE.W D6,D0        ;BYTE COUNT
0'001154  41FA FAAA             247.          LEA XDATA0(PC),A0       ;DEST ADDRESS
0'001158  1CBC 0010             248.          MOVE.B #ENABLEA,(A6)
0'00115C  4E45                  249.          TRAP #INDATA
0'00115E  4216                  250.          CLR.B (A6)
                                251.  *
                                252.  * CONFIGURE FOR SLAVE #1 TO MASTER,
                                253.  * AND GET DATA
                                254.  *
0'001160  7002                  255.          MOVEQ.L #SLV1,D0
0'001162  4E49                  256.          TRAP #SRQSRVC
0'001164  1B7C 00BC 0005        257.          MOVE.B #$BC,5(A5)
0'00116A  3006                  258.          MOVE.W D6,D0
0'00116C  41FA FB92             259.          LEA XDATA1(PC),A0
0'001170  1CBC 0010             260.          MOVE.B #ENABLEA,(A6)
0'001174  4E45                  261.          TRAP #INDATA
0'001176  4216                  262.          CLR.B (A6)
                                263.  *
                                264.  * CONFIGURE FOR SLAVE #2 TO MASTER,
                                265.  * AND GET DATA
                                266.  *
0'001178  7004                  267.          MOVEQ.L #SLV2,D0
0'00117A  4E49                  268.          TRAP #SRQSRVC
0'00117C  1B7C 00BB 0007        269.          MOVE.B #$BB,7(A5)
0'001182  3006                  270.          MOVE.W D6,D0
0'001184  41FA FC7A             271.          LEA XDATA2(PC),A0
0'001188  1CBC 0010             272.          MOVE.B #ENABLEA,(A6)
0'00118C  4E45                  273.          TRAP #INDATA
0'00118E  4216                  274.          CLR.B (A6)
                                275.  *
                                276.  * CONFIGURE FOR SLAVE #3 TO MASTER,
                                277.  * AND GET DATA
                                278.  *
0'001190  7008                  279.          MOVEQ.L #SLV3,D0
0'001192  4E49                  280.          TRAP #SRQSRVC
0'001194  1B7C 00BB 0009        281.          MOVE.B #$BB,9(A5)
0'00119A  3006                  282.          MOVE.W D6,D0
0'00119C  41FA FD62             283.          LEA XDATA3(PC),A0
0'0011A0  1CBC 0010             284.          MOVE.B #ENABLEA,(A6)
0'0011A4  4E45                  285.          TRAP #INDATA
0'0011A6  4216                  286.          CLR.B (A6)
                                287.  *
                                288.  * HAVE ALL DATA IN, NOW DO REARRANGE FROM THE
                                289.  * XDATAi ARRAY TO THE DATAII ARRAY FOR CORRECT
                                290.  * ORDER.
                                291.  *
0'0011A8  41FA FA56             292.  REORDER LEA XDATA0(PC),A0
0'0011AC  43FA F252             293.          LEA DATAII(PC),A1
0'0011B0  7203                  294.          MOVEQ.L #NPE-1,D1
0'0011B2  7A3F                  295.  RLOOPT  MOVEQ.L #NPTE-1,D5
0'0011B4  2298                  296.  RLOOPN  MOVE.L (A0)+,(A1)
0'0011B6  D3CA                  297.          ADDA.L A2,A1
0'0011B8  51CD FFFA             298.          DBRA D5,RLOOPN
0'0011BC  93CB                  299.          SUBA.L A3,A1
0'0011BE  51C9 FFF2             300.          DBRA D1,RLOOPT
                                301.  *
                                302.  *
                                303.  * PROGRAM IS DONE, CAN STOP TIMER, AND
                                304.  * OUTPUT NUMBER OF 32 CLOCK INTERVALS
                                305.  * PASSED. FIRST DISABLE TIMER
                                306.  *
0'0011C2  4214                  307.  SKIP    CLR.B (A4)
0'0011C4  222C 0006             308.          MOVE.L 6(A4),D1 ;GET NEW COUNT
0'0011C8  203C 00FFFFFF         309.          MOVE.L #CNTSTRT,D0
0'0011CE  9081                  310.          SUB.L D1,D0     ;GET ELAPSED COUNT
                                311.  *
                                312.  * OUTPUT CR, LF, THEN THE COUNT IN HEX
```

```
                                313.  *
0'0011D0  4E4F                  314.          TRAP #15
0'0011D2  000A                  315.          DC.W SENCLFC
0'0011D4  4E4F                  316.          TRAP #15
0'0011D6  0018                  317.          DC.W DISBUF8
0'0011D8  2F00                  318.          MOVE.L D0,-(SP)
0'0011DA  41FA 0119             319.          LEA SPC(PC),A0
0'0011DE  4E4F                  320.          TRAP #15
0'0011E0  0011                  321.          DC.W OUTMESC
0'0011E2  2007                  322.          MOVE.L D7,D0
0'0011E4  4E4F                  323.          TRAP #15
0'0011E6  0018                  324.          DC.W DISBUF8
0'0011E8  201F                  325.          MOVE.L (SP)+,D0
                                326.  *
                                327.  * NOW KEEP TRACK OF STATISTICS
                                328.  *
0'0011EA  D197                  329.          ADD.L D0,(SP)    ;ADD TO SUM
0'0011EC  B0AF 0004             330.          CMP.L 4(SP),D0   ;CHECK MAX
0'0011F0  6F 04                 331.          BLE.S NOTMAX     ;D0 NOT MAX
0'0011F2  2F40 0004             332.          MOVE.L D0,4(SP)  ;NEW MAX
0'0011F6  B0AF 0008             333.  NOTMAX  CMP.L 8(SP),D0   ;CHECK MIN
0'0011FA  6C 04                 334.          BGE.S NOTMIN     ;D0 NOT MIN
0'0011FC  2F40 0008             335.          MOVE.L D0,8(SP)  ;NEW MIN
0'001200  51CF FE98             336.  NOTMIN  DBRA D7,REEXE    ;DO D7 TIMES
                                337.
                                338.  *
                                339.  * WHEN HERE, PROGRAM HAS BEEN EXECUTED 256 TIMES
                                340.  * SEND AN ILLEGAL ADDRESS FOR DATA, FORCING
                                341.  * SLAVES TO PROGRAM ACCEPT MODE
                                342.  *
0'001204  41FA FE04             343.          LEA MBRDCST(PC),A0
0'001208  4E48                  344.          TRAP #NETCNF
0'00120A  103C 000F             345.          MOVE.B #ALLSLV,D0
0'00120E  4E49                  346.          TRAP #SRQSRVC
0'001210  207C FFFF0000         347.          MOVEA.L #$FFFF0000,A0    ;ILLEGAL ADDR
0'001216  13FC 0020             348.          MOVE.B #ENABLEB,PGCR
          00012FC0
0'00121E  4E47                  349.          TRAP #ADBYTOT    ;THAT'LL FIX 'EM
0'001220  4239 00012FC0         350.          CLR.B PGCR
                                351.  *
                                352.  * OUTPUT STATISTICS
                                353.  *
0'001226  4E4F                  354.          TRAP #15
0'001228  000A                  355.          DC.W SENCLFC
0'00122A  4E4F                  356.          TRAP #15
0'00122C  000A                  357.          DC.W SENCLFC
0'00122E  41FA 0050             358.          LEA AVEMES(PC),A0
0'001232  4E4F                  359.          TRAP #15
0'001234  0011                  360.          DC.W OUTMESC
0'001236  201F                  361.          MOVE.L (SP)+,D0 ;GET SUM
0'001238  E080                  362.          ASR.L #DIV256,D0          ;DO AVERAGE
0'00123A  4E4F                  363.          TRAP #15
0'00123C  0018                  364.          DC.W DISBUF8     ;OUTPUT AVERAGE
0'00123E  4E4F                  365.          TRAP #15
0'001240  000A                  366.          DC.W SENCLFC
0'001242  41FA 0063             367.          LEA MAXMES(PC),A0
0'001246  4E4F                  368.          TRAP #15
0'001248  0011                  369.          DC.W OUTMESC
0'00124A  201F                  370.          MOVE.L (SP)+,D0 ;GET MAX
0'00124C  4E4F                  371.          TRAP #15
0'00124E  0018                  372.          DC.W DISBUF8
0'001250  4E4F                  373.          TRAP #15
0'001252  000A                  374.          DC.W SENCLFC
0'001254  41FA 0078             375.          LEA MINMES(PC),A0
0'001258  4E4F                  376.          TRAP #15
0'00125A  0011                  377.          DC.W OUTMESC
0'00125C  201F                  378.          MOVE.L (SP)+,D0 ;GET MIN
0'00125E  4E4F                  379.          TRAP #15
0'001260  0018                  380.          DC.W DISBUF8
0'001262  4E4F                  381.          TRAP #15
0'001264  000A                  382.          DC.W SENCLFC
0'001266  4E75                  383.          RTS
                                384.  *
                                385.  * SUBROUTINE SENDOUT
                                386.  *
0'001268  93C8                  387.  SENDOUT SUBA.L A0,A1
0'00126A  3009                  388.          MOVE.W A1,D0
0'00126C  13FC 0020             389.          MOVE.B #ENABLEB,PGCR
          00012FC0
0'001274  4E47                  390.          TRAP #ADBYTOT
0'001276  4E46                  391.          TRAP #OUTDATA
```

```
0'001278  4239 00012FC0   392.          CLR.B PGCR
0'00127E  4E75            393.          RTS
                          394.  *
0'001280  41 76 65 72 61  395.  AVEMES  DC.B 'Average time ( x 4 for microseconds)= ',EOT
          67 65 20 74 69
          6D 65 20 28 20
          78 20 34 20 66
          6F 72 20 6D 69
          63 72 6F 73 65
          63 6F 6E 64 73
          29 3D 20 04
0'0012A7  4D 61 78 69 6D  396.  MAXMES  DC.B 'Maximum time ( x 4 for microseconds)= ',EOT
          75 6D 20 74 69
          6D 65 20 28 20
          78 20 34 20 66
          6F 72 20 6D 69
          63 72 6F 73 65
          63 6F 6E 64 73
          29 3D 20 04
0'0012CE  4D 69 6E 69 6D  397.  MINMES  DC.B 'Minimum time ( x 4 for microseconds)= ',EOT
          75 6D 20 74 69
          6D 65 20 28 20
          78 20 34 20 66
          6F 72 20 6D 69
          63 72 6F 73 65
          63 6F 6E 64 73
          29 3D 20 04
0'0012F5  20 20 20 20 04  398.  SPC     DC.B '    ',EOT
0'0012FA                  399.          END
    0 Errors
*
* LINK SPEC FOR FSTFL1NM
*
      LINK FSTFL1NM
      LINK FSTSLVNM
      ORG $1000
      SECTION 0,15
      END
```

M.9 Program FSTFL2NM: Multicomputer Frequency Domain Filtering Program #2B

```
                                1.          TTL FSTFL2NM
                                2.  *
                                3.  * This program performs a 256 complex point
                                4.  * frequency domain filtering operation on an
                                5.  * input signal at DATAII.  The filter
                                6.  * coefficients are at FILTER, and are
                                7.  * distributed among slave  using another
                                8.  * program (filter.a68).
                                9.  *
                               10.  * input/output data is stored as real/imaginary
                               11.  * pair, each component 16-bits long.
                               12.  * twiddle factors within slave programs are
                               13.  * stored similarly
                               14.  *
                               15.  * APRIL 11,1992, CHANGE FOR FASTER
                               16.  * DATA REDISTRIBUTION BEFORE INVERSE
                               17.  * TRANSFORMING
                               18.  *
                               19.  * self timed operation, 256 runs
                               20.  * THIS VERSION DOESN'T GET SLAVE TIMES
                               21.  * AT END
                               22.  *
                               23.  * EQUATES
0'000000                       24.          SECTION 0
                               25.          XDEF PGCR,ENABLEA,ENABLEB
                               26.          XDEF NPT,NPTE,NPE,SHIFT14
                               27.          XDEF TCR,CNTSTRT,ENABLTM
                               28.          XREF S0PRO,S1PRO,S2PRO,S3PRO,DONE
                               29.          XDEF DATAII,FILTER,MBRDCST,INITCNF
                               30.          XDEF XDATA0,XDATA1,XDATA2,XDATA3
                               31.          XREF GETSLV
                               32.  *
     # 00012FC0                33.  PGCR     EQU $12FC0
     # 00000010                34.  ENABLEA  EQU $10
     # 00000020                35.  ENABLEB  EQU $20
     # 00000100                36.  NPT      EQU $100
     # 00000040                37.  NPTE     EQU $40
     # 00000004                38.  NPE      EQU 4
     # 0000000E                39.  SHIFT14  EQU 14
     # 00000003                40.  SRQASRT  EQU 3
     # 00000004                41.  ADBYTIN  EQU 4
     # 00000007                42.  ADBYTOT  EQU 7
     # 00000005                43.  INDATA   EQU 5
     # 00000006                44.  OUTDATA  EQU 6
     # 00000008                45.  NETCNF   EQU 8
     # 00000009                46.  SRQSRVC  EQU 9
     # 0000000E                47.  ABORT    EQU 14
     # 00000001                48.  SLV0     EQU $01
     # 00000002                49.  SLV1     EQU $02
     # 00000004                50.  SLV2     EQU $04
     # 00000008                51.  SLV3     EQU $08
     # 0000000F                52.  ALLSLV   EQU $0F
     # 00000006                53.  NUGAM    EQU 6
     # 00012FB0                54.  PSR01    EQU $12FB0
     # 00012FB1                55.  PSR23    EQU $12FB1
     # 00012FB2                56.  PSR45    EQU $12FB2
     # 00012FB3                57.  PSR67    EQU $12FB3
     # 00012FB4                58.  PSR89    EQU $12FB4
     # 00012FB5                59.  PSR1011  EQU $12FB5
     # 00012FB6                60.  PSR1213  EQU $12FB6
     # 00012FB7                61.  PSR1415  EQU $12FB7
     # 00012FB8                62.  PSR1617  EQU $12FB8
     # 00012FB9                63.  PSR1819  EQU $12FB9
     # 00012FD0                64.  TCR      EQU $12FD0
     # 00012FD2                65.  TPLR     EQU TCR+2
     # 00012FD6                66.  TCNTR    EQU TCR+6
     # 0000000A                67.  SENCLFC  EQU 10
     # 00000018                68.  DISBUF8  EQU 24
     # 0000001A                69.  DISBUF2  EQU 26
     # 00FFFFFF                70.  CNTSTRT  EQU $00FFFFFF
     # 00000001                71.  ENABLTM  EQU 1
     # 00000011                72.  OUTMESC  EQU 17
     # 00000008                73.  DIV256   EQU 8
     # 00000100                74.  PASSCNT  EQU 256
     # 00000004                75.  EOT      EQU 4
                               76.  ****************************************
```

```
                                77.  * THIS VERSION WAITS FOR ALL SLAVES TO
                                78.  * ASSERT SYNC BEFORE COLLECTING SLAVE
                                79.  * DATA, THEN
                                80.  * COLLECTS IN ORDER 0,1,2,3
                                81.  ****************************************
                                82.  ******************
                                83.  * MASTER PROGRAM *
                                84.  ******************
                                85.  *
0'000000  <400>                 86.  DATAIN  DS.B $400
0'000400  <400>                 87.  DATAII  DS.B $400
0'000800  <400>                 88.  FILTER  DS.B $400
0'000C00  <100>                 89.  XDATA0  DS.B $100
0'000D00  <100>                 90.  XDATA1  DS.B $100
0'000E00  <100>                 91.  XDATA2  DS.B $100
0'000F00  <100>                 92.  XDATA3  DS.B $100
0'001000  BB BB BB CB BB        93.  INITCNF DC.B $BB,$BB,$BB,$CB,$BB,$8C,$CB,$8B,$B8.$8B
          8C CB 8B B8 8B
0'00100A  BB DB 00 CE E0        94.  MBRDCST DC.B $BB,$DB,$00,$CE,$E0,$7C,$C8,$07,$88,$4B
          7C C8 07 88 4B
                                95.  *
                                96.  * CONFIGURE NETWORK TO INITIAL STATE, MASTER
                                97.  * TO SLAVE 0
                                98.  * RESPOND TO SLAVE 0'S RFP SRQ, AND LOAD
                                99.  * ITS PROGRAM
                               100.  *
0'001014  287C 00012FD0         101.  STRTFFT MOVEA.L #TCR,A4          ;POINTER TIMER
0'00101A  4214                  102.          CLR.B (A4)               ;DISABLE F'SURE
0'00101C  2F3C 00FFFFFF         103.          MOVE.L #CNTSTRT,-(SP)    ;MIN TIME
0'001022  2F3C 00000000         104.          MOVE.L #$00,-(SP)        ;MAX TIME
0'001028  2F3C 00000000         105.          MOVE.L #$00,-(SP)        ;RUN'G SUM TIME
0'00102E  2E3C 000000FF         106.          MOVE.L #PASSCNT-1,D7     ;COUNTER
0'001034  41FA FFCA             107.          LEA INITCNF(PC),A0
0'001038  4E48                  108.          TRAP #NETCNF
0'00103A  103C 0001             109.          MOVE.B #SLV0,D0
0'00103E  4E49                  110.          TRAP #SRQSRVC
0'001040  41FA ****             111.          LEA S0PRO(PC),A0         ;STRT ADDR
0'001044  43FA ****             112.          LEA S1PRO(PC),A1         ;END ADDR+1
0'001048  6100 020C             113.          BSR SENDOUT              ;ADDR,COUNT,CODE
                                114.  *
                                115.  * CONFIGURE NETWORK TO MASTER TO SLAVE 1
                                116.  * RESPOND TO SLAVE 1'S RFP SRQ, AND LOAD ITS
                                117.  * PROGRAM
                                118.  *
0'00104C  13FC 00B4             119.          MOVE.B #$B4,PSR01
          00012FB0
0'001054  103C 0002             120.          MOVE.B #SLV1,D0
0'001058  4E49                  121.          TRAP #SRQSRVC
0'00105A  41FA ****             122.          LEA S1PRO(PC),A0
0'00105E  43FA ****             123.          LEA S2PRO(PC),A1
0'001062  6100 01F2             124.          BSR SENDOUT
                                125.  *
                                126.  * CONFIGURE NETWORK TO MASTER TO SLAVE 2
                                127.  * RESPOND TO SLAVE 2'S RFP SRQ, AND LOAD ITS
                                128   * PROGRAM
                                129.  *
0'001066  13FC 00B4             130.          MOVE.B #$B4,PSR45
          00012FB2
0'00106E  103C 0004             131.          MOVE.B #SLV2,D0
0'001072  4E49                  132.          TRAP #SRQSRVC
0'001074  41FA ****             133.          LEA S2PRO(PC),A0
0'001078  43FA ****             134.          LEA S3PRO(PC),A1
0'00107C  6100 01D8             135.          BSR SENDOUT
                                136.  *
                                137.  * CONFIGURE NETWORK TO MASTER TO SLAVE 3
                                138.  * RESPOND TO SLAVE 3'S RFP SRQ, AND LOAD ITS
                                139.  * PROGRAM
                                140.  *
0'001080  13FC 00B4             141.          MOVE.B #$B4,PSR89
          00012FB4
0'001088  103C 0008             142.          MOVE.B #SLV3,D0
0'00108C  4E49                  143.          TRAP #SRQSRVC
0'00108E  41FA ****             144.          LEA S3PRO(PC),A0
0'001092  43FA ****             145.          LEA DONE(PC),A1
0'001096  6100 01BE             146.          BSR SENDOUT
                                147.  *
                                148.  * COPY INPUT DATA ARRAY TO DATAII SO EACH RUN
                                149.  * HAS SAME INPUT DATA
                                150.  *
0'00109A  41FA EF64             151.  REEXE   LEA DATAIN(PC),A0    ;SOURCE
0'00109E  43FA F360             152.          LEA DATAII(PC),A1    ;DEST
```

```
0'0010A2  323C 00FF        153.          MOVE.W #$FF,D1
0'0010A6  22D8             154.  MVLP    MOVF.L (A0)+,(A1)+
0'0010A8  51C9 FFFC        155.          DBRA D1,MVLP
                           156.  *
                           157.  * ALL PROGRAMS LOADED, EACH ASSERTING READY FOR
                           158.  * DATA SRQ.  RESPOND TO ALL, THEN LOAD THE DATA
                           159.  * IN BROADCAST MODE.
                           160.  *
                           161.  * LOAD TIMER PRE-LOAD REGISTER HERE
                           162.  *
0'0010AC  297C 00FFFFFF    163.          MOVE.L #CNTSTRT,2(A4)
          0002
                           164.  *
                           165.  * SET UP SOME REGISTER VALUES
                           166.  *
0'0010B4  3C3C 0100        167.          MOVE.W #NPTE*4,D6
0'0010B8  267C 000003FC    168.          MOVEA.L #$3FC,A3
0'0010BE  247C 00000010    169.          MOVEA.L #$10,A2
0'0010C4  2C7C 00012FC0    170.          MOVEA.L #PGCR,A6
0'0010CA  2A7C 00012FB0    171.          MOVEA.L #PSR01,A5
0'0010D0  4243             172.          CLR.W D3
0'0010D2  41FA FF36        173.          LEA MBRDCST(PC),A0
0'0010D6  4E48             174.          TRAP #NETCNF
0'0010D8  103C 000F        175.          MOVE B #ALLSLV,D0
0'0010DC  4E49             176.          TRAP #SRQSRVC
0'0010DE  41FA F320        177.          LEA DATAII(PC),A0        ,START OF DATA
0'0010E2  43FA F71C        178.          LEA FILTER(PC),A1        ;END OF DATA+1
                           179.  *
                           180.  * START TIMER COUNTING HERE
                           181.  *
0'0010E6  18BC 0001        182.          MOVE.B #ENABLTM,(A4)
                           183.  *
                           184.  * DOWNLOAD INPUT DATA ARRAY
                           185.  *
0'0010EA  93C8             186.          SUBA.L A0,A1
0'0010EC  3009             187           MOVE.W A1,D0
0'0010EE  13FC 0020        188.          MOVE B #ENABLEB,PGCR
          00012FC0
0'0010F6  4E47             189.          TRAP #ADBYTOT
0'0010F8  4E46             190.          TRAP #OUTDATA
0'0010FA  4239 00012FC0    191.          CLR.B PGCR
                           192.  *
                           193.  * WHEN HERE, ALL SLAVES LOADED, WILL ASSERT
                           194.  * READY TO EXECUTE SRQ.  RESPOND TO LET THEM
                           195.  * CONTINUE.
                           196.  *
0'001100  700F             197.          MOVEQ.L #ALLSLV,D0
0'001102  4E49             198.          TRAP #SRQSRVC            ,SLAVES EXECUTE
                           199.  *
                           200.  * SLAVES ASSERT SRQ WHEN ALL DONE FILTERING
                           201.  * ACKNOWLEDGE, THEN CONFIGURE TO ALLOW
                           202.  * THE TRANSFER SLAVE#0 TO REMAINING SLAVES
                           203.  *
0'001104  4E49             204.  REDIST  TRAP #SRQSRVC
0'001106  1ABC 0000        205.          MOVE.B #$00,(A5)
0'00110A  1B7C 0070 0001   206.          MOVE B #$70,1(A5)
                           207.  *
                           208.  * SLV0 TO OTHERS COMPLETES, FOLLOWED BY SRQ
                           209.  * CYCLE. ACKNOWLEDGE, THEN CONFIGURE FOR SLV1
                           210   * TO REMAINDER OF SLAVES
                           211.  * ALLSLV IS STILL IN D0.B
                           212.  *
0'001110  4E49             213           TRAP #SRQSRVC
0'001112  1B7C 00E0 0001   214.          MOVE.B #$E0,1(A5)
0'001118  1B7C 00C7 0003   215.          MOVE.B #$C7,3(A5)
0'00111E  1B7C 004C 0005   216.          MOVE.B #$4C,5(A5)
                           217.
                           218.  *
                           219.  * SLV1 TO OTHERS COMPLETES, FOLLOWED BY SRQ
                           220.  * CYCLE. ACKNOWLEDGE, THEN CONFIGURE FOR SLV2
                           221.  * TO REMAINDER OF SLAVES
                           222.  * ALLSLV IS STILL IN D0
                           223   *
0'001124  4E49             224.          TRAP #SRQSRVC
0'001126  1B7C 00CE 0003   225.          MOVE B #$CE,3(A5)
0'00112C  1B7C 0070 0004   226.          MOVE.B #$70,4(A5)
0'001132  1B7C 004C 0007   227.          MOVE.B #$4C,7(A5)
                           228.  *
                           229.  * SLV2 TO OTHERS COMPLETES, FOLLOWED BY SRQ
                           230.  * CYCLE. ACKNOWLEDGE, THEN CONFIGURE FOR SLV3
                           231.  * TO REMAINDER OF SLAVES
```

```
                                232.  * ALLSLV IS STILL IN D0
                                233.  *
0'001138  4E49                  234.          TRAP #SRQSRVC
0'00113A  1B7C 0080 0004        235.          MOVE.B #$80,4(A5)
0'001140  1B7C 00B8 0008        236.          MOVE.B #$B8,8(A5)
                                237.  *
                                238.  * WHEN DONE, WILL HAVE AN SRQ CYCLE FROM
                                239.  * SLAVES TO BEGIN THE INVERSE FFT CALCULATION
                                240.  * ACKNOWLEDGE; THEN IT IS SAFE TO CONFIGURE
                                241.  * NETWORK FROM SLAVE #0 TO MASTER.
                                242.  *
0'001146  4E49                  243.          TRAP #SRQSRVC
0'001148  41FA FEB6             244.          LEA INITCNF(PC),A0
0'00114C  4E48                  245.          TRAP #NETCNF
                                246.  *
                                247.  * WHEN DONE THEIR INVFFT'S, WILL HAVE YET
                                248.  * ANOTHER SRQ CYCLE
                                249.  *******
                                250.  * THIS VERSION WAITS FOR ALL DONE BEFORE
                                251.  * FINAL UPLOAD
                                252.  *******
                                253.  *
0'00114E  4E49                  254.          TRAP #SRQSRVC
0'001150  3006                  255.          MOVE.W D6,D0
0'001152  41FA FAAC             256.          LEA XDATA0(PC),A0
0'001156  1CBC 0010             257.          MOVE.B #ENABLEA,(A6)
0'00115A  4E45                  258.          TRAP #INDATA
0'00115C  4216                  259.          CLR.B (A6)
                                260.  *
                                261.  * CONFIGURE FOR SLAVE #1 TO MASTER AND GET
                                262.  * DATA
                                263.  *
0'00115E  1B7C 00BC 0005        264.          MOVE.B #$BC,5(A5)
0'001164  3006                  265.          MOVE.W D6,D0
0'001166  41FA FB98             266.          LEA XDATA1(PC),A0
0'00116A  1CBC 0010             267.          MOVE.B #ENABLEA,(A6)
0'00116E  4E45                  268.          TRAP #INDATA
0'001170  4216                  269.          CLR.B (A6)
                                270.  *
                                271.  * CONFIGURE FOR SLAVE #2 TO MASTER,
                                272.  * AND GET DATA
                                273.  *
0'001172  1B7C 00BB 0007        274.          MOVE.B #$BB,7(A5)
                                275.  *       MOVE.W D6,D0
0'001178  41FA FC86             276.          LEA XDATA2(PC),A0
0'00117C  1CBC 0010             277.          MOVE.B #ENABLEA,(A6)
0'001180  4E45                  278.          TRAP #INDATA
0'001182  4216                  279           CLR.B (A6)
                                280.  *
                                281.  * CONFIGURE FOR SLAVE #3 TO MASTER,
                                282.  * AND GET DATA
                                283.  *
0'001184  1B7C 00BB 0009        284.          MOVE.B #$BB,9(A5)
                                285.  *       MOVE.W D6,D0
0'00118A  41FA FD74             286.          LEA XDATA3(PC),A0
0'00118E  1CBC 0010             287.          MOVE.B #ENABLEA,(A6)
0'001192  4E45                  288.          TRAP #INDATA
0'001194  4216                  289.          CLR.B (A6)
                                290.  *
                                291.  * HAVE ALL DATA IN, NOW DO REARRANGE FROM THE
                                292.  * XDATAi ARRAY TO THE DATAII ARRAY FOR CORRECT
                                293.  * ORDER.
                                294.  *
0'001196  41FA FA68             295.  REORDER LEA XDATA0(PC),A0
0'00119A  43FA F264             296.          LEA DATAII(PC),A1
0'00119E  7203                  297.          MOVEQ.L #NPE-1,D1
0'0011A0  7A3F                  298.  RLOOPT  MOVEQ.L #NPTE-1,D5
0'0011A2  2298                  299.  RLOOPN  MOVE.L (A0)+,(A1)
0'0011A4  D3CA                  300.          ADDA.L A2,A1
0'0011A6  51CD FFFA             301.          DBRA D5,RLOOPN
0'0011AA  93CB                  302.          SUBA.L A3,A1
0'0011AC  51C9 FFF2             303.          DBRA D1,RLOOPT
                                304.  *
                                305.  *
                                306.  * PROGRAM IS DONE, CAN STOP TIMER, AND
                                307.  * OUTPUT NUMBER OF 32 CLOCK INTERVALS
                                308.  * PASSED. FIRST DISABLE TIMER
                                309.  *
0'0011B0  4214                  310.  SKIP    CLR.B (A4)
0'0011B2  222C 0006             311.          MOVE.L 6(A4),D1 ;GET NEW COUNT
0'0011B6  203C 00FFFFFF         312.          MOVE.L #CNTSTRT,D0
```

```
0'0011BC  9081            313.          SUB.L D1,D0     ;GET ELAPSED COUNT
                          314.  *
                          315.  * OUTPUT CR, LF, THEN THE COUNT IN HEX
                          316.  *
0'0011BE  4E4F            317.          TRAP #15
0'0011C0  000A            318.          DC.W SENCLFC
0'0011C2  4E4F            319.          TRAP #15
0'0011C4  0018            320.          DC.W DISBUF8
0'0011C6  2F00            321.          MOVE.L D0,-(SP)
0'0011C8  41FA 0119       322.          LEA SPC(PC),A0
0'0011CC  4E4F            323.          TRAP #15
0'0011CE  0011            324.          DC.W OUTMESC
0'0011D0  2007            325.          MOVE.L D7,D0
0'0011D2  4E4F            326.          TRAP #15
0'0011D4  0018            327.          DC.W DISBUF8
0'0011D6  201F            328.          MOVE.L (SP)+,D0
                          329.  *
                          330.  * NOW KEEP TRACK OF STATISTICS
                          331.  *
0'0011D8  D197            332.          ADD L D0,(SP)    ;ADD TO SUM
0'0011DA  B0AF 0004       333.          CMP L 4(SP),D0   ;CHECK MAX
0'0011DE  6F 04           334.          BLE.S NOTMAX     ;D0 NOT MAX
0'0011E0  2F40 0004       335.          MOVE.L D0,4(SP)  ;NEW MAX
0'0011E4  B0AF 0008       336.  NOTMAX  CMP.L 8(SP),D0   ;CHECK MIN
0'0011E8  6C 04           337.          BGE.S NOTMIN     ;D0 NOT MIN
0'0011EA  2F40 0008       338.          MOVE.L D0,8(SP)  ;NEW MIN
0'0011EE  51CF FEAA       339.  NOTMIN  DBRA D7,REEXE    ;DO D7 TIMES
                          340.
                          341.  *
                          342.  * WHEN HERE, PROGRAM HAS BEEN EXECUTED 256 TIMES
                          343.  * SEND AN ILLEGAL ADDRESS FOR DATA, FORCING
                          344.  * SLAVES TO PROGRAM ACCEPT MODE
                          345.  * TO END THE PROGRAM
                          346.  *
0'0011F2  41FA FE16       347           LEA MBRDCST(PC),A0
0'0011F6  4E48            348.          TRAP #NETCNF
0'0011F8  103C 000F       349.          MOVE.B #ALLSLV,D0
0'0011FC  4E49            350.          TRAP #SRQSRVC
0'0011FE  207C FFFF0000   351.          MOVEA L #$FFFF0000,A0   ;ILLEGAL ADDR
0'001204  13FC 0020       352.          MOVE.B #ENABLEB,PGCR
          00012FC0
0'00120C  4E47            353.          TRAP #ADBYTOT    ;THAT'LL FIX 'EM
0'00120E  4239 00012FC0   354.          CLR B PGCR
                          355.  *
                          356.  * OUTPUT STATISTICS
                          357.  *
0'001214  4E4F            358.          TRAP #15
0'001216  000A            359           DC.W SENCLFC
0'001218  4E4F            360.          TRAP #15
0'00121A  000A            361.          DC.W SENCLFC
0'00121C  41FA 0050       362.          LEA AVEMES(PC).A0
0'001220  4E4F            363.          TRAP #15
0'001222  0011            364.          DC.W OUTMESC
0'001224  201F            365           MOVE.L (SP)+,D0 ;GET SUM
0'001226  E080            366.          ASR L #DIV256,D0         ;D0 AVERAGE
0'001228  4E4F            367.          TRAP #15
0'00122A  0018            368.          DC.W DISBUF8     ;OUTPUT AVERAGE
0'00122C  4E4F            369.          TRAP #15
0'00122E  000A            370.          DC.W SENCLFC
0'001230  41FA 0063       371.          LEA MAXMES(PC),A0
0'001234  4E4F            372.          TRAP #15
0'001236  0011            373.          DC.W OUTMESC
0'001238  201F            374.          MOVE.L (SP)+,D0 ;GET MAX
0'00123A  4E4F            375.          TRAP #15
0'00123C  0018            376.          DC.W DISBUF8
0'00123E  4E4F            377.          TRAP #15
0'001240  000A            378.          DC.W SENCLFC
0'001242  41FA 0078       379.          LEA MINMES(PC),A0
0'001246  4E4F            380.          TRAP #15
0'001248  0011            381           DC.W OUTMESC
0'00124A  201F            382.          MOVE.L (SP)+,D0 ;GET MIN
0'00124C  4E4F            383.          TRAP #15
0'00124E  0018            384.          DC.W DISBUF8
0'001250  4E4F            385.          TRAP #15
0'001252  000A            386.          DC.W SENCLFC
                          387   *
                          388.  * AFTER 256 RUNS,
                          389   * QUIT
                          390.  *
0'001254  4E75            391           RTS
                          392.  *
```

```
                                393.  * SUBROUTINE SENDOUT
                                394.  *
0'001256  93C8                  395.  SENDOUT SUBA.L A0,A1
0'001258  3009                  396.         MOVE.W A1,D0
0'00125A  13FC 0020             397          MOVE.B #ENABLEB,PGCR
          00012FC0
0'001262  4E47                  398.         TRAP #ADBYTOT
0'001264  4E46                  399.         TRAP #OUTDATA
0'001266  4239 00012FC0         400.         CLR.B PGCR
0'00126C  4E75                  401.         RTS
0'00126E  41 76 65 72 61        402.  AVEMES  DC.B 'Average time ( x 4 for microseconds)= ',EOT
          67 65 20 74 69
          6D 65 20 28 20
          78 20 34 20 66
          6F 72 20 6D 69
          63 72 6F 73 65
          63 6F 6E 64 73
          29 3D 20 04
0'001295  4D 61 78 69 6D        403.  MAXMES  DC B 'Maximum time ( x 4 for microseconds)= ',EOT
          75 6D 20 74 69
          6D 65 20 28 20
          78 20 34 20 66
          6F 72 20 6D 69
          63 72 6F 73 65
          63 6F 6E 64 73
          29 3D 20 04
0'0012BC  4D 69 6E 69 6D        404.  MINMES  DC.B 'Minimum time ( x 4 for microseconds)= ',EOT
          75 6D 20 74 69
          6D 65 20 28 20
          78 20 34 20 66
          6F 72 20 6D 69
          63 72 6F 73 65
          63 6F 6E 64 73
          29 3D 20 04
0'0012E3  20 20 20 20 04        405.  SPC     DC.B '    ',EOT
0'0012E8                        406           END
    0 Errors
*
* LINK SPEC FOR FSTFL2NM
*
      LINK FSTFL2NM
      LINK FSTSLVNM
      ORG $1000
      SECTION 0,15
      END
```

M.10 Program FSTFL3NM: Multicomputer Frequency Domain Filtering Program #3B

```
                                    1.          TTL FSTFIL3NM
                                    2.  *
                                    3.  * This program performs a 256 complex point
                                    4.  * frequency domain filtering operation on an
                                    5.  * input signal at DATAII.  The filter
                                    6.  * coefficients are at FILTER, and are
                                    7.  * distributed among slaves using another
                                    8.  * program (filter.a68).
                                    9.  *
                                   10.  * input/output data is stored as real/imaginary
                                   11.  * pair, each component 16-bits long.
                                   12.  * twiddle factors within slave programs are
                                   13.  * stored similarly
                                   14.  *
                                   15.  *
                                   16.  * self timed operation, 256 runs
                                   17.  *
                                   18.  * EQUATES
0'000000                           19.          SECTION 0
                                   20.          XDEF PGCR,ENABLEA,ENABLEB
                                   21.          XDEF NPT,NPTE,NPE,SHIFT14
                                   22.          XDEF TCR,CNTSTRT,ENABLTM
                                   23.          XREF S0PRO,S1PRO,S2PRO,S3PRO,DONE
                                   24.          XDEF DATAII,FILTER,MBRDCST
                                   25.          XDEF XDATA0,XDATA1,XDATA2,XDATA3
                                   26.  *
     # 00012FC0                    27.  PGCR     EQU $12FC0
     # 00000010                    28.  ENABLEA EQU $10
     # 00000020                    29.  ENABLEB EQU $20
     # 00000100                    30.  NPT      EQU $100
     # 00000040                    31.  NPTE     EQU $40
     # 00000004                    32.  NPE      EQU 4
     # 0000000E                    33.  SHIFT14 EQU 14
     # 00000003                    34.  SRQASRT EQU 3
     # 00000004                    35.  ADBYTIN EQU 4
     # 00000007                    36.  ADBYTOT EQU 7
     # 00000005                    37.  INDATA  EQU 5
     # 00000006                    38.  OUTDATA EQU 6
     # 00000008                    39.  NETCNF  EQU 8
     # 00000009                    40.  SRQSRVC EQU 9
     # 0000000E                    41.  ABORT   EQU 14
     # 00000001                    42.  SLV0     EQU $01
     # 00000002                    43.  SLV1     EQU $02
     # 00000004                    44.  SLV2     EQU $04
     # 00000008                    45.  SLV3     EQU $08
     # 0000000F                    46.  ALLSLV   EQU $0F
     # 00000006                    47.  NUGAM    EQU 6
     # 00012FB0                    48.  PSR01    EQU $12FB0
     # 00012FB1                    49.  PSR23    EQU $12FB1
     # 00012FB2                    50.  PSR45    EQU $12FB2
     # 00012FB3                    51.  PSR67    EQU $12FB3
     # 00012FB4                    52.  PSR89    EQU $12FB4
     # 00012FB5                    53.  PSR1011 EQU $12FB5
     # 00012FB6                    54.  PSR1213 EQU $12FB6
     # 00012FB7                    55.  PSR1415 EQU $12FB7
     # 00012FB8                    56.  PSR1617 EQU $12FB8
     # 00012FB9                    57.  PSR1819 EQU $12FB9
     # 00012FD0                    58.  TCR      EQU $12FD0
     # 00012FD2                    59.  TPLR     EQU TCR+2
     # 00012FD6                    60.  TCNTR    EQU TCR+6
     # 0000000A                    61   SENCLFC EQU 10
     # 00000018                    62.  DISBUF8 EQU 24
     # 0000001A                    63.  DISBUF2 EQU 26
     # 00FFFFFF                    64.  CNTSTRT EQU $00FFFFFF
     # 00000001                    65.  ENABLTM EQU 1
     # 00000011                    66.  OUTMESC EQU 17
     # 00000008                    67.  DIV256  EQU 8
     # 00000100                    68.  PASSCNT EQU 256
     # 00000004                    69.  EOT      EQU 4
                                   70.  ****************************************
                                   71   * THIS VERSION WIATS FOR SLAVE 0 SYNC,
                                   72.  * THEN UPLOADS, THEN WAITS FOR ALL OF
                                   73.  * SLAVE 1,2,3 SYNCS THEN UPLOADS
                                   74.  * INDIVIDUALLY IN ORDER 1,2,3
                                   75.  ****************************************
                                   76.  ******************
```

```
                                77.  * MASTER PROGRAM *
                                78.  ******************
                                79.  *
0'000000  <400>                 80.  DATAIN  DS.B $400
0'000400  <400>                 81.  DATAII  DS.B $400
0'000800  <400>                 82.  FILTER  DS.B $400
0'000C00  <100>                 83.  XDATA0  DS.B $100
0'000D00  <100>                 84.  XDATA1  DS.B $100
0'000E00  <100>                 85.  XDATA2  DS.B $100
0'000F00  <100>                 86.  XDATA3  DS.B $100
0'001000  BB BB BB CB BB        87.  INITCNF DC.B $BB,$BB,$BB,$CB,$BB,$8C,$CB,$8B,$B8,$8B
          8C CB 8B B8 8B
0'00100A  BB DB 00 CE E0        88.  MBRDCST DC.B $BB,$DB,$00,$CE,$E0,$7C,$C8,$07,$88,$4B
          7C C8 07 88 4B
                                89.  *
                                90.  * CONFIGURE NETWORK TO INITIAL STATE, MASTER
                                91.  * TO SLAVE 0
                                92.  * RESPOND TO SLAVE 0'S RFP SRQ, AND LOAD
                                93.  * ITS PROGRAM
                                94.  *
0'001014  287C 00012FD0         95.  STRTFFT MOVEA.L #TCR,A4          ;POINTER TIMER
0'00101A  4214                  96.          CLR.B (A4)               ;DISABLE F'SURE
0'00101C  2F3C 00FFFFFF         97.          MOVE.L #CNTSTRT,-(SP)    ;MIN TIME
0'001022  2F3C 00000000         98.          MOVE.L #$00,-(SP)        ;MAX TIME
0'001028  2F3C 00000000         99.          MOVE.L #$00,-(SP)        ;RUN'G SUM TIME
0'00102E  2E3C 000000FF        100.          MOVE.L #PASSCNT-1,D7     ;COUNTER
0'001034  41FA FFCA            101.          LEA INITCNF'PC),A0
0'001038  4E48                 102.          TRAP #NETC...
0'00103A  103C 0001            103.          MOVE.B #SLV0,D0
0'00103E  4E49                 104.          TRAP #SRQSRVC
0'001040  41FA ****            105.          LEA S0PRO(PC),A0         ;STRT ADDR
0'001044  43FA ****            106.          LEA S1PRO(PC),A1         ;END ADDR+1
0'001048  6100 0216            107.          BSR SENDOUT              ;ADDR,COUNT,CODE
                               108.  *
                               109.  * CONFIGURE NETWORK TO MASTER TO SLAVE 1
                               110.  * RESPOND TO SLAVE 1'S RFP SRQ, AND LOAD ITS
                               111.  * PROGRAM
                               112.  *
0'00104C  13FC 00B4            113.          MOVE.B #$B4,PSR01
          00012FB0
0'001054  103C 0002            114.          MOVE.B #SLV1,D0
0'001058  4E49                 115.          TRAP #SRQSRVC
0'00105A  41FA ****            116.          LEA S1PRO(PC),A0
0'00105E  43FA ****            117.          LEA S2PRO(PC),A1
0'001062  6100 01FC            118.          BSR SENDOUT
                               119.  *
                               120.  * CONFIGURE NETWORK TO MASTER TO SLAVE 2
                               121.  * RESPOND TO SLAVE 2'S RFP SRQ, AND LOAD ITS
                               122.  * PROGRAM
                               123.  *
0'001066  13FC 00B4            124.          MOVE.B #$B4,PSR45
          00012FB2
0'00106E  103C 0004            125.          MOVE.B #SLV2,D0
0'001072  4E49                 126.          TRAP #SRQSRVC
0'001074  41FA ****            127.          LEA S2PRO(PC),A0
0'001078  43FA ****            128.          LEA S3PRO(PC),A1
0'00107C  6100 01E2            129.          BSR SENDOUT
                               130.  *
                               131.  * CONFIGURE NETWORK TO MASTER TO SLAVE 3
                               132.  * RESPOND TO SLAVE 3'S RFP SRQ, AND LOAD ITS
                               133.  * PROGRAM
                               134.  *
0'001080  13FC 00B4            135.          MOVE.B #$B4,PSR89
          00012FB4
0'001088  103C 0008            136.          MOVE.B #SLV3,D0
0'00108C  4E49                 137.          TRAP #SRQSRVC
0'00108E  41FA ****            138.          LEA S3PRO(PC),A0
0'001092  43FA ****            139.          LEA DONE(PC),A1
0'001096  6100 01C8            140.          BSR SENDOUT
                               141.  *
                               142.  * COPY INPUT DATA ARRAY TO DATAII SO EACH RUN
                               143.  * HAS SAME INPUT DATA
                               144.  *
0'00109A  41FA EF64            145.  REEXE    LEA DATAIN(PC),A0    ;SOURCE
0'00109E  43FA F360            146.          LEA DATAII(PC),A1    ;DEST
0'0010A2  323C 00FF            147.          MOVE.W #$FF,D1
0'0010A6  22D8                 148.  MVLP     MOVE.L (A0)+,(A1)+
0'0010A8  51C9 FFFC            149.          DBRA D1,MVLP
                               150.  *
                               151   * ALL PROGRAMS LOADED, EACH ASSERTING READY FOR
                               152.  * DATA SRQ.  RESPOND TO ALL, THEN LOAD THE DATA
```

```
                                153.  * IN BROADCAST MODE.
                                154.  *
                                155.  * LOAD TIMER PRE-LOAD REGISTER HERE
                                156.  *
0'0010AC  297C 00FFFFFF         157.          MOVE.L #CNTSTRT,2(A4)
          0C02
                                158.  *
                                159.  * SET UP SOME REGISTER VALUES
                                160.  *
0'0010B4  3C3C 0100             161.          MOVE.W #NPTE*4,D6
0'0010B8  267C 000003FC         162.          MOVEA.L #$3FC,A3
0'0010BE  247C 00000010         163.          MOVEA.L #$10,A2
0'0010C4  2C7C 00012FC0         164.          MOVEA.L #PGCR,A6
0'0010CA  2A7C 00012FB0         165.          MOVEA.L #PSR01,A5
0'0010D0  4243                  166.          CLR.W D3
0'0010D2  41FA FF36             167.          LEA MBRDCST(PC),A0
0'0010D6  4E46                  168.          TRAP #NETCNF
0'0010D8  103C 000F             169.          MOVE.B #ALLSLV,D0
0'0010DC  4E49                  170.          TRAP #SRQSRVC
0'0010DE  41FA F320             171.          LEA DATAII(PC),A0       ;START OF DATA
0'0010E2  43FA F71C             172.          LEA FILTER(PC),A1       ;END OF DATA+1
                                173.  *
                                174.  * START TIMER COUNTING HERE
                                175.  *
0'0010E6  18BC 0001             176.          MOVE.B #ENABLTM,(A4)
0'0010EA  93C8                  177.          SUBA.L A0,A1
0'0010EC  3009                  178.          MOVE.W A1,D0
0'0010EE  13FC 0020             179           MOVE.B #ENABLEB,PGCR
          00012FC0
0'0010F6  4E47                  180.          TRAP #ADBYTOT
0'0010F8  4E46                  181.          TRAP #OUTDATA
0'0010FA  4239 00012FC0         182.          CLR.B PGCR
                                183.  *
                                184.  * WHEN HERE, ALL SLAVES LOADED, WILL ASSERT
                                185.  * READY TO EXECUTE SRQ.  RESPOND TO LET THEM
                                186.  * CONTINUE.
                                187   *
0'001100  700F                  188.          MOVEQ.L #ALLSLV,D0
0'001102  4E49                  189           TRAP #SRQSRVC           ;SLAVES EXECUTE
                                190.  *
                                191.  * SLAVES ASSERT SRQ WHEN ALL DONE FILTERING
                                192.  * ACKNOWLEDGE, THEN CONFIGURE TO ALLOW
                                193.  * THE TRANSFER SLAVE#0 TO REMAINING SLAVES
                                194.  *
0'001104  4E49                  195           TRAP #SRQSRVC
0'001106  1ABC 0000             196.          MOVE.B #$00,(A5)
0'00110A  1B7C 0070 0001        197.          MOVE.B #$70,1(A5)
                                198   *
                                199.  * SLV0 TO OTHERS COMPLETES, FOLLOWED BY SRQ
                                200.  * CYCLE. ACKNOWLEDGE, THEN CONFIGURE FOR SLV1
                                201.  * TO REMAINDER OF SLAVES.
                                202.  * ALLSLV IS STILL IN D0.B
                                203.  *
0'001110  4E49                  204.          TRAP #SRQSRVC
0'001112  1B7C 00E0 0001        205.          MOVE.B #$E0,1(A5)
0'001118  1B7C 00C7 0003        206.          MOVE.B #$C7,3(A5)
0'00111E  1B7C 004C 0005        207.          MOVE.B #$4C,5(A5)
                                208.
                                209.  *
                                210.  * SLV1 TO OTHERS COMPLETES, FOLLOWED BY SRQ
                                211.  * CYCLE. ACKNOWLEDGE, THEN CONFIGURE FOR SLV2
                                212.  * TO REMAINDER OF SLAVES
                                213.  * ALLSLV IS STILL IN D0
                                214.  *
0'001124  4E49                  215.          TRAP #SRQSRVC
0'001126  1B7C 00CE 0003        216.          MOVE.B #$CE,3(A5)
0'00112C  1B7C 0070 0004        217.          MOVE.B #$70,4(A5)
0'001132  1B7C 004C 0007        218.          MOVE.B #$4C,7(A5)
                                219.  *
                                220.  * SLV2 TO OTHERS COMPLETES, FOLLOWED BY SRQ
                                221.  * CYCLE. ACKNOWLEDGE, THEN CONFIGURE FOR SLV3
                                222.  * TO REMAINDER OF SLAVES
                                223.  * ALLSLV IS STILL IN D0
                                224.  *
0'001138  4E49                  225.          TRAP #SRQSRVC
0'00113A  1B7C 0080 0004        226.          MOVE.B #$80,4(A5)
0'001140  1B7C 00B8 0008        227.          MOVE.B #$B8,8(A5)
                                228.  *
                                229.  * WHEN DONE, WILL HAVE AN SRQ CYCLE FROM
                                230.  * SLAVES TO BEGIN THE INVERSE FFT CALCULATION
                                231.  * ACKNOWLEDGE; THEN IT IS SAFE TO CONFIGURE
```

```
                                232.  * NETWORK FROM SLAVE #0 TO MASTER.
                                233.  *
0'001146  4E49                  234.          TRAP #SRQSRVC
0'001148  41FA FEB6             235.          LEA INITCNF(PC),A0
0'00114C  4E48                  236.          TRAP #NETCNF
                                237.  *
                                238.  * WHEN DONE THEIR INVFFT'S, WILL HAVE YET
                                239.  * ANOTHER SRQ CYCLE
                                240.  *******
                                241.  * THIS VERSION SERVICES SLAVE #0, THEN
                                242.  * UPLOADS DATA, THEN SERVICES REST
                                243.  * AND UPLOADS IN SEQUENCE
                                244.  *******
                                245.  *
                                246.  *
0'00114E  7001                  247.  REDIST  MOVEQ.L #SLV0,D0
0'001150  4E49                  248.          TRAP #SRQSRVC
0'001152  3006                  249.          MOVE.W D6,D0        ;BYTE COUNT
0'001154  41FA FAAA             250.          LEA XDATA0(PC),A0       ;DEST ADDRESS
0'001158  1CBC 0010             251.          MOVE.B #ENABLEA,(A6)
0'00115C  4E45                  252.          TRAP #INDATA
0'00115E  4216                  253.          CLR.B (A6)
                                254.  *
                                255.  * CONFIGURE FOR SLAVE #1 TO MASTER,
                                256.  * AND GET DATA
                                257.  *
0'001160  700E                  258.          MOVEQ.L #$0E,D0
0'001162  4E49                  259.          TRAP #SRQSRVC
0'001164  1B7C 00BC 0005        260.          MOVE.B #$BC,5(A5)
0'00116A  3006                  261.          MOVE.W D6,D0
0'00116C  41FA FB92             262.          LEA XDATA1(PC),A0
0'001170  1CBC 0010             263.          MOVE.B #ENABLEA,(A6)
0'001174  4E45                  264.          TRAP #INDATA
0'001176  4216                  265.          CLR.B (A6)
                                266.  *
                                267.  * CONFIGURE FOR SLAVE #2 TO MASTER,
                                268.  * AND GET DATA
                                269.  *
0'001178  1B7C 00BB 0007        270.          MOVE.B #$BB,7(A5)
0'00117E  3006                  271.          MOVE.W D6,D0
0'001180  41FA FC7E             272.          LEA XDATA2(PC),A0
0'001184  1CBC 0010             273.          MOVE.B #ENABLEA,(A6)
0'001188  4E45                  274.          TRAP #INDATA
0'00118A  4216                  275.          CLR.B (A6)
                                276.  *
                                277.  * CONFIGURE FOR SLAVE #3 TO MASTER,
                                278.  * AND GET DATA
                                279.  *
0'00118C  1B7C 00BB 0009        280.          MOVE.B #$BB,9(A5)
0'001192  3006                  281.          MOVE.W D6,D0
0'001194  41FA FD6A             282.          LEA XDATA3(PC),A0
0'001198  1CBC 0010             283.          MOVE.B #ENABLEA,(A6)
0'00119C  4E45                  284.          TRAP #INDATA
0'00119E  4216                  285.          CLR.B (A6)
                                286.  *
                                287.  * HAVE ALL DATA IN, NOW DO REARRANGE FROM THE
                                288.  * XDATAi ARRAY TO THE DATAII ARRAY FOR CORRECT
                                289.  * ORDER.
                                290.  *
0'0011A0  41FA FA5E             291.  REORDER LEA XDATA0(PC),A0
0'0011A4  43FA F25A             292.          LEA DATAII(PC),A1
0'0011A8  7203                  293.          MOVEQ.L #NPE-1,D1
0'0011AA  7A3F                  294.  RLOOPT  MOVEQ.L #NPTE-1,D5
0'0011AC  2298                  295.  RLOOPN  MOVE.L (A0)+,(A1)
0'0011AE  D3CA                  296.          ADDA.L A2,A1
0'0011B0  51CD FFFA             297           DBRA D5,RLOOPN
0'0011B4  93CB                  298.          SUBA.L A3,A1
0'0011B6  51C9 FFF2             299.          DBRA D1,RLOOPT
                                300.  *
                                301.  * PROGRAM IS DONE, CAN STOP TIMER, AND
                                302   * OUTPUT NUMBER OF 32 CLOCK INTERVALS
                                303.  * PASSED. FIRST DISABLE TIMER
                                304.  *
0'0011BA  4214                  305.  SKIP     CLR.B (A4)
0'0011BC  222C 0006             306.          MOVE.L 6(A4),D1 ;GET NEW COUNT
0'0011C0  203C 00FFFFFF         307.          MOVE.L #CNTSTRT,D0
0'0011C6  9081                  308.          SUB.L D1,D0     ;GET ELAPSED COUNT
                                309.  *
                                310.  * OUTPUT CR, LF, THEN THE COUNT IN HEX
                                311.  *
0'0011C8  4E4F                  312.          TRAP #15
```

```
0'0011CA  000A             313.          DC.W SENCLFC
0'0011CC  4E4F             314.          TRAP #15
0'0011CE  0018             315.          DC.W DISBUF8
0'0011D0  2F00             316.          MOVE.L D0,-(SP)
0'0011D2  41FA 0119        317.          LEA SPC(PC),A0
0'0011D6  4E4F             318.          TRAP #15
0'0011D8  0011             319.          DC.W OUTMESC
0'0011DA  2007             320.          MOVE.L D7,D0
0'0011DC  4E4F             321.          TRAP #15
0'0011DE  0018             322.          DC.W DISBUF8
0'0011E0  201F             323.          MOVE.L (SP)+,D0
                           324.  *
                           325.  * NOW KEEP TRACK OF STATISTICS
                           326.  *
0'0011E2  D197             327.          ADD.L D0,(SP)    ;ADD TO SUM
0'0011E4  B0AF 0004        328.          CMP.L 4(SP),D0   ;CHECK MAX
0'0011E8  6F 04            329.          BLE.S NOTMAX     ;D0 NOT MAX
0'0011EA  2F40 0004        330.          MOVE.L D0,4(SP)  ;NEW MAX
0'0011EE  B0AF 0008        331.  NOTMAX  CMP.L 8(SP),D0   ;CHECK MIN
0'0011F2  6C 04            332.          BGE.S NOTMIN     ;D0 NOT MIN
0'0011F4  2F40 0008        333.          MOVE.L D0,8(SP)  ;NEW MIN
0'0011F8  51CF FEA0        334.  NOTMIN  DBRA D7,REEXE    ;DO D7 TIMES
                           335.
                           336.  *
                           337.  * WHEN HERE, PROGRAM HAS BEEN EXECUTED 256 TIMES
                           338.  * SEND AN ILLEGAL ADDRESS FOR DATA, FORCING
                           339.  * SLAVES TO PROGRAM ACCEPT MODE
                           340.  *
0'0011FC  41FA FE0C        341.          LEA MBRDCST(PC),A0
0'001200  4E48             342.          TRAP #NETCNF
0'001202  103C 000F        343.          MOVE.B #ALLSLV,D0
0'001206  4E49             344.          TRAP #SRQSRVC
0'001208  207C FFFF0000    345.          MOVEA.L #$FFFF0000,A0    ,ILLEGAL ADDR
0'00120E  13FC 0020        346.          MOVE.B #ENABLEB,PGCR
          00012FC0
0'001216  4E47             347.          TRAP #ADBYTOT    ;THAT'LL FIX 'EM
0'001218  4239 00012FC0    348.          CLR.B PGCR
                           349.  *
                           350.  * OUTPUT STATISTICS
                           351.  *
0'00121E  4E4F             352.          TRAP #15
0'001220  000A             353.          DC.W SENCLFC
0'001222  4E4F             354.          TRAP #15
0'001224  000A             355.          DC.W SENCLFC
0'001226  41FA 0050        356.          LEA AVEMES(PC),A0
0'00122A  4E4F             357.          TRAP #15
0'00122C  0011             358.          DC.W OUTMESC
0'00122E  201F             359           MOVE.L (SP)+,D0 ;GET SUM
0'001230  E080             360.          ASR.L #DIV256,D0          ;DO AVERAGE
0'001232  4E4F             361.          TRAP #15
0'001234  0018             362.          DC.W DISBUF8     ;OUTPUT AVERAGE
0'001236  4E4F             363.          TRAP #15
0'001238  000A             364.          DC.W SENCLFC
0'00123A  41FA 0063        365.          LEA MAXMES(PC),A0
0'00123E  4E4F             366.          TRAP #15
0'001240  0011             367.          DC.W OUTMESC
0'001242  201F             368.          MOVE.L (SP)+,D0 ;GET MAX
0'001244  4E4F             369.          TRAP #15
0'001246  0018             370.          DC.W DISBUF8
0'001248  4E4F             371.          TRAP #15
0'00124A  000A             372.          DC.W SENCLFC
0'00124C  41FA 0078        373.          LEA MINMES(PC),A0
0'001250  4E4F             374.          TRAP #15
0'001252  0011             375.          DC W OUTMESC
0'001254  201F             376.          MOVE.L (SP)+,D0 ;GET MIN
0'001256  4E4F             377.          TRAP #15
0'001258  0018             378.          DC.W DISBUF8
0'00125A  4E4F             379.          TRAP #15
0'00125C  000A             380.          DC.W SENCLFC
                           381.  *
0'00125E  4E75             382.          RTS
                           383.  *
                           384.  * SUBROUTINE SENDOUT
                           385.  *
0'001260  93C8             386.  SENDOUT SUBA.L A0,A1
0'001262  3009             387.          MOVE.W A1,D0
0'001264  13FC 0020        388.          MOVE.B #ENABLEB,PGCR
          00012FC0
0'00126C  4E47             389.          TRAP #ADBYTOT
0'00126E  4E46             390.          TRAP #OUTDATA
0'001270  4239 00012FC0    391.          CLR.B PGCR
```

```
0'001276  4E75              392.          RTS
                            393.  *
0'001278  41 76 65 72 61    394.  AVEMES  DC.B 'Average time ( x 4 for microseconds)= ',EOT
          67 65 20 74 69
          6D 65 20 28 20
          78 20 34 20 66
          6F 72 20 6D 69
          63 72 6F 73 65
          63 6F 6E 64 73
          29 3D 20 04
0'00129F  4D 61 78 69 6D    395.  MAXMES  DC.B 'Maximum time ( x 4 for microseconds)= ',EOT
          75 6D 20 74 69
          6D 65 20 28 20
          78 20 34 20 66
          6F 72 20 6D 69
          63 72 6F 73 65
          63 6F 6E 64 73
          29 3D 20 04
0'0012C6  4D 69 6E 69 6D    396.  MINMES  DC.B 'Minimum time ( x 4 for microseconds)= ',EOT
          75 6D 20 74 69
          6D 65 20 28 20
          78 20 34 20 66
          6F 72 20 6D 69
          63 72 6F 73 65
          63 6F 6E 64 73
          29 3D 20 04
0'0012ED  20 20 20 20 04    397.  SPC     DC.B '    ',EOT
                            398.  *
                            399.  *
0'0012F2                    400.          END
    0 Errors
*
* LINK SPEC FOR FSTFL3NM
*
     LINK FSTFL3NM
     LINK FSTSLVNM
     ORG $1000
     SECTION 0,15
     END
```

M.11 Program FSTFL4NM: Multicomputer Frequency Domain Filtering Program #4B

```
                                1.          TTL FSTFL4NM
                                2.  *
                                3.  * This program performs a 256 complex point
                                4.  * frequency domain filtering operation on an
                                5.  * input signal at DATAII.  The filter
                                6.  * coefficients are at FILTER, and are
                                7.  * distributed among slaves using another
                                8.  * program (filter.a68).
                                9.  *
                               10.  * input/output data is stored as real/imaginary
                               11.  * pair, each component 16-bits long.
                               12.  * twiddle factors within slave programs are
                               13.  * stored similarly
                               14.  *
                               15.  *
                               16.  * self timed operation, 256 runs
                               17.  *
                               18.  * EQUATES
0'000000                       19.          SECTION 0
                               20.          XDEF PGCR,ENABLEA,ENABLEB
                               21.          XDEF NPT,NPTE,NPE,SHIFT14
                               22.          XDEF TCR,CNTSTRT,ENABLTM
                               23.          XREF S0PRO,S1PRO,S2PRO,S3PRO,DONE
                               24.          XDEF DATAII,FILTER,MBRDCST
                               25.          XDEF XDATA0,XDATA1,XDATA2,XDATA3
                               26.  *
     # 00012FC0                27.  PGCR     EQU $12FC0
     # 00012FCC                28.  PCDR     EQU $12FCC
     # 00000010                29.  ENABLEA  EQU $10
     # 00000020                30.  ENABLEB  EQU $20
     # 00000100                31.  NPT      EQU $100
     # 00000040                32.  NPTE     EQU $40
     # 00000004                33.  NPE      EQU 4
     # 0000000E                34.  SHIFT14  EQU 14
     # 00000003                35.  SRQASRT  EQU 3
     # 00000004                36.  ADBYTIN  EQU 4
     # 00000007                37.  ADBYTOT  EQU 7
     # 00000005                38.  INDATA   EQU 5
     # 00000006                39.  OUTDATA  EQU 6
     # 00000008                40.  NETCNF   EQU 8
     # 00000009                41.  SRQSRVC  EQU 9
     # 0000000E                42.  ABORT    EQU 14
     # 00000001                43.  SLV0     EQU $01
     # 00000002                44.  SLV1     EQU $02
     # 00000004                45.  SLV2     EQU $04
     # 00000008                46.  SLV3     EQU $08
     # 0000000F                47.  ALLSLV   EQU $0F
     # 00000006                48.  NUGAM    EQU 6
     # 00012FB0                49.  PSR01    EQU $12FB0
     # 00012FB1                50.  PSR23    EQU $12FB1
     # 00012FB2                51.  PSR45    EQU $12FB2
     # 00012FB3                52.  PSR67    EQU $12FB3
     # 00012FB4                53.  PSR89    EQU $12FB4
     # 00012FB5                54.  PSR1011  EQU $12FB5
     # 00012FB6                55.  PSR1213  EQU $12FB6
     # 00012FB7                56.  PSR1415  EQU $12FB7
     # 00012FB8                57.  PSR1617  EQU $12FB8
     # 00012FB9                58.  PSR1819  EQU $12FB9
     # 00012FD0                59.  TCR      EQU $12FD0
     # 00012FD2                60.  TPLR     EQU TCR+2
     # 00012FD6                61.  TCNTR    EQU TCR+6
     # 0000000A                62.  SENCLFC  EQU 10
     # 00000018                63.  DISBUF8  EQU 24
     # 0000001A                64.  DISBUF2  EQU 26
     # 00FFFFFF                65.  CNTSTRT  EQU $00FFFFFF
     # 00000001                66.  ENABLTM  EQU 1
     # 00000011                67.  OUTMESC  EQU 17
     # 00000008                68.  DIV256   EQU 8
     # 00000100                69.  PASSCNT  EQU 256
     # 00000004                70.  EOT      EQU 4
                               71.  ****************************************
                               72.  * THIS VERSION USES FIRST COME FIRST
                               73.  * SERVED STRATEGY FOR SYNC'S AND UPLOADS
                               74.  ****************************************
                               75.  ******************
                               76.  * MASTER PROGRAM *
```

```
                                    77.  ******************
                                    78.  *
0'000000  <400>                     79.  DATAIN  DS.B $400
0'000400  <400>                     80.  DATAII  DS.B $400
0'000800  <400>                     81.  FILTER  DS.B $400
0'000C00  <100>                     82.  XDATA0  DS.B $100
0'000D00  <100>                     83.  XDATA1  DS.B $100
0'000E00  <100>                     84.  XDATA2  DS.B $100
0'000F00  <100>                     85.  XDATA3  DS.B $100
0'001000  BB BB BB CB BB            86.  INITCNF DC.B $BB,$BB,$BB,$CB,$BB,$8C,$CB,$8B,$B8,$8B
          8C CB 8B B8 8B
0'00100A  BB DB 00 CE E0            87.  MBRDCST DC.B $BB,$DB,$00,$CE,$E0,$7C,$C8,$07,$88,$4B
          7C C8 07 88 4B
                                    88.  *
                                    89.  * CONFIGURE NETWORK TO INITIAL STATE, MASTER
                                    90.  * TO SLAVE 0
                                    91.  * RESPOND TO SLAVE 0'S RFP SRQ, AND LOAD
                                    92.  * ITS PROGRAM
                                    93.  *
0'001014  287C 00012FD0             94.  STRTFFT MOVEA.L #TCR,A4          ;POINTER TIMER
0'00101A  4214                      95.          CLR.B (A4)               ;DISABLE F'SURE
0'00101C  2F3C 00FFFFFF             96.          MOVE.L #CNTSTRT,-(SP)    ;MIN TIME
0'001022  2F3C 00000000             97.          MOVE.L #$00,-(SP)        ;MAX TIME
0'001028  2F3C 00000000             98.          MOVE.L #$00,-(SP)        ;RUN'G SUM TIME
0'00102E  2E3C 000000FF             99.          MOVE.L #PASSCNT-1,D7     ;COUNTER
0'001034  41FA FFCA                100.          LEA INITCNF(PC),A0
0'001038  4E48                     101.          TRAP #NETCNF
0'00103A  103C 0001                102.          MOVE.B #SLV0,D0
0'00103E  4E49                     103.          TRAP #SRQSRVC
0'001040  41FA ****                104.          LEA S0PRO(PC),A0         ;STRT ADDR
0'001044  43FA ****                105.          LEA S1PRO(PC),A1         ;END ADDR+1
0'001048  6100 02DC                106.          BSR SENDOUT              ;ADDR,COUNT,CODE
                                   107.  *
                                   108.  * CONFIGURE NETWORK TO MASTER TO SLAVE 1
                                   109.  * RESPOND TO SLAVE 1'S RFP SRQ, AND LOAD ITS
                                   110.  * PROGRAM
                                   111.  *
0'00104C  13FC 00B4                112.          MOVE.B #$B4,PSR01
          00012FB0
0'001054  103C 0002                113.          MOVE.B #SLV1,D0
0'001058  4E49                     114.          TRAP #SRQSRVC
0'00105A  41FA ****                115.          LEA S1PRO(PC),A0
0'00105E  43FA ****                116.          LEA S2PRO(PC),A1
0'001062  6100 02C2                117.          BSR SENDOUT
                                   118.  *
                                   119.  * CONFIGURE NETWORK TO MASTER TO SLAVE 2
                                   120.  * RESPOND TO SLAVE 2'S RFP SRQ, AND LOAD ITS
                                   121.  * PROGRAM
                                   122.  *
0'001066  13FC 00B4                123.          MOVE.B #$B4,PSR45
          00012FB2
0'00106E  103C 0004                124.          MOVE.B #SLV2,D0
0'001072  4E49                     125.          TRAP #SRQSRVC
0'001074  41FA ****                126.          LEA S2PRO(PC),A0
0'001078  43FA ****                127.          LEA S3PRO(PC),A1
0'00107C  6100 02A8                128.          BSR SENDOUT
                                   129.  *
                                   130.  * CONFIGURE NETWORK TO MASTER TO SLAVE 3
                                   131.  * RESPOND TO SLAVE 3'S RFP SRQ, AND LOAD ITS
                                   132.  * PROGRAM
                                   133.  *
0'001080  13FC 00B4                134.          MOVE.B #$B4,PSR89
          00012FB4
0'001088  103C 0008                135.          MOVE.B #SLV3,D0
0'00108C  4E49                     136.          TRAP #SRQSRVC
0'00108E  41FA ****                137.          LEA S3PRO(PC),A0
0'001092  43FA ****                138.          LEA DONE(PC),A1
0'001096  6100 028E                139.          BSR SENDOUT
                                   140.  *
                                   141.  * COPY INPUT DATA ARRAY TO DATAII SO EACH RUN
                                   142.  * HAS SAME INPUT DATA
                                   143.  *
0'00109A  41FA EF64                144.  REEXE    LEA DATAIN(PC),A0    ;SOURCE
0'00109E  43FA F360                145.          LEA DATAII(PC),A1    ;DEST
0'0010A2  323C 00FF                146.          MOVE.W #$FF,D1
0'0010A6  22D8                     147.  MVLP     MOVE.L (A0)+,(A1)+
0'0010A8  51C9 FFFC                148.          DBRA D1,MVLP
                                   149.  *
                                   150.  * ALL PROGRAMS LOADED, EACH ASSERTING READY FOR
                                   151.  * DATA SRQ.  RESPOND TO ALL, THEN LOAD THE DATA
                                   152.  * IN BROADCAST MODE.
```

```
                                153.  *
                                154.  * LOAD TIMER PRE-LOAD REGISTER HERE
                                155.  *
0'0010AC  297C 00FFFFFF         156.          MOVE.L #CNTSTRT,2(A4)
          0002
                                157.  *
                                158.  * SET UP SOME REGISTER VALUES
                                159.  *
0'0010B4  3C3C 0100             160.          MOVE.W #NPTE*4,D6
0'0010B8  267C 000003FC         161.          MOVEA.L #$3FC,A3
0'0010BE  247C 00000010         162.          MOVEA.L #$10,A2
0'0010C4  2C7C 00012FC0         163.          MOVEA.L #PGCR,A6
0'0010CA  2A7C 00012FB0         164.          MOVEA.L #PSR01,A5
0'0010D0  4243                  165.          CLR.W D3
0'0010D2  41FA FF36             166.          LEA MBRDCST(PC),A0
0'0010D6  4E48                  167.          TRAP #NETCNF
0'0010D8  103C 000F             168.          MOVE.B #ALLSLV,D0
0'0010DC  4E49                  169.          TRAP #SRQSRVC
0'0010DE  41FA F320             170.          LEA DATAII(PC),A0        ;START OF DATA
0'0010E2  43FA F71C             171.          LEA FILTER(PC),A1        ;END OF DATA+1
                                172.  *
                                173.  * START TIMER COUNTING HERE
                                174.  *
0'0010E6  18BC 0001             175.          MOVE.B #ENABLTM,(A4)
0'0010EA  93C8                  176.          SUBA.L A0,A1
0'0010EC  3009                  177.          MOVE.W A1,D0
0'0010EE  13FC 0020             178.          MOVE.B #ENABLEB,PGCR
          00012FC0
0'0010F6  4E47                  179.          TRAP #ADBYTOT
0'0010F8  4E46                  180.          TRAP #OUTDATA
0'0010FA  4239 00012FC0         181.          CLR.B PGCR
                                182.  *
                                183.  * WHEN HERE, ALL SLAVES LOADED, WILL ASSERT
                                184.  * READY TO EXECUTE SRQ.  RESPOND TO LET THEM
                                185.  * CONTINUE.
                                186.  *
0'001100  700F                  187.          MOVEQ.L #ALLSLV,D0
0'001102  4E49                  188.          TRAP #SRQSRVC            ,SLAVES EXECUTE
                                189.  *
                                190.  * SLAVES ASSERT SRQ WHEN ALL DONE FILTERING
                                191.  * ACKNOWLEDGE, THEN CONFIGURE TO ALLOW
                                192.  * THE TRANSFER SLAVE#0 TO REMAINING SLAVES
                                193.  *
0'001104  4E49                  194.          TRAP #SRQSRVC
0'001106  1ABC 0000             195.          MOVE.B #$00,(A5)
0'00110A  1B7C 0070 0001        196.          MOVE.B #$70,1(A5)
                                197.  *
                                198.  * SLV0 TO OTHERS COMPLETES, FOLLOWED BY SRQ
                                199.  * CYCLE. ACKNOWLEDGE, THEN CONFIGURE FOR SLV1
                                200.  * TO REMAINDER OF SLAVES.
                                201.  * ALLSLV IS STILL IN D0.B
                                202.  *
0'001110  4E49                  203.          TRAP #SRQSRVC
0'001112  1B7C 00E0 0001        204.          MOVE.B #$E0,1(A5)
0'001118  1B7C 00C7 0003        205.          MOVE.B #$C7,3(A5)
0'00111E  1B7C 004C 0005        206.          MOVE.B #$4C,5(A5)
                                207.
                                208.  *
                                209.  * SLV1 TO OTHERS COMPLETES, FOLLOWED BY SRQ
                                210.  * CYCLE. ACKNOWLEDGE, THEN CONFIGURE FOR SLV2
                                211.  * TO REMAINDER OF SLAVES
                                212.  * ALLSLV IS STILL IN D0
                                213.  *
0'001124  4E49                  214.          TRAP #SRQSRVC
0'001126  1B7C 00CE 0003        215.          MOVE.B #$CE,3(A5)
0'00112C  1B7C 0070 0004        216.          MOVE.B #$70,4(A5)
0'001132  1B7C 004C 0007        217.          MOVE.B #$4C,7(A5)
                                218.  *
                                219.  * SLV2 TO OTHERS COMPLETES, FOLLOWED BY SRQ
                                220.  * CYCLE. ACKNOWLEDGE, THEN CONFIGURE FOR SLV3
                                221.  * TO REMAINDER OF SLAVES
                                222.  * ALLSLV IS STILL IN D0
                                223.  *
0'001138  4E49                  224.          TRAP #SRQSRVC
0'00113A  1B7C 0080 0004        225.          MOVE.B #$80,4(A5)
0'001140  1B7C 00B8 0008        226.          MOVE.B #$B8,8(A5)
                                227.  *
                                228.  * WHEN DONE, WILL HAVE AN SRQ CYCLE FROM
                                229.  * SLAVES TO BEGIN THE INVERSE FFT CALCULATION
                                230.  * ACKNOWLEDGE; THEN IT IS SAFE TO CONFIGURE
                                231.  * NETWORK FROM SLAVE #0 TO MASTER.
```

```
                                232.  *
0'001146  4E49                  233.          TRAP #SRQSRVC
0'001148  41FA FEB6             234.          LEA INITCNF(PC),A0
0'00114C  4E48                  235.          TRAP #NETCNF
                                236.  *
                                237.  * WHEN DONE THEIR INVERT'S, WILL HAVE YET
                                238.  * ANOTHER SRQ CYCLE
                                239.  ******
                                240.  * THIS VERSION USES FCFS
                                241.  * FOR FINAL UPLOAD
                                242.  *******
                                243.  *
                                244.  *
                                245.  * WHEN DONE, SLAVES ASSERT SRQ, THEN UPLOAD DATA
                                246.  * WHEN ACKNOWLEDGED.  ACKNOWLEDGE FCFS, AND UPLOAD
                                247.  * FCFS
                                248.  *
0'00114E  700F                  249.  REDIST  MOVEQ.L #ALLSLV,D0
0'001150  0800 0000             250.  TSTB0   BTST.L #0,D0
0'001154  67 44                 251.          BEQ.S TSTB1
0'001156  123C 0000             252.          MOVE.B #0,D1
0'00115A  6100 01E2             253.          BSR TSTSLV
0'00115E  0801 0007             254.          BTST.L #7,D1     ;SERVICED?
0'001162  67 36                 255.          BEQ.S TSTB1      ;IF NOT GOTO SLV1
                                256.  *
                                257.  * SLAVE 0 SENSED AND ACKNOWLEDGED, GET DATA
                                258.  *
0'001164  1B7C 0080 0003        259.          MOVE.B #$80,3(A5)
0'00116A  1B7C 0080 0005        260.          MOVE.B #$80,5(A5)
0'001170  1B7C 0080 0007        261.          MOVE.B #$80,7(A5)
0'001176  1B7C 0080 0009        262.          MOVE.B #$80,9(A5)
0'00117C  3F00                  263.          MOVE.W D0,-(SP)
0'00117E  3006                  264.          MOVE.W D6,D0        ;BYTE COUNT
0'001180  41FA FA7E             265.          LEA XDATA0(PC),A0       ;DEST ADDRESS
0'001184  1CBC 0010             266.          MOVE.B #ENABLEA,(A6)
0'001188  4E45                  267.          TRAP #INDATA
0'00118A  4216                  268.          CLR.B (A6)
0'00118C  301F                  269           MOVE.W (SP)+,D0
0'00118E  0880 0000             270.          BCLR.L #0,D0     ;CLEAR THE BIT
0'001192  0C00 0000             271.          CMPI.B #0,D0
0'001196  6700 00CE             272.          BEQ REORDER
0'00119A  0800 0001             273.  TSTB1   BTST.L #1,D0
0'00119E  67 3E                 274.          BEQ.S TSTB2
0'0011A0  123C 0001             275.          MOVE.B #1,D1
0 0011A4  6100 0198             276.          BSR TSTSLV
0'0011A8  0801 0007             277.          BTST L #7,D1     ;SERVICED?
0'0011AC  67 30                 278.          BEQ.S TSTB2      ;IF NOT GOTO SLV1
                                279.  *
                                280   * SLAVE 1 SENSED AND ACKNOWLEDGED, GET DATA
                                281.  *
0'0011AE  1B7C 00B8 0005        282.          MOVE.B #$B8,5(A5)
0'0011B4  1B7C 0080 0007        283.          MOVE.B #$80,7(A5)
0'0011BA  1B7C 0080 0009        284.          MOVE.B #$80,9(A5)
0 0011C0  3F00                  285.          MOVE.W D0,-(SP)
0'0011C2  3006                  286.          MOVE.W D6,D0        ;BYTE COUNT
0'0011C4  41FA FB3A             287.          LEA XDATA1(PC),A0       ;DEST ADDRESS
0'0011C8  1CBC 0010             288           MOVE.B #ENABLEA,(A6)
0'0011CC  4E45                  289.          TRAP #INDATA
0'0011CE  4216                  290.          CLR.B (A6)
0'0011D0  301F                  291.          MOVE.W (SP)+,D0
0'0011D2  0880 0001             292.          BCLR.L #1,D0     ;CLEAR THE BIT
0'0011D6  0C00 0000             293.          CMPI.B #0,D0
0'0011DA  6700 008A             294.          BEQ REORDER
0'0011DE  0800 0002             295.  TSTB2   BTST.L #2,D0
0'0011E2  67 3E                 296           BEQ S TSTB3
0'0011E4  123C 0002             297.          MOVE.B #2,D1
0'0011E8  6100 0154             298.          BSR TSTSLV
0'0011EC  0801 0007             299.          BTST.L #7,D1     ;SERVICED?
0'0011F0  67 30                 300.          BEQ.S TSTB3      ;IF NOT GOTO SLV1
                                301   *
                                302.  * SLAVE 2 SENSED AND ACKNOWLEDGED, GET DATA
                                303.  *
0'0011F2  1B7C 0080 0006        304.          MOVE.B #$80,6(A5)
0 0011F8  1B7C 00BB 0007        305.          MOVE.B #$BB,7(A5)
0'0011FE  1B7C 0080 0009        306.          MOVE.B #$80,9(A5)
0'001204  3F00                  307.          MOVE.W D0,-(S.)
0'001206  3006                  308.          MOVE.W D6,D0        ;BYTE COUNT
0'001208  41FA FBF6             309.          LEA XDATA2(PC),A0       ;DEST ADDRESS
0'00120C  1CBC 0010             310.          MOVE.B #ENABLEA,(A6)
0'001210  4E45                  311.          TRAP #INDATA
0'001212  4216                  312.          CLR.B (A6)
```

```
0'001214  301F              313.          MOVE.W (SP)+,D0
0'001216  0880 0002         314.          BCLR L #2,D0     ;CLEAR THE BIT
0'00121A  0C00 0000         315.          CMPI.B #0,D0
0'00121E  6700 0046         316.          BEQ REORDER
0'001222  0800 0003         317.  TSTB3   BTST.L #3,D0
0'001226  6700 FF28         318.          BEQ TSTB0
0'00122A  123C 0003         319.          MOVE.B #3,D1
0'00122E  6100 010E         320.          BSR TSTSLV
0'001232  0801 0007         321.          BTST.L #7,D1     ,SERVICED?
0'001236  6700 FF18         322.          BEQ TSTB0        ;IF NOT GOTO SLV1
                            323.  *
                            324.  * SLAVE 3 SENSED AND ACKNOWLEDGED, GET DATA
                            325.  *
0'00123A  1B7C 00B8 0008    326.          MOVE.B #$B8,8(A5)
0'001240  1B7C 00BB 0009    327.          MOVE.B #$BB,9(A5)
0'001246  3F00              328.          MOVE.W D0,-(SP)
0'001248  3006              329.          MOVE W D6,D0        ;BYTE COUNT
0'00124A  41FA FCB4         330.          LEA XDATA3(PC),A0       ,DEST ADDRESS
0'00124E  1CBC 0010         331.          MOVE.B #ENABLEA,(A6)
0'001252  4E45              332.          TRAP #INDATA
0'001254  4216              333.          CLR.B (A6)
0'001256  301F              334.          MOVE.W (SP)+,D0
0'001258  0880 0003         335.          BCLR.L #3,D0     ,CLEAR THE BIT
0'00125C  0C00 0000         336.          CMPI.B #0,D0
0'001260  67 04             337.          BEQ.S REORDER
0'001262  6000 FEEC         338.          BRA TSTB0
                            339.  *
                            340.  * HAVE ALL DATA IN, NOW DO REARRANGE FROM THE
                            341.  * XDATAi ARRAY TO THE DATAII ARRAY FOR CORRECT
                            342.  * ORDER.
                            343.  *
0'001266  41FA F998         344.  REORDER LEA XDATA0(PC),A0
0'00126A  43FA F194         345.          LEA DATAII(PC),A1
0'00126E  7203              346.          MOVEQ.L #NPE-1,D1
0'001270  7A3F              347.  RLOOPT  MOVEQ.L #NPTE-1,D5
0'001272  2298              348   RLOOPN  MOVE.L (A0)+,(A1)
0'001274  D3CA              349.          ADDA L A2,A1
0'001276  51CD FFFA         350.          DBRA D5,RLOOPN
0'00127A  93CB              351.          SUBA L A3,A1
0'00127C  51C9 FFF2         352.          DBRA D1,RLOOPT
                            353   *
                            354.  * PROGRAM IS DONE, CAN STOP TIMER, AND
                            355   * OUTPUT NUMBER OF 32 CLOCK INTERVALS
                            356.  * PASSED. FIRST DISABLE TIMER
                            357   *
0'001280  4214              358.  SKIP     CLR B (A4)
0'001282  222C 0006         359           MOVE.L 6(A4),D1 ,GET NEW COUNT
0'001286  203C 00FFFFFF     360           MOVE.L #CNTSTRT,D0
0'00128C  9081              361           SUB.L D1,D0      ,GET ELAPSED COUNT
                            362.  *
                            363.  * OUTPUT CR, LF, THEN THE COUNT IN HEX
                            364.  *
0'00128E  4E4F              365.          TRAP #15
0'001290  000A              366.          DC W SENCLFC
0'001292  4E4F              367.          TRAP #15
0'001294  0018              368.          DC.W DISBUF8
0'001296  2F00              369.          MOVE.L D0,-(SP)
0'001298  41FA 0153         370.          LEA SPC(PC),A0
0'00129C  4E4F              371.          TRAP #15
0'00129E  0011              372.          DC.W OUTMESC
0'0012A0  2007              373.          MOVE.L D7,D0
0'0012A2  4E4F              374.          TRAP #15
0'0012A4  0018              375.          DC W DISBUF8
0'0012A6  201F              376.          MOVE.L (SP)+,D0
                            377.  *
                            378.  * NOW KEEP TRACK OF STATISTICS
                            379.  *
0'0012A8  D197              380.          ADD.L D0,(SP)    ;ADD TO SUM
0'0012AA  B0AF 0004         381.          CMP.L 4(SP),D0   ;CHECK MAX
0'0012AE  6F 04             382.          BLE.S NOTMAX     ;D0 NOT MAX
0'0012B0  2F40 0004         383.          MOVE.L D0,4(SP)  ,NEW MAX
0'0012B4  B0AF 0008         384.  NOTMAX  CMP.L 8(SP),D0   ;CHECK MIN
0'0012B8  6C 04             385.          BGE.S NOTMIN     ;D0 NOT MIN
0'0012BA  2F40 0008         386.          MOVE.L D0,8(SP)  ;NEW MIN
0'0012BE  51CF FDDA         387.  NOTMIN  DBRA D7,REEXE    ;DO D7 TIMES
                            388.
                            389.  *
                            390.  * WHEN HERE, PROGRAM HAS BEEN EXECUTED 256 TIMES
                            391.  * SEND AN ILLEGAL ADDRESS FOR DATA, FORCING
                            392.  * SLAVES TO PROGRAM ACCEPT MODE
                            393.  *
```

```
0'0012C2  41FA FD46       394.          LEA MBRDCST(PC),A0
0'0012C6  4E48            395.          TRAP #NETCNF
0'0012C8  103C 000F       396.          MOVE.B #ALLSLV,D0
0'0012CC  4E49            397.          TRAP #SRQSRVC
0'0012CE  207C FFFF0000   398.          MOVEA.L #$FFFF0000,A0   ;ILLEGAL ADDR
0'0012D4  13FC 0020       399.          MOVE.B #ENABLEB,PGCR
          00012FC0
0'0012DC  4E47            400.          TRAP #ADBYTOT   ;THAT'LL FIX 'EM
0'0012DE  4239 00012FC0   401.          CLR.B PGCR
                          402.  *
                          403.  * OUTPUT STATISTICS
                          404.  *
0'0012E4  4E4F            405.          TRAP #15
0'0012E6  000A            406.          DC.W SENCLFC
0'0012E8  4E4F            407.          TRAP #15
0'0012EA  000A            408.          DC.W SENCLFC
0'0012EC  41FA 008A       409.          LEA AVEMES(PC),A0
0'0012F0  4E4F            410.          TRAP #15
0'0012F2  0011            411.          DC.W OUTMESC
0'0012F4  201F            412.          MOVE.L (SP)+,D0 ,GET SUM
0'0012F6  E080            413.          ASR.L #DIV256,D0        ;DO AVERAGE
0'0012F8  4E4F            414.          TRAP #15
0'0012FA  0018            415.          DC.W DISBUF8    ;OUTPUT AVERAGE
0'0012FC  4E4F            416.          TRAP #15
0'0012FE  000A            417.          DC.W SENCLFC
0'001300  41FA 009D       418.          LEA MAXMES(PC),A0
0'001304  4E4F            419.          TRAP #15
0'001306  0011            420.          DC.W OUTMESC
0'001308  201F            421.          MOVE.L (SP)+,D0 ;GET MAX
0'00130A  4E4F            422.          TRAP #15
0'00130C  0018            423.          DC.W DISBUF8
0'00130E  4E4F            424.          TRAP #15
0'001310  000A            425.          DC.W SENCLFC
0'001312  41FA 00B2       426.          LEA MINMES(PC),A0
0'001316  4E4F            427.          TRAP #15
0'001318  0011            428.          DC.W OUTMESC
0'00131A  201F            429.          MOVE.L (SP)+,D0 ,GET MIN
0'00131C  4E4F            430.          TRAP #15
0'00131E  0018            431.          DC.W DISBUF8
0'001320  4E4F            432.          TRAP #15
0'001322  000A            433.          DC.W SENCLFC
0'001324  4E75            434.          RTS
                          435.  *
                          436.  * NOW FINALLY DONE!
                          437.  *
                          438.  *
                          439.  * SUBROUTINE SENDOUT
                          440.  *
0'001326  93C8            441.  SENDOUT SUBA.L A0,A1
0'001328  3009            442.          MOVE.W A1,D0
0'00132A  13FC 0020       443.          MOVE.B #ENABLEB,PGCR
          00012FC0
0'001332  4E47            444           TRAP #ADBYTOT
0'001334  4E46            445.          TRAP #OUTDATA
0'001336  4239 00012FC0   446.          CLR.B PGCR
0'00133C  4E75            447.          RTS
                          448.  *
                          449.  * SUBROUTINE TSTSLV
                          450.  * SLAVE NUMBER TO TEST IS IN D1.B
                          451.  * IF ASSERTING AN SRQ, THEN ACKNOWLEDGED AND
                          452.  * HIGH BIT IN D1.B IS SET, ELSE CLEAR
                          453.  *
0'00133E  48E7 8080       454.  TSTSLV  MOVEM.L D0/A0,-(SP)
0'001342  207C 00012FCC   455.          MOVEA.L #PCDR,A0
0'001348  0810 0000       456.          BTST.B #0,(A0)
0'00134C  66 22           457.          BNE.S NOINT
0'00134E  E509            458.          LSL.B #2,D1
0'001350  08C1 0004       459.          BSET.L #4,D1
0'001354  1081            460.          MOVE.B D1,(A0)
0'001356  0890 0004       461.          BCLR.B #4,(A0)
0'00135A  1010            462.          MOVE.B (A0),D0
0'00135C  08D0 0004       463.          BSET.B #4,(A0)
0'001360  0800 0001       464.          BTST.L #1,D0
0'001364  66 0A           465.          BNE.S NOINT
0'001366  123C 00FF       466.          MOVE.B #$FF,D1
0'00136A  4CDF 0101       467.          MOVEM.L (SP)+,D0/A0
0'00136E  4E75            468.          RTS
0'001370  4201            469.  NOINT   CLR.B D1
0'001372  4CDF 0101       470.          MOVEM.L (SP)+,D0/A0
0'001376  4E75            471.          RTS
                          472.  *
```

```
0'001378  41 76 65 72 61  473.  AVEMES  DC.B 'Average time ( x 4 for microseconds)= ',EOT
          67 65 20 74 69
          6D 65 20 28 20
          78 20 34 20 66
          6F 72 20 6D 69
          63 72 6F 73 65
          63 6F 6E 64 73
          29 3D 20 04
0'00139F  4D 61 78 69 6D  474.  MAXMES  DC.B 'Maximum time ( x 4 for microseconds)= ',EOT
          75 6D 20 74 69
          6D 65 20 28 20
          78 20 34 20 66
          6F 72 20 6D 69
          63 72 6F 73 65
          63 6F 6E 64 73
          29 3D 20 04
0'0013C6  4D 69 6E 69 6D  475.  MINMES  DC.B 'Minimum time ( x 4 for microseconds)= ',EOT
          75 6D 20 74 69
          6D 65 20 28 20
          78 20 34 20 66
          6F 72 20 6D 69
          63 72 6F 73 65
          63 6F 6E 64 73
          29 3D 20 04
0'0013ED  20 20 20 20 04  476   SPC     DC.B '    ',EOT
0'0013F2                  477.          END
    0 Errors
*
* LINK SPEC FOR FSTFL4NM
*
      LINK FSTFL4NM
      LINK FSTSLVNM
      ORG $1000
      SECTION 0,15
      END
```

M.12 Program FSTSLVNM: Multicomputer Frequency Domain Filtering Program: Slave programs for FSTFL?NM

```
                                    1.          TTL FSTSLVNM
                                    2.  *
                                    3.  * THIS CODE IS USED AS THE COMMON SLAVE
                                    4.  * CODE FOR ALL PARALLEL DIGITAL FILTERING
                                    5.  * TEST PROGRAMS.  LINK WITH THE MASTER
                                    6.  * CODE.
                                    7.  * NO CODE TO MEASURE EACH PHASE
                                    8.  * OF FORWARD FFT, FILTER, INVERSE FFT
                                    9.  *
                                   10.  * XDEF'S,XREF'S,EQU'S
                                   11.  *
F'000000                           12.          SECTION 15
                                   13.          XREF DATAII,FILTER
                                   14.          XREF PGCR,ENABLEA,ENABLEB
                                   15.          XREF NPT,NPTE,NPE
                                   16.          XREF SHIFT14,TCR,CNTSTRT,ENABLTM
                                   17.          XREF FORWARD,FLTR,REVERS
                                   18.          XREF XDATA0,XDATA1,XDATA2,XDATA3
                                   19.          XDEF S0PRO,S1PRO,S2PRO,S3PRO,DONE
                                   20.  *
     # 00000003                    21.  SRQASRT EQU 3
     # 00000005                    22.  INDATA  EQU 5
     # 00000006                    23.  OUTDATA EQU 6
     # 0000000E                    24.  ABORT   EQU 14
     # 00000006                    25.  NUGAM   EQU 6
                                   26.  *
                                   27.  ********************
                                   28.  * SLAVE 0 PROGRAM *
                                   29.  ********************
                                   30.  *
                                   31.  * INPUT DATA AT DATAII
                                   32.  * FILTER DATA AT FILTER 64 POINTS
                                   33.  * NOTE THAT SLAVE 0 HAS NO TWIDDLE FACTOR
                                   34.  * MULTIPLIES
                                   35.  *
                                   36.  *
F'000000  2A7C'00000000            37.  S0PRO   MOVEA.L #NPE,A5          ;ADDR INC
F'000006  2E3C'00000000            38.          MOVE.L #SHIFT14,D7       ;/16384 SHFT
F'00000C  303C'FFFF                39.          MOVE.W #NPTE-1,D0        ;MAJ LOOP CNTR
F'000010  2C3C'00000000            40.          MOVE.L #NPT,D6           ;INCREMENT
F'000016  287C'FFFFFFFC            41.          MOVEA.L #NPT-4,A4        ,INDX TO ARRAYS
F'00001C  41FA ****                42           LEA DATAII(PC),A0        ;XDATA POINTER
F'000020  4281                     43   TWDL0   CLR.L D1                 ;INIT SUMS
F'000022  2401                     44.          MOVE.L D1,D2
F'000024  2648                     45           MOVEA.L A0,A3
F'000026  7A'FF                    46.          MOVEQ.L #NPE-1,D5
F'000028  D7CC                     47.          ADDA.L A4,A3
F'00002A  3613                     48.  ADDL0   MOVE.W (A3),D3           ;GET REAL PART
F'00002C  48C3                     49.          EXT.L D3
F'00002E  D283                     50.          ADD.L D3,D1              ;REAL SUM D1
F'000030  362B 0002                51.          MOVE.W 2(A3),D3          ;GET IMAG PART
F'000034  48C3                     52.          EXT.L D3
F'000036  D483                     53.          ADD.L D3,D2              ;IMAG SUM D2
F'000038  D7C6                     54.          ADDA.L D6,A3             ;NEXT TERM
F'00003A  51CD FFEE                55.          DBRA D5,ADDL0
F'00003E  E481                     56.          ASR.L #2,D1              ;ADJUST SUMS
F'000040  E482                     57.          ASR.L #2,D2              ;ADJUST SUMS
                                   58.  *
                                   59.  * NO TWIDDLE FACTORS FOR SLAVE 0
                                   60.  * STORE SUMMED AND TWIDDLED DATA AT DATAII IN
                                   61.  * PLACE
                                   62.  *
F'000042  3181 C800                63.          MOVE.W D1,0(A0,A4.L)
F'000046  3182 C802                64.          MOVE.W D2,2(A0,A4.L)
F'00004A  99CD                     65.          SUBA.L A5,A4
F'00004C  51C8 FFD2                66.          DBRA D0,TWDL0
                                   67.  *
                                   68.  * INSERT FAVORITE 64-POINT FFT HERE
                                   69.  * INPUT DATA AT DATAII (0-255)
                                   70   *
F'000050  78'F7                    71.  USHFL0  MOVEQ.L #NPTE-9,D4          ;COUNTER=K
F'000052  3C04                     72.  SWAPL0  MOVE.W D4,D6              ;PREPARE BIT REV
F'000054  3E06                     73.  BREV20  MOVE.W D6,D7            ;SAVE DATA
F'000056  4246                     74.          CLR.W D6                ;
F'000058  7A05                     75           MOVEQ.L #5,D5           ;BIT COUNTER
F'00005A  E257                     76.  REV20   ROXR.W #1,D7            ;SFT RIGHT,XTEND
```

```
F'00005C  E356              77.          ROXL.W #1,D6
F'00005E  51CD FFFA         78.          DBRA D5,REV20
F'000062  B846              79.          CMP W D6,D4          ,I=REV(K)<=K?
F'000064  6C 14             80.          BGE.S DECK0           ;NO SWAP IF SO
F'000066  3204              81.          MOVE.W D4,D1         ,COPY OF K
F'000068  E541              82.          ASL.W #2,D1          ,K*4
F'00006A  E546              83.          ASL.W #2,D6          ,I*4
F'00006C  2430 1000         84.          MOVE.L 0(A0,D1.W),D2 ,DO SWAP
F'000070  21B0 6000 1000    85.          MOVE.L 0(A0,D6.W),0(A0,D1.W)
F'000076  2182 6000         86.          MOVE.L D2,0(A0,D6.W) ;DONE SWAP
F'00007A  51CC FFD6         87.  DECK0   DBRA D4,SWAPL0
F'00007E  7201              88.  FFT0    MOVEQ.L #1,D1    ;D1=N2=1
F'000080  43FA 0270         89.          LEA SINBLK0(PC),A1   ;SIN VALS PNTR A1
F'000084  45FA 02EC         90.          LEA COSBLK0(PC),A2   ;COS VALS PNTR A2
                            91.  *
                            92.  * L IS DOWN COUNTED #NUGAM-1 TO 0
                            93.  *
F'000088  3F3C 0001         94.          MOVE.W #1,-(SP)   ,(SP) IS L
F'00008C  7405              95.          MOVEQ.L #NUGAM-1,D2   ,D2=NUGAM-1=NU1
F'00008E  4283              96.          CLR.L D3              ,D3=K=0
F'000090  70'00             97.          MOVEQ.L #SHIFT14,D0
                            98.  *
                            99.  * INITIALIZE I=N2-1, TO 0 IN STEPS -1
                           100.  *
F'000092  3801             101.  INITI0  MOVE.W D1,D4     ;D4=I=N2
F'000094  5344             102.          SUBQ.W #1,D4     ,SET FOR DBRA
                           103.  *
                           104.  * DETERMINE P=MOD(K*2**(NUGAM-L),64)
                           105.  *
                           106.  *
F'000096  3C03             107.  LOOPIN0 MOVE.W D3,D6     ;D6=K
F'000098  3A17             108.          MOVE.W (SP),D5   ,GET L
F'00009A  5D45             109           SUBQ.W #NUGAM,D5 ,L-NUGAM
F'00009C  4445             110.          NEG W D5         ,NUGAM-L
F'00009E  EB66             111.          ASL.W D5,D6      ,K*2**(NUGAM-L)
F'0000A0  0246 003F        112           ANDI.W #$003F,D6 ,MOD 64
                           113.  *
                           114.  * D6 HAS INDEX REQUIRED
F'0000A4  DC46             115.          ADD.W D6,D6      ;D6*2 FOR WORD BOUND
F'0000A6  3A71 6000        116.          MOVEA.W 0(A1,D6.W),A5 ,A5 PNTR IM(W**P)
F'0000AA  2E03             117.          MOVE.L D3,D7     ,D7=K
F'0000AC  2A03             118.          MOVE.L D3,D5     ;D5 TOO
F'0000AE  E545             119.          ASL.W #2,D5      ,K*4 FOR LWORD BOUND
F'0000B0  4DF0 5000        120.          LEA 0(A0,D5.W),A6    ;A6 IS INDEX X(K)
F'0000B4  DE41             121.          ADD.W D1,D7      ;D7=K+N2
F'0000B6  E547             122.          ASL.W #2,D7      ,D7=4(K+N2) LWORD BND
F'0000B8  47F0 7000        123           LEA 0(A0,D7.W),A3 ;A3 PNT RE(X(K+N2))
F'0000BC  3A32 6000        124.          MOVE.W 0(A2,D6.W),D5     ,GET RE(W**P)
F'0000C0  3E05             125.          MOVE.W D5,D7     ,PUT ALSO IN D7
F'0000C2  CBD3             126.          MULS (A3),D5     ,RE(W**P)*RE(X(K+N2))
F'0000C4  3C0D             127.          MOVE.W A5,D6  ;GET IM(W**P)*
F'0000C6  CDEB 0002        128.          MULS 2(A3),D6    ;IM(W**P)*IM(X(K+N2))
F'0000CA  9A86             129.          SUB.L D6,D5      ,D5=RE(T1)*
F'0000CC  E0A5             130.          ASR.L D0,D5      ,/16384 TO SCALE
F'0000CE  3C0D             131.          MOVE.W A5,D6  ,GET IM(W**P)
F'0000D0  CDD3             132.          MULS (A3),D6     ;IM(W**P)*RE(X(K+N2))
F'0000D2  CFEB 0002        133.          MULS 2(A3),D7    ;RE(W**P)*IM(X(K+N2))
F'0000D6  DC87             134.          ADD.L D7,D6      ;D6=IM(T1)
F'0000D8  E0A6             135.          ASR.L D0,D6      ;/16384 TO SCALE
F'0000DA  3E16             136.          MOVE.W (A6),D7   ;RE(X(K))
F'0000DC  9E45             137.          SUB.W D5,D7      ;RE(X(K))-RE(T1)
F'0000DE  E247             138.          ASR.W #1,D7      ;/2
F'0000E0  3687             139.          MOVE.W D7,(A3)   ;STORE ANSWER
F'0000E2  3E2E 0002        140.          MOVE.W 2(A6),D7  ;IM(X(K))
F'0000E6  9E46             141.          SUB.W D6,D7      ;IM(X(K))-IM(T1)
F'0000E8  E247             142.          ASR.W #1,D7      ;/2
F'0000EA  3747 0002        143.          MOVE.W D7,2(A3)  ;STORE ANSWER
F'0000EE  DB56             144.          ADD.W D5,(A6)    ;RE(X(K))+RE(T1)
F'0000F0  E0D6             145.          ASR.W (A6)       ;/2
F'0000F2  DD6E 0002        146.          ADD.W D6,2(A6)   ;IM(X(K))+IM(T1)
F'0000F6  E0EE 0002        147.          ASR.W 2(A6)      ;/2
F'0000FA  5243             148.          ADDQ.W #1,D3     ;INCREMENT K
F'0000FC  51CC FF98        149.          DBRA D4,LOOPIN0  ;DO TILL I=-1
F'000100  D641             150.          ADD.W D1,D3      ;K=K+N2
F'000102  0C43'FFFF        151.          CMPI.W #NPTE-1,D3 ;K<N-1 ?
F'000106  6D 8A            152.          BLT INITI0       ;DO AGAIN IF SO
F'000108  4243             153.          CLR.W D3         ;K=0
F'00010A  5342             154.          SUBQ.W #1,D2     ;NU1=NU1-1
F'00010C  E341             155.          ASL.W #1,D1      ;N2=N2*2
F'00010E  5257             156.          ADDQ.W #1,(SP)
F'000110  0C57 0007        157.          CMPI.W #NUGAM+1,(SP)
```

```
F'000114  6600 FF7C       158.          BNE INITI0
F'000118  301F            159.          MOVE.W (SP)+,D0 ;CLEAN STACK
                          160.  *
                          161.  * NOW DO FILTERING, FILTER DATA IS AT "FILTER"
                          162.  * 64 POINTS
                          163.  *
                          164.  * DO COMPLEX MULTIPLY BY THE FILTER.
                          165.  * FILTER CAN BE +16384 TO -16384 WITH 16384
                          166.  * BEING THE 0 db GAIN VALUE.
                          167.  * (A+iB)*(C+iD)
                          168.  *
                          169.  * PUT RESULTS AT XDATA0
                          170.  *
F'00011A  7C04            171.          MOVEQ.L #4,D6           ;ADDRESS INC
F'00011C  45FA ****       172.          LEA XDATA0(PC),A2       ;TO XDATA0
F'000120  43FA ****       173.          LEA FILTER(PC),A1       ;A1 PNT FILTER
F'000124  303C'FFFF       174.          MOVE.W #NPTE-1,D0       ;LOOP COUNTER
F'000128  7A'00           175.          MOVEQ.L #SHIFT14,D5     ;SHIFT COUNT
F'00012A  3219            176.  FILOOP0 MOVE.W (A1)+,D1         ;GET C
F'00012C  3601            177.          MOVE.W D1,D3
F'00012E  C3D8            178.          MULS (A0)+,D1           ;DO AC
F'000130  3419            179.          MOVE.W (A1)+,D2         ;GET D
F'000132  3802            180.          MOVE.W D2,D4
F'000134  C5D0            181.          MULS (A0),D2            ;DO BD
F'000136  9282            182.          SUB.L D2,D1             ;AC-BD
F'000138  EAA1            183.          ASR.L D5,D1             ;ADJUST /16384
F'00013A  C7D0            184.          MULS (A0),D3            ;DO BC
F'00013C  C9E0            185.          MULS -(A0),D4           ;DO AD
F'00013E  D883            186.          ADD.L D3,D4             ;BC+AD
F'000140  EAA4            187.          ASR.L D5,D4             ;ADJUST /16384
F'000142  34C1            188.          MOVE.W D1,(A2)+ ;STORE REAL
F'000144  34C4            189.          MOVE.W D4,(A2)+ ;STORE IMAG
F'000146  D1C6            190.          ADDA.L D6,A0
F'000148  51C8 FFE0       191.          DBRA D0,FILOOP0
                          192.  *
                          193.  * FILTERING DONE
                          194.  *
                          195.  * ASSERT SRQ AND WAIT FOR ACKNOWLEDGE
                          196.  *
F'00014C  4E43            197.          TRAP #SRQASRT
                          198.  * SEND OUT 64 COMPLEX POINTS CALCULATED
>>> WARNING line[199] Expr type   [199] FSTSLVNM.A68
F'00014E  303C ****       199.          MOVE.W #NPTE*4,D0       ;BYTE COUNT
F'000152  41FA ****       200.          LEA XDATA0(PC),A0
F'000156  13FC 00'00      201.          MOVE.B #ENABLEB,PGCR
          '00000000
F'00015E  4E46            202.          TRAP #OUTDATA
F'000160  4239'00000000   203.          CLR.B PGCR
                          204.  *
                          205.  * ASSERT AN SRQ TO SIGNAL READY FOR DATA
                          206.  * FROM SLAVE #1 AND RECEIVE; PUT AT XDATA1
                          207.  * NUMBER OF POINTS IS STILL IN D0
                          208.  *
F'000166  4E43            209.          TRAP #SRQASRT
                          210.  *       MOVE.W #NPTE*4,D0       ;BYTE COUNT
F'000168  41FA ****       211.          LEA XDATA1(PC),A0
F'00016C  13FC 00'00      212.          MOVE.B #ENABLEA,PGCR
          '00000000
F'000174  4E45            213.          TRAP #INDATA
F'000176  4239'00000000   214.          CLR.B PGCR
                          215.  *
                          216.  * ASSERT AN SRQ TO SIGNAL READY FOR DATA
                          217.  * FROM SLAVE #2 AND RECEIVE; PUT AT XDATA2
                          218.  * NUMBER OF POINTS IS STILL IN D0
                          219.  *
F'00017C  4E43            220.          TRAP #SRQASRT
                          221.  *       MOVE.W #NPTE*4,D0       ;BYTE COUNT
F'00017E  41FA ****       222.          LEA XDATA2(PC),A0
F'000182  13FC 00'00      223.          MOVE.B #ENABLEA,PGCR
          '00000000
F'00018A  4E45            224.          TRAP #INDATA
F'00018C  4239'00000000   225           CLR.B PGCR
                          2.    .
                          227.  * ASSERT AN SRQ TO SIGNAL READY FOR DATA
                          228.  * FROM SLAVE #3 AND RECEIVE; PUT AT XDATA3
                          229.  * NUMBER OF POINTS IS STILL IN D0
                          230.  *
F'000192  4E43            231.          TRAP #SRQASRT
                          232.  *       MOVE.W #NPTE*4,D0       ;BYTE COUNT
F'000194  41FA ****       233.          LEA XDATA3(PC),A0
F'000198  13FC 00'00      234.          MOVE.B #ENABLEA,PGCR
```

```
         '00000000
F'0001A0  4E45              235.          TRAP #INDATA
F'0001A2  4239'00000000     236.          CLR.B PGCR
                            237.  *
                            238.  * ASSERT AN SRQ TO SIGNIFY DONE
                            239.  * DATA REDISTRUBUTION
                            240   *
F'0001A8  4E43              241.          TRAP #SRQASRT
                            242.  *
                            243.  * NOW MUST DO DATA REORDERING
                            244.  * A 4-SHUFFLE FROM XDATA0-3 TO DATAII
                            245.  * TO PREPARE FOR INVERSE TRANSFORMING
                            246.  *
F'0001AA  267C 000003FC     247.  REORD0  MOVEA L #$3FC,A3
F'0001B0  247C 00000010     248.          MOVEA.L #$10,A2
F'0001B6  41FA ****         249.          LEA XDATA0(PC),A0
F'0001BA  43FA ****         250.          LEA DATAII(PC),A1
F'0001BE  72'FF             251.          MOVEQ.L #NPE-1,D1
F'0001C0  7A'FF             252.  RLOPT0  MOVEQ L #NPTE-1,D5
F'0001C2  2298              253   RLOPN0  MOVE.L (A0)+,(A1)
F'0001C4  D3CA              254.          ADDA.L A2,A1
F'0001C6  51CD FFFA         255.          DBRA D5,RLOPN0
F'0001CA  93CB              256           SUBA.L A3,A1
F'0001CC  51C9 FFF2         257.          DBRA D1,RLOPT0
                            258.  *
                            259.  * NOW DO THE INVERSE TRANSFORM
                            260.  * USE SAME SIN AND COS TABLES, NEGATE THE SIN
                            261.  * ENTRY, AND USE NO ASR #2 FOR DIVISION BY NPTE
                            262   * NOTE SIGN CHANGE ON 4-ADDITIONS TOO
                            263   *
F'0001D0  2A7C'00000000     264           MOVEA L #NPE,A5          ,ADDR INC
F'0001D6  2E3C'00000000     265           MOVE.L #SHIFT14,D7       ,/16384 SHFT
F'0001DC  303C'FFFF         266.          MOVE.W #NPTE-1,D0        ;MAJ LOOP CNTR
F'0001E0  2C3C'00000000     267           MOVE.L #NPT,D6           ;INCREMENT
F'0001E6  287C'FFFFFFFC     268           MOVEA.L #NPT-4,A4        ,INDX TO ARRAYS
F'0001EC  41FA ****         269           LEA DATAII(PC),A0        ;XDATA POINTER
F'0001F0  4281              270   TWDL0R  CLR L D1                 ;INIT SUMS
F'0001F2  2401              271.          MOVE L D1,D2
F'0001F4  2648              272           MOVEA L A0,A3
F'0001F6  7A'FF             273.          MOVEQ L #NPE-1,D5
F'0001F8  D7CC              274           ADDA.L A4,A3
F'0001FA  3613              275.  ADDL0R  MOVE.W (A3),D3           ,GET REAL PART
F'0001FC  D243              276           ADD W D3,D1              ,REAL SUM D1
F'0001FE  362B 0002         277           MOVE W 2(A3),D3          ,GET IMAG PART
F'000202  D443              278.          ADD W D3,D2              ,IMAG SUM D2
F'000204  D7C6              279           ADDA.L D6,A3             ,NEXT TERM
F'000206  51CD FFF2         280           DBRA D5,ADDL0R
                            281   *
                            282.  * NO TWIDDLE FACTORS FOR SLAVE 0
                            283.  * STORE SUMMED AND TWIDDLED DATA AT DATAII IN
                            284   * PLACE
                            285.  *
F'00020A  3181 C800         286.          MOVE.W D1,0(A0,A4 L)
F'00020E  3182 C802         287.          MOVE.W D2,2(A0,A4 L)
F'000212  99CD              288.          SUBA.L A5,A4
F'000214  51C8 FFDA         289.          DBRA D0,TWDL0R
                            290.  *
                            291.  * INSERT FAVORITE 64-POINT FFT HERE
                            292.  * INPUT DATA AT DATAII (0-255)
                            293.  *
F'000218  78'F7             294.  USHFL0R MOVEQ.L #NPTE-9,D4        ,COUNTER=K
F'00021A  3C04              295.  SWAPL0R MOVE.W D4,D6          ,PREPARE BIT REV
F'00021C  3E06              296.  BREV20R MOVE.W D6,D7        ;SAVE DATA
F'00021E  4246              297.          CLR.W D6             ,
F'000220  7A05              298.          MOVEQ.L #5,D5        ,BIT COUNTER
F'000222  E257              299.  REV20R  ROXR.W #1,D7         ;SFT RIGHT,XTEND
F'000224  E356              300.          ROXL.W #1,D6
F'000226  51CD FFFA         301.          DBRA D5,REV20R
F'00022A  B846              302.          CMP.W D6,D4            ;I=REV(K)<=K?
F'00022C  6C 14             303.          BGE.S DECK0R             ;NO SWAP IF SO
F'00022E  3204              304.          MOVE.W D4,D1           ;COPY OF K
F'000230  E541              305.          ASL.W #2,D1            ;K*4
F'000232  E546              306.          ASL.W #2,D6            ,I*4
F'000234  2430 1000         307.          MOVE.L 0(A0,D1.W),D2 ,DO SWAP
F'000238  21B0 6000 1000    308.          MOVE.L 0(A0,D6.W),0(A0,D1.W)
F'00023E  2182 6000         309.          MOVE.L D2,0(A0,D6.W) ;DONE SWAP
F'000242  51CC FFD6         310.  DECK0R  DBRA D4,SWAPL0R
F'000246  7201              311.  FFT0R   MOVEQ L #1,D1    ;D1=N2=1
F'000248  43FA 00A8         312.          LEA SINBLK0(PC),A1  ;SIN VALS PNTR A1
F'00024C  45FA 0124         313.          LEA COSBLK0(PC),A2  ;COS VALS PNTR A2
                            314.  *
```

```
                                315.  * L IS DOWN COUNTED #NUGAM-1 TO 0
                                316.  *
F'000250  3F3C 0001             317.          MOVE.W #1,-(SP)  ;(SP) IS L
F'000254  7405                  318.          MOVEQ.L #NUGAM-1,D2 ;D2=NUGAM-1=NU1
F'000256  4283                  319.          CLR.L D3            ;D3=K=0
F'000258  70'00                 320.          MOVEQ.L #SHIFT14,D0
                                321.  *
                                322.  * INITIALIZE I=N2-1, TO 0 IN STEPS -1
                                323.  *
F'00025A  3801                  324.  INITIOR MOVE.W D1,D4     ;D4=I=N2
F'00025C  5344                  325.          SUBQ.W #1,D4     ;SET FOR DBRA
                                326.  *
                                327.  * DETERMINE P=MOD(K*2**(NUGAM-L),64)
                                328.  *
                                329.  *
F'00025E  3C03                  330.  LPINOR  MOVE.W D3,D6     ;D6=K
F'000260  3A17                  331.          MOVE.W (SP),D5   ;GET L
F'000262  5D45                  332.          SUBQ.W #NUGAM,D5 ;L-NUGAM
F'000264  4445                  333.          NEG.W D5         ;NUGAM-L
F'000266  EB66                  334.          ASL.W D5,D6      ;K*2**(NUGAM-L)
F'000268  0246 003F             335.          ANDI.W #$003F,D6 ;MOD 64
                                336.  *
                                337.  * D6 HAS INDEX REQUIRED
F'00026C  DC46                  338.          ADD.W D6,D6      ;D6*2 FOR WORD BOUND
F'00026E  3A71 6000             339.          MOVEA.W 0(A1,D6.W),A5 ;A5 PNTR IM(W**P)
F'000272  2E03                  340.          MOVE.L D3,D7     ;D7=K
F'000274  2A03                  341.          MOVE.L D3,D5     ;D5 TOO
F'000276  E545                  342.          ASL.W #2,D5      ;K*4 FOR LWORD BOUND
F'000278  4DF0 5000             343.          LEA 0(A0,D5.W),A6   ;A6 IS INDEX X(K)
F'00027C  DE41                  344.          ADD.W D1,D7      ;D7=K+N2
F'00027E  E547                  345.          ASL.W #2,D7      ;D7=4(K+N2) LWORD BND
F'000280  47F0 7000             346.          LEA 0(A0,D7.W),A3 ;A3 PNT RE(X(K+N2))
F'000284  3A32 6000             347.          MOVE.W 0(A2,D6.W),D5    ;GET RE(W**P)
F'000288  3E05                  348.          MOVE.W D5,D7     ;PUT ALSO IN D7
F'00028A  CBD3                  349.          MULS (A3),D5     ;RE(W**P)*RE(X(K+N2))
F'00028C  3C0D                  350.          MOVE.W A5,D6  ;GET IM(W**P)*
F'00028E  4446                  351.          NEG.W D6
F'000290  CDEB 0002             352.          MULS 2(A3),D6    ;IM(W**P)*IM(X(K+N2))
F'000294  9A86                  353.          SUB.L D6,D5      ;D5=RE(T1)*
F'000296  E0A5                  354.          ASR.L D0,D5      ;/16384 TO SCALE
F'000298  3C0D                  355.          MOVE.W A5,D6  ;GET IM(W**P)
F'00029A  4446                  356.          NEG.W D6
F'00029C  CDD3                  357.          MULS (A3),D6     ;IM(W**P)*RE(X(K+N2))
F'00029E  CFEB 0002             358.          MULS 2(A3),D7    ;RE(W**P)*IM(X(K+N2))
F'0002A2  DC87                  359.          ADD.L D7,D6      ;D6=IM(T1)
F'0002A4  E0A6                  360.          ASR.L D0,D6      ;/16384 TO SCALE
F'0002A6  3E16                  361.          MOVE.W (A6),D7   ;RE(X(K))
F'0002A8  9E45                  362.          SUB.W D5,D7      ;RE(X(K))-RE(T1)
F'0002AA  3687                  363.          MOVE.W D7,(A3)   ;STORE ANSWER
F'0002AC  3E2E 0002             364.          MOVE.W 2(A6),D7 ;IM(X(K))
F'0002B0  9E46                  365.          SUB.W D6,D7      ;IM(X(K))-IM(T1)
F'0002B2  3747 0002             366.          MOVE.W D7,2(A3) ;STORE ANSWER
F'0002B6  DB56                  367.          ADD.W D5,(A6)    ;RE(X(K))+RE(T1)
F'0002B8  DD6E 0002             368.          ADD.W D6,2(A6)   ;IM(X(K))+IM(T1)
F'0002BC  5243                  369.          ADDQ.W #1,D3     ;INCREMENT K
F'0002BE  51CC FF9E             370.          DBRA D4,LPINOR   ;DO TILL I=-1
F'0002C2  D641                  371.          ADD.W D1,D3      ;K=K+N2
F'0002C4  0C43'FFFF             372.          CMPI.W #NPTE-1,D3 ;K<N-1 ?
F'0002C8  6D 90                 373.          BLT INITIOR      ;DO AGAIN IF SO
F'0002CA  4243                  374.          CLR.W D3         ;K=0
F'0002CC  5342                  375.          SUBQ.W #1,D2     ;NU1=NU1-1
F'0002CE  E341                  376.          ASL.W #1,D1      ;N2=N2*2
F'0002D0  5257                  377.          ADDQ.W #1,(SP)
F'0002D2  0C57 0007             378.          CMPI.W #NUGAM+1,(SP)
F'0002D6  66 82                 379.          BNE INITIOR
F'0002D8  301F                  380.          MOVE.W (SP)+,D0 ;CLEAN STACK
                                381.  *
                                382.  * ASSERT AN SRQ AND UPLOAD THE DATA
                                383.  *
F'0002DA  4E43                  384.          TRAP #SRQASRT
>>> WARNING line[385] Expr. type   [385] FSTSLVNM.A68
F'0002DC  303C ****             385.          MOVE.W #NPTE*4,D0       ;BYTE COUNT
F'0002E0  13FC 00'00            386.          MOVE.B #ENABLEB,PGCR
          '00000000
F'0002E8  4E46                  387.          TRAP #OUTDATA
F'0002EA  4239'00000000         388.          CLR.B PGCR
                                389.  *
                                390.  * NOW DO RTS AND RETURN TO READY FOR DATA
                                391.  * STATE TO DO OVER!
                                392.  *
F'0002F0  4E75                  393.          RTS
```

```
                                    394.  *
                                    395.  * DONE!
                                    396.  *
                                    397.  * SIN AND COS TABLES HERE
                                    398.  *
                                    399.  *
                                    400.  * SIN (-2*PI*I/64)*16384 FOR I=0 TO 63
                                    401.  *
F'0002F2  0000 F9BA F384 ED6C       402.  SINBLK0 DC.W       0, -1606, -3196, -4756
F'0002FA  E782 E1D5 DC72 D766       403.          DC.W   -6270, -7723, -9102,-10394
F'000302  D2BF CE87 CAC9 C78F       404.          DC.W  -11585,-12665,-13623,-14449
F'00030A  C4DF C2C1 C13B C04F       405.          DC.W  -15137,-15679,-16069,-16305
F'000312  C000 C04F C13B C2C1       406.          DC.W  -16384,-16305,-16069,-15679
F'00031A  C4DF C78F CAC9 CE87       407.          DC.W  -15137,-14449,-13623,-12665
F'000322  D2BF D766 DC72 E1D5       408.          DC.W  -11585,-10394, -9102, -7723
F'00032A  E782 ED6C F384 F9BA       409.          DC.W   -6270, -4756, -3196, -1606
F'000332  0000 0646 0C7C 1294       410.          DC.W       0,  1606,  3196,  4756
F'00033A  187E 1E2B 238E 289A       411.          DC.W    6270,  7723,  9102, 10394
F'000342  2D41 3179 3537 3871       412.          DC.W   11585, 12665, 13623, 14449
F'00034A  3B21 3D3F 3EC5 3FB1       413.          DC.W   15137, 15679, 16069, 16305
F'000352  4000 3FB1 3EC5 3D3F       414.          DC.W   16384, 16305, 16069, 15679
F'00035A  3B21 3871 3537 3179       415.          DC.W   15137, 14449, 13623, 12665
F'000362  2D41 289A 238E 1E2B       416           DC.W   11585, 10394,  9102,  7723
F'00036A  187E 1294 0C7C 0646       417.          DC.W    6270,  4756,  3196,  1606
                                    418.  *
                                    419   * COS(-2*PI*I/64)*16384 FOR I=0 TO 63
                                    420.  *
F'000372  4000 3FB1 3EC5 3D3F       421.  COSBLK0 DC.W   16384, 16305, 16069, 15679
F'00037A  3B21 3871 3537 3179       422.          DC W   15137, 14449, 13623, 12665
F'000382  2D41 289A 238E 1E2B       423.          DC.W   11585, 10394,  9102,  7723
F'00038A  187E 1294 0C7C 0646       424.          DC.W    6270,  4756,  3196,  1606
F'000392  0000 F9BA F384 ED6C       425           DC.W       0, -1606, -3196, -4756
F'00039A  E782 E1D5 DC72 D766       426.          DC.W   -6270, -7723, -9102,-10394
F'0003A2  D2BF CE87 CAC9 C78F       427.          DC.W  -11585,-12665,-13623,-14449
F'0003AA  C4DF C2C1 C13B C04F       428.          DC.W  -15137,-15679,-16069,-16305
F'0003B2  C000 C04F C13B C2C1       429.          DC.W  -16384,-16305,-16069,-15679
F'0003BA  C4DF C78F CAC9 CE87       430.          DC.W  -15137,-14449,-13623,-12665
F'0003C2  D2BF D766 DC72 E1D5       431.          DC.W  -11585,-10394, -9102, -7723
F'0003CA  E782 ED6C F384 F9BA       432.          DC.W   -6270, -4756, -3196, -1606
F'0003D2  0000 0646 0C7C 1294       433.          DC.W       0,  1606,  3196,  4756
F'0003DA  187E 1E2B 238E 289A       434           DC.W    6270,  7723,  9102, 10394
F'0003E2  2D41 3179 3537 3871       435.          DC.W   11585, 12665, 13623, 14449
F'0003EA  3B21 3D3F 3EC5 3FB1       436.          DC.W   15137, 15679, 16069, 16305
                                    437.
                                    438.  ********************
                                    439.  * SLAVE 1 PROGRAM *
                                    440.  ********************
                                    441   *
                                    442.  * INPUT DATA AT DATAII, TWIDDLE FACTORS AT
```

```
                                443.  * TWD1
                                444.  *
                                445.  *
F'0003F2  2A7C'00000000         446.  S1PRO   MOVEA.L #NPE,A5          ;ADDR INC
F'0003F8  2E3C'00000000         447.          MOVE.L #SHIFT14,D7       ;/16384 SHFT
F'0003FE  303C'FFFF             448.          MOVE.W #NPTE-1,D0        ;MAJ LOOP CNTR
F'000402  287C'FFFFFFFC         449.          MOVEA.L #NPT-4,A4        ;INDX TO ARRAYS
F'000408  43FA 0444             450.          LEA TWD1(PC),A1          ;TWID FACT PNTR
F'00040C  41FA ****             451.          LEA DATAII(PC),A0        ;XDATA POINTER
F'000410  2648                  452.  TWDL1   MOVEA.L A0,A3
F'000412  D7CC                  453.          ADDA.L A4,A3
F'000414  3213                  454.  ADDL1   MOVE.W (A3),D1           ;GET REAL PART
F'000416  48C1                  455.          EXT.L D1                 ;REAL SUM IN D1
F'000418  362B 0102             456.          MOVE.W 258(A3),D3
F'00041C  48C3                  457.          EXT.L D3
F'00041E  D283                  458           ADD.L D3,D1
F'000420  362B 0200             459.          MOVE.W 512(A3),D3
F'000424  48C3                  460.          EXT.L D3
F'000426  9283                  461.          SUB.L D3,D1
F'000428  362B 0302             462.          MOVE.W 770(A3),D3
F'00042C  48C3                  463.          EXT.L D3
F'00042E  9283                  464.          SUB.L D3,D1              ;REAL SUM DONE
F'000430  342B 0002             465.          MOVE.W 2(A3),D2          ;IMAG SUM IN D2
F'000434  48C2                  466.          EXT.L D2
F'000436  362B 0100             467.          MOVE.W 256(A3),D3
F'00043A  48C3                  468.          EXT.L D3
F'00043C  9483                  469.          SUB.L D3,D2
F'00043E  362B 0202             470.          MOVE.W 514(A3),D3
F'000442  48C3                  471.          EXT.L D3
F'000444  9483                  472.          SUB.L D3,D2
F'000446  362B 0300             473.          MOVE.W 768(A3),D3
F'00044A  48C3                  474.          EXT.L D3
F'00044C  D483                  475.          ADD.L D3,D2              ;IMAG SUM DONE
F'00044E  E481                  476.          ASR.L #2,D1
F'000450  E482                  477.          ASR.L #2,D2              ;ADJUST SUMS
                                478.  *
                                479.  * NOW DO TWIDDLE FACTOR MULTIPLY
                                480.  *
F'000452  3A01                  481.          MOVE.W D1,D5
F'000454  3831 C800             482.          MOVE.W 0(A1,A4.L),D4
F'000458  3631 C802             483.          MOVE.W 2(A1,A4.L),D3
F'00045C  CBC4                  484.          MULS D4,D5
F'00045E  3C02                  485.          MOVE.W D2,D6
F'000460  CDC3                  486.          MULS D3,D6
F'000462  9A86                  487.          SUB.L D6,D5              ;REAL IN D5
F'000464  C3C3                  488.          MULS D3,D1
F'000466  C5C4                  489.          MULS D4,D2
F'000468  D282                  490.          ADD.L D2,D1              ;IMAG IN D1
F'00046A  EEA5                  491.          ASR.L D7,D5
F'00046C  EEA1                  492.          ASR.L D7,D1
                                493.  *
                                494.  * STORE SUMMED AND TWIDDLED DATA AT DATAII IN
                                495.  * PLACE
                                496.  *
F'00046E  3185 C800             497.          MOVE.W D5,0(A0,A4.L)
F'000472  3181 C802             498.          MOVE.W D1,2(A0,A4.L)
F'000476  99CD                  499.          SUBA.L A5,A4
F'000478  51C8 FF96             500.          DBRA D0,TWDL1
                                501.  *
                                502.  * INSERT FAVORITE 64-POINT FFT HERE
                                503.  * INPUT DATA AT DATAII (0-255)
                                504.  *
F'00047C  78'F7                 505.  USHFL1  MOVEQ.L #NPTE-9,D4        ;COUNTER=K
F'00047E  3C04                  506.  SWAPL1  MOVE.W D4,D6          ;PREPARE BIT REV
F'000480  3E06                  507.  BREV21  MOVE.W D6,D7        ;SAVE DATA
F'000482  4246                  508.          CLR.W D6            ;
F'000484  7A05                  509.          MOVEQ.L #5,D5       ;BIT COUNTER
F'000486  E257                  510.  REV21   ROXR.W #1,D7        ;SFT RIGHT,XTEND
F'000488  E356                  511.          ROXL.W #1,D6
F'00048A  51CD FFFA             512.          DBRA D5,REV21
F'00048E  B846                  513.          CMP.W D6,D4           ;I=REV(K)<=K?
F'000490  6C 14                 514.          BGE.S DECK1            ;NO SWAP IF SO
F'000492  3204                  515.          MOVE.W D4,D1          ;COPY OF K
F'000494  E541                  516.          ASL.W #2,D1           ;K*4
F'000496  E546                  517.          ASL.W #2,D6           ;I*4
F'000498  2430 1000             518.          MOVE.L 0(A0,D1.W),D2 ;DO SWAP
F'00049C  21B0 6000 1000        519.          MOVE.L 0(A0,D6.W),0(A0,D1.W)
F'0004A2  2182 6000             520.          MOVE.L D2,0(A0,D6.W) ;DONE SWAP
F'0004A6  51CC FFD6             521.  DECK1   DBRA D4,SWAPL1
```

```
F'0004AA  7201          522.  FFT641  MOVEQ.L #1,D1   ;D1=N2=1
F'0004AC  43FA 02A0     523.          LEA SINBLK1(PC),A1  ;SIN VALS PNTR A1
F'0004B0  45FA 031C     524.          LEA COSBLK1(PC),A2  ;COS VALS PNTR A2
                        525.  *
                        526.  * L IS DOWN COUNTED #NUGAM-1 TO 0
                        527.  *
F'0004B4  3F3C 0001     528.          MOVE.W #1,-(SP)  ;(SP) IS L
F'0004B8  7405          529.          MOVEQ.L #NUGAM-1,D2  ;D2=NUGAM-1=NU1
F'0004BA  4283          530.          CLR.L D3             ;D3=K=0
F'0004BC  70'00         531.          MOVEQ.L #SHIFT14,D0
                        532.  *
                        533.  * INITIALIZE I=N2-1, TO 0 IN STEPS -1
                        534.  *
F'0004BE  3801          535.  INITI1  MOVE.W D1,D4    ;D4=I=N2
F'0004C0  5344          536.          SUBQ.W #1,D4    ;SET FOR DBRA
                        537.  *
                        538.  * DETERMINE P=MOD(K*2**(NUGAM-L),64)
                        539.  *
                        540.  *
F'0004C2  3C03          541.  LOOPIN1 MOVE.W D3,D6    ;D6=K
F'0004C4  3A17          542.          MOVE.W (SP),D5  ,GET L
F'0004C6  5D45          543.          SUBQ.W #NUGAM,D5 ;L-NUGAM
F'0004C8  4445          544.          NEG.W D5         ;NUGAM-L
F'0004CA  EB66          545.          ASL.W D5,D6      ;K*2**(NUGAM-L)
F'0004CC  0246 003F     546.          ANDI.W #$003F,D6 ;MOD 64
                        547.  *
                        548.  * D6 HAS INDEX REQUIRED
                        549.
F'0004D0  DC46          550.          ADD.W D6,D6      ,D6*2 FOR WORD BOUND
F'0004D2  3A71 6000     551.          MOVEA.W 0(A1,D6.W),A5 ;A5 PNTR IM(W**P)
F'0004D6  2E03          552.          MOVE.L D3,D7     ;D7=K
F'0004D8  2A03          553.          MOVE.L D3,D5     ;D5 TOO
F'0004DA  E545          554.          ASL.W #2,D5      ;K*4 FOR LWORD BOUND
F'0004DC  4DF0 5000     555.          LEA 0(A0,D5.W),A6   ;A6 IS INDEX X(K)
F'0004E0  DE41          556.          ADD.W D1,D7      ;D7=K+N2
F'0004E2  E547          557.          ASL.W #2,D7      ;D7=4(K+N2) LWORD BND
F'0004E4  47F0 7000     558.          LEA 0(A0,D7.W),A3 ;A3 PNT RE(X(K+N2))
F'0004E8  3A32 6000     559.          MOVE.W 0(A2,D6.W),D5     ;GET RE(W**P)
F'0004EC  3E05          560.          MOVE.W D5,D7     ;PUT ALSO IN D7
F'0004EE  CBD3          561.          MULS (A3),D5     ;RE(W**P)*RE(X(K+N2))
F'0004F0  3C0D          562.          MOVE.W A5,D6  ;GET IM(W**P)
F'0004F2  CDEB 0002     563.          MULS 2(A3),D6    ;IM(W**P)*IM(X(K+N2))
F'0004F6  9A86          564.          SUB.L D6,D5      ;D5=RE(T1)
F'0004F8  E0A5          565.          ASR.L D0,D5      ;/16384 TO SCALE
F'0004FA  3C0D          566.          MOVE.W A5,D6  ;GET IM(W**P)
F'0004FC  CDD3          567.          MULS (A3),D6     ;IM(W**P)*RE(X(K+N2))
F'0004FE  CFEB 0002     568.          MULS 2(A3),D7    ;RE(W**P)*IM(X(K+N2))
F'000502  DC87          569           ADD.L D7,D6      ,D6=IM(T1)
F'000504  E0A6          570.          ASR.L D0,D6      ;/16384 TO SCALE
F'000506  3E16          571.          MOVE.W (A6),D7   ;RE(X(K))
F'000508  9E45          572.          SUB.W D5,D7      ;RE(X(K))-RE(T1)
F'00050A  E247          573.          ASR.W #1,D7      ;/2
F'00050C  3687          574.          MOVE.W D7,(A3)   ;STORE ANSWER
F'00050E  3E2E 0002     575.          MOVE.W 2(A6),D7 ;IM(X(K))
F'000512  9E46          576.          SUB.W D6,D7      ;IM(X(K))-IM(T1)
F'000514  E247          577.          ASR.W #1,D7      ;/2
F'000516  3747 0002     578.          MOVE.W D7,2(A3) ;STORE ANSWER
F'00051A  DB56          579.          ADD.W D5,(A6)    ;RE(X(K))+RE(T1)
F'00051C  E0D6          580.          ASR.W (A6)       ;/2
F'00051E  DD6E 0002     581.          ADD.W D6,2(A6)   ;IM(X(K))+IM(T1)
F'000522  E0EE 0002     582.          ASR.W 2(A6)      ;/2
F'000526  5243          583.          ADDQ.W #1,D3     ;INCREMENT K
F'000528  51CC FF98     584.          DBRA D4,LOOPIN1  ;DO TILL I=-1
F'00052C  D641          585.          ADD.W D1,D3      ;K=K+N2
F'00052E  0C43'FFFF     586.          CMPI.W #NPTE-1,D3 ;K<N-1 ?
F'000532  6D 8A         587.          BLT INITI1       ;DO AGAIN IF SO
F'000534  4243          588.          CLR.W D3         ;K=0
F'000536  5342          589.          SUBQ.W #1,D2     ;NU1=NU1-1
F'000538  E341          590.          ASL.W #1,D1      ;N2=N2*2
F'00053A  5257          591.          ADDQ.W #1,(SP)
F'00053C  0C57 0007     592.          CMPI.W #NUGAM+1,(SP)
F'000540  6600 FF7C     593.          BNE INITI1
F'000544  301F          594.          MOVE.W (SP)+,D0
                        595.  *
                        596.  * NOW DO FILTERING, FILTER DATA IS AT "FILTER"
                        597.  * 64 POINTS
                        598.  *
                        599.  * DO COMPLEX MULTIPLY BY THE FILTER.
                        600.  * FILTER CAN BE +16384 TO -16384 WITH 16384
                        601.  * BEING THE 0 db GAIN VALUE
                        602.  * (A+iB)*(C+iD)
```

```
                                603.  *
                                604.  * PUT FILTERED DATA AT XDATA1
                                605.  *
F'000546  7C04                  606.          MOVEQ.L #4,D6           ;ADDRESS INC
F'000548  45FA ****             607.          LEA XDATA1(PC),A2
F'00054C  43FA ****             608.          LEA FILTER(PC),A1       ;A1 PNT FILTER
F'000550  303C'FFFF             609.          MOVE.W #NPTE-1,D0       ;LOOP COUNTER
F'000554  7A'00                 610.          MOVEQ.L #SHIFT14,D5     ;SHIFT COUNT
F'000556  3219                  611.  FILOOP1 MOVE.W (A1)+,D1         ;GET C
F'000558  3601                  612.          MOVE.W D1,D3
F'00055A  C3D8                  613.          MULS (A0)+,D1           ;DO AC
F'00055C  3419                  614.          MOVE W (A1)+,D2         ;GET D
F'00055E  3802                  615.          MOVE.W D2,D4
F'000560  C5D0                  616.          MULS (A0),D2            ;DO BD
F'000562  9282                  617.          SUB.L D2,D1             ;AC-BD
F'000564  EAA1                  618.          ASR.L D5,D1             ;ADJUST /16384
F'000566  C7D0                  619.          MULS (A0),D3            ;DO BC
F'000568  C9E0                  620.          MULS -(A0),D4           ;DO AD
F'00056A  D883                  621.          ADD.L D3,D4             ;BC+AD
F'00056C  EAA4                  622.          ASR.L D5,D4             ;ADJUST /16384
F'00056E  34C1                  623.          MOVE.W D1,(A2)+ ;STORE REAL
F'000570  34C4                  624.          MOVE.W D4,(A2)+ ;STORE IMAG
F'000572  D1C6                  625.          ADDA.L D6,A0
F'000574  51C8 FFE0             626.          DBRA D0,FILOOP1
                                627.  *
                                628.  * FILTERING DONE
                                629.  *
                                630.  * ASSERT SRQ AND WAIT FOR ACKNOWLEDGE
                                631.  *
F'000578  4E43                  632.          TRAP #SRQASRT
                                633.  *
                                634.  * RECEIVE DATA FROM SLAVE #0 AND PUT AT
                                635.  * XDATA0
                                636.  *
>>> WARNING line[637] Expr. type   [637] FSTSLVNM.A68
F'00057A  303C ****             637.          MOVE.W #NPTE*4,D0       ;BYTE COUNT
F'00057E  41FA ****             638.          LEA XDATA0(PC),A0
F'000582  13FC 00'00            639.          MOVE.B #ENABLEA,PGCR
          '00000000
F'00058A  4E45                  640.          TRAP #INDATA
F'00058C  4239'00000000         641.          CLR.B PGCR
                                642.  *
                                643.  * NOW SEND OUT FILTERED DATA TO OTHER
                                644.  * SLAVES
                                645.  *
                                646.  * ASSERT AN SRQ TO SIGNAL READY TO SEND
                                647.  *
F'000592  4E43                  648.          TRAP #SRQASRT
                                649.  *       MOVE.W #NPTE*4,D0       ;BYTE COUNT
F'000594  41FA ****             650.          LEA XDATA1(PC),A0
F'000598  13FC 00'00            651.          MOVE.B #ENABLEB,PGCR
          '00000000
F'0005A0  4E46                  652.          TRAP #OUTDATA
F'0005A2  4239'00000000         653.          CLR.B PGCR
                                654.  *
                                655.  * ASSERT AN SRQ TO SIGNAL READY FOR DATA
                                656.  * FROM SLAVE #2 AND RECEIVE; PUT AT XDATA2
                                657.  * NUMBER OF POINTS IS STILL IN D0
                                658.  *
F'0005A6  4E43                  659.          TRAP #SRQASRT
                                660.  *       MOVE.W #NPTE*4,D0       ;BYTE COUNT
F'0005AA  41FA ****             661.          LEA XDATA2(PC),A0
F'0005AE  13FC 00'00            662           MOVE.B #ENABLEA,PGCR
          '00000000
F'0005B6  4E45                  663.          TRAP #INDATA
F'0005B8  4239'00000000         664.          CLR.B PGCR
                                665.  *
                                666.  * ASSERT AN SRQ TO SIGNAL READY FOR DATA
                                667.  * FROM SLAVE #3 AND RECEIVE; PUT AT XDATA3
                                668.  * NUMBER OF POINTS IS STILL IN D0
                                669.  *
F'0005BE  4E43                  670.          TRAP #SRQASRT
                                671.  *       MOVE.W #NPTE*4,D0       ;BYTE COUNT
F'0005C0  41FA ****             672.          LEA XDATA3(PC),A0
F'0005C4  13FC 00'00            673.          MOVE.B #ENABLEA,PGCR
          '00000000
F'0005CC  4E45                  674.          TRAP #INDATA
F'0005CE  4239'00000000         675.          CLR.B PGCR
                                676.  *
                                677.  * ASSERT AN SRQ TO SIGNIFY DONE
                                678.  * DATA REDISTRUBUTION
```

```
                                679.  *
F'0005D4  4E43                  680.          TRAP #SRQASRT
                                681.  *
                                682.  * NOW MUST DO DATA REORDERING
                                683.  * A 4-SHUFFLE FROM XDATA0-3 TO DATAII
                                684.  * TO PREPARE FOR INVERSE TRANSFORMING
                                685.  *
F'0005D6  267C 000003FC         686.  REORD1  MOVEA.L #$3FC,A3
F'0005DC  247C 00000010         687.          MOVEA.L #$10,A2
F'0005E2  41FA ****             688.          LEA XDATA0(PC),A0
F'0005E6  43FA ****             689.          LEA DATAII(PC),A1
F'0005EA  72'FF                 690.          MOVEQ.L #NPE-1,D1
F'0005EC  7A'FF                 691.  RLOPT1  MOVEQ.L #NPTE-1,D5
F'0005EE  2298                  692.  RLOPN1  MOVE.L (A0)+,(A1)
F'0005F0  D3CA                  693.          ADDA.L A2,A1
F'0005F2  51CD FFFA             694.          DBRA D5,RLOPN1
F'0005F6  93CB                  695.          SUBA.L A3,A1
F'0005F8  51C9 FFF2             696.          DBRA D1,RLOPT1
                                697.  *
                                698.  * NOW DO THE INVERSE TRANSFORM
                                699.  * USE SAME SIN AND COS TABLES, NEGATE THE SIN
                                700.  * ENTRY, AND USE NO ASR #2 FOR DIVISION BY NPTE
                                701.  * NOTE SIGN CHANGE ON 4-ADDITIONS TOO
F'0005FC  2A7C'00000000         702.  S1PROR  MOVEA.L #NPE,A5          ;ADDR INC
F'000602  2E3C'00000000         703.          MOVE.L #SHIFT14,D7       ;/16384 SHFT
F'000608  303C'FFFF             704.          MOVE.W #NPTE-1,D0        ;MAJ LOOP CNTR
F'00060C  287C'FFFFFFFC         705.          MOVEA.L #NPT-4,A4        ;INDX TO ARRAYS
F'000612  43FA 023A             706.          LEA TWD1(PC),A1          ;TWID FACT PNTR
F'000616  41FA ****             707.          LEA DATAII(PC),A0        ;XDATA POINTER
F'00061A  2648                  708.  TWDL1R  MOVEA.L A0,A3
F'00061C  D7CC                  709.          ADDA.L A4,A3
F'00061E  3213                  710.  ADDL1R  MOVE.W (A3),D1           ;GET REAL PART
F'000620  362B 0102             711.          MOVE.W 258(A3),D3
F'000624  9243                  712.          SUB.W D3,D1
F'000626  362B 0200             713.          MOVE.W 512(A3),D3
F'00062A  9243                  714.          SUB.W D3,D1
F'00062C  362B 0302             715.          MOVE.W 770(A3),D3
F'000630  D243                  716.          ADD.W D3,D1              ;REAL SUM DONE
F'000632  342B 0002             717.          MOVE.W 2(A3),D2          ;IMAG SUM IN D2
F'000636  362B 0100             718.          MOVE.W 256(A3),D3
F'00063A  D443                  719.          ADD.W D3,D2
F'00063C  362B 0202             720.          MOVE.W 514(A3),D3
F'000640  9443                  721.          SUB.W D3,D2
F'000642  362B 0300             722.          MOVE.W 768(A3),D3
F'000646  9443                  723.          SUB.W D3,D2              ;IMAG SUM DONE
                                724.  *
                                725.  * NOW DO TWIDDLE FACTOR MULTIPLY
                                726.  * MUST NEGATE THE IMAGINARY PART OF TWDF
                                727.  *
F'000648  3A01                  728.          MOVE.W D1,D5
F'00064A  3831 C800             729.          MOVE.W 0(A1,A4.L),D4
F'00064E  3631 C802             730.          MOVE.W 2(A1,A4.L),D3
F'000652  4443                  731.          NEG.W D3
F'000654  CBC4                  732.          MULS D4,D5
F'000656  3C02                  733.          MOVE.W D2,D6
F'000658  CDC3                  734.          MULS D3,D6
F'00065A  9A86                  735.          SUB.L D6,D5              ;REAL IN D5
F'00065C  C3C3                  736.          MULS D3,D1
F'00065E  C5C4                  737.          MULS D4,D2
F'000660  D282                  738.          ADD.L D2,D1              ,IMAG IN D1
F'000662  EEA5                  739.          ASR.L D7,D5
F'000664  EEA1                  740.          ASR.L D7,D1
                                741.  *
                                742.  * STORE SUMMED AND TWIDDLED DATA AT DATAII IN
                                743.  * PLACE
                                744.  *
F'000666  3185 C800             745.          MOVE.W D5,0(A0,A4.L)
F'00066A  3181 C802             746.          MOVE.W D1,2(A0,A4.L)
F'00066E  99CD                  747.          SUBA.L A5,A4
F'000670  51C8 FFA8             748.          DBRA D0,TWDL1R
                                749.  *
                                750.  * INSERT FAVORITE 64-POINT FFT HERE
                                751.  * INPUT DATA AT DATAII (0-255)
                                752.  *
F'000674  78'F7                 753.  USHFL1R MOVEQ.L #NPTE-9,D4        ,COUNTER=K
F'000676  3C04                  754.  SWAPL1R MOVE.W D4,D6          ;PREPARE BIT REV
F'000678  3E06                  755.  BREV21R MOVE.W D6,D7        ;SAVE DATA
F'00067A  4246                  756.          CLR.W D6           ;
F'00067C  7A05                  757.          MOVEQ.L #5,D5      ;BIT COUNTER
F'00067E  E257                  758.  REV21R  ROXR.W #1,D7       ;SFT RIGHT,XTEND
F'000680  E356                  759.          ROXL.W #1,D6
```

```
F'000682  51CD FFFA        760.          DBRA D5,REV21R
F'000686  B846             761.          CMP.W D6,D4           ;I=REV(K)<=K?
F'000688  6C 14            762.          BGE.S DECK1R            ;NO SWAP IF SO
F'00068A  3204             763.          MOVE.W D4,D1          ;COPY OF K
F'00068C  E541             764.          ASL.W #2,D1           ;K*4
F'00068E  E546             765.          ASL.W #2,D6           ;I*4
F'000690  2430 1000        766.          MOVE.L 0(A0,D1.W),D2 ;DO SWAP
F'000694  21B0 6000 1000   767.          MOVE.L 0(A0,D6.W),0(A0,D1.W)
F'00069A  2182 6000        768.          MOVE.L D2,0(A0,D6.W) ;DONE SWAP
F'00069E  51CC FFD6        769.  DECK1R   DBRA D4,SWAPL1R
F'0006A2  7201             770.  FFT641R MOVEQ.L #1,D1   ;D1=N2=1
F'0006A4  43FA 00A8        771.          LEA SINBLK1(PC),A1  ;SIN VALS PNTR A1
F'0006A8  45FA 0124        772.          LEA COSBLK1(PC),A2  ;COS VALS PNTR A2
                           773.  *
                           774.  * L IS DOWN COUNTED #NUGAM-1 TO 0
                           775.  *
F'0006AC  3F3C 0001        776.          MOVE.W #1,-(SP)  ;(SP) IS L
F'0006B0  7405             777.          MOVEQ.L #NUGAM-1,D2 ;D2=NUGAM-1=NU1
F'0006B2  4283             778.          CLR.L D3              ;D3=K=0
F'0006B4  70'00            779.          MOVEQ.L #SHIFT14,D0
                           780.  *
                           781   * INITIALIZE I=N2-1, TO 0 IN STEPS -1
                           782.  *
F'0006B6  3801             783.  INITI1R MOVE.W D1,D4    ;D4=I=N2
F'0006B8  5344             784.          SUBQ.W #1,D4    ;SET FOR DBRA
                           785.  *
                           786.  * DETERMINE P=MOD(K*2**(NUGAM-L),64)
                           787.  *
                           788.  *
F'0006BA  3C03             789.  LPIN1R  MOVE.W D3,D6    ;D6=K
F'0006BC  3A17             790.          MOVE.W (SP),D5  ;GET L
F'0006BE  5D45             791.          SUBQ.W #NUGAM,D5 ;L-NUGAM
F'0006C0  4445             792.          NEG.W D5        ;NUGAM-L
F'0006C2  EB66             793.          ASL.W D5,D6     ;K*2**(NUGAM-L)
F'0006C4  0246 003F        794.          ANDI.W #$003F,D6 ;MOD 64
                           795.  *
                           796.  * D6 HAS INDEX REQUIRED

                           797.
F'0006C8  DC46             798.          ADD.W D6,D6     ;D6*2 FOR WORD BOUND
F'0006CA  3A71 6000        799.          MOVEA.W 0(A1,D6.W),A5 ;A5 PNTR IM(W**P)
F'0006CE  2E03             800.          MOVE.L D3,D7    ;D7=K
F'0006D0  2A03             801.          MOVE.L D3,D5    ;D5 TOO
F'0006D2  EB45             802.          ASL.W #2,D5     ;K*4 FOR LWORD BOUND
F'0006D4  4DF0 5000        803.          LEA 0(A0,D5.W),A6   ;A6 IS INDEX X(K)
F'0006D8  DE41             804.          ADD.W D1,D7     ;D7=K+N2
F'0006DA  E547             805.          ASL.W #2,D7     ;D7=4(K+N2) LWORD BND
F'0006DC  47F0 7000        806.          LEA 0(A0,D7.W),A3 ;A3 PNT RE(X(K+N2))
F'0006E0  3A32 6000        807.          MOVE.W 0(A2,D6.W),D5    ;GET RE(W**P)
F'0006E4  3E05             808.          MOVE.W D5,D7    ;PUT ALSO IN D7
F'0006E6  CBD3             809.          MULS (A3),D5    ;RE(W**P)*RE(X(K+N2))
F'0006E8  3C0D             810.          MOVE.W A5,D6  ;GET IM(W**P)
F'0006EA  4446             811.          NEG.W D6
F'0006EC  CDEB 0002        812.          MULS 2(A3),D6   ;IM(W**P)*IM(X(K+N2))
F'0006F0  9A86             813.          SUB.L D6,D5     ;D5=RE(T1)
F'0006F2  E0A5             814.          ASR.L D0,D5     ;/16384 TO SCALE
F'0006F4  3C0D             815.          MOVE.W A5,D6  ;GET IM(W**P)
F'0006F6  4446             816.          NEG.W D6
F'0006F8  CDD3             817.          MULS (A3),D6    ;IM(W**P)*RE(X(K+N2))
F'0006FA  CFEB 0002        818.          MULS 2(A3),D7   ;RE(W**P)*IM(X(K+N2))
F'0006FE  DC87             819.          ADD.L D7,D6     ;D6=IM(T1)
F'000700  E0A6             820.          ASR.L D0,D6     ;/16384 TO SCALE
F'000702  3E16             821.          MOVE.W (A6),D7  ;RE(X(K))
F'000704  9E45             822.          SUB.W D5,D7     ;RE(X(K))-RE  1)
F'000706  3687             823.          MOVE.W D7,(A3)  ;STORE ANSWER
F'000708  3E2E 0002        824.          MOVE.W 2(A6),D7 ;IM(X(K))
F'00070C  9E46             825.          SUB.W D6,D7     ,IM(X(K))-IM(T1)
F'00070E  3747 0002        826.          MOVE.W D7,2(A3) ;STORE ANSWER
F'000712  DB56             827.          ADD.W D5,(A6)   ;RE(X(K))+RE(T1)
F'000714  DD6E 0002        828.          ADD.W D6,2(A6)  ;IM(X(K))+IM(T1)
F'000718  5243             829.          ADDQ.W #1,D3    ;INCREMENT K
F'00071A  51CC FF9E        830.          DBRA D4,LPIN1R  ;DO TILL I=-1
F'00071E  D641             831.          ADD.W D1,D3     ;K=K+N2
F'000720  0C43'FFFF        832.          CMPI.W #NPTE-1,D3 ;K<N-1 ?
F'000724  6D 90            833.          BLT INITI1R     ;DO AGAIN IF SO
F'000726  4243             834.          CLR.W D3        ;K=0
F'000728  5342             835.          SUBQ.W #1,D2    ;NU1=NU1-1
F'00072A  E341             836.          ASL.W #1,D1     ;N2=N2*2
F'00072C  5257             837.          ADDQ.W #1,(SP)
F'00072E  0C57 0007        838.          CMPI.W #NUGAM+1,(SP)
F'000732  66 82            839.          BNE INITI1R
```

```
F'000734  301F                840.          MOVE.W (SP)+,D0
                              841.    *
                              842.    * ASSERT SRQ AND WAIT FOR ACKNOWLEDGE
                              843.    *
F'000736  4E43                844.          TRAP #SRQASRT
                              845.    * SEND OUT 64 COMPLEX POINTS CALCULATED
>>> WARNING line[846] Expr. type   [846] FSTSLVNM.A68
F'000738  303C ****           846.          MOVE.W #NPTE*4,D0       ;BYTE COUNT
F'00073C  13FC 00'00          847.          MOVE.B #ENABLEB,PGCR
          '00000000
F'000744  4E46                848.          TRAP #OUTDATA
F'000746  4239'00000000       849.          CLR.B PGCR
                              850.    *
                              851.    * DONE!
                              852.    *
F'00074C  4E75                853.          RTS
                              854.    *
                              855.    * SIN (-2*PI*I/64)*16384 FOR I=0 TO 63
                              856.    *
F'00074E  0000 F9BA F384      857.    SINBLK1 DC.W       0, -1606, -3196, -4756
          ED6C
F'000756  E782 E1D5 DC72      858.          DC.W   -6270, -7723, -9102,-10394
          D766
F'00075E  D2BF CE87 CAC9      859.          DC.W  -11585,-12665,-13623,-14449
          C78F
F'000766  C4DF C2C1 C13B      860           DC.W  -15137,-15679,-16069,-16305
          C04F
F'00076E  C000 C04F C13B      861.          DC.W  -16384,-16305,-16069,-15679
          C2C1
F'000776  C4DF C78F CAC9      862.          DC.W  -15137,-14449,-13623,-12665
          CE87
F'00077E  D2BF D766 DC72      863.          DC.W  -11585,-10394, -9102, -7723
          E1D5
F'000786  E782 ED6C F384      864.          DC.W   -6270, -4756, -3196, -1606
          F9BA
F'00078E  0000 0646 0C7C      865.          DC.W       0,  1606,  3196,  4756
          1294
F'000796  187E 1E2B 238E      866.          DC.W    6270,  7723,  9102, 10394
          289A
F'00079E  2D41 3179 3537      867.          DC.W   11585, 12665, 13623, 14449
          3871
F'0007A6  3B21 3D3F 3EC5      868.          DC.W   15137, 15679, 16069, 16305
          3FB1
F'0007AE  4000 3FB1 3EC5      869.          DC.W   16384, 16305, 16069, 15679
          3D3F
F'0007B6  3B21 3871 3537      870.          DC.W   15137, 14449, 13623, 12665
          3179
F'0007BE  2D41 289A 238E      871.          DC.W   11585, 10394,  9102,  7723
          1E2B
F'0007C6  187E 1294 0C7C      872.          DC.W    6270,  4756,  3196,  1606
          0646
                              873.    *
                              874.    * COS(-2*PI*I/64)*16384 FOR I=0 TO 63
                              875.    *
F'0007CE  4000 3FB1 3EC5      876.    COSBLK1 DC.W   16384, 16305, 16069, 15679
          3D3F
F'0007D6  3B21 3871 3537      877.          DC.W   15137, 14449, 13623, 12665
          3179
F'0007DE  2D41 289A 238E      878.          DC.W   11585, 10394,  9102,  7723
          1E2B
F'0007E6  187E 1294 0C7C      879.          DC.W    6270,  4756,  3196,  1606
          0646
F'0007EE  0000 F9BA F384      880.          DC.W       0, -1606, -3196, -4756
          ED6C
F'0007F6  E782 E1D5 DC72      881.          DC.W   -6270, -7723, -9102,-10394
          D766
F'0007FE  D2BF CE87 CAC9      882.          DC.W  -11585,-12665,-13623,-14449
          C78F
F'000806  C4DF C2C1 C13B      883.          DC.W  -15137,-15679,-16069,-16305
          C04F
F'00080E  C000 C04F C13B      884.          DC.W  -16384,-16305,-16069,-15679
          C2C1
F'000816  C4DF C78F CAC9      885.          DC.W  -15137,-14449,-13623,-12665
          CE87
F'00081E  D2BF D766 DC72      886.          DC.W  -11585,-10394, -9102, -7723
          E1D5
F'000826  E782 ED6C F384      887.          DC.W   -6270, -4756, -3196, -1606
          F9BA
F'00082E  0000 0646 0C7C      888.          DC.W       0,  1606,  3196,  4756
          1294
F'000836  187E 1E2B 238E      889.          DC.W    6270,  7723,  9102, 10394
```

```
                289A
F'00083E  2D41 3179 3537  890.          DC.W   11585, 12665, 13623, 14449
                3871
F'000846  3B21 3D3F 3EC5  891.          DC.W   15137, 15679, 16069, 16305
                3FB1
                          892.  *
                          893.  * TWIDDLE FACTOR TABLE 1 TWD1
                          894.  *
F'00084E  4000 0000 3FFB  895.  TWD1    DC.W   16384,     0, 16379,  -402
                FE6E
F'000856  3FEC FCDC 3FD4  896.          DC.W   16364,  -804, 16340, -1205
                FB4B
F'00085E  3FB1 F9BA 3F85  897.          DC.W   16305, -1606, 16261, -2006
                F82A
F'000866  3F4F F69C 3F0F  898.          DC.W   16207, -2404, 16143, -2801
                F50F
F'00086E  3EC5 F384 3E72  899.          DC.W   16069, -3196, 15986, -3590
                F1FA
F'000876  3E15 F073 3DAF  900.          DC.W   15893, -3981, 15791, -4370
                EEEE
F'00087E  3D3F ED6C 3CC5  901.          DC.W   15679, -4756, 15557, -5139
                EBED
F'000886  3C42 EA70 3BB6  902.          DC.W   15426, -5520, 15286, -5897
                E8F7
F'00088E  3B21 E782 3A82  903.          DC.W   15137, -6270, 14978, -6639
                E611
F'000896  39DB E4A3 392B  904.          DC.W   14811, -7005, 14635, -7366
                E33A
F'00089E  3871 E1D5 37B0  905.          DC.W   14449, -7723, 14256, -8076
                E074
F'0008A6  36E5 DF19 3612  906.          DC.W   14053, -8423, 13842, -8765
                DDC3
F'0008AE  3537 DC72 3453  907.          DC.W   13623, -9102, 13395, -9434
                DB26
F'0008B6  3368 D9E0 3274  908.          DC.W   13160, -9760, 12916,-10080
                D8A0
F'0008BE  3179 D766 3076  909.          DC.W   12665,-10394, 12406,-10702
                D632
F'0008C6  2F6C D505 2E5A  910.          DC.W   12140,-11003, 11866,-11297
                D3DF
F'0008CE  2D41 D2BF 2C21  911.          DC.W   11585,-11585, 11297,-11866
                D1A6
F'0008D6  2AFB D094 29CE  912.          DC.W   11003,-12140, 10702,-12406
                CF8A
F'0008DE  289A CE87 2760  913.          DC.W   10394,-12665, 10080,-12916
                CD8C
F'0008E6  2620 CC98 24DA  914.          DC.W    9760,-13160,  9434,-13395
                  AD
F'0008EE  238E CAC9 223D  915.          DC.W    9102,-13623,  8765,-13842
                C9EE
F'0008F6  20E7 C91B 1F8C  916.          DC.W    8423,-14053,  8076,-14256
                C850
F'0008FE  1E2B C78F 1CC6  917.          DC.W    7723,-14449,  7366,-14635
                C6D5
F'000906  1B5D C625 19EF  918.          DC.W    7005,-14811,  6639,-14978
                C57E
F'00090E  187E C4DF 1709  919.          DC.W    6270,-15137,  5897,-15286
                C44A
F'000916  1590 C3BE 1413  920.          DC.W    5520,-15426,  5139,-15557
                C33B
F'00091E  1294 C2C1 1112  921.          DC.W    4756,-15679,  4370,-15791
                C251
F'000926  0F8D C1EB 0E06  922.          DC.W    3981,-15893,  3590,-15986
                C18E
F'00092E  0C7C C13B 0AF1  923.          DC.W    3196,-16069,  2801,-16143
                C0F1
F'000936  0964 C0B1 07D6  924.          DC.W    2404,-16207,  2006,-16261
                C07B
F'00093E  0646 C04F 04B5  925.          DC.W    1606,-16305,  1205,-16340
                C02C
F'000946  0324 C014 0192  926.          DC.W     804,-16364,   402,-16379
                C005
                          927.
                          928.  ********************
                          929.  * SLAVE 2 PROGRAM *
                          930.  ********************
                          931.  *
                          932.  * INPUT DATA AT DATAII, TWIDDLE FACTORS AT
                          933.  * TWD2
                          934.  *
                          935.  *
```

```
F'000948  2A7C'00000000    936.  S2PRO   MOVEA.L #NPE,A5          ;ADDR INC
F'000954  2E3C'00000000    937.          MOVE.L #SHIFT14,D7       ;/16384 SHFT
F'00095A  303C'FFFF        938.          MOVE.W #NPTE-1,D0        ;MAJ LOOP CNTR
F'00095E  287C'FFFFFFFC    939.          MOVEA.L #NPT-4,A4        ;INDX TO ARRAYS
F'000964  43FA 0444        940.          LEA TWD2(PC),A1          ;TWID FACT PNTR
F'000968  41FA ****        941.          LEA DATAII(PC),A0        ;XDATA POINTER
F'00096C  2648             942.  TWDL2   MOVEA.L A0,A3
F'00096E  D7CC             943.          ADDA.L A4,A3
F'000970  3213             944.  ADDL2   MOVE.W (A3),D1           ;GET REAL PART
F'000972  48C1             945.          EXT.L D1                 ;REAL SUM IN D1
F'000974  362B 0100        946.          MOVE.W 256(A3),D3
F'000978  48C3             947.          EXT.L D3
F'00097A  9283             948.          SUB.L D3,D1
F'00097C  362B 0200        949.          MOVE.W 512(A3),D3
F'000980  48C3             950.          EXT.L D3
F'000982  D283             951.          ADD.L D3,D1
F'000984  362B 0300        952.          MOVE.W 768(A3),D3
F'000988  48C3             953.          EXT.L D3
F'00098A  9283             954.          SUB.L D3,D1              ;REAL SUM DONE
F'00098C  342B 0002        955.          MOVE.W 2(A3),D2          ;IMAG SUM IN D2
F'000990  48C2             956.          EXT.L D2
F'000992  362B 0102        957.          MOVE.W 258(A3),D3
F'000996  48C3             958.          EXT.L D3
F'000998  9483             959.          SUB.L D3,D2
F'00099A  362B 0202        960.          MOVE.W 514(A3),D3
F'00099E  48C3             961.          EXT.L D3
F'0009A0  D483             962.          ADD.L D3,D2
F'0009A2  362B 0302        963.          MOVE.W 770(A3),D3
F'0009A6  48C3             964.          EXT.L D3
F'0009A8  9483             965.          SUB.L D3,D2              ;IMAG SUM DONE
F'0009AA  E481             966.          ASR.L #2,D1
F'0009AC  E482             967.          ASR.L #2,D2              ;ADJUST SUMS
                           968.  *
                           969.  * NOW DO TWIDDLE FACTOR MULTIPLY
                           970.  *
F'0009AE  3A01             971.          MOVE.W D1,D5
F'0009B0  3831 C800        972.          MOVE.W 0(A1,A4.L),D4
F'0009B4  3631 C802        973.          MOVE.W 2(A1,A4.L),D3
F'0009B8  CBC4             974.          MULS D4,D5
F'0009BA  3C02             975.          MOVE.W D2,D6
F'0009BC  CDC3             976.          MULS D3,D6
F'0009BE  9A86             977.          SUB.L D6,D5              ;REAL IN D5
F'0009C0  C3C3             978.          MULS D3,D1
F'0009C2  C5C4             979.          MULS D4,D2
F'0009C4  D282             980.          ADD.L D2,D1              ;IMAG IN D1
F'0009C6  EEA5             981.          ASR.L D7,D5
F'0009C8  EEA1             982.          ASR.L D7,D1
                           983.  *
                           984.  * STORE SUMMED AND TWIDDLED DATA AT XDATA IN
                           985.  * PLACE
                           986.  *
F'0009CA  3185 C800        987.          MOVE.W D5,0(A0,A4.L)
F'0009CE  3181 C802        988.          MOVE.W D1,2(A0,A4.L)
F'0009D2  99CD             989.          SUBA.L A5,A4
F'0009D4  51C8 FF96        990.          DBRA D0,TWDL2
                           991.  *
                           992.  * INSERT FAVORITE 64-POINT FFT HERE
                           993.  * INPUT DATA AT XDATA 0-255
                           994.  *
F'0009D8  78'F7            995.  USHFL2  MOVEQ.L #NPTE-9,D4        ;COUNTER=K
F'0009DA  3C04             996.  SWAPL2  MOVE.W D4,D6          ;PREPARE BIT REV
F'0009DC  3E06             997.  BREV22  MOVE.W D6,D7        ;SAVE DATA
F'0009DE  4246             998.          CLR.W D6            ;
F'0009E0  7A05             999.          MOVEQ.L #5,D5       ;BIT COUNTER
F'0009E2  E257            1000.  REV22   ROXR.W #1,D7        ;SFT RIGHT,XTEND
F'0009E4  E356            1001.          ROXL.W #1,D6
F'0009E6  51CD FFFA       1002.          DBRA D5,REV22
F'0009EA  B846            1003.          CMP.W D6,D4           ;I=REV(K)<=K?
F'0009EC  6C 14           1004.          BGE.S DECK2            ;NO SWAP IF SO
F'0009EE  3204            1005.          MOVE.W D4,D1          ;COPY OF K
F'0009F0  E541            1006.          ASL.W #2,D1           ;K*4
F'0009F2  E546            1007.          ASL.W #2,D6           ;I*4
F'0009F4  2430 1000       1008.          MOVE.L 0(A0,D1.W),D2 ;DO SWAP
F'0009F8  21B0 6000 1000  1009.          MOVE.L 0(A0,D6.W),0(A0,D1.W)
F'0009FE  2182 6000       1010.          MOVE.L D2,0(A0,D6.W) ;DONE SWAP
```

```
F'000A02  51CC FFD6      1011.  DECK2   DBRA D4,SWAPL2
F'000A06  7201           1012.  FFT642  MOVEQ.L #1,D1    ;D1=N2=1
F'000A08  43FA 02A0      1013.          LEA SINBLK2(PC),A1  ;SIN VALS PNTR A1
F'000A0C  45FA 031C      1014.          LEA COSBLK2(PC),A2  ;COS VALS PNTR A2
                         1015.  *
                         1016.  * L IS DOWN COUNTED #NUGAM-1 TO 0
                         1017.  *
F'000A10  3F3C 0001      1018.          MOVE.W #1,-(SP)  ;(SP) IS L
F'000A14  7405           1019.          MOVEQ.L #NUGAM-1,D2     ;D2=NUGAM-1=NU1
F'000A16  4283           1020.          CLR.L D3              ;D3=K=0
F'000A18  70'00          1021.          MOVEQ.L #SHIFT14,D0
                         1022.  *
                         1023.  * INITIALIZE I=N2-1, TO 0 IN STEPS -1
                         1024.  *
F'000A1A  3801           1025.  INITI2  MOVE.W D1,D4     ;D4=I=N2
F'000A1C  5344           1026.          SUBQ.W #1,D4     ;SET FOR DBRA
                         1027.  *
                         1028.  * DETERMINE P=MOD(K*2**(NUGAM-L),64)
                         1029.  *
                         1030.  *
F'000A1E  3C03           1031.  LOOPIN2 MOVE.W D3,D6     ;D6=K
F'000A20  3A17           1032.          MOVE.W (SP),D5   ;GET L
F'000A22  5D45           1033.          SUBQ.W #NUGAM,D5 ;L-NUGAM
F'000A24  4445           1034.          NEG.W D5         ;NUGAM-L
F'000A26  EB66           1035.          ASL.W D5,D6      ;K*2**(NUGAM-L)
F'000A28  0246 003F      1036.          ANDI.W #$003F,D6 ;MOD 64
                         1037.  *
                         1038.  * D6 HAS INDEX REQUIRED
                         1039.  *
F'000A2C  DC46           1040.          ADD.W D6,D6      ;D6*2 FOR WORD BOUND
F'000A2E  3A71 6000      1041.          MOVEA.W 0(A1,D6.W),A5 ;A5 PNTR IM(W**P)
F'000A32  2E03           1042.          MOVE.L D3,D7     ;D7=K
F'000A34  2A03           1043.          MOVE.L D3,D5     ;D5 TOO
F'000A36  E545           1044.          ASL.W #2,D5      ;K*4 FOR LWORD BOUND
F'000A38  4DF0 5000      1045.          LEA 0(A0,D5.W),A6    ;A6 IS INDEX X(K)
F'000A3C  DE41           1046.          ADD.W D1,D7      ;D7=K+N2
F'000A3E  E547           1047.          ASL.W #2,D7      ;D7=4(K+N2) LWORD BND
F'000A40  47F0 7000      1048.          LEA 0(A0,D7.W),A3 ,A3 PNT RE(X(K+N2))
F'000A44  3A32 6000      1049.          MOVE.W 0(A2,D6.W),D5     ;GET RE(W**P)
F'000A48  3E05           1050.          MOVE.W D5,D7     ;PUT ALSO IN D7
F'000A4A  CBD3           1051.          MULS (A3),D5     ;RE(W**P)*RE(X(K+N2))
F'000A4C  3C0D           1052.          MOVE.W A5,D6  ;GET IM(W**P)
F'000A4E  CDEB 0002      1053.          MULS 2(A3),D6    ;IM(W**P)*IM(X(K+N2))
F'000A52  9A86           1054.          SUB.L D6,D5      ;D5=RE(T1)
F'000A54  E0A5           1055.          ASR.L D0,D5      ;/16384 TO SCALE
F'000A56  3C0D           1056.          MOVE.W A5,D6  ;GET IM(W**P)
F'000A58  CDD3           1057.          MULS (A3),D6     ;IM(W**P)*RE(X(K+N2))
F'000A5A  CFEB 0002      1058.          MULS 2(A3),D7    ;RE(W**P)*IM(X(K+N2))
F'000A5E  DC87           1059.          ADD.L D7,D6      ,D6=IM(T1)
F'000A60  E0A6           1060.          ASR.L D0,D6      ;/16384 TO SCALE
F'000A62  3E16           1061.          MOVE.W (A6),D7   ;RE(X(K))
F'000A64  9E45           1062.          SUB.W D5,D7      ;RE(X(K))-RE(T1)
F'000A66  E247           1063.          ASR.W #1,D7      ;/2
F'000A68  3687           1064.          MOVE.W D7,(A3)   ;STORE ANSWER
F'000A6A  3E2E 0002      1065.          MOVE W 2(A6),D7 ;IM(X(K))
F'000A6E  9E46           1066.          SUB.W D6,D7      ;IM(X(K))-IM(T1)
F'000A70  E247           1067.          ASR.W #1,D7      ;/2
F'000A72  3747 0002      1068.          MOVE.W D7,2(A3) ;STORE ANSWER
F'000A76  DB56           1069.          ADD.W D5,(A6)    ;RE(X(K))+RE(T1)
F'000A78  E0D6           1070.          ASR.W (A6)       ;/2
F'000A7A  DD6E 0002      1071.          ADD.W D6,2(A6)   ;IM(X(K))+IM(T1)
F'000A7E  E0EE 0002      1072.          ASR.W 2(A6)      ;/2
F'000A82  5243           1073.          ADDQ.W #1,D3     ;INCREMENT K
F'000A84  51CC FF98      1074.          DBRA D4,LOOPIN2  ;DO TILL I=-1
F'000A88  D741           1075.          ADD.W D1,D3      ;K=K+N2
F'000A8A  0C43'FFFF      1076.          CMPI.W #NPTE-1,D3 ;K<N-1 ?
F'000A8E  6D 8A          1077.          BLT INITI2       ;DO AGAIN IF SO
F'000A90  4243           1078.          CLR.W D3         ;K=0
F'000A92  5342           1079.          SUBQ.W #1,D2     ;NU1=NU1-1
F'000A94  E341           1080.          ASL.W #1,D1      ;N2=N2*2
F'000A96  5257           1081.          ADDQ.W #1,(SP)
F'000A98  0C57 0007      1082.          CMPI.W #NUGAM+1,(SP)
F'000A9C  6600 FF7C      1083.          BNE INITI2    ;DO TILL L=-1
F'000AA0  301F           1084.          MOVE.W (SP)+,D0
                         1085.  *
                         1086.  * NOW DO FILTERING, FILTER DATA IS AT "FILTER"
                         1087.  * 64 POINTS
                         1088.  *
                         1089.  * DO COMPLEX MULTIPLY BY THE FILTER.
                         1090.  * FILTER CAN BE +16384 TO -16384 WITH 16384
                         1091.  * BEING THE 0 db GAIN VALUE.
```

```
                          1092.  *  (A+iB)*(C+iD)
                          1093.  *
                          1094.  * PUT RESULTS AT XDATA2
                          1095.  *
F'000AA2  7C04            1096.          MOVEQ.L #4,D6            ;ADDRESS INC
F'000AA4  45FA ****       1097.          LEA XDATA2(PC),A2
F'000AA8  43FA ****       1098.          LEA FILTER(PC),A1        ;A1 PNT FILTER
F'000AAC  303C'FFFF       1099.          MOVE.W #NPTE-1,D0        ;LOOP COUNTER
F'000AB0  7A'00           1100.          MOVEQ.L #SHIFT14,D5      ;SHIFT COUNT
F'000AB2  3219            1101.  FILOOP2 MOVE.W (A1)+,D1          ;GET C
F'000AB4  3601            1102.          MOVE.W D1,D3
F'000AB6  C3D8            1103.          MULS (A0)+,D1            ;DO AC
F'000AB8  3419            1104.          MOVE.W (A1)+,D2          ;GET D
F'000ABA  3802            1105.          MOVE.W D2,D4
F'000ABC  C5D0            1106.          MULS (A0),D2             ;DO BD
F'000ABE  9282            1107.          SUB.L D2,D1              ;AC-BD
F'000AC0  EAA1            1108.          ASR.L D5,D1              ;ADJUST /16384
F'000AC2  C7D0            1109.          MULS (A0),D3             ;DO BC
F'000AC4  C9E0            1110.          MULS -(A0),D4            ;DO AD
F'000AC6  D883            1111.          ADD.L D3,D4              ;BC+AD
F'000AC8  EAA4            1112.          ASR.L D5,D4              ;ADJUST /16384
F'000ACA  34C1            1113.          MOVE.W D1,(A2)+ ;STORE REAL
F'000ACC  34C4            1114.          MOVE.W D4,(A2)+ ;STORE IMAG
F'000ACE  D1C6            1115.          ADDA.L D6,A0
F'000AD0  51C8 FFE0       1116.          DBRA D0,FILOOP2
                          1117.  *
                          1118.  * ASSERT SRQ AND WAIT FOR ACKNOWLEDGE
                          1119.  *
F'000AD4  4E43            1120.          TRAP #SRQASRT
                          1121.  *
                          1122.  * RECEIVE DATA FROM SLAVE #0
                          1123.  *
>>> WARNING line[1124] Expr. type   [1124] FSTSLVNM.A68
F'000AD6  303C ****       1124.          MOVE.W #NPTE*4,D0        ;BYTE COUNT
F'000ADA  41FA ****       1125.          LEA XDATA0(PC),A0
F'000ADE  13FC 00'00      1126.          MOVE.B #ENABLEA,PGCR
          '00000000
F'000AE6  4E45            1127.          TRAP #INDATA
F'000AE8  4239'00000000   1128.          CLR.B PGCR
                          1129.  *
                          1130.  * ASSERT AN SRQ TO SIGNAL READY FOR DATA
                          1131.  * FROM SLAVE #1 AND RECEIVE; PUT AT XDATA1
                          1132.  * NUMBER OF POINTS IS STILL IN D0
                          1133.  *
F'000AEE  4E43            1134.          TRAP #SRQASRT
                          1135.  *       MOVE.W #NPTE*4,D0        ;BYTE COUNT
F'000AF0  41FA ****       1136.          LEA XDATA1(PC),A0
F'000AF4  13FC 00'00      1137.          MOVE.B #ENABLEA,PGCR
          '00000000
F'000AFC  4E45            1138.          TRAP #INDATA
F'000AFE  4239'00000000   1139.          CLR.B PGCR
                          1140.  *
                          1141.  * ASSERT AN SRQ TO SIGNAL READY TO SEND DATA
                          1142.  * TO OTHER SLAVES
                          1143.  * NUMBER OF POINTS IS STILL IN D0
                          1144.  *
F'000B04  4E43            1145.          TRAP #SRQASRT
                          1146.  *       MOVE.W #NPTE*4,D0        ;BYTE COUNT
F'000B06  41FA ****       1147.          LEA XDATA2(PC),A0
F'000B0A  13FC 00'00      1148.          MOVE.B #ENABLEB,PGCR
          '00000000
F'000B12  4E46            1149.          TRAP #OUTDATA
F'000B14  4239'00000000   1150.          CLR.B PGCR
                          1151.  *
                          1152.  * ASSERT AN SRQ TO SIGNAL READY FOR DATA
                          1153.  * FROM SLAVE #3 AND RECEIVE; PUT AT XDATA3
                          1154.  * NUMBER OF POINTS IS STILL IN D0
                          1155.  *
F'000B1A  4E43            1156.          TRAP #SRQASRT
                          1157.  *       MOVE.W #NPTE*4,D0        ;BYTE COUNT
F'000B1C  41FA ****       1158.          LEA XDATA3(PC),A0
F'000B20  13FC 00'00      1159.          MOVE.B #ENABLEA,PGCR
          '00000000
F'000B28  4E45            1160.          TRAP #INDATA
F'000B2A  4239'00000000   1161.          CLR.B PGCR
                          1162.  *
                          1163.  * ASSERT AN SRQ TO SIGNIFY DONE
                          1164.  * DATA REDISTRUBUTION
                          1165.  *
F'000B30  4E43            1166.          TRAP #SRQASRT
                          1167.  *
```

```
                          1168.  * NOW MUST DO DATA REORDERING
                          1169.  * A 4-SHUFFLE FROM XDATA0-3 TO DATAII
                          1170.  * TO PREPARE FOR INVERSE TRANSFORMING
                          1171.  *
F'000B32  267C 000003FC   1172.  REORD2  MOVEA.L #$3FC,A3
F'000B38  247C 00000010   1173.         MOVEA.L #$10,A2
F'000B3E  41FA ****       1174.         LEA XDATA0(PC),A0
F'000B42  43FA ****       1175.         LEA DATAII(PC),A1
F'000B46  72'FF           1176.         MOVEQ.L #NPE-1,D1
F'000B48  7A'FF           1177.  RLOPT2  MOVEQ.L #NPTE-1,D5
F'000B4A  2298            1178.  RLOPN2  MOVE.L (A0)+,(A1)
F'000B4C  D3CA            1179.         ADDA.L A2,A1
F'000B4E  51CD FFFA       1180.         DBRA D5,RLOPN2
F'000B52  93CB            1181.         SUBA.L A3,A1
F'000B54  51C9 FFF2       1182.         DBRA D1,RLOPT2
                          1183.  *
                          1184.  * NOW DO THE INVERSE TRANSFORM
                          1185.  * USE SAME SIN AND COS TABLES, NEC TE THE SIN
                          1186.  * ENTRY, AND USE NO ASR #2 FOR DIVISION BY NPTE
                          1187.  * NOTE SIGN CHANGE ON 4-ADDITIONS TOO
                          1188.  *
F'000B58  2A7C'00000000   1189.  S2PROR  MOVEA.L #NPE,A5          ;ADDR INC
F'000B5E  2E3C'00000000   1190.         MOVE.L #SHIFT14,D7       ;/16384 SHFT
F'000B64  303C'FFFF       1191.         MOVE.W #NPTE-1,D0        ;MAJ LOOP CNTR
F'000B68  287C'FFFFFFFC   1192.         MOVEA.L #NPT-4,A4        ;INDX TO ARRAYS
F'000B6E  43FA 023A       1193.         LEA TWD2(PC),A1          ;TWID FACT PNTR
F'000B72  41FA ****       1194.         LEA DATAII(PC),A0        ;XDATA POINTER
F'000B76  2648            1195.  TWDL2R  MOVEA.L A0,A3
F'000B78  D7CC            1196.         ADDA.L A4,A3
F'000B7A  3213            1197.  ADDL2R  MOVE.W (A3),D1           ;GET REAL PART
F'000B7C  362B 0100       1198.         MOVE.W 256(A3),D3
F'000B80  9243            1199.         SUB.W D3,D1
F'000B82  362B 0200       1200.         MOVE.W 512(A3),D3
F'000B86  D243            1201.         ADD.W D3,D1
F'000B88  362B 0300       1202.         MOVE.W 768(A3),D3
F'000B8C  9243            1203.         SUB.W D3,D1              ;REAL SUM DONE
F'000B8E  342B 0002       1204.         MOVE.W 2(A3),D2          ;IMAG SUM IN D2
F'000B92  362B 0102       1205.         MOVE.W 258(A3),D3
F'000B96  9443            1206.         SUB.W D3,D2
F'000B98  362B 0202       1207.         MOVE.W 514(A3),D3
F'000B9C  D443            1208.         ADD.W D3,D2
F'000B9E  362B 0302       1209.         MOVE.W 770(A3),D3
F'000BA2  9443            1210.         SUB.W D3,D2              ;IMAG SUM DONE
                          1211.  *
                          1212.  * NOW DO TWIDDLE FACTOR MULTIPLY
                          1213.  *
F'000BA4  3A01            1214.         MOVE.W D1,D5
F'000BA6  3831 C800       1215.         MOVE.W 0(A1,A4.L),D4
F'000BAA  3631 C802       1216.         MOVE.W 2(A1,A4.L),D3
F'000BAE  4443            1217.         NEG.W D3
F'000BB0  CBC4            1218.         MULS D4,D5
F'000BB2  3C02            1219.         MOVE.W D2,D6
F'000BB4  CDC3            1220.         MULS D3,D6
F'000BB6  9A86            1221.         SUB.L D6,D5              ;REAL IN D5
F'000BB8  C3C3            1222.         MULS D3,D1
F'000BBA  C5C4            1223.         MULS D4,D2
F'000BBC  D282            1224.         ADD.L D2,D1              ;IMAG IN D1
F'000BBE  EEA5            1225.         ASR.L D7,D5
F'000BC0  EEA1            1226.         ASR.L D7,D1
                          1227.  *
                          1228.  * STORE SUMMED AND TWIDDLED DATA AT XDATA IN
                          1229.  * PLACE
                          1230.  *
F'000BC2  3185 C800       1231.         MOVE.W D5,0(A0,A4.L)
F'000BC6  3181 C802       1232.         MOVE.W D1,2(A0,A4.L)
F'000BCA  99CD            1233.         SUBA.L A5,A4
F'000BCC  51C8 FFA8       1234.         DBRA D0,TWDL2R
                          1235.  *
                          1236.  * INSERT FAVORITE 64-POINT FFT HERE
                          1237.  * INPUT DATA AT XDATA 0-255
                          1238.  *
                          1239.  *
F'000BD0  78'F7           1240.  USHFL2R MOVEQ.L #NPTE-9,D4        ;COUNTER=K
F'000BD2  3C04            1241.  SWAPL2R MOVE.W D4,D6          ;PREPARE BIT REV
F'000BD4  3E06            1242.  BREV22R MOVE.W D6,D7        ;SAVE DATA
F'000BD6  4246            1243.         CLR.W D6             ;
F'000BD8  7A05            1244.         MOVEQ.L #5,D5        ;BIT COUNTER
F'000BDA  E257            1245.  REV22R  ROXR.W #1,D7         ;SFT RIGHT,XTEND
F'000BDC  E356            1246.         ROXL.W #1,D6
F'000BDE  51CD FFFA       1247.         DBRA D5,REV22R
F'000BE2  B846            1248.         CMP.W D6,D4          ;I=REV(K)<=K?
```

```
F'000BE4  6C 14           1249.          BGE.S DECK2R            ;NO SWAP IF SO
F'000BE6  3204            1250.          MOVE.W D4,D1           ;COPY OF K
F'000BE8  E541            1251.          ASL.W #2,D1            ;K*4
F'000BEA  E546            1252.          ASL.W #2,D6            ;I*4
F'000BEC  2430 1000       1253.          MOVE.L 0(A0,D1.W),D2 ;DO SWAP
F'000BF0  21B0 6000 1000  1254.          MOVE.L 0(A0,D6.W),0(A0,D1.W)
F'000BF6  2182 6000       1255.          MOVE.L D2,0(A0,D6.W) ;DONE SWAP
F'000BFA  51CC FFD6       1256.  DECK2R   DBRA D4,SWAPL2R
F'000BFE  7201            1257.  FFT642R MOVEQ.L #1,D1    ;D1=N2=1
F'000C00  43FA 00A8       1258.          LEA SINBLK2(PC),A1  ;SIN VALS PNTR A1
F'000C04  45FA 0124       1259.          LEA COSBLK2(PC),A2  ;COS VALS PNTR A2
                          1260.  *
                          1261.  * L IS DOWN COUNTED #NUGAM-1 TO 0
                          1262.  *
F'000C08  3F3C 0001       1263.          MOVE.W #1,-(SP)  ;(SP) IS L
F'000C0C  7405            1264.          MOVEQ.L #NUGAM-1,D2 ;D2=NUGAM-1=NU1
F'000C0E  4283            1265.          CLR.L D3               ;D3=K=0
F'000C10  70'00           1266.          MOVEQ.L #SHIFT14,D0
                          1267.  *
                          1268.  * INITIALIZE I=N2-1, TO 0 IN STEPS -1
                          1269.  *
F'000C12  3801            1270.  INITI2R MOVE.W D1,D4     ;D4=I=N2
F'000C14  5344            1271.          SUBQ.W #1,D4     ;SET FOR DBRA
                          1272.  *
                          1273.  * DETERMINE P=MOD(K*2**(NUGAM-L),64)
                          1274.  *
                          1275.  *
F'000C16  3C03            1276.  LPIN2R  MOVE.W D3,D6     ;D6=K
F'000C18  3A17            1277.          MOVE.W (SP),D5   ;GET L
F'000C1A  5D45            1278.          SUBQ.W #NUGAM,D5 ;L-NUGAM
F'000C1C  4445            1279.          NEG.W D5         ;NUGAM-L
F'000C1E  EB66            1280.          ASL.W D5,D6      ;K*2**(NUGAM-L)
F'000C20  0246 003F       1281.          ANDI.W #$003F,D6 ;MOD 64
                          1282.  *
                          1283.  * D6 HAS INDEX REQUIRED
                          1284.  *
F'000C24  DC46            1285.          ADD.W D6,D6      ;D6*2 FOR WORD BOUND
F'000C26  3A71 6000       1286.          MOVEA.W 0(A1,D6.W),A5 ;A5 PNTR IM(W**P)
F'000C2A  2E03            1287.          MOVE.L D3,D7     ;D7=K
F'000C2C  2A03            1288.          MOVE.L D3,D5     ;D5 TOO
F'000C2E  E545            1289.          ASL.W #2,D5      ;K*4 FOR LWORD BOUND
F'000C30  4DF0 5000       1290.          LEA 0(A0,D5.W),A6    ;A6 IS INDEX X(K)
F'000C34  DE41            1291.          ADD.W D1,D7      ;D7=K+N2
F'000C36  E547            1292.          ASL.W #2,D7      ;D7=4(K+N2) LWORD BND
F'000C38  47F0 7000       1293.          LEA 0(A0,D7.W),A3 ;A3 PNT RE(X(K+N2))
F'000C3C  3A32 6000       1294.          MOVE.W 0(A2,D6.W),D5     ;GET RE(W**P)
F'000C40  3E05            1295.          MOVE.W D5,D7     ;PUT ALSO IN D7
F'000C42  CBD3            1296.          MULS (A3),D5     ;RE(W**P)*RE(X(K+N2))
F'000C44  3C0D            1297.          MOVE.W A5,D6  ;GET IM(W**P)
F'000C46  4446            1298.          NEG.W D6
F'000C48  CDEB 0002       1299.          MULS 2(A3),D6    ;IM(W**P)*IM(X(K+N2))
F'000C4C  9A86            1300.          SUB.L D6,D5      ;D5=RE(T1)
F'000C4E  E0A5            1301.          ASR.L D0,D5      ;/16384 TO SCALE
F'000C50  3C0D            1302.          MOVE.W A5,D6  ;GET IM(W**P)
F'000C52  4446            1303.          NEG.W D6
F'000C54  CDD3            1304.          MULS (A3),D6     ;IM(W**P)*RE(X(K+N2))
F'000C56  CFEB 0002       1305.          MULS 2(A3),D7    ;RE(W**P)*IM(X(K+N2))
F'000C5A  DC87            1306.          ADD.L D7,D6      ;D6=IM(T1)
F'000C5C  E0A6            1307.          ASR.L D0,D6      ;/16384 TO SCALE
F'000C5E  3E16            1308.          MOVE.W (A6),D7   ;RE(X(K))
F'000C60  9E45            1309.          SUB.W D5,D7      ;RE(X(K))-RE(T1)
F'000C62  3687            1310.          MOVE.W D7,(A3)   ;STORE ANSWER
F'000C64  3E2E 0002       1311.          MOVE.W 2(A6),D7  ;IM(X(K))
F'000C68  9E46            1312.          SUB.W D6,D7      ;IM(X(K))-IM(T1)
F'000C6A  3747 0002       1313.          MOVE.W D7,2(A3)  ;STORE ANSWER
F'000C6E  DB56            1314.          ADD.W D5,(A6)    ;RE(X(K))+RE(T1)
F'000C70  DD6E 0002       1315.          ADD.W D6,2(A6)   ;IM(X(K);+IM(T1)
F'000C74  5243            1316.          ADDQ.W #1,D3     ;INCREMENT K
F'000C76  51CC FF9E       1317.          DBRA D4,LPIN2R   ;DO TILL I=-1
F'000C7A  D641            1318.          ADD.W D1,D3      ;K=K+N2
F'000C7C  0C43'FFFF       1319.          CMPI.W #NPTE-1,D3 ;K<N-1 ?
F'000C80  6D 90           1320.          BLT INITI2R      ;DO AGAIN IF SO
F'000C82  4243            1321.          CLR.W D3         ;K=0
F'000C84  5342            1322.          SUBQ.W #1,D2     ;NU1=NU1-1
F'000C86  E341            1323.          ASL.W #1,D1      ;N2=N2*2
F'000C88  5257            1324.          ADDQ.W #1,(SP)
F'000C8A  0C57 0007       1325.          CMPI.W #NUGAM+1,(SP)
F'000C8E  66 82           1326.          BNE INITI2R    ;DO TILL L=-1
F'000C90  301F            1327.          MOVE.W (SP)+,D0
                          1328.  *
                          1329.  * ASSERT SRQ AND WAIT FOR ACKNOWLEDGE
```

```
                                1330.  *
F'000C92  4E43                  1331.          TRAP #SRQASRT
                                1332.  * SEND OUT 64 COMPLEX POINTS CALCULATED
>>> WARNING line[1333] Expr. type    [1333] FSTSLVNM.A68
F'000C94  303C ****             1333.          MOVE.W #NPTE*4,D0
F'000C98  13FC 00'00            1334.          MOVE.B #ENABLEB,PGCR
          '00000000
F'000CA0  4E46                  1335.          TRAP #OUTDATA
F'000CA2  4239'00000000         1336.          CLR.B PGCR
                                1337.  *
                                1338.  * NOW DO RTS AND RETURN TO READY FOR DATA
                                1339.  * STATE TO DO OVER!
F'000CA8  4E75                  1340.          RTS
                                1341.  *
                                1342.  * DONE!
                                1343.  *
                                1344.  * SIN (-2*PI*I/64)*16384 FOR I=0 TO 63
                                1345.  *
F'000CAA  0000 F9BA F384        1346.  SINBLK2 DC.W       0, -1606, -3196, -4756
          ED6C
F'000CB2  E782 E1D5 DC72        1347.          DC.W   -6270, -7723, -9102,-10394
          D766
F'000CBA  D2BF CE87 CAC9        1348.          DC.W  -11585,-12665,-13623,-14449
          C78F
F'000CC2  C4DF C2C1 C13B        1349.          DC.W  -15137,-15679,-16069,-16305
          C04F
F'000CCA  C000 C04F C13B        1350.          DC.W  -16384,-16305,-16069,-15679
          C2C1
F'000CD2  C4DF C78F CAC9        1351.          DC.W  -15137,-14449,-13623,-12665
          CE87
F'000CDA  D2BF D766 DC72        1352.          DC.W  -11585,-10394, -9102, -7723
          E1D5
F'000CE2  E782 ED6C F384        1353.          DC.W   -6270, -4756, -3196, -1606
          F9BA
F'000CEA  0000 0646 0C7C        1354.          DC.W       0,  1606,  3196,  4756
          1294
F'000CF2  187E 1E2B 238E        1355.          DC.W    6270,  7723,  9102, 10394
          289A
F'000CFA  2D41 3179 3537        1356.          DC.W   11585, 12665, 13623, 14449
          3871
F'000D02  3B21 3D3F 3EC5        1357.          DC.W   15137, 15679, 16069, 16305
          3FB1
F'000D0A  4000 3FB1 3EC5        1358.          DC.W   16384, 16305, 16069, 15679
          3D3F
F'000D12  3B21 3871 3537        1359.          DC.W   15137, 14449, 13623, 12665
          3179
F'000D1A  2D41 289A 238E        1360.          DC.W   11585, 10394,  9102,  7723
          1E2B
F'000D22  187E 1294 0C7C        1361.          DC.W    6270,  4756,  3196,  1606
          0646
                                1362.  *
                                1363.  * COS(-2*PI*I/64)*16384 FOR I=0 TO 63
                                1364.  *
F'000D2A  4000 3FB1 3EC5        1365.  COSBLK2 DC.W   16384, 16305, 16069, 15679
          3D3F
F'000D32  3B21 3871 3537        1366.          DC.W   15137, 14449, 13623, 12665
          3179
F'000D3A  2D41 289A 238E        1367.          DC.W   11585, 10394,  9102,  7723
          1E2B
F'000D42  187E 1294 0C7C        1368.          DC.W    6270,  4756,  3196,  1606
          0646
F'000D4A  0000 F9BA F384        1369.          DC.W       0, -1606, -3196, -4756
          ED6C
F'000D52  E782 E1D5 DC72        1370.          DC.W   -6270, -7723, -9102,-10394
          D766
F'000D5A  D2BF CE87 CAC9        1371.          DC.W  -11585,-12665,-13623,-14449
          C78F
F'000D62  C4DF C2C1 C13B        1372.          DC.W  -15137,-15679,-16069,-16305
          C04F
F'000D6A  C000 C04F C13B        1373.          DC.W  -16384,-16305,-16069,-15679
          C2C1
F'000D72  C4DF C78F CAC9        1374.          DC.W  -15137,-14449,-13623,-12665
          CE87
F'000D7A  D2BF D766 DC72        1375.          DC.W  -11585,-10394, -9102, -7723
          E1D5
F'000D82  E782 ED6C F384        1376.          DC.W   -6270, -4756, -3196, -1606
          F9BA
F'000D8A  0000 0646 0C7C        1377.          DC.W       0,  1606,  3196,  4756
          1294
F'000D92  187E 1E2B 238E        1378.          DC.W    6270,  7723,  9102, 10394
          289A
```

```
F'000D9A  2D41 3179 3537 1379.          DC.W   11585, 12665, 13623, 14449
          3871
F'000DA2  3B21 3D3F 3EC5 1380.          DC.W   15137, 15679, 16069, 16305
          3FB1
                         1381.  *
                         1382.  * TWIDDLE FACTOR TABLE TWD2
                         1383.  *
F'000DAA  4000 0000 3FEC 1384.  TWD2    DC.W   16384,     0, 16364,  -804
          FCDC
F'000DB2  3FB1 F9BA 3F4F 1385.          DC.W   16305, -1606, 16207, -2404
          F69C
F'000DBA  3EC5 F384 3E15 1386.          DC.W   16069, -3196, 15893, -3981
          F073
F'000DC2  3D3F ED6C 3C42 1387.          DC.W   15679, -4756, 15426, -5520
          EA70
F'000DCA  3B21 E782 39DB 1388.          DC.W   15137, -6270, 14811, -7005
          E4A3
F'000DD2  3871 E1D5 36E5 1389.          DC.W   14449, -7723, 14053, -8423
          DF19
F'000DDA  3537 DC72 3368 1390.          DC.W   13623, -9102, 13160, -9760
          D9E0
F'000DE2  3179 D766 2F6C 1391.          DC.W   12665,-10394, 12140,-11003
          D505
F'000DEA  2D41 D2BF 2AFB 1392.          DC.W   11585,-11585, 11003,-12140
          D094
F'000DF2  289A CE87 2620 1393.          DC.W   10394,-12665,  9760,-13160
          CC98
F'000DFA  238E CAC9 20E7 1394.          DC.W    9102,-13623,  8423,-14053
          C91B
F'000E02  1E2B C78F 1B5D 1395.          DC.W    7723,-14449,  7005,-14811
          C625
F'000E0A  187E C4DF 1590 1396.          DC.W    6270,-15137,  5520,-15426
          C3BE
F'000E12  1294 C2C1 0F8D 1397.          DC.W    4756,-15679,  3981,-15893
          C1EB
F'000E1A  0C7C C13B 0964 1398.          DC.W    3196,-16069,  2404,-16207
          C0B1
F'000E22  0646 C04F 0324 1399.          DC.W    1606,-16305,   804,-16364
          C014
F'000E2A  0000 C000 FCDC 1400.          DC.W       0,-16384,  -804,-16364
          C014
F'000E32  F9BA C04F F69C 1401.          DC.W   -1606,-16305, -2404,-16207
          C0B1
F'000E3A  F384 C13B F073 1402.          DC.W   -3196,-16069, -3981,-15893
          C1EB
F'000E42  ED6C C2C1 EA70 1403.          DC.W   -4756,-15679, -5520,-15426
          C3BE
F'000E4A  E782 C4DF E4A3 1404.          DC.W   -6270,-15137, -7005,-14811
          C625
F'000E52  E1D5 C78F DF19 1405.          DC.W   -7723,-14449, -8423,-14053
          C91B
F'000E5A  DC72 CAC9 D9E0 1406.          DC.W   -9102,-13623, -9760,-13160
          CC98
F'000E62  D766 CE87 D505 1407.          DC.W  -10394,-12665,-11003,-12140
          D094
F'000E6A  D2BF D2BF D094 1408.          DC.W  -11585,-11585,-12140,-11003
          D505
F'000E72  CE87 D766 CC98 1409.          DC.W  -12665,-10394,-13160, -9760
          D9E0
F'000E7A  CAC9 DC72 C91B 1410.          DC.W  -13623, -9102,-14053, -8423
          DF19
F'000E82  C78F E1D5 C625 1411.          DC.W  -14449, -7723,-14811, -7005
          E4A3
F'000E8A  C4DF E782 C3BE 1412.          DC.W  -15137, -6270,-15426, -5520
          EA70
F'000E92  C2C1 ED6C C1EB 1413.          DC.W  -15679, -4756,-15893, -3981
          F073
F'000E9A  C13B F384 C0B1 1414.          DC.W  -16069, -3196,-16207, -2404
          F69C
F'000EA2  C04F F9BA C014 1415.          DC.W  -16305, -1606,-16364,  -804
          FCDC
                         1416.
                         1417.  ********************
                         1418.  * SLAVE 3 PROGRAM *
                         1419.  ********************
                         1420.  *
                         1421.  * INPUT DATA AT DATAII, TWIDDLE FACTORS AT
                         1422.  * TWD3
                         1423.  *
                         1424.  *
F'000EAA  2A7C'00000000  1425.  S3PRO   MOVEA.L #NPE,A5          ;ADDR INC
```

```
F'000EB0  2E3C'00000000  1426.          MOVE.L #SHIFT14,D7       ;/16384 SHFT
F'000EB6  303C'FFFF      1427.          MOVE.W #NPTE-1,D0        ;MAJ LOOP CNTR
F'000EBA  287C'FFFFFFFC  1428.          MOVEA.L #NPT-4,A4        ;INDX TO ARRAYS
F'000EC0  43FA 0444      1429.          LEA TWD3(PC),A1          ;TWID FACT PNTR
F'000EC4  41FA ****      1430.          LEA DATAII(PC),A0        ,XDATA POINTER
F'000EC8  2648           1431.  TWDL3   MOVEA.L A0,A3
F'000ECA  D7CC           1432.          ADDA.L A4,A3
F'000ECC  3213           1433.  A  L3   MOVE.W (A3),D1           ;GET REAL PART
F'000ECE  48C1           1434.          EXT.L D1                 ;REAL SUM IN D1
F'000ED0  362B 0102      1435.          MOVE.W 258(A3),D3
F'000ED4  48C3           1436.          EXT.L D3
F'000ED6  9283           1437.          SUB.L D3,D1
F'000ED8  362B 0200      1438.          MOVE.W 512(A3),D3
F'000EDC  48C3           1439.          EXT.L D3
F'000EDE  9283           1440.          SUB.L D3,D1
F'000EE0  362B 0302      1441.          MOVE.W 770(A3),D3
F'000EE4  48C3           1442.          EXT.L D3
F'000EE6  D283           1443.          ADD.L D3,D1              ;REAL SUM DONE
F'000EE8  342B 0002      1444.          MOVE.W 2(A3),D2          ;IMAG SUM IN D2
F'000EEC  48C2           1445.          EXT.L D2
F'000EEE  362B 0100      1446.          MOVE.W 256(A3),D3
F'000EF2  48C3           1447.          EXT.L D3
F'000EF4  D483           1448.          ADD.L D3,D2
F'000EF6  362B 0202      1449.          MOVE.W 514(A3),D3
F'000EFA  48C3           1450.          EXT.L D3
F'000EFC  9483           1451.          SUB.L D3,D2
F'000EFE  362B 0300      1452.          MOVE.W 768(A3),D3
F'000F02  48C3           1453.          EXT.L D3
F'000F04  9483           1454.          SUB.L D3,D2              ;IMAG SUM DONE
F'000F06  E482           1455.          ASR.L #2,D2              ;ADJUST SUMS
F'000F08  E481           1456.          ASR.L #2,D1              ;ADJUST SUMS
                         1457.  *
                         1458.  * NOW DO TWIDDLE FACTOR MULTIPLY
                         1459.  *
F'000F0A  3A01           1460.          MOVE.W D1,D5
F'000F0C  3831 C800      1461.          MOVE.W 0(A1,A4.L),D4
F'000F10  3631 C802      1462.          MOVE.W 2(A1,A4.L),D3
F'000F14  CBC4           1463.          MULS D4,D5
F'000F16  3C02           1464.          MOVE.W D2,D6
F'000F18  CDC3           1465.          MULS D3,D6
F'000F1A  9A86           1466.          SUB.L D6,D5              ;REAL IN D5
F'000F1C  C3C3           1467.          MULS D3,D1
F'000F1E  C5C4           1468.          MULS D4,D2
F'000F20  D282           1469.          ADD.L D2,D1              ;IMAG IN D1
F'000F22  EEA5           1470.          ASR.L D7,D5
F'000F24  EEA1           1471.          ASR.L D7,D1
                         1472.  *
                         1473.  * STORE SUMMED AND TWIDDLED DATA AT XDATA IN
                         1474.  * PLACE
                         1475.  *
F'000F26  3185 C800      1476.          MOVE.W D5,0(A0,A4.L)
F'000F2A  3181 C802      1477.          MOVE.W D1,2(A0,A4.L)
F'000F2E  99CD           1478.          SUBA.L A5,A4
F'000F30  51C8 FF96      1479.          DBRA D0,TWDL3
                         1480.  *
                         1481.  * INSERT FAVORITE 64-POINT FFT HERE
                         1482.  * INPUT DATA AT XDATA (0-255)
                         1483.  *
                         1484.  *
F'000F34  78'F7          1485.  USHFL3  MOVEQ.L #NPTE-9,D4        ;COUNTER=K
F'000F36  3C04           1486.  SWAPL3  MOVE.W D4,D6             ,PREPARE BIT REV
F'000F38  3E06           1487.  BREV23  MOVE.W D6,D7            ;SAVE DATA
F'000F3A  4246           1488.          CLR.W D6                ;
F'000F3C  7A05           1489.          MOVEQ.L #5,D5           ;BIT COUNTER
F'000F3E  E257           1490.  REV23   ROXR.W #1,D7            ;SFT RIGHT,XTEND
F'000F40  E356           1491.          ROXL.W #1,D6
F'000F42  51CD FFFA      1492.          DBRA D5,REV23
F'000F46  B846           1493.          CMP.W D6,D4              ;I=REV(K)<=K?
F'000F48  6C 14          1494.          BGE.S DECK3               ;NO SWAP IF SO
F'000F4A  3204           1495.          MOVE.W D4,D1             ;COPY OF K
F'000F4C  E541           1496.          ASL.W #2,D1              ;K*4
F'000F4E  E546           1497.          ASL.W #2,D6              ;I*4
F'000F50  2430 1000      1498.          MOVE.L 0(A0,D1.W),D2  ,DO SWAP
F'000F54  21B0 6000 1000 1499.          MOVE.L 0(A0,D6.W),0(A0,D1.W)
```

```
F'000F5A  2182 6000      1500.          MOVE.L D2,0(A0,D6.W) ;DONE SWAP
F'000F5E  51CC FFD6      1501.  DECK3   DBRA D4,SWAPL3
F'000F62  7201           1502.  FFT643  MOVEQ.L #1,D1    ;D1=N2=1
F'000F64  43FA 02A0      1503.          LEA SINBLK3(PC),A1  ;SIN VALS PNTR A1
F'000F68  45FA 031C      1504.          LEA COSBLK3(PC),A2  ;COS VALS PNTR A2
                         1505.  *
                         1506.  * L IS DOWN COUNTED #NUGAM-1 TO 0
                         1507.  *
F'000F6C  3F3C 0001      1508.          MOVE.W #1,-(SP)   ;(SP) IS L
F'000F70  7405           1509.          MOVEQ.L #NUGAM-1,D2    ;D2=NUGAM-1=NU1
F'000F72  4283           1510.          CLR.L D3               ;D3=K=0
F'000F74  70'00          1511.          MOVEQ.L #SHIFT14,D0
                         1512.  *
                         1513.  * INITIALIZE I=N2-1, TO 0 IN STEPS -1
                         1514.  *
F'000F76  3801           1515.  INITI3  MOVE.W D1,D4     ;D4=I=N2
F'000F78  5344           1516.          SUBQ.W #1,D4     ;SET FOR DBRA
                         1517.  *
                         1518.  * DETERMINE P=MOD(K*2**(NUGAM-L),64)
                         1519.  *
                         1520.  *
F'000F7A  3C03           1521.  LOOPIN3 MOVE.W D3,D6     ;D6=K
F'000F7C  3A17           1522.          MOVE.W (SP),D5   ;GET L
F'000F7E  5D45           1523.          SUBQ.W #NUGAM,D5 ;L-NUGAM
F'000F80  4445           1524.          NEG.W D5         ;NUGAM-L
F'000F82  EB66           1525.          ASL.W D5,D6      ;K*2**(NUGAM-L)
F'000F84  0246 003F      1526.          ANDI.W #$003F,D6 ;MOD 64
                         1527.  *
                         1528.  * D6 HAS INDEX REQUIRED
F'000F88  DC46           1529.          ADD.W D6,D6      ;D6*2 FOR WORD BOUND
F'000F8A  3A71 6000      1530.          MOVEA.W 0(A1,D6.W),A5 ;A5 PNTR IM(W**P)
F'000F8E  2E03           1531.          MOVE.L D3,D7     ;D7=K
F'000F90  2A03           1532.          MOVE.L D3,D5     ;D5 TOO
F'000F92  E545           1533.          ASL.W #2,D5      ;K*4 FOR LWORD BOUND
F'000F94  4DF0 5000      1534.          LEA 0(A0,D5.W),A6   ;A6 IS INDEX X(K)
F'000F98  DE41           1535.          ADD.W D1,D7      ;D7=K+N2
F'000F9A  E547           1536.          ASL.W #2,D7      ;D7=4(K+N2) LWORD BND
F'000F9C  47F0 7000      1537.          LEA 0(A0,D7.W),A3 ;A3 PNT RE(X(K+N2))
F'000FA0  3A32 6000      1538.          MOVE.W 0(A2,D6.W),D5    ;GET RE(W**P)
F'000FA4  3E05           1539.          MOVE.W D5,D7     ;PUT ALSO IN D7
F'000FA6  CBD3           1540.          MULS (A3),D5     ;RE(W**P)*RE(X(K+N2))
F'000FA8  3C0D           1541.          MOVE.W A5,D6  ;GET IM(W**P)
F'000FAA  CDEB 0002      1542.          MULS 2(A3),D6    ;IM(W**P)*IM(X(K+N2))
F'000FAE  9A86           1543.          SUB.L D6,D5      ;D5=RE(T1)
F'000FB0  E0A5           1544.          ASR.L D0,D5      ;/16384 TO SCALE
F'000FB2  3C0D           1545.          MOVE.W A5,D6  ;GET IM(W**P)
F'000FB4  CDD3           1546.          MULS (A3),D6     ;IM(W**P)*RE(X(K+N2))
F'000FB6  CFEB 0002      1547.          MULS 2(A3),D7    ;RE(W**P)*IM(X(K+N2))
F'000FBA  DC87           1548.          ADD.L D7,D6      ;D6=IM(T1)
F'000FBC  E0A6           1549.          ASR.L D0,D6      ;/16384 TO SCALE
F'000FBE  3E16           1550.          MOVE.W (A6),D7   ;RE(X(K))
F'000FC0  9E45           1551.          SUB.W D5,D7      ;RE(X(K))-RE(T1)
F'000FC2  E247           1552.          ASR.W #1,D7      ;/2
F'000FC4  3687           1553.          MOVE.W D7,(A3)   ;STORE ANSWER
F'000FC6  3E2E 0002      1554.          MOVE.W 2(A6),D7 ;IM(X(K))
F'000FCA  9E46           1555.          SUB.W D6,D7      ;IM(X(K))-IM(T1)
F'000FCC  E247           1556.          ASR.W #1,D7      ;/2
F'000FCE  3747 0002      1557.          MOVE.W D7,2(A3) ;STORE ANSWER
F'000FD2  DB56           1558.          ADD.W D5,(A6)    ;RE(X(K))+RE(T1)
F'000FD4  E0D6           1559.          ASR.W (A6)       ;/2
F'000FD6  DD6E 0002      1560.          ADD.W D6,2(A6)   ;IM(X(K))+IM(T1)
F'000FDA  E0EE 0002      1561.          ASR.W 2(A6)      ;/2
F'000FDE  5243           1562.          ADDQ.W #1,D3     ;INCREMENT K
F'000FE0  51CC FF98      1563.          DBRA D4,LOOPIN3  ;DO TILL I=-1
F'000FE4  D641           1564.          ADD.W D1,D3      ;K=K+N2
F'000FE6  0C43'FFFF      1565.          CMPI.W #NPTE-1,D3 ;K<N-1 ?
F'000FEA  6D 8A          1566.          BLT INITI3       ;DO AGAIN IF SO
F'000FEC  4243           1567.          CLR.W D3         ;K=0
F'000FEE  5342           1568.          SUBQ.W #1,D2     ;NU1=NU1-1
F'000FF0  E341           1569.          ASL.W #1,D1      ;N2=N2*2
F'000FF2  5257           1570.          ADDQ.W #1,(SP)
F'000FF4  0C57 0007      1571.          CMPI.W #NUGAM+1,(SP)
F'000FF8  6600 FF7C      1572.          BNE INITI3
F'000FFC  301F           1573.          MOVE.W (SP)+,D0
                         1574.  *
                         1575.  * NOW DO FILTERING, FILTER DATA IS AT "FILTER"
                         1576.  * 64 POINTS
                         1577.  *
                         1578.  * DO COMPLEX MULTIPLY BY THE FILTER.
                         1579.  * FILTER CAN BE +16384 TO -16384 WITH 16384
                         1580.  * BEING THE 0 db GAIN VALUE.
```

```
                                1581.  * (A+iB)*(C+iD)
                                1582.  *
                                1583.  * PUT AT XDATA3
                                1584.  *
F'000FFE  7C04                  1585.          MOVEQ.L #4,D6           ;ADDRESS INC
F'001000  45FA ****             1586.          LEA XDATA3(PC),A2
F'001004  43FA ****             1587.          LEA FILTER(PC),A1       ;A1 PNT FILTER
F'001008  303C'FFFF             1588.          MOVE.W #NPTE-1,D0       ;LOOP COUNTER
F'00100C  7A'00                 1589.          MOVEQ.L #SHIFT14,D5     ;SHIFT COUNT
F'00100E  3219                  1590.  FILOOP3 MOVE.W (A1)+,D1         ;GET C
F'001010  3601                  1591.          MOVE.W D1,D3
F'001012  C3D8                  1592.          MULS (A0)+,D1           ;DO AC
F'001014  3419                  1593.          MOVE.W (A1)+,D2         ;GET D
F'001016  3802                  1594.          MOVE.W D2,D4
F'001018  C5D0                  1595.          MULS (A0),D2            ;DO BD
F'00101A  9282                  1596.          SUB.L D2,D1             ;AC-BD
F'00101C  EAA1                  1597.          ASR.L D5,D1             ;ADJUST /16384
F'00101E  C7D0                  1598.          MULS (A0),D3            ;DO BC
F'001020  C9E0                  1599.          MULS -(A0),D4           ;DO AD
F'001022  D883                  1600.          ADD.L D3,D4             ;BC+AD
F'001024  EAA4                  1601.          ASR.L D5,D4             ;ADJUST /16384
F'001026  34C1                  1602.          MOVE.W D1,(A2)+ ;STORE REAL
F'001028  34C4                  1603.          MOVE.W D4,(A2)+ ;STORE IMAG
F'00102A  D1C6                  1604.          ADDA.L D6,A0
F'00102C  51C8 FFE0             1605.          DBRA D0,FILOOP3
                                1606.  *
                                1607.  * ASSERT SRQ AND WAIT FOR ACKNOWLEDGE
                                1608.  *
F'001030  4E43                  1609.          TRAP #SRQASRT
                                1610.  *
                                1611.  * RECEIVE DATA FROM SLAVE #0
                                1612.  *
>>> WARNING line[1613] Expr. type    [1613] FSTSLVNM.A68
F'001032  303C ****             1613.          MOVE.W #NPTE*4,D0       ;BYTE COUNT
F'001036  41FA ****             1614.          LEA XDATA0(PC),A0
F'00103A  13FC 00'00            1615.          MOVE.B #ENABLEA,PGCR
          '00000000
F'001042  4E45                  1616.          TRAP #INDATA
F'001044  4239'00000000         1617.          CLR.B PGCR
                                1618.  *
                                1619.  * ASSERT AN SRQ TO SIGNAL READY FOR DATA
                                1620.  * FROM SLAVE #1 AND RECEIVE; PUT AT XDATA1
                                1621.  * NUMBER OF POINTS IS STILL IN D0
                                1622.  *
F'00104A  4E43                  1623.          TRAP #SRQASRT
                                1624.  *       MOVE.W #NPTE*4,D0       ;BYTE COUNT
F'00104C  41FA ****             1625.          LEA XDATA1(PC),A0
F'001050  13FC 00'00            1626.          MOVE.B #ENABLEA,PGCR
          '00000000
F'001058  4E45                  1627.          TRAP #INDATA
F'00105A  4239'00000000         1628.          CLR.B PGCR
                                1629.  *
                                1630.  * ASSERT AN SRQ TO SIGNAL READY FOR DATA
                                1631.  * FROM SLAVE #2 AND RECEIVE; PUT AT XDATA2
                                1632.  * NUMBER OF POINTS IS STILL IN D0
                                1633.  *
F'001060  4E43                  1634.          TRAP #SRQASRT
                                1635.  *       MOVE.W #NPTE*4,D0       ;BYTE COUNT
F'001062  41FA ****             1636.          LEA XDATA2(PC),A0
F'001066  13FC 00'00            1637.          MOVE.B #ENABLEA,PGCR
          '00000000
F'00106E  4E45                  1638.          TRAP #INDATA
F'001070  4239'00000000         1639.          CLR.B PGCR
                                1640.  *
                                1641.  * ASSERT AN SRQ TO SIGNAL READY TO SEND DATA
                                1642.  * TO OTHER SLAVES
                                1643.  * NUMBER OF POINTS IS STILL IN D0
                                1644.  *
F'001076  4E43                  1645.          TRAP #SRQASRT
                                1646.  *       MOVE.W #NPTE*4,D0       ;BYTE COUNT
F'001078  41FA ****             1647.          LEA XDATA3(PC),A0
F'00107C  13FC 00'00            1648.          MOVE.B #ENABLEB,PGCR
          '00000000
F'001084  4E46                  1649.          TRAP #OUTDATA
F'001086  4239'00000000         1650.          CLR.B PGCR
                                1651.  *
                                1652.  * ASSERT AN SRQ TO SIGNIFY DONE
                                1653.  * DATA REDISTRUBUTION
                                1654.  *
F'00108C  4E43                  1655.          TRAP #SRQASRT
                                1656.  *
```

```
                        1657.  * NOW MUST DO DATA REORDERING
                        1658.  * A 4-SHUFFLE FROM XDATA0-3 TO DATAII
                        1659.  * TO PREPARE FOR INVERSE TRANSFORMING
                        1660.  *
F'00108E  267C 000003FC 1661.  REORD3  MOVEA.L #$3FC,A3
F'001094  247C 00000010 1662.          MOVEA.L #$10,A2
F'00109A  41FA ****     1663.          LEA XDATA0(PC),A0
F'00109E  43FA ****     1664.          LEA DATAII(PC),A1
F'0010A2  72'FF         1665.          MOVEQ.L #NPE-1,D1
F'0010A4  7A'FF         1666.  RLOPT3  MOVEQ.L #NPTE-1,D5
F'0010A6  2298          1667.  RLOPN3  MOVE.L (A0)+,(A1)
F'0010A8  D3CA          1668.          ADDA.L A2,A1
F'0010AA  51CD FFFA     1669.          DBRA D5,RLOPN3
F'0010AE  93CB          1670.          SUBA.L A3,A1
F'0010B0  51C9 FFF2     1671.          DBRA D1,RLOPT3
                        1672.  *
                        1673.  * NOW DO THE INVERSE TRANSFORM
                        1674.  * USE SAME SIN AND COS TABLES, NEGATE THE SIN
                        1675.  * ENTRY, AND USE NO ASR #2 FOR DIVISION BY NPTE
                        1676.  * NOTE SIGN CHANGE ON 4-ADDITIONS TOO
                        1677.  *
F'0010B4  2A7C'00000000 1678.  S3PROR  MOVEA.L #NPE,A5          ;ADDR INC
F'0010BA  2E3C'00000000 1679.          MOVE.L #SHIFT14,D7       ;/16384 SHFT
F'0010C0  303C'FFFF     1680.          MOVE.W #NPTE-1,D0        ;MAJ LOOP CNTR
F'0010C4  287C'FFFFFFFC 1681.          MOVEA.L #NPT-4,A4        ;INDX TO ARRAYS
F'0010CA  43FA 023A     1682.          LEA TWD3(PC),A1          ;TWID FACT PNTR
F'0010CE  41FA ****     1683.          LEA DATAII(PC),A0        ;XDATA POINTER
F'0010D2  2648          1684.  TWDL3R  MOVEA.L A0,A3
F'0010D4  D7CC          1685.          ADDA.L A4,A3
F'0010D6  3213          1686.  ADDL3R  MOVE.W (A3),D1           ;GET REAL PART
F'0010D8  362B 0102     1687.          MOVE.W 258(A3),D3
F'0010DC  D243          1688.          ADD.W D3,D1
F'0010DE  362B 0200     1689.          MOVE.W 512(A3),D3
F'0010E2  9243          1690.          SUB.W D3,D1
F'0010E4  362B 0302     1691.          MOVE.W 770(A3),D3
F'0010E8  9243          1692.          SUB.W D3,D1              ;REAL SUM DONE
F'0010EA  342B 0002     1693.          MOVE.W 2(A3),D2          ;IMAG SUM IN D2
F'0010EE  362B 0100     1694.          MOVE.W 256(A3),D3
F'0010F2  9443          1695.          SUB.W D3,D2
F'0010F4  362B 0202     1696.          MOVE.W 514(A3),D3
F'0010F8  9443          1697           SUB.W D3,D2
F'0010FA  362B 0300     1698.          MOVE.W 768(A3),D3
F'0010FE  D443          1699.          ADD.W D3,D2              ;IMAG SUM DONE
                        1700.  *
                        1701.  * NOW DO TWIDDLE FACTOR MULTIPLY
                        1702.  *
F'001100  3A01          1703.          MOVE.W D1,D5
F'001102  3831 C800     1704.          MOVE.W 0(A1,A4.L),D4
F'001106  3631 C802     1705           MOVE.W 2(A1,A4.L),D3
F'00110A  4443          1706.          NEG.W D3
F'00110C  CBC4          1707.          MULS D4,D5
F'00110E  3C02          1708.          MOVE.W D2,D6
F'001110  CDC3          1709.          MULS D3,D6
F'001112  9A86          1710.          SUB.L D6,D5              ;REAL IN D5
F'001114  C3C3          1711.          MULS D3,D1
F'001116  C5C4          1712.          MULS D4,D2
F'001118  D282          1713.          ADD.L D2,D1              ;IMAG IN D1
F'00111A  EEA5          1714.          ASR.L D7,D5
F'00111C  EEA1          1715.          ASR.L D7,D1
                        1716.  *
                        1717.  * STORE SUMMED AND TWIDDLED DATA AT XDATA IN
                        1718.  * PLACE
                        1719.  *
F'00111E  3185 C800     1720.          MOVE.W D5,0(A0,A4.L)
F'001122  3181 C802     1721.          MOVE.W D1,2(A0,A4.L)
F'001126  99CD          1722.          SUBA.L A5,A4
F'001128  51C8 FFA8     1723.          DBRA D0,TWDL3R
                        1724.  *
                        1725.  * INSERT FAVORITE 64-POINT FFT HERE
                        1726.  * INPUT DATA AT XDATA (0-255)
                        1727.  *
                        1728.  *
F'00112C  78'F7         1729.  USHFL3R MOVEQ.L #NPTE-9,D4        ;COUNTER=K
F'00112E  3C04          1730.  SWAPL3R MOVE.W D4,D6          ;PREPARE BIT REV
F'001130  3E06          1731.  BREV23R MOVE.W D6,D7        ;SAVE DATA
F'001132  4246          1732.          CLR.W D6            ;
F'001134  7A05          1733.          MOVEQ.L #5,D5       ;BIT COUNTER
F'001136  E257          1734.  REV23R  ROXR.W #1,D7        ;SFT RIGHT,XTEND
F'001138  E356          1735.          ROXL.W #1,D6
F'00113A  51CD FFFA     1736.          DBRA D5,REV23R
F'00113E  B846          1737.          CMP.W D6,D4          ;I=REV(K)<=K?
```

```
F'001140  6C 14           1738.          BGE.S DECK3R            ;NO SWAP IF SO
F'001142  3204            1739.          MOVE.W D4,D1           ;COPY OF K
F'001144  E541            1740.          ASL.W #2,D1            ;K*4
F'001146  E546            1741.          ASL.W #2,D6            ;I*4
F'001148  2430 1000       1742.          MOVE.L 0(A0,D1.W),D2 ;DO SWAP
F'00114C  21B0 6000 1000  1743.          MOVE.L 0(A0,D6.W),0(A0,D1.W)
F'001152  2182 6000       1744.          MOVE.L D2,0(A0,D6.W) ;DONE SWAP
F'001156  51CC FFD6       1745.  DECK3R  DBRA D4,SWAPL3R
F'00115A  7201            1746.  FFT643R MOVEQ.L #1,D1    ;D1=N2=1
F'00115C  43FA 00A8       1747.          LEA SINBLK3(PC),A1   ;SIN VALS PNTR A1
F'001160  45FA 0124       1748.          LEA COSBLK3(PC),A2   ;COS VALS PNTR A2
                          1749.  *
                          1750.  * L IS DOWN COUNTED #NUGAM-1 TO 0
                          1751.  *
F'001164  3F3C 0001       1752.          MOVE.W #1,-(SP)      ;(SP) IS L
F'001168  7405            1753.          MOVEQ.L #NUGAM-1,D2 ;D2=NUGAM-1=NU1
F'00116A  4283            1754.          CLR.L D3             ;D3=K=0
F'00116C  70'00           1755.          MOVEQ.L #SHIFT14,D0
                          1756.  *
                          1757.  * INITIALIZE I=N2-1, TO 0 IN STEPS -1
                          1758.  *
F'00116E  3801            1759.  INITI3R MOVE.W D1,D4     ;D4=I=N2
F'001170  5344            1760.          SUBQ.W #1,D4     ;SET FOR DBRA
                          1761.  *
                          1762.  * DETERMINE P=MOD(K*2**(NUGAM-L),64)
                          1763.  *
                          1764.  *
F'001172  3C03            1765.  LPIN3R  MOVE.W D3,D6     ;D6=K
F'001174  3A17            1766.          MOVE.W (SP),D5   ;GET L
F'001176  5D45            1767.          SUBQ.W #NUGAM,D5 ;L-NUGAM
F'001178  4445            1768.          NEG.W D5         ;NUGAM-L
F'00117A  EB66            1769.          ASL.W D5,D6      ;K*2**(NUGAM-L)
F'00117C  0246 003F       1770.          ANDI.W #$003F,D6 ;MOD 64
                          1771.  *
                          1772.  * D6 HAS INDEX REQUIRED
F'001180  DC46            1773.          ADD.W D6,D6      ;D6*2 FOR WORD BOUND
F'001182  3A71 6000       1774.          MOVEA.W 0(A1,D6.W),A5 ;A5 PNTR IM(W**P)
F'001186  2E03            1775.          MOVE.L D3,D7     ;D7=K
F'001188  2A03            1776.          MOVE.L D3,D5     ;D5 TOO
F'00118A  E545            1777.          ASL.W #2,D5      ;K*4 FOR LWORD BOUND
F'00118C  4DF0 5000       1778.          LEA 0(A0,D5.W),A6    ;A6 IS INDEX X(K)
F'001190  DE41            1779.          ADD.W D1,D7      ;D7=K+N2
F'001192  E547            1780.          ASL.W #2,D7      ;D7=4(K+N2) LWORD BND
F'001194  47F0 7000       1781.          LEA 0(A0,D7.W),A3 ;A3 PNT RE(X(K+N2))
F'001198  3A32 6000       1782.          MOVE.W 0(A2,D6.W),D5     ;GET RE(W**P)
F'00119C  3E05            1783.          MOVE.W D5,D7     ;PUT ALSO IN D7
F'00119E  CBD3            1784.          MULS (A3),D5     ;RE(W**P)*RE(X(K+N2))
F'0011A0  3C0D            1785.          MOVE.W A5,D6  ;GET IM(W**P)
F'0011A2  4446            1786.          NEG.W D6
F'0011A4  CDEB 0002       1787.          MULS 2(A3),D6    ;IM(W**P)*IM(X(K+N2))
F'0011A8  9A86            1788.          SUB.L D6,D5      ;D5=RE(T1)
F'0011AA  E0A5            1789.          ASR.L D0,D5      ;/16384 TO SCALE
F'0011AC  3C0D            1790.          MOVE.W A5,D6  ;GET IM(W**P)
F'0011AE  4446            1791.          NEG.W D6
F'0011B0  CDD3            1792.          MULS (A3),D6     ;IM(W**P)*RE(X(K+N2))
F'0011B2  CFEB 0002       1793.          MULS 2(A3),D7    ;RE(W**P)*IM(X(K+N2))
F'0011B6  DC87            1794.          ADD.L D7,D6      ;D6=IM(T1)
F'0011B8  E0A6            1795.          ASR.L D0,D6      ;/16384 TO SCALE
F'0011BA  3E16            1796.          MOVE.W (A6),D7   ;RE(X(K))
F'0011BC  9E45            1797.          SUB.W D5,D7      ;RE(X(K))-RE(T1)
F'0011BE  3687            1798.          MOVE.W D7,(A3)   ;STORE ANSWER
F'0011C0  3E2E 0002       1799.          MOVE.W 2(A6),D7 ;IM(X(K))
F'0011C4  9E46            1800.          SUB.W D6,D7      ;IM(X(K))-IM(T1)
F'0011C6  3747 0002       1801.          MOVE.W D7,2(A3) ;STORE ANSWER
F'0011CA  DB56            1802.          ADD.W D5,(A6)    ;RE(X(K))+RE(T1)
F'0011CC  DD6E 0002       1803.          ADD.W D6,2(A6)   ;IM(X(K))+IM(T1)
F'0011D0  5243            1804.          ADDQ.W #1,D3     ;INCREMENT K
F'0011D2  51CC FF9E       1805.          DBRA D4,LPIN3R  ;DO TILL I=-1
F'0011D6  D641            1806.          ADD.W D1,D3      ;K=K+N2
F'0011D8  0C43'FFFF       1807.          CMPI.W #NPTE-1,D3 ;K<N-1 ?
F'0011DC  6D 90           1808.          BLT INITI3R      ;DO AGAIN IF SO
F'0011DE  4243            1809.          CLR.W D3         ;K=0
F'0011E0  5342            1810.          SUBQ.W #1,D2     ;NU1=NU1-1
F'0011E2  E341            1811.          ASL.W #1,D1      ;N2=N2*2
F'0011E4  5257            1812.          ADDQ.W #1,(SP)
F'0011E6  0C57 0007       1813.          CMPI.W #NUGAM+1,(SP)
F'0011EA  66 82           1814.          BNE INITI3R
F'0011EC  301F            1815.          MOVE.W (SP)+,D0
                          1816.  *
                          1817.  * ASSERT SRQ AND WAIT FOR ACKNOWLEDGE
                          1818.  *
```

```
F'0011EE  4E43            1819.          TRAP #SRQASRT
                          1820.  * SEND OUT 64 COMPLEX POINTS CALCULATED
>>> WARNING line[1821] Expr. type   [1821] FSTSLVNM.A68
F'0011F0  303C ****       1821.          MOVE.W #NPTE*4,D0
F'0011F4  13FC 00'00      1822.          MOVE.B #ENABLEB,PGCR
        '00000000
F'0011FC  4E46            1823.          TRAP #OUTDATA
F'0011FE  4239'00000000   1824.          CLR.B PGCR
                          1825.  *
                          1826.  * NOW DO RTS AND RETURN TO READY FOR DATA
                          1827.  * STATE TO DO OVER!
                          1828.  *
F'001204  4E75            1829.          RTS
                          1830.  *
                          1831.  * DONE!
                          1832.  *
                          1833.  * SIN (-2*PI*I/64)*16384 FOR I=0 TO 63
                          1834.  *
F'001206  0000 F9BA F384  1835.  SINBLK3 DC.W       0, -1606, -3196, -4756
          ED6C
F'00120E  E782 E1D5 DC72  1836.          DC.W   -6270, -7723, -9102,-10394
          D766
F'001216  D2BF CE87 CAC9  1837.          DC.W  -11585,-12665,-13623,-14449
          C78F
F'00121E  C4DF C2C1 C13B  1838.          DC.W  -15137,-15679,-16069,-16305
          C04F
F'001226  C000 C04F C13B  1839.          DC.W  -16384,-16305,-16069,-15679
          C2C1
F'00122E  C4DF C78F CAC9  1840.          DC.W  -15137,-14449,-13623,-12665
          CE87
F'001236  D2BF D766 DC72  1841.          DC.W  -11585,-10394, -9102, -7723
          E1D5
F'00123E  E782 ED6C F384  1842.          DC.W   -6270, -4756, -3196, -1606
          F9BA
F'001246  0000 0646 0C7C  1843.          DC.W       0,  1606,  3196,  4756
          1294
F'00124E  187E 1E2B 238E  1844.          DC.W    6270,  7723,  9102, 10394
          289A
F'001256  2D41 3179 3537  1845.          DC.W   11585, 12665, 13623, 14449
          3871
F'00125E  3B21 3D3F 3EC5  1846.          DC.W   15137, 15679, 16069, 16305
          3FB1
F'001266  4000 3FB1 3EC5  1847.          DC.W   16384, 16305, 16069, 15679
          3D3F
F'00126E  3B21 3871 3537  1848.          DC.W   15137, 14449, 13623, 12665
          3179
F'001276  2D41 289A 238E  1849.          DC.W   11585, 10394,  9102,  7723
          1E2B
F'00127E  187E 1294 0C7C  1850.          DC.W    6270,  4756,  3196,  1606
          0646
                          1851.  *
                          1852.  * COS(-2*PI*I/64)*16384 FOR I=0 TO 63
                          1853.  *
F'001286  4000 3FB1 3EC5  1854.  COSBLK3 DC.W   16384, 16305, 16069, 15679
          3D3F
F'00128E  3B21 3871 3537  1855.          DC.W   15137, 14449, 13623, 12665
          3179
F'001296  2D41 289A 238E  1856.          DC.W   11585, 10394,  9102,  7723
          1E2B
F'00129E  187E 1294 0C7C  1857.          DC.W    6270,  4756,  3196,  1606
          0646
F'0012A6  0000 F9BA F384  1858.          DC.W       0, -1606, -3196, -4756
          ED6C
F'0012AE  E782 E1D5 DC72  1859.          DC W   -6270, -7723, -9102,-10394
          D766
F'0012B6  D2BF CE87 CAC9  1860.          DC.W  -11585,-12665,-13623,-14449
          C78F
F'0012BE  C4DF C2C1 C13B  1861.          DC.W  -15137,-15679,-16069,-16305
          C04F
F'0012C6  C000 C04F C13B  1862.          DC.W  -16384,-16305,-16069,-15679
          C2C1
F'0012CE  C4DF C78F CAC9  1863.          DC.W  -15137,-14449,-13623,-12665
          CE87
F'0012D6  D2BF D766 DC72  1864.          DC.W  -11585,-10394, -9102, -7723
          E1D5
F'0012DE  E782 ED6C F384  1865.          DC.W   -6270, -4756, -3196, -1606
          F9BA
F'0012E6  0000 0646 0C7C  1866.          DC.W       0,  1606,  3196,  4756
          1294
F'0012EE  187E 1E2B 238E  1867.          DC.W    6270,  7723,  9102, 10394
          289A
```

```
F'0012F6  2D41 3179 3537 1868.          DC.W   11585, 12665, 13623, 14449
          3871
F'0012FE  3B21 3D3F 3EC5 1869.          DC.W   15137, 15679, 16069, 16305
          3FB1
                         1870.  *
                         1871.  * TWIDDLE FACTOR TABLE 3 TWD3
                         1872.  *
F'001306  4000 0000 3FD4 1873.  TWD3    DC.W   16384,     0, 16340, -1205
          FB4B
F'00130E  3F4F F69C 3E72 1874.          DC.W   16207, -2404, 15986, -3590
          F1FA
F'001316  3D3F ED6C 3BB6 1875.          DC.W   15679, -4756, 15286, -5897
          E8F7
F'00131E  39DB E4A3 37B0 1876.          DC.W   14811, -7005, 14256, -8076
          E074
F'001326  3537 DC72 3274 1877.          DC.W   13623, -9102, 12916,-10080
          D8A0
F'00132E  2F6C D505 2C21 1878.          DC.W   12140,-11003, 11297,-11866
          D1A6
F'001336  289A CE87 24DA 1879.          DC.W   10394,-12665,  9434,-13395
          CBAD
F'00133E  20E7 C91B 1CC6 1880.          DC.W    8423,-14053,  7366,-14635
          C6D5
F'001346  187E C4DF 1413 1881.          DC.W    6270,-15137,  5139,-15557
          C33B
F'00134E  0F8D C1EB 0AF1 1882.          DC.W    3981,-15893,  2801,-16143
          C0F1
F'001356  0646 C04F 0192 1883.          DC.W    1606,-16305,   402,-16379
          C005
F'00135E  FCDC C014 F82A 1884.          DC.W    -804,-16364, -2006,-16261
          C07B
F'001366  F384 C13B EEEE 1885.          DC.W   -3196,-16069, -4370,-15791
          C251
F'00136E  EA70 C3BE E611 1886.          DC.W   -5520,-15426, -6639,-14978
          C57E
F'001376  E1D5 C78F DDC3 1887.          DC.W   -7723,-14449, -8765,-13842
          C9EE
F'00137E  D9E0 CC98 D632 1888.          DC.W   -9760,-13160,-10702,-12406
          CF8A
F'001386  D2BF D2BF CF8A 1889.          DC.W  -11585,-11585,-12406,-10702
          D632
F'00138E  CC98 D9E0 C9EE 1890.          DC.W  -13160, -9760,-13842, -8765
          DDC3
F'001396  C78F E1D5 C57E 1891.          DC.W  -14449, -7723,-14978, -6639
          E611
F'00139E  C3BE EA70 C251 1892.          DC.W  -15426, -5520,-15791, -4370
          EEEE
F'0013A6  C13B F384 C07B 1893.          DC.W  -16069, -3196,-16261, -2006
          F82A
F'0013AE  C014 FCDC C005 1894.          DC.W  -16364,  -804,-16379,   402
          0192
F'0013B6  C04F 0646 C0F1 1895.          DC.W  -16305,  1606,-16143,  2801
          0AF1
F'0013BE  C1EB 0F8D C33B 1896.          DC.W  -15893,  3981,-15557,  5139
          1413
F'0013C6  C4DF 187E C6D5 1897.          DC.W  -15137,  6270,-14635,  7366
          1CC6
F'0013CE  C91B 20E7 CBAD 1898.          DC.W  -14053,  8423,-13395,  9434
          24DA
F'0013D6  CE87 289A D1A6 1899.          DC.W  -12665, 10394,-11866, 11297
          2C21
F'0013DE  D505 2F6C D8A0 1900.          DC.W  -11003, 12140,-10080, 12916
          3274
F'0013E6  DC72 3537 E074 1901.          DC.W   -9102, 13623, -8076, 14256
          37B0
F'0013EE  E4A3 39DB E8F7 1902.          DC.W   -7005, 14811, -5897, 15286
          3BB6
F'0013F6  ED6C 3D3F F1FA 1903.          DC.W   -4756, 15679, -3590, 15986
          3E72
F'0013FE  F69C 3F4F FB4B 1904.          DC.W   -2404, 16207, -1205, 16340
          3FD4
F'001406  <1>            1905.  DONE    DS.B 1
F'001407                 1906.          END
    8 Warnings
    0 Errors
```

M.13 Filter Data: ALLPASS

```
00000000                              1.          ORG.L $0000
00000000 4000 0000 4000 0000          2.          DC.W $4000,$0000,$4000,$0000
00000008 4000 0000 4000 0000          3.          DC.W $4000,$0000,$4000,$0000
00000010 4000 0000 4000 0000          4.          DC.W $4000,$0000,$4000,$0000
00000018 4000 0000 4000 0000          5.          DC.W $4000,$0000,$4000,$0000
00000020 4000 0000 4000 0000          6.          DC.W $4000,$0000,$4000,$0000
00000028 4000 0000 4000 0000          7.          DC.W $4000,$0000,$4000,$0000
00000030 4000 0000 4000 0000          8.          DC.W $4000,$0000,$4000,$0000
00000038 4000 0000 4000 0000          9.          DC.W $4000,$0000,$4000,$0000
00000040 4000 0000 4000 0000         10.          DC.W $4000,$0000,$4000,$0000
00000048 4000 0000 4000 0000         11.          DC.W $4000,$0000,$4000,$0000
00000050 4000 0000 4000 0000         12.          DC.W $4000,$0000,$4000,$0000
00000058 4000 0000 4000 0000         13.          DC.W $4000,$0000,$4000,$0000
00000060 4000 0000 4000 0000         14.          DC.W $4000,$0000,$4000,$0000
00000068 4000 0000 4000 0000         15.          DC.W $4000,$0000,$4000,$0000
00000070 4000 0000 4000 0000         16.          DC.W $4000,$0000,$4000,$0000
00000078 4000 0000 4000 0000         17.          DC.W $4000,$0000,$4000,$0000
00000080 4000 0000 4000 0000         18.          DC.W $4000,$0000,$4000,$0000
00000088 4000 0000 4000 0000         19.          DC.W $4000,$0000,$4000,$0000
00000090 4000 0000 4000 0000         20.          DC.W $4000,$0000,$4000,$0000
00000098 4000 0000 4000 0000         21.          DC.W $4000,$0000,$4000,$0000
000000A0 4000 0000 4000 0000         22.          DC.W $4000,$0000,$4000,$0000
000000A8 4000 0000 4000 0000         23.          DC.W $4000,$0000,$4000,$0000
000000B0 4000 0000 4000 0000         24.          DC.W $4000,$0000,$4000,$0000
000000B8 4000 0000 4000 0000         25.          DC.W $4000,$0000,$4000,$0000
000000C0 4000 0000 4000 0000         26.          DC.W $4000,$0000,$4000,$0000
000000C8 4000 0000 4000 0000         27.          DC.W $4000,$0000,$4000,$0000
000000D0 4000 0000 4000 0000         28.          DC.W $4000,$0000,$4000,$0000
000000D8 4000 0000 4000 0000         29.          DC.W $4000,$0000,$4000,$0000
000000E0 4000 0000 4000 0000         30.          DC.W $4000,$0000,$4000,$0000
000000E8 4000 0000 4000 0000         31.          DC.W $4000,$0000,$4000,$0000
000000F0 4000 0000 4000 0000         32.          DC.W $4000,$0000,$4000,$0000
000000F8 4000 0000 4000 0000         33.          DC.W $4000,$0000,$4000,$0000
00000100 4000 0000 4000 0000         34.          DC.W $4000,$0000,$4000,$0000
00000108 4000 0000 4000 0000         35.          DC.W $4000,$0000,$4000,$0000
00000110 4000 0000 4000 0000         36.          DC.W $4000,$0000,$4000,$0000
00000118 4000 0000 4000 0000         37.          DC.W $4000,$0000,$4000,$0000
00000120 4000 0000 4000 0000         38.          DC.W $4000,$0000,$4000,$0000
00000128 4000 0000 4000 0000         39.          DC.W $4000,$0000,$4000,$0000
00000130 4000 0000 4000              40.          DC.W $4000,$0000,$4000,$0000
```

```
                0000
00000138  4000 0000 4000    41.          DC.W $4000,$0000,$4000,$0000
                0000
00000140  4000 0000 4000    42.          DC.W $4000,$0000,$4000,$0000
                0000
00000148  4000 0000 4000    43.          DC.W $4000,$0000,$4000,$0000
                0000
00000150  4000 0000 4000    44.          DC.W $4000,$0000,$4000,$0000
                0000
00000158  4000 0000 4000    45.          DC.W $4000,$0000,$4000,$0000
                0000
00000160  4000 0000 4000    46.          DC.W $4000,$0000,$4000,$0000
                0000
00000168  4000 0000 4000    47.          DC.W $4000,$0000,$4000,$0000
                0000
00000170  4000 0000 4000    48.          DC.W $4000,$0000,$4000,$0000
                0000
00000178  4000 0000 4000    49.          DC.W $4000,$0000,$4000,$0000
                0000
00000180  4000 0000 4000    50.          DC.W $4000,$0000,$4000,$0000
                0000
00000188  4000 0000 4000    51.          DC.W $4000,$0000,$4000,$0000
                0000
00000190  4000 0000 4000    52.          DC.W $4000,$0000,$4000,$0000
                0000
00000198  4000 0000 4000    53.          DC.W $4000,$0000,$4000,$0000
                0000
000001A0  4000 0000 4000    54.          DC.W $4000,$0000,$4000,$0000
                0000
000001A8  4000 0000 4000    55.          DC.W $4000,$0000,$4000,$0000
                0000
000001B0  4000 0000 4000    56.          DC.W $4000,$0000,$4000,$0000
                0000
000001B8  4000 0000 4000    57.          DC.W $4000,$0000,$4000,$0000
                0000
000001C0  4000 0000 4000    58.          DC.W $4000,$0000,$4000,$0000
                0000
000001C8  4000 0000 4000    59.          DC.W $4000,$0000,$4000,$0000
                0000
000001D0  4000 0000 4000    60.          DC.W $4000,$0000,$4000,$0000
                0000
000001D8  4000 0000 4000    61.          DC.W $4000,$0000,$4000,$0000
                0000
000001E0  4000 0000 4000    62.          DC.W $4000,$0000,$4000,$0000
                0000
000001E8  4000 0000 4000    63.          DC.W $4000,$0000,$4000,$0000
                0000
000001F0  4000 0000 4000    64.          DC.W $4000,$0000,$4000,$0000
                0000
000001F8  4000 0000 4000    65.          DC.W $4000,$0000,$4000,$0000
                0000
00000200  4000 0000 4000    66.          DC.W $4000,$0000,$4000,$0000
                0000
00000208  4000 0000 4000    67.          DC.W $4000,$0000,$4000,$0000
                0000
00000210  4000 0000 4000    68.          DC.W $4000,$0000,$4000,$0000
                0000
00000218  4000 0000 4000    69.          DC.W $4000,$0000,$4000,$0000
                0000
00000220  4000 0000 4000    70.          DC.W $4000,$0000,$4000,$0000
                0000
00000228  4000 0000 4000    71.          DC.W $4000,$0000,$4000,$0000
                0000
00000230  4000 0000 4000    72.          DC.W $4000,$0000,$4000,$0000
                0000
00000238  4000 0000 4000    73.          DC.W $4000,$0000,$4000,$0000
                0000
00000240  4000 0000 4000    74.          DC.W $4000,$0000,$4000,$0000
                0000
00000248  4000 0000 4000    75.          DC.W $4000,$0000,$4000,$0000
                0000
00000250  4000 0000 4000    76.          DC.W $4000,$0000,$4000,$0000
                0000
00000258  4000 0000 4000    77.          DC.W $4000,$0000,$4000,$0000
                0000
00000260  4000 0000 4000    78.          DC.W $4000,$0000,$4000,$0000
                0000
00000268  4000 0000 4000    79.          DC.W $4000,$0000,$4000,$0000
                0000
00000270  4000 0000 4000    80           DC.W $4000,$0000,$4000,$0000
                0000
```

```
00000278  4000 0000 4000 0000    81.          DC.W $4000,$0000,$4000,$0000
00000280  4000 0000 4000 0000    82.          DC.W $4000,$0000,$4000,$0000
00000288  4000 0000 4000 0000    83.          DC.W $4000,$0000,$4000,$0000
00000290  4000 0000 4000 0000    84.          DC.W $4000,$0000,$4000,$0000
00000298  4000 0000 4000 0000    85.          DC.W $4000,$0000,$4000,$0000
000002A0  4000 0000 4000 0000    86.          DC.W $4000,$0000,$4000,$0000
000002A8  4000 0000 4000 0000    87.          DC.W $4000,$0000,$4000,$0000
000002B0  4000 0000 4000 0000    88.          DC.W $4000,$0000,$4000,$0000
000002B8  4000 0000 4000 0000    89.          DC.W $4000,$0000,$4000,$0000
000002C0  4000 0000 4000 0000    90.          DC.W $4000,$0000,$4000,$0000
000002C8  4000 0000 4000 0000    91.          DC.W $4000,$0000,$4000,$0000
000002D0  4000 0000 4000 0000    92.          DC.W $4000,$0000,$4000,$0000
000002D8  4000 0000 4000 0000    93.          DC.W $4000,$0000,$4000,$0000
000002E0  4000 0000 4000 0000    94.          DC.W $4000,$0000,$4000,$0000
000002E8  4000 0000 4000 0000    95.          DC.W $4000,$0000,$4000,$0000
000002F0  4000 0000 4000 0000    96.          DC.W $4000,$0000,$4000,$0000
000002F8  4000 0000 4000 0000    97.          DC.W $4000,$0000,$4000,$0000
00000300  4000 0000 4000 0000    98.          DC.W $4000,$0000,$4000,$0000
00000308  4000 0000 4000 0000    99.          DC.W $4000,$0000,$4000,$0000
00000310  4000 0000 4000 0000   100.          DC.W $4000,$0000,$4000,$0000
00000318  4000 0000 4000 0000   101.          DC.W $4000,$0000,$4000,$0000
00000320  4000 0000 4000 0000   102.          DC.W $4000,$0000,$4000,$0000
00000328  4000 0000 4000 0000   103.          DC.W $4000,$0000,$4000,$0000
00000330  4000 0000 4000 0000   104.          DC.W $4000,$0000,$4000,$0000
00000338  4000 0000 4000 0000   105.          DC.W $4000,$0000,$4000,$0000
00000340  4000 0000 4000 0000   106.          DC.W $4000,$0000,$4000,$0000
00000348  4000 0000 4000 0000   107.          DC.W $4000,$0000,$4000,$0000
00000350  4000 0000 4000 0000   108.          DC.W $4000,$0000,$4000,$0000
00000358  4000 0000 4000 0000   109.          DC.W $4000,$0000,$4000,$0000
00000360  4000 0000 4000 0000   110.          DC.W $4000,$0000,$4000,$0000
00000368  4000 0000 4000 0000   111.          DC.W $4000,$0000,$4000,$0000
00000370  4000 0000 4000 0000   112.          DC.W $4000,$0000,$4000,$0000
00000378  4000 0000 4000 0000   113.          DC.W $4000,$0000,$4000,$0000
00000380  4000 0000 4000 0000   114.          DC.W $4000,$0000,$4000,$0000
00000388  4000 0000 4000 0000   115.          DC.W $4000,$0000,$4000,$0000
00000390  4000 0000 4000 0000   116.          DC.W $4000,$0000,$4000,$0000
00000398  4000 0000 4000 0000   117.          DC.W $4000,$0000,$4000,$0000
000003A0  4000 0000 4000 0000   118.          DC.W $4000,$0000,$4000,$0000
000003A8  4000 0000 4000 0000   119.          DC.W $4000,$0000,$4000,$0000
000003B0  4000 0000 4000 0000   120.          DC.W $4000,$0000,$4000,$0000
000003B8  4000 0000 4000 0000   121.          DC.W $4000,$0000,$4000,$0000
```

```
                0000
000003C0        4000 0000 4000  122.            DC.W $4000,$0000,$4000,$0000
                0000
000003C8        4000 0000 4000  123.            DC.W $4000,$0000,$4000,$0000
                0000
000003D0        4000 0000 4000  124.            DC.W $4000,$0000,$4000,$0000
                0000
000003D8        4000 0000 4000  125.            DC.W $4000,$0000,$4000,$0000
                0000
000003E0        4000 0000 4000  126.            DC.W $4000,$0000,$4000,$0000
                0000
000003E8        4000 0000 4000  127.            DC.W $4000,$0000,$4000,$0000
                0000
000003F0        4000 0000 4000  128.            DC.W $4000,$0000,$4000,$0000
                0000
000003F8        4000 0000 4000  129.            DC.W $4000,$0000,$4000,$0000
                0000
    0 Errors
```

M.14 Filter Data: NOPASS

```
00000000                          1.        ORG.L $0000
00000000  0000 0000 0000          2.        DC.W $0000,$0000,$0000,$0000
          0000
00000008  0000 0000 0000          3.        DC.W $0000,$0000,$0000,$0000
          0000
00000010  0000 0000 0000          4.        DC.W $0000,$0000,$0000,$0000
          0000
00000018  0000 0000 0000          5.        DC.W $0000,$0000,$0000,$0000
          0000
00000020  0000 0000 0000          6.        DC.W $0000,$0000,$0000,$0000
          0000
00000028  0000 0000 0000          7.        DC.W $0000,$0000,$0000,$0000
          0000
00000030  0000 0000 0000          8         DC.W $0000,$0000,$0000,$0000
          0000
00000038  0000 0000 0000          9.        DC.W $0000,$0000,$0000,$0000
          0000
00000040  0000 0000 0000         10.        DC.W $0000,$0000,$0000,$0000
          0000
00000048  0000 0000 0000         11.        DC.W $0000,$0000,$0000,$0000
          0000
00000050  0000 0000 0000         12.        DC.W $0000,$0000,$0000,$0000
          0000
00000058  0000 0000 0000         13.        DC.W $0000,$0000,$0000,$0000
          0000
00000060  0000 0000 0000         14.        DC.W $0000,$0000,$0000,$0000
          0000
00000068  0000 0000 0000         15.        DC.W $0000,$0000,$0000,$0000
          0000
00000070  0000 0000 0000         16.        DC.W $0000,$0000,$0000,$0000
          0000
00000078  0000 0000 0000         17.        DC.W $0000,$0000,$0000,$0000
          0000
00000080  0000 0000 0000         18.        DC.W $0000,$0000,$0000,$0000
          0000
00000088  0000 0000 0000         19.        DC.W $0000,$0000,$0000,$0000
          0000
00000090  0000 0000 0000         20.        DC.W $0000,$0000,$0000,$0000
          0000
00000098  0000 0000 0000         21.        DC.W $0000,$0000,$0000,$0000
          0000
000000A0  0000 0000 0000         22.        DC.W $0000,$0000,$0000,$0000
          0000
000000A8  0000 0000 0000         23.        DC.W $0000,$0000,$0000,$0000
          0000
000000B0  0000 0000 0000         24.        DC.W $0000,$0000,$0000,$0000
          0000
000000B8  0000 0000 0000         25.        DC.W $0000,$0000,$0000,$0000
          0000
000000C0  0000 0000 0000         26.        DC.W $0000,$0000,$0000,$0000
          0000
000000C8  0000 0000 0000         27.        DC.W $0000,$0000,$0000,$0000
          0000
000000D0  0000 0000 0000         28.        DC.W $0000,$0000,$0000,$0000
          0000
000000D8  0000 0000 0000         29.        DC.W $0000,$0000,$0000,$0000
          0000
000000E0  0000 0000 0000         30.        DC.W $0000,$0000,$0000,$0000
          0000
000000E8  0000 0000 0000         31.        DC.W $0000,$0000,$0000,$0000
          0000
000000F0  0000 0000 0000         32.        DC.W $0000,$0000,$0000,$0000
          0000
000000F8  0000 0000 0000         33.        DC.W $0000,$0000,$0000,$0000
          0000
00000100  0000 0000 0000         34.        DC.W $0000,$0000,$0000,$0000
          0000
00000108  0000 0000 0000         35.        DC.W $0000,$0000,$0000,$0000
          0000
00000110  0000 0000 0000         36.        DC.W $0000,$0000,$0000,$0000
          0000
00000118  0000 0000 0000         37.        DC.W $0000,$0000,$0000,$0000
          0000
00000120  0000 0000 0000         38.        DC.W $0000,$0000,$0000,$0000
          0000
00000128  0000 0000 0000         39.        DC.W $0000,$0000,$0000,$0000
          0000
00000130  0000 0000 0000         40.        DC.W $0000,$0000,$0000,$0000
```

```
                0000
00000138  0000 0000 0000 0000    41.          DC.W $0000,$0000,$0000,$0000
                0000
00000140  0000 0000 0000 0000    42.          DC.W $0000,$0000,$0000,$0000
                0000
00000148  0000 0000 0000 0000    43.          DC.W $0000,$0000,$0000,$0000
                0000
00000150  0000 0000 0000 0000    44.          DC.W $0000,$0000,$0000,$0000
                0000
00000158  0000 0000 0000 0000    45.          DC.W $0000,$0000,$0000,$0000
                0000
00000160  0000 0000 0000 0000    46.          DC.W $0000,$0000,$0000,$0000
                0000
00000168  0000 0000 0000 0000    47.          DC.W $0000,$0000,$0000,$0000
                0000
00000170  0000 0000 0000 0000    48.          DC.W $0000,$0000,$0000,$0000
                0000
00000178  0000 0000 0000 0000    49.          DC.W $0000,$0000,$0000,$0000
                0000
00000180  0000 0000 0000 0000    50.          DC.W $0000,$0000,$0000,$0000
                0000
00000188  0000 0000 0000 0000    51.          DC.W $0000,$0000,$0000,$0000
                0000
00000190  0000 0000 0000 0000    52.          DC.W $0000,$0000,$0000,$0000
                0000
00000198  0000 0000 0000 0000    53.          DC.W $0000,$0000,$0000,$0000
                0000
000001A0  0000 0000 0000 0000    54.          DC.W $0000,$0000,$0000,$0000
                0000
000001A8  0000 0000 0000 0000    55.          DC.W $0000,$0000,$0000,$0000
                0000
000001B0  0000 0000 0000 0000    56.          DC.W $0000,$0000,$0000,$0000
                0000
000001B8  0000 0000 0000 0000    57.          DC.W $0000,$0000,$0000,$0000
                0000
000001C0  0000 0000 0000 0000    58.          DC.W $0000,$0000,$0000,$0000
                0000
000001C8  0000 0000 0000 0000    59.          DC.W $0000,$0000,$0000,$0000
                0000
000001D0  0000 0000 0000 0000    60.          DC.W $0000,$0000,$0000,$0000
                0000
000001D8  0000 0000 0000 0000    61.          DC.W $0000,$0000,$0000,$0000
                0000
000001E0  0000 0000 0000 0000    62.          DC.W $0000,$0000,$0000,$0000
                0000
000001E8  0000 0000 0000 0000    63.          DC.W $0000,$0000,$0000,$0000
                0000
000001F0  0000 0000 0000 0000    64.          DC.W $0000,$0000,$0000,$0000
                0000
000001F8  0000 0000 0000 0000    65.          DC.W $0000,$0000,$0000,$0000
                0000
00000200  0000 0000 0000 0000    66.          DC.W $0000,$0000,$0000,$0000
                0000
00000208  0000 0000 0000 0000    67.          DC.W $0000,$0000,$0000,$0000
                0000
00000210  0000 0000 0000 0000    68.          DC.W $0000,$0000,$0000,$0000
                0000
00000218  0000 0000 0000 0000    69.          DC.W $0000,$0000,$0000,$0000
                0000
00000220  0000 0000 0000 0000    70.          DC.W $0000,$0000,$0000,$0000
                0000
00000228  0000 0000 0000 0000    71.          DC.W $0000,$0000,$0000,$0000
                0000
00000230  0000 0000 0000 0000    72.          DC.W $0000,$0000,$0000,$0000
                0000
00000238  0000 0000 0000 0000    73.          DC.W $0000,$0000,$0000,$0000
                0000
00000240  0000 0000 0000 0000    74.          DC.W $0000,$0000,$0000,$0000
                0000
00000248  0000 0000 0000 0000    75.          DC.W $0000,$0000,$0000,$0000
                0000
00000250  0000 0000 0000 0000    76.          DC.W $0000,$0000,$0000,$0000
                0000
00000258  0000 0000 0000 0000    77.          DC.W $0000,$0000,$0000,$0000
                0000
00000260  0000 0000 0000 0000    78.          DC.W $0000,$0000,$0000,$0000
                0000
00000268  0000 0000 0000 0000    79.          DC.W $0000,$0000,$0000,$0000
                0000
00000270  0000 0000 0000 0000    80.          DC.W $0000,$0000,$0000,$0000
                0000
```

```
00000278  0000 0000 0000 0000   81.          DC.W $0000,$0000,$0000,$0000
00000280  0000 0000 0000 0000   82.          DC.W $0000,$0000,$0000,$0000
00000288  0000 0000 0000 0000   83.          DC.W $0000,$0000,$0000,$0000
00000290  0000 0000 0000 0000   84.          DC.W $0000,$0000,$0000,$0000
00000298  0000 0000 0000 0000   85.          DC.W $0000,$0000,$0000,$0000
000002A0  0000 0000 0000 0000   86.          DC.W $0000,$0000,$0000,$0000
000002A8  0000 0000 0000 0000   87.          DC.W $0000,$0000,$0000,$0000
000002B0  0000 0000 0000 0000   88.          DC.W $0000,$0000,$0000,$0000
000002B8  0000 0000 0000 0000   89.          DC.W $0000,$0000,$0000,$0000
000002C0  0000 0000 0000 0000   90.          DC.W $0000,$0000,$0000,$0000
000002C8  0000 0000 0000 0000   91.          DC.W $0000,$0000,$0000,$0000
000002D0  0000 0000 0000 0000   92.          DC.W $0000,$0000,$0000,$0000
000002D8  0000 0000 0000 0000   93.          DC.W $0000,$0000,$0000,$0000
000002E0  0000 0000 0000 0000   94.          DC.W $0000,$0000,$0000,$0000
000002E8  0000 0000 0000 0000   95.          DC.W $0000,$0000,$0000,$0000
000002F0  0000 0000 0000 0000   96.          DC.W $0000,$0000,$0000,$0000
000002F8  0000 0000 0000 0000   97.          DC.W $0000,$0000,$0000,$0000
00000300  0000 0000 0000 0000   98.          DC.W $0000,$0000,$0000,$0000
00000308  0000 0000 0000 0000   99.          DC.W $0000,$0000,$0000,$0000
00000310  0000 0000 0000 0000  100.          DC.W $0000,$0000,$0000,$0000
00000318  0000 0000 0000 0000  101.          DC.W $0000,$0000,$0000,$0000
00000320  0000 0000 0000 0000  102.          DC.W $0000,$0000,$0000,$0000
00000328  0000 0000 0000 0000  103.          DC.W $0000,$0000,$0000,$0000
00000330  0000 0000 0000 0000  104.          DC.W $0000,$0000,$0000,$0000
00000338  0000 0000 0000 0000  105.          DC.W $0000,$0000,$0000,$0000
00000340  0000 0000 0000 0000  106.          DC.W $0000,$0000,$0000,$0000
00000348  0000 0000 0000 0000  107.          DC.W $0000,$0000,$0000,$0000
00000350  0000 0000 0000 0000  108.          DC.W $0000,$0000,$0000,$0000
00000358  0000 0000 0000 0000  109.          DC.W $0000,$0000,$0000,$0000
00000360  0000 0000 0000 0000  110.          DC.W $0000,$0000,$0000,$0000
00000368  0000 0000 0000 0000  111.          DC.W $0000,$0000,$0000,$0000
00000370  0000 0000 0000 0000  112.          DC.W $0000,$0000,$0000,$0000
00000378  0000 0000 0000 0000  113.          DC.W $0000,$0000,$0000,$0000
00000380  0000 0000 0000 0000  114.          DC.W $0000,$0000,$0000,$0000
00000388  0000 0000 0000 0000  115.          DC.W $0000,$0000,$0000,$0000
00000390  0000 0000 0000 0000  116.          DC.W $0000,$0000,$0000,$0000
00000398  0000 0000 0000 0000  117.          DC.W $0000,$0000,$0000,$0000
000003A0  0000 0000 0000 0000  118.          DC.W $0000,$0000,$0000,$0000
000003A8  0000 0000 0000 0000  119.          DC.W $0000,$0000,$0000,$0000
000003B0  0000 0000 0000 0000  120           DC.W $0000,$0000,$0000,$0000
000003B8  0000 0000 0000 0000  121.          DC.W $0000,$0000,$0000,$0000
```

```
                0000
000003C0        0000 0000 0000  122.            DC.W $0000,$0000,$0000,$0000
                0000
000003C8        0000 0000 0000  123.            DC.W $0000,$0000,$0000,$0000
                0000
000003D0        0000 0000 0000  124.            DC.W $0000,$0000,$0000,$0000
                0000
000003D8        0000 0000 0000  125.            DC.W $0000,$0000,$0000,$0000
                0000
000003E0        0000 0000 0000  126.            DC.W $0000,$0000,$0000,$0000
                0000
000003E8        0000 0000 0000  127.            DC.W $0000,$0000,$0000,$0000
                0000
000003F0        0000 0000 0000  128.            DC.W $0000,$0000,$0000,$0000
                0000
000003F8        0000 0000 0000  129.            DC.W $0000,$0000,$0000,$0000
                0000
    0 Errors
```

M.15 Filter Data: LPHLF

```
00000000                          1.        ORG.L $0000
00000000  4000 0000 4000          2.        DC.W $4000,$0000,$4000,$0000
          0000
00000008  4000 0000 4000          3.        DC.W $4000,$0000,$4000,$0000
          0000
00000010  4000 0000 4000          4.        DC.W $4000,$0000,$4000,$0000
          0000
00000018  4000 0000 4000          5.        DC.W $4000,$0000,$4000,$0000
          0000
00000020  4000 0000 4000          6.        DC.W $4000,$0000,$4000,$0000
          0000
00000028  4000 0000 4000          7.        DC.W $4000,$0000,$4000,$0000
          0000
00000030  4000 0000 4000          8.        DC.W $4000,$0000,$4000,$0000
          0000
00000038  4000 0000 4000          9.        DC.W $4000,$0000,$4000,$0000
          0000
00000040  4000 0000 4000         10.        DC.W $4000,$0000,$4000,$0000
          0000
00000048  4000 0000 4000         11.        DC.W $4000,$0000,$4000,$0000
          0000
00000050  4000 0000 4000         12.        DC.W $4000,$0000,$4000,$0000
          0000
00000058  4000 0000 4000         13.        DC.W $4000,$0000,$4000,$0000
          0000
00000060  4000 0000 4000         14.        DC.W $4000,$0000,$4000,$0000
          0000
00000068  4000 0000 4000         15.        DC.W $4000,$0000,$4000,$0000
          0000
00000070  4000 0000 4000         16.        DC.W $4000,$0000,$4000,$0000
          0000
00000078  4000 0000 4000         17.        DC.W $4000,$0000,$4000,$0000
          0000
00000080  4000 0000 4000         18.        DC.W $4000,$0000,$4000,$0000
          0000
00000088  4000 0000 4000         19.        DC.W $4000,$0000,$4000,$0000
          0000
00000090  4000 0000 4000         20.        DC.W $4000,$0000,$4000,$0000
          0000
00000098  4000 0000 4000         21.        DC.W $4000,$0000,$4000,$0000
          0000
000000A0  4000 0000 4000         22.        DC.W $4000,$0000,$4000,$0000
          0000
000000A8  4000 0000 4000         23.        DC.W $4000,$0000,$4000,$0000
          0000
000000B0  4000 0000 4000         24.        DC.W $4000,$0000,$4000,$0000
          0000
000000B8  4000 0000 4000         25.        DC.W $4000,$0000,$4000,$0000
          0000
000000C0  4000 0000 4000         26.        DC.W $4000,$0000,$4000,$0000
          0000
000000C8  4000 0000 4000         27.        DC.W $4000,$0000,$4000,$0000
          0000
000000D0  4000 0000 4000         28.        DC.W $4000,$0000,$4000,$0000
          0000
000000D8  4000 0000 4000         29.        DC.W $4000,$0000,$4000,$0000
          0000
000000E0  4000 0000 4000         30.        DC.W $4000,$0000,$4000,$0000
          0000
000000E8  4000 0000 4000         31.        DC.W $4000,$0000,$4000,$0000
          0000
000000F0  4000 0000 4000         32.        DC.W $4000,$0000,$4000,$0000
          0000
000000F8  4000 0000 4000         33.        DC.W $4000,$0000,$4000,$0000
          0000
00000100  4000 0000 0000         34.        DC.W $4000,$0000,$0000,$0000
          0000
00000108  0000 0000 0000         35.        DC.W $0000,$0000,$0000,$0000
          0000
00000110  0000 0000 0000         36.        DC.W $0000,$0000,$0000,$0000
          0000
00000118  0000 0000 0000         37.        DC.W $0000,$0000,$0000,$0000
          0000
00000120  0000 0000 0000         38.        DC.W $0000,$0000,$0000,$0000
          0000
00000128  0000 0000 0000         39.        DC.W $0000,$0000,$0000,$0000
          0000
00000130  0000 0000 0000         40.        DC.W $0000,$0000,$0000,$0000
```

```
                0000
00000138        0000 0000 0000    41.           DC.W $0000,$0000,$0000,$0000
                0000
00000140        0000 0000 0000    42.           DC.W $0000,$0000,$0000,$0000
                0000
00000148        0000 0000 0000    43.           DC.W $0000,$0000,$0000,$0000
                0000
00000150        0000 0000 0000    44.           DC.W $0000,$0000,$0000,$0000
                0000
00000158        0000 0000 0000    45.           DC.W $0000,$0000,$0000,$0000
                0000
00000160        0000 0000 0000    46.           DC.W $0000,$0000,$0000,$0000
                0000
00000168        0000 0000 0000    47.           DC.W $0000,$0000,$0000,$0000
                0000
00000170        0000 0000 0000    48.           DC.W $0000,$0000,$0000,$0000
                0000
00000178        0000 0000 0000    49.           DC.W $0000,$0000,$0000,$0000
                0000
00000180        0000 0000 0000    50.           DC.W $0000,$0000,$0000,$0000
                0000
00000188        0000 0000 0000    51.           DC.W $0000,$0000,$0000,$0000
                0000
00000190        0000 0000 0000    52.           DC.W $0000,$0000,$0000,$0000
                0000
00000198        0000 0000 0000    53.           DC.W $0000,$0000,$0000,$0000
                0000
000001A0        0000 0000 0000    54.           DC.W $0000,$0000,$0000,$0000
                0000
000001A8        0000 0000 0000    55.           DC.W $0000,$0000,$0000,$0000
                0000
000001B0        0000 0000 0000    56.           DC.W $0000,$0000,$0000,$0000
                0000
000001B8        0000 0000 0000    57.           DC.W $0000,$0000,$0000,$0000
                0000
000001C0        0000 0000 0000    58.           DC.W $0000,$0000,$0000,$0000
                0000
000001C8        0000 0000 0000    59.           DC.W $0000,$0000,$0000,$0000
                0000
000001D0        0000 0000 0000    60.           DC.W $0000,$0000,$0000,$0000
                0000
000001D8        0000 0000 0000    61.           DC.W $0000,$0000,$0000,$0000
                0000
000001E0        0000 0000 0000    62.           DC.W $0000,$0000,$0000,$0000
                0000
000001E8        0000 0000 0000    63.           DC.W $0000,$0000,$0000,$0000
                0000
000001F0        0000 0000 0000    64.           DC.W $0000,$0000,$0000,$0000
                0000
000001F8        0000 0000 0000    65.           DC.W $0000,$0000,$0000,$0000
                0000
00000200        0000 0000 0000    66.           DC.W $0000,$0000,$0000,$0000
                0000
00000208        0000 0000 0000    67.           DC.W $0000,$0000,$0000,$0000
                0000
00000210        0000 0000 0000    68.           DC.W $0000,$0000,$0000,$0000
                0000
00000218        0000 0000 0000    69.           DC.W $0000,$0000,$0000,$0000
                0000
00000220        0000 0000 0000    70.           DC.W $0000,$0000,$0000,$0000
                0000
00000228        0000 0000 0000    71.           DC.W $0000,$0000,$0000,$0000
                0000
00000230        0000 0000 0000    72.           DC.W $0000,$0000,$0000,$0000
                0000
00000238        0000 0000 0000    73.           DC.W $0000,$0000,$0000,$0000
                0000
00000240        0000 0000 0000    74.           DC.W $0000,$0000,$0000,$0000
                0000
00000248        0000 0000 0000    75.           DC.W $0000,$0000,$0000,$0000
                0000
00000250        0000 0000 0000    76.           DC.W $0000,$0000,$0000,$0000
                0000
00000258        0000 0000 0000    77.           DC.W $0000,$0000,$0000,$0000
                0000
00000260        0000 0000 0000    78.           DC.W $0000,$0000,$0000,$0000
                0000
00000268        0000 0000 0000    79.           DC.W $0000,$0000,$0000,$0000
                0000
00000270        0000 0000 0000    80.           DC.W $0000,$0000,$0000,$0000
                0000
```

```
00000278  0000 0000 0000 0000    81.          DC.W $0000,$0000,$0000,$0000
00000280  0000 0000 0000 0000    82.          DC.W $0000,$0000,$0000,$0000
00000288  0000 0000 0000 0000    83.          DC.W $0000,$0000,$0000,$0000
00000290  0000 0000 0000 0000    84.          DC.W $0000,$0000,$0000,$0000
00000298  0000 0000 0000 0000    85.          DC.W $0000,$0000,$0000,$0000
000002A0  0000 0000 0000 0000    86.          DC.W $0000,$0000,$0000,$0000
000002A8  0000 0000 0000 0000    87.          DC.W $0000,$0000,$0000,$0000
000002B0  0000 0000 0000 0000    88.          DC.W $0000,$0000,$0000,$0000
000002B8  0000 0000 0000 0000    89.          DC.W $0000,$0000,$0000,$0000
000002C0  0000 0000 0000 0000    90.          DC.W $0000,$0000,$0000,$0000
000002C8  0000 0000 0000 0000    91.          DC.W $0000,$0000,$0000,$0000
000002D0  0000 0000 0000 0000    92.          DC.W $0000,$0000,$0000,$0000
000002D8  0000 0000 0000 0000    93.          DC.W $0000,$0000,$0000,$0000
000002E0  0000 0000 0000 0000    94.          DC.W $0000,$0000,$0000,$0000
000002E8  0000 0000 0000 0000    95.          DC.W $0000,$0000,$0000,$0000
000002F0  0000 0000 0000 0000    96.          DC.W $0000,$0000,$0000,$0000
000002F8  0000 0000 0000 0000    97.          DC.W $0000,$0000,$0000,$0000
00000300  4000 0000 4000 0000    98.          DC.W $4000,$0000,$4000,$0000
00000308  4000 0000 4000 0000    99.          DC.W $4000,$0000,$4000,$0000
00000310  4000 0000 4000 0000   100.          DC.W $4000,$0000,$4000,$0000
00000318  4000 0000 4000 0000   101.          DC.W $4000,$0000,$4000,$0000
00000320  4000 0000 4000 0000   102.          DC.W $4000,$0000,$4000,$0000
00000328  4000 0000 4000 0000   103.          DC.W $4000,$0000,$4000,$0000
00000330  4000 0000 4000 0000   104.          DC.W $4000,$0000,$4000,$0000
00000338  4000 0000 4000 0000   105.          DC.W $4000,$0000,$4000,$0000
00000340  4000 0000 4000 0000   106.          DC.W $4000,$0000,$4000,$0000
00000348  4000 0000 4000 0000   107.          DC.W $4000,$0000,$4000,$0000
00000350  4000 0000 4000 0000   108.          DC.W $4000,$0000,$4000,$0000
00000358  4000 0000 4000 0000   109.          DC.W $4000,$0000,$4000,$0000
00000360  4000 0000 4000 0000   110.          DC.W $4000,$0000,$4000,$0000
00000368  4000 0000 4000 0000   111.          DC.W $4000,$0000,$4000,$0000
00000370  4000 0000 4000 0000   112.          DC.W $4000,$0000,$4000,$0000
00000378  4000 0000 4000 0000   113.          DC.W $4000,$0000,$4000,$0000
00000380  4000 0000 4000 0000   114.          DC.W $4000,$0000,$4000,$0000
00000388  4000 0000 4000 0000   115.          DC.W $4000,$0000,$4000,$0000
00000390  4000 0000 4000 0000   116.          DC.W $4000,$0000,$4000,$0000
00000398  4000 0000 4000 0000   117.          DC.W $4000,$0000,$4000,$0000
000003A0  4000 0000 4000 0000   118.          DC.W $4000,$0000,$4000,$0000
000003A8  4000 0000 4000 0000   119.          DC.W $4000,$0000,$4000,$0000
000003B0  4000 0000 4000 0000   120.          DC.W $4000,$0000,$4000,$0000
000003B8  4000 0000 4000 0000   121.          DC.W $4000,$0000,$4000,$0000
```

```
                0000
000003C0        4000 0000 4000  122.            DC.W $4000,$0000,$4000,$0000
                0000
000003C8        4000 0000 4000  123.            DC.W $4000,$0000,$4000,$0000
                0000
000003D0        4000 0000 4000  124.            DC.W $4000,$0000,$4000,$0000
                0000
000003D8        4000 0000 4000  125.            DC.W $4000,$0000,$4000,$0000
                0000
000003E0        4000 0000 4000  126.            DC.W $4000,$0000,$4000,$0000
                0000
000003E8        4000 0000 4000  127.            DC.W $4000,$0000,$4000,$0000
                0000
000003F0        4000 0000 4000  128.            DC.W $4000,$0000,$4000,$0000
                0000
000003F8        4000 0000 4000  129.            DC.W $4000,$0000,$4000,$0000
                0000
    0 Errors
```

M.16 Filter Data: HPHLF

```
00000000                          1          ORG.L $0000
00000000  0000 0000 0000          2.         DC.W $0000,$0000,$0000,$0000
          0000
00000008  0000 0000 0000          3.         DC.W $0000,$0000,$0000,$0000
          0000
00000010  0000 0000 0000          4.         DC.W $0000,$0000,$0000,$0000
          0000
00000018  0000 0000 0000          5.         DC.W $0000,$0000,$0000,$0000
          0000
00000020  0000 0000 0000          6.         DC.W $0000,$0000,$0000,$0000
          0000
00000028  0000 0000 0000          7.         DC.W $0000,$0000,$0000,$0000
          0000
00000030  0000 0000 0000          8.         DC.W $0000,$0000,$0000,$0000
          0000
00000038  0000 0000 0000          9.         DC.W $0000,$0000,$0000,$0000
          0000
00000040  0000 0000 0000         10.         DC.W $0000,$0000,$0000,$0000
          0000
00000048  0000 0000 0000         11.         DC.W $0000,$0000,$0000,$0000
          0000
00000050  0000 0000 0000         12.         DC.W $0000,$0000,$0000,$0000
          0000
00000058  0000 0000 0000         13.         DC.W $0000,$0000,$0000,$0000
          0000
00000060  0000 0000 0000         14.         DC.W $0000,$0000,$0000,$0000
          0000
00000068  0000 0000 0000         15.         DC.W $0000,$0000,$0000,$0000
          0000
00000070  0000 0000 0000         16.         DC.W $0000,$0000,$0000,$0000
          0000
00000078  0000 0000 0000         17.         DC.W $0000,$0000,$0000,$0000
          0000
00000080  0000 0000 0000         18.         DC.W $0000,$0000,$0000,$0000
          0000
00000088  0000 0000 0000         19.         DC.W $0000,$0000,$0000,$0000
          0000
00000090  0000 0000 0000         20.         DC.W $0000,$0000,$0000,$0000
          0000
00000098  0000 0000 0000         21.         DC.W $0000,$0000,$0000,$0000
          0000
000000A0  0000 0000 0000         22.         DC.W $0000,$0000,$0000,$0000
          0000
000000A8  0000 0000 0000         23.         DC.W $0000,$0000,$0000,$0000
          0000
000000B0  0000 0000 0000         24.         DC.W $0000,$0000,$0000,$0000
          0000
000000B8  0000 0000 0000         25.         DC.W $0000,$0000,$0000,$0000
          0000
000000C0  0000 0000 0000         26.         DC.W $0000,$0000,$0000,$0000
          0000
000000C8  0000 0000 0000         27.         DC.W $0000,$0000,$0000,$0000
          0000
000000D0  0000 0000 0000         28.         DC.W $0000,$0000,$0000,$0000
          0000
000000D8  0000 0000 0000         29.         DC.W $0000,$0000,$0000,$0000
          0000
000000E0  0000 0000 0000         30.         DC.W $0000,$0000,$0000,$0000
          0000
000000E8  0000 0000 0000         31.         DC.W $0000,$0000,$0000 $0000
          0000
000000F0  0000 0000 0000         32.         DC.W $0000,$0000,$0000,$0000
          0000
000000F8  0000 0000 0000         33.         DC.W $0000,$0000,$0000,$0000
          0000
00000100  4000 0000 4000         34.         DC.W $4000,$0000,$4000,$0000
          0000
00000108  4000 0000 4000         35.         DC.W $4000,$0000,$4000,$0000
          0000
00000110  4000 0000 4000         36.         DC.W $4000,$0000,$4000,$0000
          0000
00000118  4000 0000 4000         37.         DC.W $4000,$0000,$4000,$0000
          0000
00000120  4000 0000 4000         38.         DC.W $4000,$0000,$4000,$0000
          0000
00000128  4000 0000 4000         39.         DC.W $4000,$0000,$4000,$0000
          0000
00000130  4000 0000 4000         40.         DC.W $4000,$0000,$4000,$0000
```

```
                0000
00000138  4000 0000 4000    41.          DC.W $4000,$0000,$4000,$0000
                0000
00000140  4000 0000 4000    42.          DC.W $4000,$0000,$4000,$0000
                0000
00000148  4000 0000 4000    43.          DC.W $4000,$0000,$4000,$0000
                0000
00000150  4000 0000 4000    44.          DC.W $4000,$0000,$4000,$0000
                0000
00000158  4000 0000 4000    45.          DC.W $4000,$0000,$4000,$0000
                0000
00000160  4000 0000 4000    46.          DC.W $4000,$0000,$4000,$0000
                0000
00000168  4000 0000 4000    47.          DC.W $4000,$0000,$4000,$0000
                0000
00000170  4000 0000 4000    48.          DC.W $4000,$0000,$4000,$0000
                0000
00000178  4000 0000 4000    49.          DC.W $4000,$0000,$4000,$0000
                0000
00000180  4000 0000 4000    50.          DC.W $4000,$0000,$4000,$0000
                0000
00000188  4000 0000 4000    51.          DC.W $4000,$0000,$4000,$0000
                0000
00000190  4000 0000 4000    52.          DC.W $4000,$0000,$4000,$0000
                0000
00000198  4000 0000 4000    53.          DC.W $4000,$0000,$4000,$0000
                0000
000001A0  4000 0000 4000    54.          DC.W $4000,$0000,$4000,$0000
                0000
000001A8  4000 0000 4000    55.          DC.W $4000,$0000,$4000,$0000
                0000
000001B0  4000 0000 4000    56.          DC.W $4000,$0000,$4000,$0000
                0000
000001B8  4000 0000 4000    57.          DC.W $4000,$0000,$4000,$0000
                0000
000001C0  4000 0000 4000    58.          DC.W $4000,$0000,$4000,$0000
                0000
000001C8  4000 0000 4000    59.          DC.W $4000,$0000,$4000,$0000
                0000
000001D0  4000 0000 4000    60.          DC.W $4000,$0000,$4000,$0000
                0000
000001D8  4000 0000 4000    61.          DC.W $4000,$0000,$4000,$0000
                0000
000001E0  4000 0000 4000    62.          DC.W $4000,$0000,$4000,$0000
                0000
000001E8  4000 0000 4000    63.          DC.W $4000,$0000,$4000,$0000
                0000
000001F0  4000 0000 4000    64.          DC.W $4000,$0000,$4000,$0000
                0000
000001F8  4000 0000 4000    65.          DC.W $4000,$0000,$4000,$0000
                0000
00000200  4000 0000 4000    66.          DC.W $4000,$0000,$4000,$0000
                0000
00000208  4000 0000 4000    67.          DC.W $4000,$0000,$4000,$0000
                0000
00000210  4000 0000 4000    68.          DC.W $4000,$0000,$4000,$0000
                0000
00000218  4000 0000 4000    69.          DC.W $4000,$0000,$4000,$0000
                0000
00000220  4000 0000 4000    70.          DC.W $4000,$0000,$4000,$0000
                0000
00000228  4000 0000 4000    71.          DC.W $4000,$0000,$4000,$0000
                0000
00000230  4000 0000 4000    72.          DC.W $4000,$0000,$4000,$0000
                0000
00000238  4000 0000 4000    73.          DC.W $4000,$0000,$4000,$0000
                0000
00000240  4000 0000 4000    74.          DC.W $4000,$0000,$4000,$0000
                0000
00000248  4000 0000 4000    75.          DC.W $4000,$0000,$4000,$0000
                0000
00000250  4000 0000 4000    76.          DC.W $4000,$0000,$4000,$0000
                0000
00000258  4000 0000 4000    77.          DC.W $4000,$0000,$4000,$0000
                0000
00000260  4000 0000 4000    78.          DC.W $4000,$0000,$4000,$0000
                0000
00000268  4000 0000 4000    79.          DC.W $4000,$0000,$4000,$0000
                0000
00000270  4000 0000 4000    80.          DC.W $4000,$0000,$4000,$0000
                0000
```

```
00000278 4000 0000 4000 0000    81.          DC.W $4000,$0000,$4000,$0000
00000280 4000 0000 4000 0000    82.          DC.W $4000,$0000,$4000,$0000
00000288 4000 0000 4000 0000    83.          DC.W $4000,$0000,$4000,$0000
00000290 4000 0000 4000 0000    84.          DC.W $4000,$0000,$4000,$0000
00000298 4000 0000 4000 0000    85.          DC.W $4000,$0000,$4000,$0000
000002A0 4000 0000 4000 0000    86.          DC.W $4000,$0000,$4000,$0000
000002A8 4000 0000 4000 0000    87.          DC.W $4000,$0000,$4000,$0000
000002B0 4000 0000 4000 0000    88.          DC.W $4000,$0000,$4000,$0000
000002B8 4000 0000 4000 0000    89.          DC.W $4000,$0000,$4000,$0000
000002C0 4000 0000 4000 0000    90.          DC.W $4000,$0000,$4000,$0000
000002C8 4000 0000 4000 0000    91.          DC.W $4000,$0000,$4000,$0000
000002D0 4000 0000 4000 0000    92.          DC.W $4000,$0000,$4000,$0000
000002D8 4000 0000 4000 0000    93.          DC.W $4000,$0000,$4000,$0000
000002E0 4000 0000 4000 0000    94.          DC.W $4000,$0000,$4000,$0000
000002E8 4000 0000 4000 0000    95.          DC.W $4000,$0000,$4000,$0000
000002F0 4000 0000 4000 0000    96.          DC.W $4000,$0000,$4000,$0000
000002F8 4000 0000 4000 0000    97.          DC.W $4000,$0000,$4000,$0000
00000300 4000 0000 0000 0000    98.          DC.W $4000,$0000,$0000,$0000
00000308 0000 0000 0000 0000    99.          DC.W $0000,$0000,$0000,$0000
00000310 0000 0000 0000 0000   100.          DC.W $0000,$0000,$0000,$0000
00000318 0000 0000 0000 0000   101.          DC.W $0000,$0000,$0000,$0000
00000320 0000 0000 0000 0000   102.          DC.W $0000,$0000,$0000,$0000
00000328 0000 0000 0000 0000   103.          DC.W $0000,$0000,$0000,$0000
00000330 0000 0000 0000 0000   104.          DC.W $0000,$0000,$0000,$0000
00000338 0000 0000 0000 0000   105.          DC.W $0000,$0000,$0000,$0000
00000340 0000 0000 0000 0000   106.          DC.W $0000,$0000,$0000,$0000
00000348 0000 0000 0000 0000   107.          DC.W $0000,$0000,$0000,$0000
00000350 0000 0000 0000 0000   108.          DC.W $0000,$0000,$0000,$0000
00000358 0000 0000 0000 0000   109.          DC.W $0000,$0000,$0000,$0000
00000360 0000 0000 0000 0000   110.          DC.W $0000,$0000,$0000,$0000
00000368 0000 0000 0000 0000   111.          DC.W $0000,$0000,$0000,$0000
00000370 0000 0000 0000 0000   112.          DC.W $0000,$0000,$0000,$0000
00000378 0000 0000 0000 0000   113.          DC.W $0000,$0000,$0000,$0000
00000380 0000 0000 0000 0000   114.          DC.W $0000,$0000,$0000,$0000
00000388 0000 0000 0000 0000   115.          DC.W $0000,$0000,$0000,$0000
00000390 0000 0000 0000 0000   116.          DC.W $0000,$0000,$0000,$0000
00000398 0000 0000 0000 0000   117.          DC.W $0000,$0000,$0000,$0000
000003A0 0000 0000 0000 0000   118.          DC.W $0000,$0000,$0000,$0000
000003A8 0000 0000 0000 0000   119.          DC.W $0000,$0000,$0000,$0000
000003B0 0000 0000 0000 0000   120.          DC.W $0000,$0000,$0000,$0000
000003B8 0000 0000 0000 0000   121.          DC.W $0000,$0000,$0000,$0000
```

```
                0000
000003C0        0000 0000 0000  122.            DC.W $0000,$0000,$0000,$0000
                0000
000003C8        0000 0000 0000  123.            DC.W $0000,$0000,$0000,$0000
                0000
000003D0        0000 0000 0000  124.            DC.W $0000,$0000,$0000,$0000
                0000
000003D8        0000 0000 0000  125.            DC.W $0000,$0000,$0000,$0000
                0000
000003E0        0000 0000 0000  126.            DC.W $0000,$0000,$0000,$0000
                0000
000003E8        0000 0000 0000  127.            DC.W $0000,$0000,$0000,$0000
                0000
000003F0        0000 0000 0000  128.            DC.W $0000,$0000,$0000,$0000
                0000
000003F8        0000 0000 0000  129.            DC.W $0000,$0000,$0000,$0000
                0000
    0 Errors
```

Appendix N: Alternating Series Performance Test Programs

N.1 Program PI4INFO — Uniprocessor Alternating Series Test Program, Optimized sub-tasks, measured sub-task periods.

```
                                    1.          TTL PI4INFO
                                    2.  *
                                    3.  * PROGRAM FOR SINGLE PROCESSOR CALCULATION
                                    4.  * OF PI/4 USING ALTERNATING SERIES
                                    5.  * 0+1-1/3+1/5-1/7+1/9-... FOR A NUMBER
                                    6.  * OF TERMS up to 1/65535
                                    7.  *
                                    8.  * USES GENERAL INVERT SUBROUTINE TO GIVE
                                    9.  * FLOATING POINT REPRESENTATION OF AN
                                   10.  * INTEGER IN RANGE 1<=x<=65535
                                   11.  *
                                   12.  * CALLS A FLOATING POINT ADD ROUTINE
                                   13.  *
                                   14.  * FLOATING POINT REPRESENTATION IS 32-BITS
                                   15.  *
                                   16.  * sign(1), FRACT(23), EXP (8-BITS excess 128)
                                   17.  *
                                   18.  ********************
                                   19.  * APRIL 25,1992
                                   20.  * VERSION HAS USER SPECIFIED # OF TERMS
                                   21.  * OUTPUT IS:
                                   22.  * TERM DENOM/DIV TIME/ADD TIME/PARTIAL SUM
                                   23.  ********************
                                   24.  *
                                   25.  * EQUATES
                                   26.  *
     # 00000007                     27.  SENBYC  EQU 7
     # 0000000D                     28.  CR      EQU $0D
     # 0000001A                     29.  DISBUF2 EQU 26
     # 00000018                     30.  DISBUF8 EQU 24
     # 00000011                     31.  OUTMESC EQU 17
     # 0000000A                     32.  SENCLFC EQU 10
     # 00000004                     33.  EOT     EQU $04
     # 0000001C                     34.  BLDNUM  EQU 28
     # 00FFFFFF                     35.  CNTSTRT EQU $00FFFFFF
     # 00012FD0                     36.  TCR     EQU $12FD0
     # 00000001                     37.  ENABLTM EQU 1
     # 0000000A                     38.  LF      EQU $0A
     # 000000FF                     39.  BGEXP   EQU $FF
     # 0000001F                     40.  SGNMSK  EQU 31
     # 00000080                     41.  BIAS    EQU 128
     # 00000009                     42.  ALSB    EQU 9
     # 00000008                     43.  ANTLSB  EQU 8
     # 00000200                     44.  ARNDDR  EQU $200
     # 00000017                     45.  SIGBITS EQU 23
                                   46.  *
                                   47.  *
00001000                           48.          ORG.L $1000
00001000  2C7C 00012FD0            49.          MOVEA.L #TCR,A6
00001006  4216                     50.          CLR.B (A6)
00001008  2D7C 00FFFFFF            51.          MOVE.L #CNTSTRT,2(A6)
          0002
00001010  4E4F                     52.  PI4     TRAP #15
00001012  000A                     53.          DC.W SENCLFC
00001014  41FA 0115                54.          LEA TERMS(PC),A0
00001018  4E4F                     55.          TRAP #15
0000101A  0011                     56.          DC.W OUTMESC
0000101C  4E4F                     57.          TRAP #15
0000101E  001C                     58.          DC.W BLDNUM
00001020  4E4F                     59.          TRAP #15
00001022  000A                     60.          DC.W SENCLFC
00001024  2F00                     61.          MOVE.L D0,-(SP)
00001026  41FA 0123                62.          LEA HEAD(PC),A0
0000102A  4E4F                     63.          TRAP #15
0000102C  0011                     64.          DC.W OUTMESC
0000102E  4E4F                     65.          TRAP #15
00001030  000A                     66.          DC.W SENCLFC
00001032  201F                     67.          MOVE.L (SP)+,D0
00001034  E280                     68.          ASR.L #1,D0       ;DIVIDE BY 2
00001036  5380                     69.          SUBQ.L #1,D0      ;ADJUST FOR DBRA
00001038  2E00                     70.          MOVE.L D0,D7      ;LOOP COUNTER
0000103A  4280                     71.          CLR.L D0          ;D0 <- PARTIAL SUMS
0000103C  4201                     72.          CLR.B D1          ;
0000103E  7C01                     73.          MOVEQ.L #1,D6     ;D6 <- TERM DENOM
00001040  2F00                     74.  MNLOOP  MOVE.L D0,-(SP)   ;SOME STORAGE BEFORE
00001042  2006                     75.          MOVE.L D6,D0      ;OUTPUT TERM DENOM
```

```
00001044  4E4F            76.           TRAP #15        ;
00001046  0018            77.           DC.W DISBUF8
00001048  41FA 00DC       78.           LEA SPACES(PC),A0 ;SEPARATOR
0000104C  4E4F            79.           TRAP #15
0000104E  0011            80.           DC.W OUTMESC
00001050  201F            81.           MOVE.L (SP)+,D0 ;GET BACK PARTIAL SUM
                          82.   *
                          83.   * START TIME FOR DIVIDE
                          84.   *
00001052  1CBC 0001       85.           MOVE.B #ENABLTM,(A6)
00001056  6100 011E       86.           BSR INVERT      ;RESULT IN D4.L
                          87.   *
                          88.   * STOP TIMER AND OUTPUT ELAPSED
                          89.   *
0000105A  4216            90.           CLR.B (A6)
0000105C  6100 0236       91.           BSR OUTIME
00001060  2F00            92.           MOVE.L D0,-(SP)
00001062  41FA 00C2       93.           LEA SPACES(PC),A0 ;SEPARATOR
00001066  4E4F            94.           TRAP #15
00001068  0011            95.           DC.W OUTMESC
0000106A  201F            96.           MOVE.L (SP)+,D0
                          97.   *
                          98.   * START TIME FOR ADD
                          99.   *
0000106C  2204           100.           MOVE.L D4,D1
0000106E  1CBC 0001      101.           MOVE.B #ENABLTM,(A6)
00001072  6100 0150      102.           BSR ADDFP       ;CALC ODD PARTIAL SUM
                         103.   *
                         104.   * STOP TIMER OUTPUT ELAPSED
                         105.   *
00001076  4216           106.           CLR.B (A6)
00001078  6100 021A      107.           BSR OUTIME
0000107C  2F00           108.           MOVE.L D0,-(SP)
0000107E  41FA 00A6      109.           LEA SPACES(PC),A0 ;SEPARATOR
00001082  4E4F           110.           TRAP #15
00001084  0011           111.           DC.W OUTMESC
00001086  201F           112.           MOVE.L (SP)+,D0
                         113.   *                       ;RETURNED IN D0.L
00001088  4E4F           114.           TRAP #15
0000108A  0018           115.           DC.W DISBUF8    ;OUTPUT SUM
0000108C  48E7 8000      116.           MOVEM.L D0,-(SP)
00001090  41FA 0094      117.           LEA SPACES(PC),A0
00001094  4E4F           118.           TRAP #15
00001096  0011           119.           DC.W OUTMESC
00001098  103C 000D      120.           MOVE.B #CR,D0
0000109C  4E4F           121.           TRAP #15
0000109E  0007           122.           DC.W SENBYC
000010A0  103C 000A      123.           MOVE.B #LF,D0
000010A4  4E4F           124.           TRAP #15
000010A6  0007           125.           DC.W SENBYC
000010A8  4CDF 0001      126.           MOVEM.L (SP)+,D0
000010AC  5446           127.           ADDQ.W #2,D6    ;NEW TERM DENOM
000010AE  2F00           128.           MOVE.L D0,-(SP) ;SOME STORAGE BEFORE
000010B0  2006           129.           MOVE.L D6,D0    ;OUTPUT TERM DENOM
000010B2  4E4F           130.           TRAP #15        ;
000010B4  0018           131.           DC.W DISBUF8
000010B6  41FA 006E      132.           LEA SPACES(PC),A0 ;SEPARATOR
000010BA  4E4F           133.           TRAP #15
000010BC  0011           134.           DC.W OUTMESC
000010BE  201F           135.           MOVE.L (SP)+,D0 ;GET BACK PARTIAL SUM
                         136.   *
                         137.   * START TIMER
                         138.   *
000010C0  1CBC 0001      139.           MOVE.B #ENABLTM,(A6)
000010C4  6100 00B0      140.           BSR INVERT      ;RESULT IN D4.L
                         141.   *
                         142.   * STOP, OUTPUT
                         143.   *
000010C8  4216           144.           CLR.B (A6)
000010CA  6100 01C8      145.           BSR OUTIME
000010CE  2F00           146.           MOVE.L D0,-(SP)
000010D0  41FA 0054      147.           LEA SPACES(PC),A0 ;SEPARATOR
000010D4  4E4F           148.           TRAP #15
000010D6  0011           149.           DC.W OUTMESC
000010D8  201F           150.           MOVE.L (SP)+,D0
000010DA  08C4 001F      151.           BSET.L #SGNMSK,D4       ;NEGATE
000010DE  2204           152.           MOVE.L D4,D1
                         153.   *
                         154.   * START
                         155.   *
000010E0  1CBC 0001      156.           MOVE.B #ENABLTM,(A6)
```

```
000010E4  6100 00DE          157.          BSR ADDFP        ;CALC EVEN PARTIAL SUM
                             158.  *
                             159.  * STOP,OUTPUT
                             160.  *
000010E8  4216               161.          CLR.B (A6)
000010EA  6100 01A8          162.          BSR OUTIME
000010EE  2F00               163.          MOVE.L D0,-(SP)
000010F0  41FA 0034          164.          LEA SPACES(PC),A0 ;SEPARATOR
000010F4  4E4F               165.          TRAP #15
000010F6  0011               166.          DC.W OUTMESC
000010F8  201F               167.          MOVE.L (SP)+,D0
                             168.  *                       ;RETURNED IN D0.L
000010FA  4E4F               169.          TRAP #15
000010FC  00A9               170.          DC.W DISBUF8     ;OUTPUT PARTIAL SUM
000010FE  48E7 8000          171.          MOVEM.L D0,-(SP)
00001102  41FA 0022          172.          LEA SPACES(PC),A0
00001106  4E4F               173.          TRAP #15
00001108  0011               174.          DC.W OUTMESC
0000110A  103C 000D          175.          MOVE.B #CR,D0
0000110E  4E4F               176.          TRAP #15
00001110  0007               177.          DC.W SENBYC      ;
00001112  103C 000A          178.          MOVE.B #LF,D0
00001116  4E4F               179.          TRAP #15
00001118  0007               180.          DC.W SENBYC
0000111A  4CDF 0001          181.          MOVEM.L (SP)+,D0
0000111E  5446               182.          ADDQ.W #2,D6     ;NEW DENOMINATOR
00001120  51CF FF1E          183.          DBRA D7,MNLOOP
00001124  4E75               184.          RTS
                             185.  *
00001126  20 20 20 20 04     186.  SPACES   DC.B '    ',EOT
0000112B  53 70 65 63 69     187.  TERMS    DC.B 'Specify even number of terms > '
          66 79 20 65 76
          65 6E 20 6E 75
          6D 62 65 72 20
          6F 66 20 74 65
          72 6D 73 20 3E
          20
0000114A  04                 188.          DC.B EOT
0000114B  44 65 6E 6F 6D     189.  HEAD     DC.B 'Denom.       Div.time    Add time'
          2E 20 20 20 20
          20 20 44 69 76
          2E 74 69 6D 65
          20 20 20 20 41
          64 64 20 74 69
          6D 65
0000116B  20 20 20 20 20     190.          DC.B '       Sum',EOT
          20 53 75 6D 04
                             191.  *
                             192.  *
                             193.  * SUBROUTINE INVERT
                             194.  * INTEGER TO INVERT TO FP IS IN D6.W
                             195.  * RETURNS FP NUMBER IN D4.L
                             196.  * D6.W UNAFFECTED
                             197.  *
                             198.  * VOLATILE REGISTERS
                             199.  * D2,D3,D4,D5
                             200.  *
          -- boundary align
00001176  7410               201.  INVERT   MOVEQ.L #16,D2  ;D2 <- EXPONENT
00001178  3A06               202.          MOVE.W D6,D5
0000117A  E35D               203.  SHFTLP   ROL.W #1,D5      ;GET MSBIT=1 IN D5
0000117C  55CA FFFC          204.          DBCS D2,SHFTLP
00001180  E25D               205.          ROR.W #1,D5      ;ADJUST ONE BACK
00001182  4402               206.          NEG.B D2         ;NEGATE EXPONENT
00001184  D43C 0080          207.          ADD.B #BIAS,D2   ;BIAS
00001188  7601               208.          MOVEQ.L #1,D3    ;FAST WAY FOR
0000118A  E29B               209.          ROR.L #1,D3      ;MOVE.L #$80000000,D3
0000118C  86C5               210.          DIVU.W D5,D3     ;DO MSW DIVIDE
0000118E  69 26              211.          BVS.S POWR2      ;IF OVRFLW, DIVISOR=2**k
00001190  3803               212.          MOVE.W D3,D4
00001192  4844               213.          SWAP.W D4        ;MSW OF QUO IN D4
00001194  4243               214.          CLR.W D3         ;CLEAR LOW WORD D3
                             215.  *                       ;GIVES EXTENDED REMAINDER
00001196  2603               216.          MOVE.L D3,D3     ;DUMMY MOVE FOR CCR Z
00001198  67 14              217.          BEQ.S LASTEP     ;REM=0? BRANCH, NO ROUND
0000119A  86C5               218.          DIVU.W D5,D3     ;GET LSW OF QUOTIENT
                             219.  *                       ;NEVER OVRFLW, IGNORE REM
0000119C  3803               220.          MOVE.W D3,D4     ;ALL BITS QUOTIENT -> D4
                             221.  *
                             222.  * round the result
                             223.  *
```

```
0000119E  0804 0009       224.  ROUNDQ  BTST.L #ALSB,D4
000011A2  66 0A           225.          BNE.S LASTEP
000011A4  0804 0008       226.          BTST.L #ANTLSB,D4
000011A8  67 04           227.          BEQ.S LASTEP
000011AA  0044 0200       228.          ORI.W #ARNDDR,D4
000011AE  E28C            229.  LASTEP  LSR.L #1,D4      ;SIGN=0
000011B0  5202            230.          ADDQ.B #1,D2
000011B2  1802            231.          MOVE.B D2,D4
000011B4  4E75            232.          RTS
                          233.  *
                          234.  * OVERFLOWED, POWER OF 2 DIVISOR, QUO=DIVISOR*2
                          235.  * NO REMAINDER, NO ROUNDING REQUIRED
                          236.  *
000011B6  4845            237.  POWR2   SWAP D5          ;MSW OF QUO
000011B8  4245            238.          CLR.W D5         ;NO MORE SIG DIGITS
000011BA  E28D            239.          LSR.L #1,D5      ;SIGN=0
000011BC  5402            240.          ADDQ.B #2,D2     ;ADJUST EXPONENT
000011BE  2805            241.          MOVE.L D5,D4
000011C0  1802            242.          MOVE.B D2,D4
000011C2  4E75            243.          RTS
                          244.  *
                          245.  * SUBROUTINE ADDFP
                          246.  *
                          247.  * FLOATING POINT ADDITION FOR 32-BIT
                          248.  * FLOATING POINT FORMAT:
                          249.  * SIGN (1)| FRACTION (23) | EXPONENT (8)
                          250.  * WHERE EXPONENT IS BIASED EXCESS 128;
                          251.  * FRACTION IS NORMALIZED
                          252.  * 0 = $00000000
                          253.  * (SIGN) INFINTY = $(8.OR.0)00000FF
                          254.  * NaN = $XXXXXXXFF
                          255.  *       WHERE XXXXXX IS NOT AS INFINITY
                          256.  *
                          257.  * DOES A*B=C WHERE
                          258.  * A IS IN D0
                          259.  * B IS IN D1
                          260.  * C RETURNS IN D0
                          261.  * IF A OR B = NaN, SETS C=NaN = $FFFFFFFF
                          262.  * IF A OR B = +-INFINITY, SET C=NaN
                          263.  *
000011C4  48E7 3F00       264.  ADDFP   MOVEM.L D2-D7,-(SP)      ;SAVE OFF
                          265.  *       CMPI.B #BGEXP,D0         ;A=NaN OR inf?
                          266.  *       BEQ CNAN                 ;C=NaN IF SO
                          267.  *       CMPI.B #BGEXP,D1         ;B=NaN OR inf?
                          268.  *       BEQ CNAN                 ;C=NaN IF SO
000011C8  4A81            269.          TST.L D1                 ;B=0?
000011CA  6700 0098       270.          BEQ UNSTACK              ;C=A IF SO
000011CE  4A80            271.          TST.L D0                 ;A=0?
000011D0  6700 00AA       272.          BEQ CISB                 ;C=B IF SO
000011D4  4243            273.          CLR.W D3
000011D6  4244            274.          CLR.W D4
000011D8  1600            275.          MOVE.B D0,D3             ;(D3.W)=EXPA
000011DA  1801            276.          MOVE.B D1,D4             ;(D4.W)=EXPB
000011DC  9644            277.          SUB.W D4,D3              ;(D3.W)=SHFTCNT
000011DE  0C43 0017       278.          CMPI.W #SIGBITS,D3       ;TOO BIG? C=A
000011E2  6C00 0080       279.          BGE UNSTACK
000011E6  0C43 FFE9       280.          CMPI.W #-SIGBITS,D3      ;TOO SMALL? C=B
000011EA  6F00 0090       281.          BLE CISB
                          282.  *
                          283.  * HERE, MUST DO ADD
                          284.  * D5, D6 GET FRACTIONS; D7 CARRIES SIGN
                          285.  *
000011EE  7E00            286.          MOVEQ.L #0,D7
000011F0  0800 001F       287.          BTST.L #SGNMSK,D0        ;SIGN OF A
000011F4  67 04           288.          BEQ.S POSTVE
000011F6  08C7 001F       289.          BSET.L #SGNMSK,D7        ;C IS NEG
000011FA  2A00            290.  POSTVE  MOVE.L D0,D5             ;(D5)=AF
000011FC  2C01            291.          MOVE.L D1,D6             ;(D6)=BF
000011FE  4205            292.          CLR.B D5
00001200  4206            293.          CLR.B D6                 ;KILL EXPS
00001202  E38D            294.          LSL.L #1,D5
00001204  E38E            295.          LSL.L #1,D6              ;ALIGN
00001206  7400            296.          MOVEQ.L #0,D2            ;
00001208  1400            297.          MOVE.B D0,D2             ;CEXP=AEXP
0000120A  4A43            298.          TST.W D3                 ;TEST SC
0000120C  67 08           299.          BEQ.S NOADJ              ;IF=0 NO SHFTNG
0000120E  6B00 0070       300.          BMI SHFTA                ;NEG, SHFT A
00001212  E6AE            301.  SHFTB   LSR.L D3,D6              ;SHIFTB
00001214  4206            302.          CLR.B D6
00001216  2600            303.  NOADJ   MOVE.L D0,D3             ;COPY OF A
00001218  B383            304.          EOR.L D1,D3              ;SIGNS
```

```
0000121A  0803 001F      305.          BTST.L #SGNMSK,D3        ;
0000121E  6700 0052      306.          BEQ SAMSGN               ;EQUAL SIGNS?
00001222  4486           307.  DIFFSGN NEG.L D6                 ;NEG B
00001224  DC85           308.          ADD.L D5,D6              ;D6 HAS THE SUM
00001226  6500 0042      309.          BCS NCMPC                ;IF CAR=0, NOT=0
0000122A  0847 001F      310.  COMPC   BCHG.L #SGNMSK,D7        ;COMPLEMENT SUM
0000122E  4486           311.          NEG.L D6
00001230  0806 001F      312.  NORMC   BTST.L #SGNMSK,D6        ;NORMALIZED?
00001234  66 06          313.          BNE.S CNORMD
00001236  E38E           314.          LSL.L #1,D6              ;ADJUST FRACTION
00001238  5342           315.          SUBQ.W #1,D2             ;ADJUST EXP
0000123A  60 F4          316.          BRA.S NORMC
                         317.  *
                         318.  * BOUND CHECK ON EXP
                         319.  *
0000123C  4A42           320.  CNORMD  TST.W D2
0000123E  6F00 002E      321.          BLE CZERO
00001242  0C42 00FF      322.          CMPI.W #BGEXP,D2
00001246  6700 0042      323.          BEQ CINF
                         324.  *
                         325.  * ROUND OFF FRACTION
                         326.  *
0000124A  0806 0009      327.  ROUNDC  BTST.L #ALSB,D6
0000124E  66 0A          328.          BNE.S NOROUN
00001250  0806 0008      329.          BTST.L #ANTLSB,D6
00001254  67 04          330.          BEQ.S NOROUN
00001256  0046 0200      331.          ORI.W #ARNDDR,D6
                         332.  *
                         333.  * ALIGN RESULT
                         334.  *
0000125A  E28E           335.  NOROUN  LSR.L #1,D6              ;ROOM FOR SIGN
0000125C  4206           336.          CLR.B D6                 ;ROOM FOR EXP
                         337.  *
                         338.  * WITH BOUND CHECKS ON EXP, HAS LEADING ZERCES
                         339.  *
0000125E  8486           340.          OR.L D6,D2
00001260  8487           341.          OR.L D7,D2
00001262  2002           342.          MOVE.L D2,D0
00001264  4CDF 00FC      343.  UNSTACK MOVEM.L (SP)+,D2-D7
00001268  4E75           344.          RTS
                         345.  *
0000126A  4A86           346.  NCMPC   TST.L D6                 ;C=0?
0000126C  66 C2          347.          BNE.S NORMC              ;NO, NORMALIZE
0000126E  4280           348.  CZERO   CLR.L D0
00001270  60 F2          349.          BRA UNSTACK
00001272  DC85           350.  SAMSGN  ADD.L D5,D6              ;D6 = SUM FRACT
00001274  64 D4          351.          BCC.S ROUNDC
00001276  E296           352.          ROXR.L #1,D6             ;IF CARRY,
00001278  5242           353.          ADDQ.W #1,D2             ;ADJUST RESULT
0000127A  60 C0          354.          BRA.S CNORMD             ;GO ROUND
                         355.  *CNAN   MOVEQ.L #0,D0
                         356.  *       SUBQ.L #1,D0             ;C=NaN
                         357.  *       BRA UNSTACK
0000127C  2001           358.  CISB    MOVE.L D1,D0
0000127E  60 E4          359.          BRA UNSTACK
00001280  4443           360.  SHFTA   NEG.W D3                 ;POS SHIFT CNT
00001282  E6AD           361.          LSR.L D3,D5              ;ADJUST A FRACT
00001284  4205           362.          CLR.B D5
00001286  3401           363.          MOVE.W D1,D2             ;USE BEXP=CEXP
00001288  60 8C          364.          BRA NOADJ
0000128A  7000           365.  CINF    MOVEQ.L #0,D0
0000128C  103C 00FF      366.          MOVE.B #BGEXP,D0         ;C=inf
00001290  8087           367.          OR.L D7,D0               ;TAKE SIGN
00001292  60 D0          368.          BRA UNSTACK              ;QUIT
                         369.  *
                         370.  *
                         371.  * SUBROUTINE OUTIME
                         372.  * CALCULATES ELAPSED TIME AND OUTPUTS
                         373.  * ALL REGISTERS TRANSPARENT
                         374.  *
00001294  48E7 8002      375.  OUTIME  MOVEM.L D0/A6,-(SP)
00001298  2C7C 00012FD0  376.          MOVEA.L #TCR,A6
0000129E  203C 00FFFFFF  377.          MOVE.L #CNTSTRT,D0 ;CALC ELAPSED
000012A4  90AE 0006      378.          SUB.L 6(A6),D0
000012A8  4E4F           379.          TRAP #15
000012AA  0018           380.          DC.W DISBUF8
000012AC  4CDF 4001      381.          MOVEM.L (SP)+,D0/A6
000012B0  4E75           382.          RTS
                         383.  *
                         384.  * DONE
```

```
                        385.  *
000012B2                386.          END
    0 Errors
```

N.2 Program PI4STST — Uniprocessor Alternating Series Test Program, Optimized sub-tasks, unmeasured sub-task periods.

```
                                    1.          TTL PI4STST
                                    2.  *
                                    3.  * PROGRAM FOR SINGLE PROCESSOR CALCULATION
                                    4.  * OF PI/4 USING ALTERNATING SERIES
                                    5.  * 0+1-1/3+1/5-1/7+1/9-... FOR A NUMBER
                                    6.  * OF TERMS up to 1/65535
                                    7.  *
                                    8.  * USES GENERAL INVERT SUBROUTINE TO GIVE
                                    9.  * FLOATING POINT REPRESENTATION OF AN
                                   10.  * INTEGER IN RANGE 1<=x<=65535
                                   11.  *
                                   12.  * CALLS A FLOATING POINT ADD ROUTINE
                                   13.  *
                                   14.  * FLOATING POINT REPRESENTATION IS 32-BITS
                                   15.  *
                                   16.  * sign(1), FRACT(23), EXP (8-BITS excess 128)
                                   17.  *
                                   18.  ********************
                                   19.  * APRIL 26,1992
                                   20.  * VERSION HAS USER SPECIFIED # OF TERMS
                                   21.  * OUTPUT IS:
                                   22.  * SUM, TIME
                                   23.  ********************
                                   24.  *
                                   25.  * EQUATES
                                   26.  *
     # 00000007                     27.  SENBYC  EQU 7
     # 0000000D                     28.  CR      EQU $0D
     # 0000001A                     29.  DISBUF2 EQU 26
     # 00000018                     30.  DISBUF8 EQU 24
     # 00000011                     31.  OUTMESC EQU 17
     # 0000000A                     32.  SENCLFC EQU 10
     # 00000004                     33.  EOT     EQU $04
     # 0000001C                     34.  BLDNUM  EQU 28
     # 00FFFFFF                     35.  CNTSTRT EQU $00FFFFFF
     # 00012FD0                     36.  TCR     EQU $12FD0
     # 00000001                     37.  ENABLTM EQU 1
     # 0000000A                     38.  LF      EQU $0A
     # 000000FF                     39.  BGEXP   EQU $FF
     # 0000001F                     40.  SGNMSK  EQU 31
     # 00000080                     41.  BIAS    EQU 128
     # 00000009                     42.  ALSB    EQU 9
     # 00000008                     43.  ANTLSB  EQU 8
     # 00000200                     44.  ARNDDR  EQU $200
     # 00000017                     45.  SIGBITS EQU 23
                                   46.  *
                                   47.  *
00001000                           48.          ORG.L $1000
00001000  2C7C 00012FD0            49.          MOVEA.L #TCR,A6
00001006  4216                     50.          CLR.B (A6)
00001008  2D7C 00FFFFFF            51.          MOVE.L #CNTSTRT,2(A6)
          0002
00001010  4E4F                     52.  PI4     TRAP #15
00001012  000A                     53.          DC.W SENCLFC
00001014  41FA 006B                54.          LEA TERMS(PC),A0
00001018  4E4F                     55.          TRAP #15
0000101A  0011                     56.          DC.W OUTMESC
0000101C  4E4F                     57.          TRAP #15
0000101E  001C                     58.          DC.W BLDNUM
00001020  4E4F                     59.          TRAP #15
00001022  000A                     60.          DC.W SENCLFC
00001024  2F00                     61.          MOVE.L D0,-(SP)
00001026  41FA 0079                62.          LEA HEAD(PC),A0
0000102A  4E4F                     63.          TRAP #15
0000102C  0011                     64.          DC.W OUTMESC
0000102E  4E4F                     65.          TRAP #15
00001030  000A                     66.          DC.W SENCLFC
00001032  201F                     67.          MOVE.L (SP)+,D0
00001034  E280                     68.          ASR.L #1,D0       ;DIVIDE BY 2
00001036  5380                     69.          SUBQ.L #1,D0      ;ADJUST FOR DBRA
00001038  2E00                     70.          MOVE.L D0,D7      ;LOOP COUNTER
0000103A  4280                     71.          CLR.L D0          ;D0 <- PARTIAL SUMS
0000103C  4201                     72.          CLR.B D1          ;
0000103E  7C01                     73.          MOVEQ.L #1,D6     ;D6 <- TERM DENOM
                                   74.  *
                                   75.  * START TIME FOR DIVIDE
```

```
                                    76.  *
00001040  1CBC 0001                 77.  MNLOOP  MOVE.B #ENABLTM,(A6)
00001044  6100 006C                 78.          BSR INVERT       ;RESULT IN D4.L
00001048  2204                      79.          MOVE.L D4,D1
0000104A  6100 00B4                 80.          BSR ADDFP        ;CALC ODD PARTIAL SUM
0000104E  5446                      81.          ADDQ.W #2,D6     ;NEW TERM DENOM
00001050  6100 0060                 82.          BSR INVERT       ;RESULT IN D4.L
00001054  08C4 001F                 83.          BSET.L #SGNMSK,D4       ;NEGATE
00001058  2204                      84.          MOVE.L D4,D1
0000105A  6100 00A4                 85.          BSR ADDFP        ;CALC EVEN PARTIAL SUM
0000105E  5446                      86.          ADDQ.W #2,D6     ;NEW DENOMINATOR
00001060  51CF FFDE                 87.          DBRA D7,MNLOOP
00001064  4216                      88.          CLR.B (A6)
00001066  4E4F                      89.          TRAP #15
00001068  0018                      90.          DC.W DISBUF8     ;OUTPUT SUM
0000106A  41FA 0010                 91.          LEA SPACES(PC),A0
0000106E  4E4F                      92.          TRAP #15
00001070  0011                      93.          DC.W OUTMESC
00001072  6100 015C                 94.          BSR OUTIME
00001076  4E4F                      95.          TRAP #15
00001078  000A                      96.          DC.W SENCLFC
0000107A  4E75                      97.          RTS
                                    98.  *
0000107C  20 20 20 20 04            99.  SPACES  DC.B '    ',EOT
00001081  53 70 65 63 69 100.       TERMS   DC.B 'Specify even number of terms > '
          66 79 20 65 76
          65 6E 20 6E 75
          6D 62 65 72 20
          6F 66 20 74 65
          72 6D 73 20 3E
          20
000010A0  04                        101.         DC.B EOT
000010A1  53 75 6D 20 20 102.  HEAD     DC.B 'Sum          Time',EOT
          20 20 20 20 20
          20 20 54 69 6D
          65 04
                                    103. *
                                    104. *
                                    105. * SUBROUTINE INVERT
                                    106. * INTEGER TO INVERT TO FP IS IN D6.W
                                    107. * RETURNS FP NUMBER IN D4.L
                                    108. * D6.W UNAFFECTED
                                    109. *
                                    110. * VOLATILE REGISTERS
                                    111. * D2,D3,D4,D5
                                    112. *
000010B2  7410                      113. INVERT  MOVEQ.L #16,D2  ;D2 <- EXPONENT
000010B4  3A06                      114.         MOVE.W D6,D5
000010B6  E35D                      115. SHFTLP  ROL.W #1,D5     ;GET MSBIT=1 IN D5
000010B8  55CA FFFC                 116.         DBCS D2,SHFTLP
000010BC  E25D                      117.         ROR.W #1,D5     ;ADJUST ONE BACK
000010BE  4402                      118.         NEG.B D2        ;NEGATE EXPONENT
000010C0  D43C 0080                 119.         ADD.B #BIAS,D2  ;BIAS
000010C4  7601                      120.         MOVEQ.L #1 ^3   ;FAST WAY FOR
000010C6  E29B                      121.         ROR.L #1,I      ;MOVE.L #$80000000,D3
000010C8  86C5                      122.         DIVU.W D5,.     ;DO MSW DIVIDE
000010CA  69 26                     123.         BVS.S POWR2     ;IF OVRFLW, DIVISOR=2**k
000010CC  3803                      124.         MOVE.W D3,D
000010CE  4844                      125.         SWAP.W D4       ;MSW OF QUO IN D4
000010D0  4243                      126.         CLR.W D3        ;CLEAR LOW WORD D3
                                    127. *                       ;GIVES EXTENDED REMAINDER
000010D2  2603                      128.         MOVE.L D3,D3    ;DUMMY MOVE FOR CCR Z
000010D4  67 14                     129.         BE_.S LASTEP    ;REM=0? BRANCH, NO ROUND
000010D6  86C5                      130.         DIVU.W D5,D3    ;GET LSW OF QUOTIENT
                                    131. *                       ;NEVER OVRFLW, IGNORE REM
000010D8  3803                      132.         MOVE.W D3,D4    ;ALL BITS QUOTIENT -> D4
                                    133. *
                                    134. * round the result
                                    135  *
000010DA  0804 0009                 136. ROUNDQ  BTST.L #ALSB,D4
000010DE  66 0A                     137.         BNE.S LASTEP
000010E0  0804 0008                 138.         BTST.L #ANTLSB,D4
000010E4  67 04                     139.         BEQ.S LASTEP
000010E6  0044 0200                 140.         ORI.W #ARNDDR,D4
000010EA  E28C                      141. LASTEP  LSR.L #1,D4     ;SIGN=0
000010EC  5202                      142.         ADDQ.B #1,D2
000010EE  1802                      143.         MOVE.B D2,D4
000010F0  4E75                      144.         RTS
                                    145. *
                                    146. * OVERFL_ED, POWER OF 2 DIVISOR, QUO=DIVISOR*2
                                    147. * NO REMAINDER, NO ROUNDING REQUIRED
```

```
                                148.  *
000010F2  4845                  149.  POWR2   SWAP D5           ;MSW OF QUO
000010F4  4245                  150.          CLR.W D5          ,NO MORE SIG DIGITS
000010F6  E28D                  151.          LSR.L #1,D5       ;SIGN=0
000010F8  5402                  152.          ADDQ.B #2,D2      ;ADJUST EXPONENT
000010FA  2805                  153.          MOVE.L D5,D4
000010FC  1802                  154.          MOVE.B D2,D4
000010FE  4E75                  155.          RTS
                                156.  *
                                157.  * SUBROUTINE ADDFP
                                158.  *
                                159.  * FLOATING POINT ADDITION FOR 32-BIT
                                160.  * FLOATING POINT FORMAT:
                                161.  * SIGN (1)| FRACTION (23) | EXPONENT (8)
                                162.  * WHERE EXPONENT IS BIASED EXCESS 128;
                                163.  * FRACTION IS NORMALIZED
                                164.  * 0 = $00000000
                                165.  * (SIGN) INFINTY = $(8.OR.0)00000FF
                                166.  * NaN = $XXXXXXFF
                                167.  *       WHERE XXXXXX IS NOT AS INFINITY
                                168.  *
                                169.  * DOES A*B=C WHERE
                                170.  * A IS IN D0
                                171.  * B IS IN D1
                                172.  * C RETURNS IN D0
                                173.  * IF A OR B = NaN, SETS C=NaN = $FFFFFFFF
                                174.  * IF A OR B = +-INFINITY, SET C=NaN
                                175.  *
00001100  48E7 3F00              176.  ADDFP    MOVEM.L D2-D7,-(SP)      ;SAVE OFF
                                177.  *        CMPI.B #BGEXP,D0         ;A=NaN OR inf?
                                178.  *        BEQ CNAN                 ;C=NaN IF SO
                                179.  *        CMPI.B #BGEXP,D1         ;B =aN OR inf?
                                180.  *        BEQ CNAN                 ;C=NaN IF SO
00001104  4A81                  181.          TST.L D1                 ;B=0?
00001106  6700 0098             182.          BEQ UNSTACK              ;C=A IF SO
0000110A  4A80                  183.          TST.L D0                 ;A=0?
0000110C  6700 00AA             184.          BEQ CISB                 ;C=B IF SO
00001110  4243                  185.          CLR.W D3
00001112  4244                  186.          CLR.W D4
00001114  1600                  187.          MOVE.B D0,D3             ;(D3.W)=EXPA
00001116  1801                  188.          MOVE.B D1,D4             ;(D4.W)=EXPB
00001118  9644                  189.          SUB.W D4,D3              ;(D3.W)=SHFTCNT
0000111A  0C43 0017             190.          CMPI.W #SIGBITS,D3       ;TOO BIG? C=A
0000111E  6C00 0080             191.          BGE UNSTACK
00001122  0C43 FFE9             192.          CMPI.W #-SIGBITS,D3      ;TOO SMALL? C=B
00001126  6F00 0090             193.          BLE CISB
                                194.  *
                                195.  * HERE, MUST DO ADD
                                196.  * D5, D6 GET FRACTIONS; D7 CARRIES SIGN
                                197.  *
0000112A  7E00                  198.          MOVEQ.L #0,D7
0000112C  0800 001F             199.          BTST.L #SGNMSK,D0        ;SIGN OF A
00001130  67 04                 200.          BEQ.S POSTVE
00001132  08C7 001F             201.          BSET.L #SGNMSK,D7        ;C IS NEG
00001136  2A00                  202.  POSTVE  MOVE.L D0,D5             ;(D5)=AF
00001138  2C01                  203.          MOVE.L D1,D6             ;(D6)=BF
0000113A  4205                  204.          CLR.B D5
0000113C  4206                  205.          CLR.B D6                 ;KILL EXPS
0000113E  E38D                  206.          LSL.L #1,D5
00001140  E38E                  207.          LSL.L #1,D6              ;ALIGN
00001142  7400                  208.          MOVEQ.L #0,D2            ;
00001144  1400                  209.          MOVE.B D0,D2             ;CEXP=AEXP
00001146  4A43                  210.          TST.W D3                 ;TEST SC
00001148  67 08                 211.          BEQ.S NOADJ              ;IF=0 NO SHFTNG
0000114A  6B00 0070             212.          BMI SHFTA                ;NEG, SHFT A
0000114E  E6AE                  213.  SHFTB   LSR.L D3,D6              ;SHIFTB
00001150  4206                  214.          CLR.B D6
00001152  2600                  215.  NOADJ   MOVE.L D0,D3             ;COPY OF A
00001154  B383                  216.          EOR.L D1,D3              ;SIGNS
00001156  0803 001F             217.          BTST.L #SGNMSK,D3        ;
0000115A  6700 0052             218.          BEQ SAMSGN               ;EQUAL SIGNS?
0000115E  4486                  219.  DIFFSGN NEG.L D6                ;NEG B
00001160  DC85                  220.          ADD.L D5,D6              ;D6 HAS THE SUM
00001162  6500 0042             221.          BCS NCMPC                ;IF CAR=0, NOT=0
00001166  0847 001F             222.  COMPC   BCHG.L #SGNMSK,D7        ;COMPLEMENT SUM
0000116A  4486                  223.          NEG.L D6
0000116C  0806 001F             224.  NORMC   BTST.L #SGNMSK,D6        ;NORMALIZED?
00001170  66 06                 225.          BNE.S CNORMD
00001172  E38E                  226.          LSL.L #1,D6              ;ADJUST FRACTION
00001174  5342                  227.          SUBQ.W #1,D2             ;ADJUST EXP
00001176  60 F4                 228.          BRA.S NORMC
```

```
                          229.  *
                          230.  * BOUND CHECK ON EXP
                          231.  *
00001178  4A42            232.  CNORMD  TST.W D2
0000117A  6F00 002E       233.         BLE CZERO
0000117E  0C42 00FF       234.         CMPI.W #BGEXP,D2
00001182  6700 0042       235.         BEQ CINF
                          236.  *
                          237.  * ROUND OFF FRACTION
                          238.  *
00001186  0806 0009       239.  ROUNDC  BTST.L #ALSB,D6
0000118A  66 0A           240.         BNE.S NOROUN
0000118C  0806 0008       241.         BTST.L #ANTLSB,D5
00001190  67 04           242.         BEQ.S NOROUN
00001192  0046 0200       243.         ORI.W #ARNDDR,D6
                          244.  *
                          245.  * ALIGN RESULT
                          246.  *
00001196  E28E            247.  NOROUN  LSR.L #1,D6              ;ROOM FOR SIGN
00001198  4206            248.         CLR.B D6                 ;ROOM FOR EXP
                          249.  *
                          250.  * WITH BOUND CHECKS ON EXP, HAS LEADING ZEROES
                          251.  *
0000119A  8486            252.         OR.L D6,D2
0000119C  8487            253.         OR.L D7,D2
0000119E  2002            254.         MOVE.L D2,D0
000011A0  4CDF 00FC       255.  UNSTACK MOVEM.L (SP)+,D2-D7
000011A4  4E75            256.         RTS
                          257.  *
000011A6  4A86            258.  NCMPC   TST.L D6                 ;C=0?
000011A8  66 C2           259.         BNE.S NORMC              ;NO, NORMALIZE
000011AA  4280            260.  CZERO   CLR.L D0
000011AC  60 F2           261.         BRA UNSTACK
000011AE  DC85            262.  SAMSGN  ADD.L D5,D6             ;D6 = SUM FRACT
000011B0  64 D4           263.         BCC.S ROUNDC
000011B2  E296            264.         ROXR.L #1,D6             ;IF CARRY,
000011B4  5242            265.         ADDQ.W #1,D2             ;ADJUST RESULT
000011B6  60 C0           266.         BRA.S CNORMD             ;GO ROUND
                          267.  *CNAN   MOVEQ.L #0,D0
                          268.  *        SUBQ.L #1,D0             ;C=NaN
                          269.  *        BRA UNSTACK
000011B8  2001            270.  CISB    MOVE.L D1,D0
000011BA  60 E4           271.         BRA UNSTACK
000011BC  4443            272.  SHFTA   NEG.W D3                 ;POS SHIFT CNT
000011BE  E6AD            273.         LSR.L D3,D5              ;ADJUST A FRACT
000011C0  4205            274.         CLR.B D5
000011C2  3401            275.         MOVE.W D1,D2             ;USE BEXP=CEXP
000011C4  60 8C           276.         BRA NOADJ
000011C6  7000            277.  CINF    MOVEQ.L #0,D0
000011C8  103C 00FF       278.         MOVE.B #BGEXP,D0         ;C=inf
000011CC  8087            279.         OR.L D7,D0               ;TAKE SIGN
000011CE  60 D0           280.         BRA UNSTACK              ;QUIT
                          281.  *
                          282.  *
                          283.  * SUBROUTINE OUTIME
                          284.  * CALCULATES ELAPSED TIME AND OUTPUTS
                          285.  * ALL REGISTERS TRANSPARENT
                          286.  *
000011D0  48E7 8002       287.  OUTIME  MOVEM.L D0/A6,-(SP)
000011D4  2C7C 00012FD0   288.         MOVEA.L #TCR,A6
000011DA  203C 00FFFFFF   289.         MOVE.L #CNTSTRT,D0 ;CALC ELAPSED
000011E0  90AE 0006       290.         SUB.L 6(A6),D0
000011E4  4E4F            291.         TRAP #15
000011E6  0018            292.         DC.W DISBUF8
000011E8  4CDF 4001       293.         MOVEM.L (SP)+,D0/A6
000011EC  4E75            294.         RTS
                          295.  *
                          296.  * DONE
                          297.  *
000011EE                  298.         END
    0 Errors
```

N.3 Program PI4MTST — Multicomputer Alternating Series Test Program. Optimized sub-tasks.

```
                                1.          TTL PI4MTST
                                2.  *
                                3.  * PROGRAM FOR 4 PROCESSOR CALCULATION
                                4.  * OF PI/4 USING ALTERNATING SERIES
                                5.  * 0+1-1/3+1/5-1/7+1/9-... FOR FIXED NUMBER
                                6.  * OF TERMS <=1/65535
                                7.  *
                                8.  * SLAVE 0 AND SLAVE 2 DO THE INVERT STEP
                                9.  * TO GIVE FLOATING POINT REPRESENTATION OF AN
                               10.  * INTEGER IN RANGE 1<=x<=65535
                               11.  * SLAVE 0 - 1,5,9,... (TERMS WITH POSITIVE)
                               12.  * SLAVE 2 - 3,7,11,...(TERMS WITH NEGATIVE)
                               13.  * SLAVES 1 AND 3 DO THE ADD
                               14.  * STEP.
                               15.  * SLAVE 1 -RUNNING SUM OF TERMS FROM SLAVE 0
                               16.  * SLAVE 3 -RUNNING SUM OF TERMS FROM SLAVE 2
                               17.  * MASTER COLLECTS AT END AND DOES LAST
                               18.  * SUBTRACT SLAVE1SUM-SLAVE3SUM
                               19.  *
                               20.  * SIGN (1-BIT)
                               21.  * FRACTION (23-BITS),
                               22.  * EXPONENT (8-BITS SIGNED, EXCESS 128)
                               23.  *
                               24.  **********************
                               25.  * APRIL 27,1992      *
                               26.  * VERSION TIMED,     *
                               27.  * MASTER TIMER ONLY  *
                               28.  **********************
                               29.  *
                               30.  * EQUATES
                               31.  *
# 00000003                     32.  NTFR     EQU 3    ;BYTES TFRED-1
# 00000018                     33.  DISBUF8 EQU 24
# 00000019                     34.  DISBUF4 EQU 25
# 0000001A                     35.  DISBUF2 EQU 26
# 00000011                     36.  OUTMESC EQU 17
# 0000000A                     37.  SENCLFC EQU 10
# 00000004                     38.  EOT      EQU $04
# 00012FC0                     39.  PGCR     EQU $12FC0
# 00012FC6                     40.  PACR     EQU $12FC6
# 00012FC7                     41.  PBCR     EQU $12FC7
# 00012FC1                     42.  PSRR     EQU $12FC1
# 00012FC8                     43.  PADR     EQU $12FC8
# 00012FCD                     44.  PPSR     EQU $12FCD
# 00012FC9                     45.  PBDR     EQU $12FC9
# 00000006                     46.  H3LEV    EQU 6
# 00000002                     47.  H3S      EQU 2
# 00000000                     48.  H1S      EQU 0
# 00000010                     49.  ENABLEA EQU $10
# 00000020                     50.  ENABLEB EQU $20
# 00000030                     51.  ENABLAB EQU $30
# 0000000E                     52.  ABORT    EQU 14
# 00000003                     53.  SRQASRT EQU 3
# 00000004                     54.  ADBYTIN EQU 4
# 00000007                     55.  ADBYTOT EQU 7
# 00000005                     56.  INDATA   EQU 5
# 00000006                     57.  OUTDATA EQU 6
# 00000008                     58.  NETCNF   EQU 8
# 00000009                     59.  SRQSRVC EQU 9
# 0000001C                     60.  BLDNUM   EQU 28
# 00000001                     61.  SLV0     EQU $01
# 00000002                     62.  SLV1     EQU $02
# 00000004                     63.  SLV2     EQU $04
# 00000008                     64.  SLV3     EQU $08
# 0000000F                     65.  ALLSLV   EQU $0F
# 00012FB0                     66.  PSR01    EQU $12FB0
# 00012FB1                     67.  PSR23    EQU $12FB1
# 00012FB2                     68.  PSR45    EQU $12FB2
# 00012FB3                     69.  PSR67    EQU $12FB3
# 00012FB4                     70.  PSR89    EQU $12FB4
# 00012FB5                     71.  PSR1011 EQU $12FB5
# 00012FB6                     72.  PSR1213 EQU $12FB6
# 00012FB7                     73.  PSR1415 EQU $12FB7
# 00012FB8                     74.  PSR1617 EQU $12FB8
# 00012FB9                     75.  PSR1819 EQU $12FB9
# 00012FD0                     76.  TCR      EQU $12FD0
```

```
                                                 77.  TPLR     EQU TCR+2
                                                 78.  TCNTR    EQU TCR+6
                                                 79.  CNTSTRT EQU $00FFFFFF
                                                 80.  ENABLTM EQU 1
                                                 81.  PASSCNT EQU 256
                                                 82.  DIV256  EQU 8
                                                 83.  BGEXP   EQU $FF
                                                 84.  SGNMSK  EQU 31
                                                 85.  BIAS    EQU 128
                                                 86.  ALSB    EQU 9
                                                 87.  ANTLSB  EQU 8
                                                 88.  ARNDDR  EQU $200
                                                 89.  SIGBITS EQU 23
                                                 90.  *
00001000                                         91.          ORG.L $1000
00001000  <2>                                   92.  TCOUNT   DS.W 1   ;INPUT DATA FOR SLAVES
                                                 93.  *                ;LOOP COUNTER FOR TERMS
                                                 94.  *                ;ADJUST FOR DBRA
                                                 95.  *                ;#TERMS/2-1
00001002  <4>                                    96.  TEMPDAT DS.W 2
00001006  <4>                                    97.  POSDAT  DS.W 2
0000100A  <4>                                    98.  NEGDAT  DS.W 2
0000100E  BB BB BB CB BB                         99.  INITCNF DC.B $BB,$BB,$BB,$CB,$BB,$8C,$CB,$8B
          8C CB 8B
00001016  B8 8B                                 100.          DC.B $B8,$8B
00001018  BD BB BD CB BD                        101.  BRDCAST DC.B $BD,$BB,$BD,$CB,$BD,$8C,$CB,$8B
          8C CB 8B
00001020  B8 8B                                 102.          DC.B $B8,$8B
00001022  00 00 00 70 00                        103.  PARPIPS DC.B $00,$00,$00,$70,$00,$B8,$70,$80
          B8 70 80
0000102A  B8 8B                                 104.          DC.B $B8,$8B
                                                105.  *
                                                106.  * CONFIGURE NETWORK TO INITIAL STATE, MASTER
                                                107.  * TO SLAVE 0
                                                108.  * RESPOND TO SLAVE 0'S RFP SRQ, AND LOAD
                                                109.  * ITS PROGRAM
                                                110.  *
0000102C  287C 00012FD0                         111.  STRTSER MOVEA.L #TCR,A4          ;POINTER TIMER
00001032  4214                                  112.          CLR.B (A4)              ;DISABLE F'SURE
00001034  2F3C 00FFFFFF                         113.          MOVE.L #CNTSTRT,-(SP)   ;MIN TIME
0000103A  2F3C 00000000                         114.          MOVE.L #$00,-(SP)       ;MAX TIME
00001040  2F3C 00000000                         115.          MOVE.L #$00,-(SP)       ;RUN'G SUM TIME
00001046  2E3C 000000FF                         116.          MOVE.L #PASSCNT-1,D7    ;PASS COUNTER
0000104C  2A7C 00012FCD                         117.          MOVEA.L #PPSR,A5        ;A5 POINTR PPSR
00001052  2C7C 00012FC0                         118.          MOVEA.L #PGCR,A6        ;A6 POINTR PADR
00001058  247C 00012FB9                         119.          MOVEA.L #PSR1819,A2
0000105E  41FA FFAE                             120.  MASTER  LEA INITCNF(PC),A0
00001062  4E48                                  121.          TRAP #NETCNF
00001064  103C 0001                             122.          MOVE.B #SLV0,D0
00001068  4E49                                  123.          TRAP #SRQSRVC
0000106A  41FA 0312                             124.          LEA SLVPR0(PC),A0       ;STRT ADDR
0000106E  43FA 039E                             125.          LEA SLVPR13(PC),A1      ;END ADDR+1
00001072  6100 01B2                             126.          BSR SENDOUT             ;ADDR,CNT,CODE
                                                127.  *
                                                128.  * CONFIGURE NETWORK TO MASTER TO SLAVE 1 AND 3
                                                129.  * RESPOND TO SLAVE 1 AND 3'S RFP SRQ, AND LOAD
                                                130.  * THEIR COMMON PROGRAM
                                                131.  *
00001076  13FC 00B4                             132.          MOVE.B #$B4,PSR01
          00012FB0
0000107E  13FC 00BD                             133.          MOVE.B #$BD,PSR45
          00012FB2
00001086  13FC 00B4                             134.          MOVE.B #$B4,PSR89
          00012FB4
0000108E  103C 000A                             135.          MOVE.B #$A,D0    ;SERVICE 1 AND 3
00001092  4E49                                  136.          TRAP #SRQSRVC
00001094  41FA 0378                             137.          LEA SLVPR13(PC),A0
00001098  43FA 0492                             138.          LEA SLVPR2(PC),A1
0000109C  6100 0188                             139.          BSR SENDOUT
                                                140.  *
                                                141.  * CONFIGURE NETWORK TO MASTER TO SLAVE 2
                                                142.  * RESPOND TO SLAVE 2'S RFP SRQ, AND LOAD ITS
                                                143.  * PROGRAM
                                                144.  *
000010A0  13FC 00B4                             145.          MOVE.B #$B4,PSR45
          00012FB2
000010A8  13FC 00BB                             146.          MOVE.B #$BB,PSR89
          00012FB4
000010B0  103C 0004                             147.          MOVE.B #SLV2,D0
000010B4  4E49                                  148.          TRAP #SRQSRVC
000010B6  41FA 0474                             149.          LEA SLVPR2(PC),A0
```

```
000010BA  43FA 0500       150.          LEA DONE2(PC),A1
000010BE  6100 0166       151.          BSR SENDOUT
                          152.  *
                          153.  * ALL PROGRAMS LOADED, EACH SLAVE ASSERTING
                          154.  * READY FOR DATA SRQ. ACQUIRE USER INPUT OF
                          155.  * NUMBER OF TERMS, ADJUST FOR THE ARCHITECTURE
                          156.  * AND DBRA, AND STORE AT TCOUNT. THEN RESPOND
                          157.  * TO ALL SLAVES SRQ'S, THEN LOAD THE DATA
                          158.  * IN BROADCAST MODE.
                          159.  *
000010C2  4E4F            160.  ACQCNT  TRAP #15
000010C4  000A            161.          DC.W SENCLFC
000010C6  41FA 0173       162.          LEA TERMS(PC),A0
000010CA  4E4F            163.          TRAP #15
000010CC  0011            164.          DC.W OUTMESC
000010CE  4E4F            165.          TRAP #15
000010D0  001C            166.          DC.W BLDNUM
000010D2  4E4F            167.          TRAP #15
000010D4  000A            168.          DC.W SENCLFC
000010D6  E280            169.          ASR.L #1,D0      ;DIVIDE BY 2
000010D8  5340            170.          SUBQ.W #1,D0     ;ADJUST FOR DBRA
000010DA  41FA FF24       171.          LEA TCOUNT(PC),A0
000010DE  3080            172.          MOVE.W D0,(A0)
000010E0  41FA 0179       173.          LEA HEAD(PC),A0
000010E4  4E4F            174.          TRAP #15
000010E6  0011            175.          DC.W OUTMESC
000010E8  4E4F            176.          TRAP #15
000010EA  000A            177.          DC.W SENCLFC
                          178.  *
                          179.  * THIS IS REPEATED 256 TIMES AND MEASURED
                          180.  * AND STATS KEPT
                          181.  *
000010EC  297C 00FFFFFF   182.          MOVE.L #CNTSTRT,2(A4)
          0002
000010F4  41FA FF22       183.  REXE     LEA BRDCAST(PC),A0
000010F8  4E48            184.          TRAP #NETCNF
000010FA  103C 000F       185.          MOVE.B #ALLSLV,D0
000010FE  4E49            186.          TRAP #SRQSRVC
00001100  41FA FEFE       187.          LEA TCOUNT(PC),A0        ;START OF DATA
00001104  43FA FEFC       188.          LEA TEMPDAT(PC),A1       ;END OF DAT. +1
00001108  18BC 0001       189.  TIMSTRT MOVE.B #ENABLTM,(A4)     ;TIMER STARTS
0000110C  6100 0118       190.          BSR SZNDOUT
                          191.  *
                          192.  * WHEN HERE, ALL SLAVES LOADED, WILL ASSERT
                          193.  * READY TO EXECUTE SRQ.  RESPOND TO LET THEM
                          194.  * CONTINUE.  BUT FIRST, CONFIGURE NETWORK
                          195.  * INTO 2 PARALLEL PATHS WITH SLV0 TO SLV1 TO
                          196.  * MASTER AND SLV2 TO SLV3 NEARLY TO MASTER,
                          197.  * ST SWITCH FROM SLV1 TO MASTER TO SLV3 TO
                          198.  * MASTER IS DONE IN ONE PSR REGISTER WRITE.
                          199.  *
00001110  41FA FF10       200.          LEA PARPIPS(PC),A0
00001114  4E48            201.          TRAP #NETCNF
00001116  103C 000F       202.          MOVE.B #ALLSLV,D0
0000111A  4E49            203.          TRAP #SRQSRVC            ;SLAVES EXCUTE
                          204.  *
                          205.  * WAIT FOR RESULTS FROM SLAVE 1 AND UPLOAD
                          206.  * THE FP DATA
                          207.  *
0000111C  103C 0002       208.          MOVE.B #SLV1,D0
00001120  4E49            209.          TRAP #SRQSRVC
00001122  7004            210.          MOVEQ.L #4,D0
00001124  41FA FEE0       211.          LEA POSDAT(PC),A0
00001128  1CBC 0010       212.          MOVE.B #ENABLEA,(A6)     ;EN INPUT PORT
0000112C  4E45            213.          TRAP #INDATA
0000112E  4216            214.          CLR.B (A6)
                          215.  *
                          216.  * CONFIGURE TO SLAVE3 TO MASTER
                          217.  * AND GET NEGATIVE TERM SUM
                          218.  *
00001130  14BC 00BB       219.          MOVE.B #$BB,(A2)
00001134  103C 0008       220.          MOVE.B #SLV3,D0
00001138  4E49            221.          TRAP #SRQSRVC
0000113A  7004            222.          MOVEQ.L #4,D0
0000113C  41FA FECC       223.          LEA NEGDAT(PC),A0
00001140  1CBC 0010       224.          MOVE.B #ENABLEA,(A6)
00001144  4E45            225.          TRAP #INDATA
00001146  4216            226.          CLR.B (A6)
                          227.  *
                          228.  * NOW DO POSDAT+NEGDAT
                          229.  *
```

```
00001148  203A FEBC        230.          MOVE.L POSDAT(PC),D0
0000114C  223A FEBC        231.          MOVE.L NEGDAT(PC),D1
00001150  08C1 001F        232.          BSET.L #SGNMSK,D1        ;NEGATE D1
00001154  6100 0158        233.          BSR ADDFPM
                           234.  *
                           235.  ******************************************
                           236.  * STOP TIMER AND DO STATS ACCUMULATION
                           237.  ******************************************
00001158  4214             238.  OUTRES  CLR.B (A4)        ;TIME STOPPED
0000115A  2F00             239.          MOVE.L D0,-(SP) ;SAVE RESULT
                           240.  *
                           241.  * CALCULATE LAST TERM DENOMINATOR
                           242.  *
0000115C  4280             243.          CLR.L D0
0000115E  303A FEA0        244.          MOVE.W TCOUNT(PC),D0
00001162  5240             245.          ADDQ.W #1,D0
00001164  E540             246.          ASL.W #2,D0      ;MULTIPLY BY 4
00001166  5340             247.          SUBQ.W #1,D0     ;LAST DENOM
00001168  4E4F             248.          TRAP #15
0000116A  0018             249.          DC.W DISBUF8     ;OUTPUT
0000116C  41FA 00C8        250.          LEA SPACES(PC),A0
00001170  4E4F             251.          TRAP #15
00001172  0011             252.          DC.W OUTMESC
00001174  201F             253.          MOVE.L (SP)+,D0 ;GET RESULTS
00001176  4E4F             254.          TRAP #15
00001178  0018             255.          DC.W DISBUF8     ;OUTPUT RESULT
0000117A  41FA 00BA        256.          LEA SPACES(PC),A0
0000117E  4E4F             257.          TRAP #15
00001180  0011             258.          DC.W OUTMESC
                           259.  *
                           260.  * NOW ELAPSED TIME
                           261.  *
00001182  222C 0006        262.          MOVE.L 6(A4),D1 ;GET NEW COUNT
00001186  203C 00FFFFFF    263.          MOVE.L #CNTSTRT,D0
0000118C  9081             264.          SUB.L D1,D0      ;GET ELAPSED COUNT
                           265.  *
                           266.  * OUTPUT THE COUNT IN HEX
                           267.  *
0000118E  4E4F             268.          TRAP #15
00001190  0018             269.          DC.W DISBUF8
00001192  2F00             270.          MOVE.L D0,-(SP)
00001194  41FA 00A0        271.          LEA SPACES(PC),A0
00001198  4E4F             272.          TRAP #15
0000119A  0011             273.          DC.W OUTMESC
0000119C  3007             274.          MOVE.W D7,D0
0000119E  4E4F             275.          TRAP #15
000011A0  C019             276.          DC.W DISBUF4
000011A2  4E4F             277.          TRAP #15
000011A4  000A             278.          DC.W SENCLFC
000011A6  201F             279.          MOVE.L (SP)+,D0
                           280.  *
                           281.  * NOW KEEP TRACK OF STATISTICS
                           282.  *
000011A8  D197             283.          ADD.L D0,(SP)    ;ADD TO SUM
000011AA  B0AF 0004        284.          CMP.L 4(SP),D0   ;CHECK MAX
000011AE  6F 04            285.          BLE.S NOTMAX     ;D0 NOT MAX
000011B0  2F40 0004        286.          MOVE.L D0,4(SP) ;NEW MAX
000011B4  B0AF 0008        287.  NOTMAX  CMP.L 8(SP),D0   ;CHECK MIN
000011B8  6C 0C            288.          BGE.S NOTMIN     ;D0 NOT MIN
000011BA  2F40 0008        289.          MOVE.L D0,8(SP) ;NEW MIN
                           290.  *
                           291.  * DO WHOLE PROCESS AGAIN, 256 TIMES
                           292.  *
000011BE  42B8 1006        293.          CLR.L POSDAT
000011C2  42B8 100A        294.          CLR.L NEGDAT
000011C6  51CF FF2C        295.  NOTMIN  DBRA D7,REXE
                           296.  *
                           297.  * WHEN HERE, PROG HAS BEEN EXECUTED 256 TIMES
                           298.  * SEND AN ILLEGAL ADDRESS FOR DATA, FORCING
                           299.  * SLAVES TO PROGRAM ACCEPT MODE
                           300.  *
000011CA  41FA FE4C        301.          LEA BRDCAST(PC),A0
000011CE  4E48             302.          TRAP #NETCNF
000011D0  103C 000F        303.          MOVE.B #ALLSLV,D0
000011D4  4E49             304.          TRAP #SRQSRVC
000011D6  207C FFFF0000    305.          MOVEA.L #$FFFF0000,A0   ;ILLEGAL ADDR
000011DC  1CBC 0020        306.          MOVE.B #ENABLEB,(A6)
000011E0  4E47             307.          TRAP #ADBYTOT    ;THAT'LL FIX 'EM
000011E2  4216             308.          CLR.B (A6)
                           309.  *
                           310.  * OUTPUT STATISTICS
```

```
                                311.  *
000011E4  4E4F                  312.          TRAP #15
000011E6  000A                  313.          DC.W SENCLFC
000011E8  4E4F                  314.          TRAP #15
000011EA  000A                  315.          DC.W SENCLFC
000011EC  41FA 0092             316.          LEA AVEMES(PC),A0
000011F0  4E4F                  317.          TRAP #15
000011F2  0011                  318.          DC.W OUTMESC
000011F4  201F                  319.          MOVE.L (SP)+,D0 ;GET SUM
000011F6  E080                  320.          ASR.L #DIV256,D0        ;DO AVERAGE
000011F8  4E4F                  321.          TRAP #15
000011FA  0018                  322.          DC.W DISBUF8    ;OUTPUT AVERAGE
000011FC  4E4F                  323.          TRAP #15
000011FE  000A                  324.          DC.W SENCLFC
00001200  41FA 008D             325.          LEA MAXMES(PC),A0
00001204  4E4F                  326.          TRAP #15
00001206  0011                  327.          DC.W OUTMESC
00001208  201F                  328.          MOVE.L (SP)+,D0 ;GET MAX
0000120A  4E4F                  329.          TRAP #15
0000120C  0018                  330.          DC.W DISBUF8
0000120E  4E4F                  331.          TRAP #15
00001210  000A                  332.          DC.W SENCLFC
00001212  41FA 008A             333.          LEA MINMES(PC),A0
00001216  4E4F                  334.          TRAP #15
00001218  0011                  335.          DC.W OUTMESC
0000121A  201F                  336.          MOVE.L (SP)+,D0 ;GET MIN
0000121C  4E4F                  337.          TRAP #15
0000121E  0018                  338.          DC.W DISBUF8
00001220  4E4F                  339.          TRAP #15
00001222  000A                  340.          DC.W SENCLFC
                                341.  *
                                342.  * DONE MASTER PROGRAM!
                                343.  *
00001224  4E75                  344.          RTS
                                345.  *
                                346.  * SUBROUTINE SENDOUT
                                347.  *
00001226  93C8                  348.  SENDOUT SUBA.L A0,A1
00001228  3009                  349.          MOVE.W A1,D0
0000122A  1CBC 0020             350.          MOVE.B #ENABLEB,(A6)
0000122E  4E47                  351.          TRAP #ADBYTOT
00001230  4E46                  352.          TRAP #OUTDATA
00001232  4216                  353.          CLR.B (A6)
00001234  4E75                  354.          RTS
00001236  20 20 20 20 04        355.  SPACES  DC.B '    ',EOT
0000123B  53 70 65 63 69        356.  TERMS   DC.B 'Specify even number of terms > '
          66 79 20 65 76
          65 6E 20 6E 75
          6D 62 65 72 20
          6F 66 20 74 65
          72 6D 73 20 3E
          20
0000125A  04                    357.          DC.B EOT
0000125B  4C 61 73 74 20        358.  HEAD    DC.B 'Last Den.    Sum       Exe'
          44 65 6E 2E 20
          20 20 20 53 75
          6D 20 20 20 20
          20 20 20 45 78
          65
00001275  2E 20 54 69 6D        359.          DC.B '. Time',EOT
          65 04
0000127C  20 20 20 04           360.  EXP     DC.B '   ',EOT
00001280  41 76 65 72 61        361.  AVEMES  DC.B 'Average time  ',EOT
          67 65 20 74 69
          6D 65 20 20 04
0000128F  4D 61 78 69 6D        362.  MAXMES  DC.B 'Maximum time  ',EOT
          75 6D 20 74 69
          6D 65 20 20 04
0000129E  4D 69 6E 69 6D        363.  MINMES  DC.B 'Minimum time  ',EOT
          75 6D 20 74 69
          6D 65 20 20 04
                                364.  ************************
                                365.  * FPADD HERE
                                366.  ************************
          -- boundary align
000012AE  48E7 3F00             367.  ADDFPM  MOVEM.L D2-D7,-(SP)     ;SAVE OFF
                                368.  *       CMPI.B #BGEXP,D0        ;A=NaN OR inf?
                                369.  *       BEQ CNANM               ;C=NaN IF SO
                                370.  *       CMPI.B #BGEXP,D1        ;B=NaN OR inf?
                                371.  *       BEQ CNANM               ;C=NaN IF SO
000012B2  4A81                  372.          TST.L D1                ;B=0?
```

```
000012B4  6700 0098      373.          BEQ UNSTACM              ;C=A IF SO
000012B8  4A80           374.          TST.L D0                 ;A=0?
000012BA  6700 00AA      375.          BEQ CISBM                ;C=B IF SO
000012BE  4243           376.          CLR.W D3
000012C0  4244           377.          CLR.W D4
000012C2  1600           378.          MOVE.B D0,D3             ;(D3.W)=EXPA
000012C4  1801           379.          MOVE.B D1,D4             ;(D4.W)=EXPB
000012C6  9644           380.          SUB.W D4,D3              ;(D3.W)=SHFTCNT
000012C8  0C43 0017      381.          CMPI.W #SIGBITS,D3       ;TOO BIG? C=A
000012CC  6C00 0080      382.          BGE UNSTACM
000012D0  0C43 FFE9      383.          CMPI.W #-SIGBITS,D3      ;TOO SMALL? C=B
000012D4  6F00 0090      384.          BLE CISBM
                         385.  *
                         386.  * HERE, MUST DO ADD
                         387.  * D5, D6 GET FRACTIONS; D7 CARRIES SIGN
                         388.  *
000012D8  7E00           389.          MOVEQ.L #0,D7
000012DA  0800 001F      390.          BTST.L #SGNMSK,D0        ;SIGN OF A
000012DE  67 04          391.          BEQ.S POSTVEM
000012E0  08C7 001F      392.          BSET.L #SGNMSK,D7        ;C IS NEG
000012E4  2A00           393.  POSTVEM MOVE.L D0,D5             ;(D5)=AF
000012E6  2C01           394.          MOVE.L D1,D6             ;(D6)=BF
000012E8  4205           395.          CLR.B D5
000012EA  4206           396.          CLR.B D6                 ;KILL EXPS
000012EC  E38D           397.          LSL.L #1,D5
000012EE  E38E           398.          LSL.L #1,D6              ;ALIGN
000012F0  7400           399.          MOVEQ.L #0,D2            ;
000012F2  1400           400.          MOVE.B D0,D2             ;CEXP=AEXP
000012F4  4A43           401.          TST.W D3                 ;TEST SC
000012F6  67 08          402.          BEQ.S NOADJM             ;IF=0 NO SHFTNG
000012F8  6B00 0070      403.          BMI SHFTAM               ;NEG, SHFT A
000012FC  E6AE           404.  SHFTBM  LSR.L D3,D6              ;SHIFTB
000012FE  4206           405.          CLR.B D6
00001300  2600           406.  NOADJM  MOVE.L D0,D3             ;COPY OF A
00001302  B383           407.          EOR.L D1,D3              ;SIGNS
00001304  0803 001F      408.          BTST.L #SGNMSK,D3        ;
00001308  6700 0052      409.          BEQ SAMSGNM              ;EQUAL SIGNS?
0000130C  4486           410.  DIFFSGM NEG.L D6                 ;NEG B
0000130E  DC85           411.          ADD.L D5,D6              ;D6 HAS THE SUM
00001310  6500 0042      412.          BCS NCMPCM               ;IF CAR=0, NOT=0
00001314  0847 001F      413.  COMPCM  BCHG.L #SGNMSK,D7        ;COMPLEMENT SUM
00001318  4486           414.          NEG.L D6
0000131A  0806 001F      415.  NORMCM  BTST.L #SGNMSK,D6        ;NORMALIZED?
0000131E  66 06          416.          BNE.S CNORMDM
00001320  E38E           417.          LSL.L #1,D6              ;ADJUST FRACTION
00001322  5342           418.          SUBQ.W #1,D2             ;ADJUST EXP
00001324  60 F4          419.          BRA.S NORMCM
                         420.  *
                         421.  * BOUND CHECK ON EXP
                         422.  *
00001326  4A42           423.  CNORMDM TST.W D2
00001328  6F00 002E      424.          BLE CZEROM
0000132C  0C42 00FF      425.          CMPI.W #BGEXP,D2
00001330  6700 0042      426.          BEQ CINFM
                         427.  *
                         428.  * ROUND OFF FRACTION
                         429.  *
00001334  0806 0009      430.  ROUNDCM BTST.L #ALSB,D6
00001338  66 0A          431.          BNE.S NOROUNM
0000133A  0806 0008      432.          BTST.L #ANTLSB,D6
0000133E  67 04          433.          BEQ.S NOROUNM
00001340  0046 0200      434.          ORI.W #ARNDDR,D6
                         435.  *
                         436.  * ALIGN RESULT
                         437.  *
00001344  E28E           438.  NOROUNM LSR.L #1,D6              ;ROOM FOR SIGN
00001346  4206           439.          CLR.B D6                 ;ROOM FOR EXP
                         440.  *
                         441.  * WITH BOUND CHECKS ON EXP, HAS LEADING ZEROES
                         442.  *
00001348  8486           443.          OR.L D6,D2
0000134A  8487           444.          OR.L D7,D2
0000134C  2002           445.          MOVE.L D2,D0
0000134E  4CDF 00FC      446.  UNSTACM MOVEM.L (SP)+,D2-D7
00001352  4E75           447.          RTS
                         448.  *
00001354  4A86           449.  NCMPCM  TST.L D6                 ;C=0?
00001356  66 C2          450           BNE.S NORMCM             ;NO, NORMALIZE
00001358  4280           451.  CZEROM  CLR.L D0
0000135A  60 F2          452.          BRA UNSTACM
0000135C  DC85           453.  SAMSGNM ADD.L D5,D6              ;D6 = SUM FRACT
```

```
0000135E  64 D4           454.          BCC.S ROUNDCM
00001360  E296            455.          ROXR.L #1,D6              ;IF CARRY,
00001362  5242            456.          ADDQ.W #1,D2              ;ADJUST RESULT
00001364  60 C0           457.          BRA.S CNORMDM             ;GO ROUND
                          458.  *CNANM  MOVEQ.L #0,D0
                          459.  *       SUBQ.L #1,D0              ;C=NaN
                          460.  *       BRA UNSTACM
00001366  2001            461.  CISBM   MOVE.L D1,D0
00001368  60 E4           462.          BRA UNSTACM
0000136A  4443            463.  SHFTAM  NEG.W D3                 ;POS SHIFT CNT
0000136C  E6AD            464.          LSR.L D3,D5               ;ADJUST A FRACT
0000136E  4205            465.          CLR.B D5
00001370  3401            466.          MOVE.W D1,D2              ;USE BEXP=CEXP
00001372  60 8C           467.          BRA NOADJM
00001374  7000            468.  CINFM   MOVEQ.L #0,D0
00001376  103C 00FF       469.          MOVE.B #BGEXP,D0          ;C=inf
0000137A  8087            470.          OR.L D7,D0                ;TAKE SIGN
0000137C  60 D0           471.          BRA UNSTACM               ;QUIT
                          472.  *
                          473.  *
                          474.  **********************
                          475.  * SLAVE #0 PROGRAM
                          476.  **********************
                          477.  * DOES INVERT
                          478.  * 1,1/5,1/9,1/13,...1/(#TERMS*2-2)
                          479.  * SLAVE 1 DOES ADDITIONS
                          480.  * NOTE NO CALLS TO THE I/O TRAP
                          481.  * ROUTINES FOR TRANSFERS IN PIPES
                          482.  *
                          483.  * NUMBER OF LOOPS RECEIVED ON DATA
                          484.  * UPLOAD AND IS AT TCOUNT
                          485.  *
                          486.  * CHECK TO SEE THAT SLAVE #1
                          487.  * READY TO RECEIVE
                          488.  *
0000137E  207C 00012FC9   489.  SLVPR0  MOVEA.L #PBDR,A0
00001384  227C 00012FCD   490.          MOVEA.L #PPSR,A1
0000138A  7202            491.          MOVEQ.L #H3S,D1
0000138C  137C 0020 FFF3  492.          MOVE.B #ENABLEB,-13(A1) ;EN OUTPUT PORT
00001392  0811 0006       493.  REDY0   BTST.B #H3LEV,(A1)        ;ADDER SLAVE READY?
00001396  66 FA           494.          BNE.S REDY0               ;WAIT TILL SO
00001398  3E3A FC66       495.          MOVE.W TCOUNT(PC),D7
                          496.  *
                          497.  * NOW START INVERT
                          498.  * GET NUMBER OF TERMS(/2-1 FOR DBRA)
                          499.  *
0000139C  7C01            500.          MOVEQ.L #1,D6   ;FIRST TERM=1
                          501.  *
                          502.  * INTEGER TO INVERT TO FP IS IN D6.W
                          503.  * RETURNS FP NUMBER IN D4.L
                          504.  * D6 NOT AFFECTED
                          505.  *
0000139E  45FA FC62       506.  INVERT0 LEA TEMPDAT(PC),A2
000013A2  7410            507.          MOVEQ.L #16,D2  ;D2 <- EXPONENT
000013A4  3A06            508.          MOVE.W D6,D5
000013A6  E35D            509.  SHFTLP0 ROL.W #1,D5     ;GET MSBIT=1 IN D5
000013A8  55CA FFFC       510.          DBCS D2,SHFTLP0
000013AC  E25D            511.          ROR.W #1,D5     ;ADJUST ONE BACK
000013AE  4402            512.          NEG.B D2        ;NEGATE EXPONENT
000013B0  D43C 0080       513.          ADD.B #BIAS,D2  ;BIAS
000013B4  7601            514.          MOVEQ.L #1,D3   ;FAST WAY FOR
000013B6  E29B            515.          ROR.L #1,D3     ;MOVE.L #$80000000,D3
000013B8  86C5            516.          DIVU.W D5,D3    ;DO MSW DIVIDE
000013BA  6900 0044       517.          BVS POWR20      ;IF OVRFLW, DIVISOR=2**k
000013BE  3803            518.          MOVE.W D3,D4
000013C0  4844            519.          SWAP.W D4       ;MSW OF QUO IN D4
000013C2  4243            520.          CLR.W D3        ;CLEAR LOW WORD D3
                          521.  *                       ;GIVES EXTENDED REMAINDER
000013C4  2603            522.          MOVE.L D3,D3    ;DUMMY MOVE FOR CCR Z
000013C6  67 14           523.          BEQ.S LASTEP0   ;REM=0? BRANCH, NO ROUND
000013C8  86C5            524.          DIVU.W D5,D3    ;GET LSW OF QUOTIENT
                          525.  *                       ;NEVER OVRFLW, IGNORE REM
000013CA  3803            526.          MOVE.W D3,D4    ;ALL BITS QUOTIENT -> D4
                          527.  *
                          528.  * round the result
                          529.  *
000013CC  0804 0009       530.  ROUNDQ0 BTST.L #ALSB,D4
000013D0  66 0A           531.          BNE.S LASTEP0
000013D2  0804 0008       532.          BTST.L #ANTLSB,D4
```

```
000013D6  67 04          533.          BEQ.S LASTEP0
000013D8  0044 0200      534.          ORI.W #ARNDDR,D4
000013DC  E28C           535.  LASTEP0 LSR.L #1,D4      ;SIGN=0
000013DE  5202           536.          ADDQ.B #1,D2
000013E0  1802           537.          MOVE.B D2,D4
                         538.  *
                         539.  * DONE INVERT, STORE OFF
                         540.  *
000013E2  2484           541.  STORE0  MOVE.L D4,(A2)
000013E4  7003           542.          MOVEQ.L #NTFR,D0
000013E6  0311           543.  TSTLP0  BTST.B D1,(A1)
000013E8  67 FC          544.          BEQ.S TSTLP0
000013EA  109A           545.          MOVE.B (A2)+,(A0)
000013EC  51C8 FFF8      546.          DBRA D0,TSTLP0
                         547.  *
                         548.  * CALC'D AND SENT, DO NEXT TERM
                         549.  *
000013F0  5846           550.  NEXTRM0 ADDQ.W #4,D6
000013F2  51CF FFAA      551.          DBRA D7,INVERT0
                         552.  *
                         553.  * DONE, WAIT TILL LAST SENT, THEN DISABLE
                         554.  *
000013F6  0311           555.  SENDL0  BTST.B D1,(A1)
000013F8  67 FC          556.          BEQ.S SENDL0
000013FA  4229 FFF3      557.          CLR.B -13(A1)
000013FE  4E75           558.          RTS
                         559.  *
                         560.  * OVERFLOWED, POWER OF 2 DIVISOR, QUO=DIVISOR*2
                         561.  * NO REMAINDER, NO ROUNDING REQUIRED
                         562.  *
00001400  4845           563.  POWR20  SWAP D5          ;MSW OF QUO
00001402  4245           564.          CLR.W D5        ;NO MORE SIG DIGITS
00001404  E28D           565.          LSR.L #1,D5     ;SIGN=0
00001406  5402           566.          ADDQ.B #2,D2    ;ADJUST EXPONENT
00001408  2805           567.          MOVE.L D5,D4
0000140A  1802           568.          MOVE.B D2,D4
0000140C  60 D4          569.          BRA STORE0
                         570.  *
                         571.  ****************
                         572.  * DONE SLAVE 0 *
                         573.  ****************
                         574.  *
                         575.  ********************************
                         576.  * SLAVE 1 AND SLAVE 3 PROGRAM *
                         577.  ********************************
                         578.  *
                         579.  * RECIEVES DATA FROM PREVIOUS SLAVE.
                         580.  * AND DOES REMANDER OF INVERT PHASE IF
                         581.  * NECESSARY. THEN DOES ADD FOR REQUIRED NUMBER
                         582.  * OF TERMS, THEN UPLOADS RESULT TO MASTER
                         583.  * NOTE NO SRQ MADE, JUST AWAITS MASTER'S
                         584.  * READY TO RECEIVE SIGNAL BEFORE SENDING
                         585.  *
                         586.  * SLAVE 1 DOES OPS ON TERMS FROM SLAVE 0
                         587.  * THE POSITVE SERIES, AND SLAVE 3 OPS ON
                         588.  * TERMS FROM SLAVE 2, THE NEGATIVE SERIES
                         589.  * THEN MASTER DOES (SLV1)-(SLV3)FOR PI/4
                         590.  *
                         591.  * FIRST THING TO DO IS ENABLE INPUT PORT,
                         592.  * AND AWAIT INPUT DATA
                         593.  *
                         594.  * LOOP COUNTER LOADED FROM MASTER AND IS AT
                         595.  * TCOUNT
                         596.  ********************************************
                         597.  *
0000140E  267C 00012FC0  598.  SLVPR13 MOVEA.L #PGCR,A3
00001414  227C 00012FCD  599.          MOVEA.L #PPSR,A1
0000141A  207C 00012FC8  600.          MOVEA.L #PADR,A0
00001420  16BC 0010      601.          MOVE.B #ENABLEA,(A3)
00001424  7400           602.          MOVEQ.L #H1S,D2
00001426  4280           603.          CLR.L D0        ;FIRST ADD TO 0
00001428  3E3A FBD6      604.          MOVE.W TCOUNT(PC),D7
0000142C  7603           605.  NEXTM13 MOVEQ.L #NTFR,D3
0000142E  45FA FBD2      606.          LEA TEMPDAT(PC),A2
00001432  0511           607.  REDY13  BTST.B D2,(A1)
00001434  67 FC          608.          BEQ.S REDY13
00001436  14D0           609.          MOVE.B (A0),(A2)+
00001438  51CB FFF8      610.          DBRA D3,REDY13
0000143C  223A FBC4      611.          MOVE.L TEMPDAT(PC),D1
00001440  48E7 2100      612.  ADDFP13 MOVEM.L D2/D7,-(SP)     ;SAVE OFF
                         613.  *       CMPI.B #BGEXP,D0        ;A=NaN OR inf?
```

```
                                614.  *       BEQ CNAN13              ;C=NaN IF SO
                                615.  *       CMPI.B #BGEXP,D1        ;B=NaN OR inf?
                                616.  *       BEQ CNAN13              ;C=NaN IF SO
00001444  4A81                  617.          TST.L D1                ;B=0?
00001446  6700 0098             618.          BEQ UNSTA13             ;C=A IF SO
0000144A  4A80                  619.          TST.L D0                ;A=0?
0000144C  6700 00AC             620.          BEQ CISB13              ;C=B IF SO
00001450  4243                  621.          CLR.W D3
00001452  4244                  622.          CLR.W D4
00001454  1600                  623.          MOVE.B D0,D3            ;(D3.W)=EXPA
00001456  1801                  624.          MOVE.B D1,D4            ;(D4.W)=EXPB
00001458  9644                  625.          SUB.W D4,D3             ;(D3.W)=SHFTCNT
0000145A  0C43 0017             626.          CMPI.W #SIGBITS,D3      ;TOO BIG? C=A
0000145E  6C00 0080             627.          BGE UNSTA13
00001462  0C43 FFE9             628.          CMPI.W #-SIGBITS,D3     ;TOO SMALL? C=B
00001466  6F00 0092             629.          BLE CISB13
                                630.  *
                                631.  * HERE, MUST DO ADD
                                632.  * D5, D6 GET FRACTIONS; D7 CARRIES SIGN
                                633.  *
0000146A  7E00                  634.          MOVEQ.L #0,D7
0000146C  0800 001F             635.          BTST.L #SGNMSK,D0       ;SIGN OF A
00001470  67 04                 636.          BEQ.S POST13
00001472  08C7 001F             637.          BSET.L #SGNMSK,D7       ;C IS NEG
00001476  2A00                  638.  POST13  MOVE.L D0,D5            ;(D5)=AF
00001478  2C01                  639.          MOVE.L D1,D6            ;(D6)=BF
0000147A  4205                  640.          CLR.B D5
0000147C  4206                  641.          CLR.B D6                ;KILL EXPS
0000147E  E38D                  642.          LSL.L #1,D5
00001480  E38E                  643.          LSL.L #1,D6             ;ALIGN
00001482  7400                  644.          MOVEQ.L #0,D2           ;
00001484  1400                  645.          MOVE.B D0,D2            ;CEXP=AEXP
00001486  4A43                  646.          TST.W D3                ;TEST SC
00001488  67 08                 647.          BEQ.S NOADJ13           ;IF=0 NO SHFTNG
0000148A  6B00 0072             648.          BMI SHFTA13             ;NEG, SHFT A
0000148E  E6AE                  649.  SHFTB13 LSR.L D3,D6              ;SHIFTB
00001490  4206                  650.          CLR.B D6
00001492  2600                  651.  NOADJ13 MOVE.L D0,D3            ;COPY OF A
00001494  B383                  652.          EOR.L D1,D3             ;SIGNS
00001496  0803 001F             653.          BTST.L #SGNMSK,D3       ;
0000149A  6700 0054             654.          BEQ SAMSG13             ;EQUAL SIGNS?
0000149E  4486                  655.  DIFFS13 NEG.L D6                ;NEG B
000014A0  DC85                  656.          ADD.L D5,D6             ;D6 HAS THE SUM
000014A2  6500 0044             657.          BCS NCMPC13             ;IF CAR=0, NOT=0
000014A6  0847 001F             658.  COMPC13 BCHG.L #SGNMSK,D7       ;COMPLEMENT SUM
000014AA  4486                  659.          NEG.L D6
000014AC  0806 001F             660.  NORMC13 BTST.L #SGNMSK,D6       ;NORMALIZED?
000014B0  66 06                 661.          BNE.S CNORM13
000014B2  E38E                  662.          LSL.L #1,D6             ;ADJUST FRACTION
000014B4  5342                  663.          SUBQ.W #1,D2            ;ADJUST EXP
000014B6  60 F4                 664.          BRA.S NORMC13
                                665.  *
                                666.  * BOUND CHECK ON EXP
                                667.  *
000014B8  4A42                  668.  CNORM13 TST.W D2
000014BA  6F00 0030             669.          BLE CZERO13
000014BE  0C42 00FF             670.          CMPI.W #BGEXP,D2
000014C2  6700 0044             671.          BEQ CINF13
                                672.  *
                                673.  * ROUND OFF FRACTION
                                674.  *
000014C6  0806 0009             675.  ROUND13 BTST.L #ALSB,D6
000014CA  66 0A                 676.          BNE.S NORON13
000014CC  0806 0008             677.          BTST.L #ANTLSB,D6
000014D0  67 04                 678.          BEQ.S NORON13
000014D2  0046 0200             679.          ORI.W #ARNDDR,D6
                                680.  *
                                681.  * ALIGN RESULT
                                682.  *
000014D6  E28E                  683.  NORON13 LSR.L #1,D6              ;ROOM FOR SIGN
000014D8  4206                  684.          CLR.B D6                ;ROOM FOR EXP
                                685.  *
                                686.  * WITH BOUND CHECKS ON EXP, HAS LEADING ZEROES
                                687.  *
000014DA  8486                  688.          OR.L D6,D2
000014DC  8487                  689.          OR.L D7,D2
000014DE  2002                  690.          MOVE.L D2,D0
000014E0  4CDF 0084             691.  UNSTA13 MOVEM.L (SP)+,D2/D7
000014E4  6000 002C             692.          BRA DUNAD13
                                693.  *
000014E8  4A86                  694.  NCMPC13 TST.L D6                ;C=0?
```

```
000014EA  66 C0           695.          BNE.S NORMC13           ;NO, NORMALIZE
000014EC  4280            696.  CZERO13 CLR.L D0
000014EE  60 F0           697.          BRA UNSTA13
000014F0  DC85            698.  SAMSG13 ADD.L D5,D6             ;D6 = SUM FRACT
000014F2  64 D2           699.          BCC.S ROUND13
000014F4  E296            700.          ROXR.L #1,D6            ;IF CARRY,
000014F6  5242            701.          ADDQ.W #1,D2            ;ADJUST RESULT
000014F8  60 BE           702.          BRA.S CNORM13           ;GO ROUND
                          703.  *CNAN13 MOVEQ.L #0,D0
                          704.  *       SUBQ.L #1,D0            ;C=NaN
                          705.  *       BRA UNSTA13
000014FA  2001            706.  CISB13  MOVE.L D1,D0
000014FC  60 E2           707.          BRA UNSTA13
000014FE  4443            708.  SHFTA13 NEG.W D3                ;POS SHIFT CNT
00001500  E6AD            709.          LSR.L D3,D5             ;ADJUST A FRACT
00001502  4205            710.          CLR.B D5
00001504  3401            711.          MOVE.W D1,D2            ;USE BEXP=CEXP
00001506  60 8A           712.          BRA NOADJ13
00001508  7000            713.  CINF13  MOVEQ.L #0,D0
0000150A  103C 00FF       714.          MOVE.B #BGEXP,D0        ;C=inf
0000150E  8087            715.          OR.L D7,D0              ;TAKE SIGN
00001510  60 CE           716.          BRA UNSTA13             ;QUIT
00001512  51CF FF18       717.  DUNAD13 DBRA D7,NEXTM13
00001516  4213            718.          CLR.B (A3)
00001518  41FA FAE8       719.          LEA TEMPDAT(PC),A0
0000151C  2080            720.          MOVE.L D0,(A0)
0000151E  4E43            721.          TRAP #SRQASRT
00001520  7004            722.          MOVEQ.L #NTFR+1,D0
00001522  16BC 0020       723.          MOVE.B #ENABLEB,(A3)
00001526  4E46            724.          TRAP #OUTDATA
00001528  4213            725.          CLR.B (A3)
0000152A  4E75            726.          RTS
                          727.  *
                          728.  ************************************
                          729.  * SLAVES 1 AND 3 PROGS DONE        *
                          730.  ************************************
                          731.  *
                          732.  **********************
                          733.  * SLAVE #2 PROGRAM
                          734.  **********************
                          735.  * DOES PARTIAL
                          736.  * 1/3,1/7,1/11,1/15...1/(#TERMS*2-1)
                          737.  * SLAVE #3 DOES ADDS
                          738.  * NOTE NO CALLS TO THE I/O TRAP
                          739.  * ROUTINES FOR TRANSFERS IN PIPES
                          740.  *
                          741.  * NUMBER OF LOOPS RECEIVED ON DATA
                          742.  * UPLOAD AND IS AT TCOUNT
                          743.  *
                          744.  * CHECK TO SEE THAT SLAVE #3
                          745.  * READY TO RECEIVE
                          746.  * USE A4 TO POINT TO PORT STATUS REG
                          747.  * AND A5 TO POINT TO PORT B DATA REG
                          748.  *
                          749.  *
0000152C  207C 00012FC9   750.  SLVPR2  MOVEA.L #PBDR,A0
00001532  227C 00012FCD   751.          MOVEA.L #PPSR,A1
00001538  7202            752.          MOVEQ.L #H3S,D1
0000153A  137C 0020 FFF3  753.          MOVE.B #ENABLEB,-13(A1) ;EN OUTPUT PORT
00001540  0811 0006       754.  REDY2   BTST.B #H3LEV,(A1)      ,ADDER SLAVE READY?
00001544  66 FA           755.          BNE.S REDY2             ;WAIT TILL SO
00001546  3E3A FAB8       756.          MOVE.W TCOUNT(PC),D7
                          757.  *
                          758.  * NOW START INVERT
                          759.  * GET NUMBER OF TERMS(/2-1 FOR DBRA)
                          760.  *
0000154A  7C03            761.          MOVEQ.L #3,D6   ;FIRST TERM=3
                          762.  *
                          763.  * INTEGER TO INVERT TO FP IS IN D6.W
                          764.  * RETURNS FP NUMBER IN D4.L
                          765.  * D6 NOT AFFECTED
                          766.  *
0000154C  45FA FAB4       767.  INVERT2 LEA TEMPDAT(PC),A2
00001550  7410            768.          MOVEQ.L #16,D2  ;D2 <- EXPONENT
00001552  3A06            769.          MOVE.W D6,D5
00001554  E35D            770.  SHFTLP2 ROL.W #1,D5     ;GET MSBIT=1 IN D5
00001556  55CA FFFC       771           DBCS D2,SHFTLP2
0000155A  E25D            772.          ROR.W #1,D5     ;ADJUST ONE BACK
0000155C  4402            773.          NEG.B D2        ;NEGATE EXPONENT
0000155E  D43C 0080       774.          ADD.B #BIAS,D2  ;BIAS
00001562  7601            775.          MOVEQ.L #1,D3   ;FAST WAY FOR
```

```
00001564  E29B           776.          ROR.L #1,D3      ;MOVE.L #$80000000,D3
00001566  86C5           777.          DIVU.W D5,D3     ;DO MSW DIVIDE
00001568  6900 0044      778.          BVS POWR22       ;IF OVRFLW, DIVISOR=2**k
0000156C  3803           779.          MOVE.W D3,D4
0000156E  4844           780.          SWAP.W D4        ;MSW OF QUO IN D4
00001570  4243           781.          CLR.W D3         ;CLEAR LOW WORD D3
                         782.  *                         ;GIVES EXTENDED REMAINDER
00001572  2603           783.          MOVE.L D3,D3     ;DUMMY MOVE FOR CCR 2
00001574  67 14          784.          BEQ.S LASTEP2    ;REM=0? BRANCH, NO ROUND
00001576  86C5           785.          DIVU.W D5,D3     ;GET LSW OF QUOTIENT
                         786.  *                         :NEVER OVRFLW, IGNORE REM
00001578  3803           787.          MOVE.W D3,D4     ;ALL BITS QUOTIENT -> D4
                         788.  *
                         789.  * round the result
                         790.  *
0000157A  0804 0009      791.  ROUNDQ2 BTST.L #ALSB,D4
0000157E  66 0A          792.          BNE.S LASTEP2
00001580  0804 0008      793.          BTST.L #ANTLSB,D4
00001584  67 04          794.          BEQ.S LASTEP2
00001586  0044 0200      795.          ORI.W #ARNDDR,D4
0000158A  E28C           796.  LASTEP2 LSR.L #1,D4      ;SIGN=0
0000158C  5202           797.          ADDQ.B #1,D2
0000158E  1802           798.          MOVE.B D2,D4
                         799.  *
                         800.  * DONE INVERT, STORE OFF
                         801.  *
00001590  2484           802.  STORE2  MOVE.L D4,(A2)
00001592  7003           803.          MOVEQ.L #NTFR,D0
00001594  0311           804.  TSTLP2  BTST.B D1,(A1)
00001596  67 FC          805.          BEQ.S TSTLP2
00001598  109A           806.          MOVE.B (A2)+,(A0)
0000159A  51C8 FFF8      807.          DBRA D0,TSTLP2
                         808.  *
                         809.  * CALC'D AND SENT, DO NEXT TERM
                         810.  *
0000159E  5846           811.  NEXTRM2 ADDQ.W #4,D6
000015A0  51CF FFAA      812.          DBRA D7,INVERT2
                         813.  *
                         814.  * DONE, WAIT TILL LAST SENT, THEN DISABLE
                         815.  *
000015A4  0311           816.  SENDL2  BTST.B D1,(A1)
000015A6  67 FC          817.          BEQ.S SENDL2
000015A8  4229 FFF3      818.          CLR.B -13(A1)
000015AC  4E75           819.          RTS
                         820.  *
                         821.  * OVERFLOWED, POWER OF 2 DIVISOR, QUO=DIVISOR*2
                         822.  * NO REMAINDER, NO ROUNDING REQUIRED
                         823.  *
000015AE  4845           824.  POWR22  SWAP D5          ;MSW OF QUO
000015B0  4245           825.          CLR.W D5         ;NO MORE SIG DIGITS
000015B2  E28D           826.          LSR.L #1,D5      ;SIGN=0
000015B4  5402           827.          ADDQ.B #2,D2     ;ADJUST EXPONENT
000015B6  2805           828.          MOVE.L D5,D4
000015B8  1802           829.          MOVE.B D2,D4
000015BA  60 D4          830.          BRA STORE2
                         831.  *
                         832.  **********************
                         833.  * DONE SLAVE 2  *
                         834.  **********************
000015BC  <2>            835.  DONE2   DS.W 1
000015BE                 836.          END
    0 Errors
```

N.4 Program PI4LSTST Uniprocessor Alternating Series Test Program, Lengthened sub-tasks.

```
                                    1.          TTL PI4LSTST
                                    2.  *
                                    3.  * PROGRAM FOR SINGLE PROCESSOR CALCULATION
                                    4.  * OF PI/4 USING ALTERNATING SERIES
                                    5.  * 0+1-1/3+1/5-1/7+1/9-... FOR A NUMBER
                                    6.  * OF TERMS up to 1/65535
                                    7.  *
                                    8.  * USES GENERAL INVERT SUBROUTINE TO GIVE
                                    9.  * FLOATING POINT REPRESENTATION OF AN
                                   10.  * INTEGER IN RANGE 1<=x<=65535
                                   11.  *
                                   12.  * CALLS A FLOATING POINT ADD ROUTINE
                                   13.  *
                                   14.  * FLOATING POINT REPRESENTATION IS 32-BITS
                                   15.  *
                                   16.  * sign(1), FRACT(23), EXP (8-BITS excess 128)
                                   17.  *
                                   18.  ********************
                                   19   * APRIL 28,1992
                                   20.  * VERSION HAS USER SPECIFIED # OF TERMS
                                   21.  * OUTPUT IS:
                                   22.  * SUM, TIME
                                   23.  * AND ARTIFICIALLY INCREASED EXECUTION TIME
                                   24.  ********************
                                   25.  *
                                   26.  * EQUATES
                                   27.  *
        # 00000007                 28.  SENBYC  EQU 7
        # 0000000D                 29.  CR      EQU $0D
        # 0000001A                 30.  DISBUF2 EQU 26
        # 00000018                 31.  DISBUF8 EQU 24
        # 00000011                 32.  OUTMESC EQU 17
        # 0000000A                 33.  SENCLFC EQU 10
        # 00000004                 34.  EOT     EQU $04
        # 0000001C                 35.  BLDNUM  EQU 28
        # 00FFFFFF                 36.  CNTSTRT EQU $00FFFFFF
        # 00012FD0                 37.  TCR     EQU $12FD0
        # 00000001                 38.  ENABLTM EQU 1
        # 0000000A                 39.  LF      EQU $0A
        # 000000FF                 40.  BGEXP   EQU $FF
        # 0000001F                 41.  SGNMSK  EQU 31
        # 00000080                 42.  BIAS    EQU 128
        # 00000009                 43.  ALSB    EQU 9
        # 00000008                 44.  ANTLSB  EQU 8
        # 00000200                 45.  ARNDDR  EQU $200
        # 00000017                 46.  SIGBITS EQU 23
        # 0000023A                 47.  DIVCNT  EQU $23A
        # 00000234                 48.  ADDCNT  EQU $234
                                   49.  *
                                   50.  *
00001000                           51.          ORG.L $1000
00001000  2C7C 00012FD0            52.          MOVEA.L #TCR,A6
00001006  4216                     53.          CLR.B (A6)
00001008  2D7C 00FFFFFF            54.          MOVE.L #CNTSTRT,2(A6)
          0002
00001010  4E4F                     55.  PI4     TRAP #15
00001012  000A                     56.          DC.W SENCLFC
00001014  41FA 006B                57.          LEA TERMS(PC),A0
00001018  4E4F                     58.          TRAP #15
0000101A  0011                     59.          DC.W OUTMESC
0000101C  4E4F                     60.          TRAP #15
0000101E  001C                     61.          DC.W BLDNUM
00001020  4E4F                     62.          TRAP #15
00001022  000A                     63.          DC.W SENCLFC
00001024  2F00                     64.          MOVE.L D0,-(SP)
00001026  41FA 0079                65.          LEA HEAD(PC),A0
0000102A  4E4F                     66.          TRAP #15
0000102C  0011                     67.          DC.W OUTMESC
0000102E  4E4F                     68.          TRAP #15
00001030  000A                     69.          DC.W SENCLFC
00001032  201F                     70.          MOVE.L (SP)+,D0
00001034  E280                     71.          ASR.L #1,D0      ;DIVIDE BY 2
00001036  5380                     72.          SUBQ.L #1,D0     ;ADJUST FOR DBRA
00001038  2E00                     73.          MOVE.L D0,D7     ;LOOP COUNTER
0000103A  4280                     74.          CLR.L D0         ;D0 <- PARTIAL SUMS
0000103C  4201                     75           CLR.B D1         ;
```

```
0000103E  7C01              76.          MOVEQ.L #1,D6   ;D6 <- TERM DENOM
                            77.  *
                            78.  * START TIME FOR DIVIDE
                            79.  *
00001040  1CBC 0001         80.  MNLOOP  MOVE.B #ENABLTM,(A6)
00001044  6100 006C         81.          BSR INVERT      ;RESULT IN D4.L
00001048  2204              82.          MOVE.L D4,D1
0000104A  6100 00C0         83.          BSR ADDFP       ;CALC ODD PARTIAL SUM
0000104E  5446              84.          ADDQ.W #2,D6    ;NEW TERM DENOM
00001050  6100 0060         85.          BSR INVERT      ;RESULT IN D4.L
00001054  08C4 001F         86.          BSET.L #SGNMSK,D4       ;NEGATE
00001058  2204              87.          MOVE.L D4,D1
0000105A  6100 00B0         88.          BSR ADDFP       ;CALC EVEN PARTIAL SUM
0000105E  5446              89.          ADDQ.W #2,D6    ;NEW DENOMINATOR
00001060  51CF FFDA         90.          DBRA D7,MNLOOP
00001064  4216              91.          CLR.B (A6)
00001066  4E4F              92.          TRAP #15
00001068  0018              93.          DC.W DISBUF8    ;OUTPUT SUM
0000106A  41FA 0012         94.          LEA SPACES(PC),A0
0000106E  4E4F              95.          TRAP #15
00001070  0011              96.          DC.W OUTMESC
00001072  6100 0174         97.          BSR OUTIME
00001076  4E4F              98.          TRAP #15
00001078  000A              99.          DC.W SENCLFC
0000107A  4E75             100.          RTS
                           101.  *
0000107C  20 20 20 20 04   102.  SPACES  DC.B '    ',EOT
00001081  53 70 65 63 69   103.  TERMS   DC.B 'Specify even number of terms > '
          66 79 20 65 76
          65 6E 20 6E 75
          6D 62 65 72 20
          6F 66 20 74 65
          72 6D 73 20 3E
          20
000010A0  04               104.          DC.B EOT
000010A1  53 75 6D 20 20   105.  HEAD    DC.B 'Sum          Time',EOT
          20 20 20 20 20
          20 20 54 69 6D
          65 04
                           106.  *
                           107.  *
                           108.  * SUBROUTINE INVERT
                           109.  * INTEGER TO INVERT TO FP IS IN D6.W
                           110.  * RETURNS FP NUMBER IN D4.L
                           111.  * D6.W UNAFFECTED
                           112.  *
                           113.  * VOLATILE REGISTERS
                           114.  * D2,D3,D4,D5
                           115.  * START WITH DUMMY WAIT
                           116.  *
000010B2  343C 023A        117.  INVERT  MOVE.W #DIVCNT,D2
000010B6  4E71             118.  DIVWAT  NOP
000010B8  4E71             119.          NOP
000010BA  51CA FFFA        120.          DBRA D2,DIVWAT
000010BE  7410             121.          MOVEQ.L #16,D2  ;D2 <- EXPONENT
000010C0  3A06             122.          MOVE.W D6,D5
000010C2  E35D             123.  SHFTLP  ROL.W #1,D5     ;GET MSBIT=1 IN D5
000010C4  55CA FFFC        124.          DBCS D2,SHFTLP
000010C8  E25D             125.          ROR.W #1,D5     ;ADJUST ONE BACK
000010CA  4402             126.          NEG.B D2        ;NEGATE EXPONENT
000010CC  D43C 0080        127.          ADD.B #BIAS,D2  ;BIAS
000010D0  7601             128.          MOVEQ.L #1,D3   ;FAST WAY FOR
000010D2  E29B             129.          ROR.L #1,D3     ;MOVE L #$80000000,D3
000010D4  86C5             130.          DIVU.W D5,D3    ;DO MSW DIVIDE
000010D6  69 26            131.          BVS.S POWR2     ;IF OVRFLW, DIVISOR=2**k
000010D8  3803             132.          MOVE.W D3,D4
000010DA  4844             133.          SWAP.W D4       ;MSW OF QUO IN D4
000010DC  4243             134.          CLR.W D3        ;CLEAR LOW WORD D3
                           135.  *                        ;GIVES EXTENDED REMAINDER
000010DE  2603             136.          MOVE.L D3,D3    ;DUMMY MOVE FOR CCR Z
000010E0  67 14            137           BEQ.S LASTEP    ;REM=0? BRANCH, NO ROUND
000010E2  86C5             138.          DIVU.W D5,D3    ;GET LSW OF QUOTIENT
                           139.  *                        ;NEVER OVRFLW, IGNORE REM
000010E4  3803             140.          MOVE.W D3,D4    ;ALL BITS QUOTIENT -> D4
                           141.  *
                           142.  * round the result
                           143.  *
000010E6  0804 0009        144.  ROUNDQ  BTST.L #ALSB,D4
000010EA  66 0A            145.          BNE.S LASTEP
000010EC  0804 0008        146.          BTST.L #ANTLSB,D4
000010F0  67 04            147.          BEQ.S LASTEP
```

```
000010F2  0044 0200       148.          ORI.W #ARNDDR,D4
000010F6  E28C            149.  LASTEP   LSR.L #1,D4      ;SIGN=0
000010F8  5202            150.          ADDQ.B #1,D2
000010FA  1802            151.          MOVE.B D2,D4
000010FC  4E75            152.          RTS
                          153.  *
                          154.  * OVERFLOWED, POWER OF 2 DIVISOR, QUO=DIVISOR*2
                          155.  * NO REMAINDER, NO ROUNDING REQUIRED
                          156.  *
000010FE  4845            157.  POWR2    SWAP D5          ;MSW OF QUO
00001100  4245            158.          CLR.W D5         ;NO MORE SIG DIGITS
00001102  E28D            159.          LSR.L #1,D5      ;SIGN=0
00001104  5402            160.          ADDQ.B #2,D2     ;ADJUST EXPONENT
00001106  2805            161.          MOVE.L D5,D4
00001108  1802            162.          MOVE.B D2,D4
0000110A  4E75            163.          RTS
                          164.  *
                          165.  * SUBROUTINE ADDFP
                          166.  *
                          167.  * FLOATING POINT ADDITION FOR 32-BIT
                          168.  * FLOATING POINT FORMAT:
                          169.  * SIGN (1)| FRACTION (23) | EXPONENT (8)
                          170.  * WHERE EXPONENT IS BIASED EXCESS 128;
                          171.  * FRACTION IS NORMALIZED
                          172.  * 0 = $00000000
                          173.  * (SIGN) INFINTY = $(8.OR.0)00000FF
                          174.  * NaN = $XXXXXXXFF
                          175.  *       WHERE XXXXXX IS NOT AS INFINITY
                          176.  *
                          177.  * DOES A*B=C WHERE
                          178.  * A IS IN D0
                          179.  * B IS IN D1
                          180.  * C RETURNS IN D0
                          181.  * IF A OR B = NaN, SETS C=NaN = $FFFFFFFF
                          182.  * IF A OR B = +-INFINITY, SET C=NaN
                          183.  *
                          184.  * START WITH DUMMY WAIT
                          185.  *
0000110C  48E7 3F00       186.  ADDFP    MOVEM.L D2-D7,-(SP)      ;SAVE OFF
00001110  343C 0234       187.          MOVE.W #ADDCNT,D2
00001114  4E71            188.  ADDWAT   NOP
00001116  4E71            189.          NOP
00001118  51CA FFFA       190.          DBRA D2,ADDWAT
                          191.  *       CMPI.B #BGEXP,D0         ;A=NaN OR inf?
                          192.  *       BEQ CNAN                 ;C=NaN IF SO
                          193.  *       CMPI.B #BGEXP,D1         ;B=NaN OR inf?
                          194.  *       BEQ CNAN                 ;C=NaN IF SO
0000111C  4A81            195.          TST.L D1                 ;B=0?
0000111E  6700 0098       196.          BEQ UNSTACK              ;C=A IF SO
00001122  4A80            197.          TST.L D0                 ;A=0?
00001124  6700 00AA       198.          BEQ CISB                 ;C=B IF SO
00001128  4243            199.          CLR.W D3
0000112A  4244            200.          CLR.W D4
0000112C  1600            201.          MOVE.B D0,D3             ;(D3.W)=EXPA
0000112E  1801            202.          MOVE.B D1,D4             ;(D4.W)=EXPB
00001130  9644            203.          SUB.W D4,D3              ;(D3.W)=SHFTCNT
00001132  0C43 0017       204.          CMPI.W #SIGBITS,D3       ;TOO BIG? C=A
00001136  6C00 0080       205.          BGE UNSTACK
0000113A  0C43 FFE9       206.          CMPI.W #-SIGBITS,D3      ;TOO SMALL? C=B
0000113E  6F00 0090       207.          BLE CISB
                          208.  *
                          209.  * HERE, MUST DO ADD
                          210.  * D5, D6 GET FRACTIONS; D7 CARRIES SIGN
                          211.  *
00001142  7E00            212.          MOVEQ.L #0,D7
00001144  0800 001F       213.          BTST.L #SGNMSK,D0        ;SIGN OF A
00001148  67 04           214.          BEQ.S POSTVE
0000114A  08C7 001F       215.          BSET.L #SGNMSK,D7        ;C IS NEG
0000114E  2A00            216.  POSTVE   MOVE.L D0,D5             ;(D5)=AF
00001150  2C01            217.          MOVE.L D1,D6             ;(D6)=BF
00001152  4205            218.          CLR.B D5
00001154  4206            219.          CLR.B D6                 ;KILL EXPS
00001156  E38D            220.          LSL.L #1,D5
00001158  E38E            221.          LSL.L #1,D6              ;ALIGN
0000115A  7400            222.          MOVEQ.L #0,D2            ;
0000115C  1400            223.          MOVE.B D0,D2             ;CEXP=AEXP
0000115E  4A43            224.          TST.W D3                 ;TEST SC
00001160  67 08           225.          BEQ.S NOADJ              ;IF=0 NO SHFTNG
00001162  6B00 0070       226.          BMI SHFTA                ;NEG, SHFT A
00001166  E6AE            227.  SHFTB    LSR.L D3,D6              ;SHIFTB
00001168  4206            228.          CLR.B D6
```

```
0000116A  2600            229.  NOADJ   MOVE.L D0,D3            ;COPY OF A
0000116C  B383            230.         EOR.L D1,D3             ;SIGNS
0000116E  0803 001F       231.         BTST.L #SGNMSK,D3       ;
00001172  6700 0052       232.         BEQ SAMSGN              ;EQUAL SIGNS?
00001176  4486            233.  DIFFSGN NEG.L D6               ;NEG B
00001178  DC85            234.         ADD.L D5,D6             ;D6 HAS THE SUM
0000117A  6500 0042       235.         BCS NCMPC               ;IF CAR=0, NOT=0
0000117E  0847 001F       236.  COMPC  BCHG.L #SGNMSK,D7       ;COMPLEMENT SUM
00001182  4486            237.         NEG.L D6
00001184  0806 001F       238.  NORMC  BTST.L #SGNMSK,D6       ;NORMALIZED?
00001188  66 06           239.         BNE.S CNORMD
0000118A  E38E            240.         LSL.L #1,D6             ;ADJUST FRACTION
0000118C  5342            241.         SUBQ.W #1,D2            ;ADJUST EXP
0000118E  60 F4           242.         BRA.S NORMC
                          243.  *
                          244.  * BOUND CHECK ON EXP
                          245.  *
00001190  4A42            246.  CNORMD  TST.W D2
00001192  6F00 002E       247.         BLE CZERO
00001196  0C42 00FF       248.         CMPI.W #BGEXP,D2
0000119A  6700 0042       249.         BEQ CINF
                          250.  *
                          251.  * ROUND OFF FRACTION
                          252.  *
0000119E  0806 0009       253.  ROUNDC  BTST.L #ALSB,D6
000011A2  66 0A           254.         BNE.S NOROUN
000011A4  0806 0008       255.         BTST.L #ANTLSB,D6
000011A8  67 04           256.         BEQ.S NOROUN
000011AA  0046 0200       257.         ORI.W #ARNDDR,D6
                          258.  *
                          259.  * ALIGN RESULT
                          260.  *
000011AE  E28E            261.  NOROUN  LSR.L #1,D6              ;ROOM FOR SIGN
000011B0  4206            262.         CLR.B D6                ;ROOM FOR EXP
                          263.  *
                          264.  * WITH BOUND CHECKS ON EXP, HAS LEADING ZEROES
                          265.  *
000011B2  8486            266.         OR L D6,D2
000011B4  8487            267.         OR.L D7,D2
000011B6  2002            268.         MOVE.L D2,D0
000011B8  4CDF 00FC       269.  UNSTACK MOVEM.L (SP)+,D2-D7
000011BC  4E75            270.         RTS
                          271.  *
000011BE  4A86            272.  NCMPC  TST.L D6                 ;C=0?
000011C0  66 C2           273.         BNE.S NORMC             ;NO, NORMALIZE
000011C2  4280            274.  CZERO  CLR.L D0
000011C4  60 F2           275.         BRA UNSTACK
000011C6  DC85            276.  SAMSGN  ADD.L D5,D6            ;D6 = SUM FRACT
000011C8  64 D4           277.         BCC.S ROUNDC
000011CA  E296            278.         ROXR.L #1,D6            ;IF CARRY,
000011CC  5242            279.         ADDQ.W #1,D2            ;ADJUST RESULT
000011CE  60 C0           280.         BRA.S CNORMD            ;GO ROUND
                          281.  *CNAN   MOVEQ.L #0,D0
                          282.  *       SUBQ.L #1,D0            ;C=NaN
                          283.  *       BRA UNSTACK
000011D0  2001            284.  CISB   MOVE.L D1,D0
000011D2  60 E4           285.         BRA UNSTACK
000011D4  4443            286.  SHFTA  NEG.W D3                ;POS SHIFT CNT
000011D6  E6AD            287.         LSR.L D3,D5             ;ADJUST A FRACT
000011D8  4205            288.         CLR.B D5
000011DA  3401            289.         MOVE.W D1,D2            ;USE BEXP=CEXP
000011DC  60 8C           290.         BRA NOADJ
000011DE  7000            291.  CINF   MOVEQ.L #0,D0
000011E0  103C 00FF       292.         MOVE.B #BGEXP,D0        ;C=inf
000011E4  8087            293.         OR.L D7,D0              ,TAKE SIGN
000011E6  60 D0           294.         BRA UNSTACK             ;QUIT
                          295.  *
                          296.  *
                          297.  * SUBROUTINE OUTIME
                          298.  * CALCULATES ELAPSED TIME AND OUTPUTS
                          299.  * ALL REGISTERS TRANSPARENT
                          300.  *
000011E8  48E7 8002       301.  OUTIME  MOVEM.L D0/A6,-(SP)
000011EC  2C7C 00012FD0   302.         MOVEA.L #TCR,A6
000011F2  203C 00FFFFFF   303.         MOVE.L #CNTSTRT,D0 ;CALC ELAPSED
000011F8  90AE 0006       304.         SUB.L 6(A6),D0
000011FC  4E4F            305.         TRAP #15
000011FE  0018            306.         DC.W DISBUF8
00001200  4CDF 4001       307.         MOVEM.L (SP)+,D0/A6
00001204  4E75            308.         RTS
                          309.  *
```

```
                                310.   * DONE
                                311.   *
00001206                        312.           END
    0 Errors
```

N.5 Program PI4LMTST Multicomputer Alternating Series Test Program, Lengthened sub-tasks.

```
                                1.          TTL PI4LMTST
                                2.  *
                                3.  * PROGRAM FOR 4 PROCESSOR CALCULATION
                                4.  * OF PI/4 USING ALTERNATING SERIES
                                5.  * 0+1-1/3+1/5-1/7+1/9-... FOR FIXED NUMBER
                                6.  * OF TERMS <=1/65535
                                7.  *
                                8.  * SLAVE 0 AND SLAVE 2 DO THE INVERT STEP
                                9.  * TO GIVE FLOATING POINT REPRESENTATION OF AN
                               10.  * INTEGER IN RANGE 1<=x<=65535
                               11.  * SLAVE 0 - 1,5,9,... (TERMS WITH POSITIVE)
                               12.  * SLAVE 2 - 3,7,11,...(TERMS WITH NEGATIVE)
                               13.  * SLAVES 1 AND 3 DO THE ADD
                               14.  * STEP.
                               15.  * SLAVE 1 -RUNNING SUM OF TERMS FROM SLAVE 0
                               16.  * SLAVE 3 -RUNNING SUM OF TERMS FROM SLAVE 2
                               17.  * MASTER COLLECTS AT END AND DOES LAST
                               18.  * SUBTRACT SLAVE1SUM-SLAVE3SUM
                               19.  *
                               20.  * SIGN (1-BIT)
                               21.  * FRACTION (23-BITS),
                               22.  * EXPONENT (8-BITS SIGNED, EXCESS 128)
                               23.  *
                               24.  **********************
                               25.  * APRIL 28,1992      *
                               26.  * VERSION TIMED,     *
                               27.  * MASTER TIMER ONLY  *
                               28.  * SLAVES ARE EXTENDED TIME*
                               29.  **********************
                               30.  *
                               31.  * EQUATES
                               32.  *
# 00000003                     33.  NTFR     EQU 3    ;BYTES TFRED-1
# 00000018                     34.  DISBUF8  EQU 24
# 00000019                     35.  DISBUF4  EQU 25
# 0000001A                     36.  DISBUF2  EQU 26
# 00000011                     37.  OUTMESC  EQU 17
# 0000000A                     38.  SENCLFC  EQU 10
# 00000004                     39.  EOT      EQU $04
# 00012FC0                     40.  PGCR     EQU $12FC0
# 00012FC6                     41.  PACR     EQU $12FC6
# 00012FC7                     42.  PBCR     EQU $12FC7
# 00012FC1                     43.  PSRR     EQU $12FC1
# 00012FC8                     44.  PADR     EQU $12FC8
# 00012FCD                     45.  PPSR     EQU $12FCD
# 00012FC9                     46.  PBDR     EQU $12FC9
# 00000006                     47.  H3LEV    EQU 6
# 00000002                     48.  H3S      EQU 2
# 00000000                     49.  H1S      EQU 0
# 00000010                     50.  ENABLEA  EQU $10
# 00000020                     51.  ENABLEB  EQU $20
# 00000030                     52.  ENABLAB  EQU $30
# 0000000E                     53.  ABCRT    EQU 14
# 00000003                     54.  SRQASRT  EQU 3
# 00000004                     55.  ADBYTIN  EQU 4
# 00000007                     56.  ADBYTOT  EQU 7
# 00000005                     57.  INDATA   EQU 5
# 00000006                     58.  OUTDATA  EQU 6
# 00000008                     59.  NETCNF   EQU 8
# 00000009                     60.  SRQSRVC  EQU 9
# 0000001C                     61.  BLDNUM   EQU 28
# 00000001                     62.  SLV0     EQU $01
# 00000002                     63.  SLV1     EQU $02
# 00000004                     64.  SLV2     EQU $04
# 00000008                     65.  SLV3     EQU $08
# 0000000F                     66.  ALLSLV   EQU $0F
# 00012FB0                     67.  PSR01    EQU $12FB0
# 00012FB1                     68.  PSR23    EQU $12FB1
# 00012FB2                     69.  PSR45    EQU $12FB2
# 00012FB3                     70.  PSR67    EQU $12FB3
# 00012FB4                     71.  PSR89    EQU $12FB4
# 00012FB5                     72.  PSR1011  EQU $12FB5
# 00012FB6                     73.  PSR1213  EQU $12FB6
# 00012FB7                     74.  PSR1415  EQU $12FB7
# 00012FB8                     75.  PSR1617  EQU $12FB8
# 00012FB9                     76.  PSR1819  EQU $12FB9
```

```
                                    77.  TCR      EQU $12FD0
                                    78.  TPLR     EQU TCR+2
                                    79.  TCNTR    EQU TCR+6
                                    80.  CNTSTRT EQU $00FFFFFF
                                    81.  ENABLTM EQU 1
                                    82.  PASSCNT EQU 256
                                    83.  DIV256  EQU 8
                                    84.  BGEXP   EQU $FF
                                    85.  SGNMSK  EQU 31
                                    86.  BIAS    EQU 128
                                    87.  ALSB    EQU 9
                                    88.  ANILSB  EQU 8
                                    89.  ARNDDR  EQU $200
                                    90.  SIGBITS EQU 23
                                    91.  DIVCNT  EQU $23A
                                    92.  ADDCNT  EQU $234
                                    93.  *
00001000                            94.          ORG.L $1000
00001000  <2>                     95.  TCOUNT   DS.W 1   ;INPUT DATA FOR SLAVES
                                    96.  *               ;LOOP COUNTER FOR TERMS
                                    97.  *               ;ADJUST FOR DBRA
                                    98.  *               ;#TERMS/2-1
0000_002  <4>                      99.  TEMPDAT DS.W 2
00001006  <4>                     100.  POSDAT  DS.W 2
0000100A  <4>                     101.  NEGDAT  DS.W 2
0000100E  BB BB BB CB BB          102.  INITCNF DC.B $BB,$BB,$BB,$CB,$BB,$8C,$CB,$8B
          8C CB 8B
00001016  B8 8B                   103.          DC.B $B8,$8B
00001018  BD BB BD CB BD          104.  BRDCAST DC.B $BD,$BB,$BD,$CB,$BD,$8C,$CB,$8B
          8C CB 8B
00001020  B8 8B                   105.          DC.B $B8,$8B
00001022  00 00 00 70 00          106.  PARPIPS DC.B $00,$00,$00,$70,$00,$B8,$70,$80
          B8 70 80
0000102A  B8 8B                   107.          DC.B $B8,$8B
                                  108.  *
                                  109.  * CONFIGURE NETWORK TO INITIAL STATE, MASTER
                                  110.  * TO SLAVE 0
                                  111.  * RESPOND TO SLAVE 0'S RFP SRQ, AND LOAD
                                  112.  * ITS PROGRAM
                                  113.  *
0000102C  287C 00012FD0           114.  STRTSER MOVEA.L #TCR,A4          ;POINTER TIMER
00001032  4214                    115.          CLR.B (A4)               ;DISABLE F'SURE
00001034  2F3C 00FFFFFF           116.          MOVE.L #CNTSTRT,-(SP)    ;MIN TIME
0000103A  2F3C 00000000           117.          MOVE.L #$00,-(SP)        ;MAX TIME
00001040  2F3C 00000000           118.          MOVE.L #$00,-(SP)        ;RUN'G SUM TIME
00001046  2E3C 000000FF           119.          MOVE.L #PASSCNT-1,D7     ;PASS COUNTER
0000104C  2A7C 00012FCD           120.          MOVEA.L #PPSR,A5         ;A5 POINTR PPSR
00001052  2C7C 00012FC0           121.          MOVEA.L #PGCR,A6         ;A6 POINTR PADR
00001058  247C 00012FB9           122.          MOVEA.L #PSR1819,A2
0000105E  41FA FFAE               123.  MASTER   LEA INITCNF(PC),A0
00001062  4E48                    124.          TRAP #NETCNF
00001064  103C 0001               125.          MOVE.B #SLV0,D0
00001068  4E49                    126.          TRAP #SRQSRVC
0000106A  41FA 0312               127.          LEA SLVPR0(PC),A0        ;STRT ADDR
0000106E  43FA 03AA               128.          LEA SLVPR13(PC),A1       ,END ADDR+1
00001072  6100 01B2               129.          BSR SENDOUT              ;ADDR,CNT,CODE
                                  130.  *
                                  131.  * CONFIGURE NETWORK TO MASTER TO SLAVE 1 AND 3
                                  132.  * RESPOND TO SLAVE 1 AND 3'S RFP SRQ, AND LOAD
                                  133.  * THEIR COMMON PROGRAM
                                  134.  *
00001076  13FC 00B4               135.          MOVE.B #$B4,PSR01
          00012FB0
0000107E  13FC 00BD               136.          MOVE.B #$BD,PSR45
          00012FB2
00001086  13FC 00B4               137.          MOVE.B #$B4,PSR89
          00012FB4
0000108E  103C 000A               138.          MOVE.B #$A,D0    ;SERVICE 1 AND 3
00001092  4E49                    139.          TRAP #SRQSRVC
00001094  41FA 0384               140.          LEA SLVPR13(PC),A0
00001098  43FA 04AA               141.          LEA SLVPR2(PC),A1
0000109C  6100 0188               142.          BSR SENDOUT
                                  143.  *
                                  144.  * CONFIGURE NETWORK TO MASTER TO SLAVE 2
                                  145.  * RESPOND TO SLAVE 2'S RFP SRQ, AND LOAD ITS
                                  146.  * PROGRAM
                                  147.  *
000010A0  13FC 00B4               148.          MOVE.B #$B4,PSR45
          00012FB2
000010A8  13FC 00BB               149.          MOVE.B #$BB,PSR89
          00012FB4
```

```
000010B0  103C 0004       150.          MOVE.B #SLV2,D0
000010B4  4E49            151.          TRAP #SRQSRVC
000010B6  41FA 048C       152.          LEA SLVPR2(PC),A0
000010BA  43FA 0524       153.          LEA DONE2(PC),A1
000010BE  6100 0166       154.          BSR SENDOUT
                          155.  *
                          156.  * ALL PROGRAMS LOADED, EACH SLAVE ASSERTING
                          157.  * READY FOR DATA SRQ. ACQUIRE USER INPUT OF
                          158.  * NUMBER OF TERMS, ADJUST FOR THE ARCHITECTURE
                          159.  * AND DBRA, AND STORE AT TCOUNT. THEN RESPOND
                          160.  * TO ALL SLAVES SRQ'S, THEN LOAD THE DATA
                          161.  * IN BROADCAST MODE.
                          162.  *
000010C2  4E4F            163.  ACQCNT  TRAP #15
000010C4  000A            164.          DC.W SENCLFC
000010C6  41FA 0173       165.          LEA TERMS(PC),A0
000010CA  4E4F            166.          TRAP #15
000010CC  0011            167.          DC.W OUTMESC
000010CE  4E4F            168.          TRAP #15
000010D0  001C            169.          DC.W BLDNUM
000010D2  4E4F            170.          TRAP #15
000010D4  000A            171.          DC.W SENCLFC
000010D6  E280            172.          ASR.L #1,D0      ;DIVIDE BY 2
000010D8  5340            173.          SUBQ.W #1,D0     ;ADJUST FOR DBRA
000010DA  41FA FF24       174.          LEA TCOUNT(PC),A0
000010DE  3080            175.          MOVE.W D0,(A0)
000010E0  41FA 0179       176.          LEA HEAD(PC),A0
000010E4  4E4F            177.          TRAP #15
000010E6  0011            178.          DC.W OUTMESC
000010E8  4E4F            179.          TRAP #15
000010EA  000A            180.          DC.W SENCLFC
                          181.  *
                          182.  * THIS IS REPEATED 256 TIMES AND MEASURED
                          183.  * AND STATS KEPT
                          184.  *
000010EC  297C 00FFFFFF   185.          MOVE.L #CNTSTRT,2(A4)
          0002
000010F4  41FA FF22       186.  REXE    LEA BRDCAST(PC),A0
000010F8  4E48            187.          TRAP #NETCNF
000010FA  103C 000F       188.          MOVE.B #ALLSLV,D0
000010FE  4E49            189.          TRAP #SRQSRVC
00001100  41FA FEFE       190.          LEA TCOUNT(PC),A0        ;START OF DATA
00001104  43FA FEFC       191.          LEA TEMPDAT(PC),A1       ;END OF DATA+1
00001108  18BC 0001       192.  TIMSTRT MOVE.B #ENABLTM,(A4)     ;TIMER STARTS
0000110C  6100 0118       193.          BSR SENDOUT
                          194.  *
                          195.  * WHEN HERE, ALL SLAVES LOADED, WILL ASSERT
                          196.  * READY TO EXECUTE SRQ.  RESPOND TO LET THEM
                          197.  * CONTINUE.  BUT FIRST, CONFIGURE NETWORK
                          198.  * INTO 2 PARALLEL PATHS WITH SLV0 TO SLV1 TO
                          199.  * MASTER AND SLV2 TO SLV3 NEARLY TO MASTER,
                          200.  * ST SWITCH FROM SLV1 TO MASTER TO SLV3 TO
                          201.  * MASTER IS DONE IN ONE PSR REGISTER WRITE.
                          202.  *
00001110  41FA FF10       203.          LEA PARPIPS(PC),A0
00001114  4E48            204.          TRAP #NETCNF
00001116  103C 000F       205.          MOVE.B #ALLSLV,D0
0000111A  4E49            206.          TRAP #SRQSRVC            ;SLAVES EXCUTE
                          207.  *
                          208.  * WAIT FOR RESULTS FROM SLAVE 1 AND UPLOAD
                          209.  * THE FP DATA
                          210.  *
0000111C  103C 0002       211.          MOVE.B #SLV1,D0
00001120  4E49            212.          TRAP #SRQSRVC
00001122  7004            213.          MOVEQ.L #4,D0
00001124  41FA FEE0       214.          LEA POSDAT(PC),A0
00001128  1CBC 0010       215.          MOVE.B #ENABLEA,(A6)     ;EN INPUT PORT
0000112C  4E45            216.          TRAP #INDATA
0000112E  4216            217.          CLR.B (A6)
                          218.  *
                          219.  * CONFIGURE TO SLAVE3 TO MASTER
                          220.  * AND GET NEGATIVE TERM SUM
                          221.  *
00001130  14BC 00BB       222.          MOVE.B #$BB,(A2)
00001134  103C 0008       223.          MOVE.B #SLV3,D0
00001138  4E49            224.          TRAP #SRQSRVC
0000113A  7004            225.          MOVEQ.L #4,D0
0000113C  41FA FECC       226.          LEA NEGDAT(PC),A0
00001140  1CBC 0010       227.          MOVE.B #ENABLEA,(A6)
00001144  4E45            228.          TRAP #INDATA
00001146  4216            229.          CLR.B (A6)
```

```
                                230.  *
                                231.  * NOW DO POSDAT+NEGDAT
                                232.  *
00001148  203A FEBC             233.          MOVE.L POSDAT(PC),D0
0000114C  223A FEBC             234.          MOVE.L NEGDAT(PC),D1
00001150  08C1 001F             235.          BSET.L #SGNMSK,D1       ;NEGATE D1
00001154  6100 0158             236.          BSR ADDFPM
                                237.  *
                                238.  ******************************************
                                239.  * STOP TIMER AND DO STATS ACCUMULATION
                                240.  ******************************************
00001158  4214                  241.  OUTRES  CLR.B (A4)       ;TIME STOPPED
0000115A  2F00                  242.          MOVE.L D0,-(SP) ;SAVE RESULT
                                243.  *
                                244.  * CALCULATE LAST TERM DENOMINATOR
                                245.  *
0000115C  4280                  246.          CLR.L D0
0000115E  303A FEA0             247.          MOVE.W TCOUNT(PC),D0
00001162  5240                  248.          ADDQ.W #1,D0
00001164  E540                  249.          ASL.W #2,D0     ;MULTIPLY BY 4
00001166  5340                  250.          SUBQ.W #1,D0    ;LAST DENOM
00001168  4E4F                  251.          TRAP #15
0000116A  0018                  252.          DC.W DISBUF8    ;OUTPUT
0000116C  41FA 00C8             253.          LEA SPACES(PC),A0
00001170  4E4F                  254.          TRAP #15
00001172  0011                  255.          DC.W OUTMESC
00001174  201F                  256.          MOVE.L (SP)+,D0 ;GET RESULTS
00001176  4E4F                  257.          TRAP #15
00001178  0018                  258.          DC.W DISBUF8    ;OUTPUT RESULT
0000117A  41FA 00BA             259.          LEA SPACES(PC),A0
0000117E  4E4F                  260.          TRAP #15
00001180  0011                  261.          DC.W OUTMESC
                                262.  *
                                263.  * NOW ELAPSED TIME
                                264.  *
00001182  222C 0006             265.          MOVE.L 6(A4),D1 ;GET NEW COUNT
00001186  203C 00FFFFFF         266.          MOVE.L #CNTSTRT,D0
0000118C  9081                  267.          SUB.L D1,D0     ;GET ELAPSED COUNT
                                268.  *
                                269.  * OUTPUT THE COUNT IN HEX
                                270.  *
0000118E  4E4F                  271.          TRAP #15
00001190  0018                  272.          DC.W DISBUF8
00001192  2F00                  273.          MOVE.L D0,-(SP)
00001194  41FA 00A0             274.          LEA SPACES(PC),A0
00001198  4E4F                  275.          TRAP #15
0000119A  0011                  276.          DC.W OUTMESC
0000119C  3007                  277.          MOVE.W D7,D0
0000119E  4E4F                  278.          TRAP #15
000011A0  0019                  279.          DC.W DISBUF4
000011A2  4E4F                  280.          TRAP #15
000011A4  000A                  281.          DC.W SENCLFC
000011A6  201F                  282.          MOVE.L (SP)+,D0
                                283.  *
                                284.  * NOW KEEP TRACK OF STATISTICS
                                285.  *
000011A8  D197                  286.          ADD.L D0,(SP)   ;ADD TO SUM
000011AA  B0AF 0004             287.          CMP.L 4(SP),D0  ;CHECK MAX
000011AE  6F 04                 288.          BLE.S NOTMAX    ;D0 NOT MAX
000011B0  2F40 0004             289.          MOVE.L D0,4(SP) ;NEW MAX
000011B4  B0AF 0008             290.  NOTMAX  CMP.L 8(SP),D0  ;CHECK MIN
000011B8  6C 0C                 291.          BGE.S NOTMIN    ;D0 NOT MIN
000011BA  2F40 0008             292.          MOVE.L D0,8(SP) ;NEW MIN
                                293.  *
                                294.  * DO WHOLE PROCESS AGAIN, 256 TIMES
                                295.  *
000011BE  42B8 1006             296.          CLR.L POSDAT
000011C2  42B8 100A             297.          CLR.L NEGDAT
000011C6  51CF FF2C             298.  NOTMIN  DBRA D7,REXE
                                299.  *
                                300.  * WHEN HERE, PROG HAS BEEN EXECUTED 256 TIMES
                                301.  * SEND AN ILLEGAL ADDRESS FOR DATA, FORCING
                                302.  * SLAVES TO PROGRAM ACCEPT MODE
                                303.  *
000011CA  41FA FE4C             304.          LEA BRDCAST(PC),A0
000011CE  4E48                  305.          TRAP #NETCNF
000011D0  103C 000F             306.          MOVE.B #ALLSLV,D0
000011D4  4E49                  307.          TRAP #SRQSRVC
000011D6  207C FFFF0000         308.          MOVEA.L #$FFFF0000,A0   ;ILLEGAL ADDR
000011DC  1CBC 0020             309.          MOVE.B #ENABLEB,(A6)
000011E0  4E47                  310.          TRAP #ADBYTOT   ;THAT'LL FIX 'EM
```

```
000011E2  4216                311.          CLR.B (A6)
                              312.  *
                              313.  * OUTPUT STATISTICS
                              314.  *
000011E4  4E4F                315.          TRAP #15
000011E6  000A                316.          DC.W SENCLFC
000011E8  4E4F                317.          TRAP #15
000011EA  000A                318.          DC.W SENCLFC
000011EC  41FA 0092           319.          LEA AVEMES(PC),A0
000011F0  4E4F                320.          TRAP #15
000011F2  0011                321.          DC.W OUTMESC
000011F4  201F                322.          MOVE.L (SP)+,D0 ;GET SUM
000011F6  E080                323.          ASR.L #DIV256,D0          ;DO AVERAGE
000011F8  4E4F                324.          TRAP #15
000011FA  0018                325.          DC.W DISBUF8     ;OUTPUT AVERAGE
000011FC  4E4F                326.          TRAP #15
000011FE  000A                327.          DC.W SENCLFC
00001200  41FA 008D           328.          LEA MAXMES(PC),A0
00001204  4E4F                329.          TRAP #15
00001206  0011                330.          DC.W OUTMESC
00001208  201F                331.          MOVE.L (SP)+,D0 ;GET MAX
0000120A  4E4F                332.          TRAP #15
0000120C  0018                333.          DC.W DISBUF8
0000120E  4E4F                334.          TRAP #15
00001210  000A                335.          DC.W SENCLFC
00001212  41FA 008A           336.          LEA MINMES(PC),A0
00001216  4E4F                337.          TRAP #15
00001218  0011                338.          DC.W OUTMESC
0000121A  201F                339.          MOVE.L (SP)+,D0 ;GET MIN
0000121C  4E4F                340.          TRAP #15
0000121E  0018                341.          DC.W DISBUF8
00001220  4E4F                342.          TRAP #15
00001222  000A                343.          DC.W SENCLFC
                              344.  *
                              345.  * DONE MASTER PROGRAM!
                              346.  *
00001224  4E75                347.          RTS
                              348.  *
                              349.  * SUBROUTINE SENDOUT
                              350.  *
00001226  93C8                351.  SENDOUT SUBA.L A0,A1
00001228  3009                352.          MOVE.W A1,D0
0000122A  1CBC 0020           353.          MOVE.B #ENABLEB,(A6)
0000122E  4E47                354.          TRAP #ADBYTOT
00001230  4E46                355.          TRAP #OUTDATA
00001232  4216                356.          CLR.B (A6)
00001234  4E75                357.          RTS
00001236  20 20 20 20 04      358.  SPACES  DC B '    ',EOT
0000123B  53 70 65 63 69      359.  TERMS   DC.B 'Specify even number of terms > '
          66 79 20 65 76
          65 6E 20 6E 75
          6D 62 65 72 20
          6F 66 20 74 65
          72 6D 73 20 3E
          20
0000125A  04                  360.          DC.B EOT
0000125B  4C 61 73 74 20      361.  HEAD    DC.B 'Last Den.    Sum       Exe'
          44 65 6E 2E 20
          20 20 20 53 75
          6D 20 20 20 20
          20 20 20 45 78
          65
00001275  2E 20 54 69 6D      362.          DC.B '. Time',EOT
          65 04
0000127C  20 20 20 04         363.  EXP     DC.B '   ',EOT
00001280  41 76 65 72 61      364.  AVEMES  DC.B 'Average time  ',EOT
          67 65 20 74 69
          6D 65 20 20 04
0000128F  4D 61 78 69 6D      365.  MAXMES  DC.B 'Maximum time  ',EOT
          75 6D 20 74 69
          6D 65 20 20 04
0000129E  4D 69 6E 69 6D      366.  MINMES  DC.B 'Minimum time  ',EOT
          75 6D 20 74 69
          6D 65 20 20 04
                              367.  *************************
                              368.  * FPADD HERE
                              369.  *************************
          -- boundary align
000012AE  48E7 3F00           370.  ADDFPM  MOVEM.L D2-D7,-(SP)      ;SAVE OFF
                              371.  *       CMPI.B #BGEXP,D0         ;A=NaN OR inf?
                              372.  *       BEQ CNANM                ,C=NaN IF SO
```

```
                                373.  *         CMPI.B #BGEXP,D1        ;B=NaN OR inf?
                                374.  *         BEQ CNANM               ;C=NaN IF SO
000012B2  4A01                  375.            TST.L D1                ;B=0?
000012B4  6700 0098             376.            BEQ UNSTACM             ;C=A IF SO
000012B8  4A80                  377.            TST.L D0                ;A=0?
000012BA  6700 00AA             378.            BEQ CISBM               ;C=B IF SO
000012BE  4243                  379.            CLR.W D3
000012C0  4244                  380.            CLR.W D4
000012C2  1600                  381.            MOVE.B D0,D3            ;(D3.W)=EXPA
000012C4  1801                  382.            MOVE.B D1,D4            ;(D4.W)=EXPB
000012C6  9644                  383.            SUB.W D4,D3             ;(D3.W)=SHFTCNT
000012C8  0C43 0017             384.            CMPI.W #SIGBITS,D3      ;TOO BIG? C=A
000012CC  6C00 0080             385.            BGE UNSTACM
000012D0  0C43 FFE9             386.            CMPI.W #-SIGBITS,D3     ;TOO SMALL? C=B
000012D4  6F00 0090             387.            BLE CISBM
                                388.  *
                                389.  * HERE, MUST DO ADD
                                390.  * D5, D6 GET FRACTIONS; D7 CARRIES SIGN
                                391.  *
000012D8  7E00                  392.            MOVEQ.L #0,D7
000012DA  0800 001F             393.            BTST.L #SGNMSK,D0       ;SIGN OF A
000012DE  67 04                 394.            BEQ.S POSTVEM
000012E0  08C7 001F             395.            BSET.L #SGNMSK,D7       ;C IS NEG
000012E4  2A00                  396.  POSTVEM MOVE.L D0,D5              ;(D5)=AF
000012E6  2C01                  397.            MOVE.L D1,D6            ;(D6)=BF
000012E8  4205                  398.            CLR.B D5
000012EA  4206                  399.            CLR.B D6                ;KILL EXPS
000012EC  E38D                  400.            LSL.L #1,D5
000012EE  E38E                  401.            LSL.L #1,D6             ;ALIGN
000012F0  7400                  402.            MOVEQ.L #0,D2           ;
000012F2  1400                  403.            MOVE.B D0,D2            ;CEXP=AEXP
000012F4  4A43                  404.            TST.W D3                ;TEST SC
000012F6  67 08                 405.            BEQ.S NOADJM            ;IF=0 NO SHFTNG
000012F8  6B00 0070             406.            BMI SHFTAM              ;NEG, SHFT A
000012FC  E6AE                  407.  SHFTBM  LSR.L D3,D6               ;SHIFTB
000012FE  4206                  408.            CLR.B D6
00001300  2600                  409.  NOADJM  MOVE.L D0,D3              ;COPY OF A
00001302  B383                  410.            EOR.L D1,D3             ;SIGNS
00001304  0803 001F             411.            BTST.L #SGNMSK,D3       ;
00001308  6700 0052             412.            BEQ SAMSGNM             ;EQUAL SIGNS?
0000130C  4486                  413.  DIFFSGM NEG.L D6                  ;NEG B
0000130E  DC85                  414.            ADD.L D5,D6             ;D6 HAS THE SUM
00001310  6500 0042             415.            BCS NCMPCM              ;IF CAR=0, NOT=0
00001314  0847 001F             416.  COMPCM  BCHG.L #SGNMSK,D7         ;COMPLEMENT SUM
00001318  4486                  417.            NEG.L D6
0000131A  0806 001F             418.  NORMCM  BTST.L #SGNMSK,D6         ;NORMALIZED?
0000131E  66 06                 419.            BNE.S CNORMDM
00001320  E38E                  420.            LSL.L #1,D6             ;ADJUST FRACTION
00001322  5342                  421.            SUBQ.W #1,D2            ;ADJUST EXP
00001324  60 F4                 422.            BRA.S NORMCM
                                423.  *
                                424.  * BOUND CHECK ON EXP
                                425.  *
00001326  4A42                  426.  CNORMDM TST.W D2
00001328  6F00 002E             427.            BLE CZEROM
0000132C  0C42 00FF             428.            CMPI.W #BGEXP,D2
00001330  6700 0042             429.            BEQ CINFM
                                430.  *
                                431.  * ROUND OFF FRACTION
                                432.  *
00001334  0806 0009             433.  ROUNDCM BTST.L #ALSB,D6
00001338  66 0A                 434.            BNE.S NOROUNM
0000133A  0806 0008             435.            BTST.L #ANTLSB,D6
0000133E  67 04                 436.            BEQ.S NOROUNM
00001340  0046 0200             437.            ORI.W #ARNDDR,D6
                                438.  *
                                439.  * ALIGN RESULT
                                440.  *
00001344  E28E                  441.  NOROUNM LSR.L #1,D6              ;ROOM FOR SIGN
00001346  4206                  442.            CLR.B D6                ;ROOM FOR EXP
                                443.  *
                                444.  * WITH BOUND CHECKS ON EXP, HAS LEADING ZEROES
                                445.  *
00001348  8486                  446.            OR.L D6,D2
0000134A  8487                  447.            OR.L D7,D2
0000134C  2002                  448.            MOVE.L D2,D0
0000134E  4CDF 00FC             449.  UNSTACM MOVEM.L (SP)+,D2-D7
00001352  4E75                  450.            RTS
                                451.  *
00001354  4A86                  452.  NCMPCM  TST.L D6                  ;C=0?
00001356  66 C2                 453.            BNE.S NORMCM            ;NO, NORMALIZE
```

```
00001358  4280           454.  CZEROM  CLR.L D0
0000135A  60 F2          455.          BRA UNSTACM
0000135C  DC85           456.  SAMSGNM ADD.L D5,D6              ;D6 = SUM FRACT
0000135E  64 D4          457.          BCC.S ROUNDCM
00001360  E296           458.          ROXR.L #1,D6             ;IF CARRY,
00001362  5242           459.          ADDQ.W #1,D2             ;ADJUST RESULT
00001364  60 C0          460.          BRA.S CNORMDM            ;GO ROUND
                         461.  *CNANM  MOVEQ.L #0,D0
                         462.  *       SUBQ.L #1,D0             ;C=NaN
                         463.  *       BRA UNSTACM
00001366  2001           464.  CISBM   MOVE.L D1,D0
00001368  60 E4          465.          BRA UNSTACM
0000136A  4443           466.  SHFTAM  NEG.W D3                 ;POS SHIFT CNT
0000136C  E6AD           467.          LSR.L D3,D5              ;ADJUST A FRACT
0000136E  4205           468.          CLR.B D5
00001370  3401           469.          MOVE.W D1,D2             ;USE BEXP=CEXP
00001372  60 8C          470.          BRA NOADJM
00001374  7000           471.  CINFM   MOVEQ.L #0,D0
00001376  103C 00FF      472.          MOVE.B #BGEXP,D0         ;C=inf
0000137A  8087           473.          OR.L D7,D0               ;TAKE SIGN
0000137C  60 D0          474.          BRA UNSTACM              ;QUIT
                         475.  *
                         476.  *
                         477.  **********************
                         478.  * SLAVE #0 PROGRAM
                         479.  **********************
                         480.  * DOES INVERT
                         481.  * 1,1/5,1/9,1/13,...1/(#TERMS*2-2)
                         482.  * SLAVE 1 DOES ADDITIONS
                         483.  * NOTE NO CALLS TO THE I/O TRAP
                         484.  * ROUTINES FOR TRANSFERS IN PIPES
                         485.  *
                         486.  * NUMBER OF LOOPS RECEIVED ON DATA
                         487.  * UPLOAD AND IS AT TCOUNT
                         488.  *
                         489.  * CHECK TO SEE THAT SLAVE #1
                         490.  * READY TO RECEIVE
                         491.  *
0000137E  207C 00012FC9  492.  SLVPR0  MOVEA.L #PBDR,A0
00001384  227C 00012FCD  493.          MOVEA.L #PPSR,A1
0000138A  7202           494.          MOVEQ.L #H3S,D1
0000138C  137C 0020 FFF3 495.          MOVE.B #ENABLEB,-13(A1)  ;EN OUTPUT PORT
00001392  0811 0006      496.  REDY0   BTST.B #H3LEV,(A1)       ;ADDER SLAVE READY?
00001396  66 FA          497.          BNE.S REDY0              ;WAIT TILL SO
00001398  3E3A FC66      498.          MOVE.W TCOUNT(PC),D7
                         499.  *
                         500.  * NOW START INVERT
                         501.  * GET NUMBER OF TERMS(/2-1 FOR DBRA)
                         502.  *
0000139C  7C01           503.          MOVEQ.L #1,D6    ;FIRST TERM=1
                         504.  *
                         505.  * INTEGER TO INVERT TO FP IS IN D6.W
                         506.  * RETURNS FP NUMBER IN D4.L
                         507.  * D6 NOT AFFECTED
                         508.  *
0000139E  343C 023A      509.  INVERT0 MOVE.W #DIVCNT,D2
000013A2  4E71           510.  DIVWAT0 NOP
000013A4  4E71           511.          NOP
000013A6  51CA FFFA      512.          DBRA D2,DIVWAT0
000013AA  45FA FC56      513.          LEA TEMPDAT(PC),A2
000013AE  7410           514.          MOVEQ.L #16,D2   ;D2 <- EXPONENT
000013B0  3A06           515.          MOVE.W D6,D5
000013B2  E25D           516.  SHFTLP0 ROL.W #1,D5      ;GET MSBIT=1 IN D5
000013B4  55CA FFFC      517.          DBCS D2,SHFTLP0
000013B8  E25D           518.          ROR.W #1,D5      ;ADJUST ONE BACK
000013BA  4402           519.          NEG.B D2         ;NEGATE EXPONENT
000013BC  D43C 0080      520.          ADD.B #BIAS,D2   ;BIAS
000013C0  7601           521.          MOVEQ.L #1,D3    ;FAST WAY FOR
000013C2  E29B           522.          ROR.L #1,D3      ;MOVE.L #$80000000,D3
000013C4  86C5           523.          DIVU.W D5,D3     ;DO MSW DIVIDE
000013C6  6900 0044      524.          BVS POWR20       ;IF OVRFLW, DIVISOR=2**k
000013CA  3803           525.          MOVE.W D3,D4
000013CC  4844           526.          SWAP.W D4        ;MSW OF QUO IN D4
000013CE  4243           527.          CLR.W D3         ;CLEAR LOW WORD D3
                         528.  *                        ;GIVES EXTENDED REMAINDER
000013D0  2603           529.          MOVE.L D3,D3     ;DUMMY MOVE FOR CCR Z
000013D2  67 14          530.          BEQ.S LASTEP0    ;REM=0? BRANCH, NO ROUND
000013D4  86C5           531.          DIVU.W D5,D3     ;GET LSW OF QUOTIENT
                         532.  *                        ;NEVER OVRFLW, IGNORE REM
```

```
000013D6  3803            533.          MOVE.W D3,D4    ;ALL BITS QUOTIENT -> D4
                          534.  *
                          535.  * round the result
                          536.  *
000013D8  0804 0009       537.  ROUNDQ0 BTST.L #ALSB,D4
000013DC  66 0A           538.          BNE.S LASTEP0
000013DE  0804 0008       539.          BTST.L #ANTLSB,D4
000013E2  67 04           540.          BEQ.S LASTEP0
000013E4  0044 0200       541.          ORI.W #ARNDDR,D4
000013E8  E28C            542.  LASTEP0 LSR.L #1,D4     ;SIGN=0
000013EA  5202            543.          ADDQ.B #1,D2
000013EC  1802            544.          MOVE.B D2,D4
                          545.  *
                          546.  * DONE  NVERT, STORE OFF
                          547.  *
000013EE  2484            548.  STORE0  MOVE.L D4,(A2)
000013F0  7003            549.          MOVEQ.L #NTFR,D0
000013F2  0311            550.  TSTLP0  BTST.B D1,(A1)
000013F4  67 FC           551.          BEQ.S TSTLP0
000013F6  109A            552.          MOVE.B (A2)+,(A0)
000013F8  51C8 FFF8       553.          DBRA D0,TSTLP0
                          554.  *
                          555.  * CALC'D AND SENT, DO NEXT TERM
                          556.  *
000013FC  5846            557.  NEXTRM0 ADDQ.W #4,D6
000013FE  51CF FF9E       558.          DBRA D7,INVERT0
                          559.  *
                          560.  * DONE, WAIT TILL LAST SENT, THEN DISABLE
                          561.  *
00001402  0311            562.  SENDL0  BTST.B D1,(A1)
00001404  67 FC           563.          BEQ.S SENDL0
00001406  4229 FFF3       564.          CLR.B -13(A1)
0000140A  4E75            565.          RTS
                          566.  *
                          567.  * OVERFLOWED, POWER OF 2 DIVISOR, QUO=DIVISOR*2
                          568.  * NO REMAINDER, NO ROUNDING REQUIRED
                          569.  *
0000140C  4845            570.  POWR20  SWAP D5          ;MSW OF QUO
0000140E  4245            571.          CLR.W D5        ;NO MORE SIG DIGITS
00001410  E28D            572.          LSR.L #1 D5     ;SIGN=0
00001412  5402            573.          ADDQ.B #2,D2    ;ADJUST EXPONENT
00001414  2805            574.          MOVE.L D5,D4
00001416  1802            575.          MOVE.B D2,D4
00001418  60 D4           576.          BRA STORE0
                          577.  *
                          578.  ****************
                          579.  * DONE SLAVE 0 *
                          580.  ****************
                          581.  *
                          582.  ********************************
                          583.  * SLAVE 1 AND SLAVE 3 PROGRAM *
                          584.  ********************************
                          585.  *
                          586.  * RECIEVES DATA FROM PREVIOUS SLAVE.
                          587.  * AND DOES REMANDER OF INVERT PHASE IF
                          588.  * NECESSARY. THEN DOES ADD FOR REQUIRED NUMBER
                          589.  * OF TERMS, THEN UPLOADS RESULT TO MASTER
                          590.  * NOTE NO SRQ MADE, JUST AWAITS MASTER'S
                          591.  * READY TO RECEIVE SIGNAL BEFORE SENDING
                          592.  *
                          593.  * SLAVE 1 DOES OPS ON TERMS FROM SLAVE 0
                          594.  * THE POSITVE SERIES, AND SLAVE 3 OPS ON
                          595.  * TERMS FROM SLAVE 2, THE NEGATIVE SERIES
                          596.  * THEN MASTER DOES (SLV1)-(SLV3)FOR PI/4
                          597.  *
                          598.  * FIRST THING TO DO IS ENABLE INPUT PORT,
                          599.  * AND AWAIT INPUT DATA
                          600.  *
                          601.  * LOOP COUNTER LOADED FROM MASTER AND IS AT
                          602.  * TCOUNT
                          603.  *********************************************
                          604.  *
0000141A  267C 00012FC0   605.  SLVPR13 MOVEA.L #PGCR,A3
00001420  227C 00012FCD   606.          MOVEA.L #PPSR,A1
00001426  207C 00012FC8   607.          MOVEA.L #PADR,A0
0000142C  16BC 0010       608.          MOVE.B #ENABLEA,(A3)
00001430  7400            609.          MOVEQ.L #H1S,D2
00001432  4280            610.          CLR.L D0        ;FIRST ADD TO 0
00001434  3E3A FBCA       611.          MOVE.W TCOUNT(PC),D7
00001438  3C3C 0234       612.  NEXTM13 MOVE.W #ADDCNT,D6
0000143C  4E71            613.  ADDWAT  NOP
```

```
0000143E  4E71           614.          NOP
00001440  51CE FFFA      615.          DBRA D6,ADDWAT
00001444  7603           616.          MOVEQ.L #NTFR,D3
00001446  45FA FBBA      617.          LEA TEMPDAT(PC),A2
0000144A  0511           618.  REDY13  BTST.B D2,(A1)
0000144C  67 FC          619.          BEQ.S REDY13
0000144E  14D0           620.          MOVE.B (A0),(A2)+
00001450  51CB FFF8      621.          DBRA D3,REDY13
00001454  223A FBAC      622.          MOVE.L TEMPDAT(PC),D1
00001458  48E7 2100      623.  ADDFP13 MOVEM.L D2/D7,-(SP)      ;SAVE OFF
                         624.  *       CMPI.B #BGEXP,D0        ;A=NaN OR inf?
                         625.          BEQ CNAN13              ;C=NaN IF SO
                         626.  *       CMPI.B #BGEXP,D1        ;B=NaN OR inf?
                         627.  *       BEQ CNAN13              ;C=NaN IF SO
0000145C  4A81           628.          TST.L D1                ;B=0?
0000145E  6700 0098      629.          BEQ UNSTA13             ;C=A IF SO
00001462  4A80           630.          TST.L D0                ;A=0?
00001464  6700 00AC      631.          BEQ CISB13              ;C=B IF SO
00001468  4243           632.          CLR.W D3
0000146A  4244           633.          CLR.W D4
0000146C  1600           634.          MOVE.B D0,D3            ;(D3.W)=EXPA
0000146E  1801           635.          MOVE.B D1,D4            ;(D4.W)=EXPB
00001470  9644           636.          SUB.W D4,D3             ;(D3.W)=SHFTCNT
00001472  0C43 0017      637.          CMPI.W #SIGBITS,D3      ;TOO BIG? C=A
00001476  6C00 0080      638.          BGE UNSTA13
0000147A  0C43 FFE9      639.          CMPI.W #-SIGBITS,D3     ;TOO SMALL? C=B
0000147E  6F00 0092      640.          BLE CISB13
                         641.  *
                         642.  * HERE, MUST DO ADD
                         643.  * D5, D6 GET FRACTIONS; D7 CARRIES SIGN
                         644.  *
00001482  7E00           645.          MOVEQ.L #0,D7
00001484  0800 001F      646.          BTST.L #SGNMSK,D0       ;SIGN OF A
00001488  67 04          647.          BEQ.S POST13
0000148A  08C7 001F      648.          BSET.L #SGNMSK,D7       ;C IS NEG
0000148E  2A00           649.  POST13  MOVE.L D0,D5            ;(D5)=AF
00001490  2C01           650.          MOVE.L D1,D6            ;(D6)=BF
00001492  4205           651.          CLR.B D5
00001494  4206           652.          CLR.B D6                ;KILL EXPS
00001496  E38D           653.          LSL.L #1,D5
00001498  E38E           654.          LSL.L #1,D6             ;ALIGN
0000149A  7400           655.          MOVEQ.L #0,D2           ;
0000149C  1400           656.          MOVE.B D0,D2            ;CEXP=AEXP
0000149E  4A43           657.          TST.W D3                ;TEST SC
000014A0  67 08          658.          BEQ.S NOADJ13           ;IF=0 NO SHFTNG
000014A2  6B00 0072      659.          BMI SHFTA13             ;NEG, SHFT A
000014A6  E6AE           660.  SHFTB13 LSR.L D3,D6             ;SHIFTB
000014A8  4206           661.          CLR.B D6
000014AA  2600           662.  NOADJ13 MOVE.L D0,D3            ;COPY OF A
000014AC  B383           663.          EOR.L D1,D3             ;SIGNS
000014AE  0803 001F      664.          BTST.L #SGNMSK,D3       ;
000014B2  6700 0054      665.          BEQ SAMSG13             ;EQUAL SIGNS?
000014B6  4486           666.  DIFFS13 NEG.L D6                ;NEG B
000014B8  DC85           667.          ADD.L D5,D6             ;D6 HAS THE SUM
000014BA  6500 0044      668.          BCS NCMPC13             ;IF CAR=0, NOT=0
000014BE  0847 001F      669.  COMPC13 BCHG.L #SGNMSK,D7       ;COMPLEMENT SUM
000014C2  4486           670.          NEG.L D6
000014C4  0806 001F      671.  NORMC13 BTST.L #SGNMSK,D6       ;NORMALIZED?
000014C8  66 06          672.          BNE.S CNORM13
000014CA  E38E           673.          LSL.L #1,D6             ;ADJUST FRACTION
000014CC  5342           674.          SUBQ.W #1,D2            ;ADJUST EXP
000014CE  60 F4          675.          BRA.S NORMC13
                         676.  *
                         677.  * BOUND CHECK ON EXP
                         678.  *
000014D0  4A42           679.  CNORM13 TST.W D2
000014D2  6F00 0030      680.          BLE CZERO13
000014D6  0C42 00FF      681.          CMPI.W #BGEXP,D2
000014DA  6700 0044      682.          BEQ CINF13
                         683.  *
                         684.  * ROUND OFF FRACTION
                         685.  *
000014DE  0806 0009      686.  ROUND13 BTST.L #ALSB,D6
000014E2  66 0A          687.          BNE.S NORON13
000014E4  0806 0008      688.          BTST.L #ANTLSB,D6
000014E8  67 04          689.          BEQ.S NORON13
000014EA  0046 0200      690.          ORI.W #ARNDDR,D6
                         691.  *
                         692.  * ALIGN RESULT
                         693.  *
000014EE  E28E           694.  NORON13 LSR.L #1,D6              ;ROOM FOR SIGN
```

```
000014F0  4206              695.          CLR.B D6                  ;ROOM FOR EXP
                            696.  *
                            697.  * WITH BOUND CHECKS ON EXP, HAS LEADING ZEROES
                            698.  *
000014F2  8486              699.          OR.L D6,D2
000014F4  8487              700.          OR.L D7,D2
000014F6  2002              701.          MOVE.L D2,D0
000014F8  4CDF 0084         702.  UNSTA13 MOVEM.L (SP)+,D2/D7
000014FC  6000 002C         703.          BRA DUNAD13
                            704.  *
00001500  4A86              705.  NCMPC13 TST.L D6                  ;C=0?
00001502  66 C0             706.          BNE.S NORMC13             ;NO, NORMALIZE
00001504  4280              707.  CZERO13 CLR.L D0
00001506  60 F0             708.          BRA UNSTA13
00001508  DC85              709.  SAMSG13 ADD.L D5,D6                ;D6 = SUM FRACT
0000150A  64 D2             710.          BCC.S ROUND13
0000150C  E296              711.          ROXR.L #1,D6              ;IF CARRY,
0000150E  5242              712.          ADDQ.W #1,D2              ;ADJUST RESULT
00001510  60 BE             713.          BRA.S CNORM13             ;GO ROUND
                            714.  *CNAN13 MOVEQ.L #0,D0
                            715.  *       SUBQ.L #1,D0              ;C=NaN
                            716.  *       BRA UNSTA13
00001512  2001              717.  CISB13  MOVE.L D1,D0
00001514  60 E2             718.          BRA UNSTA13
00001516  4443              719.  SHFTA13 NEG.W D3                  ;POS SHIFT CNT
00001518  E6AD              720.          LSR.L D3,D5               ;ADJUST A FRACT
0000151A  4205              721.          CLR.B D5
0000151C  3401              722.          MOVE.W D1,D2              ;USE BEXP=CEXP
0000151E  60 8A             723.          BRA NOADJ13
00001520  7000              724.  CINF13  MOVEQ.L #0,D0
00001522  103C 00FF         725.          MOVE.B #BGEXP,D0          ;C=inf
00001526  8087              726.          OR.L D7,D0                ;TAKE SIGN
00001528  60 CE             727.          BRA UNSTA13               ;QUIT
0000152A  51CF FF0C         728.  DUNAD13 DBRA D7,NEXTM13
0000152E  4213              729.          CLR.B (A3)
00001530  41FA FAD0         730.          LEA TEMPDAT(PC),A0
00001534  2080              731.          MOVE.L D0,(A0)
00001536  4E43              732.          TRAP #SRQASRT
00001538  7004              733.          MOVEQ.L #NTFR+1,D0
0000153A  16BC 0020         734.          MOVE.B #ENABLEB,(A3)
0000153E  4E46              735.          TRAP #OUTDATA
00001540  4213              736.          CLR.B (A3)
00001542  4E75              737.          RTS
                            738.  *
                            739.  ************************************
                            740.  * SLAVES 1 AND 3 PROGS DONE        *
                            741.  ************************************
                            742.  *
                            743.  **********************
                            744.  * SLAVE #2 PROGRAM
                            745.  **********************
                            746.  * DOES PARTIAL
                            747.  * 1/3,1/7,1/11,1/15...1/(#TERMS*2-1)
                            748.  * SLAVE #3 DOES ADDS
                            749.  * NOTE NO CALLS TO THE I/O TRAP
                            750.  * ROUTINES FOR TRANSFERS IN PIPES
                            751.  *
                            752.  * NUMBER OF LOOPS RECEIVED ON DATA
                            753.  * UPLOAD AND IS AT TCOUNT
                            754.  *
                            755.  * CHECK TO SEE THAT SLAVE #3
                            756.  * READY TO RECEIVE
                            757.  * USE A4 TO POINT TO PORT STATUS REG
                            758.  * AND A5 TO POINT TO PORT B DATA REG
                            759.  *
                            760.  *
00001544  207C 00012FC9     761.  SLVPR2  MOVEA.L #PBDR,A0
0000154A  227C 00012FCD     762.          MOVEA.L #PPSR,A1
00001550  7202              763.          MOVEQ.L #H3S,D1
00001552  137C 0020 FFF3    764.          MOVE.B #ENABLEB,-13(A1)   ;EN OUTPUT PORT
00001558  0811 0006         765.  REDY2   BTST.B #H3LEV,(A1)        ;ADDER SLAVE READY?
0000155C  66 FA             766.          BNE.S REDY2               ;WAIT TILL SO
0000155E  3E3A FAA0         767.          MOVE.W TCOUNT(PC),D7
                            768.  *
                            769   * NOW START INVERT
                            770.  * GET NUMBER OF TERMS(/2-1 FOR DBRA)
                            771.  *
00001562  7C03              772.          MOVEQ.L #3,D6    ;FIRST TERM=3
                            773.  *
                            774.  * INTEGER TO INVERT TO FP IS IN D6.W
                            775.  * RETURNS FP NUMBER IN D6.L
```

```
                                776.  * D6 NOT AFFECTED
                                777.  *
00001564  343C 023A             778.  INVERT2 MOVE.W #DIVCNT,D2
00001568  4E71                  779.  DIVWAT2 NOP
0000156A  4E71                  780.          NOP
0000156C  51CA FFFA             781.          DBRA D2,DIVWAT2
00001570  45FA FA90             782.          LEA TEMPDAT(PC),A2
00001574  7410                  783.          MOVEQ.L #16,D2  ;D2 <- EXPONENT
00001576  3A06                  784.          MOVE.W D6,D5
00001578  E35D                  785.  SHFTLP2 ROL.W #1,D5      ;GET MSBIT=1 IN D5
0000157A  55CA FFFC             786.          DBCS D2,SHFTLP2
0000157E  E25D                  787.          ROR.W #1,D5      ;ADJUST ONE BACK
00001580  4402                  788.          NEG.B D2         ;NEGATE EXPONENT
00001582  043C 0080             789.          ADD.B #BIAS,D2   ;BIAS
00001586  7601                  790.          MOVEQ.L #1,D3    ;FAST WAY FOR
00001588  E29B                  791.          ROR.L #1,D3      ;MOVE.L #$80000000,D3
0000158A  86C5                  792.          DIVU.W D5,D3     ;DO MSW DIVIDE
0000158C  6900 0044             793.          BVS POWR22       ;IF OVRFLW, DIVISOR=2**k
00001590  3803                  794.          MOVE.W D3,D4
00001592  4844                  795.          SWAP.W D4        ;MSW OF QUO IN D4
00001594  4243                  796.          CLR.W D3         ;CLEAR LOW WORD D3
                                797.  *                        ;GIVES EXTENDED REMAINDER
00001596  2603                  798.          MOVE.L D3,D3     ;DUMMY MOVE FOR CCR Z
00001598  67 14                 799.          BEQ.S LASTEP2    ;REM=0? BRANCH, NO ROUND
0000159A  86C5                  800.          DIVU.W D5,D3     ;GET LSW OF QUOTIENT
                                801.  *                        ;NEVER OVRFLW, IGNORE REM
0000159C  3803                  802.          MOVE.W D3,D4     ;ALL BITS QUOTIENT -> D4
                                803.  *
                                804.  * round the result
                                805.  *
0000159E  0804 0009             806.  ROUNDQ2 BTST.L #ALSB,D4
000015A2  66 0A                 807.          BNE.S LASTEP2
000015A4  0804 0008             808.          BTST.L #ANTLSB,D4
000015A8  67 04                 809.          BEQ.S LASTEP2
000015AA  0044 0200             810.          ORI.W #ARNDDR,D4
000015AE  E28C                  811.  LASTEP2 LSR.L #1,D4       ,SIGN=0
000015B0  5202                  812.          ADDQ.B #1,D2
000015B2  1802                  813.          MOVE.B D2,D4
                                814.  *
                                815.  * DONE INVERT, STORE OFF
                                816.  *
000015B4  2484                  817.  STORE2  MOVE.L D4,(A2)
000015B6  7003                  818.          MOVEQ.L #NTFR,D0
000015B8  0311                  819.  TSTLP2  BTST.B D1,(A1)
000015BA  67 FC                 820.          BEQ.S TSTLP2
000015BC  109A                  821.          MOVE.B (A2)+,(A0)
000015BE  51C8 FFF8             822.          DBRA D0,TSTLP2
                                823.  *
                                824.  * CALC'D AND SENT, DO NEXT TERM
                                825.  *
000015C2  5846                  826.  NEXTRM2 ADDQ.W #4,D6
000015C4  51CF FF9E             827.          DBRA D7,INVERT2
                                828.  *
                                829.  * D? ', WA T TILL LAST SENT, THEN DISABLE
                                830.  *
000015C8  0311                  831.  SENDL2  BTST.B D1,(A1)
000015CA  67 FC                 832.          BEQ.S SENDL2
000015CC  4229 FFF3             833.          CLR.B -13(A1)
000015D0  4E75                  834.          RTS
                                835.  *
                                836.  * OVERFLOWED, POWER OF 2 DIVISOR, QUO=DIVISOR*2
                                837.  * NO REMAINDER, NO ROUNDING REQUIRED
                                838.  *
000015D2  4845                  839.  POWR22  SWAP D5          ;MSW OF QUO
000015D4  4245                  840.          CLR.W D5         ;NO MORE SIG DIGITS
000015D6  E28D                  841.          LSR.L #1,D5      ;SIGN=0
000015D8  5402                  842.          ADDQ.B #2,D2     ;ADJUST EXPONENT
000015DA  2805                  843.          MOVE.L D5,D4
000015DC  1802                  844.          MOVE.B D2,D4
000015DE  60 D4                 845.          BRA STORE2
                                846.  *
                                847.  ******************
                                848.  * DONE SLAVE 2  *
                                849.  ******************
000015E0  <2>                   850.  DONE2   DS W 1
000015E2                        851.          END
    0 Errors
```

Appendix O: Two-Dimensional Fast Fourier Transforms

The performance tests conducted in the study utilized parallel and parallel-pipelined configurations. A test using a single, serial configuration was not conducted. A suitable application which is related to the other tests is calculation of two-dimensional fast Fourier transforms.

The two-dimensional Fourier transform is required in applications which involve two-dimensional data sets, such as image processing, and geophysical analyses (Kanasewich 81) (Dudgeon 2, 84). The two-dimensional DFT is defined by:

$$F\{g(x,y)\} = G(u,v) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} g(x,y) e^{-j2\pi(ux/M + vy/N)} \tag{O.1}$$

It is common to use a square input array, so that $M = N$, yielding:

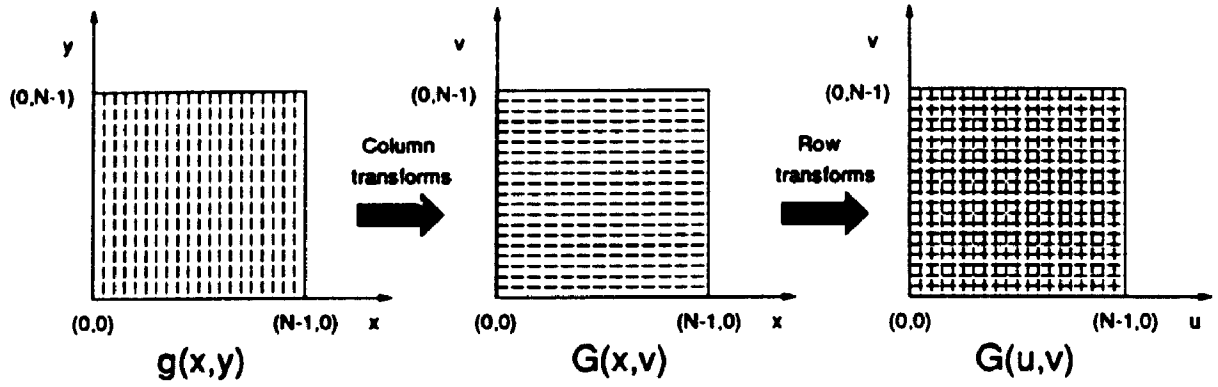
$$G(u,v) = \frac{1}{N^2} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} g(x,y) e^{-j2\pi(ux + vy)/N} \tag{O.2}$$

The equation has the property of separability, and may be alternately expressed as:

$$G(u,v) = \frac{1}{N^2} \sum_{x=0}^{N-1} e^{-j2\pi ux/N} \sum_{y=0}^{N-1} g(x,y) e^{-j2\pi vy/N} \tag{O.3}$$

The separability permits calculation of the 2-d DFT as a sequence of 1-d DFT's, and therefore the 1-d FFT algorithm may be used repeatedly. The inner summation is a set of 1-d transforms of the columns of g(x,y), yielding the function G(x,v). G(u,v) is then calculated as a set of 1-d transforms of the rows of G(x,v). Computation of an $N \times N$-point 2-d Fourier transform requires $2N$ complete, 1-d FFT calculations. The process is depicted in figure O.1 (Gonzalez 2, 87).

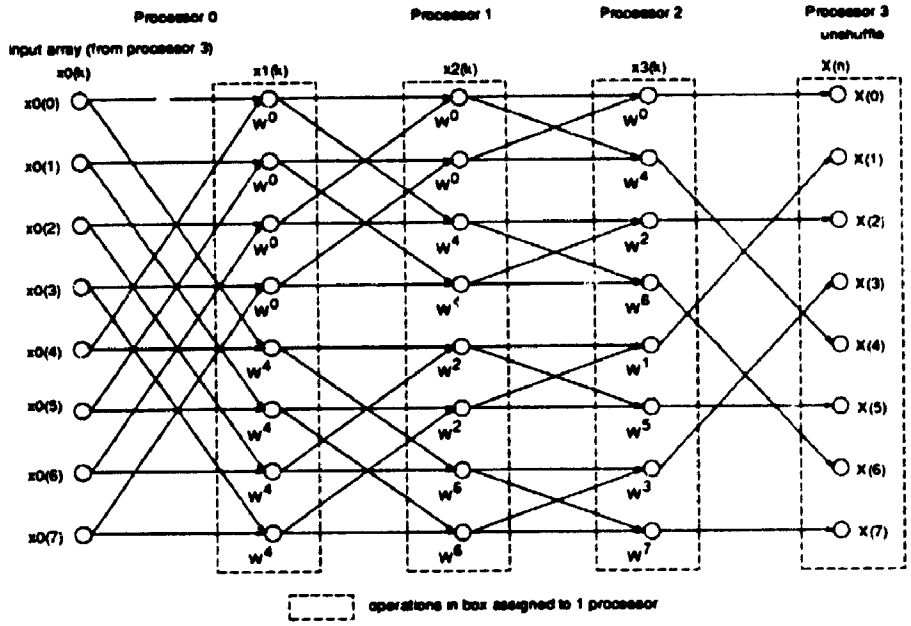
Figure O.1 2-d Discrete Fourier transforms using a sequence of 1-d transforms.



Each of the 1-d transforms could be performed by the parallel array of processors exactly as in the parallel FFT performance tests. However, the data broadcasting, collection, and reordering phases for each transform would not utilize the slave processors effectively throughout the process. A solution which computes transforms in a serial or pipelined fashion provides better resource utilization, and therefore potentially higher performance. Figure O.2 is an 8-point, radix-2 Cooley-Tukey FFT signal flow graph showing task division among four series-connected processing elements. Processor #3 is assigned the tasks of providing the column data to processor #0, as well as collecting the transformed column data from processor #2, and sorting it into natural order. After all column transforms have been computed, the new row data is transformed in a similar fashion. Each processor in the pipeline calculates a vector in the FFT calculation, and transmits its results to the next processor. For different problem sizes, the number of vectors calculated per processor must be adjusted. As with any pipeline, maximum performance is achieved when execution times of all stages are equalized. Data dependencies of execution times for each stage can reduce performance, as observed in the alternating series tests. In the case of an $2^p \times 2^p$-point transform, each processor performs 2^{p-1} multiplications and 2^p additions. The first processor performs multiplications of $W^0 = 1$ and $W^{2^p/2} = -1$, and they may be replaced by simpler and faster sign changes. The task may be assigned to processor #3, since the

operation of data reordering can be performed relatively quickly. The element has sufficient time to compute the first vector, and keep the pipeline filled as necessary. A five processor pipeline could perform a 32×32-point Fourier transform: each processor calculates a single vector in the 1-d FFT, and one of those processors also handles data unshuffling. If a 1024×1024-point transform is required, each processor may be assigned to calculate two vectors each. Problem dimension and pipeline depth must be matched to achieve optimum performance.

Figure Q.2 2-d Fast Fourier transform calculation using pipelined system.



Analysis of maximum theoretical performance based on number of multiplications can be outlined. The uniprocessor system must perform $2^p \times 2^{p-1}p = p2^{2p-1}$ multiplications to calculate a $2^p \times 2^p$-point transform. If a time unit, τ, is defined as the time required to perform a series of 2^{p-1} multiplications (time to calculate a complete vector), then the uniprocessor requires $2^{p+1}p\tau$ time to compute the entire 2-d transform.

In the pipelined system, each element performs 2^{p-1} multiplications. Assuming a pipeline p processors deep, the time to complete the first set of 2^p column transforms is the time to fill the pipeline, plus the time to produce 2^p results (section 1.3.2). The row transforms must then be calculated, which requires an equal amount of time. Thus:

$$time_{\text{2-d trans}} = time_{\text{row trans}} + time_{\text{col trans}} = 2(p - 1 + 2^p)\tau \tag{O.4}$$

The maximum speed-up factor of the uniprocessor and pipelined implementations is given by:

$$speed\text{-}up = \frac{2^{p+1}p}{2(p - 1 + 2^p)} \tag{O.5}$$

In the case of $p = 5$, (32×32-point transform, five processing elements), the speed-up provided by the pipelined system is 4.44. If the same pipeline is used to calculate a 1024×1024-point transform (each element calculates two vectors), the speed-up factor is 4.98. Calculation of 2-d FFTs may also be performed in a parallel mode, where each processor is assigned $1/r$ of the column transforms, and $1/r$ of the row transforms, where r is the number of processors available. The two methods would compete in terms of performance, with communications time and memory requirements the decisive factors in determining superiority.

References

Advanced Micro Devices, *AM29000 User's Manual 32-bit Streamlined Instruction Processor*, AMD, Sunnyvale Cal., 1990.

Agrawal, D.P., Janakiram, V.K., *Evaluating the Performance of Multicomputer Configurations*, Computer vol.22, No.5, May 1986, pp.23-37

Athas, W.C., Seitz, C.L., *Multicomputers: Message-Passing Concurrent Computers*, Computer, vol. No.8, Aug.1988, pp.9-24

Averbuch, A., Gabber, E., Gordissky, B., Medan, Y., *A Parallel FFT on a MIMD Machine*, Parallel Computing 15(1990), 61-74, North Holland.

Balasubramanian, V., Banerjee, P., *A Fault Tolerant Massively Parallel Processing Architecture*, Journal of Parallel and Distributed Computing, 4, 363-383, (1987).

Belkhale, K.P., Banerjee, P., *Reconfiguration Strategies for VLSI Processor Arrays and Trees Using a Modified Diogenes Approach*, IEEE Transactions on Computers, Vol.41, No.1, Jan.1992, pp83-96.

Bergland, G.D., Wilson, D.E., *A Fast Fourier Transform Algorithm fo a Global, Highly Parallel Processor*, IEEE Transactions on Audio and Electroacoustics, Vol.Au-17, No.2, June 1969, pp.125-127.

Bi, G., Jones, E.V., *A Pipelined FFT Processor for Word Sequential Data*, IEEE Transactions on Acoustics, Speech and Signal Processing, Vol.37, No.12, Dec.1989, pp.1982-1985.

Boubekeur, A., Patry, J.L., Saucier, G., Trilhe, J., *Configuring a Wafer-Scale, Two-Dimensional Array of Single-Bit Processors*, Computer, Vol.28, No.4, April, 1992. pp.29-39.

Brigham, E.O., *The Fast Fourier Transform*, Prentice-Hall Inc., Englewood Cliffs, N.J., 1974, ch.10.3

Burks, A.W, Goldstine, H.H, Von Neumann, J. *Preliminary discussion of the Logical Design of an Electronic Computing Instrument*, 1946, Report for U.S. Army Ord. dept, reprinted in Papers of John Von Neumann on Computing and Computer Theory; Charles Babbage Institute reprint series for the History of Computing, M. Campbell-Kelly, ed., MIT Press, Cambridge Mass., 1987.

Editorial Staff, *High-Tech Horsepower*, Byte Magazine, McGraw-Hill Inc., Vol.12, No.8, July 1987, pp.101-108.

Cavanaugh, J.J.F., *Digital Computer Arithmetic: Design and Implementation*, McGraw-Hill Book Company, New York, New York, 1984, sections 6.2, 6.3.

Charlesworth, A.E., Gustafson, J.L., *Introducing Replicated VLSI to Supercomputing: the FPS-164/MAX Scientific Computer*, Computer, vol.19, No.3, March 1986, pp.10-23

Chean, M., Fortes, J.A.B., *A Taxonomy of Reconfiguration Techniques for Fault-Tolerant Processor Arrays*, Computer, vol.26, No.1, Jan.1990, pp.55-69

Chen, P.Y., Lawrie, H., Yew, P.C., Padua, A., *Interconnection Networks Using Shuffles*, Computer, vol.14, No.12, pp55-64, Dec.1981.

Choi, Y.H., *Reconfigurable VLSI/WSI Multipipelines, Parallel Computing*, 17 (1991) pp.941-952, North Holland.

Chum, K.W.K, *Electrostatic Damage in Microcircuits and Design of Integrated Protection Networks*, M.E.Sc Thesis, Faculty of Graduate Studies, The University of Western Ontario, London, Ontario, 1989.

Canadian Microelectronics Corporation, *Guide to the Integrated Circuit Implementation Services of the Canadian Microelectronics Corporation*, Canadian Microelectronics Corp., Kingston, Ontario, 1989.

Cornejo, C., Lee, R., *Comparing IBM's Microchannel and Apple's Nubus*, Byte Magazine, McGraw-Hill publishing, Peterborough, N.H., Dec. 1987.

Cvetanovic, Z., 1: *Best and Worst Case Mappings for the Omega Network*, IBM Journal of Research and Development, vol.31, No.4, July 1987, pp.452-463:
2: *Performance Analysis of the FFT Algorithm on a Shared-Memory Parallel Architecture*, IBM Journal of Research and Development, vol.31, No.4, July 1987, pp.435-451.

Del Corso,D. Kirrman, H., Nicoud, J.D., *Microcomputer Buses and Links*, Academic Press Inc., Orlando Florida, 1986

Dexter, A.L., *Microcomputer Bus Structures and Bus Interface Design*, Marcell Dekker inc. New York, 1986

Dudgeon, D.E., Mersereau, R.M., *Multidimensional Digital Signal Processing*, Prentice-Hall Inc., Englewood Cliffs, N.J., 1,2:ch.3.

Duncan, R., *A Survey of Parallel Computer Architectures*, Computer, vol.24, No.2, Feb.1988, pp.5-16

Flynn, M.J., *Some Computer Organizations and Their Effectiveness*, IEEE Transactions on Computers, vol.c-21, No.9, Sept.1972, pp.948-960

Gajski, D.D., Peir, J., *Essential Issues in Multiprocessor Systems*, Computer, vol.18, No.6, June 1985, pp.9-27

Gonzalez, R.C., Wintz, P., *Digital Image Processing*, Addison-Wesley, Reading Mass., 1987, 1:ch.4, 2:ch.3.3.1.

Goodman, J.R., Sequin, C.H., *Hypertree: A multiprocessor Interconnection Topology*, IEEE Transactions on Computers, vol.C-30, No.12, Dec.1981, pp.923-933.

Grehan, R., Thompson, T., Franklin, C., Stewart, G., *Introducing the New BYTE Benchmarks*, Byte Magazine, McGraw-Hill Inc., Vol.13, Number 6, June 1988, pp.239-266.

Haynes, L.S., Lau, R.L., Siewiorek, D.P., Mizell, D.W., *A Survey of Highly Parallel Computing*, Computer, vol.15, No.1, Jan.1982, pp.9-24.

Hinnant, D.F., *Accurate UNIX Benchmarking: Art, Science, or Black Magic?*, IEEE Micro, Vol.8 No.5, October 1988, pp.64-75.

Hodges, A., *Alan Turing: The Enigma*, Simon and Schuster, New York, New York, 1983, ch. 5

Hon, R., Sequin, C.H., *A Guide to LSI Implementation*, second edition, Palo Alto Research Center, Xerox Corp., 1980.

Hoshino,T., *An Invitation to the World of PAX*, Computer, vol.19, No.5, May 1986, pp.68-79

Huang, Y., Paker, Y., *A Parallel FFT Algorithm for Transputer Networks*, Parallel Computing, 17(1991) 895-906, North Holland.

Hwang, K., Xu, Z., *Multipipeline Networking for Compound Vector Processing*, IEEE Transactions on Computers, vol.37, No.1, Jan., 1988, pp.33-47.

Hwang, K., Briggs, F.A. *Computer Architecture and Parallel Processing*, McGraw-Hill, New York, 1984, 1:ch.8.4.1, 2:ch.3, 3:ch.4

Hwang, K.,Tseng, P.S., 1: *An Orthogonal Multiprocessor for Parallel Scientific Computing*, IEEE Transactions on Computers, vol.38, No.1, Jan. 1989 pp.47-61, 2:Hwang, K., DeGroot,D. eds. *Parallel Processing for Super Computers and Artificial Intelligence*, McGraw-Hill, New York, 1989, ch.2.

i80386:Intel Corp., *386 Microprocessor High Performance 32-bit CHMOS Microprocessors with Integrated Memory Management*, Intel Corp., Santa Clara, Cal., 1989.

Intel Corp., *iAPX High Performance Benchmark Report*, Intel Corp, Santa Clara, California, 1985.

Jagadish, N., Kumar, J.M., Patnaik, L.M., *An Efficient Scheme for Interprocessor Communications Using Dual-Ported RAMS*, IEEE Micro, Oct. 1989 pp.10-19.

Kanasewich, E.R., *Time Domain Analysis in Geophysics*, University of Alberta Press, 1981, ch.3.7.

Kirk, D.E., Verly, J.G., *An Algorithm for Distributed Computation of FFT's*, Computers and Electrical Engineering, Vol.13, No.2, 1987, pp.83-96.

Kung, H.T., *Why Systolic Architectures?*, Computer, vol.15, No.1, Jan.1987, pp.37-45

Kung, S.Y., *VLSI Array Processors*, Prentice Hall, Englewood Cliffs, New Jersey, 1988, 1:ch.2.6, 2:ch.5.2, 3:ch 6.4

Lang, T., *Interconnection between Processors and Memory using the Shuffle-Exchange Network*, IEEE Transactions on Computers, vol.C-25, No.5, May, 1976, pp.496-503

Lawrie, D.H., *Access and Alignment of Data in an Array Processor*, IEEE Transactions on Computers, vol.C-24, No.12, December, 1975, pp.175-189

Lea, R.M, Jalowiecki, I.P., *Associative Massively Parallel Computers*, Proceedings of the IEEE, vol.79, No.4, Apr.1991, pp.469-479

Li,H., Stout, Q.F., *Reconfigurable SIMD Massively Parallel Computers*, Proceeeding of the IEEE, vol.79, No.4, April, 1991

M68020, 87:Motorola Inc, *MC68020 32-bit Virtual Memory Microprocessor* Technical Summary, Motorola Inc., Phoenix Arizona, 1987.

M88100, 88:Motorola Inc., *MC88100 32-Bit Third Generation RISC Microprocessor Technical Summary*, Motorola Inc., Phoenix, Arizona, 1988.

MC68008, 85:Motorola Semiconductor Inc., *MC68008 8/32-bit Microprocessor with 8-bit Data Bus, Advance Information*, Document ADI939R2, Motorola Semiconductor Inc., Austin, Texas, 1985.

MC68230, 83:Motorola Semiconductor Inc., *MC68230 Parallel Interface/Timer Advance Information*, Document ADI-860-R2, Motorola Semiconductor Inc., Austin, Texas, 1983.

Mead, C., Conway, L. *Introduction to VLSI Systems*, Addison Wesley publishing company, Reading, Mass, 1:p.250, 2:p.254

Motorola Inc., *Performance Report 68020 68030 32-BIt Microprocessors*, Motorola Inc., Phoenix Arizona, Document BR705/D, 1988

Motorola 1, 86:Motorola Inc., *M68000 vs. iAPX86 Benchmark Performance*, Motorola Inc., Phoenix, Arizona, Document BR150, 1986.

Motorola 2, 86:Motorola Inc., *Motorola MC68020 Benchmark Report*, Motorola Inc., Phoenix, Arizona, 1986.

Padmanabhan, K., Lawrie, D.H., *A Class of Redundant Path Multistage Interconnection Networks*, IEEE Transactions on Computers, vol.C-32, No.12, Dec.1983, pp.1099-1108

Patterson, D.A. *Reduced Instruction Set Computers* Communications of the ACM, vol.28, Jan. 1985, pp.8-21

Pease, M.C., *An Adaptation of the Fast Fourier Transform for Parallel Processing*, Journal of the Association for Computing Machinery, Vol.15, No.2, April, 1968 pp.252-264

Price, W.J., *A Benchmark Tutorial*, IEEE Micro, Vol.9, No.5, Oct.89, pp.28-43.

Quinn, M.J., *Designing Efficient Algorithms for Parallel Computers*, McGraw-Hill, New York, 1987, 1:ch.3-3, 2:p.19, 3:ch.4, 4:ch.6

Reed, D.A., Grunwald, D.C., *The Performance of Multicomputer interconnection Networks*, Computer vol.23, No.6, June 1986, pp.63-73.

Reed, D.A., Fujimoto, R.M., *Multicomputer Networks: Message Based Parallel Processing*, The MIT Press, Cambridge, Mass., 1987, ch.1

Sami, M., Stefanelli, R., *Reconfigurable Architectures for VLSI Processing Arrays*, Proceedings of the IEEE, vol.74, No.5, May 1986.

Sawchuck, A.A., Jenkins, B.K., Raghavendra, C.S., *Optical Crossbar Networks*, Computer, vol.20, No.6, June,1987, pp.50-60

Sietz, C.L., *The Cosmic Cube*, Communications of the ACM, vol.28, No.1, Jan.1985, pp.22-33

Shockley, J.E., *Calculus and Analytic Geometry*, Saunder College Press, Philadelphia, Penn., 1982, ch.12.5.

Shurkin, J., *Engines of the Mind. A History of the Computer*, W.W. Norton and Company, New York, 1984. p.251

Seigel, H.J., Armstrong, J.B., Watson, D.W., *Mapping Computer Vision-related Tasks onto Reconfigurable Parallel Processing Systems*, Computer, vol.28, No.2, Feb.1992, pp.55-63

Siegel, H.J., McMillen, R.J., 1: *The Multistage Cube: A Versatile Interconnection Network*, Computer, vol.14, No.12, Dec.1981, pp.65-76. 2: Siegel, H.J., Kemmerer, F.C., Smalley, H.E., *PASM: a Partitionable SIMD/MIMD System for Image Processing and Pattern Recognition*, IEEE Transactions on Computers, vol.c-30, No.12, Dec.1981, pp.934-947.

Siegel, H.J., *Interconnection Networks for Large Scale Parallel Processing: Theory and Case Studies*, McGraw-Hill, New York, 1990, ch.5.

Singleton, R.C., *A Method for Computing the Fast Fourier Transform with Auxiliary Memory and Limited High-Speed Storage*, IEEE Transactions on Audio and Electroacoustics, Vol. Au-15, No.2, June 1967, pp.91-98.

Skillicorn, D.B. *A Taxonomy for Computer Architectures*, Computer, vol.21, No.11, Nov.1988, pp.46-57

Smeulders, P.A., *Design and Performance Study of an Autonomous Processor Cell*, M.E.Sc. Thesis, Faculty of Graduate Studies, The University of Western Ontario, London, Ontario, 1989, c1988.

Snyder, L., *Introduction to the Configurable, Highly Parallel Computer*, Computer, vol.18, No.1, Jan.1982, pp.47-56.

Srini, *An Architectural Comparison of Dataflow Systems Computer*, vol.19, No.3, March 1986, pp.68-88

Stone, H.S., *Parallel Processing and the Perfect Shuffle*, IEEE Transactions on Computers, vol C-20, No.2, Feb.1971, pp.153-161.

Stone, H.S., *High Performance Computer Architectures*, Addison Wesley, Reading Mass., 2nd ed., 1990, 1:ch.4, 2:ch.3, 3:ch.5, 4:ch.7.2, 5:ch.6.3

Stone, H.S. and Cocke, J. *Computer Architecture in the 1990's*, Computer, vol.24, No.9, Sept.1991, pp.30-38

Tanenbaum, A., *Structured Computer Organization*, third edition, Prentice Hall, Englewood Cliffs, New Jersey, 1990. 1:p.15, 2:ch.8.2, 3:ch.8, 4:ch.8

Tucker, L.W., Robertson, G.T., *Architecture and Applications of the Connection Machine*, Computer, vol.24, No.8, Aug.1988, pp.26-38.

Van de Goor, A.J., *Computer Architecture and Design*, Addison Wesley, reading Mass., 1989, ch.16

Viswanath, T., Shanker, M., Prabhu, K.M.M., *Implementing Fast Fourier Transforms Using the Am29500 Family*, Microprocessors and Microsystems, Butterworth and Co., Oct. 1987, pp.423-430.

Von Neumann, J. *First Draft of a Report to the EDVAC*, 1945, Report for U.S. Army Ord. dept, reprinted in Papers of John Von Neumann on Computing and Computer Theory; Charles Babbage Institute reprint series for the History of Computing, M. Campbell-Kelly, ed., MIT Press, Cambridge Mass., 1987.

Watson, I., Gurd, J. *A Practical Data Flow Computer*, Computer, vol.15, No.2, Feb.1982, pp.51-57

Weicker, R.P., *A Detailed Look at Some Popular Benchmarks*, Parallel Computing 17(1991) 1153-1172, North Holland.

Worlton, J., *Toward a Taxonomy of Performance Metrics*, Parallel Computing 17(1991) 1073-1092, North Holland.

Wulforst, H. *Breakthrough to the Computer Age* Charles Scribner and Sons, New York, 1982 2:pp.63-64