HARMONIAC - A DIGITAL SIGNAL PROCESSOR

by

Philip Michael Connor B.E.

This thesis is an account of design work and its implementation submitted as full requirement for the degree of Master of Engin-eering, (Electrical), University of New South Wales, June,1981.

I hereby certify that the work contained in this thesis has not been submitted for a higher degree to any other university or institution.

Signature .

SUMMARY

OF

"HARMONIAC - A DIGITAL SIGNAL PROCESSOR"

This is an account of the design, construction and application of a low cost digital signal processor for audio frequency applications. The design shows how a fast, three address I6 bit computer with partitioned memory can be implemented with a relatively small amount of hardware. This implementation demonstrates the principles intended to achieve good performance and flexibility at low cost but is mot an attempt to build the smallest possible device.

The three address structure is achieved with a program wordlength of only I6 bits by limiting the addresses to five bits each and making all operations separate locations withim the 32 word address space. The main data memories are also accessed via locations in this space. The use of two buses allows simultaneous transfer of two operands to two destinations where they generally are operated on, providing a result at another address. The worst case execution time for normal operations is I50 nanoseconds. The use of auto-incrementing address registers on the main data memories allows greater speed in many algorithms.

An assembler written in Fortran, a debugging program and various utility programs such as a Fast Fourier Transform, real-time complex wave summation, division, logarithms and an exponential are described and listed. An integrated package for the analysis and resynthesis of the soprano singing voice which uses the above programs is described.

ERRATA

FOR

"HARMONIAC - A DIGITAL SIGNAL PROCESSOR"

Page 5 , line I2
"...30 milliseconds for a 5I2 point..."
Page I6 , line II
"...very high speed (90 nanoseconds)."
Page 50 , line II
"...via the TTY and line printer."
Page 69 , line I0
"...it would allow a shorter instruction cycle."
Page 70 , line I8
"...used (in ALU) and the memory speed."
In the Appendices:
Page iii - xvi should be moved , replacing xlii,xliii.
Page xxxv , line I3
"...see Appendix B , locations IDE - IF0."
Page xlv Appendix IV has no heading :
should be "HARMONIAC ASSEMBLER"
Page lvi , Appendix V (last page), point I0
" Total DIP count 440. (With 6K main memory , IK chips)"

On ALL Harmoniac Machine Language Listings the Decimal version of the program memory location has been inadvertently cut off the right edge of each page. The same information is given in Hexadecimal in the second column of the listing.

HARMONIAC - A DIGITAL SIGNAL PROCESSOR

INDEX

ACKNOWLEDGEMENTS

This work would not have been possible without the financial and personal assistance given by Macquarie University and the staff of the Speech and Language Research Centre (S.L.R.C.). In particular, I wish to thank A/Prof John Bernard who inspired the project from its beginning and my fellow workers, Harry Purvis, Mark Stevens, John Telec and Ian Yates, who worked tirelessly for nearly two years to complete it, and all the linguistic staff for their keen interest.

HARMONIAC - A DIGITAL SIGNAL PROCESSOR

1. INTRODUCTION

Harmoniac is a signal-processing computer designed for audio-frequency applications. It was designed as a low-cost digital processor for those signal-processing tasks which had to be performed at or near a "real time" rate in the Speech and Language Research Centre at Macquarie University. The intention was to provide a minimum-cost resource with sufficient speed in typical audio-signal-processing tasks to make flexible software simulations realistically usable for the researcher.

Even the fastest mini or micro computers are too slow to perform significant signal-processing tasks in real time. Some typical tasks in speech and music research include the production of power spectra, correlation digital filtering and the summation of sinusoids. As an example of the speed required, each sinusoid being used in the construction of a waveform requires one multiply, two adds, three memory operations and loop counting. A conventional minicomputer requires at least 15 μsec for this process when using 500 nanoseconds main memory. Since a new sample of the waveform must be produced at least once in 30 μsec, only two sinusoids would be possible. Similar operations are required for each pole or zero in a digital filter. The slow speed of a conventional machine derives from three limitations in design:

(1) There is only one main memory, so access is one datum at a time, with instruction accesses in between.

(2) Memory needs to be large for general-purpose use so its speed must be slow to keep cost down.

(3) Only one arithmetic unit is used, and it is often optimised for floating-point operation.

Without excluding any of the advantages of the conventional sequential machine on these sequential signal-processing tasks, this design (Harmoniac) avoids some of the disadvantages by the following features:

(i) Memory is divided into four simultaneously accessible parts - two data memories, program memory and a sine look-up table. (See 2.3 for rationale.)

(ii) Since the memories need not be very large for typical tasks, high speed (< 90 nanoseconds) static memory has been used.

(iii) Auto incrementing/decrementing address registers have been incorporated on the data memories to avoid unnecessary instructions steps when processing arrays of data.

(iv) The arithmetic unit, multiply and shift circuitry are separate and data retaining to provide very high speed processing and to reduce the need to store intermediate results in memory.

(v) Schottky bipolar logic is used throughout to keep size and costs low. (Relative to more exotic logic such as ECL.)

(vi) Multiple data paths (2) to allow full speed use of the two main memories and the processing elements. (See Fig. 1.)

Only 16-bit integer operations are provided as floating point is not required for normal signal processing. All instructions for

this machine are executed in a 150 nanoseconds cycle except the multiply which requires two cycles.

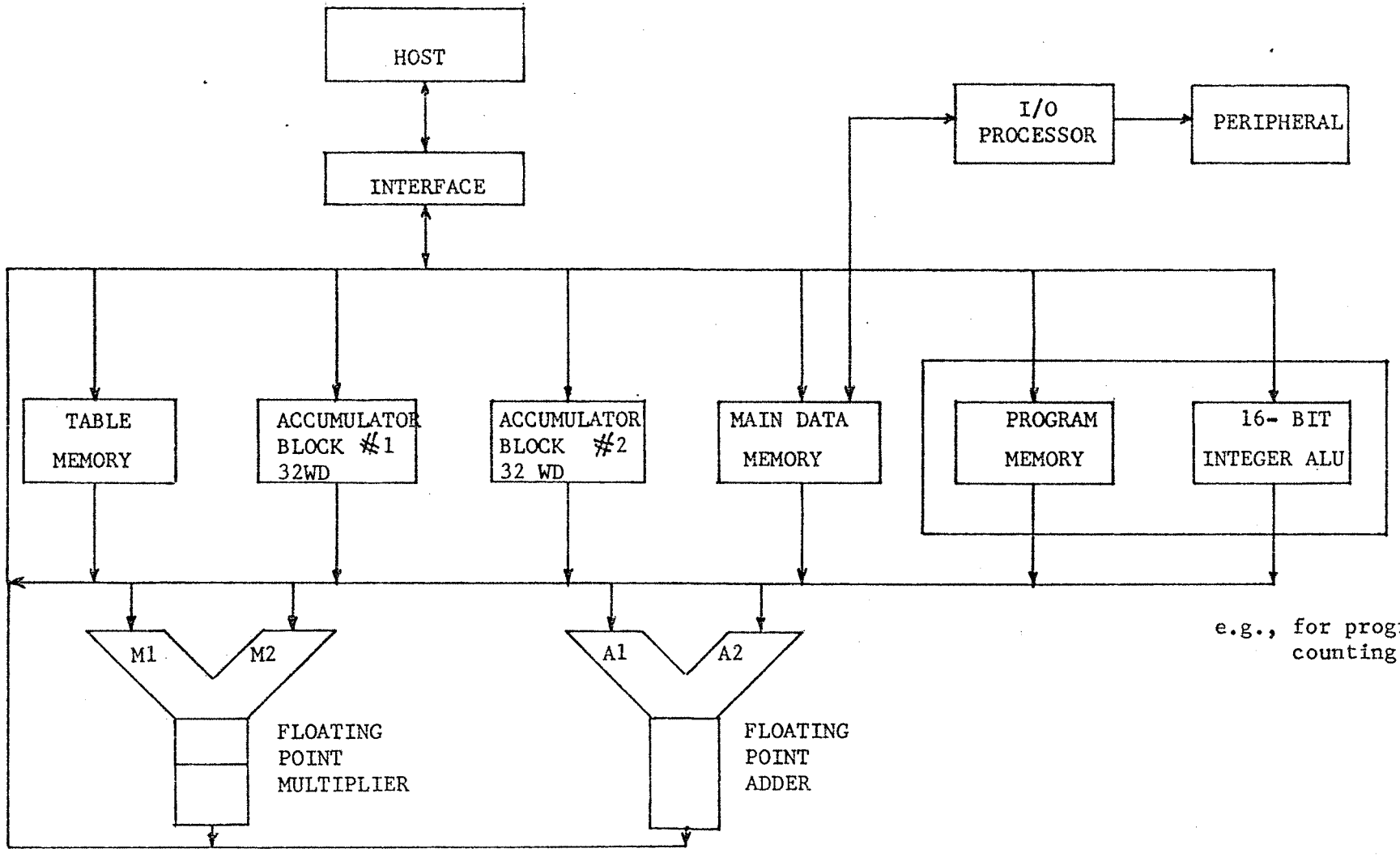
The normal mode for use of this type of processor is as a slave to a normal minicomputer which provides program and data through a 16-bit interface. This type of connection minimises tedious machine-language software development as many of the non-critical tasks can be performed in a higher level language on the host machine (see 4.4 on Singing Voice synthesis as an example of such a division of labour). In this way only a few basic algorithms need to be developed for the signal processor. Those already written and in use are detailed in section 4.3. See section 4.5 for stand-alone operation.

1.1 Historical Review

Most recent types of stored-program digital computers have at some stage been used for some kind of signal processing, but here we will restrict our discussion to machines capable of some useful real-time processing of full audio-bandwidth signals. Such processing can be performed on a large mainframe computer although the cost is usually very high as such a machine is very inefficiently used in real-time audio tasks. The inefficiency derives from the very low usage of the large random-access memory and discs which are lying idle while the central processor is virtually fully occupied. This is an inappropriate use of the general-purpose mainframe type of processor. Some of the more appropriate, non real-time uses of mainframes in the musical field are covered by Mathews in ref. 17. The conventional minicomputer or microcomputer is simply too slow for significant real-time processing (see the introduction). So a more specialised machine has usually been employed in real-time audio applications.

These specialised machines have used a variety of approaches to achieve the processing speeds required, and most of these approaches are relatively expensive. The fastest machines have used Emitter Coupled Logic (ECL) in multiple-arithmetic units. Such machines are far more complex and expensive than Harmoniac so they will not be covered here. An example of a multi-arithmetic-unit machine is the Lincon Laboratory FDP, a fast programmable signal processor described in reference 1 (1971). It uses four AU's with a separate array multiplier, two data memories and employs 18-bit fixed-point arithmetic. The multiple arithmetic units (AU's) make the programming organisation of this machine somewhat difficult. It has been designed basically to be able to perform the basic butterfly (inner loop) operation of an FFT in one instruction, while allowing more general-purpose operations to be performed with the same hardware. Another rather special-purpose machine is described in ref. 2 (1975). This machine, designed by Renato and De Mori, uses an ECL arithmetic unit, 14-bit precision and TTL memories to achieve high speeds without very high costs but the precision is a little low and the complexity rather too high to make comparison with a machine like Harmoniac fair. Both of these machines (ref. 1,2) can perform 512 ft. real time FFT's in under 3 milliseconds (versus approx. 20 milliseconds in Harmoniac).

Reference 10 describes the AP120B, a so-called "array processor" which achieves similar speed to the FDP (above) but operates with 38-bit floating-point arithmetic. This is considerably higher precision than is required for most audio processing tasks but it is interesting to see that its degree of parallelism is lower than Harmoniac in some respects (see Figure 25). It uses two blocks of accumulators, similar to the "scratch" memories of Harmoniac, with separate program and table memories as does Harmoniac. 64-bit in-



e.g., for program counting

Fig. 25

struction words allow a more powerful addressing and interconnection system, so that the ALU and multiplier can connect to all memories freely. There are some restrictions on such interconnections in Harmoniac, due to the 16-bit instruction word. Only the most useful connections are readily available. The AP120B allows for pipelining of operations to a larger extent than Harmoniac, mainly because it has a longer multiply time. The normal operation-execution time is very similar at 167 nanoseconds.

There are several simpler machines which are more fairly to be compared with Harmoniac. The SPS-41, described in ref. 14 (1975) is a 16-bit fixed-point machine with a 200 nanosecond instruction cycle which takes approximately 300 milliseconds for a 512-point real FFT (Harmoniac 20 milliseconds). It is a triple microprocessor machine with six ALU's, four multipliers and four memories. It seems to be more costly and complex than Harmoniac and of slightly lower performance. An even more comparable machine is described in ref. 6 (1978).

This is called G.A.S.P., a general-purpose signal processor designed and built at the University of Adelaide at about the same time as Harmoniac using similar chip types (similar level of integration). The main differences are the use of a floating-point arithmetic, 20-bit wordlengths, multiplexers instead of tristate buses and a single data memory. The cost and complexity of this machine is three times that of Harmoniac and its speed is similar on typical tasks. The greater wordlength is a definite advantage over Harmoniac.

The only powerful real-time audio-signal processor so far located which is simpler and cheaper than Harmoniac is the Lincon Laboratory microprocessor Linear Predictive Vocoder described in ref. 4. This is actually a general-purpose machine with a fixed program in ROM. It is

a 16-bit integer machine with a 150-nanosecond instruction cycle, one data memory, a separate 48-bit program memory and a four cycle (600 nanosecond) multiplier. The very large width of the instruction word allows powerful instructions but the single data memory would limit its performance relative to Harmoniac. It uses only 162 dual in-line packages compared to Harmoniac's 440. Package count has been kept low because very little data memory is provided (2000 words), high-density chips have been used and the multiplier is only one quarter of a full array.

There are now several single-chip bipolar microprocessors intended for simple real-time signal processing. These are too limited in capacity to be compared with Harmoniac. It seems that they are intended for low-bandwidth digital filtering.

With this background it can be seen that Harmoniac fits in as a low cost, moderately high-performance signal processor. It has no exact equivalent amongst its peers but seems to give a higher performance to cost ratio than any in the literature except perhaps the Lincon Lab. machine in ref. 4. But the Lincon Lab. processor is not entirely comparable as its real-time signal-processing power is probably about half that of Harmoniac on tasks such as filtering and sinewave synthesis because of its single data memory and slow multiply.

The details of speed, precision, memory and instructions necessary for audio-signal processing are considered in the following sections 2.1, 2.2, and in greater detail in ref. 5.

2. ARCHITECTURE AND DESIGN CONSIDERATIONS

2.1 General Requirements

The design of Harmoniac was undertaken after the author had completed the programming of a number of signal processing tasks in speech work. These included the design of an interactive Fourier transform, power-spectrum analysis package, various pitch-detection algorithms and an additive sinewave-synthesis routine. This experience showed that most of the speech processing tasks in the S.L.R.C. (Speech and Language Research Centre, Macquarie University) could be accomplished with a 16-bit integer machine, but that certain algorithms either required pre-scaling of the data (block floating point) or a longer wordlength in critical sections. A typical example is FFT's performed on 12-bit data. When the number of points in the transform exceeds 256, greater than 16-bit precision may be required to prevent overflow as the data grows by $\sqrt{N}$. Recursive digital-filtering processes often require wordlengths of 24-bits and more for stability and low noise (refs. 5, 16) although most filters for speech synthesis and linear-prediction analysis of speech can be implemented in 16- to 20-bit wordlengths.

A good compromise which can cope with most audio processing tasks is 20-bits (fixed point) per word but it was decided to stick with the minicomputer standard of 16-bits in this implementation, using block floating-point techniques (software exponent) where necessary to maintain precision. Occasionally double precision is necessary (see 4.3.1).

This compromise was made on the basis of lower cost, simpler interfacing to 16-bit machines and generally simpler hardware.

The processing speed required in a digital signal processor is always ultimately limited by a cost benefit ratio. If FFT processing of real-time audio data is taken as an example, there is ultimately a judgment to be made as to how often in time the results are required and how much detail in frequency is required. Typically results are required every 10 milliseconds but the frequency resolution is a compromise between smearing the analysis over too much time and getting the best resolution of frequency detail. To see only the major resonances in speech the frequency resolution need not be better than 100 or even 200 Hz.

The overall bandwidth to be dealt with is usually a much easier decision. For speech, 4 to 8 KHz is sufficient whereas music may require up to 15 or 20 KHz bandwidth. Musical analysis is perhaps an even finer art than speech analysis as it needs to be seen at several different resolutions in time and frequency at the same instant for every aspect to be covered.

Given a bandwidth requirement of 5 KHz and a resolution requirement of 40 Hz with 10 milliseconds between result frames, the system must generate spectra of 128 pts. ($\sim \frac{5\ \text{KHz}}{40\ \text{Hz}}$) every 10 milliseconds. This requires a 256 point real FFT every 10 milliseconds, just possible in Harmoniac. (FFT execution time is proportional to $N \log_2 N$.)

As mentioned in the introduction, each independent sinusoid generated or each pole/zero of a digital filter requires about one multiply, two adds, three main memory accesses and loop counting overhead - minimum of eight instructions in a two-operand machine such as Harmoniac, and generally more. Hence the processing speed required in such algorithms is easily calculated as (approx.): instruction time x 10 = time per pole (or sine). So a 150 nanoseconds instruction

time implies 1.5 milliseconds for updating each pole in a simple filter.

Higher speeds can be obtained either by using a faster logic type or by eliminating instructions in the inner loop by the inclusion of more specialised hardware.

2.2 Operations and Memory Required

The usual integer operations must be available - logical (and/or), add, subtract, multiply, divide, shift and compare, and the use of two's complement arithmetic for these eight operations seemed to be the most pratical to use. The basic add, subtract, logical and compare operations have been implemented in a medium scale integrated arithmetic unit (using 74S181 chips). The frequent requirement for fast multiplies dictated the choice of an array multiplier rather than the usual shift/add variety. The very infrequent requirement for division in the signal processing allowed it to be left to a conventional softare shift and subtract algorithm employing the aritmetic unit.

The choice of logic type to be used was dictated by the need to keep the machine simple and cheap but at the same time as fast as possible. Emitter coupled logic (ECL) is expensive, large, and power-hungry while metal-oxide-semiconductor (MOS) large-scale integrated circuites (LSI) are far too slow. The availability of a large range of functions in medium-scale integrated-circuit chips in Schottky Transistor Transitor Logic STTL) made this the natural choice for a fast, cheap machine (in 1975).

Memory requirements for signal processing in real-time applications are usually quite modest. There are a few algorithms which require more

than four thousand words of data memory and program/memory requirements are usually in the hundreds of words for a reasonably efficient machine-code implementtation of a Fast Fourier Transform. An early decision was taken to have each word in the program memory correspond to a complete instruction for speed and simplicity.

2.3 The Chosen Structure

Since most arithmetic operations require two operands and produce one or two results, it seemed natural that the machine should have two data buses in order to move both operands at once. For the same reason the memory in which the bulk operands are to be stored should be divided into two pieces separately accessible for the two buses. The other important structural choice from a speed point of view was to keep the stored program in a separate memory so that one instruction can be executed while another is being fetched - pipelined instruction fetches. The block diagram in Fig. 1 shows the basic two-bus structure with three independent memories plus a table memory.

To keep the instruction wordlength short but allow powerful instructions, it was decided normally not to specify main memory addresses directly in the instruction word. Instead, addresses are normally set up be a separate instruction which stores the address in a memory-address register.

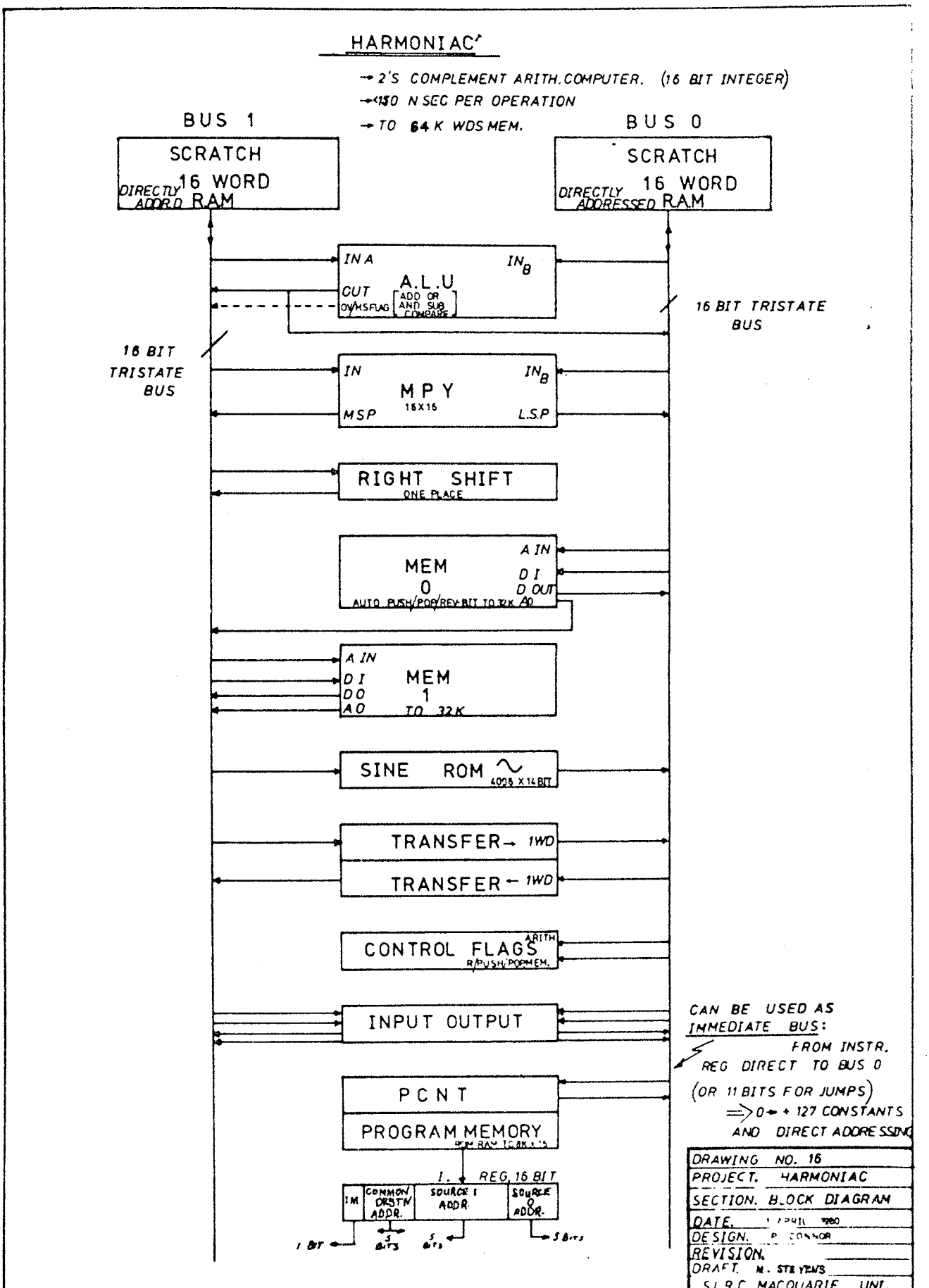

HARMONIAC
→ 2'S COMPLEMENT ARITH. COMPUTER. (16 BIT INTEGER)
→ <150 N SEC PER OPERATION
→ TO 64 K WDS MEM.
BUS 1
BUS 0
SCRATCH 16 WORD DIRECTLY ADDR'D RAM
SCRATCH 16 WORD DIRECTLY ADDRESSED RAM
IN A
IN B
A.L.U
OUT
OVFL FLAG
ADD OR AND SUB COMPARE
16 BIT TRISTATE BUS
16 BIT TRISTATE BUS
IN
IN B
MPY
16X16
MSP
LSP
RIGHT SHIFT
ONE PLACE
MEM 0
A IN
D I
D OUT
AUTO PUSH/POP/REV BIT TO 32K A0
A IN
D I
D O
A O
MEM 1
TO 32K
SINE ROM
4095 X 14 BIT
TRANSFER → 1WD
TRANSFER ← 1WD
CONTROL FLAGS
ARITH
R/PUSH/POP MEM.
INPUT OUTPUT
CAN BE USED AS IMMEDIATE BUS:
FROM INSTR. REG DIRECT TO BUS 0
(OR 11 BITS FOR JUMPS)
⇒ 0 → + 127 CONSTANTS AND DIRECT ADDRESSING
PCNT
PROGRAM MEMORY
I. REG 16 BIT
IM
COMMON DEST'N ADDR.
SOURCE 1 ADDR.
SOURCE 0 ADDR.
1 BIT
5 BITS
5 BITS
5 BITS
DRAWING NO. 16
PROJECT. HARMONIAC
SECTION. BLOCK DIAGRAM
DATE. APRIL 1980
DESIGN. P. CONNOR
REVISION.
DRAFT. M. STEVENS
S.L.R.C MACQUARIE UNI.

Fig. 1.

The number of basic instructions required is small (approx. 8 sufficient) so it was decided to arrange the instructions and memory access ports together with a small set of general-purpose registers as one thirty-two-word address space. All instructions are executed as transfers within this directly addressable thirty-two-location space. This means that only five bits are required for any address so that a three-address specification can be given in fifteen bits, one address being the common-destination address for both buses and the other two being the source addresses on each bus. As can be seen in the block diagram (Fig. 1) the top sixteen addresses are the "scratch" or general-purpose register set on each bus. The lower sixteen addresses are instruction inputs and data memory inputs and outputs.

Each instruction is a separate piece of hardware, except for those performed in the arithmetic unit. This allows for the possibility of asynchronous instruction execution where a slow instruction such as the multiply can be left to go to completion while several other instructions are executed. For this purpose an input register is provided on every instruction so that the results of the instruction are available any time after the propagation delay of that instruction. This feature, together with the use of tristate bus elements makes the design very flexible - one instruction could be substituted for another or new ones added if the address space is expanded. The general structure of an instruction is shown in Fig. 2.

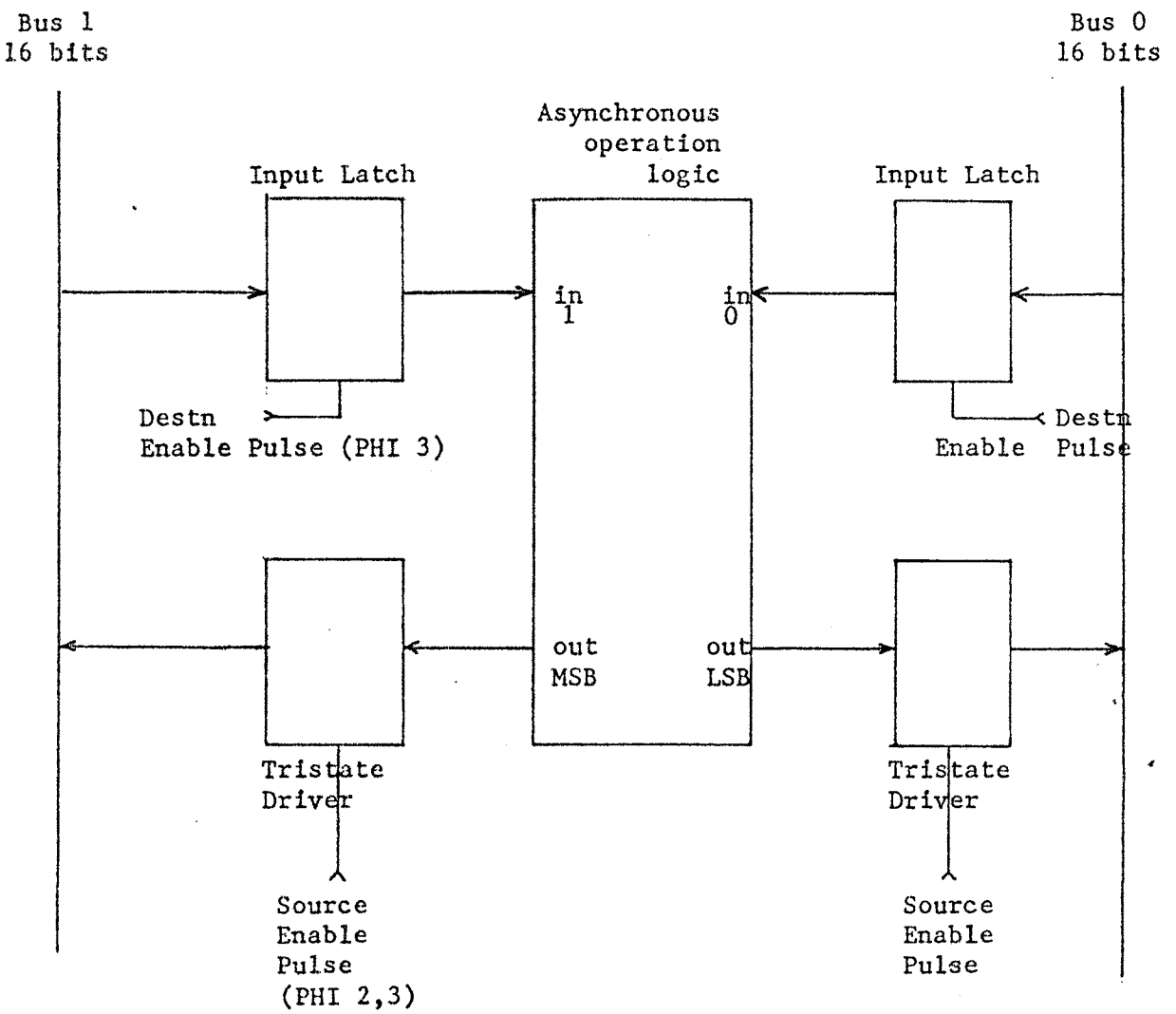


Fig. 2

The initial specification of addresses for the data memories is often done using a special mode where most of the instruction word is used as data directly onto bus zero, whence it can be stored into a memory-address register or any other register. If it is stored into the program counter, a jump is performed. This mode is called the immediate mode and a bit is reserved in the instruction word to specify it. In immediate-mode instructions, some

special decoding is performed to allow the maximum number of bits possible, particularly for jump instructions so that all jumps can be directly addressed (Fig. 8). If the immediate-mode bit is not set, the instruction is always a straight transfer from two of the thirty-two locations on the buses to a common pair of destination locations. Hence all the fifteen bits remaining in the instruction word are used to specify these three 5-bit addresses. Transfers between the lower addresses (containing instructions and main data memory) are effected by demultiplexers driven by the respective address fields of the instruction word and synchronised with the appropriate phases of the three phase clock. (See 3.2.1 and Fig. 6.)

For many signal processing tasks, programs can be devised where main data memory addresses do not need to be specified, except at the beginning of a processing loop, by using the hardware memory-address counters. These allow auto increment, decrement and reversed bit counting of addresses while processing an array (or two arrays) of data with the option of the address counting being triggered by either a read or a write to the respective memory. (See 3.2.3.)

2.4 Operation Timing

The basic timing of an operation is very simple as all that is necessary is to enable an output to drive the bus and, after data settles, provide a positive edge pulse to latch the data from the bus into its destination.

This can be seen in Fig. 3.

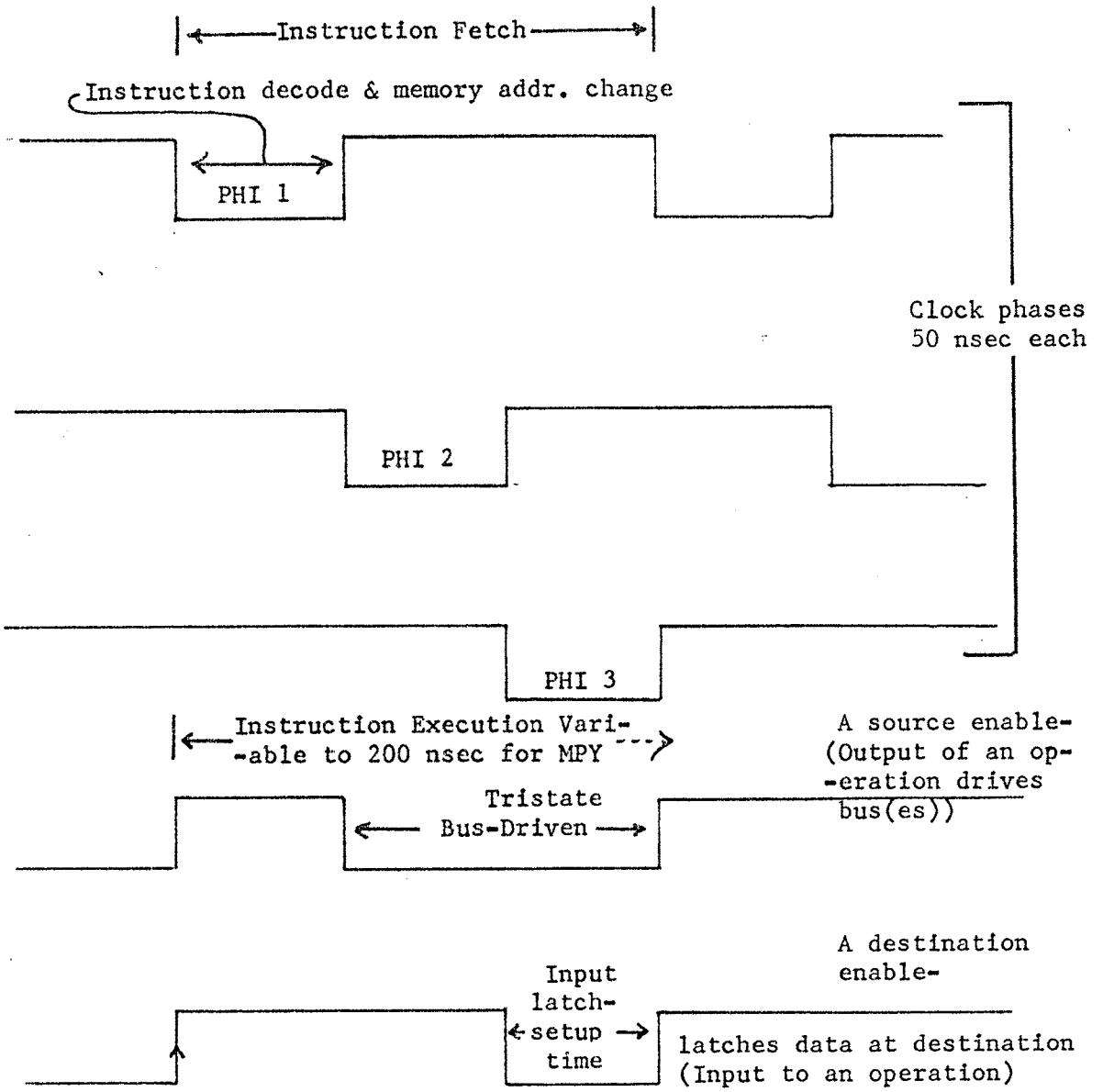


Fig. 3

The source is enabled first to allow for capacitive delays in charging the bus lines and then the destination pulse is generated to latch the data. A third phase (PHI 1) is provided to allow the next instruction to be decoded after it is fetched from the program memory, and to allow the main-memory-address setup time.

2.5 Other Possibilities

The unusual structure chosen did not lend itself to the use of bit-slice microprocessors - the other most suitable way to implement the arithmetic unit, compare-and-shift circuitry. Microprogramming was rejected on the grounds that it introduces further delay and would make the design more complex without providing a significant increase in speed, although it would have made programming easier. There is no significant speed improvement possible from microcoding as the instruction fetch is performed simultaneously with the execution of the previous instruction and all the memories are equally high speed. The instructions provided are almost microinstructions in their simplicity and the instruction decode and timing are extremely simple. (See Figs. 6 and 8 and ref, 15.)

Another major method of achieving a fast processor is the use of microprocessor arrays where each microprocessor has only moderate performance but the overall result is very fast execution of a complex algorithm (ref. 11, 12, 13). This approach was considered to be too difficult for the programmer in the case of many of the signal-processing algorithms although ways of treating them in parallel may be evolved in the future.

In a stand-alone signal processor it is often (not always) necessary to buffer a set of samples before beginning processing. This usually requires an interrupt structure and real time clock so that processing on a previous block can continue while a new block is being stored. This feature was not provided in Harmoniac as it was thought that a host microprocessor could provide such functions for minimum cost. See Table 1 for a summary of some of the relevant design alternatives.

TABLE 1

Summary of possible alternatives NOT used in this design

* Floating point - not essential.

* Longer wordlength - not essential.

* Mos microprocessor array - difficult to program (not always efficient).

* Bit-slice micro - existing designs do not suit a double-data memory, double-data bus design.

* Microprogramming - no faster because instruction fetch is a doubly overlapped pipeline arrangement and main memory is very high speed (100 milliseconds).

* ECL logic - more expensive and physically larger.

* Interrupt logic - not essential if host processor used. (For real-time operation the processor must be faster than necessary so some time is always wasted.)

Summary of Design Features of Harmoniac

* Very high-speed memory (90 nanoseconds) - only a small amount required for typical algorithms.

* 4 separate, simultaneously accessible memories - three can be accessed at one time and program memory fetch is at same time as instruction and execution.

* Two separate data buses - provides simultaneous transfers of two operands to an operation.

* Major operations implemented in independent, asynchronous, data-latching blocks of hardware - to allow pipelining and to minimize storage of intermediate results.

* Operations and data memory parts treated as locations in a small 32 word memory space to minimise instruction width.

3. SECTIONAL DETAILS OF THE HARDWARE

3.1 Construction

A single array of wire wrap sockets ("cambion") was employed to hold the 440 Schottky chips used in Harmoniac. This fits into a standard.19 inch (47.5 cm) rack mounting chassis approx. 10 cm high. No switches or controls were provided on the front panel as all control is executed through the host processor. A photograph of the chassis with the top up is shown in Fig. 4. The power-supply regulator is a conventional series-pass type mounted on the rear of the chassis to share the cooling fans which pressurise the interior where the logic chips are mounted. The airflow is in through the top and out through a slot along each side and a hole at the rear for the regulator heat-sink. The unregulated voltage (+ 12 VDC) is supplied from a separate chassis containing the power transformer, rectifier and associated components. All the logic is powered by 5 volts (at 27 amps).

The wire wrapping was done manually from computer-generated and checked listings. The program to verify the wire wrapping was specially written for this project in ALPHA-16 machine language for a Computer Automation ALPHA-16 minicomputer. Wrapping was point to point, level ordered over a ground plane.

Bus interconnections between sections of the machine were achieved by 16-core flat cable plugged into standard 16-pin sockets in the logic array. This system allows any section to be isolated from the bus for fault-finding purposes.

The layout of chips on the chassis is shown in Fig. 5.

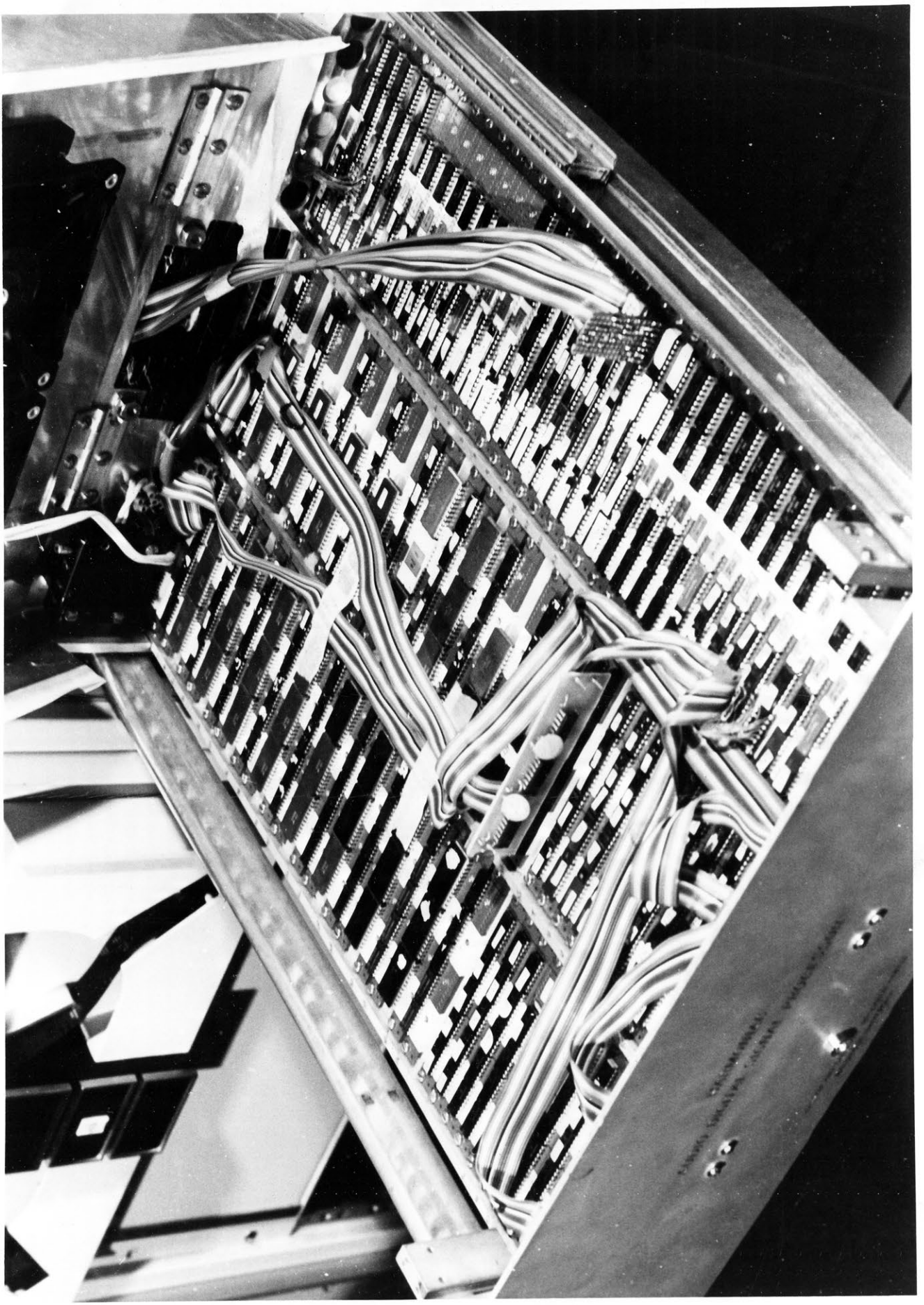
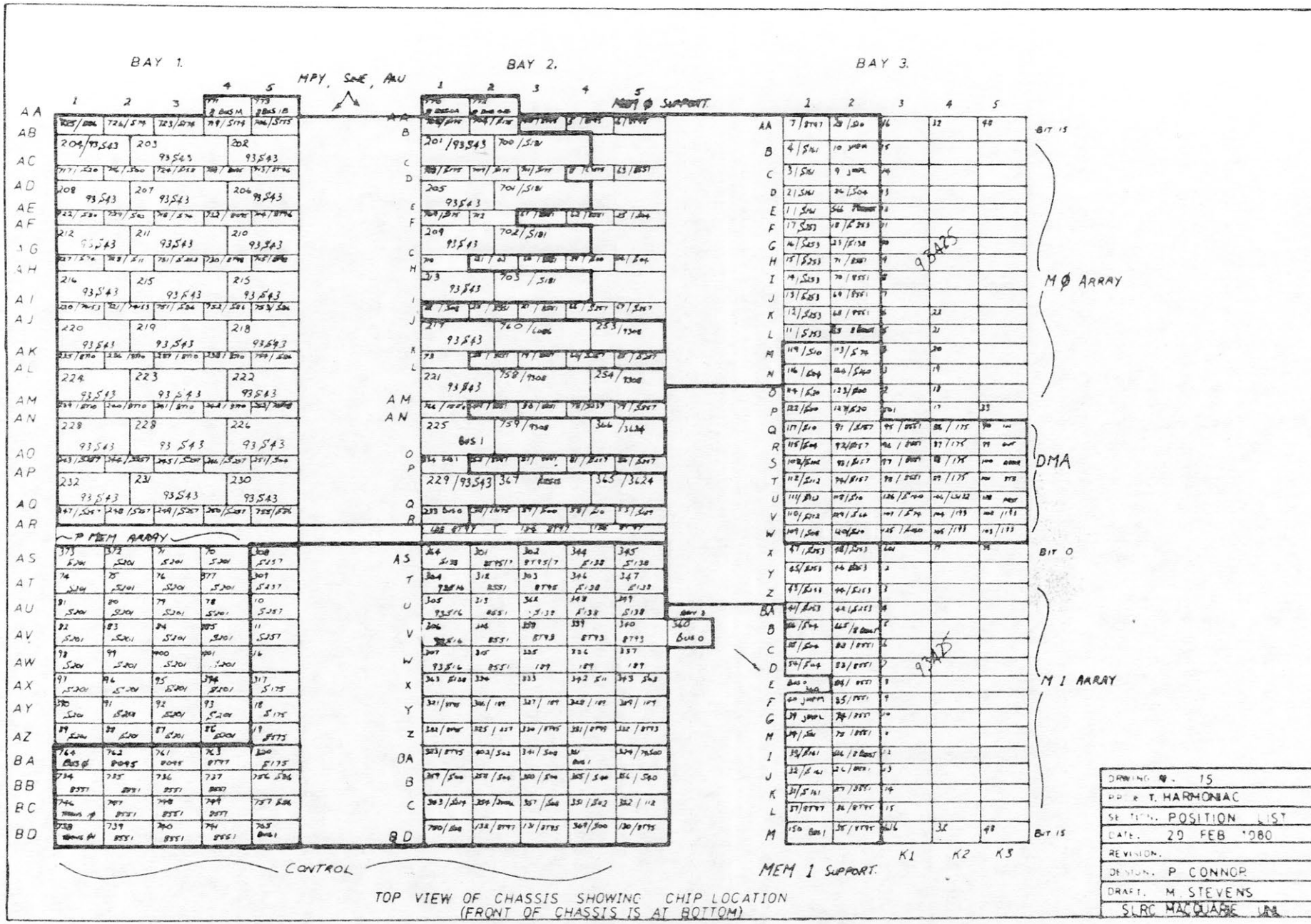


Fig. 4

Fig. 5


BAY 1.
BAY 2.
BAY 3.
MPY, SQRT, ABS
MØ SUPPORT
MØ ARRAY
DMA
M1 ARRAY
P MEM ARRAY
CONTROL
MEM 1 SUPPORT
BIT 15
BIT 0
K1
K2
K3
TOP VIEW OF CHASSIS SHOWING CHIP LOCATION
(FRONT OF CHASSIS IS AT BOTTOM)
DRAWING #. 15
DRAWN T. HARMONIAC
SECTION. POSITION LIST
DATE. 20 FEB '080
REVISION.
DESIGN. P CONNOR
DRAFT. M STEVENS
SLRC MACQUARIE UNI

3.2 The Subsections

3.2.1 The Control Section

The basic timing of the machine, the scratch registers, the program memory and instruction decode are provided in the control section which can be seen in the lower left part of Fig. 5. It consists of thirty-two chips in the program memory array of 512 words and fifty-nine chips in the remainder. Detailed circuit diagrams are shown in Figs. 6, 7, 8 and 9.

The clock oscillator which times all events in the machine is basically a crystal oscillator, but for flexibility in the prototype a voltage controllable oscillator was used. The socket position 354 can be occupied either by a crystal of about 20 mHz or a voltage-control trim pot assembly, as shown in Fig. 6 (left centre). This oscillator drives a ring counter of three J - K flip flops which generates the basic three-hase timing waveforms (chips 351, 352). A stop switch is provided on the oscillator for static testing - Harmoniac is completely static in operation and can be run at any clock rate up to the maximum (approx. 20 mHz).

The program counter and associated logic is shown in Fig. 7. It is advanced at the beginning of each cycle by Φ1, addressing the next location in the program memory. Initially it is set to zero by the host processor vial MRL - . Jumps are executed by a store into the program counter via 301, 302, 303. Subroutine jumps simultaneously save the previous contents of the program counter into a latch (312-315) so that a return is possible. The "fill mux" is used only during direct memory access transfers which store into the program memory. Otherwise the "fill mux" (308-310) acts as a memory address driver for the program memory. The PROMS (programm-

able read-only memories) shown (365, 366) are not installed as yet in the prototype, but are intended to provide either a set of "system" subroutines or be used for a stand-alone, fixed software arrangement.

The program memory array (Fig. 8) stores 512 words of 16 bits. Its output is latched at the start of a cycle ($\Phi 1$) to provide the next instruction.

Also in Fig. 8 can be seen the "immediate-mode" bus drivers (321-323). These are used to provide part of the instruction word direct onto bus zero for execution of jumps (direct) and to give small positive numbers for simple arithmetic. A separate "immediate-mode" bit of the instruction word is used to disable the normal source address for bus zero and to enable the five-bit source-zero field (see Fig. 1 bottom) directly onto bus zero together with 2 bits borrowed from the source and destination fields. The presence of a jump or jump-to-subroutine destination with the immediate-bit set is decoded to use the whole of the five-bit bus one-source address directly on bus zero as well as the normal seven immediate bits to give eleven bits for the jump address on bus zero with no operation being performed on bus one. This allows direct addressing for jumps within 2048 locations of program memory - more than sufficient for realistic algorithms. For convenience this address space has been divided into two parts with random-access memory (RAM) at the bottom and PROM at the top in normal operation. A switch is provided (see Fig. 7 Chip 756) to reverse this order for stand-alone operation putting the PROM at the bottom so that execution will begin in the PROM at power up (512 words are allowed for).

There are two immediate modes, one using six bits of the instruction word and the other seven bits. The mode is changed by a store into the arithmetic control flag (see Fig. 1 and ALU Fig. 15). The seven-bit mode restricts source addressing on bus one to the lower eight sources of operations and scratch registers and both modes prohibit storing into a scratch-register destination.

Fig. 9 shows the scratch-register files which are the top sixteen locations of each of the three address fields. The demultiplexers, chips 344-349, on Fig. 6 decode the source and destination addresses to drive the operation-input latches and tristate outputs respectively. The "NOP" operation, which inhibits an operation on one or both buses, is specified by all zeroes in one of the source fields. This is decoded by the source-enabling demultiplexers 347 and 349 to produce "BUSONOP" or "BUS1NOP" which then inhibits the destination demultiplexer and prevents a tranfer on that bus. Note that the jump to subroutine or call instruction is destination zero, which normally causes a no-operation, but the presence of the immediate-bit set causes the jump to subroutine to be executed.

The fact that the next instruction is being fetched at the same time as the present instruction is being executed means that a jump instruction (or "subroutine call") will not prevent execution of the very next instruction in line. This next instruction could be called the "jump shadow" and must be kept in mind always when programming or some very unusual "bugs" can turn up.

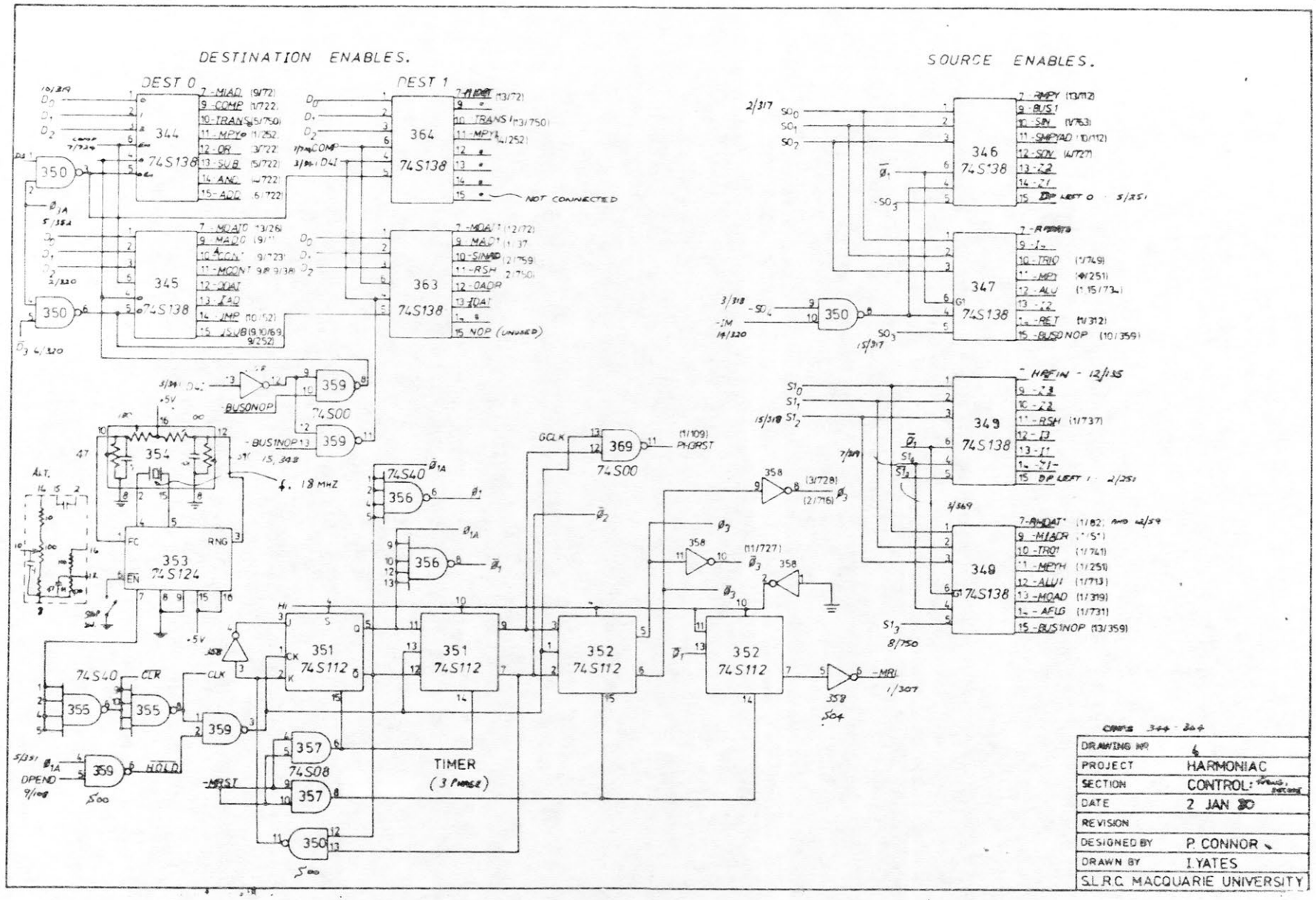

DESTINATION ENABLES.
SOURCE ENABLES.
DEST 0
DEST 1
344
74S138
364
74S138
345
74S138
363
74S138
346
74S138
347
74S138
349
74S138
349
74S138
350
359
74S00
354
353
74S124
74S40
356
369
74S00
351
74S112
352
74S112
355
357
74S08
TIMER
(3 Phase)
DRAWING NR 4
PROJECT HARMONIAC
SECTION CONTROL:
DATE 2 JAN 80
REVISION
DESIGNED BY P. CONNOR
DRAWN BY I. YATES
S.L.R.C. MACQUARIE UNIVERSITY

Fig. 6.

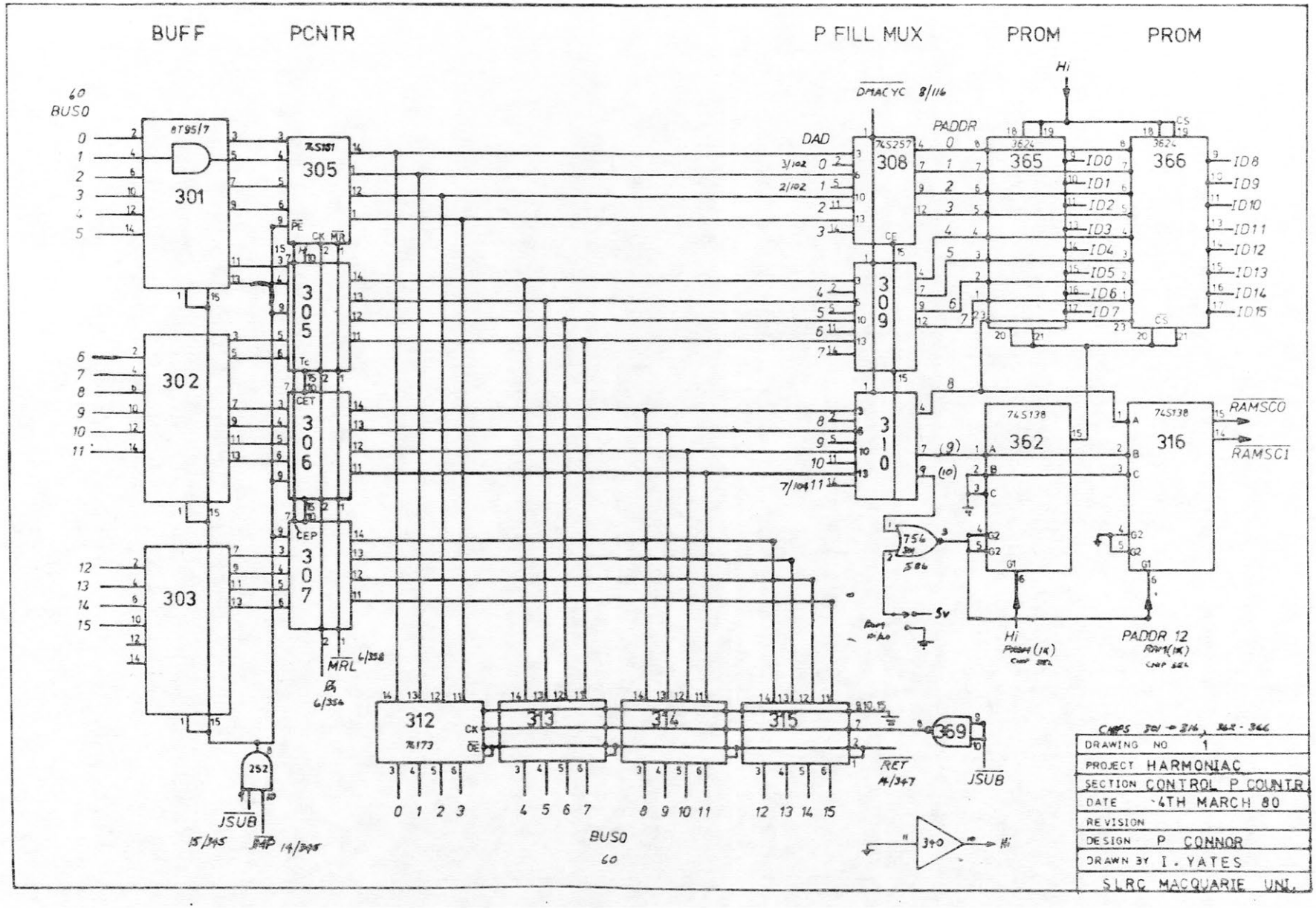

BUFF
PCNTR
P FILL MUX
PROM
PROM
BUS0
301
302
303
305
306
307
308
309
310
312
313
314
315
362
316
365
366
369
252
DAD
PADDR
DMACYC
Hi
5v
MRL
JSUB
RET
RAMSC0
RAMSC1
PADDR 12
BUS0
DRAWING NO 1
PROJECT HARMONIAC
SECTION CONTROL P COUNTR
DATE 4TH MARCH 80
REVISION
DESIGN P CONNOR
DRAWN BY I. YATES
SLRC MACQUARIE UNI.

Fig. 7.

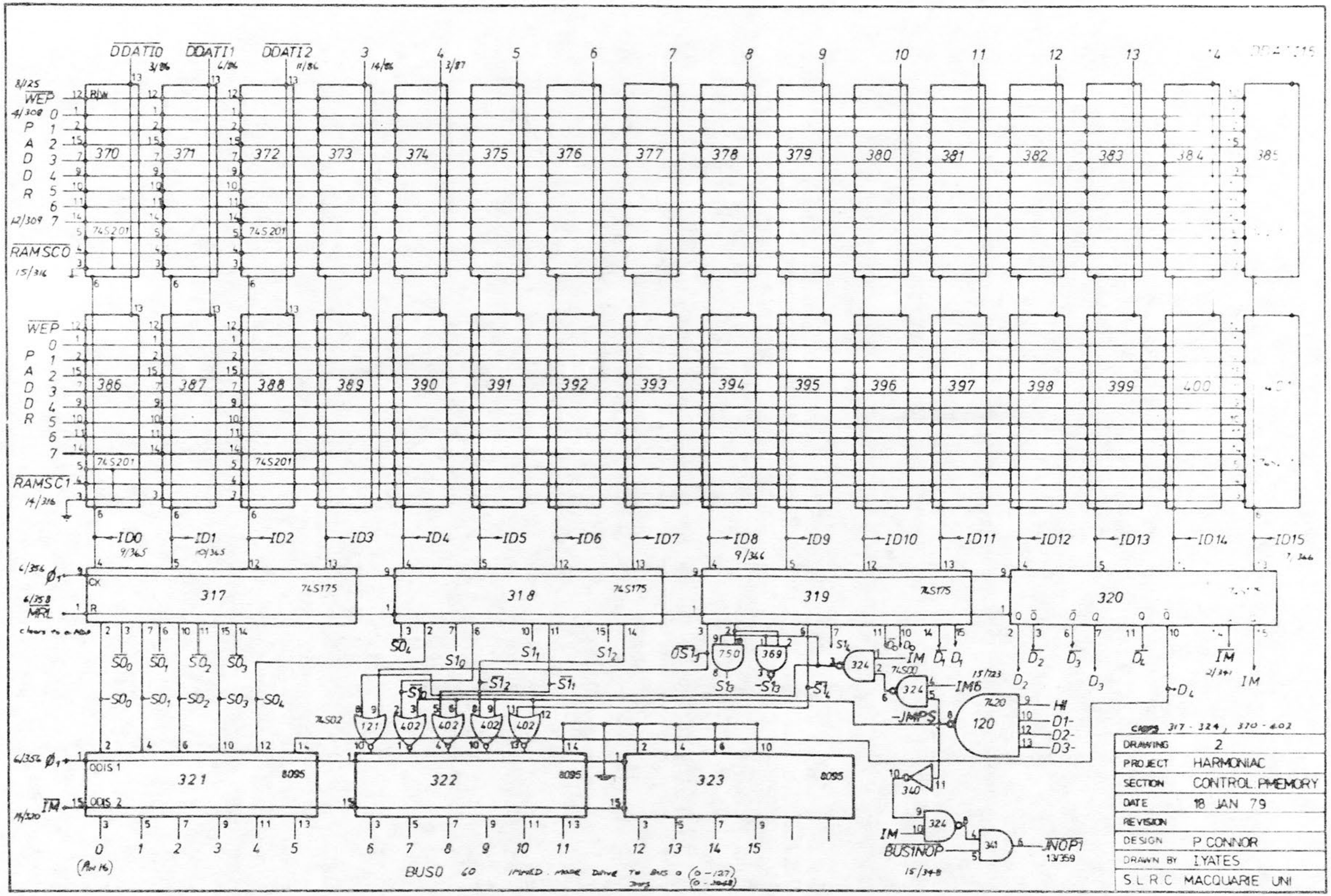

DDATI0
DDATI1
DDATI2
WEP
PADDR
RAMSC0
RAMSC1
74S201
ID0 ID1 ID2 ID3 ID4 ID5 ID6 ID7 ID8 ID9 ID10 ID11 ID12 ID13 ID14 ID15
317 318 319 320
74S175
321 322 323
8095
BUS0
INOP1
BUSINOP
CHIPS 317 - 324, 370 - 402
DRAWING 2
PROJECT HARMONIAC
SECTION CONTROL PMEMORY
DATE 18 JAN 79
REVISION
DESIGN P CONNOR
DRAWN BY I YATES
S.L.R.C MACQUARIE UNI

Fig. 8.

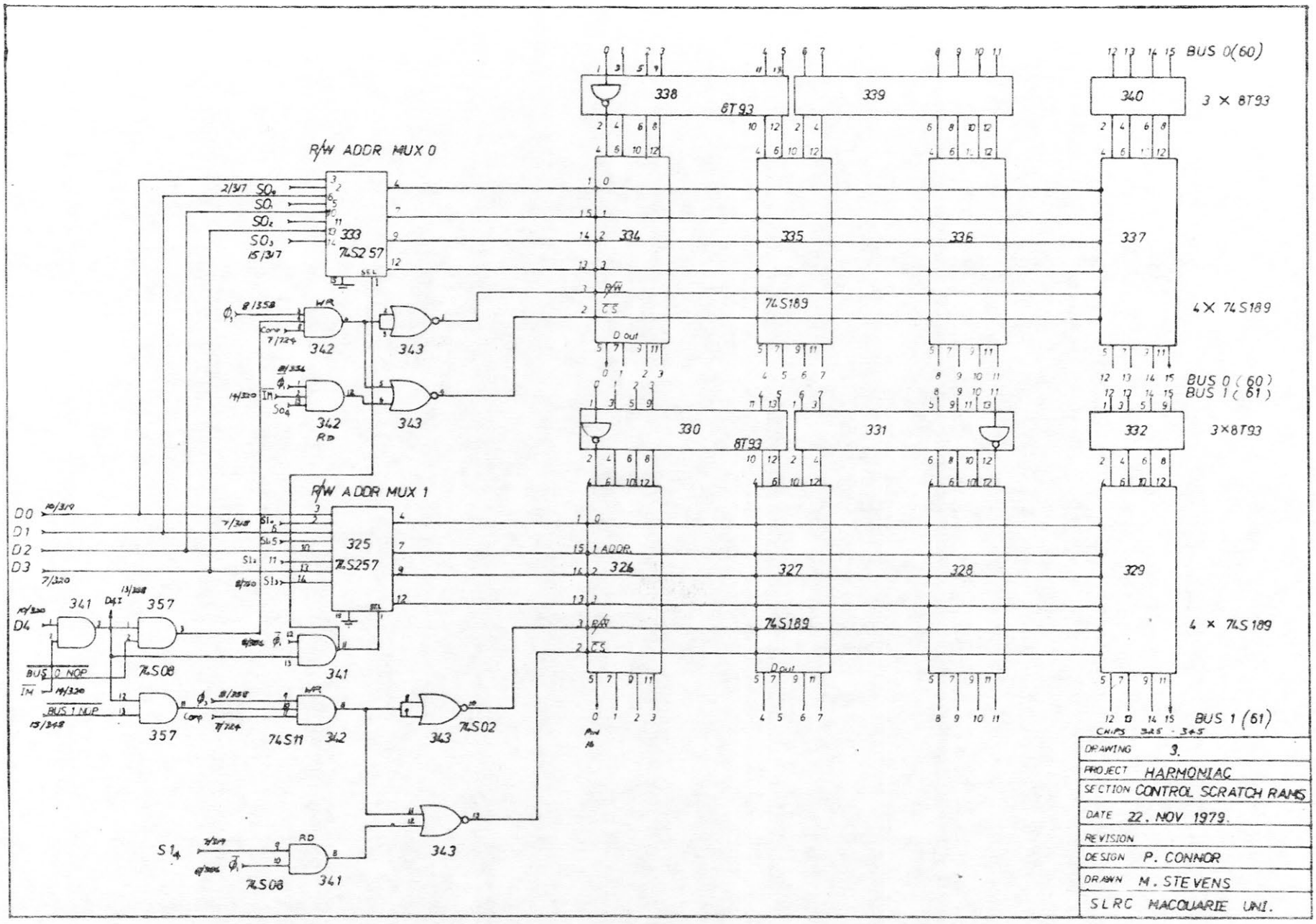

R/W ADDR MUX 0
R/W ADDR MUX 1
333
74LS257
325
74LS257
338
339
340
330
331
332
8T93
3 × 8T93
326
327
328
329
334
335
336
337
74LS189
4 × 74LS189
BUS 0(60)
BUS 0 (60)
BUS 1 (61)
BUS 1 (61)
342
343
341
357
74LS08
74LS11
74LS02
D0
D1
D2
D3
D4
S1
RD
WR
BUS 0 NOP
BUS 1 NOP
D out
DRAWING 3.
PROJECT HARMONIAC
SECTION CONTROL SCRATCH RAMS
DATE 22. NOV 1979.
REVISION
DESIGN P. CONNOR
DRAWN M. STEVENS
SLRC MACQUARIE UNI.

Fig. 9.

3.2.2 The Direct Memory-Access Section (DMA)

In the prototype all communications with the outside world have been provided through the DMA channel. Although a programmed input/output section was allowed for in the design, it was not considered necessary in the prototype since all control and input/output could be achieved through the host processor access to Harmoniac's memory. Some type of programmed or interrupt driven input/output would be essential in a stand-alone signal processor. In the prototype implementation the program and data are loaded via the DMA channel and any suitable locations in main memory are used as flags to tell Harmoniac to begin a process and to tell the host processor that a given stage is complete. (See section 4.3.1 and appendix III for examples.) For some applications an interrupt to the host is an advantage and this is provided.

A detailed circuit diagram of the DMA section is shown in Fig. 10 and its position in the chassis in Fig. 5 can be seen halfway up the right column of chips (bay 3). A total of forty-three chips is used in this section. A part of the DMA circuit is included in Fig. 14 (chips 86-99 and 133-135). The DMA signals to and from the host enter the chassis in four sixteen-core flat cables which plug into the chip-array socket positions 90, 99, 100 and 101. They are respectively, input data (to Harmoniac), output data, address for Harmoniac and the strobes and flags for timing the data transfer.

When a DMA transfer is requested by the host, the next instruction cycle is halted in $\Phi 1$ (phase 1) via DPEND (9/108 in Fig. 10 top RH) and chip 359 (Fig. 6 bottom GH). The "hold state" freezes the main timer (chips 351, 352) but allows the synchronised

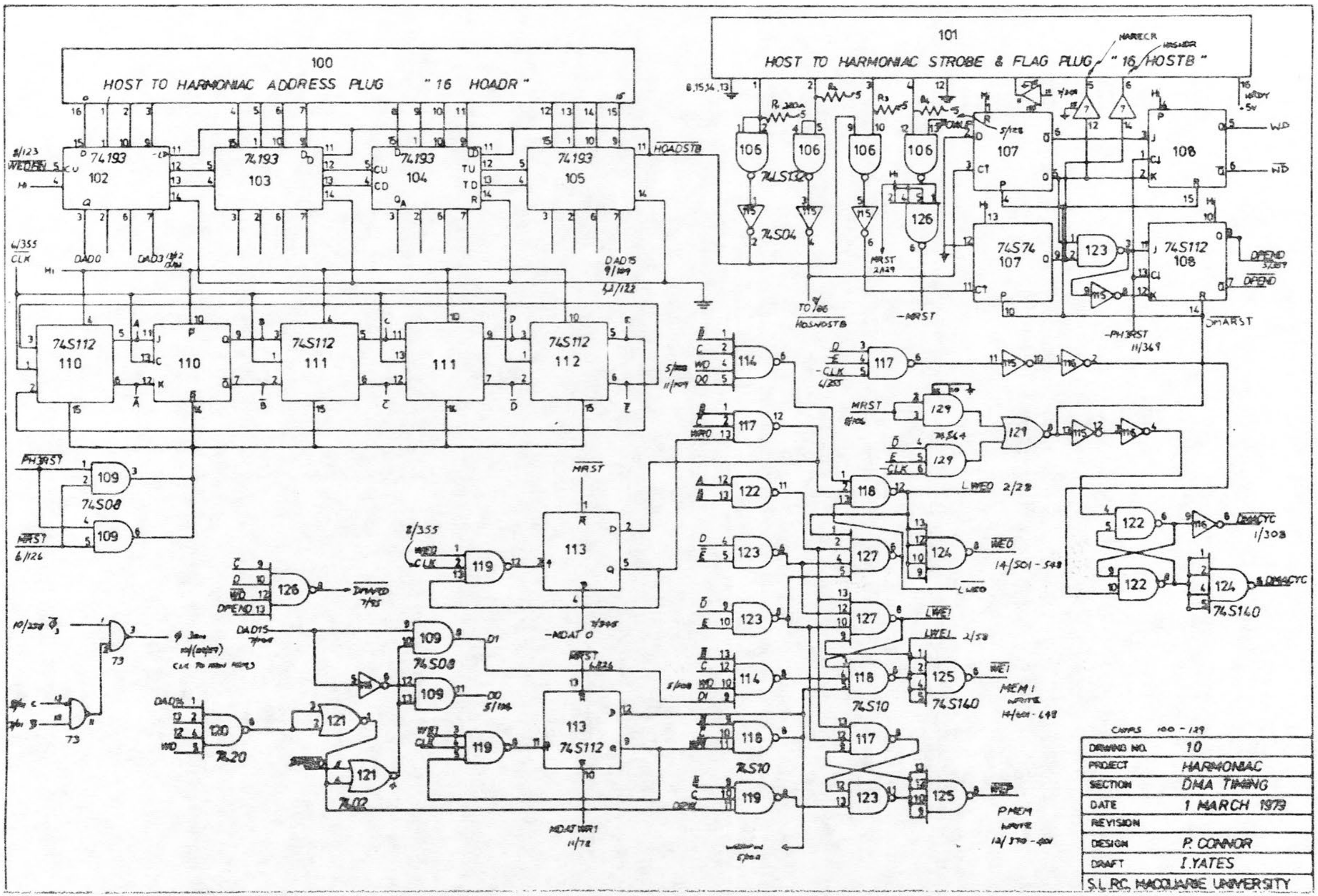

100
HOST TO HARMONIAC ADDRESS PLUG "16 HOADR"
101
HOST TO HARMONIAC STROBE & FLAG PLUG "16 HOSTB"
CHIPS 100 - 129
DRAWING NO. 10
PROJECT HARMONIAC
SECTION DMA TIMING
DATE 1 MARCH 1979
REVISION
DESIGN P. CONNOR
DRAFT I. YATES
S.L.R.C. MACQUARIE UNIVERSITY

Fig. 10.

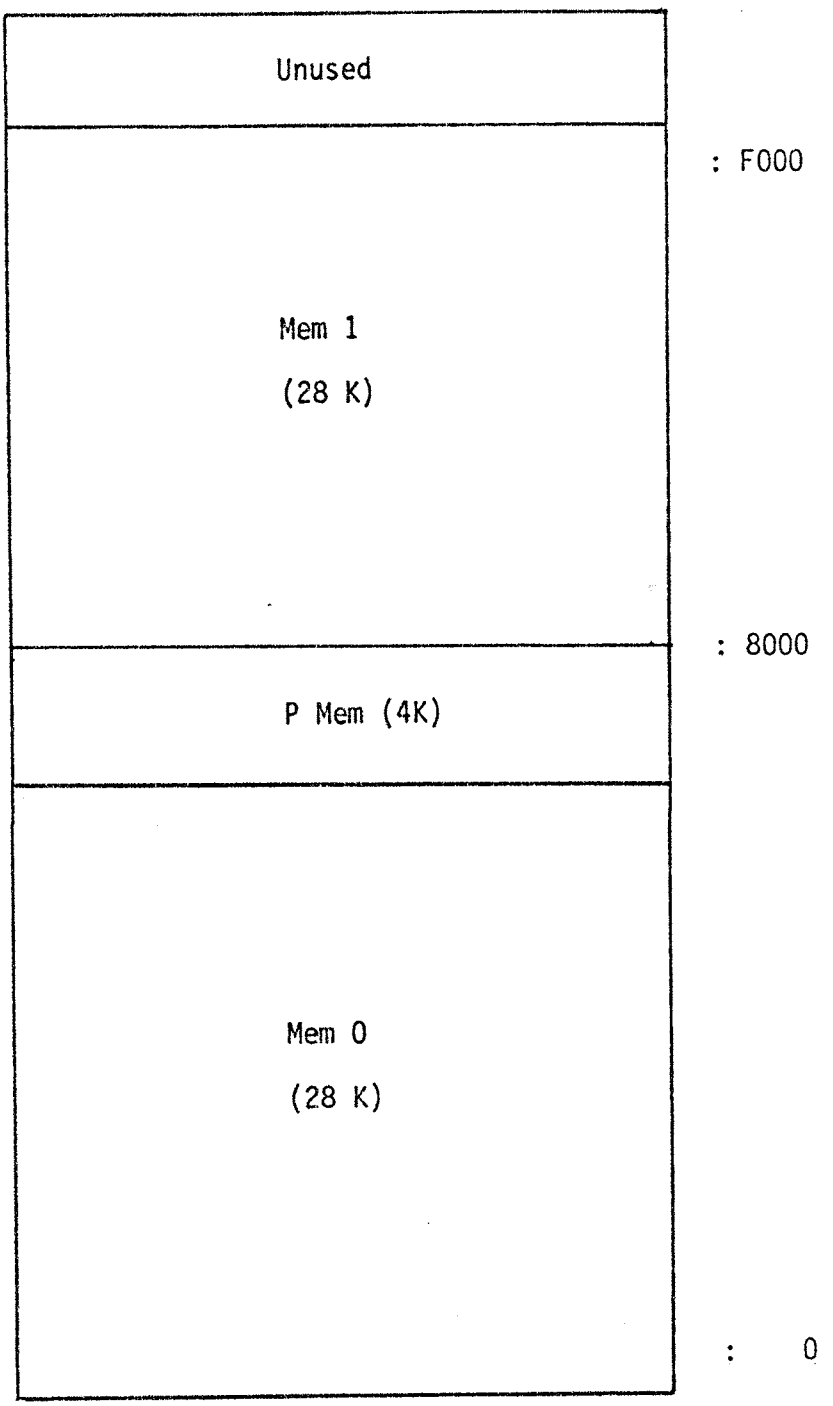


Fig. 21

DMA timer (chips 110, 111, 112) to continue. The memory write enables are generated from this DMA timer, so any writes underway to memory are completed. After the end of the normal phase 3, an extra seven phases are counted by the DMA timer to execute the DMA transfer requested.

All the memories are switched to the DMA address for the duration of the DMA cycle (DMACYC, 6/124, Fig. 10 bottom RH), but only the particular memory required is read or written. The top four bits of the DMA address select which memory is required. Fig. 21 shows the map of the address space as seen by the host.

Circuitry has been provided to read the contents of the program memory from the host although this is not essential for normal operation (chips 133-135 in Fig. 14 centre). This facility allows easy memory testing from the host.

The DMA address register is a counter so that only one address need be supplied to read or write any block of memory. When an address is received by Harmoniac, a DMA read is automatically executed at that address and the address is incremented ready for the next read. If the operation is to be a write to Harmoniac, the address supplied must be one less than the desired write address. The master reset pulse from the host (MRST-) initialises all flip flops in the machine and sets the program counter to zero. This can be used to execute a view program at any time by inserting a jump at location zero.

3.2.3 The Main Data Memories

Tow banks of three thousand (3K) words of 16 bits are installed in the prototype although each bank can be extended to 28K. The full memory-circuit diagrams are shown in Figs. 11, 12, 13 and 14. The memory-array circuit (Fig. 13) and the data-latch circuitry (Fig. 14)

each show chip numbering for both memories as these sections of the memories are identical except for the connection to pin 7 on chip 63/77 etc. (Bottom left of Fig. 14.) The memory arrays and support circuitry are located in the top and bottom of bay 3 (right column) and some of bay 2 in the chip-position diagram (Fig. 5). One hundred and eighty-four chips are used in the memory section, most of them 1024 bit static RAMS - either TTL 93425 or VMOS 2125 types.

The address and data inputs of the memories are latches as in any other operation, but the address latch is a counter and a multiplexer is provided on the address so that count up, count down and reverse-bit addressing can be achieved by a simple mode change (see Figs. 11 and 12). The bit reversal is achieved by a set of plug in jumpers on sixteen-pin headers (chips 9, 10, 39, 40). This feature is used in certain FFT algorithms. For a 1024 point transform the first ten bits of the address ($0 \rightarrow 9$) are transposed ($9 \rightarrow 0$) and the other bits connected normally so that each one K (1024 point) page is in bit-reverse order but the pages are in normal order. The address-mode change multiplexer is also used during a DMA cycle to drive the memory arrays with the DMA address. The current memory addresses are buffered onto the bus at locations in the operations field so that the addresses can be tested when the memories are in an auto-incrementing or decrementing mode.

Output data from the memory array is not latched as this would slow operation. A set of tristate buffers is provided to isolate the memory array from the bus in both memories.

To provide single instruction transfers of data and address to memory, a special address port is provided on memory one (see app. I, "MID/AD") which allows immediate mode to generate the address on bus zero while the data is transferred into memory via bus one.

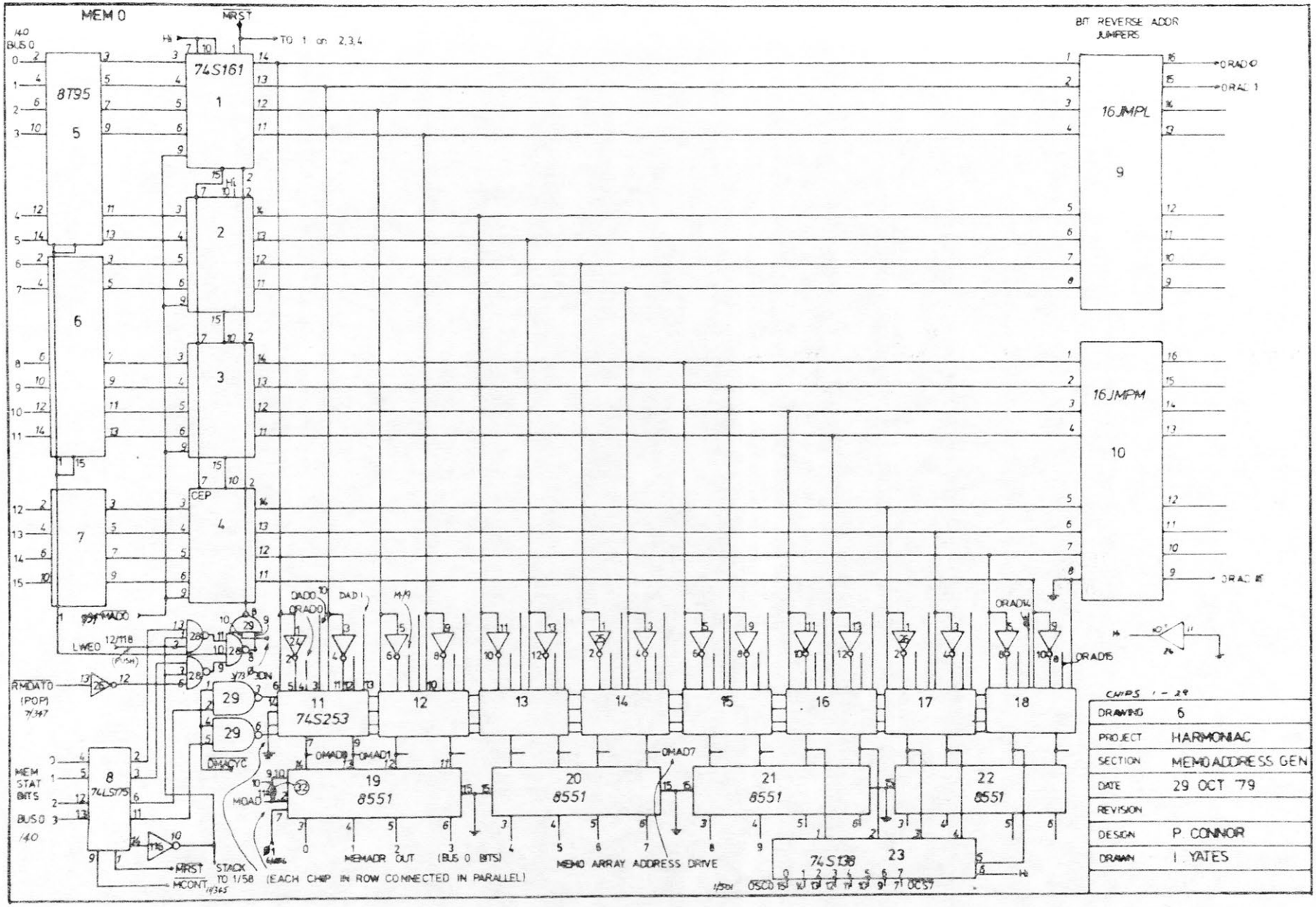

MEM 0
MRST
TO 1 on 2,3,4
BIT REVERSE ADDR JUMPERS
74S161
8T95
16JMPL
16JMPM
74S253
74LS175
74S138
8551
MEMADR OUT (BUS 0 BITS)
MEMO ARRAY ADDRESS DRIVE
STACK TO 1/58 (EACH CHIP IN ROW CONNECTED IN PARALLEL)
CHIPS 1 - 29
DRAWING 6
PROJECT HARMONIAC
SECTION MEMO ADDRESS GEN
DATE 29 OCT '79
REVISION
DESIGN P. CONNOR
DRAWN I. YATES

Fig. 11

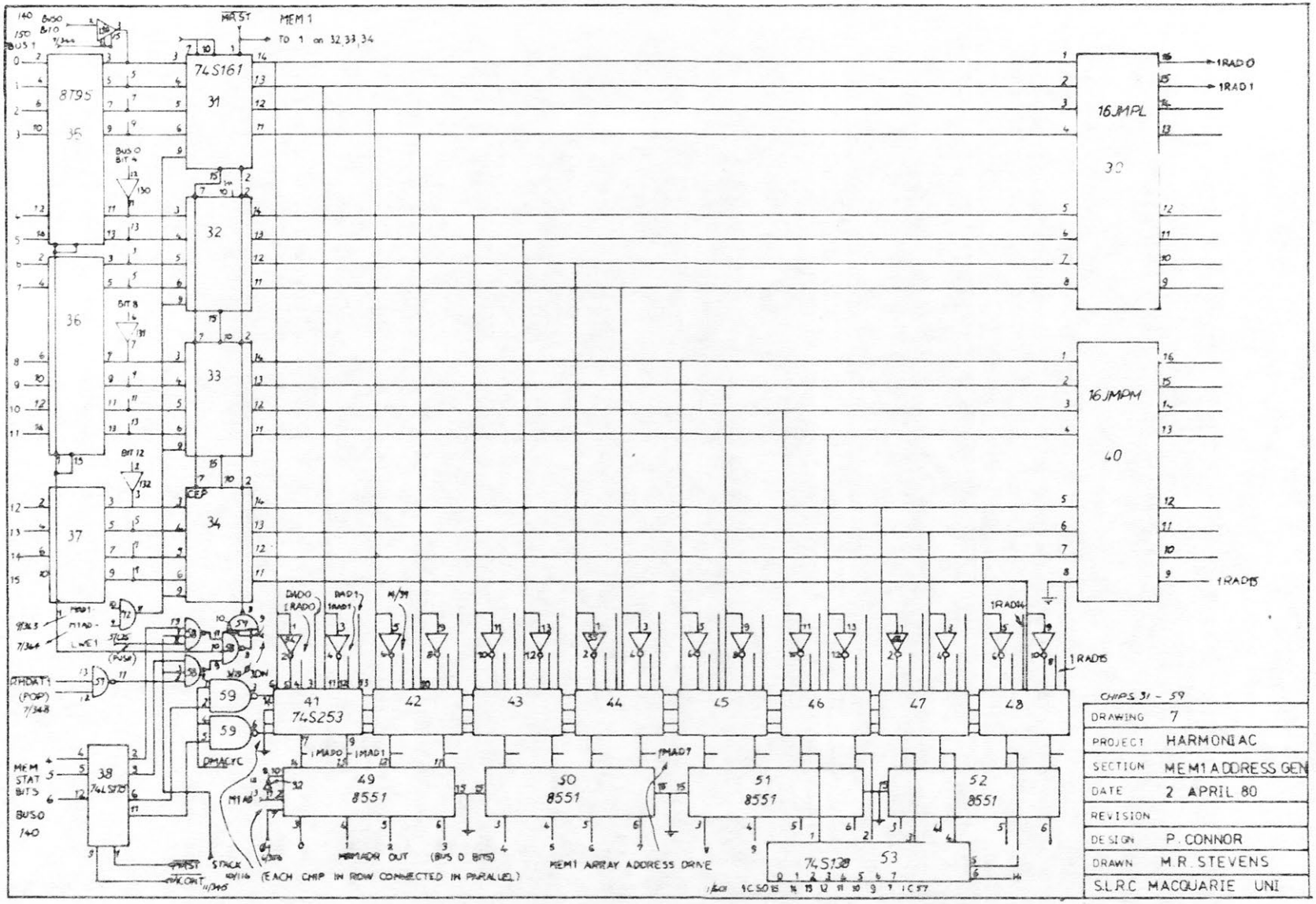

MEM 1
TO 1 on 32,33,34
74S161
31
8795
35
32
36
33
37
34
16JMPL
30
16JMPM
40
1RAD0
1RAD1
1RAD15
41
74S253
42
43
44
45
46
47
48
49
8551
50
8551
51
8551
52
8551
53
74S138
38
74LS151
59
59
DMACYC
MEM STAT BITS
BUS0
140
MEM1 ARRAY ADDRESS DRIVE
(EACH CHIP IN ROW CONNECTED IN PARALLEL)
CHIPS 31 - 59
DRAWING 7
PROJECT HARMONIAC
SECTION MEM1 ADDRESS GEN
DATE 2 APRIL 80
REVISION
DESIGN P. CONNOR
DRAWN M.R. STEVENS
S.L.R.C MACQUARIE UNI

Fig. 12.

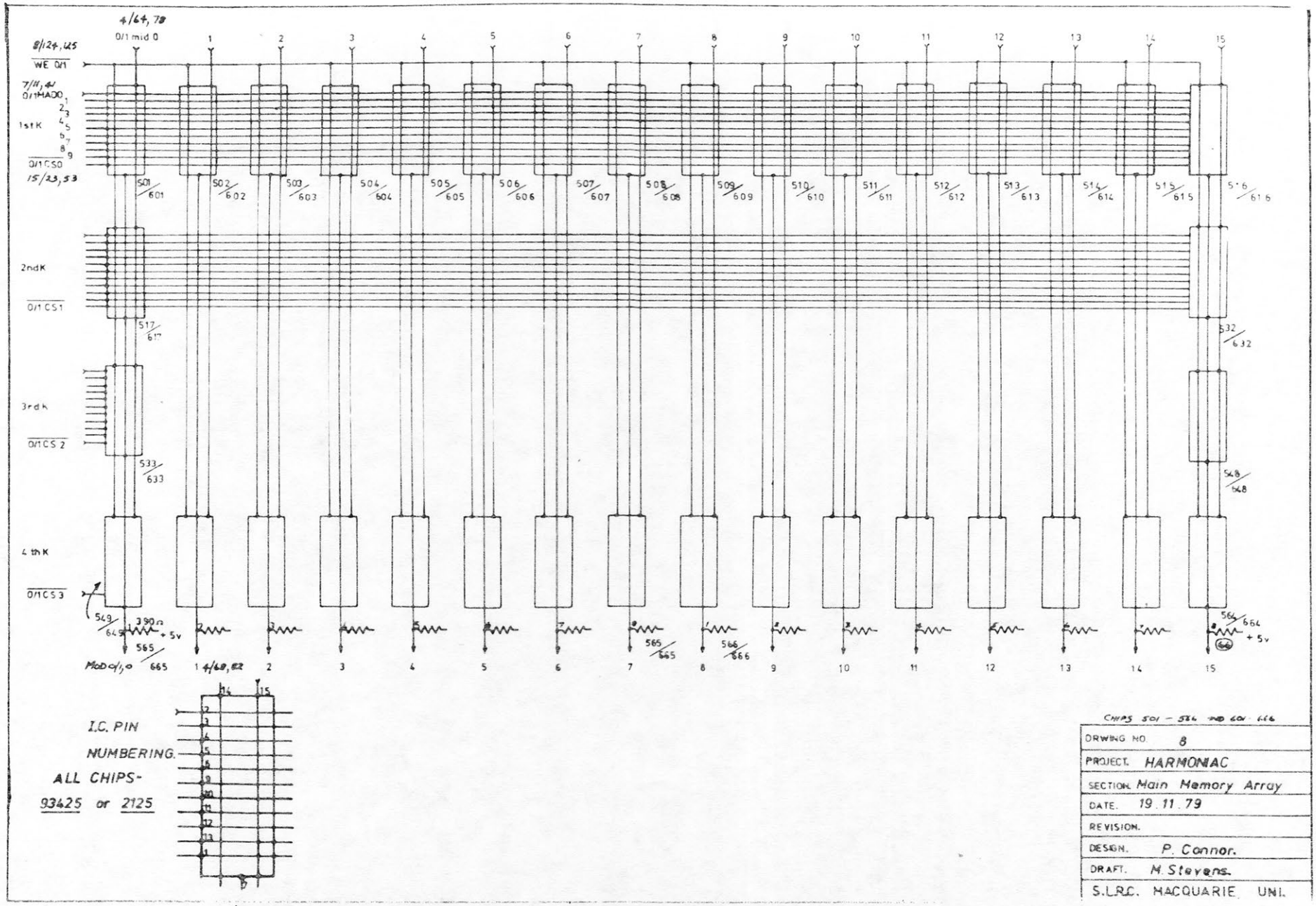

4/64, 78
0/1 mid 0
8/124, 125
WE 0/1
7/11, 41
0/1 MADD
1st K
0/1 CS0
15/23, 53
2nd K
0/1 CS1
3rd K
0/1 CS 2
4 th K
0/1 CS 3
390Ω
+ 5v
Mod 1, 0
4/60, 62
I.C. PIN
NUMBERING.
ALL CHIPS-
93425 or 2125
CHIPS 501 – 536 AND 601 – 636
DRWING NO. 8
PROJECT. HARMONIAC
SECTION. Main Memory Array
DATE. 19.11.79
REVISION.
DESIGN. P. Connor.
DRAFT. M. Stevens.
S.L.R.C. MACQUARIE UNI.

Fig. 13.

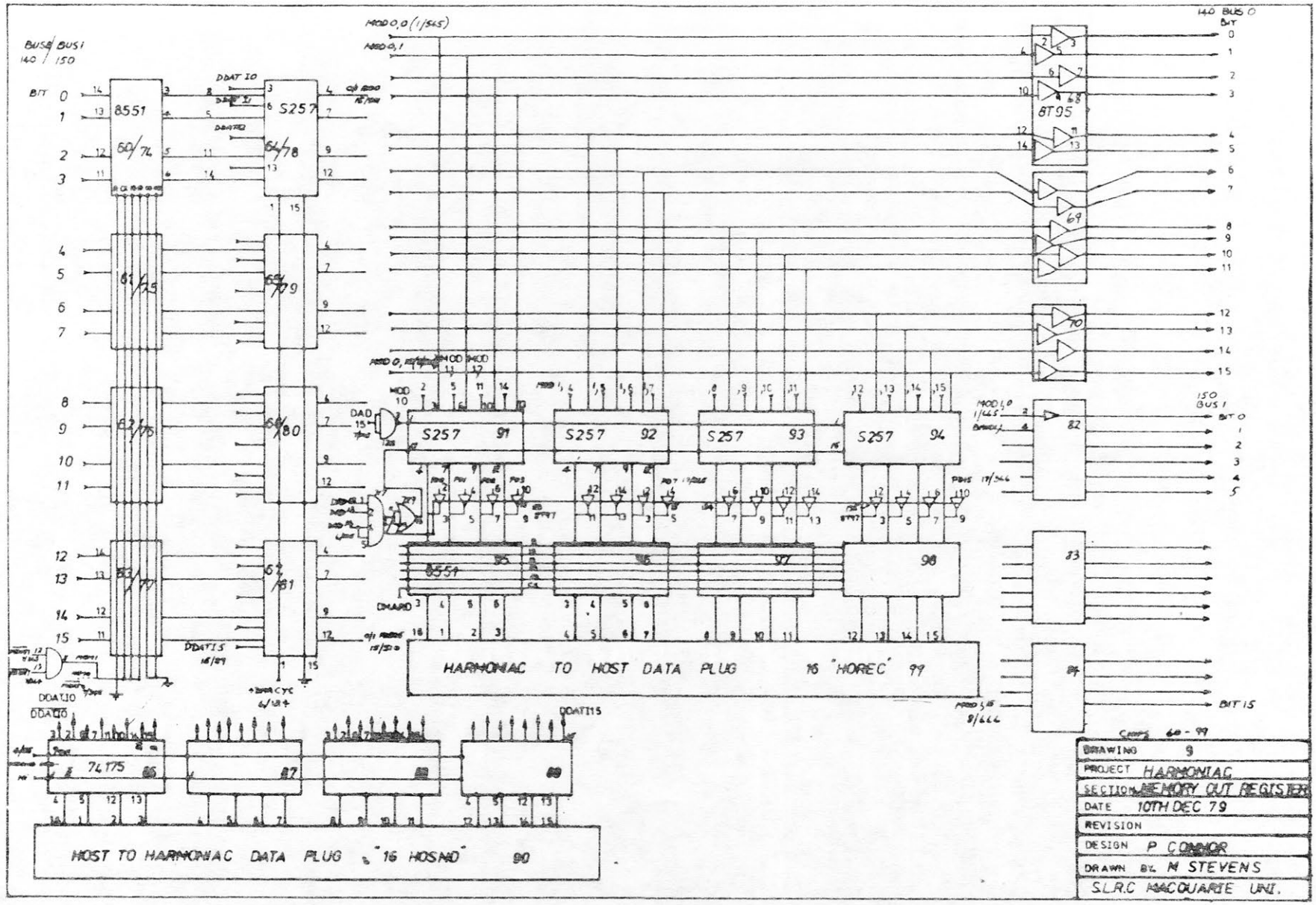

BUS0/BUS1
140 / 150
BIT
8551
S257
8T95
S257 91
S257 92
S257 93
S257 94
8551
HARMONIAC TO HOST DATA PLUG 16 "HOREC" 99
74175
HOST TO HARMONIAC DATA PLUG "16 HOSND" 90
140 BUS 0
150 BUS 1
BIT 15
DRAWING 9
PROJECT HARMONIAC
SECTION MEMORY OUT REGISTER
DATE 10TH DEC 79
REVISION
DESIGN P CONNOR
DRAWN BY N STEVENS
S.L.R.C MACQUARIE UNI.

Fig. 14

3.2.4 The Arithmetic and Logic Unit

This section does adds, subtracts, "and", "or" and comparison tests all in two's complement arithmetic. The circuit is given in Fig. 15. The data is inverted by the input latches and output tristate buffers. The 74S181 medium-scale integration ALU chips are operating on inverted data and producing inverted results.

The ALU is an unusual section in that its input takes up five locations in the operations address field, but it has only one address for output (see appendix I). The output is available at the same address on both buses. This is extremely convenient when programming as the results of simple arithmetic are often required on either bus (or both) for subsequent operations.

The comparison can be in a variety of modes: equal, not equal, greater than (>), less than (<) and greater than or equal to ($\geq$) and the effect of a comparison instruction is to cause a skip of the following instruction if the condition specified is true. This is achieved by disabling the destination demultiplexers in the control section during phase 3 so that the destination registers are not clocked. The propagation delays which occur during comparison seem to be the main speed limitation in Harmoniac and could possibly be improved. The equals test uses the open collector "wired or" - the only critical resistive pull up.

An overflow flag and a half-scale flag are provided. The half-scale flag is set when an ALU operation makes the result bits 14 and 15 different, indicating that overflow is imminent. These flags can be read on the bus as bit zero of a location called AFLG (see app. I). The flag to be read is selected by an arithmetic

mode change using chip 723. The mode change also sets the comparison mode and resets the overflow (OV) and half scale (HS) flags if required. The top bits AFLG are always zero (see 715, 732, 733, 730 Fig. 15).

There would be some speed advantage to be gained by using separate hardware for comparison so that the contents of the ALU were not destroyed when testing for the end of a loop in a program. This option was not chosen because of space limitations in the prototype chassis.

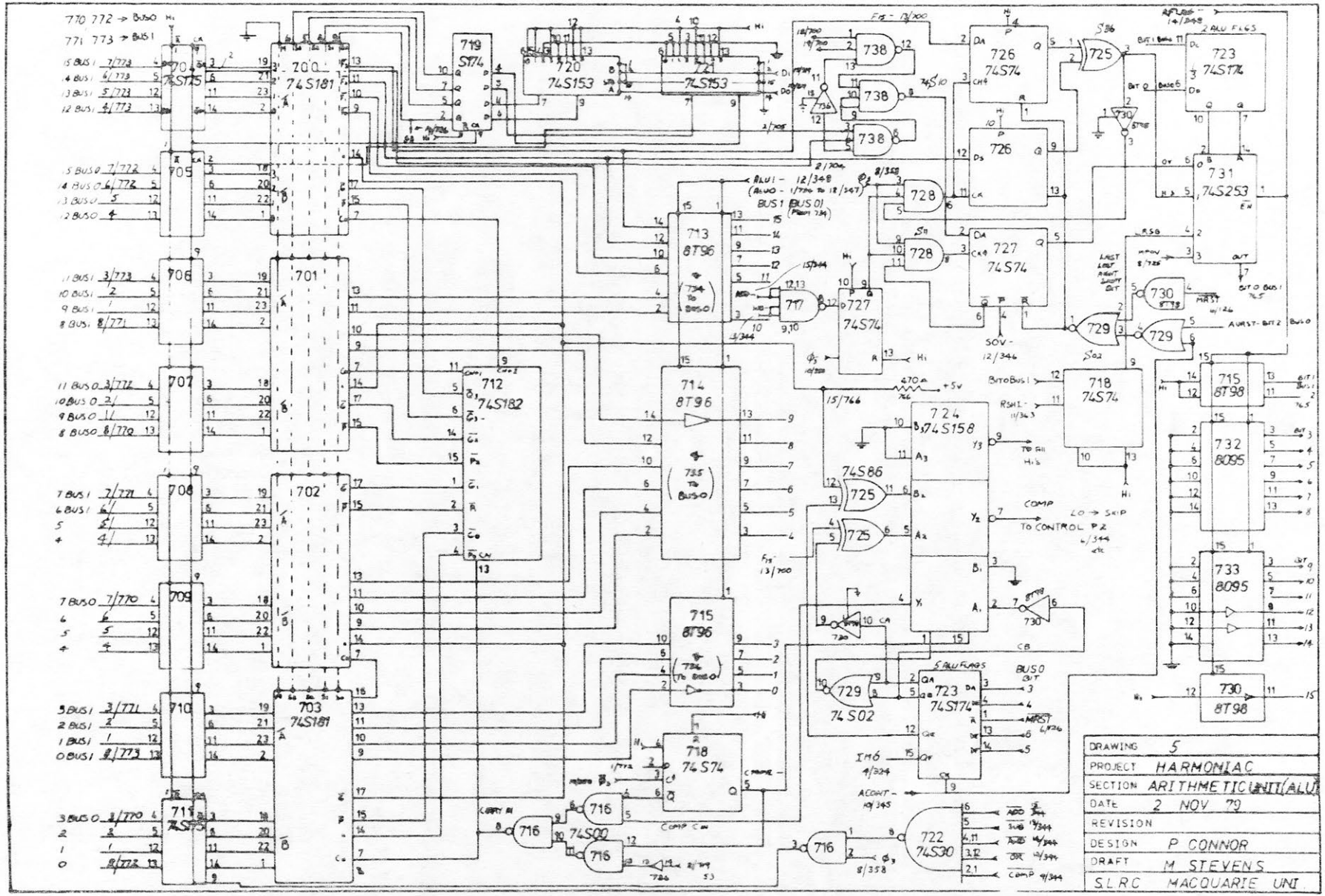

DRAWING 5
PROJECT HARMONIAC
SECTION ARITHMETIC UNIT (ALU)
DATE 2 NOV 79
REVISION
DESIGN P CONNOR
DRAFT M STEVENS
SLRC MACQUARIE UNI

Fig. 15

3.2.5 Transfer and Right Shift Section

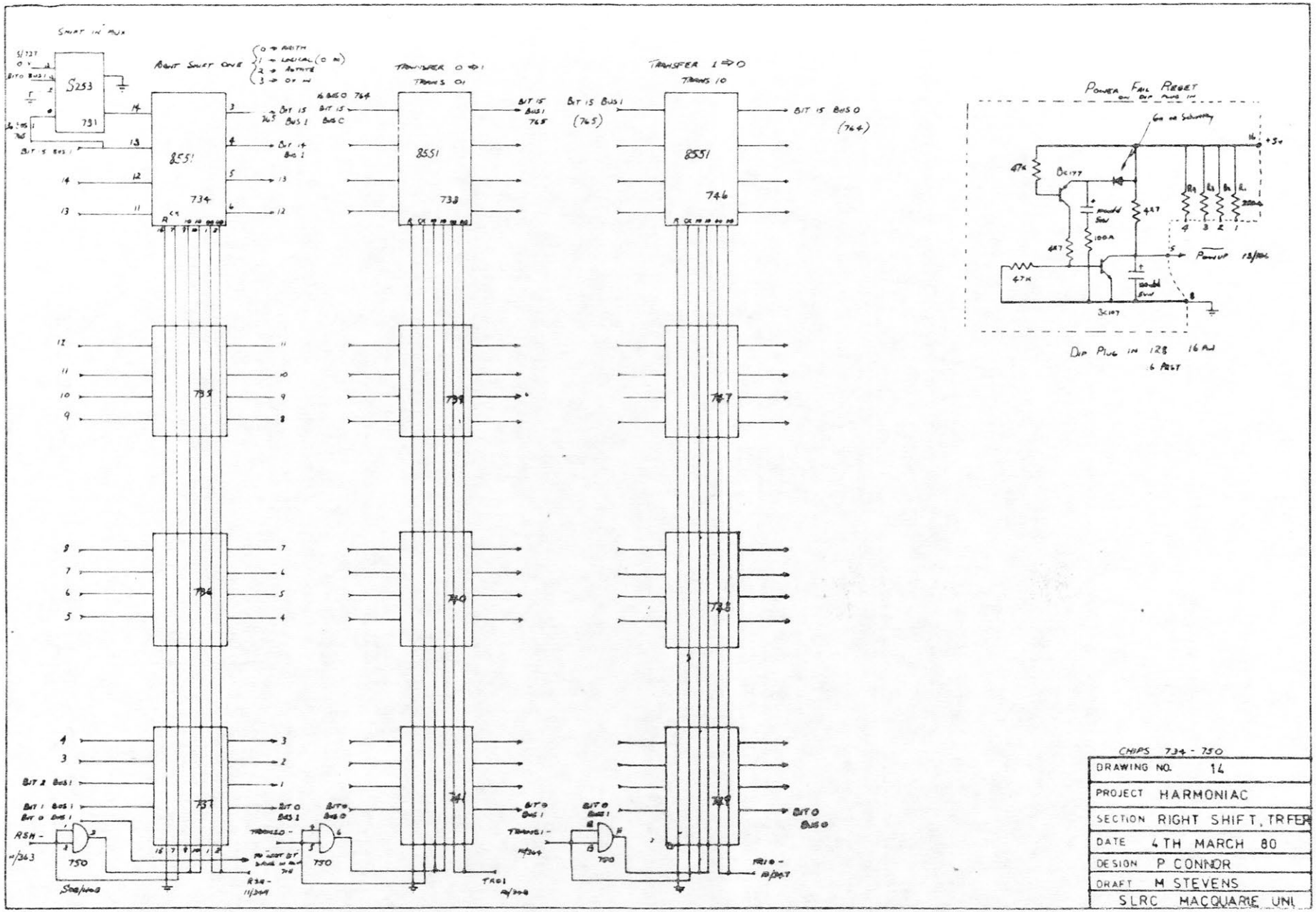
The use of two independent buses creates the necessity for transfers of data between locations which are on different buses. This could be done through the ALU but this would waste time in certain situations where the ALU is holding an intermediate result, so separate tristate latch/buffers have been provided to transfer in each direction.

Simple scaling of data by arithmetic right shifts is a common requirement in signal processing. For this purpose a single-place right shift register has been provided. It can be set up to shift into the most significant bit (MSB) either of: the previous MSB (arithmetic shift), zero (logical shift), the previous least significant bit LSB (rotate) or the overflow bit. A full circuit diagram for this section is shown in Fig. 16. It uses a latch which has the output bits wired one further up on the bus relative to the input bits.

3.2.6 The Sine Table Memory

For simplicity in stand-alone applications and speed in host-assisted applications a separate read-only memory (ROM) sine table has been provided (Fig. 17). This is addressed by sixteen bits (12 bit accurate) and produces a sixteen-bit result (accurate to 15 bits). It uses a commercial 1024 point by ten-bit quarter-cycle sine-table bipolar ROM (MMI 6068) combined with a set of four PROMS (DM8574) to give another 1024 points by four bits to provide a total of fourteen bits in ROM for each point on the quarter of the sine cycle.

The address and the table output are inverted as required to produce the full sine cycle from minus PI to plus PI where minus PI



CHIPS 734 - 750	
DRAWING NO.	14
PROJECT	HARMONIAC
SECTION	RIGHT SHIFT, TRFER
DATE	4TH MARCH 80
DESIGN	P CONNOR
DRAFT	M STEVENS
	SLRC MACQUARIE UNI

Fig. 16

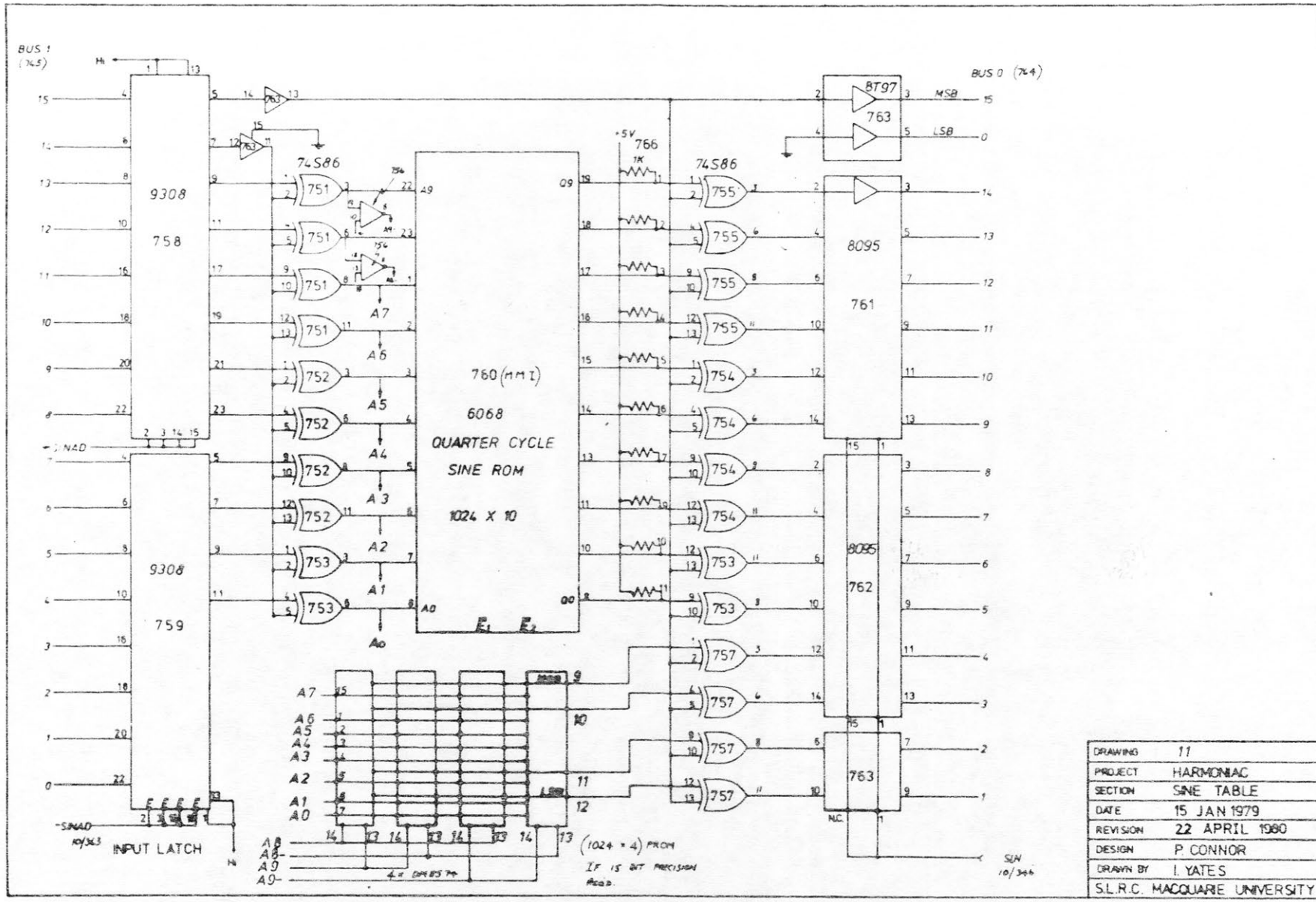

BUS 1 (765)
BUS 0 (764)
9308
758
9308
759
INPUT LATCH
74S86
751
752
753
754
755
757
760 (HMI)
6068
QUARTER CYCLE
SINE ROM
1024 X 10
766
1K
BT97
763
MSB
LSB
8095
761
8095
762
763
A7
A6
A5
A4
A3
A2
A1
A0
(1024 × 4) FROM
IF 15 BIT PRECISION
REQD.
DRAWING 11
PROJECT HARMONIAC
SECTION SINE TABLE
DATE 15 JAN 1979
REVISION 22 APRIL 1980
DESIGN P. CONNOR
DRAWN BY I. YATES
S.L.R.C. MACQUARIE UNIVERSITY

Fig. 17.

corresponds to an input of -32768 and plus PI corresponds to an input of +32767. The resultant output is scaled so that plus one corresponds to +32767 and minus one corresponds to -32768. This scaling given maximum precision and simplifies both hardware and programming. The maximum access time of the sine ROM is approximately two hundred nseconds with its associated logic, so an extra instruction time should be allowed before using the result of a sine table look up (PI = 3.1412 Radians).

3.2.7 The Multiplier

Multiplication is heavily used in most signal-processing tasks so a high-speed multiplier is necessary if fast processing rates are to be achieved. For this purpose a full two's complement 16 x 16 bit-array multiplier has been used in Harmoniac. Maximum delay through the multiplier is just over 200 nanoseconds, so a one-instruction delay is necessary for reliable results although in practice no delay was required in the prototype. Although single-chip multipliers are now available in 16 bit x 16 bit sizes, the prototype has used an array of 4 bit x 2 bit chips (93 S 43) that were available when it was under construction in 1976. Circuit diagrams of the array and the input/output latches are shown in Figs. 18 and 19 respectively. The chips implement the Booth-McSorley algorithm.

A double-precision result is produced with the high-order bits available on bus one and the low-order bits on bus zero. Signle-precision multiply and accumulate is provided via chips 253 and 254 (Fig. 18). An extra function, a long logical left shift (double precision) is included in the tristate output bus drivers by the use of four-bit multiplexers (74 S 257) in chips

243 to 250. This left shift is used in division algorithms and the other successive approximation tasks which need a double-precision left shift.

3.2.8 The Power Supply

As mentioned previously in 3.1 the regulator section is mounted on the logic chassis and the transformer, rectifier, electrolytics and circuit breaker are mounted on a separate chassis with two metre cables between the two chassis. A micro-switch is provided on the lid of the logic chassis to cut the main supply (240 volt ac) if the lid is opened, thus preventing overheating caused by low air circulation.

The use of one logic family (STTL) exclusively in Harmoniac allows a single five-volt regulated supply. Current drawn in the prototype was approximately twenty-seven amps at five volts. A conventional series regulator was used for simplicity. No over-current shut down was provided, except for a D.C. circuit breaker just before the regulator (see Fig. 20 for details). Overvoltage and overtemperature protection are provided with a zener and a thermal bimetallic switch being used to trigger a large silicon-controlled rectifier (SCR) if either condition occurs. The SCR is mounted on the regulator heatsink and, once fired, it short circuits the unregulated voltage until the circuit breaker (or the mains fuse) opens.

There would be a considerable power saving if a switching power supply was used, but this was not done in the prototype because of cost considerations.

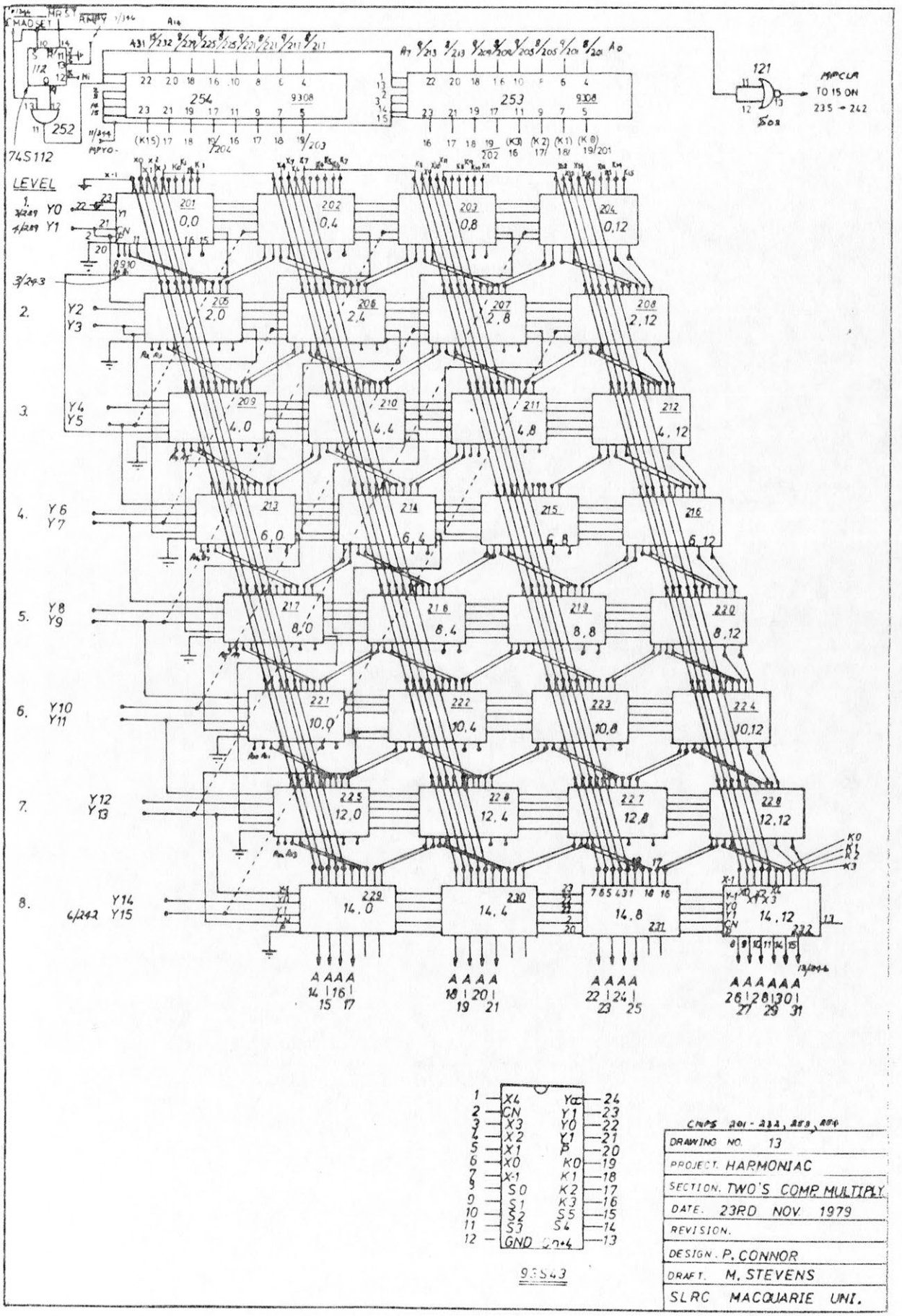

74S112
LEVEL
1.
2.
3.
4.
5.
6.
7.
8.
201 0,0
202 0,4
203 0,8
204 0,12
205 2,0
206 2,4
207 2,8
208 2,12
209 4,0
210 4,4
211 4,8
212 4,12
213 6,0
214 6,4
215 6,8
216 6,12
217 8,0
218 8,4
219 8,8
220 8,12
221 10,0
222 10,4
223 10,8
224 10,12
225 12,0
226 12,4
227 12,8
228 12,12
229 14,0
230 14,4
231 14,8
232 14,12
AAAA 14 16 15 17
AAAA 18 20 19 21
AAAA 22 24 23 25
AAAAAA 26 28 30 27 29 31
TO 15 ON 235 → 242
X4 CN X3 X2 X1 X0 X-1 S0 S1 S2 S3 GND
Y0 Y1 K0 K1 K2 K3 S5 S4
9S43
CHIPS 201 - 232, 253, 254
DRAWING NO. 13
PROJECT. HARMONIAC
SECTION. TWO'S COMP MULTIPLY
DATE. 23RD NOV 1979
REVISION.
DESIGN. P. CONNOR
DRAFT. M. STEVENS
SLRC MACQUARIE UNI.

Fig. 18.

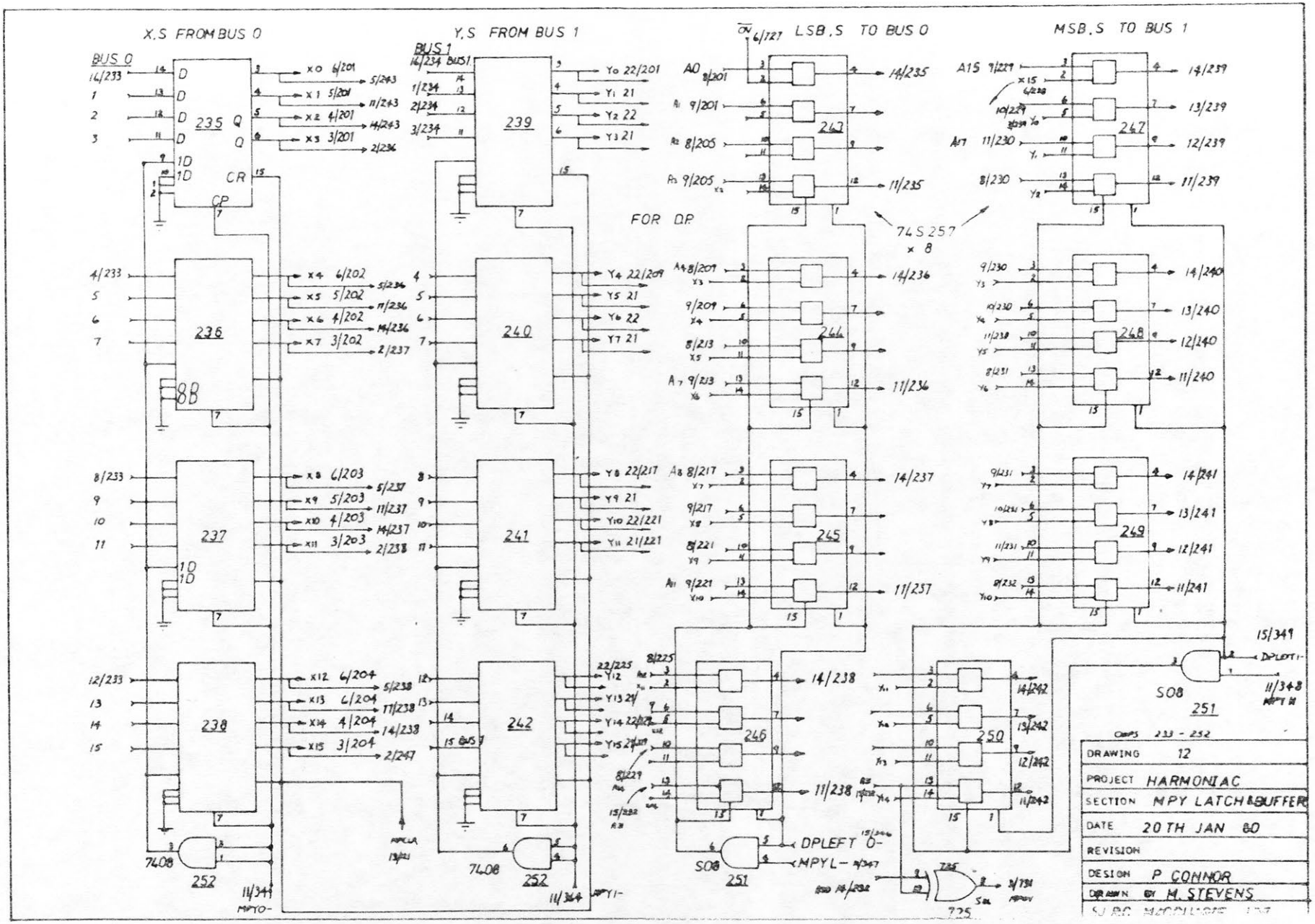

X.S FROM BUS 0
Y.S FROM BUS 1
LSB.S TO BUS 0
MSB.S TO BUS 1
BUS 0
BUS 1
235
239
247
267
236
240
244
268
237
241
245
249
238
242
246
250
251
252
74S257
× 8
FOR DP
SO8
7408
DPLEFT 0-
MPYL-
DPLEFT 1-
CR
CP
CHIPS 233 - 252
DRAWING 12
PROJECT HARMONIAC
SECTION MPY LATCH & BUFFER
DATE 20TH JAN 80
REVISION
DESIGN P CONNOR
DRAWN BY M. STEVENS

Fig. 19.

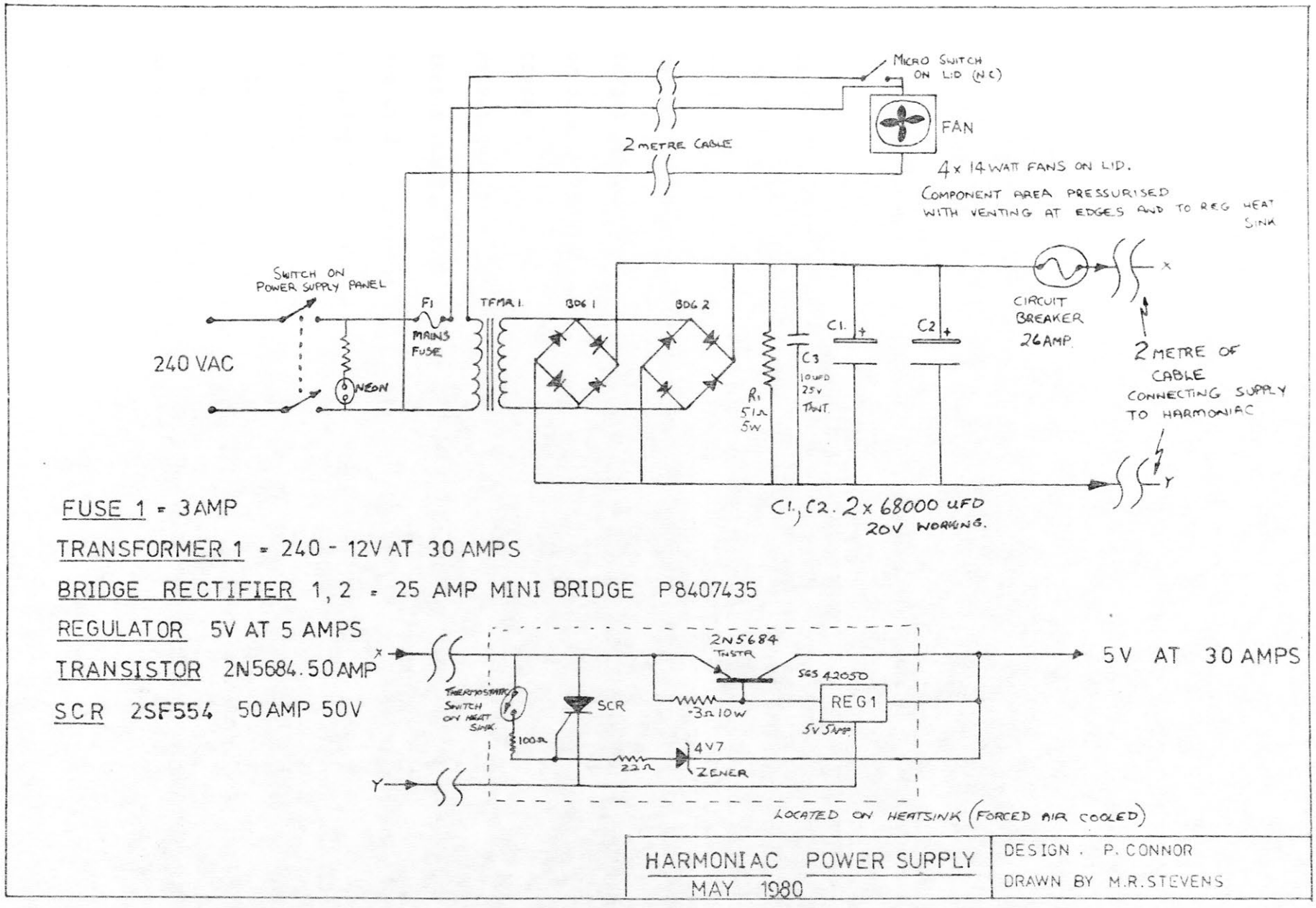

MICRO SWITCH ON LID (N.C)
FAN
2 METRE CABLE
4 x 14 WATT FANS ON LID.
COMPONENT AREA PRESSURISED WITH VENTING AT EDGES AND TO REG HEAT SINK
SWITCH ON POWER SUPPLY PANEL
240 VAC
NEON
F1
MAINS FUSE
TFMR 1.
BDG 1
BDG 2
R1 51Ω 5W
C3 10 UFD 25V TANT.
C1.
C2
CIRCUIT BREAKER 26AMP.
2 METRE OF CABLE CONNECTING SUPPLY TO HARMONIAC
X
Y
C1, C2. 2 x 68000 UFD 20V WORKING.
FUSE 1 = 3AMP
TRANSFORMER 1 = 240 - 12V AT 30 AMPS
BRIDGE RECTIFIER 1, 2 = 25 AMP MINI BRIDGE P8407435
REGULATOR 5V AT 5 AMPS
TRANSISTOR 2N5684. 50AMP
SCR 2SF554 50AMP 50V
2N5684 THSTR
SGS 42050
REG 1
5V 5AMP
SCR
THERMOSTAT SWITCH ON HEAT SINK.
100Ω
·3Ω 10W
22Ω
4V7 ZENER
5V AT 30 AMPS
LOCATED ON HEATSINK (FORCED AIR COOLED)
HARMONIAC POWER SUPPLY
MAY 1980
DESIGN . P. CONNOR
DRAWN BY M.R. STEVENS

Fig. 20.

APPLICATIONS AND SOFTWARE

4.1 Applications

Basically applications for such a processor are either real time or non-real time applications. Most stand-alone work would require real-time operation to avoid build-up of unprocessed data, whereas more flexibility is possible in a system supported by a host processor with fast mass storage like a hard disc (eg. a 16-bit microprocessor with a "Winchester"-type disc.

4.1.1 Real Time

The limits on real-time operation are easy to determine. The capacity depends on the bandwidth of the signal to be processed, the instruction rate, and the complexity of the algorithm. The maximum instruction rate in the prototype is 6.6 MHz or 150 nano-seconds per complete double word transfer and operation. A typical target bandwidth for high-quality audio processing might be 15 KHz or 33 microseconds per sample processed. Each sample must be processed synchronously with the external crystal clock of the analog data-acquisition system. It was found in a real-time sinusoid synthesis routine (Fig. 22 in 4.3.1) that approximately ten instructions are used in synchronisation by polling of a memory location (used as a timing flag from the host processor). Another twenty or thirty instructions are usually required to set up loops, so that the inner loop of the signal processing algorithm is limited to about 160 instruction times total for high-quality sound. This is sufficient for many useful algorithms (see 4.3.1) but not every possibility. Digital filters, sine-wave summation, linear predictive processes, and FFT analysis are all possible within bandwidth and complexity restrictions. Processes such as the FFT which require a buffer full of

samples can easily be double buffered to provide continuous processing. A fast disc is not usually necessary if real-time processing is possible.

4.1.2 Non Real-Time Applications

There are no fundamental limitations on algorithms which do not need to be executed within a specified time interval but many would require a large program memory space and hence become impractical. Another aspect to be considered is the programming effort required to implement the wide range of functions (such as floating-point operations) which may be required in the more complex algorithms. It would usually be simpler, and almost as fast, to implement algorithms with complex requirements on a machine with a higher level language. The type of non real-time algorithm which might be expected to be suitable to run on Harmoniac would be one using integer operations, but so many that they take a fair amount of time to complete (eg. a large or multiple FFT task, see 4.4).

4.1.3 The Appropriate Applications

The main advantage of this type of machine over a standard "black box" such as a spectrum analyser for rapid signal processing is that a more sophisticated, tailored algorithm can be run at high speed. In some cases, such as plain spectral analysis, there is no need for a tailored algorithm and in other cases speed may not be important or cost no object. But there is a class of tasks which require more speed than a standard LSI microprocessor can provide but not the sophistication or cost of a "mainframe". The disadvantage of the Harmoniac structure is that software is a little harder to write than in a single-memory machine. This difficulty arises from the "handedness" of the machine. Many operations require a particular operand on the left-hand bus (bus 1) when it was deposited on the right-hand bus (bus 0) by the last instruction. Hence some care needs to be taken

in choosing the initial placing of operands in memory and their subsequent handling in the "scratch RAM" memories. With practice this is not very difficult.

4.2 The Assembler

To facilitate the writing of software for Harmoniac a two pass assembler has been written. It is written entirely in Fortran IV and runs on a Hewlett Packard 21 MX which is interfaced to Harmoniac. The assembler should be easily adapted to run on any Fortran system. A full listing of the assembler itself is given in appendix IV. It is part of a package which includes a simple file-handling system, editor and loader. The instruction mnemonic set and some of the rules in using it are given in appendix I ("Haref"). Typical examples of assembler listings of programs can be seen in section 4.3.1 and 4.3.2 which follow.

In the program body the left column is a destination (common for both bus destinations) followed by a source one (left bus) and a source zero (right bus). Labels for jump destinations and data addresses are indicated by a "#" character. At the branch point of a jump (or call) the label occurs in the third column while the destination of the jump is labelled in the fourth column. Immediate mode is automatically used for labelled jumps. A no-operation on a given bus is indicated by a blank for the source on that bus (or "NOP"). Labels for addresses in the data memories appear in the second column, after the contents of the location are specified. Reference to these labels must occur in the third column and immediate mode is used to generate the address (limit 127). When a label in any field is not defined as data or a jump it is assumed to be a location in "scratch RAM".

A new program origin is indicated by "@PM, n n" in the first column and a new data origin is indicated by "@M1, nn" where n n is a positive decimal number. Data values for the current memory location are given as decimal numbers between -32768 and +32767.

When the 16-bit instruction code (or data) has been assembled it is listed in terms of field addresses and as a four-digit hexadecimal number with a decimal and hexadecimal address, and the operation code is stored in a buffer. At the end of assembly, the buffer (object code) is transferred to disc or direct to Harmoniac. The listing can be suppressed if desired.

Many common errors are detected and flagged. These are listed in "HAREF" Appendix 1. A summary of the errors and warnings and the memory space used is listed at the end of assembly, in case no other listing is generated. The warnings are to assist in the use of immediate mode - to make the programmer aware that he is using a combination of addresses which cannot be used with the seven-bit immediate mode (which uses source one bit three). If the 7-bit mode is not in use, the instruction is valid.

Usually the scratch locations (16) are automatically allocated as the otherwise undefined variable destination labels are encountered, but it is possible to define a set of eight labels which become the first eight scratch labels. These do not use the third source bit and are thus always available for use.

The assembler takes source from disc in successive small files, each separately accessed by six character names. For a given assembly run only the first four characters of a name are used, the other two allowing separate editing of each of the small files. All files

assembled in one run must have the same first four characters. Hence a kind of linked assembly of several files is possible.

Results of an assembly are optionally sent direct to Harmoniac or placed in a disc file for later use.

4.2.1 Debug Utility "HBUG"

Since Harmoniac has no front-panel controls, all control must be exercised by software, using the direct memory-access port. To enable hardware and software verification a de-bugging program called "HBUG" was written to run on the host processor. The initial version of this runs on the ALPA-16 processor and was written in machine language, communicating via the TTY.

"HBUG" has facilities to inspect and change ("I") any location of the program and data memories as well as search ("S"), fill ("F") and copy ("C") facilities. These can be used for simple memory testing, using the "search for not equal" command (S nn.mm.v N") after the memory has been filled from nn to mm with value v with a "F" command. The instruction registers and scratch RAM registers are not directly accessible via the DMA port so a breakpoint subroutine was written for Harmoniac which copies all the registers into standard memory tables so that they can be communicated to the user. This facility is exercised using a "B" command which inserts calls to the breakpoint subroutine whenever they are needed. A few locations in each memory and a few registers in the scratchpad memory must be reserved for the breakpoint routine to operate smoothly.

4.2.2 "MTEST" Memory and Communications Tester

To verify that the DMA link to the host processor and Harmoniac's memory are in good working order, a simple diagnostic was written in

Fortran. This writes patterns into Harmoniac's memory and reports any discrepancies when they are read back. It does not use any of the processing circuitry in Harmoniac so it can be used as the first stage in isolating a fault. The patterns used are firstly fixed-bit patterns, then rotating patterns and then an incrementing binary number.

The program is called from disc in the Hewlett Packard 21MX host system.

4.3 Signal Processing Utilities Available for Harmoniac

This section describes some of the utilities written for Harmoniac which are now in use at the Speech and Language Research Centre.

4.3.1 Sine Wave Synthesis ("SINSUM")

A good general purpose utility for musical applications is a routine which can gnerate successive samples of a sum of sinusoids in real time. Such a utility is shown in Figs. 22, 24. It is based on a similar concept to the digital oscillator described in ref. 7. Parametric input is in a file of amplitudes and phase increments in memory one which can be changed at will by the host processor. Synchronisation with the sample rate set by the host is achieved through a location in memory defined as a flag. The host sets it to non-zero to initiate the processing for one sample and Harmoniac sets it back to zero when finished (see lines 70 to 75 for polling, lines 77 to 79 for reset, Fig. 22). Sample value is left in a predetermined memory location for the host to access. This routine produces 15 sinewaves with a 15 KHz bandwidth. There are several deficiencies in this routine which are corrected in a more advanced version shown in Fig. 23.

The version in Fig. 23 uses double-precision phase angles to achieve 2 ** 31 points in frequency over a 15 KHz range and it changes parameters only at a zero crossing so that no discontinuities are heard. It is used in the singing-voice resynthesis system described in 4.4 and appendix III. It uses a buffer to store a large number of samples and is not oriented towards real time operation (parameters are changed at every zero crossing for program simplicity).

A close look at the single-precision sine-synthesis routine in Figs. 24, 22 will clarify some programming techniques. A table of the phases of each sine wave starts at memory zero location zero. The phase increments and amplitudes are stored in a coefficients table in memory one starting at location 121. Overall amplitude is stored at memory one, location 120. The result and flag are at 100 and 101. The arithmetic control word is set for seven-bit immediate mode at location one of the program. The memory control word is set (at location zero) to sine to pop memory one (increment on a read) and push memory zero (increment on a store - update of phase). Program memory locations are given in decimal in the RH column of the listing.

Fig. 22

```
PAGE    1   HARMONIAC ASSEMBLY OF : SINSUM  GEN N SINES
LINE ADDR MEM.      SOURCE CODE          #LABELS(JMP) *CMTS      DECODED OBJECT


 2  NEW PROGRAM SEGMENT BEGINS  @PM,500
 3 01F4 C574  JMP              #HERE   #HERE            IM   1   0 500    50
 4 01F5 0000  NOP                                            0   0   0    50
 5  NEW PROGRAM SEGMENT BEGINS  @PM,0
 6 0000 C574  JMP              #HERE                    IM   1   0 500
 7 0001 0000  NOP                                            0   0   0
 8  DATA FILE FOR MAIN MEM BEGINS  @M0,0
 9 0000 0000 0       #PHASES
10 0001 0000 0
11 0002 0000 0
12  DATA FILE FOR MAIN MEM BEGINS  @M1,120
13 0078 01F4 500     #AMPLA
14 0079 0064 100     #PHIAMP
15 007A 1770 6000
16 007B 00C8 200
17 007C 1770 6000
18 007D 012C 300
19 007E 1388 5000
20 007F 0190 400
21 0080 1388 5000
22 0081 01F4 500
23 0082 1388 5000
24 0083 0258 600
25 0084 1388 5000
26 0085 02BC 700
27 0086 0FA0 4000
28 0087 0320 800
29 0088 0BB8 3000
30 0089 0384 900
31 008A 07D0 2000
32 008B 03E8 1000
33 008C 03E8 1000
34 008D 044C 1100
35 008E 0320 800
36 008F 04B0 1200
37 0090 0258 600
38 0091 0514 1300
39 0092 01F4 500
40  DATA FILE FOR MAIN MEM BEGINS  @M1,100
41 0064 0000 0       #DATAD
42 0065 0000 0
43 0066 0000 0       #FLGAD
44  NEW PROGRAM SEGMENT BEGINS  @PM,0
45 0000 9009  RSH/MC =NOP    ,9                  *POP M1  IM   4   0   9
46  **1 PUSH M0
47 0001 9400  SIN/AC  NOP     0          SETUP ALU      IM   5   0   0
48 0002 FD19  M1D/AD  NOP     #PHIAMP #SAMLO            IM  15   0 121
49 0003 B420  TRANS   AFLG    0          A ZERO         IM  13   1   0
50 0004 9800  MADR    NOP     #PHASES                   IM   6   0   0
51 0005 40A5  #SAMPL  TRANS   TRANS                         16   5   5
52 0006 20E7  ADD     =MDAT   ,MDAT            *ADD PH IN    8   7   7
53  **C TO PH
54 0007 9460  SIN     =ALU    ,0      #SINLO            IM   5   3   0
55  * MPY SINE BY AMPL OF THIS COMPONENT(M1
```

Fig. 22

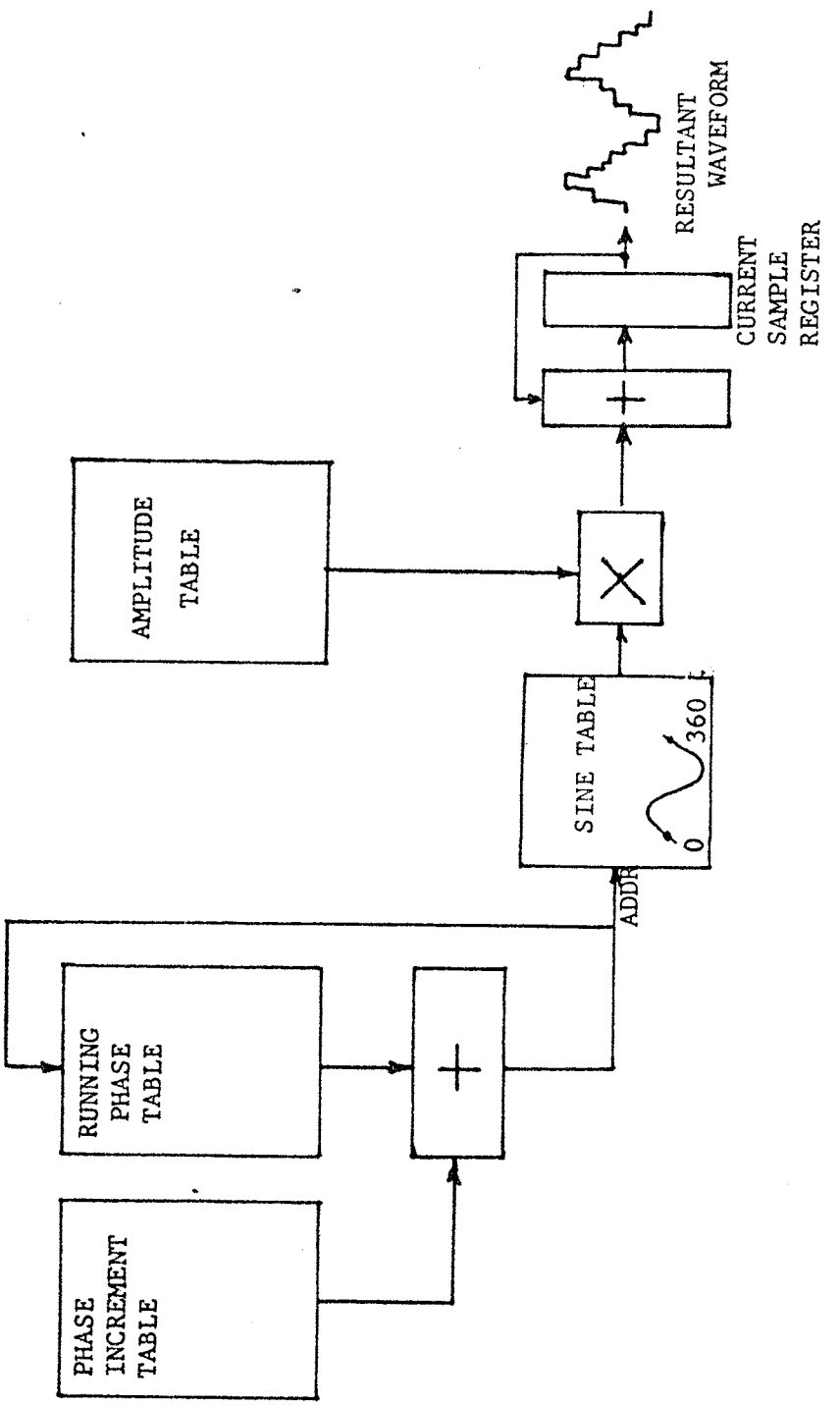
```
PAGE    2   HARMONIAC ASSEMBLY OF : SINSUM
 56 0008 30ED  MPY     =MDAT    ,SIN                              12   7  13
 57 0009 1C03  MDAT    =NOP     ,ALU          *DLY F               7   0   3
 58  **OR MPY
 59 000A 2090  ADD      MPYH     #SAMPL                           8   4  16
 60 000B 4063  #SAMPL   ALU      ALU                             16   3   3
 61 000C B84C  COMP     MOADDR   12                        IM    14   2  12
 62 000D 8407  JMP     =NOP     ,#SINLO       *DO MOIM           1   0   7
 63  **RE
 64 000E 20E7  ADD     =MDAT    ,MDAT         *JMP S             8   7   7
 65  **SHADOW
 66  * MPY BY OVERALL AMPL COEFT
 67 000F FD18  M1D/AD   NOP      #AMPLA                    IM    15   0 120
 68 0010 0000  NOP      NOP                                      0   0   0
 69 0011 30F0  MPY      MDAT     #SAMPL                          12   7  16
 70  *WAIT FOR HOST TO SET FLAG = 0(DATA REC
 71 0012 FD06  M1D/AD   NOP      #FLGAD  #HOSTWT           IM    15   0 102
 72 0013 B8E0  COMP     MDAT     0                         IM    14   7   0
 73  * WAITING FOR HOST TO TAKE LAST SAMPLE
 74 0014 8412  JMP      NOP      #HOSTWT                   IM     1   0  18
 75 0015 0000  NOP      NOP                                      0   0   0
 76 0016 FD84  M1D/AD   MPYH     #DATAD                    IM    15   4 100
 77  * SET FLG =1 TO TELL HOST DATA READY
 78 0017 B401  TRANS    NOP      1                         IM    13   0   1
 79 0018 FDA6  M1D/AD   TRANS    #FLGAD                    IM    15   5 102
 80 0019 8402  JMP     =NOP     ,#SAMLO                    IM     1   0   2
 81 001A 0000  NOP      NOP                                      0   0   0
0 WARNINGS TOTAL
0  ERRORS TOTAL
```

Fig. 23

```
 2  * RUNNING SINUSOID SUMMATION SYNTHESIS
 3  *       =          ,
 4  * USES DOUBLE PRECISION PHASE ADDITION
 5  * FOR HIGH ACCURACY PITCH CONTROL.
 6  * AMPL SUMMATION IS S.PRECISION.
 7  * RUNS AS SUBROUTINE THAT CALCS NHOP
 8  * PTS OF WAVEFORM ON EA.  CALL(STACKED)
 9 01CA 9008  MC                 8        #SINSUM  ENTRY   IM   4   0   8
10 01CB 9400  AC                 0  IM7 SET                  IM   5   0   0
11  * POP THE CNT/ADDR PARAMETERS
12  **SET UP TO DO "NHOP" SAMPLES
13 01CC BC15  M1D/AD  NOP        #ENDRES                    IM  15   0  21
14 01CD 54E0  #T1     MDAT       NOP          PTS TO END         21   7   0
15  * SET UP PTR TO RESULTS STACK IN M1
16 01CE 58E0  #T2     MDAT       NOP                             22   7   0
17 01CF FC1C  M1D/AD             #NSINES #SAMLOP            IM  15   0  60
18 01D0 E1F7  ADD     MDAT       #PHASES                    IM   8   7 119
19 01D1 5063  #CNT    ALU        ALU                             20   3   3
20 01D2 B4E1  TRANS   MDAT       1 IS OVERALL AMP           IM  13   7   1
21 01D3 A460  AND     ALU        0                          IM   9   3   0
22 01D4 4063  #ALUS   ALU        ALU  USED FOR SAMPLE            16   3   3
23 01D5 D917  MADDR              #PHASES                    IM   6   0 119
24 01D6 9009  MC                 9        POP M1,PUSH 0     IM   4   0   9
25 01D7 20E7  ADD     MDAT       MDAT ADD LSPS                   8   7   7
26 01D8 1C03  MDAT    NOP        ALU      #SINLOP  *              7   0   3
27 01D9 0000  NOP                           WAIT ON MO PU        0   0   0
28 01DA 2027  ADD     AFLG       MDAT                *          8   1   7
29 01DB 20E3  ADD     MDAT       ALU       ADD MSPS *           8   7   3
30 01DC 1C03  MDAT               ALU  PUSH MSP                  7   0   3
31 01DD A068  ADD     ALU        8 ROUND FOR SINE           IM   8   3   8
32  *
33  * NOTE 65DB S/N POSS W 11B*1024 SINE
34 01DE 9478  SIN/AC  ALU        24  ROV&SEE OV & >=        IM   5   3  24
35  * MPY BY AMPL OF THIS COMPONENT
36 01DF 30ED  MPY     MDAT       SIN                             12   7  13
37 01E0 2090  ADD     MPYH       #ALUS  UPDATE SAMPLE            8   4  16
38 01E1 4063  #ALUS   ALU        ALU                             16   3   3
39 01E2 3854  COMP    MOADDR     #CNT                            14   2  20
40 01E3 8578  JMP                #SINLOP                    IM   1   0 472
41 01E4 20E7  ADD     MDAT       MDAT  NEXT LSP ADD             8   7   7
42  * 12 INSTRUCTION INNER LOOP
43  * NOW CHECK IF ZERO CROSSING SO CAN
44  * CHANGE PARAMETERS WITHOUT CLICK
45 01E5 BA21  COMP    #DPO       1  EQUIV TO >0             IM  14  17   1
46 01E6 C56A  JMP                #NEG  EXEC IF <=0 NOW     IM   1   0 490
47 01E7 0000  NOP                                               0   0   0
48 01E8 C574  JMP                #CONTIN  EXCE IF IS >0    IM   1   0 500
49 01E9 0000  NOP                                               0   0   0
50 01EA BA00  COMP    #ALUS      0        #NEG  >=TEST      IM  14  16   0
51 01EB C574  JMP                #CONTIN #CONTI  IF<0 LA   IM   1   0 500
52  * BLOCK MOVE OF CONTROL PARAMS
53 01EC FC1C  M1D/AD             #NSINES                    IM  15   0  60
54 01ED 9819  MADDR              #PARIN (HOST INPUT)        IM   6   0  25
55 01EE 9018  MC                 24  PUSH1,POPO             IM   4   0  24
56 01EF 22E7  ADD     #ZRO       MDAT                            8  23   7
57 01F0 1C60  MDAT    ALU        NOP      #MOVP                  7   3   0
58 01F1 F956  COMP    MOADR      #ENDPIN                    IM  14   2 118
59 01F2 C570  JMP                #MOVP                      IM   1   0 496
60 01F3 22E7  ADD     #ZRO       MDAT                            8  23   7
```

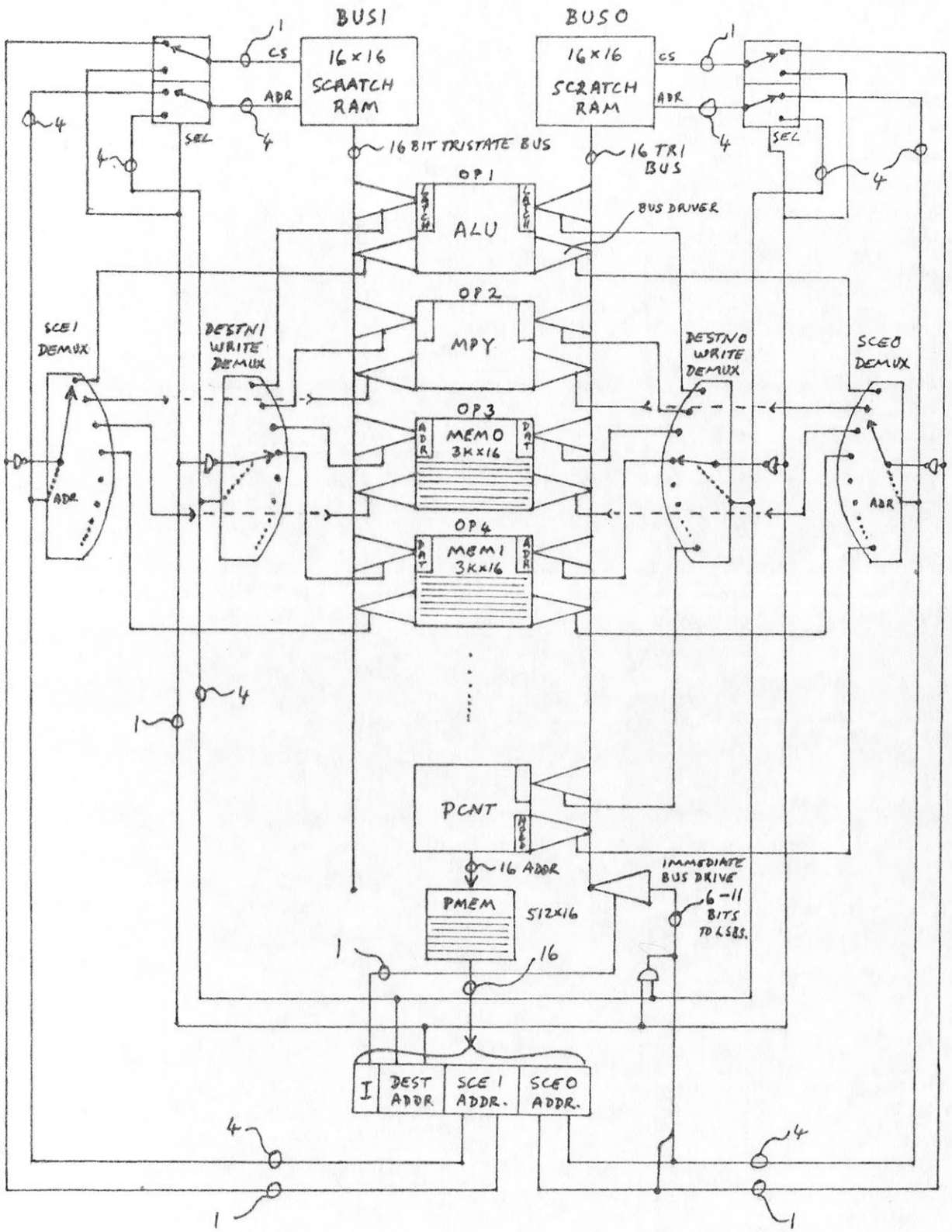
Fig. 23 (Contd.)

```
61   * (20 PARAM SETS ALLOWED)
62   * MPY BY OVERALL AMPL
63 01F4 3205   MPY      #ALUS    TRANS    #CONTIN  AMPL          12   16    5
64 01F5 A2C1   ADD      #T2      1                          IM    8   22    1
65 01F6 3C83   M1D/AD   MPYH     ALU                              15    4    3
66 01F7 4480   #DP0     MPYH     NOP                              17    4    0
67 01F8 5863   #T2      ALU      ALU                              22    3    3
68 01F9 36A3   TRANS    #T1      ALU                              13   21    3
69 01FA 38A5   COMP     TRANS    TRANS  REACHED END ?                14    5    5
70 01FB 856F   JMP               #SAMLOP                    IM    1    0  463
71 01FC 9009   MC                9 RESTORE MEM STATUS       IM    4    0    9
72 01FD 0401   JMP               RETURN                           1    0    1
73 01FE 0000   NOP                                                0    0    0
74   DATA FILE FOR MAIN MEM BEGINS  @M1,20
75 0014 0100 256      #NHOP
76 0015 0900 2304     #ENDRES
77 0016 0800 2048     #RPTR
78   * END RESULTS AREA OF M1
79 0017 0014 20       #LITTLE
80   DATA FILE FOR MAIN MEM BEGINS  @M0,25  PARAMETERS INPUT FILE
81 0019 0002 2        #PARIN
82 001A 07D0 2000
83 001B 0000 0
84 001C 00C8 200
85 001D 01F4 500
86   *ENDS AT 118
87   DATA FILE FOR MAIN MEM BEGINS  @M0,118   RUNNING PHASE TABLE
88 0076 0000 0        #ENDPIN
89 0077 0000 0        #PHASES
90 0078 0000 0                 MSP
91   DATA FILE FOR MAIN MEM BEGINS  @M1,60 ** PARAMETERS WORKING FILE
92 003C 0002 2        #NSINES  TWICE NO. OF SINES REQ
93 003D 03E8 1000     #AMPLA   OVERALL AMPL
94 003E 0000 0        #PHIAMP   PHASE INC LSP
95 003F 07D0 2000           PHASE INC MSP
96 0040 1388 5000           AMPL OF THIS COMPONENT
97   * THREE WORDS DESCRIBE EA COMPONENT
5 WARNINGS TOTAL
0  ERRORS TOTAL
```



Block diagram of sine-wave synthesis algorithm (Harmoniac machine language).

Fig. 24



An expanded block diagram of Harmoniac

Fig. 26

At location two of the program, memory one address is set to point at the first of the coefficients, beginning the loop which adds sine components to build up the resultant sample. The running-phase table address is initialised at location four and the initial value of the sample is zeroed at location five. At location six the current value of the running phase from memory zero is added to the phase increment. The updated value is stored back in memory zero at location nine. This phase is used to look up the sine table at location seven and the resultant sine is multiplied by the amplitude of this component (from memory one) in location eight. By location ten (Hexadecimal:A) the result of the multiply must be ready so it is added to the accumulating sample value in location ten and saved in location eleven (:B). A check is made at location twelve (:C) to see if all the sine components have been added in, if not a loop is made back to location seven. Note that the instruction after the jump is also part of the loop (location 14) and it is used instead of going back to the instruction at location six.

At the end of the loop the sample has accumulated and it is multiplied by the overall amplitude at locations 15(:F) and 17(:11) and placed in memory where the host will find it at 22(:16) after waiting for the host to accept the previous result by polling the flag memory location in a small loop at 18(:12), 19(:13) and 20(:14) for zero flag. The flag is then set to one to tell the host that the new sample is ready (at 24(:18)). After this the program loops back to do the next sample.

The listing of the double-precision sine-synthesis subroutine in Fig. 23 (SINSUM in FFT1DS) has several improvements to make it more practical to use. Double-precision phase calculation gives

much finer control of frequency at a small cost in execution time (25% slower). It also detects positive zero crossings of the resultant waveform, changing the input coefficients only at these points to avoid step changes in the waveform. This section should be improved if real-time operation is required as the coefficients are changed at every zero crossing and every zero. It would be better to change at the first zero crossing of a buffer full of samples and do a default change of coefficients at the end of a buffer in coase no zero crossing occurred. (Limited space prevented this.)

Using a buffer, the double-precision sine-wave generator should be able to produce about eight to ten sines at a thirty microsecond sample rate. It has a signal to noise ratio of about 65 dB in the prototype of Harmoniac (see ref. 8, "Noise in Digital Oscillators").

4.3.2 Maths, FFT and Power Spectral Analysis Package

In appendix II can be found the listings of a spectral-analysis package designed for rapid generation of power spectra, pitch analysis and smoothing of power spectra. It includes several general-purpose mathematical routines such as division, integer logarithmic routines and an exponential base two. It is basically a cepstral analysis routine used for speech analysis. In the initial implementation at the Speech and Language Research Centre, this transform package is used together with display and A/D (analog to digital) routines to produce four-colour spectrograms on a television display with a hardcopy facility.

The FFT (Fast Fourier Transform) subroutine itself begins on page eight, location 307 (see right column) of the listing in

appendix II. It uses a software system for bit reversed reordering (at location 39 and following) so that it can be used for any number of points that is an integer power of two. A program which uses the bit-reverse jumpers on the main memories has been written which runs significantly faster, but it lacks flexibility in the number of points that can be used. The routine shown is a translation of one given in Fortran by Markel in ref. 9 ("FFT Pruning"), which gives a time saving when smoothing transforms are being executed. (Where the number of input points is less than the number of output points.) The time-saving check is performed at location 327. A normal 512-point complex transform takes approximately 35 milliseconds using this routine.

The FFT inner loop contains twenty-five instructions, performing one "butterfly" of the transform per pass. Most of the instructions are used for address calculation. At locations 370-372 can be seen a typical programming trick to allow the multiply instruction to execute fully without wasting any time waiting the one instruction delay required. Another instruction whose position was not very critical has been put between the initiation of the multiply and use of the result. This has been done to ensure the multiply was complete, although it was found to be unnecessary in the prototype.

The integer-divide subroutine can be found at location 198 in appendix II. It uses a conventional shift-left-and-subtract method and hence is not very fast. It was assumed that division will not be much used in signal processing. The inner loop contains eleven instructions, executed sixteen times so a division takes approximately thirty microseconds.

The integer logarithm (base "e") is from location 27 to 111. It is based on the fractional log base e from 84 to 111, which calculates a power-series approximation to the logarithm. Execution takes approximately sixty microseconds. The log algorithm was translated from C.A.I. ALPHA-16 Assembler utilities. A table-look-up algorithm would be about sixty times faster.

The method used to call the FFT and the power spectrum options is a loop polling a single flag location which is set up by the host processor when some operation is to be performed on a buffer of data. This loop (from 0 to 20) does a series of comparisons in the "not-equals" mode so that a given option is not performed until the number of that option appears in the flag location.

Nested subroutines are used throughout this package. This has been achieved by careful attention to subroutine heirarchy and reservation of scratch registers to store return points - Harmoniac does not have a hardware stack. As an example DPNEG(248) is used by DIV(198) which is used by LOGF(84) which is used by 1 LOGE(27) which is used by 1 LOG 10(21) which is used by PWRLOG(155) which is used by PSPECT(465).

4.4 Signal Processing from the Host Computer

As an example of a full signal-processing algorithm which uses Harmoniac together with a host computer, a listing is given in appendix III of a singing-voice analysis and resynthesis routine which operates on the HP21MX in Fortran. It uses a modified version of the FFT package shown in appendix II for analysis with the sine-wave resynthesis routine of Fig. 23 as one of the options (instead of the cepstrum option). This system has been implemented to re-

process noisy acoustic gramophone recordings of opera singers, and it has worked quite effectively, especially on the soprano voice. It is fully described in an article in the Speech and Language Research Centre's (S.L.R.C.) Working Papers which is included in appendix III.

The system is arranged so that both computers are processing simultaneously for maximum speed, but program memory limitations on the prototype of Harmoniac required that the pitch extraction section by partly done in the HP21MX and this is the limiting factor on processing speed. (Approx. 500 milliseconds per ten milliseconds processed.) Note that block floating point has been used to maintain high amplitude precision in the FFT's.

4.5 Stand-Alone Operation

The use of read only memory (ROM) for the program and addition of some analogue interface circuitry would make it possible to use Harmoniac as a stand-alone signal processor. The ROMs have been provided in the prototype together with a switch which allows the ROMs to be the lowest part of memory so that execution will begin on the program in ROM when power is switched on. (See Fig. 7.)

The extra circuitry required to implement an analog interface would be: a real time clock (2 chips), an interrupt line (six chips), analogue to digital (A/D) and digital to analogue (D/A) converters (six chips) plus a DC -DC converter to provide the negative supply required. This could not be implemented in the prototype because of a lack of space and time. In any case it would probably be cheaper and easier to use a standard 16-bit microcomputer as a host so that standard interfaces could be provided.

4.6 Maximum Possible Speed

Great care was exercised in the initial design of Harmoniac to keep the timing as simple as possible with a minimum number of gate delays in the more critical paths in order to allow high speed operation. In the case of bus transfers, phases two and three are used to place the data on the bus while phase three provides the setup time for the input latch on each operation. The chip configuration used would allow a clock period of 35 nanoseconds per phase, worst case, but this cannot be achieved in practice because of the comparison function loop and the main-memory address counters.

To achieve a simple comparison facility, the loop was incorporated into the arithmetic unit (using 74S181 chips). The comparison must either inhibit or allow the instruction which follows the comparison. To delay the decision of comparison any more than one instruction would make programming awkward. In order to inhibit the instruction following the comparison instruction and at the same time allow as much time as possible for the decision to be made, the result (true or false) of the comparison is used at the last possible point in the instruction execution. This is during phase three when the destination-enable pulse is generated via the demultiplexers 344, 345, 363 and 364 (Fig. 6, 74S138's). To be sure that no partial destination-enable pulse is generated, the comparison decision must be available at the input to these demultiplexers just before the start of phase three. The data upon which the comparison is to be made reach the arithmetic unit up to 43 nanoseconds after the previous phase three and the arithmetic unit may take up to 49 nanoseconds to deliver the decision back to chips 344 etc. (the demultiplexers). Hence the interval between the end of a phase three to

the start of the next phase three cannot be less than 92 nanoseconds. So the clock period must be 46 nanoseconds or more, giving an instruction execution time of 138 nanoseconds. The breakdown of these comparison delays is as follows:

(Refer to Figs. 15 and 6.)

PINS	CHIP NO.	CHIP TYPE	MAX.DELAY (nanoseconds)	SIGNAL GENERATED
2, 3	350	74S00	5	from ϕ_{3A} to demux. enable
5, 9	344(etc)	S138	11	through demux. to destination pulse
1, 8	722	S 30	5)	to clock at ALU
1, 3	716	S 00	5)	destination
9, 3	704(etc)	S175	17	to data at ALU (through latch)
19, 14	700(etc)	S181	30	to "=" output of ALU
12, 11	725	S 86	10.5)	through comparison
6, 7	724	S158	7.5)	mode control to "COMP"
2, 3	350	S 00	-2	(only differential delay of 350 relevant)
	1M LEAD LENGTH DELAY		3	
			92	MAXIMUM DELAY FOR COMPARISON

This 92 nanoseconds represents the periods of two phases of the clock. Hence the worst-case minimum instruction cycle (three clock phases) is 138 nanoseconds if the comparison path is the limiting factor.

The main memories have a similar worst-case speed restriction. These will be analysed with reference to Fig. 10, where the memory write-enable pulses are generated, and Figs. 11, 12 which show the main-memory address generators. During the real cycles, there is no great problem as the address-change time and memory-access times are

just added together and need only be less than the full three-phase cycle by a margin of the time to charge up the bus and the setup time of the latch at the destination of the data. The address-change time is the time taken for the address counters on the memories to change after the previous read cycle (when "popping" data) and for that address to reach the memory chips. This period is 50 nanoseconds worst case for a read. The read-access time 60 nanoseconds worst case for the 93425 memory chips used. This bus-charge and set-up time for the destination latch (8551) is 30 nanoseconds. These add up to a 140 nanosecond instruction cycle, but none of these periods need to be synchronised so that the worst-case conditions mentioned here are extremely unlikely to all occur together. It should be noted that there is a possible extra period of 8 nanoseconds in this cycle due to the 74S138 destination demultiplexers. The S138 which starts the address change may be different to that which latches the data at the destination so the difference between the maximum and minimum delays is relevant. This would lead to a possible 148-nanosecond cycle but this is even more unlikely in practice.

The timing of the main-memory write cycle is a little more critical as the address change during a succession of "push" instructions to main memory must occur in less than one clock period as the other two clock phases (2, 3) are used to generate the write-enable pulse. The new address must be stable at the memory chip inputs a few nanoseconds before the write-enable pulse arrives (the address setup time). An analysis of the delays in the chain from the last write-enable to the new address follows:

PINS	CHIP NO.	CHIP TYPE	MAX.DELAY (nanoseconds)	SIGNAL GENERATED (Figs. 10, 12)
2, 12	58	74S12	5)	from LWEI neg. edge
11, 3	58	S12	5)	
9, 3	59	S00	5)	to addr. cntr. clock
2, 14	31(etc)	S161	10	to addr. change out of cntr.
6, 7	41(etc)	S253	20	to address at memory array (50 pF)

Total delay from LWEI to address = 45 nanoseconds

The worst-case memory address setup time available before the next write to memory is:

$$T_{SUP} = T_{CLK} - T_{ADDRCH} + T_{WEDLY}$$

T_{ADDRCH} is 45 nanoseconds

T_{WEDLY} is the minimum delay from LWEI to the write-enable pulse to the whole memory array, assumed 3 nanoseconds

Hence $T_{CLK} = T_{SUP} + T_{ADDRCH} - T_{WEDLY}$

$= 5 + 45 - 3$

= 47 nanoseconds, assuming 5 nanoseconds setup time on the 93425 or 2125 RAMs. (T_{SUP})

The worst-case cycle-time limit due to main memory writes is 141 nanoseconds. This limit is more likely to be significant in practice than the read cycle because one slow memory chip or one slow address counter has a delay which is a greater proportion of the available time for the event, the address change, which must be completed in less than one clock period.

So, in a particular implementation of this design the most likely cause of the top-speed limit would be either the comparison

delay (max. 138 nanoseconds per instruction) or the memory-write cycle which leads to an instruction cycle of 141 nanoseconds. Hence the design speed could be set at 140 nanoseconds.

In the prototype it would be expected that an instruction time of better than 140 nanoseconds should be possible as the delays in typical chips are generally less than the maximum figures quoted above. In actual operation of complex algorithms the minimum instruction time was found to be approximately 150 nanoseconds. This was apparently due to jitter in the master clock generator which could be seen in its waveform. The jitter was due to the use of a voltage controlled oscillator (74 S 124) in a relatively noisy electrical environment. This jitter could cause certain instruction periods to be up to 20% shorter than the average.

Unfortunately, it was not possible to acquire a suitable crystal to test the machine at full speed, although an "outboard" oscillator would be possible if it used a separate power supply. In any case the performance of the machine was quite adequate for the tasks for which it was designed at the Speech and Language Research Laboratory.

Several ways to improve the speed of the compare operation are possible. One would be to use selected chips in this area for maximum speed. Another method that would give a similar speed in the comparison as in transfer and other functions is to reduce the number of gates in the comparison path by reducing the flexibility of the compare. This would eliminate chips 725 and 724 in Fig. 15, giving a cycle of 111 nanoseconds. Unfortunately, this method would make programming very awkward as only the "equals" compare would be

available. Another method would be to extend the time available for the comparison by making the conditional instruction the second one after the comparison rather than the first. The conditional instruction is usually a jump so the instruction which follows it is always executed, even if the jump is not. Hence the comparison at present usually takes three instruction times (one is the jump shadow) so it would take four instruction times if the compare execution was delayed. There would then be a "compare shadow" instruction as well. This would be rather awkward for the programmer, although it would allow an instruction cycle.

A fourth method would be the use of separate comparison hardware. This would avoid the destruction of the previous ALU contents which occurs in the prototype but would be very expensive in terms of extra hardware. This method would both speed up the cycle time as well as eliminating instructions currently used to save and restore the ALU contents in certain loops. All of these methods for improving the comparison speed were rejected because they were either too expensive to implement (especially since the available chassis space was full) or too awkward for the programmer. The use of a crystal-controlled clock generator should allow the machine to run at a speed limited only by the comparison or the memory address change time, that is 140 nanoseconds worst case. The direct memory access feature, which stops and restarts the main timing ring, is designed to operate at clock periods down to 30 nanoseconds so it is not a limiting factor on Harmoniac's speed.

5. CONCLUSION

Although Harmoniac has been successfully applied to most of the tasks for which it was designed, some areas leave room for improvement. The main limitation discovered was word length. Although 16-bit words are adequate in most applications envisaged, it was found that 20- or 24-bit data would have been desirable for FFT and filtering processes. In the FFT a kind of block floating point had to be used when 1024 PT. transforms were performed on 12-bit input data to avoid overflow during the transform, (i.e. pre-scaling and post-scaling of data). Extension of the machine to twenty bits would be quite easy except for the interface to a 16-bit machine. It would probably be simplest to provide direct access to the lower sixteen bits from the host and leave the program memory to sixteen bits.

The other area which could be improved in Harmoniac is processing speed. With the logic type and architecture used it should be possible to derive an instruction execution time of 110 nanoseconds maximum. It seems that this target was not met because of the method of comparison used (in ALU).

The most notable feature of this design is the use of only 16-bit wordlength in the program memory while maintaining speed and efficiency of hardware usage. Most signal processors have employed very large wordlengths in the program memory so that several addresses could easily be provided simultaneously. The method used to keep the program word short was to keep the number of instructions small and to treat main memory ports and all instructions like a small address space of thirty-two words. This has allowed very simple interfacing to cheap mini-computers and a low overall cost of implementation of a fast signal processor.

REFERENCES

1. "The FDP, a fast Programmable Signal Processor", I.E.E.E. T.C. Vol. C20 No. 1 Jan. 71 pp. 33-38, Bernard Gold, Irwin Lehow, Paul McHew, Charles Rader.

2. "A Special Purpose Computer for Digital Signal Processing" De Mori, Renato, T - C 75, Dec. 1202-1211.

3. "Digital Filter realisations using a special purpose stored program computer", White and Nagle, I.E.E.E. T.A.E. Oct. 1972, pp. 289-294.

4. Microprocessor realization of a Linear Predictive Vocoder, Hofstetter, Tierney, Wheeler, I.E.E.E. A.S.S.P. Vol. 25 No.5, October 1977.

5. "Digital Signal Processing", Rabiner and Gold, 1977.

6. "G.A.S.P. A Fast General Purpose Signal Processor", Fensom, Smith and Ackland (University of Adelaide) paper given at the Conference on Computers in Engineering in 1978 Canberra, 23-25 August.

7. "A digital oscillator which can generate up to 256 sine waves in Real Time", Computer Music Journal No. 2, 1978.7 J. Snell

9. "FFT Pruning", J. Markel, I.E.E.E. Transactions on Audio and Electroacoustics, Vol. AU. 19 No. 4, December 1971.

10. "Array Processor Provides High Throughput Rates", W.R. Willmayer, Computer Design.

11 "Reflections in a pool of processors", S. Harbison and W.A. Wulf, Technical Report, Dept. Comput. Sci., Carnegie-Mellon University, Pittsburg, P.A., November 1977.

12. "Some Issues in Programming. Multi-mini-processors", A. Newell and G. Robertson, C.M.U. Report January 1975.

13. "Programming Issues raised by a multi-microprocessor", A.K. Jones, R. Chansler et al Proc. I.E.E.E., Jan. 1978.

14. "Real-Time Linear - Predictive Coding of Speech on the SPS-41 Triple Microprocessor Machine", Michael J. Knudsen I.E.E.E. & A.S.S.P. Feb. 1975, 140-145.

15. "Microprogramming a mini-computer for fast signal processing", T. Mulrooney, Electronics, March 16, 1978.

16. Effects of finite register length in digital filtering and fast Fourier transforms", A.V. Oppenheim and C.J. Weinstein, Proc. I.E.E.E. Vol. 60, pp. 956-976, Aug. 1972.

17. The Technology of Computer Music, M.V. Mathews, M.I.T. Press, Mass. 1969.

18. "LDVT: High Performance mini-computer for real-time speech processing", presented at the 1975 EASCOM Conf. Washington, D.C., Sept. 29-Oct. 1, 1975.

APPENDICES

APPENDIX 1

```
 PAGE    1   HARMONIAC ASSEMBLY OF : HAREF   9 AUG 1979 ALL INSTR. SET
LINE ADDR MEM.      SOURCE CODE          #LABELS(JMP) *CMTS     DECODED OBJEC


   2  NEW PROGRAM SEGMENT BEGINS  @PM,0
   3 0000 0000  NOPCALL=NOP     ,NOP      #START (JSUB)          0   0   0
   4 0001 0421  JMP     =AFLG   ,RETURN                        1   1   1
   5 0002 0842  IOADR   =MOADR  ,I2                            2   2   2
   6 0003 0C63  ODAT    =ALU    ,ALU                           3   3   3
   7 0004 1084  RSH/MC  =MPYH   ,MPYL                          4   4   4
   8 0005 14A5  SIN/AC  =TR01   ,TR10                          5   5   5
   9 0006 18C6  MADR    =M1ADR  ,I4                            6   6   6
  10 0007 1CE7  MDAT    =MDAT1  ,MDAT0                         7   7   7
  11 0008 2108  ADD     =Z1     ,Z1                            8   8   8
  12 0009 2529  AND     =Z2     ,Z2                            9   9   9
  13 000A 294A  SUB     =I1     ,Z3                           10  10  10
  14 000B 2D6B  OR      =I3     ,SOV                          11  11  11
  15 000C 318C  MPY     =RSH    ,SMPYAD                       12  12  12
  16 000D 35AD  TRANS   =Z3     ,SIN                          13  13  13
  17 000E 39CE  COMP    =Z4     ,BUS1 *NOT INSTALLED          14  14  14
  18 000F 3DEF  M1D/AD =Z5      ,RMPYA *SETS NORMAL MPY       15  15  15
  19 0010 4210  SCRnn   =SCRnn  ,SCRnn                        16  16  16 DES
SOURCE ONE ERROR !  99
SOURCE ZERO ERROR !  99
  20  *      =      ,
  21  * FIRST 2 CHS ONLY REQ'D
  22  *NUMERIC SCE ZERO CAUSES IMMEDIATE MODE
  23 0011 8400  JMP     =NOP    ,#START                  IM    1   0   0
  24  * SUBROUTINE JMP/RETURN SEQUENCE CAUSES
  25  * THE "JMP SHADOW" INSTR STRAIGHT AFTER
  26  * CALL TO BE EXECUTED TWICE-BEWARE!
  27  * INSTRUCTION AFTER A JMP IS EXECUTED
  28  * BEFORE! THE JMP ("JMP SHADOW").
  29  * LABELS START WITH "#",ARE 6CHS LONG.
  30  * JMP DEST LABELS ARE IN 4TH FIELD.
  31  * SCRATCH VAR LABELS IN 1ST 3 FIELDS.
  32  * DATA MEM ADDR LABELS IN 2ND FIELD
  33  *
  34  * DURING 7 BIT IMMED INSTRS, NO HI SCR
  35  * (>24) OR HI SCES(>7) AVAIL ON BUS 1.
  36  * (AS BIT 3 OF SCE1 USED BY 7BIT IMMED)
  37  * 6BIT IMMEDS GIVE ALL OPS AS NORMAL
  38  * EXCEPT THAT SCR CANNOT BE WRITTEN.
  39  *
  40  * RSH/MEM CONTROL BIT USAGE AS BELOW
  41  * USE SUM OF OPTIONS REQ'D
  42  *
  43  *       MEM1        STACK          MEMO
  44  *                   BOTH!
  45  *--------|-----|-------|--------|------
  46  *NORM (0)| POP | STACK| NORM (0)|POP(0)
  47  *DOWN(32)| (0) |  (8)  |DOWN (2)|
  48  * RAD(64)|PUSH | NON(0)| RAD (4)| PUSH
  49  * DMA(96)| (16)|       | DMA (6)|  (1)
  50  *
  51  * SIN/ARITH CONTROL BIT USAGE AS BELOW
  52  * USE SUM OF OPTIONS REQ'D
  53  *NOT EQU | IM6 |COMPARE|OV&HS |AC OUT&
  54  *  (64)  | SET | TEST  |RESET |RSH IN
  55  *NORM EQU| (32)| =(0)  | IF(0)| OV(0)
```

```
PAGE    2   HARMONIAC ASSEMBLY OF : HAREF
 56  *    (0)   !RESET! >(8)   !SAVED ! HSC(1)
 57  *          ! (0) ! <(16)  ! IF(4)!LRSB(2)
 58  *          !     !>=(24)  !      !MPOV(3)
 59  *                                  &
 60  *                                !ARITH(0)
 61  *                                !LOGRT(1)
 62  *                                !ROTRT(2)
 63  *                                ! OV (3)
 64  *
 65  * COMPARE CAUSES SKIP OF FOLLOWING INST
 66  * WHEN SET CONDITION IS TRUE(EG IF=)
 67  *
 68  *  ASSEMBLER ERROR MESSAGE TYPES :
 69  * S / D ERR 1 - OP. CODE DOES'NT EXIST
 70  * DEST ERR  2 - FORBIDDEN DEST SEQUENCE
 71  *   (NOT ENOUGH TIME FOR OP. TO FINISH)
 72  * SCE0 ERR  3 - JMP/CALL/SCR0 SYMB UNDE
 73  * DEST ERR  4 - SCRATCH VARIBLE OVERFLO
 74  * SCE1 ERR  5 - LABEL DOESN'T EXIST
 75  * DEST ERR  6 - NO SCR DEST W IMMED SCE
 76  * SCE0 ER 101 - IMMED>2047 OR <0 ON JMP
 77  *               OR A CONST OUT OF RANGE
 78  * SCE0 ERR 97 -
 79  * SCE1 ERR 99 - SCR RAM BEING READ &
 80  *             WRITTEN IN ONE OPERATION
 81  * SCE0 ERR 99 -   "     "     "    "
 82  * SCE0 ER 127 -  IMMED >127 (IM7)
 83  *             (NOTE: 63 MAX ON IM6)
 84  * SCE1 ERR 24 - SCE1 ADDR >24 WITH IM7
 85  *          (NO HI SCR ALLOWED WITH IM7)
 86  * --------WARNING ONLY-----OK IF IM6
 87  * SCE1 ERR 15 - SCE1 ADR >7 & <16 W IM7
 88  * ----------WARNING ONLY--------------
 89  *          (NO HI OPS(SCE1) WITH IM7)
 90  *  (NOTE - THIS IS NOT ERR IF IM6 SET)
 91  *
0 WARNINGS TOTAL
1  ERRORS TOTAL
```

APPENDIX 1 (Contd.)

```
 PAGE    1   HARMONIAC ASSEMBLY OF : FFT1MN  MAIN CALLING PROG FOR FFT
LINE ADDR MEM.      SOURCE CODE          #LABELS(JMP) *CMTS      DECODED OBJEC


  2  * 21APR1980 SYNC. RESYNTH PARAM CHANGE
  3  * N PT TRANSFORM
  4  * FOR ANY "N" (SOFT BIT REVERSE)
  5  * -NOT AS FAST AS HARD BIT REVERSE
  6  * LOG/LIN PWR SPECTRUM
  7  * & SMOOTHED PWR SPECT & PITCH AVAIL.
  8  * SET UP FOR 12BIT INPUT,10 BIT DISPLAY
  9  * MOD'D TO INCL RESYNTHESIS W PSPECT
 10  * PHIL CONNOR,SLRC,MAQUARIE UNIVERSITY
 11  * FIRST A "PROTECT" PROG TO OPERATE
 12  * DURING FILL UP OF HARM'S MEMS
 13  NEW PROGRAM SEGMENT BEGINS  @PM,510
 14 01FE C57E  JMP              #HERE   #HERE          IM   1   0 510
 15 01FF 0000  NOP                                         0   0   0
 16  NEW PROGRAM SEGMENT BEGINS  @PM,0
 17 0000 C57E  JMP              #HERE                  IM   1   0 510
 18 0001 0000  NOP                                         0   0   0
 19  *
 20  $#ALUS #DP0 #DP1 #TEMP #CNT
 21  $ #T1 #T2 #ZRO
 22  DATA FILE FOR MAIN MEM BEGINS  @M1,0
 23 0000 0200 512     #ARGS
 24 0001 0200 512
 25 0002 000A 10      #L
 26 0003 000A 10      #M
 27 0004 0000 0       #PITCHP    PITCH PERIOD RESULT
 28 0005 0000 0       #PITCH
 29 0006 0005 5       #SMOOTH
 30 0007 0400 1024    #DISCL  FOR A 10 BIT DISPLAY
 31 0008 000F 15      #VTHRSH
 32 0009 FFFE -2      #FFFEM1
 33 000A 0000 0       #RDYFL2
 34 000B 3FFF 16383   #PI/2
 35  DATA FILE FOR MAIN MEM BEGINS  @M1,50
 36 0032 7FFF 32767   #SGNMSK  THESE 2 RESTORED
 37 0033 0000 0       #RDYFLG  BY FFTHD(CALLER)
 38 0034 8000 -32768  #8000
 39  * BEWARE ,BUGFLG IS AT 53
 40  DATA FILE FOR MAIN MEM BEGINS  @M1,54
 41 0036 3796 14230   #PLOG1E
 42 0037 4000 16384   #H4000
 43  DATA FILE FOR MAIN MEM BEGINS  @M0,0
 44 0000 2710 10000   #SAMRAT
 45 0001 0080 128     #LINSCF
 46 0002 FFFE -2      #FFFE
 47 0003 58BA 22714   #PLOGE2
 48 0004 4000 16384   #HLF
 49 0005 0E39 3641    #C4
 50 0006 1249 4681    #C5
 51 0007 1999 6553    #C6
 52 0008 2AAA 10922   #C7
 53 0009 0400 1024    #P1024
 54 000A 0000 0       #PWRRET
 55 000B 0000 0       #PSPR
```

```
PAGE    2   HARMONIAC ASSEMBLY OF : FFT1MN
 56 000C 0000 0         #PSSPR
 57 000D 0000 0         #FFTRET
 58  NEW PROGRAM SEGMENT BEGINS  @PM,0
 59  * MOST OF PROGRAM USES IM6 MODE TO
 60  * GIVE ACCESS TO ALL REGS & 0-63 IMMEDS
 61 0000 9000  RSH/MC =NOP     ,0        #START              IM   4   0   0
 62 0001 9400  SIN/AC  NOP      0  SETS IM7                  IM   5   0   0
 63 0002 D500  SIN/AC  NOP      96  SETS NOT EQU,I6          IM   5   0  96
 64 0003 A460  AND     ALU      0  GENERATE A ZERO          IM   9   3   0
 65 0004 5C63  #ZRO    ALU      ALU                              23   3   3
 66 0005 FC13  M1D/AD =NOP     ,#RDYFLG #WAIT               IM  15   0  51
 67 0006 B8E1  COMP   =MDAT    ,1                           IM  14   7   1
 68 0007 C043  CALL    NOP      #FFT                         IM   0   0 291
 69 0008 FC13  M1D/AD  NOP      #RDYFLG   FOR NEXT IN T      IM  15   0  51
 70 0009 B8E5  COMP    MDAT     5                            IM  14   7   5
 71 000A C07B  CALL    NOP      #PSPECT DOES LOG SPECTR      IM   0   0 443
 72 000B FC13  M1D/AD  NOP      #RDYFLG                      IM  15   0  51
 73  * COMP    MDAT     6 REM'D-RESYNTH
 74  *CALL    NOP      #SSPECT SMOOTHED SPECT
 75  * M1D/AD  NOP      #RDYFLG
 76 000C 9000  RSH/MC  NOP      0          MEM NORM          IM   4   0   0
 77 000D BC0A  M1D/AD  NOP      #RDYFL2                      IM  15   0  10
 78 000E B8E1  COMP    MDAT     1                            IM  14   7   1
 79 000F FEF3  M1D/AD =#ZRO    ,#RDYFLG                     IM  15  23  51
 80 0010 8400  JMP    =NOP     ,#START                      IM   1   0   0
 81 0011 BEEA  M1D/AD  #ZRO     #RDYFL2                      IM  15  23  10
 82  *
  2  * 20 JULY 1979 ,
  3  * THERE IS SLIGHT ERROR IN LOGE ???
  4  *
  5  * INTEGER LOG BASE E
  6 0012 6001  #L/M     NOP      RETURN  #ILOGE                   24   0   1
  7  *IN ALU,OUT ALU
  8  *RETURN ZERO IF <=ZERO IN
  9 0013 4063  #ALUS    ALU      ALU                              16   3   3
 10 0014 D408  AC                40  >,I6                    IM   5   0  40
 11 0015 B860  COMP    ALU      0  >0?                      IM  14   3   0
 12 0016 0418  JMP              #L/M  SKIP THRU IF TRUE          1   0  24
 13 0017 A2E0  ADD     #ZRO     0                            IM   8  23   0
 14 0018 C01D  CALL    NOP      #NORM                        IM   0   0  61
 15 0019 A200  ADD     #ALUS    0  RESTORE ALU              IM   8  16   0
 16  *GIVES EXPONENT BASE 2 IN #DP1 MSP
 17 001A 810B  CALL    NOP      #LOGF  *FRACTIONAL LOG       IM   0   0  75
 18  *RESULT ALWAYS NEG SINCE LOG OF FRACTIO
 19 001B D400  SIN/AC  NOP      32                           IM   5   0  32
 20  *ARITH R.S. OF FRACT *4
 21 001C 3412  TRANS   NOP      #DP1                             13   0  18
 22 001D 10A0  RSH     TRANS    NOP                               4   5   0
 23 001E 1180  RSH     RSH      NOP                               4  12   0
 24 001F 1180  RSH     RSH      NOP                               4  12   0
 25 0020 1180  RSH     RSH      NOP                               4  12   0
 26 0021 3580  TRANS   RSH      NOP                              13  12   0
 27 0022 4805  #DP1     NOP      TRANS                           18   0   5
 28 0023 9803  MADDR   NOP      #PLOGE2                      IM   6   0   3
```

```
PAGE    3   HARMONIAC ASSEMBLY OF : FFT1LG
 29 0024 3247   MPY      #DP1     MDAT  GET EXP BASE E          12  18   7
 30 0025 4484   #DPO     MPYH     MPYL                          17   4   4
 31 0026 3231   MPY      #DPO     #DPO  *LLEFT LSP              12  17  17
 32 0027 4408   #DPO     NOP      Z1                            17   0   8
 33 0028 A2E0   ADD      #ZRO     0                       IM     8  23   0
 34 0029 5460   #T1      ALU      NOP                           21   3   0
 35  * LINE UP BIN PT WITH EXPONENT
 36 002A C012   CALL     NOP      #DPRT1  #LINRT          IM     0   0  50
 37 002B A2A1   ADD      #T1      1                       IM     8  21   1
 38 002C 5460   #T1      ALU      NOP                           21   3   0
 39 002D BAA5   COMP     #T1      5                       IM    14  21   5
 40 002E C40A   JMP      NOP      #LINRT                  IM     1   0  42
 41 002F 3631   TRANS    #DPO     #DPO                          13  17  17
 42 0030 0418   JMP      NOP      #L/M                          1   0  24
 43 0031 20B2   ADD      TRANS    #DP1                          8   5  18
 44  *
 45  * SINGLE D.P. LOGICAL RIGHT OF #DPO
 46 0032 D401   SIN/AC   NOP      33       #DPRT1         IM     5   0  33
 47 0033 1220   RSH      #DPO     NOP                           4  17   0
 48  *LOST BIT IN AFLG
 49 0034 D402   SIN      NOP      34 SEE LRSB&ROTATE      IM     5   0  34
 50 0035 4580   #DPO     RSH      NOP                           17  12   0
 51 0036 BC09   M1D/AD   NOP      #FFFEM1                 IM    15   0   9
 52 0037 24F1   AND      MDAT     #DPO                          9   7  17
 53 0038 2C23   OR       AFLG     ALU PUT IN CARRY BIT          11   1   3
 54 0039 1060   RSH      ALU      NOP  ROTATE RT                4   3   0
 55 003A A180   ADD      RSH      0                       IM     8  12   0 SOU
 58
 56 003B 0401   JMP      NOP      RETURN                        1   0   1
 57 003C 4403   #DPO     NOP      ALU                           17   0   3
 58  *
 59  * NORM BUILDS INTEGER EXPONENT
 60  * IN ALU, OUT EXP MSP #DP1, FRAC LSP
 61 003D D400   SIN      NOP      32       #NORM          IM     5   0  32
 62 003E 9000   RSH/MC   NOP      0                       IM     4   0   0
 63 003F B061   MPY      ALU      1                       IM    12   3   1
 64 0040 B402   TRANS    NOP      2                       IM    13   0   2
 65 0041 58A0   #T2      TRANS    NOP                           22   5   0
 66 0042 FC17   M1D/AD   NOP      #H4000                  IM    15   0  55
 67 0043 A2EF   ADD      #ZRO     15  EXP CNT             IM     8  23  15
 68 0044 4864   #DP1     ALU      MPY      #NLOOP               18   3   4
 69 0045 24E4   AND      MDAT     MPY                           9   7   4
 70 0046 32C4   MPY      #T2      MPY                           12  22   4
 71 0047 38E3   COMP     MDAT     ALU                           14   7   3
 72 0048 8504   JMP      NOP      #NLOOP UNTIL BIT 14 SET IM     1   0  68
 73 0049 AA41   SUB      #DP1     1                       IM    10  18   1
 74 004A 0401   JMP      NOP      RETURN                        1   0   1
 75  *
 76  *
 77  * LOGF IS FRACTIONAL LOG BASE E
 78 004B 6401   #RTN3             RETURN  #LOGF                 25   0   1
 79 004C 9804   MADDR    NOP      #HLF                    IM     6   0   4
 80 004D 36F2   TRANS    #ZRO     #DP1                          13  23  18
 81 004E 90A0   RSH      TRANS    0                       IM     4   5   0
 82 004F 2987   SUB      RSH      MDAT  USUALLY NEG RESUL      10  12   7
```

```
PAGE    4   HARMONIAC ASSEMBLY OF : FFT1LG
 83 0050 4465  #DPO     ALU     TRANS                          17   3   5
 84 0051 C039  CALL     NOP     #DIV  NEG NUMERATOR      IM    0   0 185
 85 0052 2187  ADD      RSH     MDAT                           8  12   7
 86 0053 3411  TRANS    NOP     #DPO  RESULT DPO LSP          13   0  17
 87 0054 30B1  MPY      TRANS   #DPO  SQUARE IT             12   5  17
 88 0055 9008  RSH/MC   NOP     8  POP MQ               IM    4   0   8
 89 0056 4484  #DPO     MPYH    MPYL  SAVE SQ               17   4   4
  2 0057 9805  MADDR   =NOP    ,#C4                    IM    6   0   5
  3 0058 3227  MPY      #DPO    MDAT  (C4)               12  17   7
  4 0059 2087  ADD      MPYH    MDAT   (C5)               8   4   7
  5 005A 3223  MPY      #DPO    ALU                      12  17   3
  6 005B 2087  ADD      MPYH    MDAT  (C6)               8   4   7
  7 005C 3223  MPY      #DPO    ALU                      12  17   3
  8 005D 2087  ADD      MPYH    MDAT  (C7)               8   4   7
  9 005E 3223  MPY      #DPO    ALU                      12  17   3
 10 005F 34A0  TRANS    TRANS   NOP  RESULT OF DIV       13   5   0
 11 0060 3085  MPY      MPYH    TRANS                    12   4   5
 12 0061 2085  ADD      MPYH    TRANS                     8   4   5
 13 0062 B062  MPY      ALU     2                   IM   12   3   2
 14 0063 9000  MC               0                   IM    4   0   0
 15 0064 0419  JMP      NOP     #RTN3                     1   0  25
 16 0065 4804  #DP1     NOP     MPYL                     18   0   4
 17  *
 18  * PUT COSINE WINDOW ON SIGNAL
 19  * HANNS MEM1 BUFFER, ZEROES MEMO
 20  * SIGNAL INPUT IN MEMO AFTER 12SEPT
 21  * CHECKED OK 31 AUG 1979
 22 0066 6801  #LI/MX   NOP     RETURN  #HANN              26   0   1
 23  *GET PARAMETERS FROM MEM1
 24 0067 8050  CALL     NOP     #PARAS                IM    0   0 272
 25 0068 5804  #T2      NOP     MPYL                     22   0   4
 26 0069 FC12  M1D/AD   NOP     #SGNMSK  GET :7FFF    IM   15   0  50
 27  *CALC PHASE INCR 2PI/N
 28 006A B4E1  TRANS    MDAT    1                     IM   13   7   1
 29 006B 44A5  #DPO     TRANS   TRANS                    17   5   5
 30 006C C039  CALL     NOP     #DIV                  IM    0   0 185
 31 006D 22F6  ADD      #ZRO    #T2                       8  23  22
 32  *RESULT IN #DPO LSP
 33  *START READING SINE FROM -PI/2
 34 006E 90F9  RSH      MDAT    25   PUSH MEMS        IM    4   7  25
 35 006F B580  TRANS    RSH     0                     IM   13  12   0 SOU
111
 36 0070 28A5  SUB      TRANS   TRANS  TO-:3FFF(-PI/2)    10   5   5
 37 0071 4C03  #TEMP    NOP     ALU                      19   0   3
 38 0072 5460  #T1      ALU     NOP                      21   3   0
 39 0073 1BBD  MADDR    #BASE   #BASE                      6  29  29
 40 0074 58A0  #T2      TRANS   NOP   CNTR               22   5   0
 41  * 1ST 1/2 OF DATA
 42 0075 D6A8  SIN      #T1     40                    IM    5  21  40
 43 0076 22ED  ADD      #ZRO    SIN     #HANL01  >         8  23  13
 44 0077 1060  RSH      ALU     NOP                       4   3   0
 45 0078 2993  SUB      RSH     #TEMP  ADD H.S.           10  12  19
 46  * TAKE INPUT DATA FROM MEM 0 (IMG STOR)
 47 0079 3067  MPY      ALU     MDAT                     12   3   7
 48 007A 22B1  ADD      #T1     #DPO UPDATE SIN ADDR      8  21  17
```

```
PAGE    5   HARMONIAC ASSEMBLY OF : FFT1L1
 49 007B 9C80  MDAT    MPYH    0                          IM   7   4   0
 50 007C 5460  #T1     ALU     NOP                             21   3   0
 51 007D A2C2  ADD     #T2     2  UPDATE CNTR             IM   8  22   2
 52 007E 5860  #T2     ALU     NOP                             22   3   0
 53 007F 3876  COMP    ALU     #T2                             14   3  22
 54 0080 C516  JMP     NOP     #HANL01                    IM   1   0 118
 55  * 2ND 1/2 OF DATA
 56 0081 D6A8  SIN     #T1     40                         IM   5  21  40
 57 0082 58A0  #T2     TRANS   NOP  ZERO CNTR                 22   5   0
 58 0083 22ED  ADD     #ZRO    SIN     #HANL02                 8  23  13
 59 0084 1060  RSH     ALU     NOP                             4   3   0
 60 0085 2993  SUB     RSH     #TEMP                          10  12  19
 61 0086 3067  MPY     ALU     MDAT *DATA FROM MO             12   3   7
 62 0087 2AB1  SUB     #T1     #DP0                           10  21  17
 63 0088 9C80  MDAT    MPYH    0                          IM   7   4   0
 64 0089 5460  #T1     ALU     NOP                             21   3   0
 65 008A A2C2  ADD     #T2     2                          IM   8  22   2
 66 008B 5860  #T2     ALU     NOP                             22   3   0
 67 008C 3876  COMP    ALU     #T2                             14   3  22
 68 008D 8423  JMP     NOP     #HANL02                    IM   1   0 131
 69 008E D6A8  SIN     #T1     40                         IM   5  21  40
 70 008F 041A  JMP     NOP     #LI/MX                          1   0  26
 71 0090 9000  RSH     NOP     0                          IM   4   0   0
 72  *
 73  * LOG POWER CALCS FOR SPECTRUM
 74  * DOES 1/2[LOG(X^2+Y^2)]
 75  *
 76 0091 980A  MADDR   NOP     #PWRRET #PWRLOG            IM   6   0  10
 77 0092 1C01  MDAT    NOP     RETURN                          7   0   1
 78 0093 D418  SIN     NOP     56  I6,>=                  IM   5   0  56
 79 0094 A2E0  ADD     #ZRO    0                          IM   8  23   0
 80 0095 6C60  #L2/N1  ALU     NOP  LOOP CNTR                 27   3   0
 81 0096 9801  MADDR           #LINSCF                    IM   6   0   1
 82 0097 5007  #CNT    NOP     MDAT0                           20   0   7
 83 0098 1BBD  MADDR   #BASE   #BASE                           6  29  29
 84 0099 9000  RSH     NOP     0 NORM MEMS                IM   4   0   0
 85 009A 34E0  TRANS   MDAT    NOP                             13   7   0
 86 009B 30E5  MPY     MDAT    TRANS   #PWRLOP                 12   7   5
 87 009C 3447  TRANS   MOADR   MDAT                            13   2   7
 88 009D 5805  #T2     NOP     TRANS                           22   0   5
  2 009E 4484  #DP0    MPY     MPY                             17   4   4
  3  * NOW Y^2
  4 009F 30A7  MPY     TRANS   MDAT                            12   5   7
  5 00A0 C06E  CALL    NOP     #DPADD                     IM   0   0 430
  6 00A1 4884  #DP1    MPY     MPY                             18   4   4
  7 00A2 C039  CALL    NOP     #DIV                       IM   0   0 185
  8 00A3 22F4  ADD     #ZRO    #CNT SCALE TO INTEGER           8  23  20
  9  * COPY LINEAR PWR SPECT TO EXTRA BUFFER
 10  * AT #BASE+N IN MEM 0 SO IT'S AVAILABLE
 11  * FOR RESYNTHESIS
 12 00A4 3416  TRANS           #T2 GET CUR ADDR                13   0  22
 13 00A5 208B  ADD     TRANS   #L2/N1                          8   5  27
 14 00A6 1803  MADDR   NOP     ALU                             6   0   3
 15 00A7 1C11  MDAT    NOP     #DP0                            7   0  17
```

```
PAGE    6   HARMONIAC ASSEMBLY OF : FFT1L2
 16 00A8 22F1  ADD      #ZRO     #DPO                          8  23  17
 17 00A9 8012  CALL     NOP      #ILOGE                   IM   0   0  18
 18 00AA BC07  M1D/AD             #DISCL                   IM  15   0   7
 19  * SCALE DISPLAY BY MPY & DISCARD OF LSP
 20 00AB 30E3  MPY      MDAT     ALU                           12   7   3
 21 00AC 3C16  M1D/AD   NOP      #T2     GET CURR ADDR         15   0  22
 22 00AD 1816  MADDR              #T2                           6   0  22
 23 00AE A362  ADD      #L2/N1   2                        IM   8  27   2 SO
174
 24 00AF 6C60  #L2/N1   ALU      NOP  DOES N/2 PTS             27   3   0
 25 00B0 A0C1  ADD      M1ADR    1  UPDATE                 IM   8   6   1
 26 00B1 9C80  MDAT     MPY      0       SPECT RESULT      IM   7   4   0
 27 00B2 D408  SIN      NOP      40  > ,I6                  IM   5   0  40
 28 00B3 1863  MADDR    ALU      ALU  FOR NEXT FETCH           6   3   3
 29 00B4 3B7B  COMP     #L2/N1   #L2/N1                        14  27  27
 30 00B5 843B  JMP      NOP      #PWRLOP                  IM   1   0 155
 31 00B6 B4E1  TRANS    MDAT     1                        IM  13   7   1
 32  * MUST ZERO OUT REST OF R&I FOR SMOOTH
 33 00B7 8565  JMP      NOP      #MRET                    IM   1   0 453
 34 00B8 980A  MADDR    NOP      #PWRRET                  IM   6   0  10
 35  *
  2 00B9 7001  #RTN1    NOP      RETURN  #DIV  *DIV SUBR       28   0   1
  3  *HI &LO DIVIDEND INPUT #DPO  (D.P.)
  4  *DIVISOR IN ALU, RESULT IN #DPO  LSP
  5  * REMAINDER IN #DPO MSP(SGN OF DVDEND)
  6  * CHECKED OK AUG 1979
  7 00BA 4063  #ALUS    ALU      ALU                           16   3   3
  8 00BB 3223  MPY     =#DPO    ,ALU                *QUOT     12  17   3
  9  *SIGN GOT BY MPY
 10 00BC 9400  SIN      NOP      0                        IM   5   0   0
 11 00BD D500  SIN      NOP      96                       IM   5   0  96
 12  *IF MSP OF DPO IS 0 ,USE DIVISR AS SGN
 13 00BE BA20  COMP     #DPO     0                        IM  14  17   0
 14 00BF B201  MPY      #ALUS    1  DONE IF =0            IM  12  16   1
 15 00C0 D410  SIN/AC   NOP      48          *IM6,<       IM   5   0  48
 16 00C1 B620  TRANS    #DPO     0                        IM  13  17   0
 17 00C2 4C85  #TEMP    MPY      TRANS                         19   4   5
 18  *MAKE DIVISOR POS
 19 00C3 BA00  COMP     #ALUS    0                        IM  14  16   0
 20 00C4 8527  JMP      NOP      #POSDVR                  IM   1   0 199
 21 00C5 28B0  SUB      TRANS    #ALUS                         10   5  16
 22 00C6 4003  #ALUS    NOP      ALU                           16   0   3
 23  *MAKE DVDEND POS
 24 00C7 BA20  COMP     #DPO     0       #POSDVR <        IM  14  17   0
 25 00C8 852B  JMP      NOP      #POSDND                  IM   1   0 203
 26 00C9 D400  SIN/AC   NOP      32                       IM   5   0  32
 27 00CA C12B  CALL     NOP      #DPNEG                   IM   0   0 235
 28  *PACK LSP LEFT BEFORE USING D.P. LEFT
 29 00CB 080B  IOADR    NOP      SOV     #POSDND               2   0  11
 30 00CC 30B1  MPY      TRANS    #DPO * PACK LSP               12   5  17
 31 00CD 4408  #DPO     NOP      Z1                            17   0   8
 32 00CE 50A0  #CNT     TRANS    NOP                           20   5   0
 33 00CF 2A30  SUB      #DPO     #ALUS   #DIVL SUB DIVOR       10  17  16
 34  *RESTORE DIVIDEND IF OV=1
 35 00D0 B820  COMP     AFLG     0                        IM  14   1   0
```

```
PAGE    7   HARMONIAC ASSEMBLY OF : FFT1DV
 36 00D1 8534  JMP      NOP      #REST                          IM   1   0 212
 37 00D2 AE20  OR       #DPO     0                              IM  11  17   0
 38 00D3 2A30  SUB      #DPO     #ALUS  *DO REAL SUBTR              10  17  16
 39 00D4 3071  MPY      ALU      #DPO    #REST                      12   3  17
 40 00D5 4508  #DPO     Z1       Z1                                 17   8   8
 41 00D6 A281  ADD      #CNT     1           *INCR CNT          IM   8  20   1
 42 00D7 5060  #CNT     ALU      NOP                                20   3   0
 43 00D8 B870  COMP     ALU      16                             IM  14   3  16
 44 00D9 852F  JMP      NOP      #DIVL                          IM   1   0 207
 45 00DA D400  SIN/AC   NOP      32                             IM   5   0  32
 46 00DB D413  SIN/AC   NOP      51                             IM   5   0  51
 47   * MOVE OV(0) INTO MSP OF REM
 48 00DC 1220  RSH/MC   #DPO     NOP                                4  17   0
 49   *SET REM SIGN SAME AS DVDEND
 50 00DD 3413  TRANS    NOP      #TEMP  GET DVDND                  13   0  19
 51 00DE B8A0  COMP     TRANS    0  <                           IM  14   5   0
 52 00DF C524  JMP      NOP      #POSR                          IM   1   0 228
 53 00E0 4580  #DPO     RSH      NOP       *REM POS                 17  12   0
 54 00E1 B620  TRANS    #DPO     0                              IM  13  17   0
 55 00E2 28A5  SUB      TRANS    TRANS                              10   5   5
 56 00E3 4460  #DPO     ALU      NOP                                17   3   0
 57 00E4 BA60  COMP     #TEMP    0        #POSR                 IM  14  19   0
 58 00E5 C529  JMP      NOP      #POSQ                          IM   1   0 233
 59   *NEGATE QUOT IF REQ'D
 60 00E6 B400  TRANS    NOP      0                              IM  13   0   0
 61 00E7 28B1  SUB      TRANS    #DPO                               10   5  17
 62 00E8 4403  #DPO     NOP      ALU                                17   0   3
 63 00E9 041C  JMP      NOP      #RTN1   #POSQ                       1   0  28
 64 00EA D404  SIN/AC   NOP      36                             IM   5   0  36
 65   *  DOUBLE PRECISION NEGATE OF #DPO
 66 00EB D400  AC                32       #DPNEG               IM   5   0  32
 67 00EC FC12  M1D/AD   NOP      #SGNMSK     *GET SGNMSK       IM  15   0  50
 68 00ED B620  TRANS    #DPO     0                              IM  13  17   0
 69 00EE 28B1  SUB      TRANS    #DPO        *NEGATE LSP            10   5  17
 70 00EF 24E3  AND      MDATA    ALU         *CLR SGN                9   7   3
 71 00F0 4403  #DPO     NOP      ALU         *SAV LSP               17   0   3
 72 00F1 38B1  COMP     TRANS    #DPO                               14   5  17
 73   *IF LSP=/0 MAKE MSP COMPL, NOT NEGATE
 74 00F2 C535  JMP      NOP      #NOTO                          IM   1   0 245
 75 00F3 28A5  SUB      TRANS    TRANS       *NEG MSP               10   5   5
 76 00F4 0401  JMP      NOP      RETURN                             1   0   1
 77 00F5 4460  #DPO     ALU      NOP      #NOTO *SAV MSP            17   3   0
 78 00F6 AA21  SUB      #DPO     1                *COMPLEM      IM  10  17   1
 79 00F7 0401  JMP      NOP      RETURN                             1   0   1
 80 00F8 4460  #DPO     ALU      NOP                                17   3   0
  2   * FILE FFT1EX
  3   *  EXPONENTIAL BASE TWO
  4   * LEVL ZERO SUBROUTINE(IN ALU, OUT MPYL)
  5 00F9 4063  #ALUS   =ALU     ,ALU     #EXP2                     16   3   3
  6 00FA B401  TRANS   =NOP     ,1                             IM  13   0   1
  7 00FB B0A1  MPY     =TRANS   ,1                             IM  12   5   1
  8 00FC B860  COMP    =ALU     ,0       GEN CASE              IM  14   3   0
  9 00FD 8440  JMP     =NOP     ,#MPYM0  GEN CASE              IM   1   0 256
 10 00FE B402  TRANS   =NOP     ,2                             IM  13   0   2
```

```
PAGE    8   HARMONIAC ASSEMBLY OF : FFT1EX
 11 00FF 0401   JMP     =NOP     ,RETURN GEN CASE (REQD)          1   0   1
 12 0100 AA01   SUB      #ALUS    1        #MPYMO           IM   10  16   1
 13 0101 30A4   MPY     =TRANS   ,MPY                             12   5   4
 14 0102 4060   #ALUS   =ALU     ,NOP                             16   3   0
 15 0103 B860   COMP    =ALU     ,0                           IM  14   3   0
 16 0104 8440   JMP     =NOP     ,#MPYMO                      IM   1   0 256
 17 0105 22E4   ADD      #ZRO     MPYL                             8  23   4
 18 0106 0401   JMP     =NOP     ,RETURN                          1   0   1
 19  * SCL2 - DIVIDES BY A PWR OF 2
 20  * INPUT IN RSH (ALREADY/2 ONCE)
 21  * WITH CNT OF TOTAL SHIFTS REQ'D IN #T2
 22  * OUTPUT IN ALU
 23 0107 AAC1   SUB      #T2      1        #SCL2 T2 MSP CN  IM   10  22   1
 24 0108 4063   #ALUS    ALU      ALU                              16   3   3
 25 0109 AA01   SUB      #ALUS    1        #SHLOP           IM   10  16   1
 26 010A 4063   #ALUS    ALU      ALU                              16   3   3
 27 010B 3AE3   COMP     #ZRO     ALU                              14  23   3
 28 010C 8449   JMP      NOP      #SHLOP                       IM   1   0 265
 29 010D 1180   RSH      RSH      NOP                              4  12   0
 30 010E 0401   JMP      NOP      RETURN                           1   0   1
 31 010F A180   ADD      RSH      0  FOR EASY ACCESS           IM   8  12   0 SC
271
  2  * FFT BEGINS
  3  * ARGUMENTS FROM MEM1 (0-> )
  4  * ARE: BASER,BASEI,M,L(^2)
  5  * BASES MUST BE SAME IN EA MEM(H400)
  6  * M IS TOT PTS (PWR OF 2),L NON 0 PTS
  7 0110 5801   #T2       NOP      RETURN  #PARAS                   22   0   1
  8 0111 D400   SIN/AC    NOP      32        *IM6,=           IM    5   0  32
  9 0112 9008   RSH/MC    NOP      8         POPM1            IM    4   0   8
 10 0113 BC00   M1D/AD    NOP      #ARGS                      IM   15   0   0
 11 0114 74E0   #BASE    =MDAT    ,NOP                             29   7   0
 12 0115 A0E0   ADD       MDAT    0                          IM    8   7   0
 13 0116 7463   #BASE     ALU     ALU ***AVOID M1 ERR              29   3   3
 14 0117 34E0   TRANS    =MDAT    ,NOP                             13   7   0
 15 0118 6005   #L/M     =NOP     ,TRANS                           24   0   5
 16 0119 60E0   #L/M     =MDAT    ,NOP                             24   7   0
 17  * CALC NON ZERO PTS (PWR OF 2)
 18 011A 2317   ADD       #L/M    #ZRO                             8  24  23
 19 011B C139   CALL      NOP     #EXP2                      IM    0   0 249
 20 011C 3404   TRANS     NOP     MPYL                             13   0   4
 21 011D 6CA0   #L2/N1    TRANS   NOP                              27   5   0
 22  * CALC TOT PTS (PWR OF 2)
 23 011E 22F8   ADD       #ZRO    #L/M                             8  23  24
 24 011F C139   CALL      NOP     #EXP2                      IM    0   0 249
 25 0120 6C04   #L2/N1    NOP     MPYL                             27   0   4
 26 0121 0416   JMP       NOP     #T2                              1   0  22
 27 0122 9000   RSH       NOP     0                          IM    4   0   0
 28  *
 29 0123 980D   MADR                #FFTRET #FFT             IM    6   0  13
 30 0124 1C01   MDAT                RETURN                           7   0   1
 31 0125 8050   CALL      NOP     #PARAS                     IM    0   0 272
 32  * SET UP OUTER LOOP
 33 0126 A2E1   ADD       #ZRO    1                          IM    8  23   1
 34 0127 7803   #L1/0     NOP     ALU                              30   0   3
```

```
PAGE    9   HARMONIAC ASSEMBLY OF : FFT1A
 35 0128 3418  TRANS   =NOP     ,#L/M     #LOLOO                13   0  24
 36 0129 28BE  SUB     =TRANS   ,#L1/O                        10   5  30
 37 012A C139  CALL    =NOP     ,#EXP2                   IM    0   0 249
 38 012B 6804  #LI/MX  =NOP     ,MPYL                          26   0   4
 39 012C 5804  #T2     =NOP     ,MPYL                          22   0   4
 40 012D B402  TRANS    NOP      2                        IM   13   0   2
 41 012E 30A4  MPY     =TRANS   ,MPYL                          12   5   4
 42 012F FC12  M1D/AD   NOP      #SGNMSK                  IM   15   0  50
 43 0130 3404  TRANS   =NOP     ,MPYL                          13   0   4
 44 0131 68A0  #LI/MX  =TRANS   ,NOP                           26   5   0
 45   * SET UP 2PI IN D. P. : 1 7FFF
 46 0132 2357  ADD     =#LI/MX  ,#ZRO                          8  26  23
 47 0133 B4E1  TRANS    MDAT     1                        IM   13   7   1
 48 0134 44A5  #DPO     TRANS    TRANS                          17   5   5
 49 0135 C039  CALL     NOP      #DIV                     IM    0   0 185
 50 0136 D410  SIN/AC   NOP      48   * <                 IM    5   0  48
 51   *
 52   *PRUNING TEST WAS REMOVED TO SAVE SPACE
 53   *
 54 0137 B6E0  TRANS   =#ZRO    ,0         #I3           IM   13  23   0
 55 0138 50A5  #CNT    =TRANS   ,TRANS                         20   5   5
 56 0139 22F1  ADD      #ZRO     #DPO                           8  23  17
 57 013A 7C03  #RTN2    NOP      ALU   TEMP SAVE             31   0   3
 58 013B 9000  RSH/MC  =NOP     ,0                       IM    4   0   0
 59 013C 329F  MPY      #CNT     #RTN2     #LMLOO               12  20  31
 60 013D BC0B  M1D/AD   NOP      #PI/2                    IM   15   0  11
 61 013E 3404  TRANS   =NOP     ,MPY                           13   0   4
 62 013F A0A8  ADD      TRANS    8    TO ROUND SINE/COS   IM    8   5   8
 63 0140 D460  SIN     =ALU     ,32                       IM    5   3  32
 64 0141 20E3  ADD      MDAT     ALU   *ADDS PI/2 FOR COS       8   7   3
 65 0142 4C0D  #TEMP    NOP      SIN                            19   0  13
 66 0143 D468  SIN     =ALU     ,40                       IM    5   3  40
 67 0144 2357  ADD     =#LI/MX  ,#ZRO   FOR INNER LOOP         8  26  23
 68 0145 340D  TRANS   =NOP     ,SIN                           13   0  13
 69 0146 4CA0  #TEMP   =TRANS   ,NOP                           19   5   0
 70   * SET UP INNER LOOP
 71 0147 7860  #L1/O   =ALU     ,NOP                           30   3   0
 72 0148 23D4  ADD     =#L1/O   ,#CNT                          8  30  20
 73 0149 3740  TRANS   =#LI/MX  ,NOP       #L1LOO              13  26   0
 74 014A 2865  SUB     =ALU     ,TRANS                         10   3   5
 75 014B 4063  #ALUS   =ALU     ,ALU                           16   3   3
 76 014C 23A3  ADD      #BASE    ALU                            8  29   3
 77 014D 4463  #DPO     ALU      ALU                           17   3   3
 78 014E 207A  ADD      ALU      #LI/MX                         8   3  26
 79 014F 4863  #DP1     ALU      ALU                           18   3   3
 80 0150 1A51  MADDR   =#DP1    ,#DPO                          6  18  17
 81 0151 34E7  TRANS   =MDAT    ,MDAT                         13   7   7
 82 0152 1A32  MADDR   =#DPO    ,#DP1                          6  17  18
 83 0153 28E5  SUB     =MDAT    ,TRANS                        10   7   5
 84 0154 5463  #T1     =ALU     ,ALU                          21   3   3
 85 0155 20E5  ADD     =MDAT    ,TRANS                         8   7   5
 86 0156 1C60  MDAT    =ALU     ,NOP                           7   3   0
 87 0157 28A7  SUB     =TRANS   ,MDAT                         10   5   7
 88 0158 4063  #ALUS   =ALU     ,ALU                          16   3   3
```

```
PAGE   10   HARMONIAC ASSEMBLY OF : FFT1A
 89 0159 20A7  ADD     =TRANS  ,MDAT                              8   5   7
  2 015A 1811  MADDR    NOP     #DPO                              6   0  17
  3 015B 3275  MPY      #TEMP   #T1                              12  19  21
  4 015C 1C03  MDAT     NOP     ALU                               7   0   3
  5 015D A080  ADD      MPYH    0                          IM     8   4   0
  6 015E 3213  MPY      #ALUS   #TEMP                            12  16  19
  7 015F 1A52  MADDR    #DP1    #DP1 ACTS AS MPY DELAY            6  18  18
  8 0160 2083  ADD      MPYH    ALU                               8   4   3
  9 0161 32B3  MPY      #T1     #TEMP                            12  21  19
 10 0162 1C60  MDAT     ALU     NOP                               7   3   0
 11 0163 A080  ADD      MPYH    0                          IM     8   4   0
 12 0164 3270  MPY      #TEMP   #ALUS                            12  19  16
 13 0165 3740  TRANS   =#LI/MX ,NOP  ACTS AS MPY DELAY           13  26   0
 14 0166 2883  SUB      MPYH    ALU                              10   4   3
 15 0167 1C03  MDAT    =NOP     ,ALU                              7   0   3
 16   * END INNER LOOP CALC'S
 17 0168 23C5  ADD     =#L1/O   ,TRANS                            8  30   5
 18 0169 7860  #L1/O   =ALU     ,NOP                             30   3   0
 19   *CHECK INNER LOOP STAGE
 20 016A 387B  COMP    =ALU     ,#L2/N1                          14   3  27
 21 016B 8549  JMP     =NOP     ,#L1LOO                    IM     1   0 329
 22 016C 23D4  ADD      #L1/O   #CNT                             8  30  20
 23 016D A281  ADD     =#CNT    ,1                         IM     8  20   1
 24 016E 5063  #CNT    =ALU     ,ALU                             20   3   3
 25   *CHECK MIDDLE LOOP STAGE
 26 016F D400  SIN/AC   NOP     32     *=&IM6              IM     5   0  32
 27 0170 3876  COMP    =ALU     ,#T2                             14   3  22
 28 0171 C45C  JMP     =NOP     ,#LMLOO                    IM     1   0 316
 29 0172 341E  TRANS   =NOP     ,#L1/O                           13   0  30
 30 0173 A0A1  ADD     =TRANS   ,1                         IM     8   5   1
 31 0174 7803  #L1/O   =NOP     ,ALU                             30   0   3
 32 0175 A861  SUB     =ALU     ,1                         IM    10   3   1
 33 0176 3878  COMP    =ALU     ,#L/M                            14   3  24
 34 0177 C448  JMP     =NOP     ,#LOLOO                    IM     1   0 296
 35   * NOW FFT FINISHED - DO REORDERING
 36   * DOES SOFTWARE REVERSE BIT REORDER
 37   * & SCALE DOWN BY SQRT NON-0 NO. PTS.
 38   * (THIS IS APPROX GROWTH RATE OF DATA)
 39 0178 9300  RSH      #L/M    0                          IM     4  24   0 SOU
376
 40 0179 2197  ADD      RSH     #ZRO                              8  12  23
 41 017A 1BBD  MADDR    #BASE   #BASE                             6  29  29
 42 017B 5863  #T2       ALU     ALU                             22   3   3
 43 017C 9000  MC                0        #RLOOP            IM     4   0   0
 44 017D A0C0  ADD      M1ADR   0                          IM     8   6   0
 45 017E 4463  #DP0      ALU     ALU                SAV NORM      17   3   3
 46 017F 807D  CALL       NOP     #REVBIT                   IM     0   0 413
 47 0180 D418  AC                 56  >=I6                  IM     5   0  56
 48 0181 3A71  COMP     #TEMP   #DPO  N>R->SKIP                 14  19  17
 49 0182 8475  JMP              #SKIP                     IM     1   0 405
 50   * DO THE EXCH OF REV/NORM
 51 0183 90E0  RSH        MDAT    0                          IM     4   7   0
 52 0184 8047  CALL               #SCL2                     IM     0   0 263
 53 0185 4980  #DP1      RSH     NOP  SCLD NORM 1                18  12   0
 54 0186 3407  TRANS              MDAT                            13   0   7
```

```
PAGE   11   HARMONIAC ASSEMBLY OF : FFT1B
 55 0187 8047  CALL                #SCL2                          IM   0   0 263
 56 0188 10A0  RSH      TRANS      NOP                                4   5   0
 57 0189 4803  #DP1     NOP        ALU SCLD NORM 0                   18   0   3
 58 018A 1A73  MADDR    #TEMP      #TEMP SET REV                      6  19  19
 59 018B 8047  CALL                #SCL2                          IM   0   0 263
 60 018C 90E0  RSH      MDAT       0                              IM   4   7   0
 61 018D 7060  #RTN1    ALU        NOP  SAVE SCLD REV 1              28   3   0
 62 018E 3407  TRANS               MDAT                              13   0   7
 63 018F 8047  CALL                #SCL2                          IM   0   0 263
 64 0190 10A0  RSH      TRANS      NOP                                4   5   0
 65 0191 7003  #RTN1    NOP        ALU SAV SCLD REV 0                28   0   3
 66 0192 1E52  MDAT     #DP1       #DP1 PUT NORMS AT REV              7  18  18
 67 0193 1A31  MADDR    #DPO       #DPO  SET NORM ADDR                6  17  17
 68 0194 1F9C  MDAT     #RTN1      #RTN1 REVS TO NORMS                7  28  28
 69 0195 28DD  SUB      M1ADR      #BASE   #SKIP                     10   6  29
 70 0196 A061  ADD      ALU        1                              IM   8   3   1
 71 0197 9008  MC                 8  POP                         IM   4   0   8
 72 0198 387B  COMP     ALU        #L2/N1  END?                      14   3  27
 73 0199 C55C  JMP                 #RLOOP                         IM   1   0 380
 74 019A 24E7  AND      MDAT       MDAT                               9   7   7
 75 019B 8565  JMP                 #MRET                          IM   1   0 453
 76 019C 980D  MADR                #FFTRET                        IM   6   0  13
 77   *
 78 019D 32F7  MPY      #ZRO       #ZRO     #REVBIT                  12  23  23
 79 019E 28DD  SUB      M1ADR      #BASE                             10   6  29
 80 019F 4C80  #TEMP    MPY        NOP                               19   4   0
 81 01A0 D402  AC                  34  LRSB TO AFLG               IM   5   0  34
 82 01A1 1060  RSH      ALU        NOP                                4   3   0
 83 01A2 2C24  OR       AFLG       MPYL     #SWLOP                   11   1   4
 84 01A3 B062  MPY      ALU        2                              IM  12   3   2
 85 01A4 A261  ADD      #TEMP      1                              IM   8  19   1
 86 01A5 4C60  #TEMP    ALU        NOP                               19   3   0
 87 01A6 3878  COMP     ALU        #L/M                              14   3  24
 88 01A7 C462  JMP      NOP        #SWLOP                         IM   1   0 418
 89 01A8 1180  RSH      RSH        NOP                                4  12   0
 90 01A9 22E4  ADD      #ZRO       MPYL                               8  23   4
 91 01AA 9060  RSH      ALU        0                              IM   4   3   0
 92 01AB 219D  ADD      RSH        #BASE                              8  12  29
 93 01AC 0401  JMP                 RETURN                             1   0   1
 94 01AD 4C63  #TEMP    ALU        ALU  REVERSED ADDR                19   3   3
  2   * D.P. ADD OF DPO &DP1 TO DPO, OV ERR
  3 01AE 9000  RSH/MC  =NOP      ,0         #DPADD                 IM   4   0   0
  4 01AF FC12  M1D/AD  =NOP      ,#SGNMSK                          IM  15   0  50
  5 01B0 24F1  AND     =MDAT     ,#DPO                              9   7  17
  6 01B1 D400  SIN/AC  =NOP      ,32                            IM   5   0  32
  7 01B2 2072  ADD     =ALU      ,#DP1      ADDLO ORD               8   3  18
  8 01B3 24E3  AND     =MDAT1    ,ALU       CLEAR SIGN              9   7   3
  9 01B4 4403  #DPO    =NOP      ,ALU       LOSUM                  17   0   3
 10 01B5 3620  TRANS   =#DPO     ,NOP                              13  17   0
 11 01B6 2025  ADD     =AFLG     ,TRANS     ADD CARRY               8   1   5
 12 01B7 D400  SIN/AC  =NOP      ,32                            IM   5   0  32
 13 01B8 2243  ADD     =#DP1     ,ALU       HISUM                   8  18   3
 14 01B9 0401  JMP     =NOP      ,RETURN    RESLT DPO               1   0   1
 15 01BA 4460  #DPO    =ALU      ,NOP                              17   3   0
```

```
PAGE   12   HARMONIAC ASSEMBLY OF : FFT1CL
  2  *
  3  * HI LEV SPECTRUM CALLERS
  4  *
  5 01BB 980B  MADDR  =NOP    ,#PSPR   #PSPECT          IM   6   0  11
  6 01BC 1C01  MDAT   =NOP    ,RETURN  SAVERETURN            7   0   1
  7 01BD C106  CALL   =NOP    ,#HANN                    IM   0   0 102
  8 01BE 0000  NOP    =       ,                              0   0   0
  9 01BF C043  CALL   =NOP    ,#FFT                     IM   0   0 291
 10 01C0 0000  NOP    =       ,                              0   0   0
 11 01C1 8031  CALL   =NOP    ,#PWRLOG                  IM   0   0 145
 12 01C2 0000  NOP                                      IM   0   0   0
 13 01C3 816A  CALL            #SINSUM                  IM   0   0 458
 14 01C4 980B  MADDR  =NOP    ,#PSPR                    IM   6   0  11
 15 01C5 B401  TRANS   NOP     1        #MRET            IM  13   0   1
 16 01C6 BCAA  M1D/AD  TRANS   #RDYFL2                  IM  15   5  10
 17 01C7 9400  SIN     NOP     0                        IM   5   0   0
 18 01C8 0407  JMP     NOP     MDAT                           1   0   7
 19 01C9 D500  SIN     NOP     96                       IM   5   0  96
 20  *       =       ,
 21  * SMOOTHED SPECT USED TO BE HERE
 22  *   HAS BEEN MOD'D FOR RESYNTH
  2  * RUNNING SINUSOID SUMMATION SYNTHESIS
  3  *       =       ,
  4  * USES DOUBLE PRECISION PHASE ADDITION
  5  * FOR HIGH ACCURACY PITCH CONTROL.
  6  * AMPL SUMMATION IS S.PRECISION.
  7  * RUNS AS SUBROUTINE THAT CALCS NHOP
  8  * PTS OF WAVEFORM ON EA.  CALL(STACKED)
  9 01CA 9008  MC              8        #SINSUM  ENTRY   IM   4   0   8
 10 01CB 9400  AC              0  IM7 SET                 IM   5   0   0
 11  * POP THE CNT/ADDR PARAMETERS
 12  **SET UP TO DO "NHOP" SAMPLES
 13 01CC BC15  M1D/AD  NOP     #ENDRES                   IM  15   0  21
 14 01CD 54E0  #T1     MDAT    NOP          PTS TO END        21   7   0
 15  * SET UP PTR TO RESULTS STACK IN M1
 16 01CE 58E0  #T2     MDAT    NOP                            22   7   0
 17 01CF FC1C  M1D/AD          #NSINES #SAMLOP           IM  15   0  60
 18 01D0 E1F7  ADD     MDAT    #PHASES                   IM   8   7 119
 19 01D1 5063  #CNT    ALU     ALU                            20   3   3
 20 01D2 B4E1  TRANS   MDAT    1 IS OVERALL AMP          IM  13   7   1
 21 01D3 A460  AND     ALU     0                         IM   9   3   0
 22 01D4 4063  #ALUS   ALU     ALU  USED FOR SAMPLE            16   3   3
 23 01D5 D917  MADDR           #PHASES                   IM   6   0 119
 24 01D6 9009  MC              9        POP M1,PUSH 0    IM   4   0   9
 25 01D7 20E7  ADD     MDAT    MDAT ADD LSPS                  8   7   7
 26 01D8 1C03  MDAT    NOP     ALU      #SINLOP  *            7   0   3
 27 01D9 0000  NOP                        WAIT ON M0 PU       0   0   0
 28 01DA 2027  ADD     AFLG    MDAT              *            8   1   7
 29 01DB 20E3  ADD     MDAT    ALU       ADD MSPS *           8   7   3
 30 01DC 1C03  MDAT            ALU  PUSH MSP                 7   0   3
 31 01DD A068  ADD     ALU     8 ROUND FOR SINE          IM   8   3   8
 32  *
 33  * NOTE 65DB S/N POSS W 11B*1024 SINE
 34 01DE 9478  SIN/AC  ALU     24  ROV&SEE OV & >=       IM   5   3  24
```

```
PAGE   13   HARMONIAC ASSEMBLY OF : FFT1DS
 35  * MPY BY AMPL OF THIS COMPONENT
 36 01DF 30ED  MPY      MDAT     SIN                                  12   7  13
 37 01E0 2090  ADD      MPYH     #ALUS  UPDATE SAMPLE                   8   4  16
 38 01E1 4063  #ALUS    ALU      ALU                                  16   3   3
 39 01E2 3854  COMP     MOADDR   #CNT                                 14   2  20
 40 01E3 8578  JMP               #SINLOP                          IM   1   0 472
 41 01E4 20E7  ADD      MDAT     MDAT  NEXT LSP ADD                    8   7   7
 42  * 12 INSTRUCTION INNER LOOP
 43  * NOW CHECK IF ZERO CROSSING SO CAN
 44  * CHANGE PARAMETERS WITHOUT CLICK
 45 01E5 BA21  COMP     #DPO     1  EQUIV TO >0                    IM  14  17   1
 46 01E6 C56A  JMP               #NEG  EXEC IF <=0 NOW            IM   1   0 490
 47 01E7 0000  NOP                                                    0   0   0
 48 01E8 C574  JMP               #CONTIN  EXCE IF IS >0           IM   1   0 500
 49 01E9 0000  NOP                                                    0   0   0
 50 01EA BA00  COMP     #ALUS    0        #NEG  >=TEST             IM  14  16   0
 51 01EB C574  JMP               #CONTIN #CONTI  IF<0 LA          IM   1   0 500
 52  * BLOCK MOVE OF CONTROL PARAMS
 53 01EC FC1C  M1D/AD            #NSINES                          IM  15   0  60
 54 01ED 9819  MADDR             #PARIN (HOST INPUT)              IM   6   0  25
 55 01EE 9018  MC                24  PUSH1,POP0                   IM   4   0  24
 56 01EF 22E7  ADD      #ZRO     MDAT                                 8  23   7
 57 01F0 1C60  MDAT     ALU      NOP      #MOVP                       7   3   0
 58 01F1 F956  COMP     MOADR    #ENDPIN                          IM  14   2 118
 59 01F2 C570  JMP               #MOVP                            IM   1   0 496
 60 01F3 22E7  ADD      #ZRO     MDAT                                 8  23   7
 61  * (20 PARAM SETS ALLOWED)
 62  * MPY BY OVERALL AMPL
 63 01F4 3205  MPY      #ALUS    TRANS   #CONTIN  AMPL                12  16   5
 64 01F5 A2C1  ADD      #T2      1                                IM   8  22   1
 65 01F6 3C83  M1D/AD   MPYH     ALU                                 15   4   3
 66 01F7 4480  #DPO     MPYH     NOP                                 17   4   0
 67 01F8 5863  #T2      ALU      ALU                                 22   3   3
 68 01F9 36A3  TRANS    #T1      ALU                                 13  21   3
 69 01FA 38A5  COMP     TRANS    TRANS  REACHED END ?                  14   5   5
 70 01FB 856F  JMP               #SAMLOP                          IM   1   0 463
 71 01FC 9009  MC                9 RESTORE MEM STATUS             IM   4   0   9
 72 01FD 0401  JMP               RETURN                               1   0   1
 73 01FE 0000  NOP                                                    0   0   0
 74  DATA FILE FOR MAIN MEM BEGINS  @M1,20
 75 0014 0100 256       #NHOP
 76 0015 0900 2304      #ENDRES
 77 0016 0800 2048      #RPTR
 78  * END RESULTS AREA OF M1
 79 0017 0014 20        #LITTLE
 80  DATA FILE FOR MAIN MEM BEGINS  @M0,25  PARAMETERS INPUT FILE
 81 0019 0002 2         #PARIN
 82 001A 07D0 2000
 83 001B 0000 0
 84 001C 00C8 200
 85 001D 01F4 500
 86  *ENDS AT 118
 87  DATA FILE FOR MAIN MEM BEGINS  @M0,118   RUNNING PHASE TABLE
 88 0076 0000 0         #ENDPIN
```

```
PAGE   14   HARMONIAC ASSEMBLY OF : FFT1DS
 89 0077 0000 0        #PHASES
 90 0078 0000 0                  MSP
 91   DATA FILE FOR MAIN MEM BEGINS  @M1,60 ** PARAMETERS WORKING FILE
 92 003C 0002 2        #NSINES  TWICE NO. OF SINES REQ
 93 003D 03E8 1000     #AMPLA   OVERALL AMPL
 94 003E 0000 0        #PHIAMP   PHASE INC LSP
 95 003F 07D0 2000               PHASE INC MSP
 96 0040 1388 5000               AMPL OF THIS COMPONENT
 97   * THREE WORDS DESCRIBE EA COMPONENT
 5 WARNINGS TOTAL
 0 ERRORS TOTAL
```

APPENDIX II

```
 PAGE    1   HARMONIAC ASSEMBLY OF : FFT1MN  MAIN CALLING PROG FOR FFT
LINE ADDR MEM.     SOURCE CODE          #LABELS(JMP) #CMTS     DECODED OB


  2  * N PT TRANSFORM
  3  * FOR ANY "N" (SOFT BIT REVERSE)
  4  * -NOT AS FAST AS HARD BIT REVERSE
  5  * LOG/LIN PWR SPECTRUM
  6  * & SMOOTHED PWR SPECT & PITCH AVAIL.
  7  * SET UP FOR 12BIT INPUT,10 BIT DISPLAY
  8  * 19TH OCT 1979 SOFTWARE XFORM (12B)
  9  * PHIL CONNOR
 10  *
 11  * FIRST A "PROTECT" PROG TO OPERATE
 12  * DURING FILL UP OF HARM'S MEMS
 13   NEW PROGRAM SEGMENT BEGINS  @PM,510
 14 01FE C57E  JMP              #HERE   #HERE          IM   1    0 510
 15 01FF 0000  NOP                                          0    0   0
 16   NEW PROGRAM SEGMENT BEGINS  @PM,0
 17 0000 C57E  JMP              #HERE                  IM   1    0 510
 18 0001 0000  NOP                                          0    0   0
 19  *
 20  *#ALUS #DP0 #DP1 #TEMP #CNT
 21  * #T1 #T2 #ZRO
 22   DATA FILE FOR MAIN MEM BEGINS  @M1,0
 23 0000 0200 512     #ARGS
 24 0001 0200 512
 25 0002 000A 10      #L
 26 0003 000A 10      #M
 27 0004 0000 0       #PITCHP    PITCH PERIOD RESULT
 28 0005 0000 0       #PITCH
 29 0006 0005 5       #SMOOTH
 30 0007 0200 512     #DISCL  FOR A  7 BIT DISPLAY
 31 0008 000F 15      #VTHRSH
 32 0009 FFFE -2      #FFFEM1
 33 000A 0000 0       #RDYFL2
 34 000B 3FFF 16383   #PI/2
 35   DATA FILE FOR MAIN MEM BEGINS  @M1,50
 36 0032 7FFF 32767   #SGNMSK  THESE 2 RESTORED
 37 0033 0000 0       #RDYFLG  BY FFTHD(CALLER)
 38 0034 8000 -32768  #8000
 39  * BEWARE ,BUGFLG IS AT 53
 40   DATA FILE FOR MAIN MEM BEGINS  @M1,54
 41 0036 3796 14230   #PLOG1E
 42 0037 4000 16384   #H4000
 43   DATA FILE FOR MAIN MEM BEGINS  @M0,0
 44 0000 2710 10000   #SAMRAT
 45 0001 FFFE -2      #FFFE
 46 0002 588A 22714   #PLOGE2
 47 0003 4000 16384   #HLF
 48 0004 0E39 3641    #C4
 49 0005 1249 4681    #C5
 50 0006 1999 6553    #C6
 51 0007 2AAA 10922   #C7
 52 0008 0400 1024    #P1024
 53 0009 0000 0       #PWRRET
 54 000A 0000 0       #PSPR
 55 000B 0000 0       #PSSPR
 56   NEW PROGRAM SEGMENT BEGINS  @PM,0
```

```
PAGE    2   HARMONIAC ASSEMBLY OF : FFT1MN
 57  * MOST OF PROGRAM USES IM6 MODE TO
 58  * GIVE ACCESS TO ALL REGS & 0-63 IMMEDS
 59 0000 9000  RSH/MC =NOP     ,0        #START              IM   4   0   0
 60 0001 9400  SIN/AC  NOP      0  SETS IM7                  IM   5   0   0
 61 0002 D500  SIN/AC  NOP      96  SETS NOT EQU,I6          IM   5   0  96
 62 0003 A460  AND     ALU      0  GENERATE A ZERO          IM   9   3   0
 63 0004 9C63  #ZRO    ALU      ALU                              23   3   3
 64 0005 FC13  M1D/AD =NOP     ,#RDYFLG #WAIT               IM  15   0  51
 65 0006 B8E1  COMP   =MDAT    ,1                           IM  14   7   1
 66 0007 C053  CALL    NOP      #FFT                         IM   0   0 307
 67 0008 FC13  M1D/AD  NOP      #RDYFLG   FOR NEXT IN T      IM  15   0  51
 68 0009 B8E5  COMP    MDAT     5                            IM  14   7   5
 69 000A 8171  CALL    NOP      #PSPECT  DOES SPECTRUM       IM   0   0 465
 70 000B FC13  M1D/AD  NOP      #RDYFLG                      IM  15   0  51
 71 000C B8E6  COMP    MDAT     6                            IM  14   7   6
 72 000D 8178  CALL    NOP      #SSPECT  SMOOTHED SPECT      IM   0   0 475
 73 000E FC13  M1D/AD  NOP      #RDYFLG                      IM  15   0  51
 74 000F 9000  RSH/MC  NOP      0          MEM NORM          IM   4   0   0
 75 0010 BC0A  M1D/AD  NOP      #RDYFL2                      IM  15   0  10
 76 0011 B8E1  COMP    MDAT     1                            IM  14   7   1
 77 0012 FEF3  M1D/AD =#ZRO    ,#RDYFLG                     IM  15  23  51
 78 0013 8400  JMP    =NOP     ,#START                      IM   1   0   0
 79 0014 BEEA  M1D/AD  #ZRO     #RDYFL2                      IM  15  23  10
 80  *
  2  * 20 JULY 1979 ,
  3 0015 6001  #LI/MX =NOP     ,RETURN  #ILOG10                 24   0   1
  4  * INPUT IN ALU,OUT IN ALU
  5 0016 801B  CALL    =NOP     ,#ILOGE                      IM   0   0  27
  6 0017 FC16  M1D/AD  NOP      #PLOG1E                      IM  15   0  54
  7 0018 30E3  MPY     MDAT     ALU                              12   7   3
  8 0019 0418  JMP     NOP      #LI/MX                           1   0  24
  9 001A A080  ADD     MPYH     0                            IM   8   4   0
 10  *
 11  * INTEGER LOG BASE E
 12 001B 6401  #L/M    NOP      RETURN  #ILOGE                   25   0   1
 13  *IN ALU,OUT ALU
 14  *RETURN ZERO IF <=ZERO IN
 15 001C 4063  #ALUS   ALU      ALU                              16   3   3
 16 001D D408  AC               40  >,I6                     IM   5   0  40
 17 001E B860  COMP     ALU     0  >0?                         IM  14   3   0
 18 001F 0419  JMP              #L/M  SKIP THRU IF TRUE          1   0  25
 19 0020 A2E0  ADD      #ZRO    0                            IM   8  23   0
 20 0021 8106  CALL     NOP     #NORM                        IM   0   0  70
 21 0022 A200  ADD      #ALUS   0  RESTORE ALU               IM   8  16   0
 22  *GIVES EXPONENT BASE 2 IN #DP1 MSP
 23 0023 8114  CALL     NOP     #LOGF  *FRACTIONAL LOG       IM   0   0  84
 24  *RESULT ALWAYS NEG SINCE LOG OF FRACTIO
 25 0024 D400  SIN/AC   NOP     32                           IM   5   0  32
 26  *ARITH R.S. OF FRACT *4
 27 0025 3412  TRANS    NOP     #DP1                             13   0  18
 28 0026 10A0  RSH      TRANS   NOP                              4   5   0
 29 0027 1180  RSH      RSH     NOP                              4  12   0
 30 0028 1180  RSH      RSH     NOP                              4  12   0
 31 0029 1180  RSH      RSH     NOP                              4  12   0
```

```
PAGE    3   HARMONIAC ASSEMBLY OF : FFT1LG
 32 002A 3580  TRANS   RSH     NOP                                13  12   0
 33 002B 4805  #DP1    NOP     TRANS                              18   0   5
 34 002C 9802  MADDR   NOP     #PLOGE2                       IM    6   0   2
 35 002D 3247  MPY     #DP1    MDAT  GET EXP BASE E               12  18   7
 36 002E 4484  #DP0    MPYH    MPYL                               17   4   4
 37 002F 3231  MPY     #DP0    #DP0  #LLEFT LSP                  12  17  17
 38 0030 4408  #DP0    NOP     Z1                                 17   0   8
 39 0031 A2E0  ADD     #ZRO    0                             IM    8  23   0
 40 0032 5460  #T1     ALU     NOP                                21   3   0
 41  * LINE UP BIN PT WITH EXPONENT
 42 0033 C01B  CALL    NOP     #DPRT1  #LINRT                IM    0   0  59
 43 0034 A2A1  ADD     #T1     1                             IM    8  21   1
 44 0035 5460  #T1     ALU     NOP                                21   3   0
 45 0036 BAA5  COMP    #T1     5                             IM   14  21   5
 46 0037 C413  JMP     NOP     #LINRT                        IM    1   0  51
 47 0038 3631  TRANS   #DP0    #DP0                               13  17  17
 48 0039 0419  JMP     NOP     #L/M                               1   0  25
 49 003A 20B2  ADD     TRANS   #DP1                               8   5  18
 50  *
 51  * SINGLE D.P. LOGICAL RIGHT OF #DP0
 52 003B D401  SIN/AC  NOP     33       #DPRT1               IM    5   0  33
 53 003C 1220  RSH     #DP0    NOP                                4  17   0
 54  *LOST BIT IN AFLG
 55 003D D402  SIN     NOP     34 SEE LRSB&ROTATE            IM    5   0  34
 56 003E 4580  #DP0    RSH     NOP                               17  12   0
 57 003F BC09  M1D/AD  NOP     #FFFEM1                       IM   15   0   9
 58 0040 24F1  AND     MDAT    #DP0                               9   7  17
 59 0041 2C23  OR      AFLG    ALU PUT IN CARRY BIT              11   1   3
 60 0042 1060  RSH     ALU     NOP  ROTATE RT                     4   3   0
 61 0043 A180  ADD     RSH     0                             IM    8  12   0 SOU
 67
 62 0044 0401  JMP     NOP     RETURN                             1   0   1
 63 0045 4403  #DP0    NOP     ALU                               17   0   3
 64  *
 65  * NORM BUILDS INTEGER EXPONENT
 66  * IN ALU,OUT EXP MSP #DP1,FRAC LSP
 67 0046 D400  SIN     NOP     32       #NORM                IM    5   0  32
 68 0047 9000  RSH/MC  NOP     0                             IM    4   0   0
 69 0048 B061  MPY     ALU     1                             IM   12   3   1
 70 0049 B402  TRANS   NOP     2                             IM   13   0   2
 71 004A 58A0  #T2     TRANS   NOP                                22   5   0
 72 004B FC17  M1D/AD  NOP     #H4000                        IM   15   0  55
 73 004C A2EF  ADD     #ZRO    15  EXP CNT                   IM    8  23  15
 74 004D 4864  #DP1    ALU     MPY      #NLOOP                   18   3   4
 75 004E 24E4  AND     MDAT    MPY                                9   7   4
 76 004F 32C4  MPY     #T2     MPY                               12  22   4
 77 0050 38E3  COMP    MDAT    ALU                               14   7   3
 78 0051 850D  JMP     NOP     #NLOOP UNTIL BIT 14 SET       IM    1   0  77
 79 0052 AA41  SUB     #DP1    1                             IM   10  18   1
 80 0053 0401  JMP     NOP     RETURN                             1   0   1
 81  *
 82  *
 83  * LOGF IS FRACTIONAL LOG BASE E
 84 0054 D400  AC      NOP     32       #LOGF                IM    5   0  32
 85 0055 6801  #RTN3   NOP     RETURN                            26   0   1
```

```
PAGE    4   HARMONIAC ASSEMBLY OF : FFT1LG
 86 0056 9803  MADDR   NOP     #HLF                          IM   6   0   3
 87 0057 36F2  TRANS   #ZRO    #DP1                              13  23  18
 88 0058 90A0  RSH     TRANS   0                             IM   4   5   0
 89 0059 2987  SUB     RSH     MDAT  USUALLY NEG RESUL            10  12   7
 90 005A 4465  #DPO    ALU     TRANS                             17   3   5
 91 005B 8126  CALL    NOP     #DIV  NEG NUMERATOR          IM   0   0 198
 92 005C 2187  ADD     RSH     MDAT                              8  12   7
 93 005D 3411  TRANS   NOP     #DPO  RESULT DPO LSP              13   0  17
 94 005E 30B1  MPY     TRANS   #DPO  SQUARE IT                 12   5  17
 95 005F 9008  RSH/MC  NOP     8  POP MQ                     IM   4   0   8
 96 0060 4484  #DPO    MPYH    MPYL  SAVE SQ                   17   4   4
  2 0061 9804  MADDR   =NOP    ,#C4                          IM   6   0   4
  3 0062 3227  MPY     #DPO    MDAT  (C4)                      12  17   7
  4 0063 2087  ADD     MPYH    MDAT   (C5)                     8   4   7
  5 0064 3223  MPY     #DPO    ALU                             12  17   3
  6 0065 2087  ADD     MPYH    MDAT  (C6)                      8   4   7
  7 0066 3223  MPY     #DPO    ALU                             12  17   3
  8 0067 2087  ADD     MPYH    MDAT  (C7)                      8   4   7
  9 0068 3223  MPY     #DPO    ALU                             12  17   3
 10 0069 34A0  TRANS   TRANS   NOP  RESULT OF DIV              13   5   0
 11 006A 3085  MPY     MPYH    TRANS                           12   4   5
 12 006B 2085  ADD     MPYH    TRANS                            8   4   5
 13 006C B062  MPY     ALU     2                             IM  12   3   2
 14 006D 4804  #DP1    NOP     MPYL                              18   0   4
 15 006E 041A  JMP     NOP     #RTN3                             1   0  26
 16 006F 9000  RSH/MC  NOP     0                             IM   4   0   0
 17   *
 18   * PUT COSINE WINDOW ON SIGNAL
 19   * HANNS MEM1 BUFFER, ZEROES MEMO
 20   * SIGNAL INPUT IN MEMO AFTER 12SEPT
 21   * CHECKED OK 31 AUG 1979
 22 0070 6001  #LI/MX  NOP     RETURN  #HANN                     24   0   1
 23   *GET PARAMETERS FROM MEM1
 24 0071 C040  CALL    NOP     #PARAS                        IM   0   0 288
 25 0072 5804  #T2     NOP     MPYL                              22   0   4
 26 0073 FC12  M1D/AD  NOP     #SGNMSK  GET :7FFF            IM  15   0  50
 27   *CALC PHASE INCR 2PI/N
 28 0074 B4E1  TRANS   MDAT    1                             IM  13   7   1
 29 0075 44A5  #DPO    TRANS   TRANS                             17   5   5
 30 0076 8126  CALL    NOP     #DIV                          IM   0   0 198
 31 0077 22F6  ADD     #ZRO    #T2                               8  23  22
 32   *RESULT IN #DPO LSP
 33   *START READING SINE FROM -PI/2
 34 0078 90F9  RSH     MDAT    25   PUSH MEMS                IM   4   7  25
 35 0079 B580  TRANS   RSH     0                             IM  13  12   0 SO
121
 36 007A 28A5  SUB     TRANS   TRANS  TO-:3FFF(-PI/2)             10   5   5
 37 007B 4C03  #TEMP   NOP     ALU                               19   0   3
 38 007C 5460  #T1     ALU     NOP                               21   3   0
 39 007D 1BBD  MADDR   #BASE   #BASE                              6  29  29
 40 007E 58A0  #T2     TRANS   NOP   CNTR                        22   5   0
 41   * 1ST 1/2 OF DATA
 42 007F D6A8  SIN     #T1     40                            IM   5  21  40
 43 0080 22ED  ADD     #ZRO    SIN     #HANL01  >                8  23  13
 44 0081 1060  RSH     ALU     NOP                               4   3   0
```

```
PAGE    5   HARMONIAC ASSEMBLY OF : FFT1L1
 45 0082 2993  SUB     RSH     #TEMP  ADD H.S.                    10  12  19
 46  * TAKE INPUT DATA FROM MEM 0 (IMG STOR)
 47 0083 3067  MPY     ALU     MDAT                                12   3   7
 48 0084 22B1  ADD     #T1     #DPO UPDATE SIN ADDR                 8  21  17
 49 0085 9C80  MDAT    MPYH    0                              IM    7   4   0
 50 0086 5460  #T1     ALU     NOP                                 21   3   0
 51 0087 A2C2  ADD     #T2     2  UPDATE CNTR                IM    8  22   2
 52 0088 5860  #T2     ALU     NOP                                 22   3   0
 53 0089 3876  COMP    ALU     #T2                                 14   3  22
 54 008A 8420  JMP     NOP     #HANL01                        IM    1   0 128
 55  * 2ND 1/2 OF DATA
 56 008B D6A8  SIN     #T1     40                             IM    5  21  40
 57 008C 58A0  #T2     TRANS   NOP  ZERO CNTR                      22   5   0
 58 008D 22ED  ADD     #ZRO    SIN     #HANL02                      8  23  13
 59 008E 1060  RSH     ALU     NOP                                  4   3   0
 60 008F 2993  SUB     RSH     #TEMP                               10  12  19
 61 0090 3067  MPY     ALU     MDAT *DATA FROM M0                  12   3   7
 62 0091 2AB1  SUB     #T1     #DPO                                10  21  17
 63 0092 9C80  MDAT    MPYH    0                              IM    7   4   0
 64 0093 5460  #T1     ALU     NOP                                 21   3   0
 65 0094 A2C2  ADD     #T2     2                              IM    8  22   2
 66 0095 5860  #T2     ALU     NOP                                 22   3   0
 67 0096 3876  COMP    ALU     #T2                                 14   3  22
 68 0097 842D  JMP     NOP     #HANL02                        IM    1   0 141
 69 0098 D6A8  SIN     #T1     40                             IM    5  21  40
 70 0099 0418  JMP     NOP     #LI/MX                              1   0  24
 71 009A 9000  RSH     NOP     0                              IM    4   0   0
 72  *
 73  * LOG POWER CALCS FOR SPECTRUM
 74  * DOES 1/2[LOG(X^2+Y^2)]
 75  *
 76 009B 9809  MADDR   NOP     #PWRRET #PWRLOG                IM    6   0   9
 77 009C 1C01  MDAT    NOP     RETURN                              7   0   1
 78 009D D418  SIN     NOP     56  I6,>=                      IM    5   0  56
 79 009E A2E0  ADD     #ZRO    0                              IM    8  23   0
 80 009F 6C60  #L2/N1  ALU     NOP  LOOP CNTR                      27   3   0
 81 00A0 1BBD  MADDR   #BASE   #BASE                               6  29  29
 82 00A1 9000  RSH     NOP     0 NORM MEMS                    IM    4   0   0
 83 00A2 34E0  TRANS   MDAT    NOP                                 13   7   0
 84 00A3 30E5  MPY     MDAT    TRANS   #PWRLOP                     12   7   5
 85 00A4 3447  TRANS   MOADR   MDAT                                13   2   7
 86 00A5 5805  #T2     NOP     TRANS                               22   0   5
  2 00A6 4484  #DPO    MPY     MPY                                 17   4   4
  3  * NOW Y^2
  4 00A7 30A7  MPY     TRANS   MDAT                                12   5   7
  5 00A8 8164  CALL    NOP     #DPADD                         IM    0   0 452
  6 00A9 4884  #DP1    MPY     MPY                                 18   4   4
  7 00AA 8126  CALL    NOP     #DIV                           IM    0   0 198
  8 00AB E2FF  ADD     #ZRO    63   SCALE TO INTEGER          IM    8  23  63
  9  * COPY LINEAR PWR SPECT TO EXTRA BUFFER
 10  * AT #BASE+N IN MEM 0 SO IT'S AVAILABLE
 11  * FOR RESYNTHESIS
 12 00AC 20DB  ADD     M1ADDR  #L2/N1  ADV ADDR W M1               8   6  27
 13 00AD 1803  MADDR   NOP     ALU                                 6   0   3
```

```
PAGE    6   HARMONIAC ASSEMBLY OF : FFT1L2
 14 00AE 1C11  MDAT     NOP      #DPO                              7   0  17
 15 00AF 22F1  ADD      #ZRO     #DPO                              8  23  17
 16 00B0 8015  CALL     NOP      #ILOG10                    IM     0   0  21
 17 00B1 BC07  M1D/AD             #DISCL                     IM    15   0   7
 18  * SCALE DISPLAY BY MPY & DISCARD OF LSP
 19 00B2 30E3  MPY      MDAT     ALU                               12   7   3
 20 00B3 3C16  M1D/AD   NOP      #T2     GET CURR ADDR             15   0  22
 21 00B4 1816  MADDR              #T2                               6   0  22
 22 00B5 A362  ADD      #L2/N1   2                          IM     8  27   2 SOU
181
 23 00B6 6C60  #L2/N1   ALU      NOP  DOES N/2 PTS                 27   3   0
 24 00B7 A0C1  ADD      M1ADR    1  UPDATE                  IM     8   6   1
 25 00B8 9C80  MDAT     MPY      0        SPECT RESULT      IM     7   4   0
 26 00B9 D408  SIN      NOP      40  > ,I6                  IM     5   0  40
 27 00BA 1863  MADDR    ALU      ALU  FOR NEXT FETCH               6   3   3
 28 00BB 3B7B  COMP     #L2/N1   #L2/N1                            14  27  27
 29 00BC C423  JMP      NOP      #PWRLOP                    IM     1   0 163
 30 00BD B4E1  TRANS    MDAT     1                          IM    13   7   1
 31  * MUST ZERO OUT REST OF R&I FOR SMOOTH
 32 00BE 23BB  ADD      #BASE    #L2/N1  FORM TARGET ADD           8  29  27
 33 00BF 5823  #T2      AFLG     ALU                               22   1   3
 34 00C0 9019  MC                25 PUSH BOTH MEMS          IM     4   0  25
 35 00C1 38D6  COMP     M1ADDR   #T2      #PZOT                   14   6  22
 36 00C2 8521  JMP               #PZOT                      IM     1   0 193
 37 00C3 9EE0  MDAT     #ZRO     0                          IM     7  23   0
 38 00C4 C579  JMP      NOP      #MRET                      IM     1   0 505
 39 00C5 9809  MADDR    NOP      #PWRRET                    IM     6   0   9
 40  *
  2 00C6 7001  #RTN1    NOP      RETURN  #DIV  #DIV SUBR           28   0   1
  3  #HI &LO DIVIDEND INPUT #DPO  (D.P.)
  4  #DIVISOR IN ALU, RESULT IN #DPO  LSP
  5  * REMAINDER IN #DPO MSP(SGN OF DVDEND)
  6  * CHECKED OK AUG 1979
  7 00C7 4063  #ALUS    ALU      ALU                               16   3   3
  8 00C8 3223  MPY      =#DPO    ,ALU                 #QUOT        12  17   3
  9  #SIGN QOT BY MPY
 10 00C9 9400  SIN      NOP      0                          IM     5   0   0
 11 00CA D500  SIN      NOP      96                         IM     5   0  96
 12  #IF MSP OF DPO IS 0 ,USE DIVISR AS SGN
 13 00CB BA20  COMP     #DPO     0                          IM    14  17   0
 14 00CC B201  MPY      #ALUS    1  DONE IF =0              IM    12  16   1
 15 00CD D410  SIN/AC   NOP      48          #IM6,<         IM     5   0  48
 16 00CE B620  TRANS    #DPO     0                          IM    13  17   0
 17 00CF 4C85  #TEMP    MPY      TRANS                             19   4   5
 18  #MAKE DIVISOR POS
 19 00D0 BA00  COMP     #ALUS    0                          IM    14  16   0
 20 00D1 8534  JMP      NOP      #POSDVR                    IM     1   0 212
 21 00D2 28B0  SUB      TRANS    #ALUS                             10   5  16
 22 00D3 4003  #ALUS    NOP      ALU                               16   0   3
 23  #MAKE DVDEND POS
 24 00D4 BA20  COMP     #DPO     0        #POSDVR <         IM    14  17   0
 25 00D5 8538  JMP      NOP      #POSDND                    IM     1   0 216
 26 00D6 D400  SIN/AC   NOP      32                         IM     5   0  32
 27 00D7 C138  CALL     NOP      #DPNEG                     IM     0   0 248
 28  #PACK LSP LEFT BEFORE USING D.P. LEFT
```

```
PAGE    7   HARMONIAC ASSEMBLY OF : FFT1DV
 29 00D8 080B   IOADR    NOP      SOV      #POSDND                    2   0  11
 30 00D9 30B1   MPY      TRANS    #DPO * PACK LSP                    12   5  17
 31 00DA 4408   #DPO     NOP      Z1                                 17   0   8
 32 00DB 50A0   #CNT     TRANS    NOP                                20   5   0
 33 00DC 2A30   SUB      #DPO     #ALUS    #DIVL SUB DIVOR           10  17  16
 34  *RESTORE DIVIDEND IF OV=1
 35 00DD B820   COMP     AFLG     0                             IM   14   1   0
 36 00DE C521   JMP      NOP      #REST                         IM    1   0 225
 37 00DF AE20   OR       #DPO     0                             IM   11  17   0
 38 00E0 2A30   SUB      #DPO     #ALUS   *DO REAL SUBTR             10  17  16
 39 00E1 3071   MPY      ALU      #DPO     #REST                     12   3  17
 40 00E2 4508   #DPO     Z1       Z1                                 17   8   8
 41 00E3 A281   ADD      #CNT     1            *INCR CNT        IM    8  20   1
 42 00E4 5060   #CNT     ALU      NOP                                20   3   0
 43 00E5 B870   COMP     ALU      16                            IM   14   3  16
 44 00E6 853C   JMP      NOP      #DIVL                         IM    1   0 220
 45 00E7 D400   SIN/AC   NOP      32                            IM    5   0  32
 46 00E8 D413   SIN/AC   NOP      51                            IM    5   0  51
 47  * MOVE OV(0) INTO MSP OF REM
 48 00E9 1220   RSH/MC   #DPO     NOP                                4  17   0
 49  *SET REM SIGN SAME AS DVDEND
 50 00EA 3413   TRANS    NOP      #TEMP   GET DVDND                  13   0  19
 51 00EB B8A0   COMP     TRANS    0  <                          IM   14   5   0
 52 00EC C531   JMP      NOP      #POSR                         IM    1   0 241
 53 00ED 4580   #DPO     RSH      NOP        *REM POS                17  12   0
 54 00EE B620   TRANS    #DPO     0                             IM   13  17   0
 55 00EF 28A5   SUB      TRANS    TRANS                              10   5   5
 56 00F0 4460   #DPO     ALU      NOP                                17   3   0
 57 00F1 BA60   COMP     #TEMP    0        #POSR                IM   14  19   0
 58 00F2 C536   JMP      NOP      #POSQ                         IM    1   0 246
 59  *NEGATE QUOT IF REQ'D
 60 00F3 B400   TRANS    NOP      0                             IM   13   0   0
 61 00F4 28B1   SUB      TRANS    #DPO                               10   5  17
 62 00F5 4403   #DPO     NOP      ALU                                17   0   3
 63 00F6 041C   JMP      NOP      #RTN1    #POSQ                      1   0  28
 64 00F7 D404   SIN/AC   NOP      36                            IM    5   0  36
 65  *  DOUBLE PRECISION NEGATE OF #DPO
 66 00F8 9000   RSH/MC   NOP      0        #DPNEG               IM    4   0   0
 67 00F9 D400   SIN/AC   NOP      32                            IM    5   0  32
 68 00FA FC12   M1D/AD   NOP      #SGNMSK     *GET SGNMSK       IM   15   0  50
 69 00FB B620   TRANS    #DPO     0                             IM   13  17   0
 70 00FC 28B1   SUB      TRANS    #DPO        *NEGATE LSP            10   5  17
 71 00FD 24E3   AND      MDATA    ALU         *CLR SGN               9   7   3
 72 00FE 4403   #DPO     NOP      ALU         *SAV LSP               17   0   3
 73 00FF 38B1   COMP     TRANS    #DPO                               14   5  17
 74  *IF LSP=/0 MAKE MSP COMPL, NOT NEGATE
 75 0100 8443   JMP      NOP      #NOTO                         IM    1   0 259
 76 0101 28A5   SUB      TRANS    TRANS       *NEG MSP               10   5   5
 77 0102 0401   JMP      NOP      RETURN                             1   0   1
 78 0103 4460   #DPO     ALU      NOP      #NOTO *SAV MSP            17   3   0
 79 0104 AA21   SUB      #DPO     1              *COMPLEM       IM   10  17   1
 80 0105 0401   JMP      NOP      RETURN                             1   0   1
 81 0106 4460   #DPO     ALU      NOP                                17   3   0
  2  *  EXPONENTIAL BASE TWO
```

```
PAGE    8   HARMONIAC ASSEMBLY OF : FFT1EX
  3  * LEVL ZERO SUBROUTINE(IN ALU,OUT MPYL)
  4 0107 4063  #ALUS   =ALU     ,ALU      #EXP2                      16   3   3
  5 0108 B401  TRANS   =NOP     ,1                            IM   13   0   1
  6 0109 B0A1  MPY     =TRANS   ,1                            IM   12   5   1
  7 010A B860  COMP    =ALU     ,0       GEN CASE              IM   14   3   0
  8 010B 844E  JMP     =NOP     ,#MPYMO GEN CASE               IM    1   0 270
  9 010C B402  TRANS   =NOP     ,2                            IM   13   0   2
 10 010D 0401  JMP     =NOP     ,RETURN GEN CASE (REQD)             1   0   1
 11 010E AA01  SUB      #ALUS   1        #MPYMO                IM   10  16   1
 12 010F 30A4  MPY     =TRANS   ,MPY                                12   5   4
 13 0110 4060  #ALUS   =ALU     ,NOP                                 16   3   0
 14 0111 B860  COMP    =ALU     ,0                            IM   14   3   0
 15 0112 844E  JMP     =NOP     ,#MPYMO                       IM    1   0 270
 16 0113 22E4  ADD      #ZRO    MPYL                                 8  23   4
 17 0114 0401  JMP     =NOP     ,RETURN                              1   0   1
 18  * SCL2 - DIVIDES BY A PWR OF 2
 19  * INPUT IN RSH (ALREADY/2 ONCE)
 20  * WITH CNT OF TOTAL SHIFTS REQ'D IN #T2
 21 0115 D400  SIN/AC  NOP      32       #SCL2    I6&RS        IM    5   0  32
 22 0116 AAC1  SUB     #T2      1  *CNT IN T2 MSP               IM   10  22   1
 23 0117 4063  #ALUS   ALU      ALU                                  16   3   3
 24 0118 AA01  SUB     #ALUS    1        #SHLOP                IM   10  16   1
 25 0119 4063  #ALUS   ALU      ALU                                  16   3   3
 26 011A B860  COMP    ALU      0                             IM   14   3   0
 27 011B 8458  JMP     NOP      #SHLOP                        IM    1   0 280
 28 011C 1180  RSH     RSH      NOP                                  4  12   0
 29 011D A180  ADD     RSH      0  FOR EASEY ACCESS           IM    8  12   0 SO
285
 30 011E 0401  JMP     NOP      RETURN                               1   0   1
 31 011F 9418  SIN/AC  NOP      24                            IM    5   0  24
  2  * FFT BEGINS
  3  * ARGUMENTS FROM MEM1 (0-> )
  4  * ARE: BASER,BASEI,M,L(^2)
  5  * BASES MUST BE SAME IN EA MEM(H400)
  6  * M IS TOT PTS (PWR OF 2),L NON 0 PTS
  7 0120 5801  #T2      NOP      RETURN  #PARAS                      22   0   1
  8 0121 D400  SIN/AC   NOP      32        *IM6,=              IM    5   0  32
  9 0122 9008  RSH/MC   NOP      8         POPM1               IM    4   0   8
 10 0123 BC00  M1D/AD   NOP      #ARGS                         IM   15   0   0
 11 0124 74E0  #BASE   =MDAT    ,NOP                                29   7   0
 12 0125 A0E0  ADD      MDAT     0                             IM    8   7   0
 13 0126 7463  #BASE    ALU      ALU ***AVOID M1 ERR                 29   3   3
 14 0127 34E0  TRANS   =MDAT    ,NOP                                13   7   0
 15 0128 6405  #L/M    =NOP     ,TRANS                              25   0   5
 16 0129 64E0  #L/M    =MDAT    ,NOP                                25   7   0
 17  * CALC NON ZERO PTS (PWR OF 2)
 18 012A 2337  ADD      #L/M     #ZRO                                 8  25  23
 19 012B 8047  CALL     NOP      #EXP2                         IM    0   0 263
 20 012C 3404  TRANS    NOP      MPYL                                13   0   4
 21 012D 6CA0  #L2/N1   TRANS    NOP                                 27   5   0
 22  * CALC TOT PTS (PWR OF 2)
 23 012E 22F9  ADD      #ZRO     #L/M                                 8  23  25
 24 012F 8047  CALL     NOP      #EXP2                         IM    0   0 263
 25 0130 6C04  #L2/N1   NOP      MPYL                                27   0   4
 26 0131 0416  JMP      NOP      #T2                                  1   0  22
```

```
PAGE    9   HARMONIAC ASSEMBLY OF : FFT1A
 27 0132 9000  RSH      NOP      0                                IM   4   0   0
 28  *
 29 0133 6801  #RTN3    NOP      RETURN  #FFT                          26   0   1
 30 0134 C040  CALL     NOP      #PARAS                           IM   0   0 288
 31  * SET UP OUTER LOOP
 32 0135 A2E1  ADD      #ZRO     1                                IM   8  23   1
 33 0136 7803  #L1/0    NOP      ALU                                  30   0   3
 34 0137 3419  TRANS    =NOP     ,#L/M     #LOLOO                     13   0  25
 35 0138 28BE  SUB      =TRANS   ,#L1/0                               10   5  30
 36 0139 8047  CALL     =NOP     ,#EXP2                           IM   0   0 263
 37 013A 6004  #LI/MX   =NOP     ,MPYL                                24   0   4
 38 013B 5804  #T2      =NOP     ,MPYL                                22   0   4
 39 013C B402  TRANS    NOP      2                                IM  13   0   2
 40 013D 30A4  MPY      =TRANS   ,MPYL                                12   5   4
 41 013E FC12  M1D/AD   NOP      #SGNMSK                          IM  15   0  50
 42 013F 3404  TRANS    =NOP     ,MPYL                                13   0   4
 43 0140 60A0  #LI/MX   =TRANS   ,NOP                                 24   5   0
 44  * SET UP 2PI IN D. P. : 1 7FFF
 45 0141 2317  ADD      =#LI/MX  ,#ZRO                                 8  24  23
 46 0142 B4E1  TRANS    MDAT     1                                IM  13   7   1
 47 0143 44A5  #DPO     TRANS    TRANS                                17   5   5
 48 0144 8126  CALL     NOP      #DIV                             IM   0   0 198
 49 0145 D410  SIN/AC   NOP      48  * <                          IM   5   0  48
 50 0146 233E  ADD      =#L/M    ,#L1/0                               8  25  30
 51 0147 3879  COMP     ALU      #L/M                                 14   3  25
 52  * DO PRUNE IF LO+L<M
 53 0148 854B  JMP      =NOP     ,#I3                             IM   1   0 331
 54 0149 2377  ADD      =#L2/N1  ,#ZRO                                8  27  23
 55 014A 5803  #T2      =NOP     ,ALU                                 22   0   3
 56 014B B6E0  TRANS    =#ZRO    ,0        #I3                    IM  13  23   0
 57 014C 50A5  #CNT     =TRANS   ,TRANS                               20   5   5
 58 014D 22F1  ADD      #ZRO     #DPO                                 8  23  17
 59 014E 7C03  #RTN2    NOP      ALU  TEMP SAVE                       31   0   3
 60 014F 9000  RSH/MC   =NOP     ,0                               IM   4   0   0
 61 0150 329F  MPY      #CNT     #RTN2   #LMLOO                       12  20  31
 62 0151 BC0B  M1D/AD   NOP      #PI/2                            IM  15   0  11
 63 0152 3404  TRANS    =NOP     ,MPY                                 13   0   4
 64 0153 D4A0  SIN      =TRANS   ,32                              IM   5   5  32
 65 0154 20E4  ADD      MDAT     MPY  *ADDS PI/2 FOR COS              8   7   4
 66 0155 4C0D  #TEMP    NOP      SIN                                  19   0  13
 67 0156 D468  SIN      =ALU     ,40                              IM   5   3  40
 68 0157 2317  ADD      =#LI/MX  ,#ZRO  FOR INNER LOOP                8  24  23
 69 0158 340D  TRANS    =NOP     ,SIN                                 13   0  13
 70 0159 4CA0  #TEMP    =TRANS   ,NOP                                 19   5   0
 71  * SET UP INNER LOOP
 72 015A 7860  #L1/0    =ALU     ,NOP                                 30   3   0
 73 015B 23D4  ADD      =#L1/0   ,#CNT                                8  30  20
 74 015C 3700  TRANS    =#LI/MX  ,NOP      #L1LOO                     13  24   0
 75 015D 2865  SUB      =ALU     ,TRANS                               10   3   5
 76 015E 4063  #ALUS    =ALU     ,ALU                                 16   3   3
 77 015F 23A3  ADD      #BASE    ALU                                   8  29   3
 78 0160 4463  #DPO     ALU      ALU                                  17   3   3
 79 0161 2078  ADD      ALU      #LI/MX                                8   3  24
 80 0162 4863  #DP1     ALU      ALU                                  18   3   3
```

```
PAGE   10   HARMONIAC ASSEMBLY OF : FFT1A
 81 0163 1A51  MADDR  =#DP1    ,#DPO                                  6  18  17
 82 0164 34E7  TRANS  =MDAT    ,MDAT                                 13   7   7
 83 0165 1A32  MADDR  =#DPO    ,#DP1                                  6  17  18
 84 0166 28E5  SUB    =MDAT    ,TRANS                                10   7   5
 85 0167 5463  #T1    =ALU     ,ALU                                  21   3   3
 86 0168 20E5  ADD    =MDAT    ,TRANS                                 8   7   5
 87 0169 1C60  MDAT   =ALU     ,NOP                                   7   3   0
 88 016A 28A7  SUB    =TRANS   ,MDAT                                10   5   7
 89 016B 4063  #ALUS  =ALU     ,ALU                                  16   3   3
 90 016C 20A7  ADD    =TRANS   ,MDAT                                 8   5   7
  2 016D 1811  MADDR   NOP      #DPO                                  6   0  17
  3 016E 3275  MPY     #TEMP    #T1                                  12  19  21
  4 016F 1C03  MDAT    NOP      ALU                                   7   0   3
  5 0170 A080  ADD     MPYH     0                              IM     8   4   0
  6 0171 3213  MPY     #ALUS    #TEMP                                 12  16  19
  7 0172 1A52  MADDR   #DP1     #DP1 ACTS AS MPY DELAY                 6  18  18
  8 0173 2083  ADD     MPYH     ALU                                   8   4   3
  9 0174 32B3  MPY     #T1      #TEMP                                 12  21  19
 10 0175 1C60  MDAT    ALU      NOP                                   7   3   0
 11 0176 A080  ADD     MPYH     0                              IM     8   4   0
 12 0177 3270  MPY     #TEMP    #ALUS                                 12  19  16
 13 0178 3700  TRANS   =#LI/MX ,NOP  ACTS AS MPY DELAY               13  24   0
 14 0179 2883  SUB     MPYH     ALU                                  10   4   3
 15 017A 1C03  MDAT    =NOP     ,ALU                                  7   0   3
 16  * END INNER LOOP CALC'S
 17 017B 23C5  ADD     =#L1/O   ,TRANS                                8  30   5
 18 017C 7860  #L1/O   =ALU     ,NOP                                  30   3   0
 19  *CHECK INNER LOOP STAGE
 20 017D 387B  COMP    =ALU     ,#L2/N1                               14   3  27
 21 017E 855C  JMP     =NOP     ,#L1LOO                        IM     1   0 348
 22 017F 23D4  ADD      #L1/O    #CNT                                 8  30  20
 23 0180 A281  ADD     =#CNT    ,1                             IM     8  20   1
 24 0181 5063  #CNT    =ALU     ,ALU                                  20   3   3
 25  *CHECK MIDDLE LOOP STAGE
 26 0182 D400  SIN/AC   NOP      32     *=&IM6                  IM     5   0  32
 27 0183 3876  COMP    =ALU     ,#T2                                  14   3  22
 28 0184 8550  JMP     =NOP     ,#LMLOO                        IM     1   0 336
 29 0185 341E  TRANS   =NOP     ,#L1/O                                13   0  30
 30 0186 A0A1  ADD     =TRANS   ,1                             IM     8   5   1
 31 0187 7803  #L1/O   =NOP     ,ALU                                  30   0   3
 32 0188 A861  SUB     =ALU     ,1                             IM    10   3   1
 33 0189 3879  COMP    =ALU     ,#L/M                                 14   3  25
 34 018A C457  JMP     =NOP     ,#LOLOO                        IM     1   0 311
 35  * NOW FFT FINISHED - DO REORDERING
 36  * DOES SOFTWARE REVERSE BIT REORDER
 37  * & SCALE DOWN BY SQRT NON-0 NO. PTS.
 38  * (THIS IS APPROX GROWTH RATE OF DATA)
 39 018B 9320  RSH     #L/M     0                              IM     4  25   0 SO
395
 40 018C 2197  ADD     RSH      #ZRO                                  8  12  23
 41 018D 1BBD  MADDR   #BASE    #BASE                                 6  29  29
 42 018E 5863  #T2     ALU      ALU                                  22   3   3
 43 018F 9000  MC               0        #RLOOP                IM     4   0   0
 44 0190 A0C0  ADD     M1ADR    0                              IM     8   6   0
 45 0191 4463  #DPO    ALU      ALU              SAV NORM             17   3   3
```

```
PAGE   11   HARMONIAC ASSEMBLY OF : FFT1B
 46 0192 C073  CALL     NOP      #REVBIT                    IM   0   0 435
 47 0193 9418  AC                24  >=                     IM   5   0  24
 48 0194 3A71  COMP     #TEMP    #DP0  N>R->SKIP                14  19  17
 49 0195 C468  JMP               #SKIP                      IM   1   0 424
 50   * DO THE EXCH OF REV/NORM
 51 0196 90E0  RSH      MDAT     0                          IM   4   7   0
 52 0197 8055  CALL              #SCL2                      IM   0   0 277
 53 0198 4980  #DP1     RSH      NOP  SCLD NORM 1               18  12   0
 54 0199 3407  TRANS             MDAT                           13   0   7
 55 019A 8055  CALL              #SCL2                      IM   0   0 277
 56 019B 10A0  RSH      TRANS    NOP                            4   5   0
 57 019C 4803  #DP1     NOP      ALU SCLD NORM 0                18   0   3
 58 019D 1A73  MADDR    #TEMP    #TEMP SET REV                  6  19  19
 59 019E 8055  CALL              #SCL2                      IM   0   0 277
 60 019F 90E0  RSH      MDAT     0                          IM   4   7   0
 61 01A0 7060  #RTN1    ALU      NOP  SAVE SCLD REV 1           28   3   0
 62 01A1 3407  TRANS             MDAT                           13   0   7
 63 01A2 8055  CALL              #SCL2                      IM   0   0 277
 64 01A3 10A0  RSH      TRANS    NOP                            4   5   0
 65 01A4 7003  #RTN1    NOP      ALU SAV SCLD REV 0             28   0   3
 66 01A5 1E52  MDAT     #DP1     #DP1 PUT NORMS AT REV          7  18  18
 67 01A6 1A31  MADDR    #DP0     #DP0  SET NORM ADDR            6  17  17
 68 01A7 1F9C  MDAT     #RTN1    #RTN1 REVS TO NORMS            7  28  28
 69 01A8 28DD  SUB      M1ADR    #BASE   #SKIP                 10   6  29
 70 01A9 A061  ADD      ALU      1                          IM   8   3   1
 71 01AA 9008  MC                8  POP                     IM   4   0   8
 72 01AB 387B  COMP     ALU      #L2/N1  END?                  14   3  27
 73 01AC 846F  JMP               #RLOOP                     IM   1   0 399
 74 01AD 24E7  AND      MDAT     MDAT                           9   7   7
 75 01AE B401  TRANS    NOP      1                          IM  13   0   1
 76 01AF BCAA  M1D/AD   TRANS    #RDYFL2                    IM  15   5  10
 77 01B0 9400  SIN/AC   NOP      0                          IM   5   0   0
 78 01B1 D500  SIN/AC   NOP      96  /=,I6                  IM   5   0  96
 79   * STATUS LEFT READY TO CONT TREE SEARCH
 80   * WITH NOT EQU TEST
 81 01B2 041A  JMP     =NOP      ,#RTN3                         1   0  26
 82   *
 83 01B3 32F7  MPY      #ZRO     #ZRO    #REVBIT                12  23  23
 84 01B4 28DD  SUB      M1ADR    #BASE                          10   6  29
 85 01B5 4C80  #TEMP    MPY      NOP                            19   4   0
 86 01B6 D402  AC                34  LRSB TO AFLG           IM   5   0  34
 87 01B7 1060  RSH      ALU      NOP                            4   3   0
 88 01B8 2C24  OR       AFLG     MPYL    #SWLOP                 11   1   4
 89 01B9 B062  MPY      ALU      2                          IM  12   3   2
 90 01BA A261  ADD      #TEMP    1                          IM   8  19   1
 91 01BB 4C60  #TEMP    ALU      NOP                            19   3   0
 92 01BC 3879  COMP     ALU      #L/M                           14   3  25
 93 01BD C478  JMP      NOP      #SWLOP                     IM   1   0 440
 94 01BE 1180  RSH      RSH      NOP                            4  12   0
 95 01BF 22E4  ADD      #ZRO     MPYL                           8  23   4
 96 01C0 9060  RSH      ALU      0                          IM   4   3   0
 97 01C1 219D  ADD      RSH      #BASE                          8  12  29
 98 01C2 0401  JMP               RETURN                         1   0   1
 99 01C3 4C63  #TEMP    ALU      ALU  REVERSED ADDR             19   3   3
```

```
PAGE   12   HARMONIAC ASSEMBLY OF : FFT1DA
  2  * D.P. ADD OF DPO &DP1 TO DPO, OV ERR
  3 01C4 9000  RSH/MC =NOP     ,0          #DPADD              IM   4   0   0
  4 01C5 FC12  M1D/AD =NOP     ,#SGNMSK                        IM  15   0  50
  5 01C6 24F1  AND    =MDAT    ,#DPO                            9   7  17
  6 01C7 D400  SIN/AC =NOP     ,32                             IM   5   0  32
  7 01C8 2072  ADD    =ALU     ,#DP1       ADDLO ORD            8   3  18
  8 01C9 24E3  AND    =MDAT1   ,ALU        CLEAR SIGN           9   7   3
  9 01CA 4403  #DPO   =NOP     ,ALU        LOSUM               17   0   3
 10 01CB 3620  TRANS  =#DPO    ,NOP                            13  17   0
 11 01CC 2025  ADD    =AFLG    ,TRANS      ADD CARRY            8   1   5
 12 01CD D400  SIN/AC =NOP     ,32                             IM   5   0  32
 13 01CE 2243  ADD    =#DP1    ,ALU        HISUM                8  18   3
 14 01CF 0401  JMP    =NOP     ,RETURN     RESLT DPO            1   0   1
 15 01D0 4460  #DPO   =ALU     ,NOP                            17   3   0
  2  *
  3  * HI LEV SPECTRUM CALLERS
  4  *
  5 01D1 980A  MADDR  =NOP     ,#PSPR      #PSPECT             IM   6   0  10
  6 01D2 1C01  MDAT   =NOP     ,RETURN     SAVERETURN           7   0   1
  7 01D3 C110  CALL   =NOP     ,#HANN                          IM   0   0 112
  8 01D4 0000  NOP    =        ,                                0   0   0
  9 01D5 C053  CALL   =NOP     ,#FFT                           IM   0   0 307
 10 01D6 0000  NOP    =        ,                                0   0   0
 11 01D7 803B  CALL   =NOP     ,#PWRLOG                        IM   0   0 155
 12 01D8 9000  RSH     NOP      0                              IM   4   0   0
 13 01D9 C579  JMP     NOP      #MRET                          IM   1   0 505
 14 01DA 980A  MADDR  =NOP     ,#PSPR                          IM   6   0  10
 15  *       =        ,
 16  * SMOOTHED PWR SPECT
 17 01DB 980B  MADDR    NOP      #PSSPR   #SSPECT              IM   6   0  11
 18 01DC 1C01  MDAT     NOP      RETURN                         7   0   1
 19 01DD 8171  CALL     NOP      #PSPECT                       IM   0   0 465
 20 01DE 0000  NOP                                             0   0   0
 21  * NOW DO CEPSTRUM XFORM
 22 01DF C053  CALL     NOP      #FFT                          IM   0   0 307
 23  * DO SEARCH FOR HIGHEST PEAK(PITCH)
 24  * THEN TRUNCATE CEPSTRUM BY ZEROING
 25  * HIGH TIME ELEMENTS
 26  * AND PRODUCE SMOOTHED SPECTRUM BY
 27  * A TRANSFORM
 28 01E0 23BB  ADD      #BASE    #L2/N1                         8  29  27
 29 01E1 5863  #T2      ALU      ALU                           22   3   3
 30 01E2 BC06  M1D/AD             #SMOOTH                     IM  15   0   6
 31 01E3 BCE3  M1D/AD   MDAT     #M  SET FOR SMOOTH          IM  15   7   3
 32 01E4 9019  RSH      NOP      25  PUSH BOTH               IM   4   0  25
 33 01E5 D408  SIN      NOP      40                          IM   5   0  40
 34 01E6 A2EA  ADD      #ZRO     10  * VOICING THRESHOLD     IM   8  23  10
 35 01E7 5463  #T1      ALU      ALU                           21   3   3
 36 01E8 A3BF  ADD      #BASE    31 ****                     IM   8  29  31 SC
488
 37 01E9 187D  MADDR    ALU      #BASE                          6   3  29
 38  * SCAN ALL REALS ABOVE FORMANT INFO
 39  * FOR HIGHEST PEAK -> PITCH PERIOD
 40 01EA 3440  TRANS    MOAD     NOP      #ZOT                 13   2   0
 41 01EB A0E0  ADD      MDAT     0                           IM   8   7   0
```

```
PAGE   13   HARMONIAC ASSEMBLY OF : FFT1CL
 42 01EC 3AA3  COMP     #T1      ALU  >PREVIOUS PKS?          14  21   3
 43 01ED 54E5  #T1      MDAT     TRANS  SKIP IF T1 IS >       21   7   5
 44  * PITCH PERIOD IN LSP OF T1 AT END
 45 01EE 3856  COMP     MOADR    #T2  ALL DONE?              14   2  22
 46 01EF C56A  JMP      NOP      #ZOT                    IM   1   0 490
 47 01F0 9EE0  MDAT     #ZRO     0                       IM   7  23   0
 48  * DONE-PUT PITCH PERIOD IN MEM
 49 01F1 2AFD  SUB      #ZRO     #BASE  REMOVE OFFSET        10  23  29
 50 01F2 2075  ADD      ALU      #T1                          8   3  21
 51 01F3 A061  ADD      ALU      1  ADDR LAGS BY 1       IM   8   3   1
 52 01F4 BC64  M1D/AD   ALU      #PITCHP                 IM  15   3   4
 53 01F5 C053  CALL     NOP      #FFT  DO SMOOTHER       IM   0   0 307
 54  * RESTORE M TO SAME AS L
 55 01F6 BC02  M1D/AD            #L                      IM  15   0   2
 56 01F7 BCE3  M1D/AD   MDAT     #M                      IM  15   7   3
 57 01F8 980B  MADDR    NOP      #PSSPR                  IM   6   0  11
 58 01F9 B401  TRANS    NOP      1        #MRET          IM  13   0   1
 59 01FA BCAA  M1D/AD   TRANS    #RDYFL2                 IM  15   5  10
 60 01FB 9400  SIN      NOP      0                       IM   5   0   0
 61 01FC 0407  JMP      NOP      MDAT                        1   0   7
 62 01FD D500  SIN      NOP      96                      IM   5   0  96
 6 WARNINGS TOTAL
 0  ERRORS TOTAL
```

APPENDIX III

A System for the Analysis and Resynthesis of the Soprano Singing Voice.

INTRODUCTION

The electronic enhancement of vocal recordings made by the "acoustic" process has to date most usually involved some form of filtering, or, as in the case of the Soundstream Process adopted by R.C.A., compensation for postulated recording horn resonances. The inadequate musical accompaniment has typically been left intact and surface noise, turntable rumble etc., have at best been reduced but not eliminated.

This paper describes a more thorough-going noise-reduction algorithm which, under certain conditions, can take the voice alone from such early recordings and make it available to re-recording under modern conditions. It is basically an analysis/synthesis system in which FFT analysis produces consecutive power spectra from which the fundamental frequency, F_o, of the voice is continually estimated together with the changing amplitudes of its harmonics. From these two parameters the voice, and only the voice, is resynthesised. The block diagram of the system is shown in Figure 1.

Implementation has been in software on a dual processor system made up of an HP21MX computer (16-bit) and Harmoniac, a 16-bit high-speed computer designed for audio research at the Macquarie University Speech and Language Research Centre (1). Considerations of speed and accuracy have led to some complexity in the flow chart shown in Figure 1. The processes marked B, C and G are performed in Harmonia at the same time that the remainder of the algorithm is executed in the HP because the FFT and resynthesis are time-consuming on a conventional computer. There would be some advantage in performing the F_o estimation in the faster computer but limitations on the available program memory have prevented this in the initial implementation.

SPECTRUM COLLAPSE F_o ESTIMATION

Time-domain (cepstral) techniques of F_o estimation become less accurate as

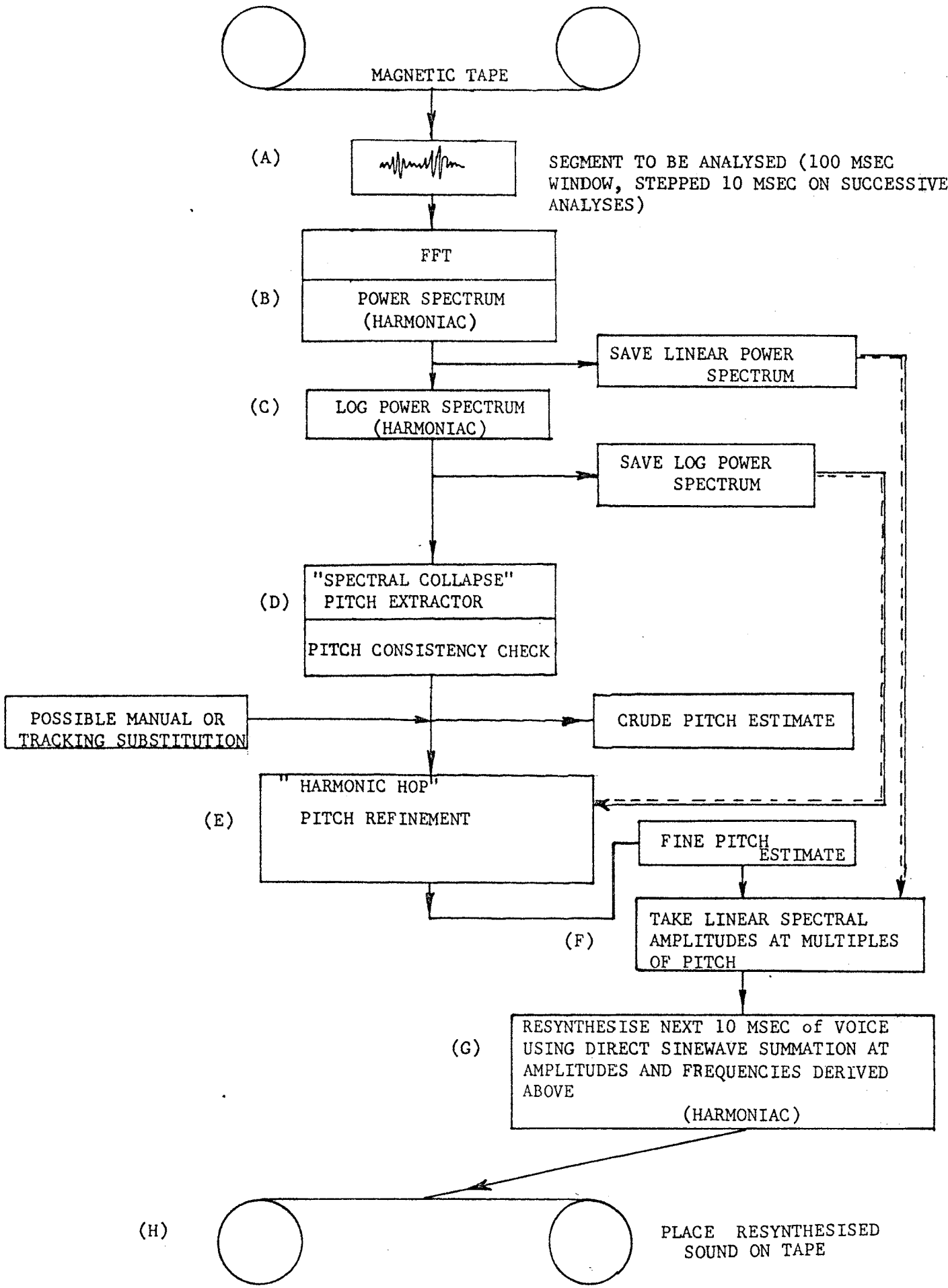


Figure 1. Block diagram of noise-reduction algorithm.

voice F_o increases and are not appropriate for the soprano voice in the upper part of its range. Moreover they do not work well in the presence of harmonic noise such as is provided by the old musical accompaniments.

In order to have the system cope well with all voice types a new algorithm for estimating voice F_o was designed. It has proved reliable in the presence of both surface noise and most instruments. The algorithm has some similarities with Schroeder's harmonic product method (2) and with other frequency-domain methods such as the SIFT algorithms (3) (4), but has the peculiarities of not being dependent on peaks in the spectrum and of giving effectively less weight to the upper harmonics.

For each point (IK) in the log power spectrum which exceeds in amplitude a variable preset threshold, starting at the left, an amplitude (KINC) is added to a collapsed-spectrum store, initially all zero, at frequencies which are integral approximations to the frequency of point (IK) divided by NI, where NI = 1,2,3.... NAHMAX. NAHMAX, the maximum number of analysed harmonics, is preset for each run.

When NI = 1, the (KINC) coming from point (IK) is equal to the spectral amplitude at point (IK). As NI increases, it becomes progressively less. The same process is repeated for each point in the spectrum which exceeds the amplitude threshold so that a set of (KINC)s is accumulated in each sub-multiple frequency position. When all is done, the frequency of the point of greatest amplitude in the collapsed spectrum can serve as a first estimate of the F_o of the strongest harmonic sound present. In acoustic vocal recordings this is almost always the voice.

The Fortran program relevant to this part of the spectral-collapse algorithm is shown in Appendix A, lines 329 to 393.

The accuracy of the first estimate is not great, being only +/- one point in the frequency spectrum. (With a 512-point spectrum and a 12.5 KHz sample rate, each point represents 12.5 Hz). Jn order to achieve greater accuracy, the original

log power spectrum is searched. The frequency of the second harmonic is estimated by doubling the first F_o estimate. The largest peak in the spectrum near this frequency and within the error bounds is assumed to be the second harmonic and from its frequency a more accurate second estimate of F_o is made. Should no significant harmonic be found in the expected region, the error bounds are increased by one point before moving on from it. The process is subsequently repeated on and on up the spectrum and it is the highest number significant harmonic, N, which is ultimately used to define F_o which is calculated in floating point for good accuracy.

The Fortran listing of this part of the algorithm in in Appendix A, lines 394 to 450.

The whole process is not excessively slow but, if the number of harmonics to be analysed and the number of spectral points to be treated are made large, it can be time-consuming on a normal computer. In the initial tests on acoustic recordings of Dame Nellie Melba, the recorded material was already bandlimited to about 2.5 KHz so only 256 points of the spectrum, i.e., about 3KHz, were analysed and NAHMAX was set at 7. The F_o estimation then took about 200 msec on the HP21MX, not including FFT and power spectrum.

RESYNTHESIS ALGORITHM

Once F_o is known to good accuracy it is easy to take the amplitudes of each component of the voice from the linear power spectrum (stage F, fig. 1) and then to resynthesize ten milliseconds of the waveform of the voice using machine language software in the Harmoniac. The resynthesis algorithm adds up a set of sine waves, which in this case are harmonically related, at the amplitudes in question. A block diagram is given in fig. 2.

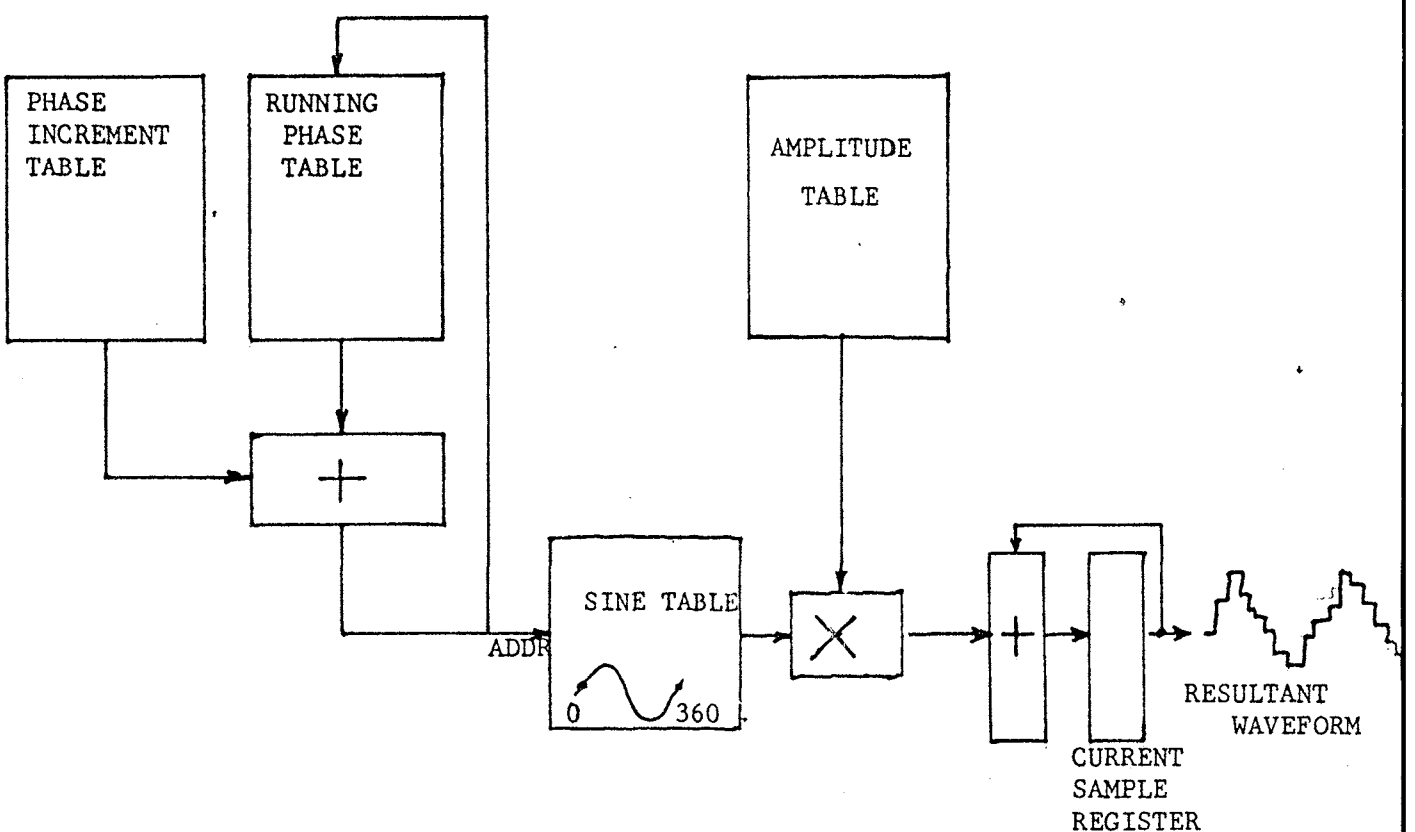


Figure 2. Block diagram of sine-wave synthesis algorithm (Harmoniac machine language)

The algorithm keeps a table of the current phases of each of the sines and, for each sample being generated, the phases are incremented by amounts which depend on the frequency of the component. Hence the frequencies of the components must be converted to phase increments before being given to the synthesis algorithm. The phase of each component is used to look up a sine table, giving the maximum amplitude of the component at that instant. This amplitude is multiplied by the overall amplitude of the component and the result is accumulated with all the other components to produce the resultant pressure wave sample. In this algorithm ten milliseconds of samples are stored in a buffer after each new set of frequencies and amplitudes are received. To avoid discontinuities in the waveform at the start of a ten millisecond interval, the new set is not used until the waveform passes through a positive zero crossing. See Appendix B.

Frequency calculations are done to thirty-one bit accuracy so there is no problem with audible frequency steps. Noise level in the generated sinewaves is approximately 65dB below peak signal level. The noise source is mainly from limitations on the number of phase steps (4096) and the number of amplitude steps in the sine table (2048 in the prototype of Harmoniac). This algorithm operates very quickly in Harmoniac machine language and is capable of producing up to ten sinusoids at a thirty microsecond sample rate in real time. It is used in the singing resynthesis in a buffered mode so that speed of execution is not important. Most of the execution time of the singing processing is spent in the pitch extraction algorithm and the FFT.

PERFORMANCE AND DIFFICULTIES

At present the process yields good results when the accompaniment is not prominent but further work is needed to make it fully viable.

No non-harmonic components such as occur in fricatives have yet been added. They are few in early recordings in any case but it is clearly desirable that

some at least be "sketched in". No difficulty is anticipated in using a speech synthesiser for this purpose.

Apart from this, there are situations when dealing with the harmonic signal which call for special attention.

F_o Errors:

Type 1 - Surface Noise

No F_o result is produced when the information in the log. power spectrum is below the threshold set for the spectrum-collapse algorithm. This is to avoid F_o errors due to surface noise, but it can leave abrupt terminations and initiations which sometimes need overriding operator intervention to make smooth.

Type 2 - Competing Voice

When the voice becomes lower in amplitude than the accompaniment the wrong F_o may be selected. An algorithm has been developed which is manually guided only for the first point of an F_o track and which thereafter uses the nearest large peak in the next spectral frame analysed each time until the voice stops or there is some other discontinuity. At present "starting points" for the F_o tracks are inserted as comments on the digital magnetic tape after a preliminary analysis/resynthesis precedure has been run to allow the operator to locate points of discontinuity.

Type 3 - Submultiple Voice

Accompanying instruments can cause difficulties even when their level is low if they have a fairly strong component at half the pitch of the voice. In this case the pitch extractor will build up this half pitch component as as a submultiple of the true fundamental, as its structure is such that it enhances all submultiples. In the case where only one or two harmonics of

the voice are present any algorithm could be forgiven for this type of error and in fact this is the condition which usually gives rise to it. Since all the correct harmonics are part of the series based on a half frequency fundamental the error is not quite as serious as others but unfortunately the creation of sudden new component at half the previous F_0 causes a discontinuity in the resynthesised waveform heard as a click.

To overcome this error the manually-guided F_0-tracking algorithm described above is again used. The incidence of the error is lower when the rate of change of harmonic amplitude is limited. See "Resynthesis Errors - Type 1".

Resynthesis Errors

Type 1 - Surface Noise

Some of the harmonics of the voice may be close in amplitude to the surface noise level and thus become audibly modulated by the surface-noise component in the analysis. At present this effect is reduced by the use of a resynthesis threshold causing components below a predetermined level in the linear power spectrum (which reflects recording signal to noise ratio) to be set at zero amplitude.

It has been found that if the surface noise is high,momentary peaks may still breakthrough this threshold and lead to objectionable intermittent false upper harmonics. These are noticeable as a sort of "high bubbling" when the signal-to-noise ratio is only 20 or 25 dB,but can occur to some extent at all signal-to-noise ratios.

To counter them further,the rate of change of harmonic amplitude is limited. The magnitude of the limitation represents a compromise between freedom from the false harmonics and ability to duplicate the original voice accurately. In the prototype only +/- 4.5 dB change in amplitude per 10 msec frame was allowed on each harmonic unless the amplitude of the harmonic in the previous frame was zero,in which case the initial amplitude of the component was set at the level of t

noise threshold. If all the components were zero in the previous frame, all harmonics were allowed to take the levels specified by the unmodified analysed spectrum.

Type 2 - Marginal Discontinuities

This is not strictly a noise, but rather the discontinuity which may occur at the end and sometimes the beginning of sounds - an abrupt passing of the noise threshold when the signal can no longer be determined reliably by the pitch extractor. It causes the sound to terminate or begin suddenly - typically at about - 25 dB for the Dame Nellie Melba tests. It should be possible to extend this dynamic range considerably by the use of an F_o tracking algorithm or by manual control, but this has not yet been fully explored.

Type 3 - Very Rapid F_o Change

This is caused by rapid F_o changes which take place in a time less than the effective length of one transform analysis window, i.e. 100 msec x ⅔ msec (factor of ⅔ caused by Hanning window used). Such rapid changes broaden the spectral peaks and can cause errors in estimated F_o. Vibrato seems to be adequately well tracked but the rapid F_o changes in the onsets of some plosives like 'b' and 'd' seem sometimes to be missed,with consequent 'slurring' of the consonant. It is proposed that this problem could be reduced by using an adaptive transform length which drops to one half or one quarter of its full length during rapid pitch or amplitude changes and in silences. The present implementation uses a 12.5 KHz sample rate and 1024 point transforms with 125 point (10 msec) hop between analysis frames. Complications in software caused by over-lapping operations on different frames in the two computers have so far prevented implementation of the adaptive transform length. Resynthesis of plosives is often considered adequate as it is.

CONCLUSION

The process described has already given very promising results and work continues on its refinement. While it cannot make intelligible a voice which is not intelligible, it has been shown able substantially to isolate and to improve the signal to noise ratio of a single voice recorded by the "acoustic" process. The cleaner the original recording the more successful the result.

The process seems likely to be of use for separating one voice from another but has not been applied as yet to this. It may also be of interest to writers of musique concrete to whom it gives the possibility to modify the discrete sound elements they assemble while in parametric form.

APPENDIX A

```
0329  C    PITCH DETERMINATION FROM PWR SPECTRUM
0330  C  USING COLLAPSED SPECTRUM METHOD
0331  2000  IPKLEV = 0
0332        KPKS = 0
0333  C  COMPENSATE AMPL THRESH FOR PESCALING BEFORE FFT
0334  C FIND NO OF SHIFTS DONE
0335        LTHRSH = LTHRH-IFIX((ALOG(FLOAT(MAXL))/ALOG2)*128.0 + 0.5)
0336  C THRESH MOVES DOWN WHEN DIVISOR IS LARGE AS SIGNAL IS STRONGER
0337  C IE THERE IS LESS NOISE VISIBLE IN SPECTRUM
0338        IF(LTHRSH.LT.1)LTHRSH = 1
0339        DO 2010 IK = 1,NQRT
0340        RBLOCK(IK) = 0
0341  2010  CONTINUE
0342  C  DO SUBMULTIPLE COLLAPSE OF SPECTRUM
0343  C  FIRST HALF OF SPECTRUM TO 3KHZ
0344  C THIS IS A TIME CONSUMING SECTION - APPROX 120 MSEC FOR 256 PTS.
0345        DO 2025 IK = PMIN,NQRT
0346        IF(DISBUF(IK).GT.IPKLEV) IPKLEV = DISBUF(IK)
0347        IF(DISBUF(IK).LT.LTHRSH) GO TO 2025
0348        DO 2025 NI = 1,NAHMAX
0349        IKL = ((( IK + IK )/NI )  + 1 ) /2
0350        IF(DISBUF(IKL).LT.LTHRSH) GO TO 2025
0351        IF(IKL.LT.PMIN.OR.IKL.GT.PMAX) GO TO 2025
0352  C  GIVE PRIMARY COMPONENT MORE PROMINENCE
0353  C
0354        KINC = DISBUF(IK)/NI
0355  C
0356  C  ADD UPPER COMPONENT TO SUBMULTIPLE
0357  2026  RBLOCK(IKL) = RBLOCK(IKL)  + KINC
0358  2025  CONTINUE
0359  C HISTOGRAM OF LOG SPECTRUM SUBMULTIPLES IS FINISHED
0360  C  GET HIGHEST ENTRY IN HISTOGRAM AS 1ST PITCH ESTIMATE
0361  2500  IEST = 0
0362        IPEST = 0
0363        DO 2510 IK = PMIN,PMAX
0364        IF(RBLOCK(IK).LE.IEST)GO TO 2510
0365        IEST = RBLOCK(IK)
0366        IPEST = IK
0367  2510  CONTINUE
0368  C  SET AS SILENCE IF CORRELATION OF SPECTRUM VERY POOR
0369        IF(IEST.LT.2*LTHRSH) IPEST = 0
0370  C CHECK FOR PITCH HALVING ERROR
0371        IPR = (IPEST*3.0 + 0.5)
0372        IPR1 = DISBUF(IPR)
0373        IPR2 = DISBUF(IPR + 1)
0374        IPR3 = DISBUF(IPR-1)
0375        IF(IPR2.GT.IPR1)IPR1 = IPR2
0376        IF(IPR3.GT.IPR1)IPR1 = IPR3
0377        IPESH = (IPESL + 1)/2
0378        IF(IPESH.EQ.IPEST.OR.IPESH+1.EQ.IPEST.OR.IPESH-1.EQ.IPEST
0379       1.AND.IPR1.LT.LTHRSH+20.AND.3*DISBUF(IPEST).LT.DISBUF(IPEST+
0380       2 IPEST = 2*IPEST
0381        IPESL = IPEST
0382        IPSAVE = IPEST
0383  C
0384  C CHECK FOR EDIT PITCH CORECTION
0385  C
0386        IND = IAND (PITCHT (96 + NO1), 377B)
0387        IF (IND .GT. 0) IPEST = IND
```

APPENDIX A (continued).

```
0388  C
0389  C SAVE VALUE IN PITCH TABLE
0390  C
0391        PITCHT (48 + NO1) = IPEST
0392  C
0393  C  NOW WE HAVE A CRUDE PITCH ESTIMATE
0394  C    USE INDIVIDUAL HARMONICS FROM BOT OF SPECTRUM UP TO
0395  C    PROGRESSIVELY REFINE THE PITCH ESTIMATE, STARTING AT FUND.
0396  C    ASSUME INITIAL ESTIMATE MAY BE +OR- 1
0397        NHARM = 1
0398        NER = 1
0399        NSH = 0
0400        NA = 8
0401        NPEST = IPEST
0402  C  LOOK AT EA HARMONIC ,REFINING EST IF WE CAN FIND MATCHING PEA
0403        NBEST = IPEST
0404  C LOOK FOR LARGEST PEAK INSIDE RANGE OF EA HARMONIC
0405        WFILE(1) = 1
0406  C
0407        DO 2560 LHARM = 1,NSHMAX
0408        IF((NPEST + NER).GE.INH) GO TO 2560
0409        NSH = NSH + 1
0410  C  THRESHOLD ON USEFULLNESS OF UPPER HARMS RISES W HARM NO.
0411        ITHRSH = IPKLEV/NA
0412  C   SET UP DEFAULT FOR NPES AS ESTIMATED HARMONIC POSITION
0413  2540  NPES = NPEST
0414        LPT = 0
0415  C TRY EA PT AROUND EXPECTED POSITION OF THE HARMONIC
0416        DO 2550 IK = (NPEST-NER),(NPEST + NER)
0417        IF(IK.LT.1) GO TO 2550
0418        IF(DISBUF(IK).LT.LPT) GO TO 2550
0419  C GOT LARGEST
0420        LPT = DISBUF(IK)
0421        IF(LPT.GT.ITHRSH) NPES = IK
0422  2550  CONTINUE
0423  C GOT LGEST PK
0424  C   ESTIMATE POSITION OF NEXT HARMONIC
0425  C   USING ROUNDING
0426        NPEST = NPES + (( NPES+NPES )/ LHARM +1 ) / 2
0427  C  PUT PRESENT HARM FRQ IT TABLE FOR RESYNTH
0428        WFILE(LHARM) = NPES
0429        IF(LPT.GT.ITHRSH) GO TO 2555
0430  C NO SIGNIFICANT PEAK FOUND- GREATER POSS ERROR
0431        NER = NER + 1
0432        GO TO 2556
0433  C USE PK TO REFINE NPES
0434  2555  NER = 1
0435  C MAKE A NOTE OF BEST RELIABLE ESTIMATE SO FAR
0436        NBEST = NPES
0437        NHARM = LHARM
0438  2556  NA = NA-2
0439        IF(NA.LT.2) NA = 2
0440  2560  CONTINUE
0441  C  AT END OF ALL PEAKS , CALC F.P. PITCH
0442  2561  CONTINUE
0443        PITCH = (FLOAT(NBEST)*FRESLN)/(FLOAT(NHARM))
0444        IF(PITCH.LT.PITMIN) PITCH = 0.0
0445        WRITE(1,2531)IPSAVE, IPEST, IND
0446  2531  FORMAT("CRUDEP A",I6," B ",I6," C ",I6)
0447        WRITE(1,2530)PITCH
0448  2530  FORMAT("PITCH ",F7.2)
0449        GO TO ISUBR
0450  C
```

APPENDIX B

```
 2  * RUNNING SINUSOID SUMMATION SYNTHESIS
 3  *      =       ,
 4  * USES DOUBLE PRECISION PHASE ADDITION
 5  * FOR HIGH ACCURACY PITCH CONTROL.
 6  * AMPL SUMMATION IS S.PRECISION.
 7  * RUNS AS SUBROUTINE THAT CALCS NHOP
 8  * PTS OF WAVEFORM ON EA.  CALL(STACKED)
 9 01CA 9008  MC                8       #SINSUM  ENTRY   IM   4   0   8
10 01CB 9400  AC                0  IM7 SET               IM   5   0   0
11  * POP THE CNT/ADDR PARAMETERS
12  **SET UP TO DO "NHOP" SAMPLES
13 01CC BC15  M1D/AD  NOP       #ENDRES                  IM  15   0  21
14 01CD 54E0  #T1     MDAT      NOP          PTS TO END      21   7   0
15  * SET UP PTR TO RESULTS STACK IN M1
16 01CE 58E0  #T2     MDAT      NOP                          22   7   0
17 01CF FC1C  M1D/AD            #NSINES #SAMLOP          IM  15   0  60
18 01D0 E1F7  ADD     MDAT      #PHASES                  IM   8   7 119
19 01D1 5063  #CNT    ALU       ALU                          20   3   3
20 01D2 B4E1  TRANS   MDAT      1 IS OVERALL AMP         IM  13   7   1
21 01D3 A460  AND     ALU       0                        IM   9   3   0
22 01D4 4063  #ALUS   ALU       ALU  USED FOR SAMPLE        16   3   3
23 01D5 D917  MADDR             #PHASES                  IM   6   0 119
24 01D6 9009  MC                9        POP M1,PUSH 0   IM   4   0   9
25 01D7 20E7  ADD     MDAT      MDAT ADD LSPS                8   7   7
26 01D8 1C03  MDAT    NOP       ALU      #SINLOP  *           7   0   3
27 01D9 0000  NOP                           WAIT ON MO PU    0   0   0
28 01DA 2027  ADD     AFLG      MDAT                *       8   1   7
29 01DB 20E3  ADD     MDAT      ALU       ADD MSPS *         8   7   3
30 01DC 1C03  MDAT              ALU  PUSH MSP               7   0   3
31 01DD A068  ADD     ALU       8 ROUND FOR SINE         IM   8   3   8
32  *
33  * NOTE 65DB S/N POSS W 11B*1024 SINE
34 01DE 9478  SIN/AC  ALU       24  ROV&SEE OV & >=      IM   5   3  24
35  * MPY BY AMPL OF THIS COMPONENT
36 01DF 30ED  MPY     MDAT      SIN                          12   7  13
37 01E0 2090  ADD     MPYH      #ALUS  UPDATE SAMPLE         8   4  16
38 01E1 4063  #ALUS   ALU       ALU                          16   3   3
39 01E2 3854  COMP    MOADDR    #CNT                         14   2  20
40 01E3 8578  JMP               #SINLOP                  IM   1   0 472
41 01E4 20E7  ADD     MDAT      MDAT  NEXT LSP ADD           8   7   7
42  * 12 INSTRUCTION INNER LOOP
43  * NOW CHECK IF ZERO CROSSING SO CAN
44  * CHANGE PARAMETERS WITHOUT CLICK
45 01E5 BA21  COMP    #DPO      1  EQUIV TO >0             IM  14  17   1
46 01E6 C56A  JMP               #NEG  EXEC IF <=0 NOW    IM   1   0 490
47 01E7 0000  NOP                                           0   0   0
48 01E8 C574  JMP               #CONTIN  EXCE IF IS >0   IM   1   0 500
49 01E9 0000  NOP                                           0   0   0
50 01EA BA00  COMP    #ALUS     0        #NEG  >=TEST     IM  14  16   0
51 01EB C574  JMP               #CONTIN #CONTI  IF<0 LA  IM   1   0 500
52  * BLOCK MOVE OF CONTROL PARAMS
53 01EC FC1C  M1D/AD            #NSINES                  IM  15   0  60
54 01ED 9819  MADDR             #PARIN (HOST INPUT)      IM   6   0  25
55 01EE 9018  MC                24  PUSH1,POP0           IM   4   0  24
56 01EF 22E7  ADD     #ZRO      MDAT                         8  23   7
57 01F0 1C60  MDAT    ALU       NOP      #MOVP               7   3   0
58 01F1 F956  COMP    MOADR     #ENDPIN                  IM  14   2 118
59 01F2 C570  JMP               #MOVP                    IM   1   0 496
60 01F3 22E7  ADD     #ZRO      MDAT                         8  23   7
```

APPENDIX B continued.

```
61   * (20 PARAM SETS ALLOWED)
62   * MPY BY OVERALL AMPL
63 01F4 3205  MPY      #ALUS    TRANS    #CONTIN  AMPL          12  16   5
64 01F5 A2C1  ADD      #T2      1                          IM    8  22   1
65 01F6 3C83  M1D/AD   MPYH     ALU                          15   4   3
66 01F7 4480  #DPO     MPYH     NOP                          17   4   0
67 01F8 5863  #T2      ALU      ALU                          22   3   3
68 01F9 36A3  TRANS    #T1      ALU                          13  21   3
69 01FA 38A5  COMP     TRANS    TRANS  REACHED END ?          14   5   5
70 01FB 856F  JMP                #SAMLOP                   IM   1   0 463
71 01FC 9009  MC                 9 RESTORE MEM STATUS      IM   4   0   9
72 01FD 0401  JMP                RETURN                         1   0   1
73 01FE 0000  NOP                                               0   0   0
74   DATA FILE FOR MAIN MEM BEGINS  @M1,20
75 0014 0100 256       #NHOP
76 0015 0900 2304      #ENDRES
77 0016 0800 2048      #RPTR
78   * END RESULTS AREA OF M1
79 0017 0014 20        #LITTLE
80   DATA FILE FOR MAIN MEM BEGINS  @M0,25  PARAMETERS INPUT FILE
81 0019 0002 2         #PARIN
82 001A 07D0 2000
83 001B 0000 0
84 001C 00C8 200
85 001D 01F4 500
86   *ENDS AT 118
87   DATA FILE FOR MAIN MEM BEGINS  @M0,118   RUNNING PHASE TABLE
88 0076 0000 0         #ENDPIN
89 0077 0000 0         #PHASES
90 0078 0000 0                  MSP
91   DATA FILE FOR MAIN MEM BEGINS  @M1,60 ** PARAMETERS WORKING FILE
92 003C 0002 2         #NSINES  TWICE NO. OF SINES REQ
93 003D 03E8 1000      #AMPLA   OVERALL AMPL
94 003E 0000 0         #PHIAMP   PHASE INC LSP
95 003F 07D0 2000              PHASE INC MSP
96 0040 1388 5000              AMPL OF THIS COMPONENT
97   * THREE WORDS DESCRIBE EA COMPONENT
```

REFERENCES

1. P.M. Connor, "Harmoniac - A Digital Signal Processor" S.L.R.C. Working Papers, Macquarie University, 1981.

2. M.R. Schroeder, "Period Histogram and Product Spectrum: New Methods for Fundamental Frequency Measurement", JASA, Vol. 43, No.4, January, 1968.

3. A.M. Noll, "Pitch Determination of Human Speech by the Harmonic Product Spectrum, The Harmonic Sum Spectrum and a Maximum Likelihood Estimate," presented at the Symposium on Computer Processing in Communications, Polytechnic Institute of Brooklyn, Brooklyn, N.Y. Apr. 8 - 10, 1969.

4. J.D. Markel, "The Sift Algorithm for Fundamental Frequency Estimation", I.E.E.E., T.A.E., Vol. AU-20 December, 1972.

5. Seneff, Stephanie, "A Real Time Harmonic Pitch Detector" I.E.E.E., T - A.S.S.P., Vol. 26, No. 4, August 1978.

6. R.L. Miller, "Performance Characteristics of an Experimental Harmonic Identification Pitch Extraction (Hipex) System", J.A.S.A., Vol. 47, June 1970.

7. W.H. Tucker, "A Pitch Estimation Algorithm for Speech and Music" I..E.E.E., T-A.S.S.P., Vol. 26, No.6, Dec. 1978.

8. T.W. Parsons, "Separation of Speech from Interfering Speech by means of Harmonic Selection", J.A.S.A., Vol. 60, No. 4, October 1976.

APPENDIX IV

```
      PAGE  0001          FTN4 COMPILER: HP24177  (SEPT. 1974)


0001  FTN,L,C
0002        SUBROUTINE HASSY
0003  C                      ON FILE "HARA7"
0004  C               BY  P.M.CONNOR  MACQUARIE UNIV S.L.R.C.
0005  C                    REVISION 7TH SEPT 1979 (WARNINGS,LAST ADDR)
0006        COMMON RFILE
0007        COMMON LLINE
0008        INTEGER RFILE(20,100),FNUMB,COMAND,FNAME(3),HLIST(3)
0009        INTEGER HEXL(4),HEXA(4)
0010        INTEGER LLINE(80)
0011        INTEGER AFNAME(3)
0012        INTEGER HOBJN(3)
0013        INTEGER OFILE(1000),TNAME(3),OFNAME(3),CFILE(20,100)
0014        INTEGER ERFILE(20)
0015        INTEGER SYMBT(7,100)
0016        INTEGER SCRSYM(6,17)
0017        INTEGER DEST(20),SCE1(19),SCE0(19)
0018        INTEGER ERD,ERS0,ERS1,COFIL(1000)
0019        DATA TNAME/2HHM/,OFNAME/2HHB/
0020        DATA ERFILE/2HMP,19*0/
0021        DATA DEST/2HNO,2HJM,2HIO,2HOD,2HRS,2HSI,2HMA,2HMD,
0022       #2HAD,2HAN,2HSU,2HOR,2HMP,2HTR,2HCO,2HM1,2HSC,2HCA,2HJS,2H
0023        DATA SCE1/2HNO,2HAF,2HMO,2HAL,2HMP,2HTR,2HM1,2HMD,
0024       #2HZ1,2HZ2,2HI1,2HI3,2HRS,2HZ3,2HZ4,2HZ5,2HSC,2HJS/
0025        DATA SCE0/2HNO,2HPS,2HI2,2HAL,2HMP,2HTR,2HI4,2HMD,
0026       #2HZ1,2HZ2,2HZ3,2HSO,2HSM,2HSI,2HBU,2HRM,2HSC,2HRE/
0027        DATA IRCDE/10/,IBEEP/3400B/,ICON/103B/
0028        DATA HLIST/2HHL,2HIS,1HT/,FNAME/2HHM/
0029        DATA ICLR/15473B/
0030        DATA AFNAME/2HHM/
0031        DATA ICON4/1607B/
0032        DATA HOBJN/2HHO,2HBJ,2H1 /
0033        ISTRT=-2
0034  20    WRITE(1,21)IBEEP,IBEEP
0035  21    FORMAT(A2,"NAME OF PROG TO BE ASSEMBLED(4KEY CH)..."
0036       #A1,"_@")
0037        READ(1,22)FNAME
0038  22    FORMAT(3A2)
0039        IF(FNAME(1).EQ.2H  ) RETURN
0040        LU=1
0041        LI=1
0042        WRITE(1,845)
0043  845   FORMAT (" LIST ON VT(1),LP(6) OR NONE(0)?","_")
0044        READ(1,*)LU
0045        IF(LU.EQ.0) LU=99
0046        IF(LU.NE.99)LI=LU
0047        WRITE(1,848)
0048  848   FORMAT("SAVE ON DISC(D) OR SEND TO HARMONIAC(H)?","_m")
0049        READ(1,849)LOBJD
0050  849   FORMAT(A1)
0051        KZRO=0
0052        DO 100 FNUMB=0,99
0053  90    CALL ASCII(FNUMB,AFNAME)
0054        CALL EXEC(18,AFNAME,ISECT)
0055        IF(ISECT.EQ.0)GOTO 120
0056        CALL EXEC(14,ICON,RFILE,2000,AFNAME,0)
```

```
0057          IF(RFILE(1,1).EQ.0) KZRO=KZRO+1
0058          IF(KZRO.GT.20) GO TO 120
0059          DO 99 J=1,2
0060          IF(RFILE(J,1).NE.FNAME(J))GOTO 100
0061    99    CONTINUE
0062    C RESET LIST LINE COUNT WHEN CHANGING FILES FOR READER'S CONVENIE
0063          LINE=2
0064          GOTO 140
0065    100   CONTINUE
0066    120   WRITE(1,130)
0067    130   FORMAT(/"FILE DOES NOT EXIST , TRY AGAIN.........."/)
0068          GO TO 903
0069    140   IF(ISTRT.GE.0) GO TO 141
0070    C  GENERATE A JUMP SYMBOL TABLE
0071          IF(ISTRT.GE.-1) GO TO 142
0072          IADD=0
0073          ISTRT=-1
0074          ISYPT=1
0075          DO 3003 K=1,700
0076    3003  SYMBT(K)=1H
0077    C INITIALISE BOTH JMP &SCR SYMBOL TABLES
0078          DO 3004 K=1,102
0079    3004  SCRSYM(K)=1H
0080          NSYMB=0
0081    C  NOW BUILD SYMBOL TABLE FOR CURRENT FILE
0082    C   -JUMP & MAIN MEM ADDR SYMBOLS
0083    142   JSTRT=20
0084          DO 2901 K=2,100
0085          IF(RFILE(1,K).EQ.0) GO TO 3001
0086          CALL PULLH(K)
0087          IF(LLINE(1).EQ.1H^.OR.RFILE(1,2).EQ.0) GO TO 3100
0088          IF(LLINE(1).EQ.1H$) GO TO 3000
0089    C SET PADDR WHEN WE FIND IT
0090          IF(LLINE(1).NE.1H@)GOTO 3007
0091          IF(LLINE(2).EQ.1HP) JSTRT=20
0092          IF(LLINE(2).EQ.1HM) JSTRT=6
0093          CALL NUMB(LLINE,5,IADD,ISTAT,0,8192)
0094    3007  DO 3005 J=1,40
0095    C EA LINE SCANNED FOR "#"   (IGNORE 1ST 20 CHS)
0096          IF(LLINE(J).EQ.1H*)GO TO 3000
0097          IF(J.LT.JSTRT) GO TO 3005
0098          IF(LLINE(J).NE.1H#)GO TO 3005
0099    C GOT A LABEL - STACK IT
0100          SYMBT(ISYPT+6)=IADD
0101          JS=0
0102          DO 3011 JT=J+1,J+6
0103          SYMBT(ISYPT+JS)=LLINE(JT)
0104          JS=JS+1
0105          IF(LLINE(JT).EQ.1H ) GO TO 3012
0106          IF( ISYPT.LT.700) GO TO 3011
0107          WRITE (LI,3020)
0108    3020  FORMAT("1    SYMBOL TABLE OVERFLOW HAS OCCURRED !!!!!")
0109          GO TO 3100
0110    3011  CONTINUE
0111    3012  ISYPT=ISYPT+7
0112          NSYMB=NSYMB+1
```

```
0113          GOTO 3000
0114    3005  CONTINUE
0115    3000  IF(LLINE(1).NE.1H@.AND.LLINE(1).NE.1H*.AND.LLINE(1).NE.
0116         #1H$.AND.LLINE(1).NE.0)  IADD=IADD+1
0117          IF(LLINE(1).NE.1H$) GO TO 3001
0118    C HERE DOING HI PRIORITY SCRATCH SYMBOL LIST (MUST BE 1ST IN PROG
0119          KPRC=2
0120          DO 317 KPRC=KPRC,40
0121          IF(LLINE(KPRC).EQ.1H#) GO TO 316
0122          GO TO 317
0123    C  INSERT HI PRIORITY SCR SYM (6 CHS)IF IT'S NOT IN
0124    316   DO 381 KSCY=1,16
0125          DO 382 JSC=1,6
0126          IF(SCRSYM(1,KSCY).EQ.1H ) GO TO 319
0127          IF(LLINE(JSC+KPRC).NE.SCRSYM(JSC,KSCY))GO TO 381
0128          IF(SCRSYM(JSC,KSCY).EQ.1H ) GO TO 317
0129    382   CONTINUE
0130    C GOT MATCH -LEAVE ALONE
0131          GO TO 317
0132    381   CONTINUE
0133          GO TO 384
0134    319   DO 318 KPCC=1,6
0135          SCRSYM(KPCC,KSCY)=LLINE(KPRC+KPCC)
0136    318   CONTINUE
0137          GO TO 317
0138    384   WRITE(LI,3022)
0139    3022  FORMAT("/   PRIORITY SCRATCH TABLE OVERFLOW !")
0140          GO TO 3001
0141    317   CONTINUE
0142    3001  IF(LLINE(1).NE.1H#) GO TO 2901
0143    C DEST IS SCRATCH SYMBOL-PUT IT IN TABLE IF NOT ALREADY THERE
0144          DO 3510 KSCY=1,16
0145          DO 3520 JSC=1,6
0146          IF(SCRSYM(1,KSCY).EQ.1H ) GOTO 3570
0147          IF(LLINE(JSC+1).NE.SCRSYM(JSC,KSCY)) GO TO 3510
0148          IF(SCRSYM(JSC,KSCY).EQ.1H ) GO TO 2901
0149    3520  CONTINUE
0150    C   MATCH FOUND
0151          GOTO 2901
0152    3510  CONTINUE
0153    C IF TABLE FULL & NO MATCH->ERROR
0154          WRITE(LI,2904)
0155    2904  FORMAT("/ TOO MANY SCRATCH SYMBOLS !")
0156          GO TO 2901
0157    C PUTTING IN A NEW SYMBOL
0158    3570  DO 3560 JSC=1,6
0159    3560  SCRSYM(JSC,KSCY)=LLINE(JSC+1)
0160    2901  CONTINUE
0161    C FILE FINISHED - GET ANOTHER
0162          GO TO 100
0163    C NOW FINISHED BUILD OF SYMBOL TABLES SO
0164    C GO BACK TO 1ST FILE TO ASSEMBLE ALL FILES
0165    3100  ISTRT=0
0166          GO TO 849
0167    C
0168    C SYMB TABLE FINISHED - INITIALISE FOR ASSY
```

```
0169  C
0170  141    LFC =1
0171           IF(RFILE(1,1).EQ.0) GO TO 903
0172           IF(ISTRT.EQ.1)GOTO 899
0173           ISTRT=1
0174  C ZERO OUTPUT FILE
0175           DO 152 J=1,1000
0176           COFIL(J)=0
0177  152      OFILE(J)=0
0178  C      INSERT PROGRAM NAME IN OBJECT OUTPUT FILE
0179  150      DO 200 J=1,3
0180  200      OFILE(J)=RFILE(J,1)
0181  C ADDR IS WITHIN P MEM
0182  C  SET UP A DEFAULT ADDR IN CASE NO ORG
0183           OFILE(5)=070000B
0184           IADD=070000B
0185           IADDR=0
0186           IKNTP=4
0187           IDAT=0
0188           KNTWD=0
0189  C    OBJ FILE INSERT PTR IS KRES
0190           KRES=6
0191           LINE=2
0192           LDEST=0
0193           LPC=1
0194           KER=0
0195           KWARN=0
0196           KLIN=1
0197           IF(LU.EQ.99) GO TO 899
0198           WRITE(LU,890)LPC,(RFILE(JK,1),JK=1,20)
0199           IF(ISTAT.GT.0)WRITE(LU,602)
0200  602      FORMAT("START ADDR IS OUT OF RANGE (0-8192)")
0201  890      FORMAT("1  PAGE  ",I3 "    HARMONIAC ASSEMBLY OF :  "
0202         2,20A2/,"  LINE ADDR MEM.       SOURCE CODE          ",
0203         3" #LABELS(JMP) *CMTS      DECODED OBJECT"/)
0204  C
0205  C
0206  C  ONLY 1ST LINE (NAME OF PROG) IS IGNORED DURING ASSEMBLY.
0207  C                    ASSEMBLY START
0208  899      DO 1000 J=2,100
0209  C OBJ BUILD LOOP
0210           ERD=0
0211           ERS0=0
0212           ERS1=0
0213           LWARN=0
0214           KIM=0
0215           ISCRP=0
0216           CALL PULLH(J)
0217           IF(LLINE(1).EQ.1H$) GO TO 793
0218  C CHECK IF IT'S AN EMPTY END OF FILE - SKIP THRU
0219  315      IF(RFILE(1,J).EQ.0) GO TO 1000
0220           IF(LLINE(1).EQ.1H^) GO TO 695
0221           IF(LLINE(1).EQ.1H*) GO TO 793
0222  C CHECK FOR NEW ORG
0223           IF(LLINE(1).EQ.1H@) GO TO 988
0224  C  CHECK IF DOING A DATA AREA
```

```
0225          IF(IDAT.EQ.1HD) GO TO 981
0226    C  NOW START ON PROGRAM ASSEMBLY
0227    C RESET DEFAULT ORG INDIC AS WE ARE ABOUT TO DO SOME PROGRAM
0228          KORGI=0
0229    C  (CAN NO LONGER OVERWRITE DEFAULT
0230    C
0231    C DO DEST FIRST( CHECK IF SYMBOLIC SCRATCH)
0232    C
0233          IF(LLINE(1).NE.1H#) GO TO 358
0234    C DEST IS SCR SYMBOL-CHECK IF ALREADY IN SCRSYM-PUT IN IF NOT((OR
0235          DO 351 KSCY=1,16
0236          DO 352 JSC=1,6
0237          IF(SCRSYM(1,KSCY).EQ.1H ) GOTO 357
0238          IF(LLINE(JSC+1).NE.SCRSYM(JSC,KSCY)) GO TO 351
0239          IF(SCRSYM(JSC,KSCY).EQ.1H ) GO TO 353
0240    352   CONTINUE
0241    C   MATCH FOUND
0242    353   KDAD=KSCY-1
0243          GO TO 354
0244    351   CONTINUE
0245    C IF TABLE FULL & NO MATCH->ERROR
0246          ERD=4
0247          KDAD=0
0248          GO TO 354
0249    C PUTTING IN A NEW SYMBOL
0250    357   DO 356 JSC=1,6
0251    356   SCRSYM(JSC,KSCY)=LLINE(JSC+1)
0252          GO TO 353
0253    C  NORMAL "OP2 DESTINATION PROCESSING FOLLOWS
0254    358   DO 300 KDAD =1,20
0255          IF(RFILE(1,J).EQ.DEST(KDAD)) GOTO 320
0256    300   CONTINUE
0257          IF(RFILE(1,J).NE.2HMC) GO TO 2800
0258          KDAD=5
0259          GOTO 320
0260    2800  IF(RFILE(1,J).NE.2HAC) GO TO 2801
0261          KDAD=6
0262          GO TO 320
0263    2801  CONTINUE
0264          IF(RFILE(1,J).NE.2HGO) GO TO 301
0265          KDAD=2
0266          GO TO 320
0267    301   KDAD=1
0268    302   ERD=1
0269    320   KDAD=KDAD-1
0270          IF(KDAD.EQ.16) GO TO 350
0271          IF(KDAD.EQ.17)KDAD=0
0272          IF(KDAD.EQ.18)KDAD=0
0273          IF(KDAD.NE.19) GO TO 303
0274          KDAD=0
0275          KDA=0
0276          KS1AD=0
0277          KS1=0
0278          KSOAD=0
0279          KSO=0
0280          GO TO 600
```

```
0281  303    CONTINUE
0282         GO TO 360
0283  350    CALL PULLH(J)
0284  C  SCRATCH DEST PROCESS
0285         CALL NUMB(LLINE,4,KDAD,ERD,0,15)
0286  354    ISCRP=1
0287         KDAD=KDAD+16
0288  C NOW CHECK FOR FORBIDDEN SEQUENCE OF DESTS
0289  360    IF(RFILE(1,J).EQ.LDEST) ERD=2
0290         LDEST=0
0291         DO 370 KERP=1,20
0292         IF(RFILE(1,J).EQ.ERFILE(KERP))LDEST=ERFILE(KERP)
0293  370    CONTINUE
0294  C
0295  C    NOW DO SOURCE ONE
0296  C
0297         KDA=KDAD
0298  C CHECK FOR SYMBOLIC SCR SOURCE 1
0299         IF(LLINE(9).NE.1H#) GO TO 458
0300  C IT IS SYMBOLIC SCR,FIND IT & REPLACE WITH ADDR
0301         DO 451 KSCY=1,17
0302         DO 452 JSC=1,6
0303         IF(LLINE(9+JSC).NE.SCRSYM(JSC,KSCY)) GO TO 451
0304         IF(SCRSYM(JSC,KSCY).EQ.1H ) GO TO 453
0305  452    CONTINUE
0306  C  MATCH
0307  453    KS1AD =KSCY-1
0308         GOTO 454
0309  451    CONTINUE
0310         ERS1=5
0311         KS1AD=0
0312         GO TO 454
0313  458    DO 400 KS1AD=1,18
0314         IF(RFILE(5,J).EQ.SCE1(KS1AD)) GO TO 420
0315  400    CONTINUE
0316         IF(RFILE(5,J).NE.2H  ) GO TO 401
0317         KS1AD=0
0318         GO TO 460
0319  401    KS1AD = 1
0320  402    ERS1 = 1
0321  420    KS1AD=KS1AD-1
0322         IF(KS1AD.EQ.16) GO TO 450
0323         IF(KS1AD.EQ.17) KS1AD=1
0324         GOTO 460
0325  450    CALL NUMB(LLINE,12,KS1AD,ERS1,0,15)
0326  454    KS1AD=KS1AD+16
0327         IF(ISCRP.NE.0)ERS1=99
0328  C LATER DO SOURCE ONE ERROR CHECK
0329  C
0330  C  NOW SOURCE ZERO
0331  C
0332  460    KS1=KS1AD
0333         IF(RFILE(9,J).EQ.2H  ) GO TO 533
0334         IF(LLINE(17).GE.1H0.AND.LLINE(17).LE.1H9) GO TO 544
0335         IF(LLINE(17).NE.1H#) GO TO 558
0336  C SYMBOLIC SCE ZERO
```

```
0337  C CHECK IF IT IS JMP SYMBOL OR SCR SYMBOL
0338        IADI=0
0339        IF(KDAD.EQ.1.AND.KS1AD.EQ.0) GO TO 4000
0340        IF(KDAD.EQ.0.AND.KS1AD.EQ.0)GO TO 4000
0341  C SEE IF IT'S MEM ADDR 1ST
0342        GO TO 3990
0343  4070  DO 551 KSCY =1,17
0344        DO 552 JSC=1,6
0345        IF(LLINE(17+JSC).NE.SCRSYM(JSC,KSCY))GO TO 551
0346        IF(SCRSYM(JSC,KSCY).EQ.1H ) GO TO 553
0347  552   CONTINUE
0348  C  MATCH
0349  553   KSOAD=KSCY-1
0350        GO TO 554
0351  551   CONTINUE
0352  C  NO SYMBOL WAS FOUND -ERROR
0353        KSOAD=1
0354        ERSO=3
0355        GO TO 520
0356  C  SET UP IMMEDIATE JUMP TO LOCATION GIVEN IN JMP SYMBOL TABLE(O-
0357  C OR MEM ADDR LABEL FIND
0358  3990  IADI=1
0359  4000  DO 4001 KSCY=1,NSYMB
0360        DO 4010 JSC=1,6
0361        IF(LLINE(17+JSC).NE.SYMBT(JSC,KSCY)) GO TO 4001
0362        IF(SYMBT(JSC,KSCY).EQ.1H ) GO TO 4011
0363  4010  CONTINUE
0364  C GOT FULL MATCH
0365  4011  KSOAD=SYMBT(7,KSCY)
0366        IF(KSOAD.LT.0.OR.KSOAD.GT.2047) ERSO=101
0367        KSO=KSOAD
0368        IF(IADI.EQ.1) GO TO 545
0369        GO TO 4060
0370  4001  CONTINUE
0371  C NO SYMBOL WAS FOUND TO MATCH - POSSIBLY A SCR SYMB -TRY
0372        GO TO 4070
0373  C NORMAL SCR O OPERATION SCAN
0374  558   DO 500 KSOAD=1,18
0375        IF(RFILE(9,J).EQ.SCEO(KSOAD)) GO TO 520
0376  500   CONTINUE
0377  501   KSOAD=1
0378  502   ERSO=1
0379  520   KSOAD=KSOAD-1
0380        IF(KSOAD.EQ.16) GOTO 550
0381        IF(KSOAD.EQ.17)KSOAD=1
0382        KSO=KSOAD
0383        GO TO 600
0384  C SET SCEO =SCE1 IF NO SCEO SPECIFIED
0385  533   KSOAD=KS1AD
0386        KSO=KSOAD
0387        GO TO 600
0388  C NUMERIC SOURCE ZERO INDICATES IMMEDIATE MODE
0389  544   CALL NUMB(LLINE,17,KSOAD,ERSO,0,2047)
0390        KSO=KSOAD
0391        IF(ISCRP.NE.0)ERSO=97
0392  C  CHECK IF IT'S AN IM JMP->NO SCE 1 AT ALL
```

PAGE 0008 HASSY FTN4 COMPILER: HP24177 (SEPT. 1974)

```
0393  C USE SCE1 BITS 0,1,2,4 TO GIVE 11 FOR JMPS
0394  C
0395        IF(KDAD.NE.1.AND.KDAD.NE.0)GO TO 545
0396  C IT'S A JMP (OR JSUBR) SO INSERT EXTRA IMMED BITS
0397  C BIT 7 :
0398  4060  KT=IAND(KSOAD,000200B)
0399        KSOAD=KSOAD-KT
0400        KT=KT/128
0401        KS1AD=IOR(KS1AD,KT)
0402  C  BIT 8 :
0403        KT=IAND(KSOAD,000400B)
0404        KSOAD=KSOAD-KT
0405        KT=KT/128
0406        KS1AD=IOR(KS1AD,KT)
0407  C  BIT 9 :
0408        KT=IAND(KSOAD,001000B)
0409        KSOAD=KSOAD-KT
0410        KT=KT/128
0411        KS1AD=IOR(KS1AD,KT)
0412  C  BIT 10 :
0413        KT=IAND(KSOAD,002000B)
0414        KSOAD=KSOAD-KT
0415        KT=KT/64
0416        KS1AD=IOR(KS1AD,KT)
0417        GO TO 589
0418  545   CONTINUE
0419  C     THERE WAS A CHECK HERE FOR 7 BIT IMMED INDICATOR CHAR -REMO
0420        IF(KS1AD.GT.23)  LWARN=24
0421        IF(KS1AD.GT.7.AND.KS1AD.LT.16) LWARN =15
0422         IF(KSOAD.GT.127) ERSO=127
0423  C REMOVE BIT 6 FROM SCE 0
0424  589   KT=IAND(KSOAD,000100B)
0425        KSOAD=KSOAD-KT
0426        KT=KT/8
0427  C TO BIT 3 IN SCE 1
0428        KS1AD=IOR(KS1AD,KT)
0429  C  CANNOT USE BIT 4 OF DEST DURING IM
0430  555   CONTINUE
0431        IF(KDAD.GT.15) ERD=6
0432        KT=IAND(KSOAD,000040B)
0433        KSOAD=KSOAD-KT
0434        KT=KT/2
0435  C  INSERT BIT 5 SCE 0 IN DEST BIT 4
0436        KDAD=IOR(KDAD,KT)
0437        KIM=100000B
0438        GO TO 600
0439  C  THIS IS FOR SCRATCH RAM 0
0440  550   CALL NUMB(LLINE,20,KSOAD,ERSO,0,15)
0441  554   KSOAD=KSOAD+16
0442        KSO=KSOAD
0443        IF(ISCRP.NE.0)ERSO=99
0444  600   KS1AD1=KS1AD*32
0445        KSOAD1=IOR(KSOAD,KS1AD1)
0446        KDAD1=KDAD*1024
0447        KSOAD1=IOR(KSOAD1,KDAD1)
0448        KSOAD1=IOR(KSOAD1,KIM)
```

```
0449  C
0450  C INSERT ASSEMBLED OBJECT WORD IN OUTPUT FILE
0451  C
0452        OFILE(KRES)=KSOAD1
0453        KRES=KRES+1
0454        IF(ERD.NE.0.OR.ERSO.NE.0.OR.ERS1.NE.0)KER=KER+1
0455        IF(LWARN.NE.0) KWARN=KWARN+1
0456  793   CONTINUE
0457        IF(LU.EQ.99) GO TO 630
0458        IM=2H
0459        IF(KIM.EQ.100000B) IM=2HIM
0460        IF(KLIN.LT.55)  GO TO 630
0461        LPC=LPC+1
0462        WRITE(LU,890)LPC,(RFILE(JK,1),JK=1,3)
0463        KLIN=1
0464  630   IF(LLINE(1).NE.1H*.AND.LLINE(1).NE.1H$) GO TO 620
0465  627   IF(LU.EQ.99) GO TO 640
0466        WRITE(LU,894)LINE,(RFILE(JK,J),JK=1,20)
0467  894   FORMAT(X,I5,"   ",20A2)
0468        GOTO 640
0469  620   CALL OHEX(KSOAD1,HEXL)
0470        CALL OHEX(IADDR,HEXA)
0471        IF(LU.EQ.99) GO TO 921
0472         IF(IDAT.EQ.1HD) GO TO 631
0473        WRITE(LU,895)LINE,HEXA(1),HEXA(2),HEXA(3),HEXA(4),
0474       *HEXL(1),HEXL(2),HEXL(3),HEXL(4)
0475       1,(RFILE(JK,J),JK=1,20),IM,
0476       2 KDA,KS1,KSO
0477  895   FORMAT(X,I5,"  ",4A1,"  ",4A1,"   ",21A2,I4,I4,I4,"_")
0478        IF(ERD.NE.0.OR.ERSO.NE.0.OR.ERS1.NE.0) GOTO 920
0479        IF(LWARN.NE.0)WRITE(LU,2910)LWARN
0480  2910  FORMAT(" SOURCE ONE WARNING(IM6 REQD) !",I3)
0481        IF(LU.EQ.6)WRITE(LU,6900)IADDR
0482  6900  FORMAT("   ",I5,"_ ")
0483        WRITE(LU,898)
0484  898   FORMAT(" ")
0485        GO TO 921
0486  920   IF(ERD.NE.0)WRITE(LU,691)ERD
0487  691   FORMAT(" DESTINATION ERROR ! ",I3)
0488        IF(ERS1.NE.0)WRITE(LU,692)ERS1
0489  692   FORMAT(" SOURCE ONE ERROR ! ",I3)
0490        IF(ERSO.NE.0)WRITE(LU,693)ERSO
0491  693   FORMAT(" SOURCE ZERO ERROR ! ",I3)
0492  921   CONTINUE
0493        KNTWD=KNTWD+1
0494        OFILE(IKNTP)=KNTWD
0495  922   IADDR=IADDR+1
0496  640   LINE=LINE+1
0497        KLIN=KLIN+1
0498        IF(KRES.GE.1000) GO TO 913
0499  1000  CONTINUE
0500  C    GO GET ANOTHER FILE OF SAME NAME
0501        GO TO 100
0502  891   FORMAT(X,I4,"  ERRORS TOTAL")
0503  695   CONTINUE
0504        IF(LU.EQ.99) GO TO 786
```

```
0505          WRITE(LU,2911)KWARN
0506  2911    FORMAT(X,I4," WARNINGS TOTAL")
0507          WRITE(LU,891)KER
0508  786     WRITE(1,891) KER
0509          WRITE(1,2911) KWARN
0510          WRITE (1,2712) IADDR
0511  2712    FORMAT (X,I5," WAS THE LAST ADDR USED")
0512          IF(LOBJD.NE.1HH) GO TO 906
0513  911     JK=4
0514  C JK POINTING AT FIRST COUNT IN OFILE
0515  912     KOSND=OFILE(JK)+2
0516          IF(KOSND.EQ.2) GO TO 905
0517          IF(KOSND.LT.2) STOP 77
0518          IF(KOSND.GT.1000) GO TO 913
0519  C NOW COPY OUT ONE FILE TO COFILE & SEND IT TO HARM
0520          J=1
0521          DO 910 JK=JK,KOSND+JK-1
0522          IF(J.GT.1000) GO TO 913
0523          COFIL(J)=OFILE(JK)
0524  910     J=J+1
0525  1075    CONTINUE
0526  C SEND ONE CONTIGUOUS SLAB TO HARMONIAC
0527          CALL EXEC(2,ICON4,COFIL,KOSND)
0528  C GO COPY OUT ANOTHER DATA FILE
0529          GO TO 912
0530  C
0531  C BEGIN DATA SECTION (FOR MAIN MEM 0 OR 1)
0532  C  OR NEW ORG - PMEM
0533  988     IERDAT =0
0534          IF(LLINE(2).NE.1HP) GO TO 2000
0535  C  START NEW PROGRAM ORIGIN - SEPARATED SEGMENT
0536          IDAT=0
0537  C OVERWRITE DEFAULT ORG.  IF NO PROGRAM HAS PRECEDED THIS
0538          IF(KRES.EQ.6)KRES=4
0539          CALL NUMB (LLINE,5,IADDR,IERDAT,0,8192)
0540          IADD=IOR(IADDR,070000B)
0541  C ADDR WITHIN P
0542          OFILE(KRES+1)=IADD
0543          IKNTP=KRES
0544          KRES=KRES+2
0545          KNTWD=0
0546          IF(LU.EQ.99) GO TO 2010
0547          WRITE(LU,2010)LINE,(RFILE(JK,J),JK=1,20)
0548  2010    FORMAT(X,I5,"  NEW PROGRAM SEGMENT BEGINS  ",20A2)
0549          IF(IERDAT.NE.0)WRITE(LI,987)IERDAT
0550          IF(IERDAT.NE.0)KER=KER+1
0551          GO TO 640
0552  2000    CONTINUE
0553          IF(LLINE(2).NE.1HM)IERDAT=1
0554          IDAT=1HD
0555          CALL  NUMB(LLINE,3,MEM,IERDAT,0,1)
0556          CALL NUMB(LLINE,5,IADDR,IERDAT,0,28672)
0557          IADD=IADDR
0558          IF(MEM.EQ.1)IADD=IOR(IADDR,100000B)
0559          KNTWD=0
0560  C   OVERWRITE DEFAULT ORG IF NO PROG/DATA HAS PRECEEDED THIS
```

```
        PAGE  0011  HASSY   FTN4 COMPILER: HP24177  (SEPT. 1974)

0561          IF(KRES.EQ.6) KRES=4
0562          IKNTP=KRES
0563          OFILE(KRES+1)=IADD
0564          KRES=KRES+2
0565          IF(LU.EQ.99) GO TO 640
0566          WRITE(LU,979)LINE,(RFILE(JK,J),JK=1,20)
0567  979     FORMAT(X,I5,"  DATA FILE FOR MAIN MEM BEGINS  ",20A2)
0568          GO TO 640
0569  C DOING DATA AREA
0570  981     CALL NUMB(LLINE,1,KSOAD1,IERDAT,-32768,32767)
0571          OFILE(KRES)=KSOAD1
0572          KRES=KRES+1
0573          IF(LU.EQ.99) GO TO 984
0574          GO TO 793
0575  631     WRITE(LU,982)LINE,HEXA(1),HEXA(2),HEXA(3),HEXA(4)
0576         1,HEXL(1),HEXL(2),HEXL(3),HEXL(4)
0577         2,(RFILE(JK,J),JK=1,20)
0578  982     FORMAT(X,I5," ",4A1," ",4A1," ",20A2,"_ ")
0579          IF(IERDAT.EQ.0) GO TO 985
0580          WRITE(LU,987)IERDAT
0581          KER=KER+1
0582  987     FORMAT(" DATA CODE/ADDR ERROR !",I5,"_@")
0583  985     WRITE(LU,984)
0584  984     FORMAT(" ")
0585          KNTWD=KNTWD+1
0586          OFILE(IKNTP)=KNTWD
0587          GO TO 922
0588  903     WRITE(LI,904)
0589  904     FORMAT(X,/," NO END MARKER (^) !")
0590          GO TO 905
0591  913     WRITE(LI,914)
0592  914     FORMAT(X,/," OBJECT FILE IS FULL !"
0593         2 )
0594          GOTO 905
0595  906     CALL EXEC(15,ICON,OFILE,1000,HOBJN,0)
0596  905     RETURN
0597          END
0598  $
**** LIST END ****
```

APPENDIX V - HARMONIAC SPECIFICATIONS

* Instruction time: 140 nanoseconds (150 nanoseconds in prototype due to clock jitter).

* Master clock period: 47 nanoseconds.

* Timing: Three phase.

* Method of operation: Simultaneous transfer of two operands from their sources to a destination which may be an arithmetic operation or memory, using twin tristate buses.

* Memories: Program - up to 1 K ROM, 1K RAM (32X 74 S 201)

  Data memories - two identical, up to 28K each (96x 93425)

  Table memory - one of 1 K ROM (1x 6068 sine)

  (Prototype Program memory is 512 words RAM, data memories are 3K words each, table is 1K x 10 sine).

* Memory addressing: automatic push or pop.

* Array Multiplier: Full 16 x 16 multiply, 225 nanosecond maximum, using 32 x 93 S 43 (2 x 4 bit).

* Arithmetic: ADD, AND, OR, SUB, compare using 4 x 74 S 181 (compare is =, $\neq$, >, <, $\geqslant$).

* Input, output: Direct memory access.

* Total DIP count 440.

* Power consumption: 5V @ 27A Typical (135 W).

* Construction: Wire wrap socket array 45 cm x 40 cm approx.