Microprogramming 21MX Computers

operating and reference manual

HEWLETT hp PACKARD

HEWLETT-PACKARD COMPANY
11000 WOLFE ROAD, CUPERTINO, CALIFORNIA, 95014

MANUAL PART NO. 02108-90008
MICROFICHE PART NO. 02108-90009

Printed: AUG 1974
Printed in U.S.A.

LIST OF EFFECTIVE PAGES

Pages	Effective Date
Title	August 1974
ii to vii	August 1974
1-1 to 1-2	August 1974
2-1 to 2-4	August 1974
3-1 to 3-24	August 1974
4-1 to 4-26	August 1974
5-1 to 5-17	August 1974
6-1 to 6-7	August 1974
A-1 to A-3	August 1974
B-1	August 1974
C-1 to C-2	August 1974
D-1 to D-4	August 1974
E-1 to E-19	August 1974
I-1 to I-4	August 1974

HP Computer Museum
www.hpmuseum.net

For research and education purposes only.

PREFACE

This manual is a complete reference source for microprogramming the Hewlett-Packard 21MX Computer Series. With the facilities of the HP 12978A Writable Control Store the user can expand the already powerful capability of his 21MX Series Computer by adding custom-tailored instructions to the existing set of microprogrammed basic instructions.

The HP 12978A Writable Control Store is provided with two options. The 12978A option 001 provides software that operates in the DOS-III operating system. The 12978A option 002 provides software that operates in the Basic Control System. Refer to Section VI of this manual for a complete description of the options.

This manual is written for an individual who already has considerable experience as an assembly language programmer. HP 21MX Computer Series microprogramming is no more complex than normal assembly language programming on larger computers. Thus, with little more investment that learning a new assembly language, large computer capability can be had for small computer expense.

RELATED DOCUMENTATION

It is assumed that the microprogrammer has read the **HP 21MX Computer Series Reference Manual (HP 02108-90002)** and that he knows how to use his operating system, DOS-III (HP 24307B), or the Basic Control System (HP 20855A). These operating systems are described in the following publications:

HP 24307B DOS-III Disc Operating System (HP 24307-90006)

Basic Control System (HP 02116-9017)

During the process of writing, debugging, and using a microprogram, the user should also have access to and be familiar with the following additional publications.

The assembler used with the DOS-III-B system is described in:

Assembler Reference Manual (HP 24307-90014)

The assembler used with the Basic Control System is described in:

HP Assembler (HP 02116-2014)

The 21MX computer is described in:

HP 21MX Computer Series Operator's Manual (HP 02108-90004)

HP 21MX Computer Series Installation and Service Manual (HP 02108-90006)

The HP 12909B pROM Writer, which is used in conjunction with the six mask tapes produced by the Micro Debug Editor, and installing pROMs is described in:

HP 12909B pROM Writer Operating and Reference Manual (HP 12909-90009)

HP 12909B Programmable ROM Writer Interface Kit Installation and Service Manual (HP 12909-90005)

HOW TO USE THIS MANUAL

This manual is intended to be used in the following way:

a. Read Section I for the introduction to user microprogramming.

b. Study Section II to learn the structure of the system that is being controlled by microprogramming. Section II explains the relationship between the Control Section and the other sections of the computer.

c. Become familiar with the reference material in Sections IV, V, and VI so that when the time comes to use the material, it may be found easily. These sections describe the microprogramming language, the Micro-assembler, the Micro Debug Editor, and the 12978A Writable Control Store.

d. Study Section III to learn how to write a microprogram.

CONTENTS

CONTENTS (continued)

ILLUSTRATIONS

ILLUSTRATIONS (continued)

TABLES

INTRODUCTION TO USER MICROPROGRAMMING

SECTION I

The Control Section of a computer contains circuitry which decodes each machine instruction and then executes the required sequence of operations. Machine instructions can be decoded and executed by either a conventional Control Section or a microprogrammed Control Section.

1-1. CONVENTIONAL CONTROL SECTION

In a conventional computer Control Section, specific hardware is dedicated to each function performed by the instruction set. The major advantage of this specially designed hardware is speed for the instruction set. The major disadvantage is the loss of flexibility for special applications or for enhancements. Changes and additions to hardware components are required to implement changes and additions to existing capabilities.

This is no problem for a conventional computer if no new machine instructions are required. The hardware has been designed to minimize timing for the instruction set. Rarely however, does a computer manufacturer produce an instruction set that fully meets the requirements of most potential users. Hence, the manufacturer must either focus his attention on one group of users (specialize) or widen his scope and generalize the hardware design to meet the needs of a number of user groups. In the latter case, the user must modify his discipline to some extent to meet the limitations of his hardware.

1-2. MICROPROGRAMMED CONTROL SECTION

In the microprogrammed computer, all distinct logical functions are separated from the sequence in which those functions are performed. Hardware redundancy is thus reduced. The logical functions are defined by a bit pattern or micro-instruction held in Control Store. Each machine instruction in Main Memory is performed by a sequence of micro-instructions in Control Store that defines the logical functions to be performed. This sequence of micro-instructions is called a microprogram and is often referred to as firmware, because it lies somewhere between hardware and software in origin and permanence.

Software can execute much faster with the application of microprogramming. This speed is achieved by two factors: the ratio of Control Store speed over Main Memory and the relative flexibility of a micro-instruction over normal machine instructions. The HP 21MX Control Store, where micro-instructions reside, cycles more than twice as fast as Main Memory, where normal machine instructions reside. Control Store words are 24 bits whereas Main Memory words are 16 bits. In addition, micro-instructions have access to many internal registers and logical functions that Main Memory programs cannot use.

For example, the 21MX floating point software subroutines were identified as being very time consuming. They were then microcoded by a Hewlett-Packard microprogrammer and made available in Read Only Memory to users. Implementation of the floating point firmware requires no change to user programs. The microprogrammed floating point instructions run about 20 times faster than the corresponding software subroutines.

As in the floating point microprogram, the user can study his software, determine the most time consuming functions performed, and then microprogram those functions, that is, execute them in Control Store using a single Main Memory instruction instead of a sequence of Main Memory instructions. Any software that uses those microprogrammed functions will execute at a higher speed.

1-3. LIMITATIONS OF HP 21MX MICROPROGRAMMING

The user should be aware of the following limitations imposed by HP 21MX microprogramming:

a. Since the origin of a microprogram is specified during micro-assembly, HP 21MX microprograms are not relocatable.

b. Since there is only one register available to the microprogrammer to save subroutine return addresses, the HP 21MX design allows for no more than one logical microprogram subroutine level. This limitation can be circumvented by using other registers or Main Memory to simulate subroutine nesting.

c. The microprocessor cannot be interrupted. If the microprogram execution time exceeds the interval between interrupts (75 μs. is the maximum interval

allowed by Hewlett-Packard instruction set microprograms), the microprogram must test for pending interrupts or they can be lost. When a pending interrupt is detected, the microprogram must yield control to the interrupt handler. For a discussion of microprogram interrupt handling, refer to sections 3-32 and 3-33 in this manual.

1-4. SUMMARY

The advantages of microprogrammed control are:

a. The user can use a fully-supported general purpose computer to aid in the generation and debugging of extensions to the computer's own instruction set.

b. The user can speed up the overall execution time of his software by microprogramming its most time consuming or repetitious routines.

c. The user can implement enhancements of the instruction set and special purpose processors produced by the manufacturer with little impact on his existing software.

THE MICROPROGRAMABLE COMPUTER

SECTION II

To successfully implement microprograms, the assembly language programmer must learn more about the computer. This section of the manual is the introduction to the structure of the computer. A functional block diagram of the microprogrammable machine is provided in Appendix D. This diagram describes what paths data can follow. Control commands or micro-instructions spell out what paths the data does follow and what modifications and tests are performed in the process.

Functionally, a computer consists of four major sections:

- Control
- Main Memory
- Input and Output
- Arithmetic and Logic

and data are stored in the Main Memory. Parameters, status, commands, and processor results (data) are exchanged with external devices such as teleprinters, magnetic tape units, and line printers via the Input and Output (I/O) section. Add, subtract, and other mathematical functions and shift, "or", "and", and other logical functions are performed in the Arithmetic and Logic section. The Front Panel registers and switches provide direct operator communication.

Each section executes under the direction of the Control Section by means of a microprogram. The Control Section reads the user's program stored in Main Memory and directs the appropriate hardware in each of the other sections.

Figure 2-1 shows the four major sections of the computer.

2-1. RELATIONSHIP BETWEEN SECTIONS

These four sections and the Front Panel are interconnected by a network of signal paths. Data processing programs

2-2. CONTROL SECTION

To write a microprogram an understanding of the Control Section is required. The Control Section takes an instruction from Main Memory and stores it into the Instruction

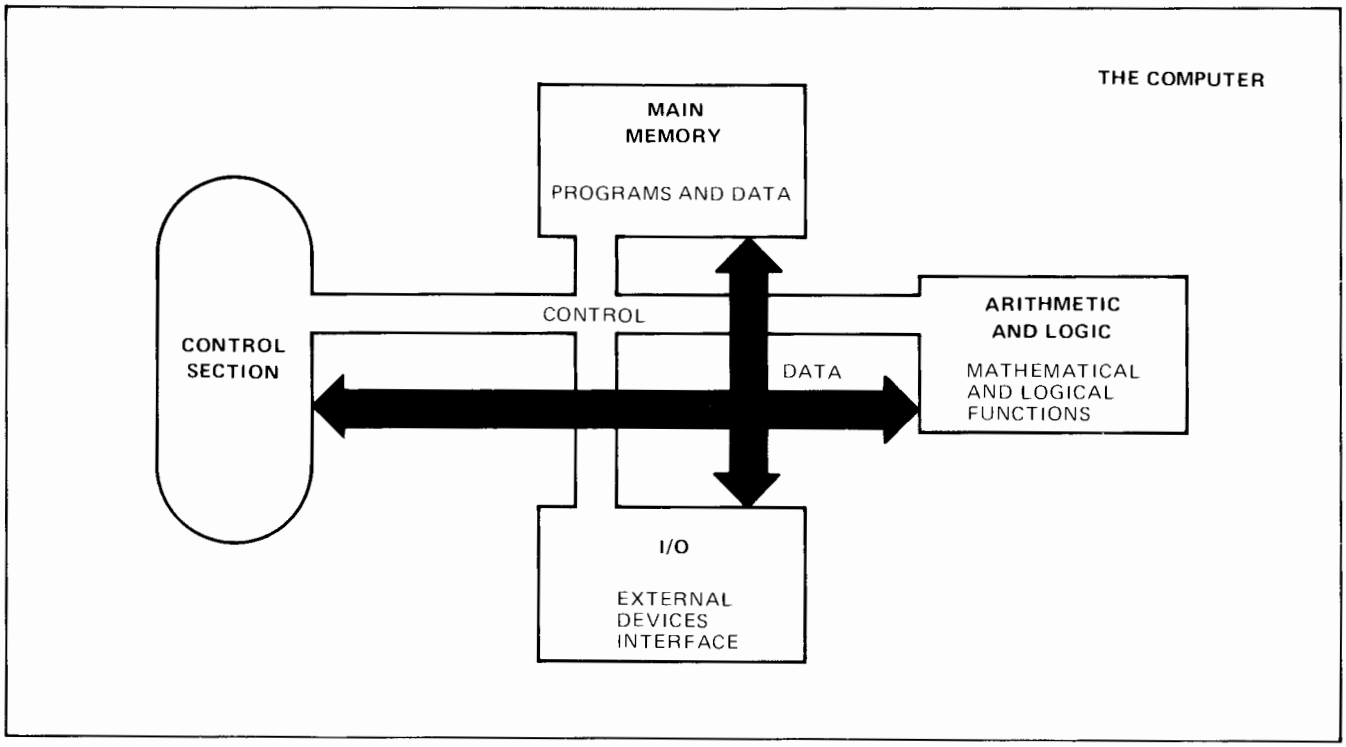


Figure 2-1. Four Major Computer Sections

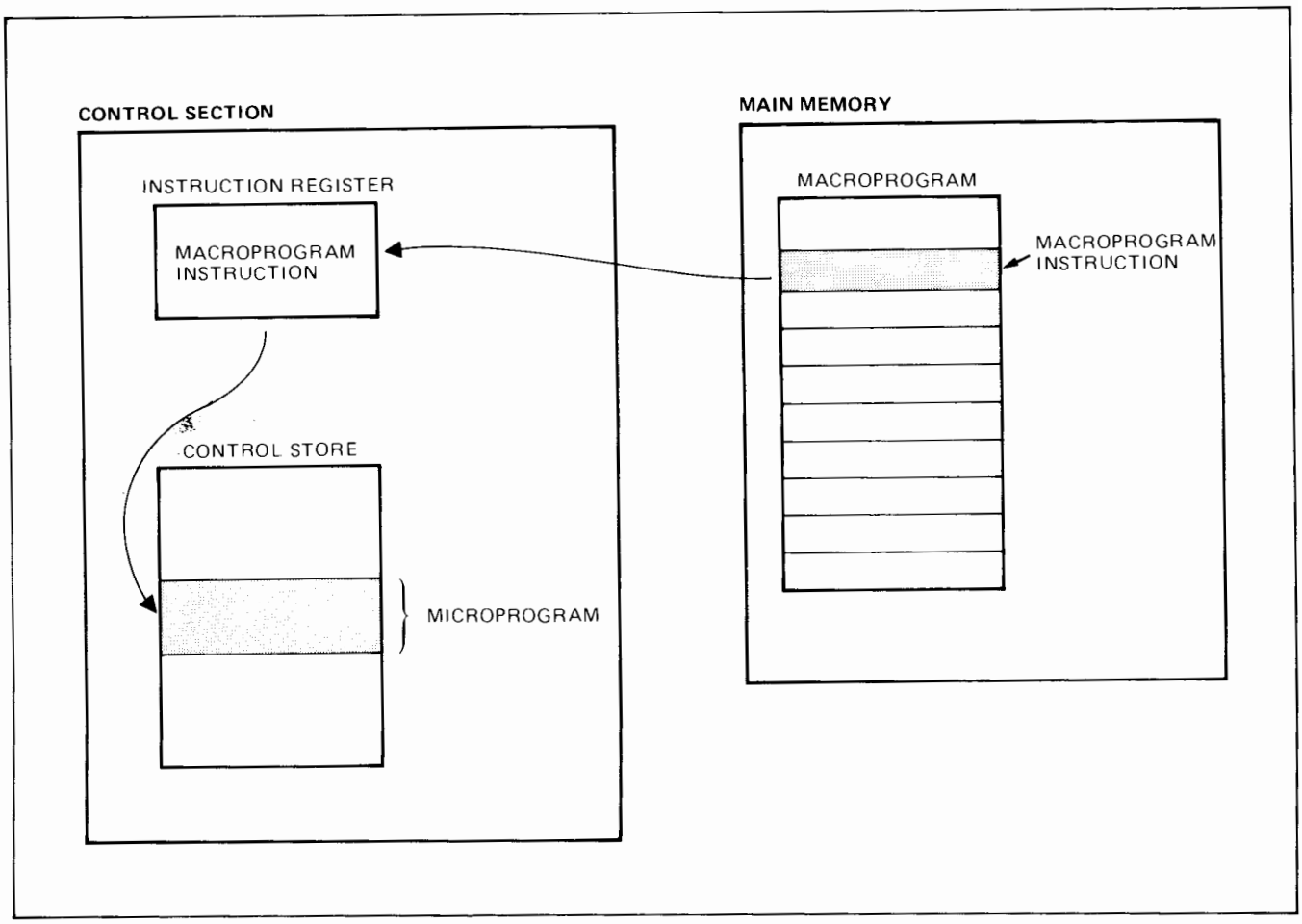


Figure 2-2. A Microprogram Implements **One** Macroprogram Instruction

Register (IR), as shown in figure 2-2. An appropriate microprogram is executed whose Control Store entry point address is determined by the IR. View, then, each program instruction in Main Memory as a jump to a microprogrammed routine, which resides in Control Store.

The storage area for these microprograms is Control Store which may be either a Read Only Memory (ROM) or Writable Control Store (WCS). In this manual, to distinguish programs in Main Memory from microprograms in ROM, Main Memory programs are called macroprograms. We refer to a Control Section that executes microprograms from ROM, as a Control Processor.

2-3. THE CONTROL PROCESSOR

A microprogram in the Control Processor is in command of the computer at all times. A microprogram which is part of the basic 21MX instruction set microprogram takes program instructions from Main Memory and stores them into the Instruction Register. The upper eight bits of the Instruction Register determine the microprogram address within one of the following instruction groups:

Basic Instruction Set

Extended Instruction Group

Floating Point Instruction Group

User Microprogram Group

Since the user is mainly interested in writing and executing his own microprograms, he can regard the Basic Instruction Set microprogram as a supervisor microprogram that determines when a user microprogram is called and then passes control to the user microprogram.

When the Instruction Register holds an octal 101rrr or 105rrr (see table 3-1 for possible values of rrr), a branch is made to the user microprogram area of Control Store.

When a microprogram has run to completion, it returns to location 0 in Control Store to take the next instruction from Main Memory and store it into the Instruction Register.

2-4. THE MICROPROGRAMMER'S ROADMAP

Appendix D holds the fundamental diagram of the computer required by the microprogrammer. This functional

block diagram is the "roadmap" that is used to determine possible data paths and to determine where logical decisions can be made. This diagram can be unfolded and referred to while reading other parts of the manual. Note that the four sections of the computer, illustrated in figure 2-1, are shown in more detail in the functional block diagram.

To read the functional block diagram, begin with a 101rrr or 105rrr instruction in the Instruction Register. The rrr specifies the octal Control Store entry point address according to the description in section 3-24, Mapping to a Module Address. This address is moved into the ROM Address Register (RAR). With a first address specified, the user microprogram begins execution. The contents of the Control Store location given in the ROM Address Register are moved into the ROM Instruction Register (RIR). The ROM Instruction Register now holds a 24 bit micro-instruction. The micro-instruction is decoded and the specified control functions are executed.

Successive micro-instruction addresses are determined in the following way. The ROM Address Register is incremented at the start of execution of each micro-instruction. When a jump is executed, the ROM Address Register is loaded with the jump target address. When a jump to subroutine is executed, the ROM Address Register is stored into the SAVE Register (save return address) and the jump target address is stored into the ROM Address Register. When a return from subroutine is executed (RTN), the SAVE Register contents are transferred into the ROM Address Register and the SAVE Register is cleared. Thus at the completion of execution of each micro-instruction, the ROM Address Register holds the address of the next micro-instruction.

2-5. DATA PATHS

The central data transfer path is the S-bus. The contents of all regesters except the following can be directed onto the S-bus: L-register, RAR, SAVE Register, Extend Register, and the Overflow Register. The following registers can receive data from the S-bus:

M-register
T-register
L-register
Counter Register
Display Register
Display Indicator
Instruction Register

The T-bus receives data only from the Rotate/Shifter (R/S) but can pass data to these registers:

A-register
B-register
Scratch Pad Registers (S1 through S12)
X-register
Y-register
P-register
S-register (Front Panel Switch Register)

The I/O-bus serves to transfer data to and from external devices under programmed control.

Note in Appendix D, the functional block diagram, that the arrows are significant. For example, the B-register contents can be sent to the S-bus and thence to the M-register. However, the contents of the B-register cannot be sent to S12 (Scratch Pad 12) without passing through the ALU.

2-6. MAIN MEMORY

The M-register is a 15 bit register which holds memory addresses for reading from or writing into Main Memory. When storing from the M-register, bit 15 is clear (0). The T-register or Transfer register holds the data being transferred to or from memory. Contents of both these registers are transferred to and from the S-bus. Four loader ROMs, selectable by Instruction Register bits 15 and 14, each can contain a 64 word Main Memory program which may be loaded into Main Memory and used to load Main Memory from a peripheral device or to perform any other function desired by the user.

Two flags are associated with memory: the A-register Addressable Flag (AAF) and the B-register Addressable Flag (BAF). These flags are required to allow the A- and B-registers to be addressed as locations 0 and 1, respectively, of Main Memory.

2-7. I/O SECTION

The Central Interrupt Register (CIR) is a 6 bit register associated with the I/O interrupt circuitry. It is loaded with the Select Code of the interrupting device under program control and passed to the S-bus. Whenever the Central Interrupt Register is loaded, an Interrupt Acknowledge (IAK) signal is issued to the I/O device.

The I/O-bus transfers data to and from external devices.

Two flags are associated with I/O: the Interrupt Pending flag and the I/O Skip Condition Met (Main Memory instructions SFS and SFC) flag.

The Interrupt Enable Register is used to disable or enable the recognition of all interrupts, except Memory Protect, Parity, and Power Fail interrupts.

2-8. ARITHMETIC AND LOGIC SECTION

This section consists of the Arithmetic and Logic Unit (ALU), the Rotate/Shifter (R/S), 22 registers and six flags.

The ALU and R/S are the only units that execute functional modifications on the data. The ALU receives input from the S-bus and from the L-register (Latch Register). Output from the ALU goes to the R/S which places its output on the T-bus.

Output from the ALU and R/S can be stored in one of the following registers via the T-bus:

A-register
B-register
Scratch Pad Registers (S1 through S12)
X-register
Y-register
P-register
S-register

Remember that the P-register holds the macroprogram (Main Memory) address. The P-register must be under control of the microprogram which must insure that it contains the proper address after the microprogram is complete. When the microprogram is complete, the resulting P-register value is the address of the next macro-instruction to be executed. Note that the Basic Instruction Set fetch routine (at Control Store address 0) automatically increments the P-register after the macro-instruction is fetched. Thus for one word user macro-instruction function codes, no further incrementing of the P-register is necessary in the user microprogram.

The S-register is reserved for internal storage of the Front Panel switch register. Note that all of these registers can also be sent along the S-bus for storage into memory, passage to an external device, or input to the ALU.

The Extend Register is a one bit register used in shift operations to link the A- and B-register or to indicate a "carry" arithmetic result out of the A- or B-registers. The Overflow is a one-bit register used to indicate an arithmetic overflow from the ALU. (See **21MX Computer Series Reference Manual**, where Overflow and Extend Register arithmetic results are fully explained.) These two registers can also be used as flags.

The 8 bit Counter Register, which passes to and from the S-bus, is used for repeat instructions, for Loader ROM addressing, and other general purposes, such as looping in a microprogram.

There are six flags dedicated to the Arithmetic and Logic Section. The CPU Flag is a general purpose flag. Four others signal output results from the ALU and one indicates the last T-bus value. ALU Ones is set when all ones are output from the ALU. ALU Carry Out is set when an ALU function produces a "carry" out of bit 15. ALU Bit 0 and ALU Bit 15 flags represent the last value of the specified bit in the ALU output. T-bus Zero flag is set if all bits of the T-bus are zero.

2-9. FRONT PANEL

Two registers and two flags are associated with the Front Panel Section. The Display Register holds the contents of the register A, B, M, T, P, or S, indicated by the Display Indicator. The Display Register and the Display Indicator are displayed on the Front Panel, as illustrated in figure 2-3.

The Run Mode flag indicates that the computer is in a Run or Halt condition. The Run Enable flag indicates whether the four position key-operated switch on the front panel is in Lock or Operate mode.

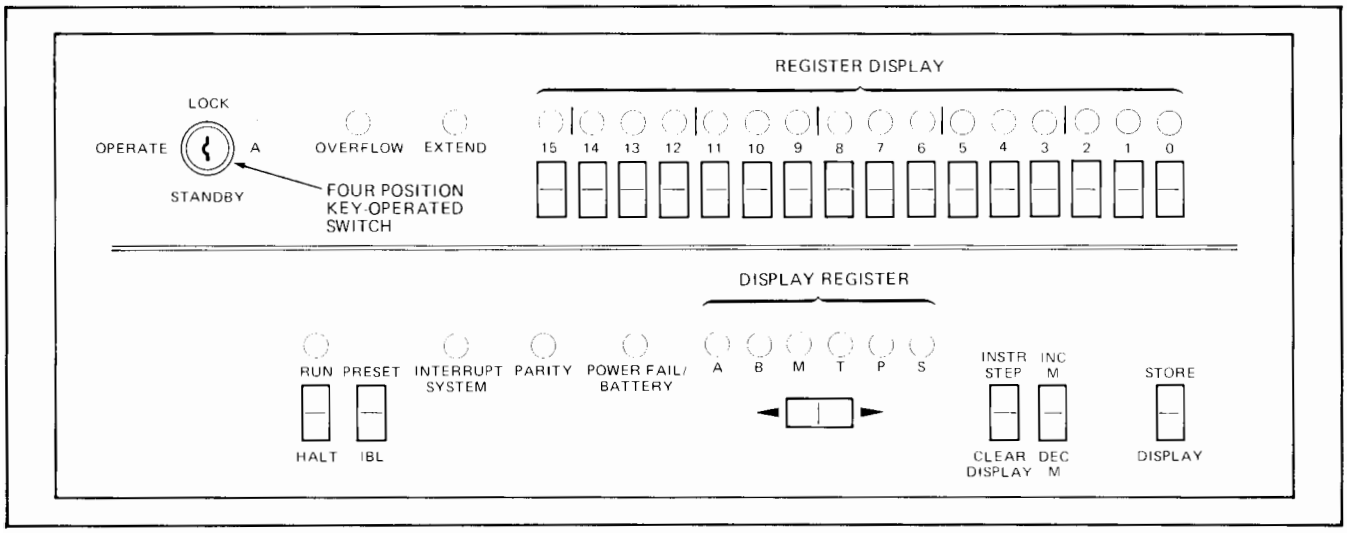


Figure 2-3. Front Panel Displays and Switches

WRITING A MICROPROGRAM

SECTION III

This section introduces the basics of writing and debugging a microprogram in the micro-assembly language.

An assembly language programmer who codes programs for Main Memory may shun microprogramming because he regards it as too complex, mysterious, and the exclusive field of the computer designer.

However, Hewlett-Packard has especially designed the HP 21MX series computers to enable assembly language programmers to quickly get to the microcode level of computer logic so that they can attack the most time-consuming and least efficient parts of the software. Execution times can be cut with the proper application of microcode.

3-1. AN EXAMPLE

Figure 3-1 illustrates a segment of a microprogram. Ten micro-instructions are shown coded on the 21MX Micro-coding Form. The second micro-instruction shaded in figure 3-1 consists of the following four codes:

```
COV   PASS M   P
```

Each of the four codes are called micro-orders:

a. **P** takes the 16 bits in the P-register and puts them onto the S-bus.

b. **M** stores the 16 bits on the S-bus into the M-register (bit 15 of M-register is always 0).

c. **PASS** passes the 16 bits on the S-bus through the ALU without modification.

d. **COV** clears the Overflow Register.

Note in figure 3-1 that the various micro-orders of the micro-instruction begin in certain columns of the micro-coding form. These columns define the location of fields of the micro-instruction and each field holds a certain type of micro-order. In the case of the example micro-instruction, field 3 holds the special operation COV, field 4 holds the ALU operation PASS, field 5 holds the store operation M, and field 6 holds the data source P, that is, the data placed on the S-bus.

Section IV of this manual gives a full explanation of micro-instruction formats and micro-orders.

3-2. COMPARISON BETWEEN ASSEMBLY AND MICRO-ASSEMBLY LANGUAGE PROGRAMMING

The assembly language programmer is already familiar with the basic concepts of programming: the instruction, data source, data destination, data modification, data test, and branch. These concepts hold in microprogramming.

3-3. THE INSTRUCTION

The normal macro-instruction in Main Memory is 16 bits long. Most macro-instructions consist of one operation command (for example Add to A-register) and a data source or destination (for example Memory Location 1237). Thus there are usually two orders in a macro-instruction [Add to A-register] [Memory location 1237]. This is coded in Assembly Language as ADA VALU, where VALU is the label of memory location 1237.

The micro-instruction in Control Store is 24 bits long, which allows more control and flexibility to be coded into each instruction. A micro-instruction consists of up to five orders called micro-orders. Section 3-1 gives an example of four micro-orders coded into a micro-instruction.

There are four micro-instruction formats. Each format defines a micro-instruction Word Type (Word Type 1, Word Type 2, etc.) and determines a set of micro-orders which may be coded into the format. Micro-instruction Word Types and micro-orders are described in Section IV.

3-4. DATA SOURCE AND DATA DESTINATION

Both assembly and micro-assembly language instructions specify data source and data destination. In assembly language one of these is usually a Main Memory address and the other is a register, as in ADA VALU where the A-register is the destination of the data and VALU is the source of the data. With microprogramming both data source and data destination are usually registers, as more registers are available to the microprogram than to the assembly language program.

3-5. DATA MODIFICATION

Add, shift, set flag, and logical functions are performed similarly in both types of programming. In microprogramming, a wider range of basic operations, especially logical functions, is available. Complex operations, such as divide, multiply, and byte move, are performed by microprogrammed subroutines and are available in the Basic Instruction Set and Extended Instruction Group microprograms.

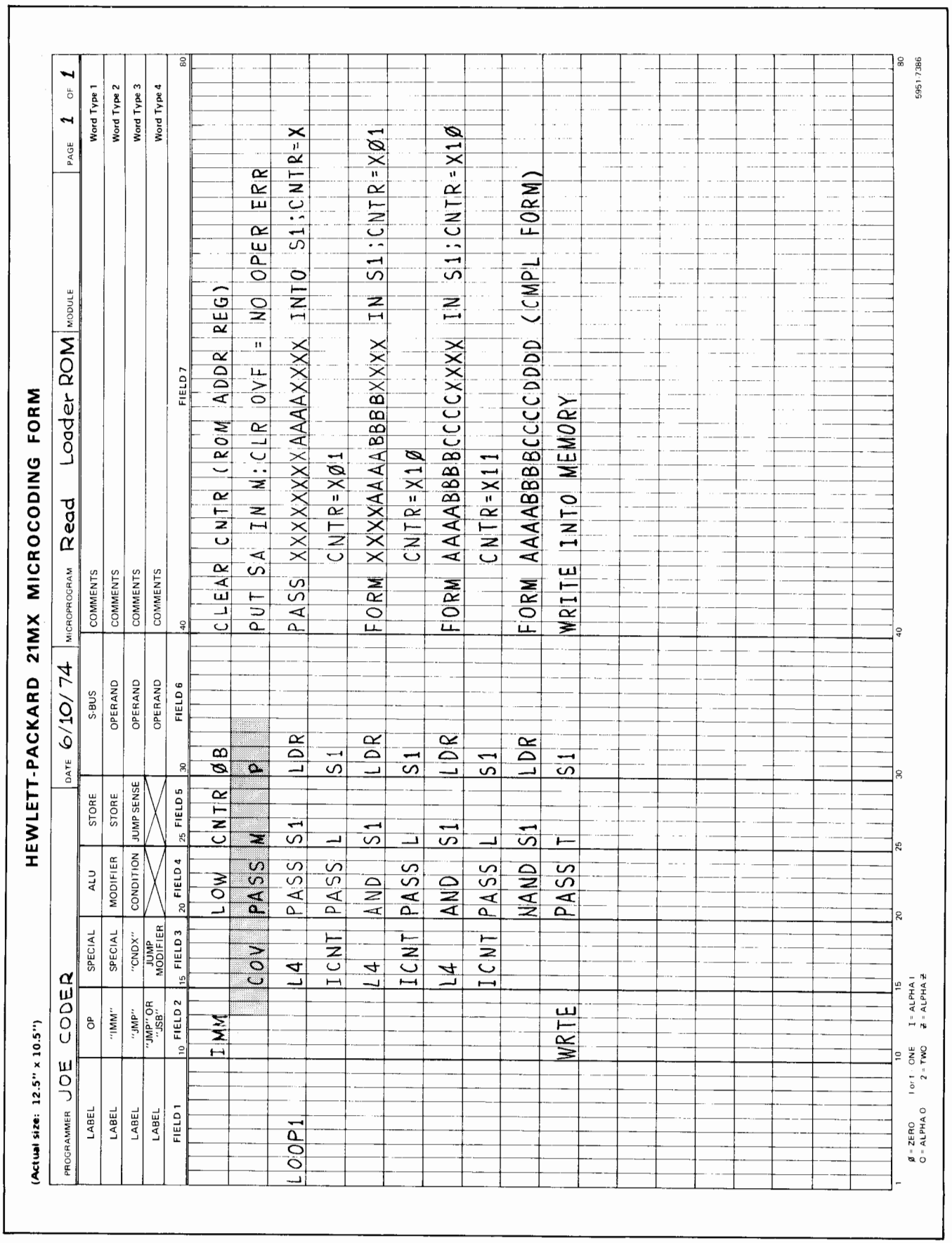


Figure 3-1. Microprogram Segment On the 21MX Microcoding Form

3-6. DATA TEST AND BRANCH

These operations are quite similar in the two languages. Many tests occur automatically in the course of transferring data in a microprogram. A test and branch out of a line of macro-instructions in normal assembly language, however, requires two instructions (4.6 μs): a test instruction and a skip instruction.

For example:

SLA	skip if LSB of A=0
JMP OUT	branch out of code sequence

A test and branch out of a line of micro-instructions requires only one micro-instruction (.325 μs).

For example:

JMP CNDX AL0 OUT	branch out of code sequence if bit 0 of last ALU output = 0

3-7. MICRO-INSTRUCTION FORMATS

Just as in normal assembly language coding, micro-assembly language source statements are coded in mnemonic form to define an instruction. Each source language statement defines a micro-instruction and consists of an optional label, five micro-order fields some of which may be left blank, and a comment field. The label is used when needed as a reference by other micro-instruction statements. The micro-orders consist of one to four mnemonic characters and specify functions to be performed by the Control Section. According to the type of micro-instruction being defined, one of the micro-orders is sometimes interpreted as an operand. When an operand is specified, it defines an integer or an address, depending on the type of micro-instruction being defined.

3-8. STATEMENT CHARACTERISTICS

Micro-assembly language source statements are divided into four formats, according to the function the micro-instruction is to perform. Each format is called a Word Type.

- Word Type 1 is the most commonly used micro-instruction format and specifies data transfer and modification. Word Type 1 source statement fields are:

  Label
  Op
  Special
  ALU
  Store
  S-bus
  Comments

- Word Type 2 is used to send an 8 bit constant (immediate data) specified in the micro-instruction to a register. Word Type 2 source statement fields are:

  Label
  "IMM"
  Special
  Modifier
  Store
  Operand
  Comments

- Word Type 3 is used to specify a conditional branch in the microprogram. Word Type 3 source statements fields are:

  Label
  "JMP"
  "CNDX"
  Condition
  Jump Sense
  Operand
  Comments

- Word Type 4 is used to specify an unconditional branch in the microprogram. Word Type 4 source statement fields are:

  Label
  "JMP" or "JSB"
  Jump Modifier
  Operand
  Comments

3-9. FIELDS

As shown in figure 3-1, the fields are fixed for micro-assembly language source statements. An entry in any field (except comments) must begin in the first column of that field.

- Field 1 begins in column 1 and holds a label that is no longer than eight characters.
- Field 2 begins in column 10 and contains a micro-order no longer than four characters. This field can also hold a Pseudo Instruction (refer to section 4-21 for the explanation of Pseudo Instruction mnemonic codes).
- Field 3 begins in column 15 and contains a micro-order no longer than four characters.
- Field 4 begins in column 20 and contains a micro-order no longer than four characters.

- Field 5 begins in column 25 and contains a micro-order no longer than four characters.
- Field 6 begins in column 30 and contains a micro-order no longer than four characters (Word Type 1) or an operand (Word Types 2, 3, and 4).
- Field 7 begins in column 40 and contains comments only; comments may begin and be placed anywhere from column 40 to column 80 (if column 39 contains the last character of the field 6 operand, field 7 must begin in column 41).

3-10. CHARACTER SET

The characters that may appear in a source statement are as follows:

A through Z
0 through 9
. (period)
* (asterisk)
+ (plus)
- (minus)
(space)

Any ASCII character may appear in the comments field.

A space may only begin a field if no micro-order is specified in that field.

3-11. LABEL SYMBOL

A label may be one to eight characters consisting of A through Z, 0 through 9, and a period. The first character must be a letter.

Each label must be unique within the microprogram. Names which appear in $EXTERNALS micro-assembler control input statements (refer to section 5-5) may not be used as statement labels in the same microprogram.

3-12. ASTERISK COMMENT

An asterisk in column one of the source statement indicates that the entire micro-assembler source statement is a comment.

3-13. MICRO-ORDERS: FIELDS 2 THROUGH 6

The micro-order fields define operations that are to be performed by the Control Section of the computer. The micro-orders applicable to each field are determined by the source statement Word Type. Section IV describes the micro-orders that apply to each Word Type and describes the operations that they specify.

3-14. OPERANDS IN FIELD 6

Word Types 2, 3, and 4 contain an operand in field 6.

In Word Type 2, the operand must be either a decimal or octal number. It cannot be an expression (refer to section 4-10 for definition of a Word type 2 operand).

In Word Types 3 and 4, the operand is a decimal number, octal number, or a number computed from an expression which can include a label (refer to section 4-16 for the definition of a Word Type 3 operand. Refer to section 4-20 for the definition of a Word Type 4 operand).

3-15. CODING THE FOUR WORD TYPES

The following sections describe how to code source statements in micro-assembly language. The reader should be familiar with Section IV of this manual before proceeding with these descriptions. Section IV describes the micro-orders that can be used with each Word Type. By referring to Section IV, the reader can see the options that are available to him as each Word Type is described. The reader will also need to refer to the functional block diagram in Appendix D.

3-16. CODING WITH WORD TYPE 1 — COMMON

This word type specifies data transfer and modification. The format of Word Type 1 is shown in section 4-1. As an example, a micro-instruction is developed that executes the following control functions:

- Store the A-register contents into the M-register
- Perform a memory protect check on the A-register contents
- Transfer the A-register contents to the ALU, increment this value in the ALU, and store the result into the P-register

a. Specify the register that is to be placed on the S-bus; the A-register is specified in the example:

OP	SPEC	ALU	STORE	S-BUS
				A

b. Specify the function of the ALU; the increment function is specified in the example:

OP	SPEC	ALU	STORE	S-BUS
		INC		A

c. Specify the Op field function; no Op field function is specified in the example. When no Op function is required, the standard operation is specified by either leaving the field blank or inserting NOP into the field:

OP	SPEC	ALU	STORE	S-BUS
NOP		INC		A

d. Specify a Special function, if required; a memory protect check is specified in the example:

OP	SPEC	ALU	STORE	S-BUS
NOP	MPCK	INC		A

e. Finally, specify where the resulting data is to be stored. Two store operations are required in the example. The unmodified A-register value on the S-bus must be stored into the M-register and the incremented A-register value on the T-bus must be stored into the P-register. The micro-order PNM performs both of these store operations and serves to illustrate that data stored from the S-bus is unmodified data and data stored from the T-bus can be modified by the ALU or R/S:

OP	SPEC	ALU	STORE	S-BUS
NOP	MPCK	INC	PNM	A

PNM is a unique micro-order. No other micro-order provides the ability to store into two registers in the same micro-instruction.

3-17. CODING WITH WORD TYPE 2 — IMMEDIATE DATA

This word type sends an 8 bit constant (immediate data) specified in the micro-instruction to a register. The format of Word Type 2 is shown in section 4-7. As an example, a micro-instruction is developed that specifies the following control function:

- Repeat the micro-instruction following this one ten times

a. Specify IMM in the Op Code field:

"IMM"	SPEC	MODIF	STORE	OPERAND
IMM				

b. Specify the octal or decimal data to be placed on the S-bus; a decimal -10 is specified in the example:

"IMM"	SPEC	MODIF	STORE	OPERAND
IMM				-10

If an octal -10 is specified, it is written -10B.

c. Specify one of the four possible data modifiers (refer to section 4-9); LOW (place the 8 bit operand in the lower half of the S-bus and ones in the upper half) is specified in the example:

"IMM"	SPEC	MODIF	STORE	OPERAND
IMM		LOW		-10

d. Specify where the resulting data is to be stored; the Counter Register is specified in the example:

"IMM"	SPEC	MODIF	STORE	OPERAND
IMM		LOW	CNTR	-10

e. Specify any special operations required; RPT (repeat the micro-instruction following this one the number of times specified in the Counter Register) is specified in the example:

"IMM"	SPEC	MODIF	STORE	OPERAND
IMM	RPT	LOW	CNTR	-10

3-18. CODING WITH WORD TYPE 3 — CONDITIONAL JUMP

This word type specifies a conditional branch in the microprogram. The format of Word Type 3 is shown in section 4-11. As an example, a micro-instruction is developed that specifies the following control function:

- Jump to the microprogram address labeled ERR2, if the last data on the T-bus was not zero.

a. Specify JMP and CNDX in the Op Code and Special fields:

"JMP"	"CNDX"	COND	JUMP	SENSE	OPERAND
JMP	CNDX				

b. Specify the condition that must be tested for the jump to take place; T-bus equal to 0 is specified in the example:

"JMP"	"CNDX"	COND	JUMP	SENSE	OPERAND
JMP	CNDX	TBZ			

c. Specify, if required, RJS (Reverse Jump Sense), which establishes whether the Condition code "true" means jump or "false" means jump. The TBZ used in the example means the test condition is T-bus equal to 0. If RJS is specified, T-bus not equal to 0 means perform the jump. If RJS is not specified (blank in the field), then T-bus equal to 0 means jump. RJS is specified in the example:

"JMP"	"CNDX"	COND	JUMP	SENSE	OPERAND
JMP	CNDX	TBZ	RJS		

d. Specify the target address of the jump. The target address must have the same most significant three bits as the address of this micro-instruction. The address label ERR2 (an address label in the current page) is specified in the example:

"JMP"	"CNDX"	COND	JUMP	SENSE	OPERAND
JMP	CNDX	TBZ	RJS		ERR2

3-19. CODING WITH WORD TYPE 4 — UNCONDITIONAL JUMP

This word type specifies an unconditional branch in the microprogram. The format of Word Type 4 is shown in section 4-17. As an example, a micro-instruction is developed that specifies the following control function:

- Jump to a microprogram subroutine whose address is derived by the following: the address labeled CLSUB supplies all bits of the subroutine address except bits 3-0; bits 3-0 are supplied by the Instruction Register.

a. Specify JSB in the Op code field:

"JMP" OR "JSB"	JUMP MODIFIER	--	--	OPERAND
JSB				

b. Specify a target address (to be modified) of the jump anywhere within the Control Store (0-7777); CLSUB is specified in the example:

"JMP" OR "JSB"	JUMP MODIFIER	--	--	OPERAND
JSB				CLSUB

c. Specify any modification to the target address; J30 (replace bits 3 to 0 of the operand with bits 3 to 0 of the Instruction Register) is specified in the example:

"JMP" OR "JSB"	JUMP MODIFIER	--	--	OPERAND
JSB	J30			CLSUB

3-20. FROM CODE TO EXECUTION SUMMARY

Figure 3-2 helps to illustrate the process of implementing a microprogram. Writing a micro-assembly language program is essentially the same process as writing an assembly language program. Micro-instructions are combined to form a microprogram. The microprogram is punched onto cards or paper tape and this source is read by the Micro-assembler. The Micro-assembler produces a listing and an object tape.

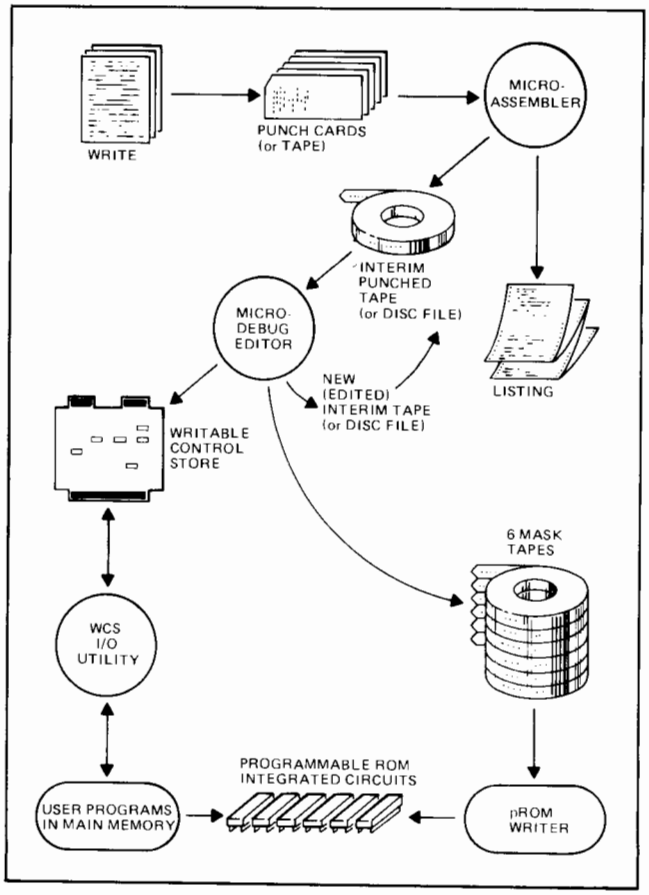


Figure 3-2. Microprogram Implementation Process

The object tape is loaded into Writable Control Store (WCS), executed, and debugged interactively using the Micro Debug Editor (MDE). When the microprogram is debugged, the source is corrected and the microprogram is reassembled. The microprogram can be loaded in two ways. It can be loaded into WCS by a call to the WCS I/O Utility subroutine from the user's Main Memory program or it can be burned into a programmable Read Only Memory. In the latter case, the object tape of the debugged microprogram is loaded into a buffer in Main Memory, using the Micro Debug Editor, and a set of six mask tapes are punched. These tapes are used by the HP 12909 pROM Writer to create ("burn") the programmed Read Only Memory (pROM) chip. The pROM chip is installed on an HP 12945A User Control Store board that is set by jumper wires to specify the proper Control Store module number.

3-21. ACCESS TO MICROPROGRAMS IN CONTROL STORE

The control processor microprograms are divided into three groups.

a. The 21MX Instruction Set microprograms including the Basic Instruction Set, the Extended Instruction Group, and Floating Point.

b. Hewlett-Packard supplied special microprograms (for example, the 12977A Fast FORTRAN Processor option) if installed.

c. User microprograms, if installed.

The control processor reads a 16 bit instruction from Main Memory into the Instruction Register (IR), decodes it, and then determines which microprogram is called for by the instruction. This reading, decoding, and address determination is performed by microprograms that are an integral part of the Basic Instruction Set. The Basic Instruction Set microprogram is in some ways analogous to system software in a normal Main Memory operating system, since the Basic Instruction set performs the general control functions and passes control to the user microprogram area when the Instruction Register calls for a user microprogram. This enables the user-microprogrammer to concentrate effort on his special application.

For the purposes of decoding and implementing macro-instructions, the 21MX Instruction Set is divided into groups according to the general functions they perform. As shown in figure 3-3, there are five groups that encompass the 21MX Instruction Set. A sixth group called the User Instruction Group consists of the macro-instructions that allow the user to access the micro-programs which he writes. Most instruction set enhancements or special microprograms will be accessed by the general classification of "user" macro-instructions.

Figure 3-3 summarizes the processing of the Instruction Register. A microprogram within the Basic Instruction Set reads an instruction from Main Memory into the Instruction Register and determines to which macro-instruction group (Alter/skip, Memory Reference, etc.) that instruction belongs. This is accomplished by a ROM table branch command (SPECIAL micro-order "JTAB") that uses the upper eight bits of the Instruction Register to jump, via the fixed ROM Main Look Up Table, to a Control Store microprogram address, according the value of those eight bits. Once the general instruction group is determined, the Instruction Register is further decoded and the logic implemented by the microprogram designed to implement that macro-instruction.

For example, if the instruction in the Instruction Register is in the Extended Arithmetic Unit (EAU) Group, the EAU Group microprogram address is found in the Main Look Up Table based on the Op Code of the instruction. Then the EAU Group microprogram executes the EAU instruction. Provided in the micro-instruction set are special jump parameters, such as "JEAU", to branch within the EAU Group microprogram according to which member of the group is being processed. Jump parameters are explained in Section IV of this manual.

3-22. USER FUNCTION CODE IN ASSEMBLY LANGUAGE

The assembly language program calls a microprogram using mnemonic codes that are assigned in the assembly language program. The pseudo op "MIC" is used to assign the mnemonic code. Refer to the HP Assembler Reference Manual (HP 24307-90014) for the use of the MIC pseudo op.

Using the MIC instruction, a binary function code is assigned to the mnemonic so that whenever the mnemonic appears, the function code is written into that location of the assembled program. The number of parameters is also specified.

The octal function code that calls the user microprogram is:

105rrr if bit 8 of the IR = 0

101rrr or 105rrr if bit 8 of the IR = 1

The value of rrr (bits 8-0) determines the Control Store module address. rrr is defined in table 3-1. Bit 11 in the third digit (5 or 1) is used by micro-instructions which test data in the Instruction Register, where the function code is interpreted. For example, see the "CAB" S-bus micro-order.

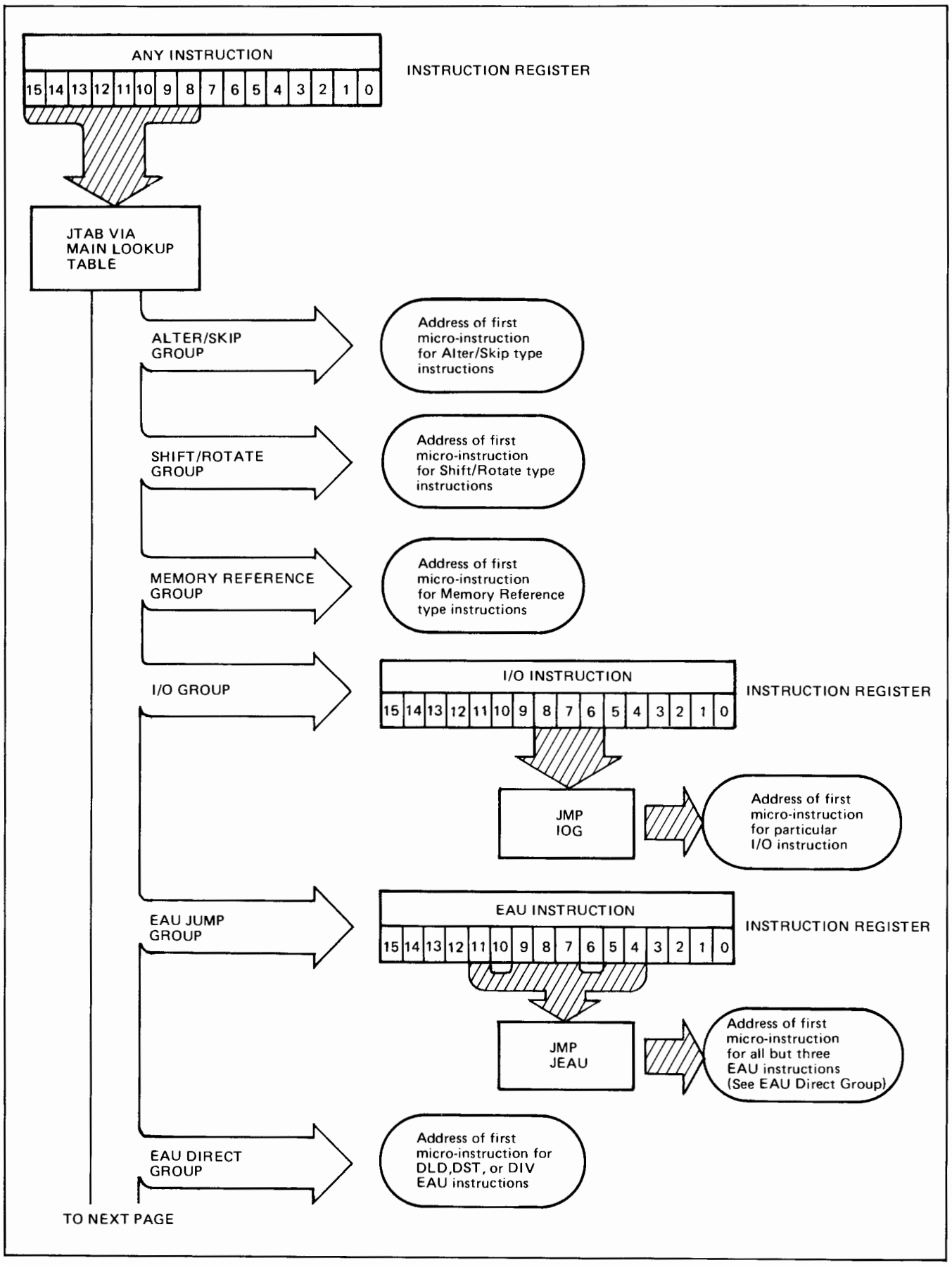


Figure 3-3. Processing the Instruction Register (Sheet 1 of 2)

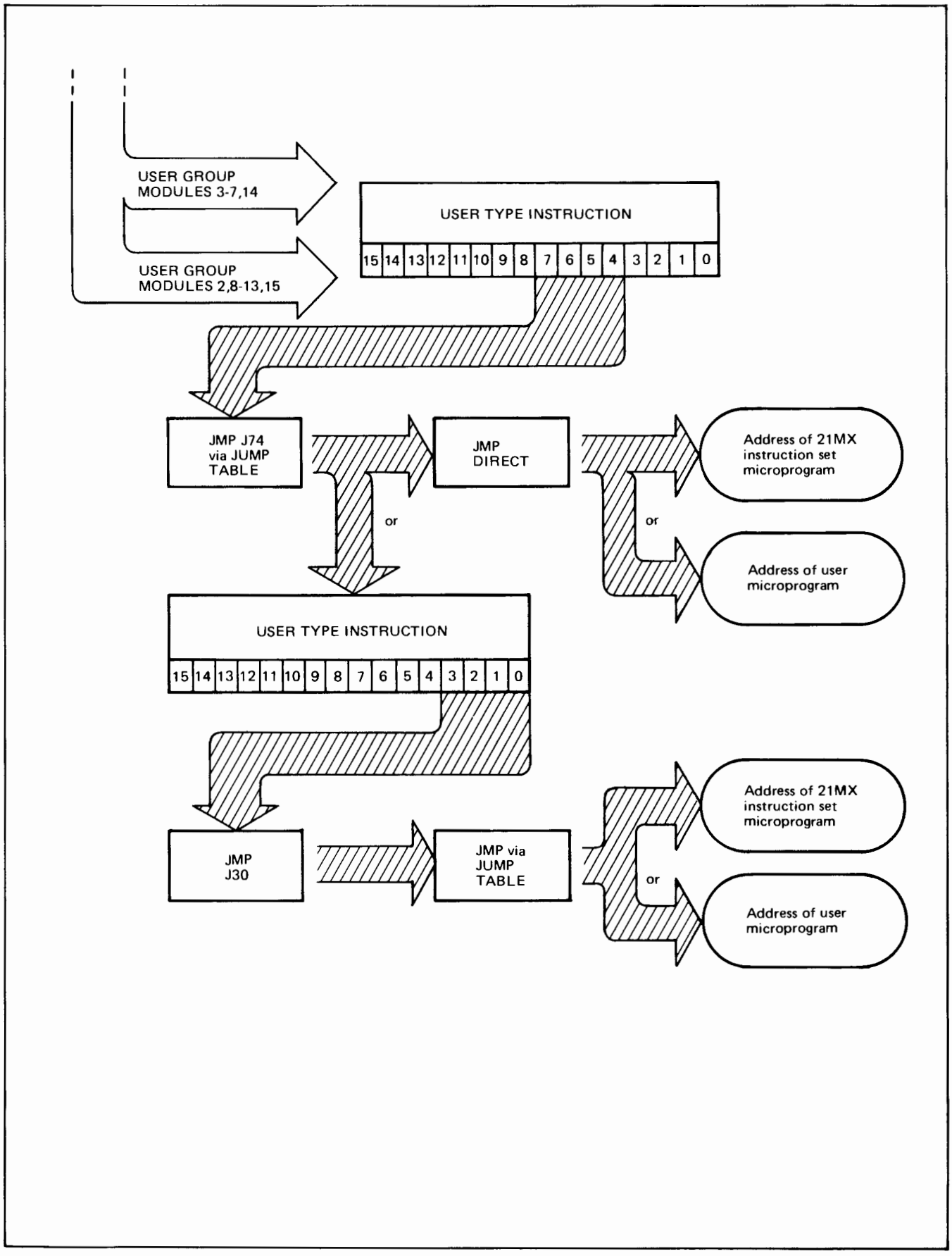


Figure 3-3. Processing the Instruction Register (Sheet 2 of 2)

3-23. CONTROL STORE MODULES AVAILABLE TO USER

The 4096 words of ROM are divided into sixteen 256-word modules, module 0 through module 15. Modules 0, 1, 14, and 15 hold the 21MX Instruction Set and are not available to the user microprogrammer. Modules 12 and 13 are reserved exclusively for user microprograms. Any other Control Store space, not filled by a microprogrammed option, is available to the user microprogrammer. Figure 3-4 summarizes the allocation of Control Store.

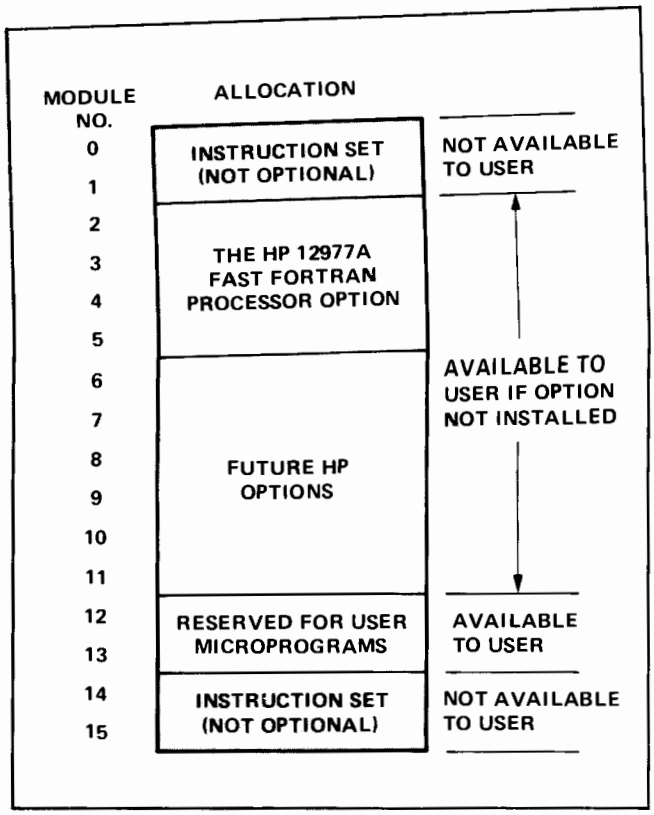


Figure 3-4. Allocation of Control Store by Modules

3-24. MAPPING TO A MODULE ADDRESS

Function codes available to the user are listed in table 3-1 together with the module address to which these function codes map. Some of these user function codes are assigned to the microprogrammed processors and options produced by Hewlett-Packard. The following function codes cannot be used:

105000 through 105137
105740 through 105777
101740 through 101777

If the HP 12977A Fast FORTRAN Processor is installed, the following function codes are not available to the user:

105140 through 105277
105700 through 105737
101700 through 101737

Note: If the function code maps to a Control Store module which is not present, the micro-instruction

JEAU PASS S S

is executed for each non-existent Control Store location. The ROM Address Register is incremented after each execution of the above micro-instruction until an installed module is encountered. No notification is given to the user or system that a non-existent module is being executed.

Table 3-1. User Function Code Mapping

Function codes $101rrr_8$ and $105rrr_8$ map to the module address given:

	RANGE OF rrr VALUES	MODULE	RANGE OF OCTAL ADDRESSES
$105rrr_8$ only	140 to 157	3	1400
	160 to 177	3	1400 to 1417
	200 to 217	4	2000
	220 to 237	4	2000 to 2017
	240 to 257	5	2400
	260 to 277	5	2400 to 2417
	300 to 317	6	3000
	320 to 337	6	3000 to 3017
	340 to 357	7	3400
	360 to 377	7	3400 to 3417
$101rrr_8$ or $105rrr_8$	400 to 417	8	4000
	420 to 437	8	4000 to 4017
	440 to 457	9	4400
	460 to 477	9	4400 to 4417
	500 to 517	10	5000
	520 to 537	10	5000 to 5017
	540 to 557	11	5400
	560 to 577	11	5400 to 5417
	600 to 617	12	6000
	620 to 637	12	6000 to 6017
	640 to 657	13	6400
	660 to 677	13	6400 to 6417
	700 to 717	2	1000
	720 to 737	2	1000 to 1017

3-25. MICROPROGRAMMING INPUT AND OUTPUT FUNCTIONS

Microprogramming Input and Output (I/O) functions requires more care than any other type of microprogramming, because there are strict timing dependencies. The microprogram described in section 3-40 is an example of I/O microprogramming.

To maintain integrity of the I/O system, every control signal which goes to the I/O devices is generated in a specific time period (T-period). All micro-instructions, except those containing READ or WRTE micro-orders, are executed in one I/O T-period, where T = 325 ns. READ and WRTE each require two I/O T-periods. An I/O time cycle consists of five T-periods labelled T2, T3, T4, T5, and T6. Specific I/O activity is restricted to certain T-periods in order to synchronize setting of data flags, latching of data, and resolving of multiple interrupt requests.

The microprocessor must synchronize with T2 before initiating an I/O cycle. Thereafter, special consideration must be given to the order and timing of the I/O micro-instructions given.

3-26. SYNCHRONIZING WITH THE I/O SYSTEM

To initiate an I/O cycle, the IOG micro-order must be specified. When this occurs, the processor "freezes" (ceases executing micro-instructions) until time T2. The next micro-instruction is executed during time T3, the next during T4, etc. IOG may occur with any micro-instruction which does not require some other Special or Jump Modifier (Field 3) micro-order.

Examples:

a. READ IOG INC PNM P

b. IOG PASS IR S3

3-27. I/O SIGNAL GENERATION

When IOG is specified, the I/O system generates control signals to the I/O devices starting at the next T2 time and according to the contents of the Instruction Register (IR).

IR bits 5-0 hold a Select Code (SC) signal (SC = The I/O slot number on the backplane or in I/O extenders) that determines which device will respond to the control signal. IR bits 11-6 determine which I/O signals are sent, as shown in table 3-2. The IR must be loaded prior to or during occurrence of the IOG to insure that the correct signals are generated to the proper SC.

Select Codes 0, 1, 2, 3, 4, and 5 have special functions concerning, respectively, the interrupt system, the Front Panel, the Dual Channel Port Controller (DCPC), Power Fail, and Memory Protect/parity. The "Interrupt and Control summary" table in the Appendix of the **HP 21MX Computer Series Reference** manual (HP 02108-90002) holds a description of the effect of these select codes (S.C. in the table).

Table 3-2. I/O Control Signal Generation Determined by IR Bits 11-6

IR*								
11	10	9	8	7	6	I/O SIGNAL	TIME	GENERAL USE
x	x	x	0	0	0	none	T3	Turns off the Run Flag on the CPU.
x	x	x	0	0	1	STF	T3	Set device flag.
x	x	1	x	x	x	CLF	T4	Clear device flag.
x	x	x	0	1	0	SFC	T3-T5	SKPF condition is true if and only if the device flag is clear.
x	x	x	0	1	1	SFS	T3-T5	SKPF condition is true if and only if the device flag is set.
—	—	—	—	—	—	IOI	T4-T5	Buffer the input data latch on the device onto the I/O-bus; this command must be stated explicitly in micro-code during these times.
x	x	x	1	1	0	IOO	T3-T4	Store the I/O-bus into the input data latch on the device.
0	x	x	1	1	1	STC	T4	Set device control flag.
1	x	x	1	1	1	CLC	T4	Clear device control flag.

*Bits marked with x are not significant for the I/O signal specified.

3-28. MEMORY PROTECTION IN RELATION TO I/O MICROPROGRAMMING

When the Instruction Register is loaded, the Memory Protect (MP) feature (12892A) records information on the instruction (from Main Memory) being stored in the IR. When an IOG micro-order is specified, MP checks the select code. If it is not equal to 1 (Front Panel) and MP control is set, MP will inhibit any I/O signals and prevent the CPU from altering memory or the P- or S-registers, and will generate an interrput request. The microprogrammer cannot prevent this function, so the software operating system maintains security of I/O programming with MP in the microprogramming environment.

3-29. I/O CONTROL ROUTINE

This type of I/O function requires no data transfer. The IR must specify:

STF
CLF
SFS
SFC
STC
CLC

Note that CLF can be generated in conjunction with any other signal by merely letting bit 9 of the IR equal one. To simulate a CLF macro-instruction, specify CLF with STF. Once IOG has been given in an I/O control routine, there are no limitations in using micro-instructions because I/O signals are generated automatically.

For SFS and SFC, the state of the flag on the device may be tested with a "JMP CNDX SKPF" instruction. SKPF is true only when SFS is being executed and the flag is set, or when SFC is being executed and the flag is clear. The SKPF test should occur during T4 or T5 of a SFS or SFC routine. Any operation desired may be implemented as a result of this test. To cause a macroprogram skip, simply increment the P-register contents.

3-30. I/O OUTPUT ROUTINE

This routine is characterized by generation of the IOO micro-order. The IOO sends data from the I/O-bus into the input data latch on the device. The microprogram must put the proper data on the S-bus, then direct it onto the I/O-bus. The detailed timing requirements are:

a. During T3, the S-bus must be driven by the register containing the output data to prepare for the transfer to the I/O bus.

b. During T4 and T5, the S-bus must be driven by the same register and IOO must appear in the Store field. This insures valid data on the I/O bus.

For example, the sequence for a standard OTA macro-instruction is:

```
(Time T2)   IOG
(Time T3)         PASS   CAB
(Time T4)         IOO    CAB
(Time T5)   RTN   IOO    CAB
```

3-31. I/O INPUT ROUTINE

This routine is characterized by use of IOI in the S-bus field. IOI is used in the I/O cycle during T4 or T5 to input data from the I/O device PCA onto the I/O-bus and then onto the S-bus. Any normal Word Type 1 instruction may be used to store the data input from the S-bus.

For example:

```
(Time T2)   IOG
(Time T3)   NOP
(Time T4)   NOP
(Time T5)   RTN   PASS   CAB   IOI
```

It can be seen that during some parts of some I/O routines, there are instruction times which are unused. Caution is required when using these times. Do not use micro-instructions which may cause the processor to freeze (listed in section 3-36), until all I/O related code has been executed for that I/O cycle. In the above example, if the T3 and T4 NOPs were replaced by READ and T (S-bus field) micro-orders, the CPU would freeze in the middle of T4 and IOI would not be executed until T6 — too late to correctly handle the data transfer. On the other hand, during a control type routine which is not performing an SFS or SFC, many kinds of micro-instructions can be performed after the IOG. These include READ or even another IOG, since the I/O system requires no further assistance from the microprocessor.

3-32. INTERRUPT HANDLING

The presence of a pending interrupt or halt request may be detected by microcode in two ways:

a. Performing a test with JMP CNDX on INT, NHOI, or RUN.

b. Attempting to JMP or RTN to location 0 in Control Store; a pending interrupt or halt will cause Control Store address 4 to be loaded into the RAR.

The interrupt device select code (SC) can be read onto the S-bus (high order bits = 0) by specifying CIR in the S-bus field. This freezes the CPU until T6 and then sends IAK to the interrupting device. In the Basic Instruction Set microprogram, the select code from the CIR is loaded into the M-register and the Main Memory instruction at that address is executed. Note that the P-register is not altered during this process.

3-33. NORMAL USER INTERRUPT HANDLING APPLICATIONS

If a long microprogram is entered, the program itself has complete control over when it is terminated or suspended for a detected interrupt. It is not desirable to hold off interrupts very long. Magnetic tape, for example, might request an interrupt every 27 microseconds, if not transferring data by way of the Dual Channel Port Controller.

It is up to the microprogrammer to decide how long to wait before testing for an interrupt. When an interrupt is detected, a jump should be made to a routine to save whatever is necessary to allow the microprogram to continue after the interrupt is serviced or to provide for complete restart of the microprogram. The P-register must be reset to point to the Main Memory address of the macro-instruction interrupted. If parameters are saved, a test must be made at the beginning of the microprogram to determine if it was interrupted or if it executes from the beginning.

When the interrupt servicing is started, a JMP or RTN is made to Control Store location 4 where the Basic Set microcode takes the trap cell address from the Central Interrupt Register and then gives control to Main Memory programs which service the interrupt. After the interrupt routine is complete, the interrupted microprogram is restarted (assuming the P-register was reset upon interrupt detection).

3-34. MICRO-ORDERS AFFECTING MEMORY PROTECT

To fully use the level of protection afforded by the 12892A Memory Protect feature, some conventions must be followed in microprogramming to assure proper communication between the processor and the Memory Protect feature (MP).

Note that MP can only be enabled and disabled by the I/O system. There are no microcode commands for it. Refer to the Memory Protect Interrupt section in the **HP 21MX Reference Manual** for further discussion. The micro-orders which communicate with MP are listed below together with a description of their rules and functions:

a. FTCH (Special field). This reads the M-register into the MP Violation register, clears out the MP Violation flag and resets the Indirect counter. It should be given when the address of the current instruction from Main Memory is being read (READ micro-order) or immediately after. FTCH occurs in the following places in the Basic Instruction Set Microprogram:

   1. At location 0, the Fetch routine.

   2. At the location MGOOD+1 in the Halt routine to reset the MP Violation flag and to enable alteration of P-register, S-register, and Main Memory from the Front Panel.

   3. At location SCAN+12 as part of the single instruction fetch routine, where it serves the same purpose as at location 0.

b. IR (Store field). Whenever the IR is specified in the Store field, MP records whether the instruction is a Halt, JMP, or neither, and whether or not IR bits 5-0 equal 01 or not. The IR must be loaded prior to initiating an I/O cycle with IOG to insure that the signal decoding logic will take effect.

c. INCI (Special field). This micro-order should be used whenever another level of indirect addressing is detected by a microprogram. After 3 counts of the Indirect Counter, an ION (enable interrupts) micro-order is effectively performed by the Memory Protect option. A microprogrammed IOFF micro-order will have no effect after this occurs until after the next FTCH is executed.

d. MPCK (Special field). There is no need to use this memory protect check micro-order if the Memory Protect feature (HP 12892A) is not installed. This micro-order should be used to insure that a microprogram will not alter protected memory. When this micro-order is used and a MP violation is detected:

   1. All future READ instructions put invalid data into the T-register.

   2. No WRTE instructions are performed.

   3. All attempts to alter the P- or S-registers fail.

   4. All I/O signals from the processor are inhibited until after the next FTCH or CIR is executed.

e. IOG (Special and Jump Modifier). If Memory Protect has been enabled, this micro-order will set the Memory Protect Violation flag if the select code (IR bits 5-0) is not equal to one. If a MP violation is detected, the actions 1 through 4 described in d. MPCK take place.

f. CIR (S-bus field). This micro-order causes a freeze until T6 and then issues an IAK to acknowledge the granting of an interrupt to the requesting device. If the select code is 5, the Parity indicator on the Front Panel is cleared and the Memory Protect Violation flag is cleared. Whenever CIR occurs, special logic on the Memory Protect PCA determines whether or not the MP should be disabled (Clear the Control bit). This determination is made six micro-instructions after the last CIR:

   1. MP is not disabled if an I/O instruction (IOG) is executed that is not a halt.

   2. MP is disabled if no I/O instruction (IOG) is executed or a halt is executed.

   To re-enable Memory Protect, an STC 5 is required.

3-35. THE EFFECT OF THE DUAL CHANNEL PORT CONTROLLER ON MICROPROGRAMS

The Dual Channel Port Controller (optional hardware) steals full I/O cycles to perform direct transfers between external devices and Main Memory. This process is essentially transparent to the microprogram. The Dual Channel Port Controller (DCPC) is a hardware function that does not employ microcode. If the microprogram interferes with a DCPC cycle, the Control Processor freezes until DCPC completes its cycle. If DCPC takes a sequence of consecutive I/O cycles for input transfers, any attempted IOG, READ, or WRTE micro-orders will freeze the processor until DCPC is finished. If DCPC takes a sequence of consecutive I/O cycles for output transfers, the Memory Reference Group, the Alter/skip Group, and Shift Rotate Group macro-instructions can still proceed at between 40% and 60% normal execution rate; IOG will still freeze the Control Processor.

If DCPC takes as much as 50% of all I/O cycles, the overall efficiency of the basic instruction set execution is 60% to 70% for input or output transfers. Non-main Memory and I/O micro-instruction execution are only frozen 20% of each DCPC cycle. Thus arithmetic and logical micro-instructions execute at 80% efficiency, when DCPC takes every I/O cycle.

3-36. SUMMARY OF SPECIAL TIMING RULES

a. Always load the M-register before specifying WRTE in the OP micro-order field.

b. Load the M-register before or during micro-instructions containing READ in the OP field.

c. Do not alter the T-register unless initiating a WRTE, since the T-register is internal to the Main Memory system and is used by DCPC and the CPU. The T-register is not intended to be a general purpose register, but to be used in referencing Main Memory.

d. Load the T-register with data to be written in the same instruction as WRTE appears, or DCPC could alter it before WRTE is executed.

e. The T-register must be placed on the S-bus no later than two micro-instructions after a READ is specified or the T-register will be disabled by the Memory system.

f. When an I/O cycle (using IOG) is in progress, a READ or WRTE **must not be initiated** before T6 in the cycle under either of the following conditions:

1. An input or output routine (refer to sections 3-29 and 3-30) is in progress.

2. A skip flag test of the I/O system is taking place.

g. Do not specify a READ or WRTE micro-order in the same micro-instruction that is transferring data from the T-register (T or TAB micro-order in the S-bus field). The reason is that if a freeze occurs as a result of such a READ or WRTE micro-order (see i. below) the data in the T-register will be invalid after the freeze.

For example, a sequence of micro-instruction similar to the following must not take place:

READ	—	INC	PNM	P
—	—	PASS	S4	L
READ	—	INC	M	TAB

h. Do not start an I/O cycle (using IOG) before data is transferred from the T-register following a READ operation. The reason is that if the IOG results in a freeze (see i. below), the data in the T-register will be invalid.

For example, a sequence of micro-instructions similar to the following must not take place:

READ	—	INC	PNM	P
—	IOG	PASS	S4	TAB

i. The following conditions always cause a micro-processor freeze:

1. The CIR micro-order is in the S-bus field and either the I/O cycle time is not T6 or the Dual Channel Port Controller is stealing a full I/O cycle.

2. The IOG micro-order is in the Special field and either the I/O cycle time is not T2 or the Dual Channel Port Controller is stealing a full I/O cycle.

3. A T or TAB micro-order is in the S-bus field and a READ or WRTE micro-order memory cycle is still in progress.

4. A READ or WRTE micro-order is in the Op field and one of the following conditions is true:

   (a) The semi-conductor Main Memory is being refreshed (two micro-instruction cycles are required every 32.5 microseconds for this purpose).

   (b) The Dual Channel Port Controller is stealing an I/O cycle.

   (c) A READ or WRTE memory cycle is still in progress.

3-37. SAMPLE MICROPROGRAMS

While reading the sample microprograms, the reader may find it useful to refer to the fold out functional block diagram in Appendix D. This diagram and the micro-order definitions in Section IV are the two basic sets of information used by the programmer in writing a microprogram.

3-38. SWAP MEMORY LOCATIONS

The sample microprogram illustrated in figure 3-5 swaps the contents of two Main Memory locations that are pointed to by the A- and B-registers (no indirect addresses).

Micro-instruction Commentary

```
READ        INC   M    A
```

a. Put the address in the A-register onto the S-bus.

b. Store the S-bus into the M-register.

c. Pass the S-bus through the ALU and increment data enabling the A- or B-register addressable test.

d. Read the location in Main Memory pointed to by the M-register (this requires 2 micro-instruction cycles).

```
MPCK  PASS  M
```

a. Put the M-register onto the S-bus.

b. Pass the S-bus through the ALU (output not used).

c. Since READ requires two cycles, an instruction cycle is available before data is available from memory. And since the M-register holds the address of the location that will eventually be written into, this cycle is used for the memory protect check.

```
PASS  S1    TAB
```

a. The read is complete and data from the memory location is in the T-register unless the AAF or BAF Flag is set. If AAF is set, the data is in the A-register. If BAF is set, the data is in the B-register.

b. Put memory data on the S-bus.

c. Pass S-bus through the ALU and R/S to the T-bus.

d. Store data on T-bus into Scratch Pad Register 1 (S1).

```
READ        INC   M    B
```

a. Put the address in the B-register onto the S-bus.

b. Store S-bus into the M-register.

c. Pass the S-bus through the ALU and increment data enabling the A- or B-register addressable test.

d. Read the Main Memory location pointed to by the M-register.

```
MPCK  PASS        M
```

a. Put M-register (memory address) onto the S-bus.

b. Pass the S-bus data through the ALU.

c. Test the address for a Memory Protect violation.

```
PASS  S2    TAB
```

a. Put memory data (T-, A-, or B-register contents) onto the S-bus.

```
                Op Code  Special  ALU  Store  S-bus        Comment

$ORIGIN=2000B
$SYMTAB
          READ           INC   M     A          READ WORD POINTED TO BY A
               MPCK      PASS        M          CHECK ADDRESS
                         PASS  S1    TAB        STORE DATA IN S1
          READ           INC   M     B          READ WORD POINTED TO BY B
               MPCK      PASS        M          CHECK ADDRESS
                         PASS  S2    TAB        STORE DATA IN S2
          WRTE           PASS  TAB   S1         BEGIN WRITE
                         INC   M     A          LOAD M WITH A
          WRTE RTN       PASS  TAB   S2         WRITE AND RETURN
$END
```

Figure 3-5. Swap Microprogram

b. Pass S-bus through the ALU and R/S to the T-bus.

c. Store data on the T-bus into Scratch Pad Register 2 (S2).

WRTE	PASS	TAB	S1

a. The contents of the first memory location is in S1. Put S1 onto the S-bus.

b. Store the S-bus into T-register (or A- or B-register if AAF or BAF, respectively are set).

c. Pass S-bus data through the ALU.

d. Write T-register contents into Main Memory at address pointed to by the M-register. Note that the M-register still holds the second memory location address. It was loaded during last read operation.

INC	M	A

a. The A-register holds the first memory location. Put the A-register contents onto the S-bus.

b. Store the S-bus into the M-register.

c. Pass S-bus data through the ALU and increment data enabling the A- or B-register addressable test.

WRTE	RTN	PASS	TAB	S2

a. The contents of the second memory location is in S2. Put S2 onto the S-bus.

b. Store the S-bus into the T-register (or A- or B-register, if AAF or BAF, respectively, are set).

c. Pass S-bus data through the ALU.

d. Write the T-register contents into Main Memory at the address pointed to by the M-register.

e. Exit (RTN micro-order).

3-39. BLOCK MOVE MICROPROGRAM

The sample program illustrated in figure 3-6 moves a group of words in Main Memory from one location to another. When the microprogram receives control, it is assumed that:

- The negative value of the number of words to be moved is in the A-register in two's complement form.
- The FROM address is in the B-register.
- The TO address is in the Main Memory location pointed to by the P-register and cannot be indirect.

```
                Op Code  Special  ALU   Store   S-bus        Comment

$ORIGIN=2000
$SYMTAB
$FILE=FILMOV
          JMP                             MOVE
MOVE                     PASS   S1        A          WORD COUNT = 0 ?
          JMP    CNDX    TBZ             OUT        IF ZERO, THEN GO TO "OUT"
*
          READ             INC    M       P          GET "TO" ADDRESS
                           PASS   S2      TAB        PUT IT IN S2
*
LOOP      READ             INC    M       B          READ A DATA WORD
                           PASS   S3      TAB        STORE THE WORD IN S3 REG
          READ             INC    M       S2         GET "TO" ADDRESS
                           INC    S2      S2         INCREMENT "TO" ADDRESS
          WRTE             PASS   T       S3         WRITE A DATA WORD TO MEMORY
                           INC    B       B          INCREMENT "FROM" ADDRESS
                           INC    S1      S1         INCREMENT   WORD COUNT
          JMP    CNDX    TBZ    RJS     LOOP       GO TO "LOOP" IF WORD
*                                                   COUNT IS NOT ZERO
OUT               RTN     INC    P       P          INCREMENT THE P REG AND EXIT
$END
```

Figure 3-6. Block Move Microprogram

The HP assembly language calling sequence is as follows:

```
LDA — (number-of-words)
LDB FROM-address
OCT 105200
DEF TO-address
```

Note: This microprogram is a translation of the Block Move microprogram shown in Section VI of the **HP 2100 Computer Microprogramming Software** manual (HP 02100-90133). Thus it can be used to compare HP 2100 microprogramming to HP 21MX microprogramming.

Micro-instruction Commentary

```
MOVE  —     —     PASS  S1  A
      JMP   CNDX  TBZ   —   OUT
```

Store the contents of the A-register in Scratch Pad Register 1. If the contents of the A-register are zero, then go to OUT address and return to the calling program.

```
      READ  —     INC   M   P
      —     —     PASS  S2  TAB
```

Get the TO address and store it in Scratch Pad Register 2. The TO address cannot be indirect.

```
LOOP  READ  —     INC   M   B
      —     —     PASS  S3  TAB
```

Read a data word from the Main Memory location pointed to by the FROM address and store the data word in Scratch Pad Register 3. Note that a Control Processor freeze will occur.

```
                  INC   M   S2
                  INC   S2  S2
      WRTE  —     PASS  T   S3
```

Write the data (in Scratch Pad Register 3) into memory. Increment TO address pointer.

```
      —     —     INC   B   B
      —     —     INC   S1  S1
      JMP   CNDX  TBZ   RJS LOOP
```

Increment the FROM address pointer. Increment the word count. If the word cound is not zero, go to LOOP.

```
OUT   —     RTN   INC   P   P
```

Increment the P-register beyond the word containing the TO address and exit.

3-40. INPUT, SUM, AND SUM OF SQUARES MICROPROGRAM

The sample microprogram illustrated in figure 3-7 loads a 16 bit word from a device specified by its select code "SC". If the word is equal to 177777 (end of transmission word), the microprogram is finished and this is signalled by executing the next instruction in Main Memory; otherwise:

a. The word is stored in memory location "DATA" indexed by the X-register.

b. The word is added to a running total kept in memory location "SUM"

c. The word is squared and added to a running total of squares in memory location "SQUAR".

d. Another input is initiated from the specified device (STC SC,C).

e. The next instruction in Main Memory is skipped to indicate that 177777 was not input from the specified device.

Conditions:

a. All numbers are 16 bit positive integers.

b. If SUM exceeds 2^{16}-1, the Extend Register is set.

c. If SQUAR exceeds 2^{16}-1, the Overflow Register is set.

d. If both SUM and SQUAR are less than 2^{16}-1, the Extend and Overflow Registers are clear.

e. Memory protect check is performed on addresses used for a write into Main Memory.

Microprogram storage:

The microprogram resides in module 12 starting at octal address 6017.

Microprogram initiation:

Entry into the microprogram is caused by the execution of the following 5 words in Main Memory:

```
105637  USER CALL TO CONTROL STORE ADDRESS 6017
0000nn  nn = SELECT CODE "SC"
0aaaaa  "DATA" STORAGE ADDRESS (a table holding all input data)
0bbbbb  "SUM" STORAGE ADDRESS
cccccc  "SQUAR" STORAGE ADDRESS
(end of transmission return)  SUMMING TERMINATED BY EOT
(normal return)               SUMMING CONTINUES
```

```
          Op Code Special  ALU   Store   S-bus          Comments

$ORIGIN=6017B
          READ            INC   PNM     P            01/READ SC, INC P, SET UP TAB LOGIC
          IMM   L1        CMLO  S1      137B         02/000500 INTO S1-USE FOR INP COM LATER
                          PASS  L       TAB          03/STORE SC INTO L
                          IOR   S11     S1           04/CREATE INPUT COM 0005NN IN S11
          READ            INC   PNM     P            05/READ DATA ADR,INCR P,SET UP TAB LOGIC
          IMM   L4        CMLO  S1      303B         06/001700 INTO S1 FOR SET CONT COM LATER
                          PASS  S3      TAB          07/STORE DATA ADR INTO S3
                          PASS  IR      S11          08/LOAD IR WITH INPUT COMMAND
                IOG       IOR   S10     S1           09/
*09/      FREEZE TILL T2,START I/O, CREATE SET CONTROL COMMAND 0017NN IN S10
                          PASS  L       S3           10/T3 STORE DATA ADDRESS INTO L
                          ADD   S3      X            11/T4 ADD INDEX TO L, STR INTO S3
          ASG             PASS  A       IOI          12/T5 GET DEV WRD FROM I/O BUS, ST INTO A
*                                                    12.5/CLEAR E (IR6=1)
          JMP   CNDX      ONES          OUT          13/T6 JUMP OUT IF ALL ONES IN  DEV WORD
          READ            INC   PNM     P            14/READ SUM ADR, INCR P, SETUP TAB LOGIC
                          INC   X       X            15/INCR INDEX
                          INC   M       TAB          16/ STORE SUM ADR IN M, PREPARE TAB LOGIC
          READ                                       16.5/ READ SUM
          IMM             LOW   CNTR    0B           17/CLEAR CNTR TO PREPARE FOR REPEAT
                          PASS  L       TAB          18/STORE SUM INTO L
          ENVE            ADD   S7      A            19/ADD DEVICE WORD TO T, ENBL O&E,ST INS7
                MPCK      PASS          M            20/MEMORY PROTECT TEST ON SUM ADDRESS
          WRTE            PASS  TAB     S7           21/WRITE TOTAL INTO SUM ADDRESS
                COV       PASS  IR      S10          22/CL OV,PUT SET CNTRL-CL FLG COM INTO IR
                IOG       PASS  L       A            23/FRZ TILL T2, ST A INTO L, START I/O
                MPCK      INC   M       S3           24/T3;S3(DATA ADR&X) ST INTO M,MEM PROT
          WRTE            PASS  TAB     A            25/T4;WRITE DEV WORD INTO (DATA&X
          READ            INC   PNM     P            26/T5,T6;READ ADR OF SQUAR, SETUP TAB LOG
                          INC   M       TAB          26.5/ PREPARE TAB LOGIC
          READ            INC   P       P            27/INCR P=NORMAL RETURN, READ SQUARE
                RPT       PASS  B       TAB          28/STORE SQUAR INTO B, SETUP REPEAT
          MPY   R1        ADD   B       B            29/
*29/      (A TIMES L)&B, STORE RESULT INTO B,A
          JMP   CNDX      TBZ           NO.OVER      30/JMP IF MPY RESULTED IN B=0 (MSB IN B)
                SOV                                  31/SET OV BIT;RESULT GR TH ACCEPTABLE
NO.OVER         MPCK      PASS          M            32/MEM PROT CK ON SQUAR ADDRESS
          WRTE  RTN       PASS  TAB     A            33/WRITE RESULT INTO SQUAR LOCATION, RTN
OUT                       INC   P       P            34/INCREMENT P
                RTN       INC   P       P            35/INCR P TO INDICATE EOT RETURN, RETURN
$END
```

Figure 3-7. Input, Sum, and Sum of Squares Microprogram

The above instruction is coded in assembly language by defining the mnemonic SSI, function code, and four parameters:

a. Use the MIC pseudo op in the assembler to define the five word instruction by its mnemonic and number of parameters: MIC SSI,105637B,4

b. Code the following when calling the SSI microprogram:

```
SSI SC DATA SUM SQUAR
(end of transmission return) SUMMING     TERMI-
                                NATED BY EOT
(normal return)              SUMMING CONTINUES
      :
      :
**DATA AREA ****************
SC      EQU nnB  SELECT CODE OF DEVICE
DATA    BSS mm   BUFFER ARE TO HOLD ALL
                 INPUT DATA
SUM     OCT 0    "SUM" STORAGE LOCATION
SQUAR   OCT 0    "SQUAR"  STORAGE  LO-
                 CATION
```

Micro-instruction Commentary:

READ	—	INC	PNM	P

a. Upon entry into the microprogram, P is the address in Main Memory that follows the instruction that calls microprogram. Hence P is the address of the address containing the select code.

b. Place the P-register contents on the S-bus. Store the S-bus into the M-register. Pass the S-bus contents through the ALU incrementing the data in the ALU and store the result (from the T-bus) into the P-register. The address on the T-bus is tested by the T-or-A-or-B logic for use by the TAB micro-order.

c. Read the contents of the location in Main Memory specified by the address in the M-register. The read requires two cycles.

IMM	L1	CMLO	S1	137B

a. While the read is still in progress, a memory cycle is used to construct an input command to be used later.

b. Place an octal 137 in bits 7-0 of the S-bus. Bits 15-8 are automatically filled with ones.

c. Pass the S-bus through the ALU complementing the data. Shift the data left one bit as it passes through the Rotate/Shifter inserting a zero into bit 0.

d. Store the T-bus result into Scratch Pad Register 1. The result in S1 = 000500.

—	—	PASS	L	TAB

a. Store the result of the read from Main Memory (contents of T- or A- or B-register) onto the S-bus (the select code nn was read).

b. Store the S-bus into the L-register and pass the S-bus contents through the ALU (the PASS is effectively a non-operation since the T-bus data is not stored).

—	—	IOR	S11	S1

a. Place Scratch Pad Register 1 on the S-bus. Perform an "inclusive or" of L-register and S-bus in the ALU and store the result in S11.

b. S1 = 00050
L = nn (select code) } IOR = 0005nn in S11

The result in S11 is the complete input command for select code = nn.

READ	—	INC	PNM	P

a. The P-register now points to the DATA address.

b. Place the P-register on the S-bus. Store the S-bus into the M-register. Increment the S-bus contents as it passes through the ALU and store the resulting address into the P-register. The address on the T-bus is tested by the T-or-A-or-B logic for use by the TAB micro-order.

c. Read the contents of the address in Main Memory specified by the M-register (read the DATA address).

IMM	L4	CMLO	S1	303B

a. While the read is still in progress, the memory cycle is used to construct a set control-clear flag I/O command.

b. Place an octal 303 in bits 7-0 of the S-bus. Bits 15-8 are automatically filled with ones.

c. Pass the S-bus through the ALU complementing the data. Rotate the data left four bits as it passes through the Rotate/Shifter.

d. Store the T-bus result into Scratch Pad Register 1. The result in S1 = 001700.

—	—	PASS	S3	TAB

a. Place the result of the read from Main Memory (contents of T- or A- or B-register) onto the S-bus (the DATA address was read).

b. Pass the S-bus data through the ALU and store it into Scratch Pad Register 3.

—	—	PASS	IR	S11

a. Place Scratch Pad Register 11 on the S-bus and store the S-bus into the Instruction Register (IR). IR now holds the input command 0005nn, where nn is the device select code.

—	IOG	IOR	S10	S1

a. IOG commands the microprocessor to freeze until time T2. At time T2 the input command in the Instruction Register is executed (transmitted to the device).

b. The L-register still holds the select code of device.

c. Place Scratch Pad Register 1 (holding 001700) on the S-bus. Perform an "inclusive or" with the L-register in the ALU. Store the result (0017nn) into Scratch Pad Register 10.

d. The net result in S10 is the completed set control — clear flag command.

```
        —       PASS    L       S3
```

a. Place Scratch Pad Register 3 (holding DATA address) onto the S-bus and then store S-bus into the L-register.

b. The PASS is essentially a non-operation.

```
        —       ADD     S3      X
```

a. Place the X-register (index to the number of words so far input from the device) onto the S-bus.

b. Add the S-bus to the L-register (now containing DATA address).

c. Store the result in Scratch Pad Register 3.

```
ASG     —       PASS    A       IOI
```

a. The time is T5. Take the word input from the Device from the I/O-bus and place it on the S-bus.

b. Pass the S-bus data through the ALU and store it into the A-register.

c. The IR = 0005nn, where nn is the device select code. Perform an Alter/Skip Group instruction (ASG) according to bits 7 and 6 in the IR. Since bits 7 and 6 = 01, perform a CLE (Clear Extend register bit).

```
JMP     CNDX    ONES    —       OUT
```

If the word last passed through the ALU (see previous micro-instruction) was all ones (end of transmission), jump to the location with the label OUT.

```
READ    —       INC     PNM     P
```

a. The P-register now points to the SUM address.

b. Place the P-register onto the S-bus. Store the S-bus into the M-register. Increment the S-bus contents as they pass through the ALU and store the resulting address into the P-register. The address on the T-bus is tested by the T-or-A-or-B logic for use by the TAB micro-order.

c. Read the contents of the address in Main Memory specified by the M-register (read the SUM address).

```
        —       —       INC     X       X
```

Increment the X-register, which is an index to the number of words input from the device.

```
        —       INC     M       TAB
```

a. Place the result of the read from Main Memory (contents of T- or A- or B-register) onto the S-bus (the address of the SUM was read).

b. Store the data on the S-bus into the M-register.

c. Increment the data in the ALU and place it on the T-bus so that the data is tested by the T-or-A-or-B logic.

```
READ    —       —       —       —
```

Read the contents of the address in Main Memory specified by the M-register (the present SUM value).

```
IMM     —       LOW     CNTR    0B
```

a. While the read is still in progress, the memory cycle is used to clear the Counter Register in preparation for the RPT used later in the microprogram.

b. Place zero on the lower eight bits of the S-bus. All ones are automatically stored in the upper eight bits.

c. .Store the S-bus into the Counter Register.

```
        —       —       PASS    L       TAB
```

a. Place the result of the read from Main Memory (contents of T- or A- or B-register) onto the S-bus (the present SUM value was read).

b. Store the S-bus into the L-register.

```
ENVE    —       ADD     S7      A
```

a. The A-register still contains the word input from the device. Place the A-register onto the S-bus.

b. Enable the Overflow test and Extend Register test in this micro-instruction only.

c. Add the L-register (current SUM value) to the S-bus in the ALU.

d. Store the result in Scratch Pad Register 7.

```
        —       MPCK    PASS    —       M
```

a. The M-register still holds the Main Memory address of SUM. Place the M-register onto the S-bus.

b. Pass the S-bus through the ALU.

c. Perform a memory protect check on the address since this address will be used for a write into Main Memory.

WRTE	—	PASS	TAB	S7

a. Place Scratch Pad Register 7 (holding the current DATA total) onto the S-bus.

b. Store the S-bus into the T-register (or A- or B-register according to AAF or BAF flags).

c. Initiate a write to Main Memory of the data in the T-register to the address in the M-register. This stores the new total of data words from the device back into the Main Memory address of SUM.

—	COV	PASS	IR	S10

a. Scratch Pad Register 10 holds the set control-clear flag command, 0017nn, where nn = the select code. Place Scratch Pad Register 10 onto the S-bus.

b. Store the S-bus into the Instruction Register.

c. Clear the Overflow Register.

—	IOG	PASS	L	A

a. IOG commands the microprocessor to freeze until time T2. At time T2 the set control-clear flag command in the Instruction Register is executed (transmitted to the device).

b. Place the A-register (which still holds the word input from the device) onto the S-bus.

c. Store the S-bus into the L-register.

—	MPCK	INC	M	S3

a. Place Scratch Pad Register 3 (which holds DATA address + index X) onto the S-bus.

b. Store the S-bus into the M-register.

c. Increment the data as it passes through the ALU and place it onto the T-bus. The data is tested by the T-or-A-or-B logic.

d. Perform a memory protect check on the S-bus data.

WRTE	—	PASS	TAB	A

a. Place the A-register (which still holds the word input from the device) onto the S-bus.

b. Store the S-bus into the T-register (or A- or B-register if the AAF or BAF flag is set).

c. Initiate a write to Main Memory of the data in the T-register to the address in the M-register. This stores the word input from the device into the Main Memory table of DATA values.

READ	—	INC	PNM	P

a. The P-register now points to the SQUAR address. Place the P-register onto the S-bus.

b. Store the S-bus into the M-register.

c. Increment the S-bus data as it passes through the ALU and then store the T-bus into the P-register.

d. Read the SQUAR address pointed to by the M-register.

—	—	INC	M	TAB

a. Freeze until last READ is complete, then place SQUAR address just read from Main Memory onto the S-bus.

b. Store the S-bus into the M-register.

c. Increment the data as it passes through the ALU and place it onto the T-bus. The data is tested by the T-or-A-or-B logic.

READ	—	INC	P	P

a. Place the P-register onto the S-bus.

b. Increment the data as it passes through the ALU and store it into the P-register. The P-register now contains the normal Main Memory return address.

c. Read the SQUAR contents from Main Memory (contains the current total of data squares).

—	RPT	PASS	B	TAB

a. Place the SQUAR contents (in the T- or A- or B-register) onto the S-bus.

b. Pass the S-bus through the ALU onto the T-bus and then store the T-bus into the B-register. The B-register now holds the current total of device input word squares.

c. Repeat the following micro-instruction incrementing the Counter Register after each repeat. When the Counter Register is equal to 377, execute the next micro-instruction.

MPY	R1	ADD	B	B

a. Perform a multiply step where the multiplier is in the L-register and the multiplicand is in the A-register.

b. Both the A- and L-registers hold the last word input from the device. The B-register holds the current total of word squares. Thus the result of 16 repeats of this multiply step is to square the word input from the device adding the result to the past total of squares [(A x L) + B].

c. The 32 bit result is in the B- and A-registers with the most significant bits in the B-register.

JMP	CNDX	TBZ	—	NO.OVER

a. Jump to the location in the microprogram with the label NO.OVER if the last value that passed onto the T-bus was equal to zero.

b. In a multiply step operation, the last data to go along the T-bus is the data that is stored into the B-register. Since the B-register holds the most significant bits of the multiplication result, if the result exceeds 2^{16}-1, bits will be set in the B-register.

—	SOV	—	—	—

Set the Overflow Register. The result of the multiplication operation (added to the B-register) exceeds 2^{16}-1.

NO.OVER	—	MPCK	PASS	—	M

a. Place the M-register (the SQUAR address) onto the S-bus.

b. Perform a memory protect check on the address on the S-bus. (To prepare to write the multiplication result back into the Main Memory data location (SQUAR.)

—	WRTE	RTN	PASS	TAB	A

a. Place the A-register (the current total of squares) onto the S-bus.

b. Store the S-bus into the T-register (or A- or B-register, if AAF or BAF flag is set).

c. Write the contents of the T-register into Main Memory at the address given in the M-register (the address of SQUAR).

d. Return to the Control Store address held in the SAVE Register. In general, this means return to 0 to read the next instruction from Main Memory at the address pointed to by the P-register.

OUT	—	—	INC	P	P

a. This micro-instruction (label OUT) is branched to, if the end of transmission character (177777) has been received from the device.

b. Increment the P-register.

—	—	RTN	INC	P	P

a. Increment the P-register again to point to the end of transmission return address in Main Memory.

b. Return to the Control Store address held in the SAVE Register. In general, this means return to 0 to read the next instruction from Main Memory at the address pointed to by the P-register.

3-41. READ A WORD FROM A LOADER ROM

The sample program segment illustrated in figure 3-8 reads four 4-bit bytes from a Loader ROM, constructs a 16 bit word, and then stores the word into Main Memory.

Conditions:

a. The A-register holds the Main Memory address into which the 16 bits read from the Loader ROM are to be stored.

b. The Loader ROM is selected by bits 15 and 14 of the Instruction Register. The particular Loader ROM selected does not affect the example.

c. The Counter Register is set to address the first location in the Loader ROM at the beginninng of the micro-program segment.

Micro-instruction Commentary:

—	IMM	—	LOW	CNTR	0B

a. Place a 0 onto the S-bus in bits 7-0; bits 15-8 are automatically filled with ones.

b. Store the S-bus into the Counter Register. Since the Counter Register is eight bits long, only bits 7-0 of the S-bus are stored into the Counter Register.

c. The Counter Register is now zero.

—	—	—	PASS	M	A

a. The P-register holds the Main Memory Address into which 16 bits are to be stored from the Loader ROM.

b. Place the P-register contents onto the S-bus.

c. Store the S-bus into the M-register for use later in the write to Main Memory of the word from the Loader ROM.

LOOP1	—	L4	PASS	S1	LDR

a. The LOOP1 label is used to identify this microprogram segment in the Basic Instruction Set microprogram.

b. Place a 4-bit-byte, addressed by the Counter Register, onto the S-bus. The Counter Register is equal 0; thus addressing byte 0 (there are 256 bytes addressed octal 0-377 in each Loader ROM). Note that each byte is stored on the S-bus in complemented form. Thus before a 16 bit word is stored into Main Memory, it must be complemented. This is taken care of by the next to last micro-instruction in this program segment.

c. Pass the S-bus through the ALU to the Rotate/Shifter. Left shift the data four bits.

d. Store the data on the T-bus into Scratch Pad Register 1 (S1). S1 now holds 16 bits of the form:

xxxxxxxxAAAAxxxx

where AAAA is the 4 bit byte just read.

—	—	INCT	PASS	L	S1

a. Place Scratch Pad Register 1 onto the S-bus.

b. Store the S-bus into the L-register.

c. Increment the Counter Register to address Loader ROM byte 1.

—	—	L4	AND	S1	LDR

a. Place byte 1 of the Loader ROM onto the S-bus.

b. Perform a logical "and" of the S-bus and the L-register in the ALU.

c. Left shift the data four bits in the Rotate/Shifter.

```
          Op Code  Special  ALU    Store  S-bus    Comments
            IMM             LOW    CNTR   B        CLEAR CNTR (ROM ADDR REG)
                    COV     PASS   M      A        PUT SA IN M

LOOP1               L4      PASS   S1     LDR      PASS XXXXXXXXAAAAXXXX INTO S1;CNTR=X00
                    INCT    PASS   L      S1       CNTR=X01
                    L4      AND    S1     LDR      FORM XXXXAAAABBBBXXXX IN S1;CNTR=X01
                    INCT    PASS   L      S1       CNTR=X10
                    L4      AND    S1     LDR      FORM AAAABBBBCCCCXXXX IN S1;CNTR=X10
                    INCT    PASS   L      S1       CNTR=X11
                            NAND   S1     LDR      FORM AAAABBBBCCCCDDDD (CMPL FORM)
            WRTE            PASS   T      S1       WRITE INTO MEMORY
```

Figure 3-8. Reading From a Loader ROM

d. Store the T-bus into Scratch Pad Register 1. S1 is now of the form:

xxxxAAAABBBBxxxx

where BBBB is the 4-bit-byte just read.

—	—	INCT	PASS	L	S1

a. Place the contents of Scratch Pad Register 1 onto the S-bus.

b. Store the S-bus into the L-register.

c. Increment the Counter Register to address Loader ROM byte 2.

—	—	L4	AND	S1	LDR

a. Place byte 2 of the Loader ROM onto the S-bus.

b. Perform a logical "and" of the S-bus and the L-register in the ALU.

c. Left shift the data four bits in the Rotate/Shifter.

d. Store the T-bus into Scratch Pad Register 1. S1 is now of the form:

AAAABBBBCCCCxxxx

where CCCC is the 4-bit-byte just read.

—	—	INCT	PASS	L	S1

a. Place S1 onto the S-bus.

b. Store the S-bus into the L-register.

c. Increment the Counter Register to address Loader ROM byte 3.

—	—	—	NAND	S1	LDR

a. Place byte 3 of the Loader ROM onto the S-bus.

b. Perform a logical "nand" of the L-register and the S-bus (L "and" S, the result complemented) in the ALU.

c. Store the T-bus in S1. S1 is now of the form:

AAABBBCCCDDD

where DDD is the 4-bit-byte just read. S1 now holds the completed 16 bit macro-instruction.

—	WRTE	—	PASS	T	S1

a. Place S1 onto the S-bus.

b. Store the S-bus in the T-register (the Main Memory Data Register).

c. Initiate a write to Main Memory (address in the M-register) of the data in the T-register.

This completes the reading of 4 bytes from the Loader ROM, constructing a 16 bit macro-instruction, and storing the macro-instruction in Main Memory.

MICROPROGRAMMING LANGUAGE

SECTION IV

This section serves as a reference to micro-instruction word definitions and formats.

There are four micro-instructions word types. Their general uses are defined below:

- **Word Type 1** executes
  a. Data transfers between Main Memory, I/O, and arithmetic and logic sections.
  b. Logical and arithmetic functions on data.
- **Word Type 2** specifies octal data to be transferred to a specific register.
- **Word Type 3** executes a conditional jump based on flags or data values.
- **Word Type 4** executes an unconditional jump or subroutine jump.

In addition, there are five Pseudo Instructions recognized by the micro-assembler.

Each word type has two formats. One format is the 24-bit **Binary Instruction Format.** This is the machine-language format; the format of the micro-instruction as it is stored in the ROM. The second format is the **Mnemonic Format.** This is the micro-assembler source format; the mnemonic-character representation of the micro-instruction.

Each micro-instruction consists of a number of **micro-orders,** which define the control steps to be executed within the system. The binary representation of the micro-orders falls within certain bits of the 24-bit Binary Instruction. The mnemonic representation of each micro-order falls within seven fields of the micro-instruction input record (e.g. a card). The binary and mnemonic formats are defined for word types in the following sections.

Common to all word types are the LABEL (Field 1), COMMENTS (Field 7), and "*" (column 1).

- LABEL

  This optional field is a string containing any ASCII characters except +, -, or a space. The string of characters can be one through eight characters long and must always start in column one with a "." (period) or a letter. A maximum of 256 locations address labels are allowed in any microprogram.

- COMMENT

  This optional field can be any string of up to 30 characters.

- *

  The asterisk indicates that the entire input record (card) is a comment field.

4-1. WORD TYPE 1 — COMMON

Charactor Column:

1	10	15	20	25	30	40	80
Field 1	Field 2	Field 3	Field 4	Field 5	Field 6	Field 7	
* or LABEL	OP	SPECIAL	ALU	STORE	S-BUS	COMMENTS	

Figure 4-1. Word Type 1 Micro-assembler Mnemonic Format

Bit No.	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Fields	OP				ALU					S-BUS					STORE					SPECIAL					

Figure 4-2. Word Type 1 Binary Format

There are five micro-order classifications in Word Type 1:

- OP — 12 operations
- SPECIAL — 32 special operations
- ALU — 32 ALU functions
- STORE — 32 destinations of data generated by the micro-instruction
- S-BUS — 32 sources for data to be used by the micro-instruction.

Micro-orders for Word Type 1 are defined in the following paragraphs. The mnemonic code is defined first, followed by its binary equivalent, the meaning, and any special conventions in the use of the micro-order.

4-2. OP MICRO-ORDERS

Many operation codes require specific micro-orders in other fields of the micro-instruction. Those that do will be defined in terms of all required and optional micro-orders in the fields of the micro-instruction.

OP
ARS

Required micro-instruction mnemonic fields:

OP	SPECIAL	ALU	STORE	S-BUS
ARS	L1 or R1	PASS	B	B

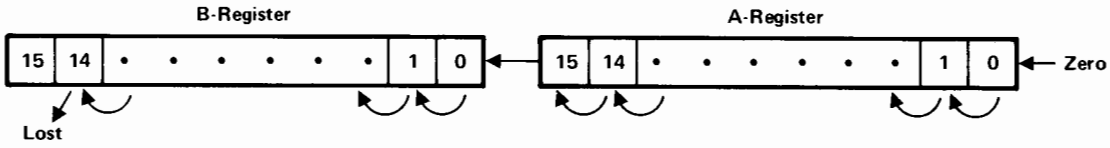
Equivalent micro-instruction binary fields:

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OP				ALU					S-BUS					STORE					SPECIAL				
0	0	0	1	1	1	1	1	1	0	1	0	1	0	0	1	0	1	0	L1 or R1 Code				

Meaning: Perform a single bit Arithmetic shift of the A- and B-register combined, with the A-register forming the low-order 16 bits. The direction of the shift is specified in the SPECIAL field: L1 for left, R1 for right.

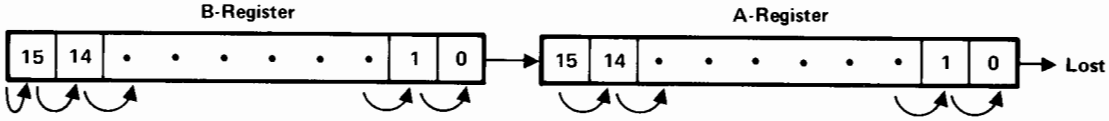
If L1, a 0 is shifted into bit 0 of the A-register; bit 14 of the B-register is lost, but the sign bit remains unchanged. The overflow register bit is set if bits 14 and 15 differ before the shift operation.

ARITHMETIC LEFT SHIFT: SPECIAL=L1



If R1, the sign is copied into bit 14 of the B-register and bit 0 of the A-register is lost.

ARITHMETIC RIGHT SHIFT: SPECIAL=R1



OP	BIT NO.	23	22	21	20
ASG	CONTENT	1	0	0	0

Meaning: Let bits 6 and 7 of the Instruction Register determine which of the following functions is to be performed; then clear the L-register.

IR Bit No.	7	6		
CLE	0	1	Clear Extend Register	Alter/Skip instruction
CME	1	0	Compliment Extend Register	
CCE	1	1	Set the Extend Register	

Conventions: This micro-order is used by the Basic Instruction Set microprograms which implement the Alter/skip Macro-instruction Group.

OP
CRS

Required micro-instruction mnemonic fields:

OP	SPECIAL	ALU	STORE	S-BUS
CRS	L1 or R1	PASS	B	B

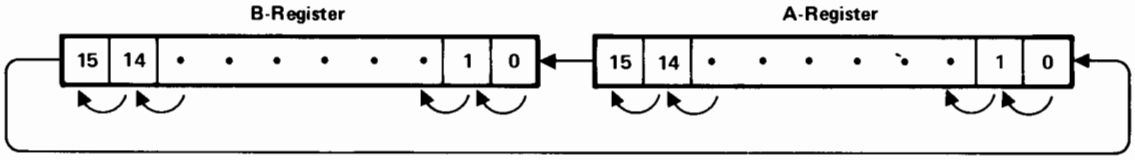
Equivalent micro-instruction binary fields:

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OP				ALU					S-BUS					STORE					SPECIAL				
0	0	1	0	1	1	1	1	1	0	1	0	1	0	0	1	0	1	0	L1 or R1 Code				

Meaning: Perform a single bit circular Rotate Shift of the A- and B-registers combined, with the A-register forming the low order 16 bits. The direction of the shift is specified in the SPECIAL field: L1 for left, R1 for right.

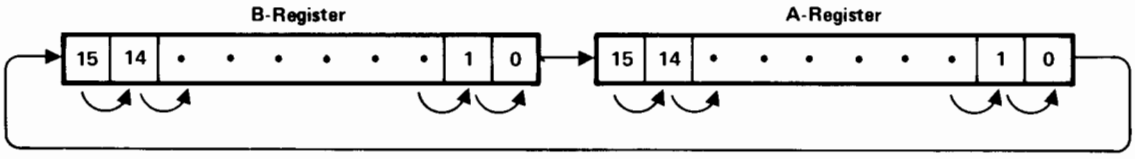
If L1, bit 15 of the B-register is transferred to bit 0 of the A-register.

CIRCULAR LEFT SHIFT: SPECIAL=L1



If R1, bit 0 of the A-register is transferred to bit 15 of the B-register.

CIRCULAR RIGHT SHIFT: SPECIAL=R1



OP
DIV

Required micro-instruction mnemonic fields:

OP	SPECIAL	ALU	STORE	S-BUS
DIV	L1	SUB	B	B

Equivalent micro-instruction binary fields:

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OP				ALU					S-BUS					STORE					SPECIAL				
0	1	0	1	0	0	1	1	0	0	1	0	1	0	0	1	0	1	0	0	0	0	1	0

Meaning: Perform a divide step where the divisor is in the L-register and the 32 bit dividend is in the A- and B-registers (least significant bits in the A-register). This micro-order is repeated (16 times for a full word divisor) by specifying the Special micro-order RPT in the preceding micro-instruction. This performs the successive subtractions required in a divide algorithm.

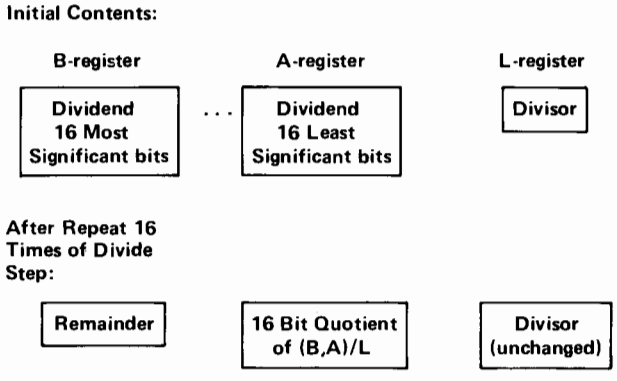
The divide step is executed as follows:

a. Subtract the L-register from the B-register (ALU = B -L).

b. If borrow is required to complete the subtraction, the ALU Carry Out Flag is clear (0). This Carry Out result means that the divisor (L-register) is too big. The ALU result is not stored. The A-register and B-register are left shifted one bit and the divide step is complete.

c. If a borrow is not required to complete the subtraction, the ALU Carry Out Flag is set (1). This Carry Out result means that the divisor is small enough. The result of the subtraction is contained in the ALU and is left shifted one bit and stored back into the B-register. Bit 15 of the A-register shifts into bit 0 of the B-register and bit 0 of the A-register is set to 1 (the Carry Out result). The divide step is complete.

Usage: The base set divide operation is shown in the Basic Instruction Set microprogram in Appendix E at the label = DIV.



OP
ENV

BIT NO.	23	22	21	20
CONTENT	1	0	1	0

Meaning: Enable the overflow test for the current ALU operation.

Usage: To detect an overflow condition (that is, set the Overflow register bit), ENV or ENVE (see below) must be specified as the OP Code of the micro-instruction in which the condition is to be tested. Overflow is set if the S-bus and L-register bits 15 are the same and bit 15 output from the ALU is different.

Caution: Caution is advised in the use of DEC (decrement) or INC (increment) in conjunction with ENV. The L-register is always compared.

OP
ENVE

BIT NO.	23	22	21	20
CONTENT	1	0	1	1

Meaning: Enable the overflow test and the extend test for the current ALU operation.

Usage: To detect an Overflow condition (that is, set the Overflow register bit), ENV (see above) or ENVE must be specified as the OP Code of the micro-instruction. To set the Extend Register as a result of the ALU operation, the ENVE micro-order must be specified as the OP code of the micro-instruction. The Extend Register bit is set if there is a carry generated by the ALU (ALU Carry Out = 1).

OP
LGS

Required micro-instruction mnemonic fields:

OP	SPECIAL	ALU	STORE	S-BUS
LGS	L1 or R1	PASS	B	B

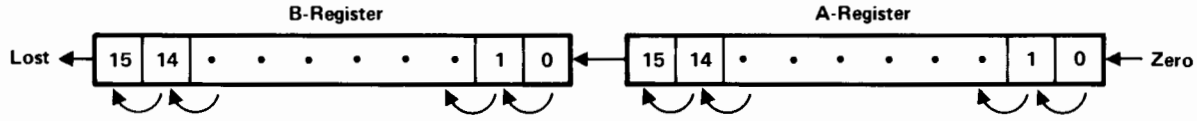
Required micro-instruction binary fields:

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OP				ALU					S-BUS					STORE					SPECIAL				
0	0	1	1	1	1	1	1	1	0	1	0	1	0	0	1	0	1	0	L1 or R1 Code				

Meaning: Perform a single bit Logical Shift of the A- and B-registers combined, with the A-register forming the low order 16 bits. The direction of the shift is specified in the SPECIAL field: L1 for left, R1 for right.

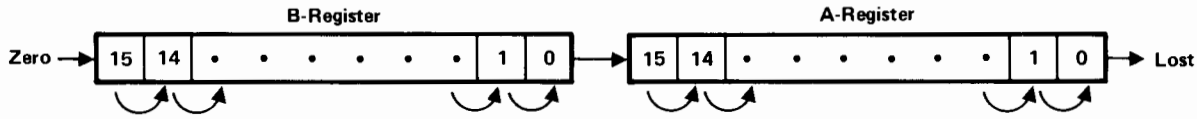
If L1, a 0 is shifted into bit 0 of the A-register and bit 15 of the B-register is lost.

LOGICAL LEFT SHIFT: SPECIAL=L1



If R1, a 0 is shifted into bit 15 of the B-register and bit 0 of the A-register is lost.

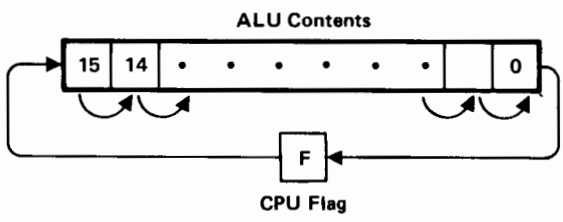
LOGICAL RIGHT SHIFT: SPECIAL=R1



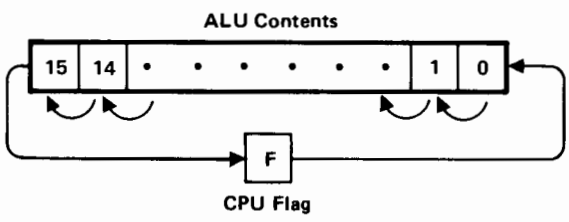
OP	BIT NO.	23	22	21	20
LWF	CONTENT	0	1	1	0

Meaning: Perform a one bit rotational shift of a 17 bit operand in the Rotate/Shifter where bit 17 is formed by the CPU Flag. The rotate moves left one bit, if L1 is the SPECIAL code, or right one bit, if R1 is the SPECIAL code. If neither L1 or R1 are specified, LWF has no effect.

ROTATIONAL RIGHT SHIFT: SPECIAL=R1



ROTATIONAL LEFT SHIFT: SPECIAL=L1



OP
MPY

Required micro-instruction mnemonic fields:

OP	SPECIAL	ALU	STORE	S-BUS
MPY	R1	ADD	B	B

Required micro-instruction binary fields:

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OP				ALU					S-BUS					STORE					SPECIAL				
0	1	0	0	0	1	0	0	1	0	1	0	1	0	0	1	0	1	0	0	0	1	0	0

Meaning: Perform a multiply step where the multiplier is in the L-register and the multiplicand is in the A-register. The multiply step is executed as follows:

a. Test bit 0 of the A-register.

b. If the test bit is a one, the L-register is added to the S-bus (B-register value) in the ALU. The result is shifted right one bit and stored back into the B-register with the ALU Carry Out bit forming bit 15.

c. If the test bit is a zero, the S-bus (B-register value) is shifted right one bit and stored back into the B-register with the ALU Carry Out bit forming bit 15.

d. In either case, the A-register is shifted right and ALU bit 0 fills vacated bit position 15. Bit 0 of the A-register is lost. The multiply step is complete.

Usage: This micro-instruction, repeated 16 times by specifying the SPECIAL code RPT in the preceding micro-instruction, performs the successive additions required in a multiply algorithm. The base set multiply operation is shown in the Basic Instruction Set microprogram in Appendix E at the label =MPY.

Each step of the multiply algorithm effectively multiplies the L-register by the A-register bit that corresponds to the step; that is, step one multiplies the L-register by bit 0 of A-register, step two multiplies the L-register by bit 1 of the A-register, etc. Thus to multiply the L-register by all 16 bits of the A-register, MPY must be repeated 16 times.

Since the B-register goes through successive right shifts and additions as described under "Meaning", the initial contents of the B-register are added to the final result of the multiply algorithm. If the B-register is not zero before the multiply steps are begun, 16 multiply steps will yield the 32 bit result in the B- and A-registers (where the Least Significant Bits (LSB's) are in the A-register):

$$(B,A) = [(AxL) + B]$$

This may be useful in some computational procedures. For example: X(2) = X(1) + (YxZ).

Initial Contents:

B-register	A-register	L-register
Value to be added to the final result	Multiplicand	Multiplier

After Repeating the Multiply Step 16 Times:

(AxL)+B 16 Most Significant bits	(AxL)+B 16 Least Significant bits	Multiplier (unchanged)

OP	BIT NO.	23	22	21	20
READ	CONTENT	1	0	0	1

Meaning: Read data into the T-register from the Main Memory address specified in the M-register. The CPU will freeze until Main Memory is not busy.

Usage: The data must be removed from the T-register two micro-instructions after the READ instruction. Note that the M-register must be loaded (M, PNM, or CM in the Store field) prior to or during the Read micro-instruction. The A- or B-register Addressable Flags (AAF or BAF, respectively) are set, according to data present on the T-bus when the M-register is loaded. Specify INC in the ALU field when the address being stored into the M-register could be a 0 or 1 (A- or B-register addressed). This assures that data is extracted from the proper register when TAB micro-order is used in the S-bus field.

T-bus when M Store is specified	AAF	BAF	Register Referenced By TAB in S-bus or Store Field
1	1	0	A
2	0	1	B
any other value	0	0	T

OP	BIT NO.	23	22	21	20
NOP	CONTENT	0	0	0	0

Meaning: Standard Operation. No operation is specified for the Op Code field.

Usage: This is the default micro-order when the OP Code Field is left blank.

OP	BIT NO.	23	22	21	20
WRTE	CONTENT	0	1	1	1

Meaning: Write data from the T-register into the Main Memory address specified in the M-register. The CPU will freeze until Main Memory is not busy. Two micro-instruction times are required to complete the write.

Usage: The T-register should be loaded during the write instruction and must not be altered by the next sequential micro-instruction; otherwise the Dual Channel Port Controller data-transfers could destroy the data.

4-3. SPECIAL MICRO-ORDERS

SPECIAL	BIT NO.	4	3	2	1	0
CLFL	CONTENT	0	1	0	0	1

Meaning: Clear the CPU Flag.

SPECIAL	BIT NO.	4	3	2	1	0
COV	CONTENT	0	1	1	0	0

Meaning: Clear the Overflow Register bit.

SPECIAL	BIT NO.	4	3	2	1	0
FTCH	CONTENT	0	1	0	1	0

Meaning: Move the Main Memory address contained in the M-register (usually the address of the next macro-instruction to be executed) to the Memory Protect Violation Register. Clear out the Memory Protect Violation flag and reset the Indirect Counter.

Usage: This micro-order must be used during, or one micro-instruction after, the initiation of a READ from the address of the next macro-instruction to be executed. This micro-order must be used if the Memory Protect feature is installed on the computer.

SPECIAL	BIT NO.	4	3	2	1	0
ICNT	CONTENT	1	0	0	1	1

Meaning: Increment the Counter Register by one.

SPECIAL	BIT NO.	4	3	2	1	0
INCI	CONTENT	1	0	1	0	1

Meaning: Increment the Indirect Counter in the Memory Protect Option (if installed) by one.

Usage: Used by microprograms that implement indirect addressing. If INCI is executed three times within the same microprogram, the Interrupt Enable Flag is set to allow the CPU to recognize interrupts. Used to prevent multiple indirect addressing levels from holding off recognition of I/O interrupt requests.

SPECIAL	BIT NO.	4	3	2	1	0
IOFF	CONTENT	0	0	0	0	0

Meaning: Turn off the Interrupt Enable flag to disable recognition of normal interrupts (does not disable memory protect, parity, or power fail interrupts).

Usage: After three occurrences of INCI (see INCI Usage) in the SPECIAL Field, interrupts are again recognized and cannot be disabled until a FTCH micro-order occurs. The ION micro-order is normally used to re-enable interrupt recognition.

IOFF should be used with caution, since holding off interrupts could cause the loss of input and output data.

SPECIAL	BIT NO.	4	3	2	1	0
IOG	CONTENT	1	0	0	1	0

Meaning: Freeze the CPU until time period T2. Then execute the base set I/O macro-instruction that is in the Instruction Register.

Usage: Microprogrammed input and output require cooperation between the I/O Section and microprogram control. Familiarity with the I/O system is mandatory. The user may execute input or output in micro-code by loading a standard base set I/O macro-instruction into the Instruction Register and then executing the following micro-instruction:

JSB IOG — — IOCNTRL

where IOCNTRL = octal 62

See section 3-25 and the following sections for a more detailed description of I/O microprogramming.

SPECIAL	BIT NO.	4	3	2	1	0
ION	CONTENT	0	0	1	0	1

Meaning: Turn the Interrupt Enable flag on to enable recognition of interrupts. Allow the CPU to recognize standard device interrupts until the micro-order IOFF is executed.

Usage: After ION has been executed, the CPU can detect an interrupt from any I/O device in two ways:

a. If a JMP or RTN to location 0 of Control Store (the macro-instruction read and decode routine) is executed and an interrupt is pending or the Run flag is clear, execution is forced to location 4 in Control Store, which is the interrupt handler routine.

b. A test for interrupt pending or Run flag clear can be performed by the executing microprogram by executing INT, NHOI, or RUN in the Jump Condition field.

ION allows interrupts to be recognized. However interrupts are not generated by the interrupt system until a STF 0 I/O control command is executed. Refer to the discussion of the interrupt system in the **HP 21MX Computer Series Reference Manual.**

SPECIAL	BIT NO.	4	3	2	1	0
JTAB	CONTENT	1	1	0	1	1

Meaning: Perform a jump to a location within the Basic Instruction Set microprogram, based on the eight most significant bits (bits 15 through 8) of the Instruction Register. This is accomplished via a table look-up of the address in the main jump table for the basic instruction set (see figure 3-2).

The Save Register is cleared to 0. JTAB overrides the effects of JMP or JSB in the OP code field.

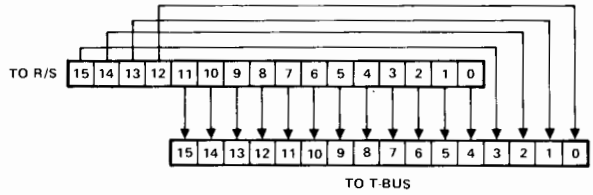
SPECIAL	BIT NO.	4	3	2	1	0
L1	CONTENT	0	0	0	1	0

Meaning: Left one bit command to the Rotate/Shifter.

Usage: See MPY, DIV, CRS, LGS, ARS, LWF. Without one of the previous Op Codes, L1 performs a one bit logical left shift on data leaving the ALU.

SPECIAL	BIT NO.	4	3	2	1	0
L4	CONTENT	0	0	0	1	1

Meaning: Four bit circular left shift command to the Rotate/Shifter (R/S).



Usage: Used in conjunction with the shift and rotate operations.

SPECIAL	BIT NO.	4	3	2	1	0
MPCK	CONTENT	1	0	0	0	1

Meaning: Check the address placed on the S-bus for a memory protect violation.

Usage: An S-BUS micro-order must be used in conjunction with MPCK.

This check should be performed before any write to Main Memory (WRTE OP-code), if the memory protect feature is installed. Refer to section 3-27 for details on use of MPCK with the I/O system.

SPECIAL	BIT NO.	4	3	2	1	0
NOP	CONTENT	0	0	1	1	1

Meaning: No SPECIAL operation is performed.

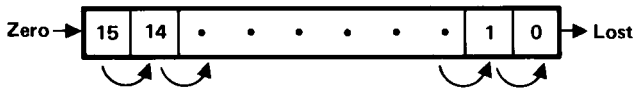
Usage: This is the default operation if none is specified in the SPECIAL field.

SPECIAL	BIT NO.	4	3	2	1	0
RPT	CONTENT	0	1	1	0	1

Meaning: Repeat the following micro-instruction incrementing the Counter Register after each time the repeat is executed. When the lower four bits of the Counter Register are set, execute the following micro-instruction once. The lower four bits of the Counter Register are set at the completion of the repeat sequence. Thus, the repeat is executed the number of times specified in the lower four bits of the Counter Register in two's complement form.

SPECIAL	BIT NO.	4	3	2	1	0
R1	CONTENT	0	0	1	0	0

Meaning: Right one bit command to the Rotate/Shifter.



Usage: Used in conjunction with the shift and rotate instructions. See MPY, DIV, ARS, CRS, LGS, LWF. Without one of the previous micro-orders, a single bit logical right shift is executed.

SPECIAL	BIT NO.	4	3	2	1	0
RTN	CONTENT	1	1	1	1	0

Meaning: Return from subroutine. Jump to the address held in the Save register and clear the Save register.

Usage: No more than one subroutine level is permissable. The second RTN encountered causes a jump to ROM address 0 (the address contained in the Save register) where the macro-instruction pointed to by the P-register is read. RTN overrides the effect of a JMP or JSB in the OP code field.

SPECIAL	BIT NO.	4	3	2	1	0
SHLT	CONTENT	1	0	1	0	0

Meaning: Clear the Run Flag (request a CPU halt).

Usage: The Run Flag is actually cleared at the completion of the micro-instruction following the one specifying SHLT. This micro-order should be used with caution by the microprogrammer. Once the Run Flag is clear, the halt request (SHLT) is detected:

a. when a RTN or JMP to address 0 in Control Store (fetch routine) is executed

b. when the Run Flag is tested by RUN or NHOI Jump Condition micro-order.

SPECIAL	BIT NO.	4	3	2	1	0
SOV	CONTENT	0	1	0	1	1

Meaning: Set the Overflow Register

SPECIAL	BIT NO.	4	3	2	1	0
SRGE	CONTENT	0	1	1	1	0

Meaning: If Instruction Register bit 5 is set, clear the Extend Register bit.

Conventions: This micro-order is used by the Basic Instruction Set that implements the Extend Register instructions.

SPECIAL	BIT NO.	4	3	2	1	0
SRG1	CONTENT	0	0	1	1	0

Meaning: Execute the Shift/Rotate function specified by bits 6 through 9 of the Instruction Register (Shift/Rotate instruction in the first position; see **HP 21MX Computer Series Reference Manual.**) The Shift/Rotate function is performed on the data that leaves the ALU. The function performed in the R/S is determined by IR bits 6 through 9 as follows:

Bits 9 8 7 6	Function Performed In R/S
1000	Arithmetic left shift one bit
1001	Arithmetic right shift one bit
1010	Rotational left shift one bit
1011	Rotational right shift one bit
1100	Arithmetic left shift one bit, clear sign bit 15
1101	Rotational right shift one bit with E-register forming bit 16 (the 17th bit)
1110	Rotational left shift one bit with E-register forming bit 16 (the 17th bit)
1111	Rotational left shift four bits
0xxx	No shift (bits 8, 7, and 6 can have any setting)

SPECIAL	BIT NO.	4	3	2	1	0
SRG2	CONTENT	0	0	0	0	1

Meaning: Execute the Shift/Rotate function specified by bits 0 1, 2 and 4 of the Instruction Register (Shift/Rotate instruction in the second position; see **HP 21MX Computer Series Reference Manual**). The Shift/Rotate function is performed on the data that leaves the ALU. The function performed in the R/S is determined by IR bits 0, 1, 2 and 4.

Bits 4 2 1 0	Function Performed in R/S
1000	Arithmetic left shift one bit
1001	Arithmetic right shift one bit
1010	Rotational left shift one bit
1011	Rotational right shift one bit
1100	Arithmetic left shift one bit, clear sign bit 15
1101	Rotational right shift one bit with E-register forming bit 16 (the 17th bit)
1110	Rotational left shift one bit with E-register forming bit 16 (the 17th bit)
1111	Rotational left shift four bits
0xxx	No shift (bits 8, 7, and 6 can have any setting)

SPECIAL	BIT NO.	4	3	2	1	0
SRUN	CONTENT	1	0	1	1	1

Meaning: Set the Run Flag (remove the CPU halt request).

SPECIAL	BIT NO.	4	3	2	1	0
STFL	CONTENT	0	1	0	0	0

Meaning: Set the CPU flag.

4-4. ALU MICRO-ORDERS

ALU	BIT NO.	19	18	17	16	15
ADD	CONTENT	0	1	0	0	1

Meaning: Add the data placed on the S-bus to the contents of the L-register; the L-register contents are not disturbed; pass the result to R/S.

Usage: The L-register must be loaded in a previous micro-instruction.

ALU	BIT NO.	19	18	17	16	15
AND	CONTENT	1	1	0	1	1

Meaning: Logical and of L-register and S-bus (L•S); the L-register contents are not disturbed; pass the result to R/S.

Usage: The L-register must be loaded in a previous micro-instruction.

ALU	BIT NO.	19	18	17	16	15
CMPL	CONTENT	1	0	1	0	1

Meaning: Ones complement the L-register; pass the result to Rotate/Shifter.

ALU	BIT NO.	19	18	17	16	15
CMPS	CONTENT	1	0	0	0	0

Meaning: Ones complement the data on the S-bus; pass the result to Rotate/Shifter.

ALU	BIT NO.	19	18	17	16	15
DEC	CONTENT	0	1	1	1	1

Meaning: Decrement the data on the S-bus by one; pass the result to the Rotate/Shifter.

ALU	BIT NO.	19	18	17	16	15
INC	CONTENT	0	0	0	0	0

Meaning: Increment the data on the S-bus by one; pass the result to the Rotate/Shifter.

ALU	BIT NO.	19	18	17	16	15
IOR	CONTENT	1	1	1	1	0

Meaning: Logical inclusive or of L-register and S-bus (L+S); L-register contents are not disturbed; pass result to Rotate/Shifter.

ALU	BIT NO.	19	18	17	16	15
NAND	CONTENT	1	0	1	0	0

Meaning: Logical nand of L-register and S-bus ($\overline{L \cdot S}$); pass result to Rotate/Shifter.

ALU	BIT NO.	19	18	17	16	15
NOR	CONTENT	1	0	0	0	1

Meaning: Logical nor of L-register and S-bus ($\overline{L+S}$); pass result to Rotate/Shifter.

ALU	BIT NO.	19	18	17	16	15
NSAL	CONTENT	1	0	0	1	0

Meaning: Logical and of the complement of the S-bus and the L-register ($\overline{S} \cdot L$); pass result to Rotate/Shifter.

ALU	BIT NO.	19	18	17	16	15
NSOL	CONTENT	1	1	0	0	0

Meaning: Logical or of the complement of the S-bus and the L-register ($\overline{S}+L$); pass result to Rotate/Shifter.

ALU	BIT NO.	19	18	17	16	15
ONE	CONTENT	1	1	1	0	0

Meaning: Set all 16 bits (logical one) and pass them to the Rotate/Shifter.

ALU	BIT NO.	19	18	17	16	15
OP1	CONTENT	0	0	0	0	1

Meaning: Perform the following logical function in the ALU with the L-register and the S-bus:

(S+L) plus 1

where "+" means logical function "or".

ALU	BIT NO.	19	18	17	16	15
OP2	CONTENT	0	0	0	1	0

Meaning: Perform the following logical function in the ALU with the L-register and the S-bus:

(S+$\overline{L}$) plus 1

where "+" means logical function "or" and L means the ones complement of the L-register (not L).

ALU	BIT NO.	19	18	17	16	15
OP3	CONTENT	0	0	1	0	0

Meaning: Perform the following logical function in the ALU with the L-register and the S-bus:

S plus ($\overline{S \cdot L}$) plus 1

where "•" means logical function "and" and $\overline{L}$ means the ones complement of the L-register (not L).

ALU	BIT NO.	19	18	17	16	15
OP4	CONTENT	0	0	1	0	1

Meaning: Perform the following logical function in the ALU with the L-register and the S-bus:

(S+L) plus (S•$\overline{L}$) plus 1

where "•" means logical function "and", "+"means logical function "or", and $\overline{L}$ means the ones complement of the L-register (not L).

ALU	BIT NO.	19	18	17	16	15
OP5	CONTENT	0	0	1	1	1

Meaning: Perform the following logical function in the ALU with the L-register and the S-bus:

(S•$\overline{L}$)

where "•" means the logical function "and" and $\overline{L}$ means the ones complement of the L-register (not L).

ALU	BIT NO.	19	18	17	16	15
OP6	CONTENT	0	1	0	0	0

Meaning: Perform the following logical function in the ALU with the L-register and the S-bus:

S plus (S•L)

where "•" means the logical function "and".

ALU	BIT NO.	19	18	17	16	15
OP7	CONTENT	0	1	0	1	0

Meaning: Perform the following logical function in the ALU with the L-register and the S-bus:

(S+$\overline{L}$) plus (S•L)

where "+" means logical function "or", "•" means logical function "and", and L means the ones complement of the L-register (not L).

ALU	BIT NO.	19	18	17	16	15
OP8	CONTENT	0	1	0	1	1

Meaning: Perform the following logical function in the ALU with the L-register and the S-bus:

(S•L) minus 1

where "•" means the logical function "and".

ALU	BIT NO.	19	18	17	16	15
OP9	CONTENT	0	1	1	0	0

Meaning: Perform the following logical function in the ALU with the S-bus:

S plus S

ALU	BIT NO.	19	18	17	16	15
OP10	CONTENT	0	1	1	0	1

Meaning: Perform the following logical function in the ALU with the L-register and the S-bus:

(S+L) plus S

where "+" means the logical function "or".

ALU	BIT NO.	19	18	17	16	15
OP11	CONTENT	0	1	1	1	0

Meaning: Perform the following logical function in the ALU with the L-register and the S-bus:

$(S+\overline{L})$ plus S

where "+" means the logical function "or" and $\overline{L}$ means the complement of the L-register (not L).

ALU	BIT NO.	19	18	17	16	15
PASL	CONTENT	1	1	0	1	0

Meaning: Pass the L-register to the Rotate/Shifter.

ALU	BIT NO.	19	18	17	16	15
PASS	CONTENT	1	1	1	1	1

Meaning: Pass the S-bus data to the Rotate/Shifter.

ALU	BIT NO.	19	18	17	16	15
SANL	CONTENT	1	0	1	1	1

Meaning: Logical and of the S-bus and the complement of the L-register $(S \cdot \overline{L})$; pass the result to the Rotate/Shifter.

ALU	BIT NO.	19	18	17	16	15
SONL	CONTENT	1	1	1	0	1

Meaning: Logical or of the S-bus and the complement of the L-register $(S+\overline{L})$; pass the result to the Rotate/Shifter.

ALU	BIT NO.	19	18	17	16	15
SUB	CONTENT	0	0	1	1	0

Meaning: Subtract the L-register from the S-bus and pass the result to Rotate/Shifter.

ALU	BIT NO.	19	18	17	16	15
XNOR	CONTENT	1	1	0	0	1

Meaning: Logical exclusive nor of the L-register and the S-bus; $(\overline{L \oplus S})$ and pass it to the Rotate/Shifter ($\oplus$ means "exclusive or".)

ALU	BIT NO.	19	18	17	16	15
XOR	CONTENT	1	0	1	1	0

Meaning: Logical exclusive or of the L-register and the S-bus $(L \oplus S)$; pass the result to the Rotate/Shifter ($\oplus$ means "exclusive or".)

ALU	BIT NO.	19	18	17	16	15
ZERO	CONTENT	0	0	0	1	1

Meaning: Pass All zeros to the Rotate/Shifter.

4-5. STORE MICRO-ORDERS

STORE	BIT NO.	9	8	7	6	5
A	CONTENT	0	1	0	1	1

Meaning: Store the data on the T-bus in the A-register.

STORE	BIT NO.	9	8	7	6	5
B	CONTENT	0	1	0	1	0

Meaning: Store the data on the T-bus in the B-register.

STORE	BIT NO.	9	8	7	6	5
CAB	CONTENT	0	0	0	0	1

Meaning: Store the data on the T-bus in the A- or B-register according to the value of IR bit 11:

IR bit 11 set means B-register

IR bit 11 clear means A-register

STORE	BIT NO.	9	8	7	6	5
CM	CONTENT	0	1	1	0	1

Meaning: Store the data on the S-bus in the M-register, if the IR holds any Memory Reference instruction except a direct jump (JMP). Refer to the **HP 21MX Computer Series Reference Manual**, for a description of the Memory Reference instructions.

AAF or BAF is set as described under Usage for the M Store micro-order, whether or not the IR holds a Memory Reference instruction.

STORE	BIT NO.	9	8	7	6	5
CNTR	CONTENT	0	0	1	0	1

Meaning: Store the lower eight bits of the S-bus (bits 0-7) in the Counter Register.

STORE	BIT NO.	9	8	7	6	5
DSPI	CONTENT	0	0	1	1	1

Meaning: Store the lower six bits of the S-bus in the Display Indicator on the front panel.

Display Indicator Bit	5	4	3	2	1	0
Register Displayed	S	P	T	M	B	A

Usage: The six indicators on the front panel, labelled A, B, M, T, P and S are lit according to the bit(s) cleared in the Display Indicator. At power-up all bits are set until programmatically changed.

STORE	BIT NO.	9	8	7	6	5
DSPL	CONTENT	0	0	1	1	0

Meaning: Store the data on the S-bus in the Display Register on the Front Panel.

STORE	BIT NO.	9	8	7	6	5
IOO	CONTENT	0	0	1	0	0

Meaning: Direct the S-bus onto the I/O-bus.

Usage: This micro-order when used must be in the second and third instructions (T3 and T4) after IOG Special micro-order. See section 3-25 and the following sections for a description of I/O microprogramming.

STORE	BIT NO.	9	8	7	6	5
IR	CONTENT	0	1	0	0	0

Meaning: Store the data on the S-bus in the Instruction Register. Record the type of macro-instruction stored there in the Memory Protect hardware for use in determining error conditions during Instruction Register execution. See sections 3-28 and 3-34 for a description of Interfacing With Memory Protect feature.

STORE	BIT NO.	9	8	7	6	5
L	CONTENT	0	0	0	1	1

Meaning: Store the data on the S-bus in the L-register (Latch).

STORE	BIT NO.	9	8	7	6	5
M	CONTENT	0	1	0	0	1

Meaning: Store the data on the S-bus in the M-register.

Usage: An ALU micro-order (for example, INC) should also be specified in the micro-instruction. This will activate an A- or B-register addressable test. If bits 14 through 0 on the T-bus equal 1 or 2, the AAF or BAF, respectively, will be set. The M-register may be stored into immediately after a READ or WRTE Op micro-order.

STORE	BIT NO.	9	8	7	6	5
NOP	CONTENT	0	1	1	1	1

Meaning: No store operation is performed; this is the default micro-order when the Store field is left blank.

STORE	BIT NO.	9	8	7	6	5
P	CONTENT	1	1	1	1	0

Meaning: Store the data on the T-bus in the P-register (Program Address Register).

STORE	BIT NO.	9	8	7	6	5
PNM	CONTENT	0	1	1	1	0

Meaning: Store the data on the T-bus in the P-register (Program Address Register), and the data on the S-bus into the M-register (Memory Address Register).

Usage: Useful in microprograms which perform multiword READ operations from Main Memory, where the P-register points to the address in Main Memory to be read. In a single micro-instruction the microprogram can store P into the M-register via the S-bus and then increment P via the T-bus. An example of such an application is the following:

```
READ - -    INC    PNM    P
```

The A- or B-register addressable test is activated. See Usage under M micro-order, above.

STORE	BIT NO.	9	8	7	6	5
S	CONTENT	1	1	1	1	1

Meaning: Store the data on the T-bus in the S-register (Switch Register).

STORE	THRU	STORE
S1		S12

BIT NO.	9	8	7	6	5
CONTENT	1	n	n	n	n

nnnn is binary representation of decimal number 0 + 11

Meaning: Store the data on the T-bus in the indicated Scratch Pad Register S1 to S12.

STORE	BIT NO.	9	8	7	6	5
T	CONTENT	0	0	0	1	0

Meaning: Store the data on the S-bus in the T-register (Memory Data Register).

Usage: This micro-order should occur concurrently when a WRTE micro-order is used. The T-register is internal to the Memory System. It must not be used as a working register.

STORE	BIT NO.	9	8	7	6	5
TAB	CONTENT	0	0	0	0	0

Meaning: Store the data on the T-bus in the A-register if the AAF (A addressable Flag) is set; store the data on the T-bus in the B-register if the BAF (B addressable Flag) is set; store the data on the S-bus into the T-register (Memory Data Register) if neither AAF nor BAF is set.

Usage: Same as T micro-order.

STORE	BIT NO.	9	8	7	6	5
X	CONTENT	1	1	1	0	0

Meaning: Store the data on the T-bus in the X-register.

STORE	BIT NO.	9	8	7	6	5
Y	CONTENT	1	1	1	0	1

Meaning: Store the data on the T-bus in the Y-register.

4-6. S-BUS MICRO-ORDERS

S-BUS	BIT NO.	14	13	12	11	10
A	CONTENT	0	1	0	1	1

Meaning: Direct the data in the A-register onto the S-bus.

S-BUS	BIT NO.	14	13	12	11	10
ADR	CONTENT	0	1	0	0	0

Meaning: An address is formed on the S-bus using IR bits 0-9 and M-register bits 10-14; if IR bit 10 is clear, bits 10-14 of the address formed on the S-bus are clear. Bit 15 is always clear. IR bit 10 is the zero page/current page flag.

S-BUS	BIT NO.	14	13	12	11	10
B	CONTENT	0	1	0	1	0

Meaning: Direct the contents of the B-register onto the S-bus.

S-BUS	BIT NO.	14	13	12	11	10
CAB	CONTENT	0	0	0	0	1

Meaning: Direct the contents of the A- or B-register onto the S-bus according to the value of IR bit 11:

IR bit 11 set means B-register

IR bit 11 clear means A-register

S-BUS	BIT NO.	14	13	12	11	10
CIR	CONTENT	0	0	0	1	1

Meaning: At I/O time T6 place the contents of the Central Interrupt Register onto the S-bus and generate an IAK (Interrupt Acknowledge) signal to the I/O device. (See section 3-33 for CIR description in relation to Interrupt Handling).

Usage: This micro-order must be used after detection of an I/O interrupt to determine the select code of the interrupting device and to acknowledge that the interrupt is being serviced.

S-BUS	BIT NO.	14	13	12	11	10
CNTR	CONTENT	0	0	1	0	1

Meaning: Direct the contents of the Counter Register onto the S-bus. The 8 bit Counter Register is placed onto the low 8 bits of the S-bus; the upper 8 bits are set to ones.

S-BUS	BIT NO.	14	13	12	11	10
DSPI	CONTENT	0	0	1	1	1

Meaning: Direct the six bits of the Display Indicator from the Front Panel to the S-bus.

Usage: See DSPI Store field definition for Display Indicator bit significance.

S-BUS	BIT NO.	14	13	12	11	10
DSPL	CONTENT	0	0	1	1	0

Meaning: Direct the contents of the Front Panel Display Register onto the S-bus.

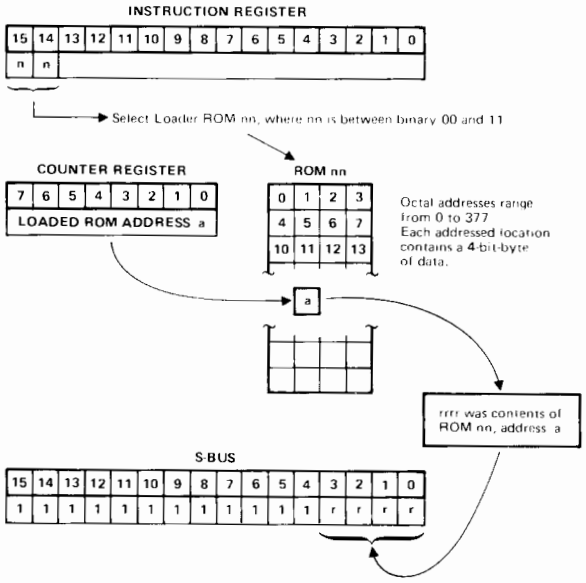
S-BUS	BIT NO.	14	13	12	11	10
IOI	CONTENT	0	0	1	0	0

Meaning: Direct the I/O bus onto the S-bus. (See section 3-25, Microprogramming Input and Output Functions.)

Usage: This is used to transfer data from an I/O device to the S-bus. When not in use, the I/O bus is all zeros. However, do not try to use the I/O bus for a source of zero data, since it is used by the Dual Channel Port Controller at indeterminate times.

S-BUS	BIT NO.	14	13	12	11	10
LDR	CONTENT	0	1	1	0	0

Meaning: Place one 4-bit-byte from a Loader ROM on the S-bus. The 4-bit-byte address is contained in the Counter Register. Determination of which Loader ROM, of the four Loader ROMs available, is specified by bits 15 and 14 in the Instruction Register.



Usage: See sample microprogram in section 3-41 for an illustration of the use of the LDR micro-order.

S-BUS	BIT NO.	14	13	12	11	10
M	CONTENT	0	1	0	0	1

Meaning: Direct the 15 bit contents of the M-register onto the S-bus. Bit 15 of the S-bus is cleared.

S-BUS	BIT NO.	14	13	12	11	10
NOP	CONTENT	0	1	1	1	1

Meaning: The S-bus holds all ones.

Usage: This is the default micro-order when the S-bus field is left blank.

S-BUS	BIT NO.	14	13	12	11	10
P	CONTENT	1	1	1	1	0

Meaning: Direct the contents of the P-register onto the S-bus.

S-BUS	BIT NO.	14	13	12	11	10
S	CONTENT	1	1	1	1	1

Meaning: Place the contents of the S-register (Front Panel Switch Register) onto the S-bus.

S-BUS		S-BUS	BIT NO.	14	13	12	11	10
S1	THRU	S12	CONTENT	1	n	n	n	n

nnnn is binary representation of decimal numbers 0 to 11

Meaning: Place the contents of the indicated Scratch Pad Register S1 to S12 onto the S-bus.

S-BUS	BIT NO.	14	13	12	11	10
T	CONTENT	0	0	0	1	0

Meaning: Direct the contents of the T-register (Memory Data Register) onto the S-bus.

Usage: Data in the T-register that resulted from a READ operation must be removed within two micro-instructions afer the READ or the Dual Channel Port Controller could alter the data.

S-BUS	BIT NO.	14	13	12	11	10
TAB	CONTENT	0	0	0	0	0

Meaning: Direct the contents of the T-register (Memory Data Register) onto the S-bus if neither AAF (A addressable Flag) nor the BAF (B addressable Flag) is set; read the A-register onto the S-bus, if the AAF is set; read the B-register onto the S-bus if the BAF is set.

Usage: See T-register Usage description.

S-BUS	BIT NO.	14	13	12	11	10
X	CONTENT	1	1	1	0	0

Meaning: Direct the contents of the X-register onto the S-bus.

S-BUS	BIT NO.	14	13	12	11	10
Y	CONTENT	1	1	1	0	1

Meaning: Direct the contents of the Y-register onto the S-bus.

4-7. WORD TYPE 2 — IMMEDIATE DATA

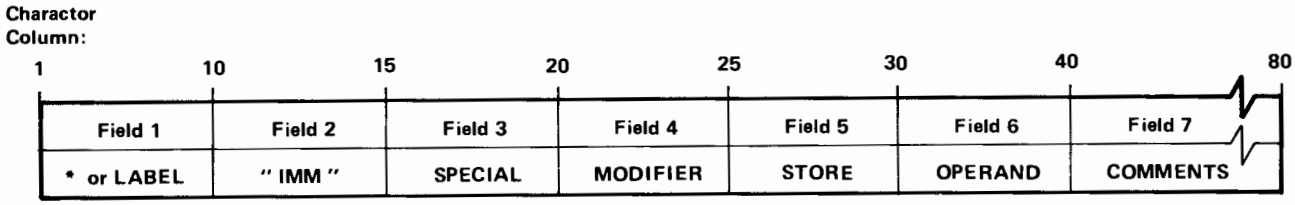


Figure 4-3. Word Type 2 Micro-assembler Mnemonic Format

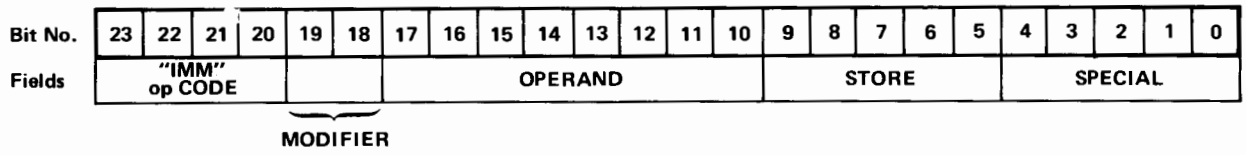


Figure 4-4. Word Type 2 Binary Format

There are five micro-order classifications in Word Type 2:

- "IMM" — OP Code specifying Word Type 2.
- SPECIAL — Special operations and modifiers.
- MODIFIER — A special modifier for the Immediate Operation.
- STORE — Destination of the data.
- OPERAND — Binary data that is to be placed on the S-bus.

The STORE and SPECIAL micro-orders applicable to Word Type 2 are exactly the same as those defined for Word Type 1. Consequently, only the other three micro-order groups are defined in the following sections. The "IMM" and MODIFIER micro-order groups are defined by the mnemonic, by its binary equivalent, and finally, by the meaning.

4-8. "IMM" MICRO-ORDER

"IMM"	BIT NO.	23	22	21	20
IMM	CONTENT	1	1	1	0

Meaning: Place 16 bits onto the S-bus consisting of the 8 bit binary OPERAND and another 8 bits of all ones. Determination of which 8 bits of the S-bus receive the OPERAND and which 8 bits receive all ones is made by the MODIFIER.

4-9. MODIFIER MICRO-ORDERS (BITS 19 AND 18 OF THE MICRO-INSTRUCTION)

Bit 19 Set: specifies complement the S-bus data in the ALU.

Bit 19 Clear: specifies pass the S-bus data through the ALU.

Bit 18 Set: specifies OPERAND goes in bits 7-0 of the S-bus.

Bit 18 Clear: specifies OPERAND goes in bits 15-8 of the S-bus.

MODIFIER	BIT NO.	19	18
CMHI	CONTENT	1	0

Meaning: The 16 bits received by the S-bus consist of the following:

Bits 15-8 = OPERAND

Bits 7-0 = all ones

The S-bus is then complemented as it passes through the ALU.

	BIT NO.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
S-Bus	CONTENT	OPERAND								1	1	1	1	1	1	1	1
Out of ALU	BIT NO.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	CONTENT	OPERAND Complemented								0	0	0	0	0	0	0	0

MODIFIER	BIT NO.	19	18
CMLO	CONTENT	1	1

Meaning: The 16 bits received by the S-bus consist of the following:

Bits 15-8 = all ones
Bits 7-0 = OPERAND

The S-bus is then complemented as it passes through the ALU.

	BIT NO.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
S-Bus	CONTENT	1	1	1	1	1	1	1	1	OPERAND							
Out of ALU	BIT NO.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	CONTENT	0	0	0	0	0	0	0	0	OPERAND Complemented							

MODIFIER	BIT NO.	19	18
HIGH	CONTENT	0	0

Meaning: The 16 bits received by the S-bus consist of the following:

Bits 15-8 = OPERAND
Bits 7-0 = all ones

The S-bus is then passed through the ALU without modification.

	BIT NO.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
S-Bus and Out of ALU	CONTENT	OPERAND								1	1	1	1	1	1	1	1

MODIFIER	BIT NO.	19	18
LOW	CONTENT	0	1

Meaning: The 16 bits received by the S-bus consist of the following:

Bits 15-8 = all ones
Bits 7-0 = OPERAND

The S-bus is then passed through the ALU without modification.

S-Bus and Out of ALU	BIT NO.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	CONTENT	1	1	1	1	1	1	1	1	OPERAND							

4-10. OPERAND MICRO-ORDER

OPERAND	BIT NO.	19	18	17	16	15	14	13	12	11	10
Integer	CONTENT	Binary Integer Equivalent									

The Integer can be an octal number or decimal number:

- Decimal number in range 0 to 255.
- Octal number in range 0 to 377, followed by "B".

Examples:

117B, 117, 198, 5, 10B

4-11. WORD TYPE 3 — CONDITIONAL JUMP

Charactor Column :

1	10	15	20	25	30	40	80
Field 1	Field 2	Field 3	Field 4	Field 5	Field 6	Field 7	
* or LABEL	" JMP "	" CNDX "	CONDITION	JUMP SENSE	OPERAND	COMMENTS	

Figure 4-5. Word Type 3 Micro-assembler Mnemonic Format

Bit No.	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Fields	"JM op CODE				CONDITION					JUMP SENSE	OPERAND									"CNDX" SPECIAL CODE				

Figure 4-6. Word Type 3 Binary Format

There are five micro-order classifications in Word Type 3:

- "JMP" — Op Code used in conjunction with "CNDX" specifies Word Type 3, a conditional jump.
- "CNDX" — SPECIAL Code specifying Word Type 3.
- CONDITION — Condition that must be satisfied before jump is executed.
- JUMP SENSE — Optional code to invert the jump condition.
- OPERAND — Target address of jump.

All micro-order groups, except the OPERAND, are defined by the mnemonic, its binary equivalent, meaning, and, where necessary, by conventions in their use.

4-12. "JMP" MICRO-ORDER

"JMP"	BIT NO.	23	22	21	20
JMP	CONTENT	1	1	0	1

Meaning: Used in conjunction with the SPECIAL Code "CNDX", the CONDITION code specifies the condition under which a jump to the address specified in the OPERAND will take place. If the JUMP SENSE code "RJS" is specified, the CONDITION code specifies the condition under which no jump will take place.

4-13. "CNDX" MICRO-ORDER

"CNDX"	BIT NO.	4	3	2	1	0
CNDX	CONTENT	1	1	0	0	1

Meaning: Used in conjunction with the Op code "JMP", this micro-order specifies a conditional jump and Word Type 3.

4-14. CONDITION MICRO-ORDERS

The ALU and T-bus condition flags are set after each Word Type 1 or 2 micro-instruction. They are not changed during JMP or JSB micro-instructions (Word Types 3 and 4). Thus, several different jump tests can be made without losing the flag results.

CONDITION	BIT NO.	19	18	17	16	15
AL0	CONTENT	0	0	0	1	1

Meaning: Bit 0 of the last output from the ALU was set (tested before the Rotate/Shifter) by the last Word Type 1 or 2 micro-instruction.

CONDITION	BIT NO.	19	18	17	16	15
AL15	CONTENT	0	0	1	0	0

Meaning: Bit 15 of the last output from the ALU was set (tested before the Rotate/Shifter) by the last Word Type 1 or 2 micro-instruction.

CONDITION	BIT NO.	19	18	17	16	15
ASGN	CONTENT	0	1	1	1	0

Meaning: Alter/skip macro-instruction condition is not satisfied.

CONDITION	BIT NO.	19	18	17	16	15
CNT4	CONTENT	1	1	1	1	0

Meaning: The right (least significant) 4 bits of the Counter Register are all ones.

CONDITION	BIT NO.	19	18	17	16	15
CNT8	CONTENT	0	0	1	1	0

Meaning: All eight bits of the Counter Register are ones.

CONDITION	BIT NO.	19	18	17	16	15
COUT	CONTENT	0	0	0	1	0

Meaning: The ALU Carry Out Flag bit was set by the last ALU operation (tested before the Rotate/Shifter) of the last Word Type 1 or 2 micro-instruction.

CONDITION	BIT NO.	19	18	17	16	15
E	CONTENT	0	1	0	0	1

Meaning: The Extend Register bit is set.

CONDITION	BIT NO.	19	18	17	16	15
FLAG	CONTENT	0	1	0	0	0

Meaning: The CPU FLAG bit is set.

CONDITION	BIT NO.	19	18	17	16	15
FPSP	CONTENT	0	0	1	1	1

Meaning: A special signal is present issued by certain non-standard CPU Front Panels.

CONDITION	BIT NO.	19	18	17	16	15
INT	CONTENT	1	1	0	1	0

Meaning: An Interrupt is pending.

CONDITION	BIT NO.	19	18	17	16	15
IR2	CONTENT	0	1	1	1	1

Meaning: Instruction Register bit 2 is set.

CONDITION	BIT NO.	19	18	17	16	15
NDEC	CONTENT	1	0	0	1	1

Meaning: The "DEC M" (Decrement M-register) button on the Front Panel was **not** actuated.

CONDITION	BIT NO.	19	18	17	16	15
NHOI	CONTENT	0	1	1	0	0

Meaning: The RUN/HALT switch on the Front Panel is set to "Run" and there is no interrupt pending (i.e. no halt and no interrupt).

Usage: This micro-order is recommended for use in long microprograms. (75 microseconds or longer is the criterion used by Hewlett-Packard produced microprograms.) This is necessary since microprograms cannot be interrupted. A pending interrupt or halt condition is not detected unless a specific test is made for them.

CONDITION	BIT NO .	19	18	17	16	15
NINC	CONTENT	1	0	0	1	0

Meaning: The "INC M" (Increment M-register) button on the Front Panel was not actuated.

CONDITION	BIT NO.	19	18	17	16	15
NLDR	CONTENT	1	0	0	0	0

Meaning: The "IBL" (loader) button on the Front Panel was not actuated.

CONDITION	BIT NO.	19	18	17	16	15
NLT	CONTENT	1	0	1	0	1

Meaning: The "←" REGISTER SELECT LEFT button on the Front Panel was not actuated.

CONDITION	BIT NO.	19	18	17	16	15
NMLS	CONTENT	0	0	1	0	1

Meaning: Memory was not lost as a result of the last power down or power failure.

CONDITION	BIT NO.	19	18	17	16	15
NOP	CONTENT	1	1	1	0	1

Meaning: No condition test is made; no jump occurs.

Usage: This is the default micro-order if none is specified in the condition field.

CONDITION	BIT NO.	19	18	17	16	15
NRST	CONTENT	1	0	1	1	1

Meaning: The DISPLAY button on the Front Panel was not actuated.

CONDITION	BIT NO.	19	18	17	16	15
NRT	CONTENT	1	0	1	0	0

Meaning: The "→" REGISTER SELECT RIGHT button on Front Panel was not selected.

CONDITION	BIT NO.	19	18	17	16	15
NSFP	CONTENT	1	1	0	0	1

Meaning: A standard Front Panel is not installed on the CPU.

CONDITION	BIT NO.	19	18	17	16	15
NSNG	CONTENT	1	0	0	0	1

Meaning: The INSTR STEP (Instruction Step) button on the Front Panel was not actuated.

CONDITION	BIT NO .	19	18	17	16	15
NSTB	CONTENT	1	1	0	0	0

Meaning: None of the following Front Panel buttons were actuated:

INSTR STEP (Instruction Step)
"→" REGISTER SELECT RIGHT
"←" REGISTER SELECT LEFT
DISPLAY
IBL (Binary Loader)
INC M (Increment M-register)
DEC M (Decrement M-register)
STORE
RUN
PRESET

CONDITION	BIT NO.	19	18	17	16	15
NSTR	CONTENT	1	0	1	1	0

Meaning: The STORE button on the Front Panel was not actuated.

CONDITION	BIT NO.	19	18	17	16	15
ONES	CONTENT	0	0	0	0	1

Meaning: All 16 bits of the last output from the ALU were set (tested before Rotate/Shifter) as a result of the last Word Type 1 or 2 micro-instruction.

CONDITION	BIT NO.	19	18	17	16	15
OVFL	CONTENT	0	1	0	1	0

Meaning: The Overflow Register bit is set.

CONDITION	BIT NO.	19	18	17	16	15
RUN	CONTENT	0	1	0	1	1

Meaning: The CPU is in RUN mode (the Front Panel RUN flag is set).

CONDITION	BIT NO.	19	18	17	16	15
RUNE	CONTENT	1	1	1	0	0

Meaning: The four position STANDBY/OPERATE/LOCK/R switch on the Front Panel is not in the LOCK position.

CONDITION	BIT NO.	19	18	17	16	15
SKPF	CONTENT	0	1	1	0	1

Meaning: The I/O signal SFS is present (I/O time is T3 to T5) and the addressed I/O device Flag is set or the I/O signal SFC is present (I/O time is T3 to T5) and the addressed I/O device Flag is clear.

Usage: See section 3-25, Microprogramming Input and Output Functions, for the use of the micro-order SKPF.

CONDITION	BIT NO.	19	18	17	16	15
SRGL	CONTENT	1	1	0	1	1

Meaning: Bit 3 of the Instruction Register is set and bit 0 of the last output from the ALU was cleared as a result of the last Word Type 1 or 2 micro-instruction.

Usage: This micro-order is used by the Basic Instruction Set microprogram which implements the SLA and SLB macro-instructions of the Shift/rotate Group.

CONDITION	BIT NO.	19	18	17	16	15
TBZ	CONTENT	0	0	0	0	0

Meaning: The last output from the Rotate/Shifter onto the T-bus was equal to zero as a result of the last Word Type 1 or 2 micro-instruction.

4-15. JUMP SENSE MICRO-ORDER

JUMP SENSE	BIT NO.	14
RJS	CONTENT	0

Meaning: Perform the jump, if the jump condition is not met. The CONDITION micro-order specifies the condition under which a jump can take place; the RJS micro-order in effect reverses the sense of the jump. For example, if a conditional jump is specified if the Flag bit is set (jump if Flag bit set), the RJS micro-order will reverse the condition so that the jump occurs if the Flag bit is not set.

4-16. OPERAND MICRO-ORDER

OPERAND
An Address

BIT NO.	13	12	11	10	9	8	7	6	5
CONTENT	Binary Address Equivalent								

The address can be an octal, decimal or computed number:

Decimal number, d, in the range 0 to 511

Octal number, kB, in the range 0B to 777B, where the B signifies octal.

Computed number, c, which is within the decimal or octal range, according to whether it is computed from octal or decimal values, of the form:

a. *+kB
b. *−kB
c. *+d
d. *−d
e. LABEL+kB
f. LABEL−kB
g. LABEL+d
h. LABEL−d
i. LABEL

where * means "this address" and LABEL means a micro-instruction or pseudo-instruction label that is defined elsewhere in the microprogram.

The target address of the jump is not relative and must be within the current 1000 octal locations (two modules). The complete absolute address must be specified. For example, if a conditional jump micro-instruction is within Control Store addresses 3000 and 3777, no target address may be outside the range 3000 to 3777. A target address of 3377B would initiate a jump to the octal address 3377.

Examples:

1005, 2632, 2632B, START, START−11B, END−11

4-17. WORD TYPE 4 — UNCONDITIONAL JUMP

Character Column:

	1	10	15	20	25	30	40	80
Fields:	Field 1	Field 2	Field 3	Field 4	Field 5	Field 6	Field 7	
Content:	* or LABEL	"JMP" or "JSB"	JUMP MODIFIER	(blank)	(blank)	OPERAND	COMMENTS	

Figure 4-7. Word Type 4 Micro-assembler Mnemonic Format

Bit No:	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Fields:	" JMP" or "JSB" Op Code				(zero)			binary OPERAND												JUMP MODIFIER				

Figure 4-8. Word Type 4 Binary Format

Word Type 4 consists of three micro-order classifications:

- "JMP" or "JSB" — Operation, code used in conjunction with the JUMP MODIFIER, specifies Word Type 4, an unconditional jump or subroutine jump.
- JUMP MODIFIER — Specifies modification to the OPERAND jump address.
- OPERAND — Target address of jump, prior to any modification.

Micro-orders, except the OPERAND, are defined by the mnemonic, binary equivalent, meaning, and, where necessary, by conventions in their use.

4-18. "JMP" AND "JSB" MICRO-ORDERS

"JMP" or "JSB"	BIT NO.	23	22	21	20
JMP	CONTENT	1	1	0	1

Meaning: Jump unconditionally to the address specified in the OPERAND, modified according to the JUMP MODIFIER micro-order.

"JMP" or "JSB"	BIT NO.	23	22	21	20
JSB	CONTENT	1	1	0	0

Meaning: Perform a subroutine jump unconditionally to the address specified in the OPERAND, modified according to the JUMP MODIFIER micro-order. The return address is stored in the Save register and recalled by the RTN micro-order (see section 4-3, SPECIAL Micro-orders for RTN definition).

4-19. JUMP MODIFIER MICRO-ORDERS

JUMP MODIFIER	BIT NO.	4	3	2	1	0
IOFF	CONTENT	0	0	0	0	0

Meaning: Disable recognition of normal interrupts (does not disable memory protect, parity, or power fail interrupts). Perform an unconditional jump. No modification is made to the jump OPERAND.

JUMP MODIFIER	BIT NO.	4	3	2	1	0
IOG	CONTENT	1	0	0	1	0

Meaning: Freeze the CPU until time period T2. Execute the I/O function according to the base set I/O macro-instruction that is in the Instruction Register. Perform the JMP or JSB modifying OPERAND bits 2 and 3 according to the I/O instruction jump table (bits 6, 7, and 8 of the I/O macro-instruction in the Instruction Register actually determine the OPERAND address modification):

IR Contains I/O Macro-instruction	IR Bits 8 7 6	OPERAND Bits 3 & 2 Replaced By:
MIA or MIB	1 0 0	1 1
LIA or LIB	1 0 1	1 0
OTA or OTB	1 1 0	0 1
HLT	0 0 0	0 0
CLO or CLF	0 0 1	0 0
STO or STF	0 0 1	0 0
SFC or SOC	0 1 0	0 0
SFS or SOS	0 1 1	0 0
STC or CLC	1 1 1	0 0

See section 3-25 and those following for a more complete description of the use of the IOG micro-order.

JUMP MODIFIER	BIT NO.	4	3	2	1	0
JEAU	CONTENT	1	1	1	1	1

Meaning: Enable the EAU jump table. According to the particular EAU macro-instruction held in the Instruction Register, the least significant three bits (0-2) of the OPERAND are replaced by EAU jump table bits (bits 4-9 and 11 of the Instruction Register actually determine the OPERAND address modification):

EAU Macro-instruction	Three LSB's of Address
RRR	000
ASR	001
LSR	010
(not used)	011
RRL	100
ASL	101
LSL	110
MPY	111

JUMP MODIFIER	BIT NO.	4	3	2	1	0
JIO	CONTENT	1	1	0	1	0

Meaning: Perform the JMP or JSB modifying OPERAND bits 2 and 3 according to the I/O instruction jump table (bits 6, 7, and 8 of the I/O macro-instruction in the Instruction Register actually determine the OPERAND address modification):

IR Contains I/O Macro-instruction	IR Bits 8 7 6	OPERAND Bits 3 & 2 Replaced By:
MIA or MIB	1 0 0	1 1
LIA or LIB	1 0 1	1 0
OTA or OTB	1 1 0	0 1
HLT	0 0 0	0 0
CLO or CLF	0 0 1	0 0
STO or STF	0 0 1	0 0
SFC or SOC	0 1 0	0 0
SFS or SOS	0 1 1	0 0
STC or CLC	1 1 1	0 0

JUMP MODIFIER	BIT NO.	4	3	2	1	0
JTAB	CONTENT	1	1	0	1	0

Meaning: Perform a jump to a location within the Basic Instruction Set microprogram based on the eight most significant bits of the Instruction Register. This is accomplished via a table look up of the address in the Main Jump Table for the basic instruction set. This micro-order is executed independently of word types; hence JMP or JSB need not be specified.

JUMP MODIFIER	BIT NO.	4	3	2	1	0
J30	CONTENT	1	1	1	0	1

Meaning: Replace the four Least Significant Bits of the OPERAND with bits 3 through 0 of the Instruction Register.

JUMP MODIFIER	BIT NO.	4	3	2	1	0
J74	CONTENT	1	1	1	0	0

Meaning: Replace the four Least Significant Bits of the OPERAND with bits 7 through 4 of the Instruction Register.

JUMP MODIFIER	BIT NO.	4	3	2	1	0
RTN	CONTENT	1	1	1	1	0

Meaning: Return to the address stored in the Save Register as a result of a subroutine jump (JSB); if the Save Register is equal to zero (no subroutine is active), return to address 0 of Control Store to initiate the reading of the next macro-instruction from Main Memory.

JUMP MODIFIER	BIT NO.	4	3	2	1	0
STFL	CONTENT	0	1	0	0	0

Meaning: Set the CPU Flag and then perform the JMP or JSB to the OPERAND address. No modification is made to the OPERAND address.

JUMP MODIFIER	BIT NO.	4	3	2	1	0
UNCD	CONTENT	1	1	0	0	0

Meaning: Perform the JMP or JSB to the OPERAND address. No modification is made to the OPERAND address.

Usage: This is the default micro-order if no JUMP MODIFIER is specified.

4-20. THE OPERAND MICRO-ORDER

OPERAND
An Address

BIT NO.	16	15	• • • • • • • • •	6	5
CONTENT	Binary Address Equivalent				

The ADDRESS can be a decimal, octal or computed number:

Decimal number, d, in the range 0 to 4095

Octal number, kB, in the range 0B to 7777B where B signifies octal

Computed number, c, which is within the decimal or octal range, according to whether it is computed from octal or decimal values, of the form:

a. *+kB
b. *−kB
c. *+d
d. *−d
e. LABEL+kB
f. LABEL−kB
g. LABEL+d
h. LABEL−d
i. LABEL

where * means "this address" and LABEL means a micro-instruction label that is defined elsewhere in the microprogram.

Examples:

*+11B, *+9, HERE+5, START

4-21. PSEUDO INSTRUCTIONS

There are five pseudo instructions recognized by the micro-assembler: DEF, EQU, ONES, SKP, and Zeroes.

4-22. DEF

The DEF statement creates a 24 bit micro-instruction word in ROM the contents of which is a 12 bit binary address defined by "ADDRESS" in the micro-assembler input record (Field 6). The binary address is associated in the microprogram with the optional LABEL, if defined.

The ADDRESS can be a decimal, octal or computed number:

Decimal number, d, in the range 0 to 4095

Octal number, kB, in the range 0B to 7777B, where B signifies octal

Computed number, c, which is within the decimal or octal range, according to whether it is computed from octal or decimal values, of the form:

a. *+kB
b. *−kB
c. *+d
d. *−d
e. LABEL+kB
f. LABEL−kB
g. LABEL+d
h. LABEL−d
i. LABEL

where * means "this address" and LABEL means a micro-instruction label that is defined elsewhere in the microprogram.

Examples of DEF statements:

Character Column:	1	10	30
Fields:	Field 1	Field 2	Field 6
Content:	AD1	DEF DEF DEF	SRF+150 ASGNOP 416B

Character Column:	1	10	15	20	25	30	40	80
Fields:	Field 1	Field 2	Fields 3-5			Field 6	Field 7	
Content:	LABEL (optional)	"DEF"	(blank)			ADDRESS	COMMENTS	

4-23. EQU

Character Column:	1	10	15	20	25	30	40	80
Fields:	Field 1	Field 2	Field 3	Field 4	Field 5	Field 6	Field 7	
Content:	LABEL	"EQU"	(blank)	(blank)	(blank)	ADDRESS	COMMENTS	

The EQU statement associates the stated LABEL with a 12 bit address. This statement does not result in an address being stored in ROM. The ADDRESS can be a decimal, octal or computed number:

Decimal number, d, in the range 0 to 4095

Octal number, kB, in the range 0B to 7777B, where B signifies octal

Computed number, c, which is within the decimal or octal range, according to whether it is computed from octal or decimal values, of the form:

a. *+kB
b. *-kB
c. *+d
d. *-d
e. LABEL+kB
f. LABEL-kB
g. LABEL+d
h. LABEL-d
i. LABEL

where * means "this address" and LABEL means a micro-instruction label that is defined in the micro-program before this statement.

Examples of EQU statements:

Character Column:	1	10	30
Fields:	Field 1	Field 2	Field 6
Content:	HALT	EQU	400B
	RELO	EQU	6000B
	START	EQU	RELO

4-24. ONES

Character Column:	1	10	15	20	25	30	40	80
Fields:	Field 1	Field 2	Field 3	Field 4	Field 5	Field 6	Field 7	
Content:	LABEL	"ONES"	(blank)	(blank)	(blank)	(blank)	COMMENTS	

The ONES statement creates a 24 bit micro-instruction word in ROM consisting of ones in all 24 bits.

Example of a ONES statement:

Character Column:	1	10
Fields:	Field 1	Field 2
Content:	NEG 1	ONES

4-25. SKP

Character Column:	1	10	15	20	25	30	40	80
Fields:	Field 1	Field 2	Field 3	Field 4	Field 5	Field 6	Field 7	
Content:	(blank)	"SKP"	(blank)	(blank)	(blank)	(blank)	COMMENTS	

The SKP statement commands the micro-assembler to skip to the Top of the next page (TOP OF FORM command) during the listing of the microprogram. No locations in ROM are used, when this statement is specified.

Example of a SKP statement:

Character Column:	1	10
Fields:	Field 1	Field 2
Content:		SKP

4-26. ZEROES

Character Column:	1	10	15	20	25	30	40	80
Fields:	Field 1	Field 2	Field 3	Field 4	Field 5	Field 6	Field 7	
Content:	LABEL	"ZEROS"	(blank)	(blank)	(blank)	(blank)	COMMENTS	

The ZEROES statement creates a 24 bit micro-instruction word in ROM consisting of zeroes in all 24 bits.

Example of a ZEROES statement:

Character Column:	1	10	40
Fields:	Field 1	Field 2	Field 7
Content:	NULL	ZEROS	NO BITS

MICROPROGRAMMING SOFTWARE

SECTION V

Two sets of programs are provided to assemble, debug, and implement microprograms. One set operates in the BCS (Basic Control System) environment and the other operates in the DOS-III (Disc Operating System) environment.

5-1. MICROPROGRAMMING SOFTWARE SUMMARY

The following microprogramming software is provided:

- A two-pass micro-assembler, which converts the user's source microprogram record into an object tape and microcode listing.
- A Micro Debug Editor, which reads the object tape into Main Memory, outputs it to Writable Control Store (WCS), and allows the user to run the microprogram in WCS. The user can set breakpoints, change micro-instructions, change registers, etc. This program also provides the ability to punch the paper tapes that are used to create ("burn") programs into the ROM.
- A WCS I/O Utility subroutine, callable from FORTRAN and ALGOL libraries, that allows a microprogram, stored in a regular FORTRAN, ALGOL, or Assembler program buffer (in Main Memory), to be written into WCS.

5-2. MICRO-ASSEMBLER

The Micro-assembler accepts 80-character fixed-field card format records from a card reader, paper tape reader, or disc (using the DOS-III JFILE directive). Each record contains one micro-instruction coded in mnemonic format as described in Section IV of this manual. The micro-assembler processes input records and produces an object program paper tape which contains micro-instructions in binary format. Optionally output is a microprogram listing in both mnemonic and binary format, a symbol table, and error messages.

5-3. HARDWARE ENVIRONMENT

The BCS version requires the following as the minimum hardware:

a. An HP 2105 or HP 2108 Processor with 8K of Main Memory.

b. A Teleprinter.

This minimum system means that the assembly of the microprogram will be slow, since all input, listing, and punching must take place on the teleprinter.

The following additional hardware is supported:

a. Paper Tape Reader for source microprogram input.

b. Paper Tape Punch for binary object tape output.

c. Card Reader for source microprogram input.

d. Line Printer for microprogram assembly listing and symbol table listing.

e. 7970 or 3030 Magnetic Tape Unit for temporary storage of source microprogram that is input to Pass 2 of the micro-assembler.

The DOS-III version of the micro-assembler requires the same hardware as the DOS-III system.

5-4. MICRO-INSTRUCTION SOURCE RECORD

A micro-instruction source record has the following characteristics:

a. Length $\leqslant 80$ characters.

b. If not on a punched card, terminated by RETURN and LINE FEED.

c. Seven fields with the starting column of each field as follows:

Field Number	Character Column
1	1
2	10
3	15
4	20
5	25
6	30
7	40

Figure 5-1 shows a card record.

Refer to Section IV, "Microprogramming Language," for a description of the micro-orders appropriate to the seven fields.

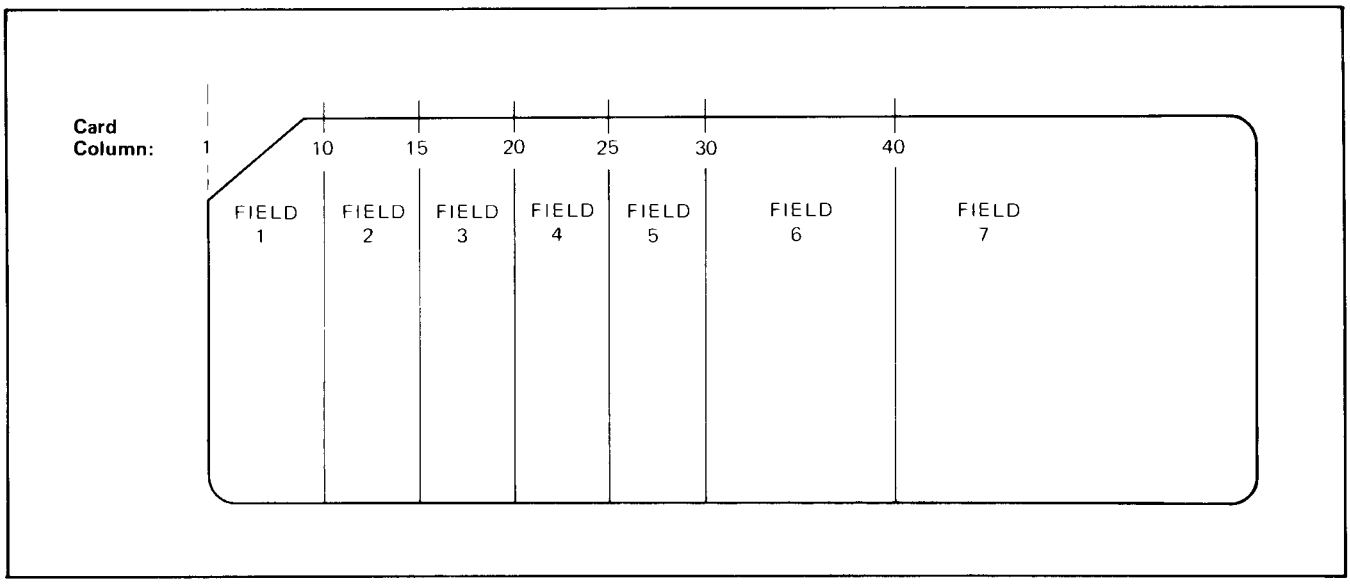


Figure 5-1. Micro-instruction Card Source Record

5-5. MICRO-ASSEMBLER CONTROL RECORD

Control statements are interspersed with micro-assembler language statements and specify control over the assembly process. For example, they may define the logical unit number of an input or output device or suppress listings.

There is one control statement per Control Record. If not on a card, it must be terminated by RETURN and LINE FEED.

Two control statements are required for every microprogram:

a. $ORIGIN statement
b. $END statement

All control statements start with a "$" (Dollar character) in column 1. No intervening spaces are allowed in any control statement other than as specified. Details on each statement text and meaning are given below.

$END

General Form: $END

Meaning: End of microprogram

Purpose: Required as the last statement in every microprogram

Example: $END

$EXTERNALS

General Form: $EXTERNALS = name1*b*address1, *b*name*b*address2, *b*. . .name*n**b*address**n**

A comma and a space (*b*) separate each external name and address pair. Each "name" conforms to the Label definition in Section 4-1 and "address" means an octal address in the range 0 - 7777.

Meaning: Define the following label names:
name1 refers to address1
name2 refers to address2
.
.
.
name**n** refers to address**n**

Purpose: Each $EXTERNALS control statement provides for one or more branch (JMP or JSB) target addresses outside of the microprogram.

Example: $EXTERNALS = OUTPUT 1012, CHAR 736.

$FILE (Used by DOS-III systems only)

General Form: $FILE = filename

The filename must be in accordance with DOS-III file name requirements.

Meaning: The object output file name for this microprogram is "filename."

Purpose: Provides the DOS-III micro-assembler with the name of the disc file into which the binary object code is to be stored.

Example: $FILE=MOBJ

Note: Prior to assembling a microprogram with a $FILE control statement, the user must have reserved a disc file using the DOS-III ":ST,B, . . ." directive.

$INPUT

(Used by BCS systems only)

General Form: $INPUT = lun

The logical unit number, lun, must be octal and in the range 1 - 74.

Meaning: The logical unit number of the device through which all subsequent input (to the next $END statement) is to be read is "lun."

Purpose: When the assembly process is begun in BCS systems, the micro-assembler expects the first source statement to be entered through the system console device (logical unit number 5). The user may enter the whole source program through the system console device. Normally, however, the user enters a $INPUT command specifying the logical unit number of the card reader or paper tape reader from which the rest of the source program is to be read.

Example: $INPUT = 12

$LIST

General Form: $LIST = lun
The logical unit number, lun, must be octal and in the range 1 - 74.

Meaning: The logical unit number of the listing device is "lun".

Purpose: To cause the assembly listing to be printed on the device having the specified unit number. If omitted, logical unit number is assumed to be 6 (standard list device).

Example: $LIST = 16

$ORIGIN

General Form: $ORIGIN = nnn

The origin, nnn, must be octal and in the range 0 - 7777.

Meaning: Set microprogram origin at octal address nnn in Control Store.

Purpose: Every microprogram must have its program address origin defined. New origins may be specified within the microprogram.

Example: $ORIGIN = 427

$RCASE

General Form: $RCASE

Meaning: Punch a special 32-micro-instructions/record object tape.

Purpose: This special object tape is reserved for system maintenance. Refer to Section 5-6 Micro-Assembler Output for a description of this special object tape.

Example: $RCASE

$OUTPUT

General Form: $OUTPUT = lun
The logical unit number, lun, must be octal and in the range 1 - 74. This statement may come anywhere before the $END statement.

Meaning: lun is the logical unit number of the output device.

Purpose: To specify the device on which the micro-assembler object code is to be output. If this statement is omitted, logical unit of 4 is assumed.

Example: $OUTPUT = 10

$PASS 2

(Used by BCS systems only)

General Form: $PASS2 = lun
The logical unit number, lun, must be octal and in the range 1 - 74. If present, this must be the first statement in the source deck or tape.

Meaning: lun is the logical unit number of the magnetic tape unit onto which all subsequent micro-assembler input is to be written.

Purpose: To cause all source input to be recorded on magnetic tape for use as input to Pass 2 of the micro-assembler. If this control statement is omitted, the computer halts at the end of Pass 1 to allow the operator to reload the microprogram source into the "$INPUT" device.

Note: The only magnetic tape units supported by the micro-assembler are the HP 3030 and HP 7970.

Example: $PASS2 = 23

$SUPPRESS

General Form: $SUPPRESS

Meaning: Suppress all warning error messages.

Purpose: To cut down the volume of messages to the console device. Fatal error messages will still be printed.

Example: $SUPPRESS

$SYMTAB

General Form: $SYMTAB

Meaning: Print symbol table

Purpose: To provide the user with label names and corresponding octal addresses used in his microprogram.

Example: $SYMTAB

5-6. MICRO-ASSEMBLER OUTPUT

This section describes all forms of output from the micro-assembler. They are:

- Binary Object
- Symbol Table
- Source and Binary Microprogram Listing
- Error Messages

5-7. BINARY OBJECT OUTPUT

The Standard Object Tape output by the micro-assembler to paper tape or a disc file consists of one or more Instruction Records, the format of which is shown in Appendix A, Figure A-1. One Instruction Record holds up to 27 micro-instructions and five words of header information. Each micro-instruction requires 32 bits or two words in the format: an eight bit address and 24 bits for the micro-instruction. Hence the length of the record =

5 words of header

2**n** words for **n** micro-instructions (2 words for each micro-instruction)

5+2**n** words for one Instruction Record

No more than 27 micro-instructions are written into an Instruction Record. Hence the maximum length = 5+(2x27)=59 words. The last object record is a four word End Record. When the microprogram consists of more than 27 micro-instructions, a series of Instruction Records are produced with the last one holding 27 or less micro-instructions. For example, if 57 micro-instructions have been assembled, three Instruction Records and an End Record are required consisting of the following:

a. Instruction Record 1 holds 27 micro-instructions and consists of

5 words of header
54 words for 27 micro-instructions
59 words

b. Instruction Record 2 holds 27 micro-instructions and consists of

5 words of header
54 words for 27 micro-instructions
59 words

c. Instruction Record 3 holds 3 micro-instructions and consists of

5 words of header
6 words for 3 micro-instructions
11 words

d. The End Record consists of

4 words
133 words for the entire microprogram Binary Object.

The Standard Object format is accepted by all programs which accept standard relocatable format. Thus a Standard Object tape can be stored in a DOS-III file using the ":STORE,R,..." directive. However, if the DOS-III user wants the Binary Object stored automatically in a disc file by the micro-assembler, the DOS-III directive "STORE,B,. . ." must have previously been used to reserve a disc file.

The Micro-assembler can also produce a non-standard object as the result of the inclusion of the $RCASE control statement. This optional object is the HP ROM Simulator Object tape. The format of this tape is shown in Appendix A, Figure A-2.

5-8. SYMBOL TABLE LISTING

If the user has a $SYMTAB control statement in his microprogram source input, then the micro-assembler will print a symbol table on the device with logical unit number 6 or on the device defined by the $LIST control statement, if present.

An example of a symbol table is shown in Figure 5-2.

On the left are the symbols or labels in the microprogram. On the right is the value of the symbol; that is the six digit absolute octal address of the symbol. Where X follows the address, the symbol has been defined by a $EXTERNAL control statement.

```
SYMBOL TABLE

MOVE        002412X
GOTO        003421X
RET         002427X
LAST        002717X
OUT         002011
ERR1        002012
```

Figure 5-2. Symbol Table

5-9. MICROASSEMBLY LISTING

Unless suppressed by the $NOLIST control statement, the micro-assembler provides a listing like the one shown in Figure 5-3. This listing is associated with the symbol table illustrated in Figure 5-2.

5-10. MICRO-ASSEMBLER ERROR MESSAGES

During the assembly process the micro-assembler checks each instruction for errors. If an error is detected, an error message of the following general form is printed in the Micro-assembly Listing.

****ERROR eeee IN LINE nnnn**

where

eeee
is an Error Code defined in Table 5-1 and

nnnn
is a line number in the Micro-assembly Listing.

Table 5-1 gives the meaning of each error code and the recovery procedure. Note that Figure 5-2 holds examples of two error messages in lines 9 and 11.

5-11. DOS-III OPERATION OF MICRO-ASSEMBLER

Before using the DOS-III version of the Micro-assembler, the following items must be available.

a. A current DOS-III system.

b. A source microprogram, on cards, paper tape, or in a source file on disc.

```
0001                    $ORIGIN=2000B FIRST ADDRESS OF MODULE 4
0002                    $SYMTAB   PRINT SYMBOL TABLE
0003                    $EXTERNAL=MOVE 2412, GOTO 3421, RET 2427, LAST 2717
0004                    * P2=A&P1
0005 2000 220 074457            READ      INC  M    P         READ ADDEND P
0006 2001 017 126157                      PASS L    A         PUT AUGEND IN L AND ENABLE E & O
0007 2002 264 101557            ENVE      ADD  S12  TAB       ADD MEMORY TO L AND STORE IN S12
0008 2003 324 140531            JMP  CNDX E         ERR1      IF E SET, GO TO ERR1

**ERROR 0008 IN LINE 0009
0009 2004 320 000030            JMP  CNDX OVFL      ERR2      IF O SET, GO TO ERR2
0010 2005 000 075717                      INC  P    P         BUMP P FOR NEXT PARAMETER

**ERROR 0003 IN LINE 0011
0011 2006 017 136757           READ       INC  M    P         READ NEST PARAMETER P2 ADDRESS
0012 2007 000 000461                 MPCK  INC  M    TAB       PUT IN M AND CHECK FOR M P ERR
0013 2010 177 166017            WRTE      PASS TAB  S12       PUT ADD RESULT INTO MEM ADD P2
0014 2011 017 136776  OUT             RTN                      THE RETURN
0015 2012 344 001757  ERR1      IMM       LOW  S    0         SET UPPER BYTE FOR E ERR
0016 2013 320 100470            JMP                 OUT       RETURN
0017 2014 340 001757   ERR2     IMM       HIGH S    0         SET LOWER BYTE FOR O ERR
0018 2015 320 100470            JMP                 OUT       RETURN
0019                    $END
** 0002 ERRORS**
```

Line Number	ROM Address	Bits 23-16	Bits 15-0	Field 1	Field 2	Field 3	Field 4	Field 5	Field 6	Field 7
		Binary Micro-instruction								

Figure 5-3. Micro-Assembly Listing

c. The Micro-assembler program named MICRO stored in the DOS-III user library. If MICRO still is on relocatable object paper tape (HP 12978-160001), it can be loaded in the same way as any other relocatable object program.

For the detailed description of DOS-III operation, see **HP 24307B DOS-III** Reference Manual (HP 24307-90006).

a. If there is a $FILE control statement in the microprogram source, a binary file must be reserved on the disc before beginning the micro-assembly process to hold the relocatable object. The name of the reserved disc file must be the same as the one specified in the $FILE control statement.

b. Place the microprogram source in the input device; turn the device on; turn on the paper tape punch and the list device.

Table 5-1. Micro-assembly Error Messages

Error Code	Meaning/Recovery
1	Duplicate Label. The statement label of the micro-instruction in line **nnnn** is the same as another statement in the microprogram or the same as a declared $EXTERNAL symbol. Assign a new statement label and reassemble.
2	Illegal Control Statement. Correct control statement in line **nnnn** and reassemble.
3	Illegal Field 2 Micro-order. A NOP is inserted in field 2 and assembly continues. Correct line **nnnn** and reassemble.
4	Illegal Field 3 Micro-order. A NOP is inserted in field 3 and assembly continues. Correct line **nnnn** and reassemble.
5	Illegal Field 4 Micro-order. A NOP is inserted in field 4 and assembly continues. Correct line **nnnn** and reassemble.
6	Illegal Field 5 Micro-order. A NOP is inserted in field 5 and assembly continues. Correct line **nnnn** and reassemble.
7	Illegal Field 6 Micro-order. A NOP is inserted in field 6 and assembly continues. Correct line **nnnn** and reassemble.
8	Illegal JMP or JSB Address. Address is outside permitted range, or target label address is undefined. A value of 0 will be inserted into address field of line **nnnn** and assembly continues. Redefine address and reassemble.
9	Microprogram Too Large. The last relative address in the program is 400 or greater. A $ORIGIN statement must be changed or the program broken up into smaller parts before reassembly.
10	Missing $ORIGIN Control Statement. At least one $ORIGIN control statement is required. Insert $ORIGIN statement and reassemble.
11	Illegal Word Type 2 Operand. Operand of the IMM micro-instruction is outside the permitted range. A value of 0 is inserted into the operand and assembly continues. Correct line **nnnn** and reassemble.
OR **aaaa**	Insufficient DOS-III File Space Reserved. Reserve a binary file with more sectors for storage of the file named in the $FILE control statement (**aaaa** is an address in the micro-assembler and can be disregarded). See DOS-III manual section 15 under Error Conditions.
ABORT!	An irrecoverable error has occurred; correct error and reassemble.

c. Summon the Micro-assembler with statement

:PR,MICRO,[p1,p2,p3,p4,99]

where

p1 = the input device logical unit number

p2 = list device logical unit number

p3 = paper tape punch device logical unit number

p4 = maximum number of lines-per-page on the list device.

If 99 is entered for any of the above parameters, that parameter and all those that follow are defaulted to "standard" values.

d. The program title

MICRO-ASSEMBLER

is printed and Pass 1 begins. If a $SYMTAB control statement is in the source microprogram, the symbol table is printed at the conclusion of Pass 1. Pass 2 begins immediately (from disc) and the listing and relocatable object tape are output. Micro-assembly is complete.

Note: If Pass 2 fails to begin, check that the paper tape punch is turned on. The micro-assembler will cycle in a loop until the punch is turned on.

5-12. BCS OPERATION OF MICRO-ASSEMBLER

Before proceeding, the following items must be available:

- An absolute BCS binary tape.
- A reloctable object tape of the Micro-assembler program MICRO (HP 12978-160003).
- A source microprogram either on cards or paper tape.

For a detailed description of BCS usage, see the **Basic Control System** manual (HP 02116-9017).

The following procedure need be performed only once. When an absolute binary tape of the Micro-assembler is punched, it is used as described in the procedure "Executing the Micro-assembler."

Making an Absolute Micro-assembler tape:

a. Load the absolute BCS binary tape using the Basic Binary Loader.

b. Set the P-register to 2. Set bit 14 of the Switch Register and clear all other Switch Register bits.

c. Place the MICRO relocatable object tape in the paper tape reader. Check that the paper tape reader and the console device are on. Turn on the paper tape punch. Press PRESET and RUN on the CPU front panel. MICRO reads in and absolute binary tape is punched.

d. The message

*LOAD

is printed and the computer waits. Set Switch Register bits 2 and 14 leaving all others clear. Load BCS Library tape into the paper tape reader. Press RUN.

e. The BCS Library tape reads in and the rest of the absolute binary tape is punched. Linkage information is printed on the console device.

This is the absolute binary tape of MICRO, used for input to the next step.

Executing the Micro-assembler:

a. Load the MICRO absolute binary tape using the Basic Binary Loader.

b. When loading is complete, set P-register to 2. Press PRESET and RUN. The message

MICRO-ASSEMBLER

is printed followed by a request for the logical unit number of the source input device.

c. Enter the logical unit number followed by carriage return/line feed. Pass 1 now begins. If a $SYMTAB control statement is in the microprogram source, the symbol table is printed at the conclusion of Pass 1. (See Section 5-5 for a description of the $SYMTAB control statement.)

d. Turn on the paper tape punch.

e. Pass 2 begins immediately. If no $PASS2 control statement was included in the source, the message

RELOAD SOURCE, PRESS RUN

is printed. Reload the source microprogram into the input device and then press RUN on the front panel of the computer.

Note: If Pass 2 fails to begin, check that the paper tape punch is turned on. The micro-assembler will cycle in a loop until the punch is turned on.

If a teletype is used for both listing and punching, the computer halts (M-register = 102052) so that the operator can press the paper tape punch ON button to punch the microprogram object tape. The operator then presses RUN on the computer front panel.

When the paper tape is punched, another halt (M-register = 102053) occurs, so that the paper tape punch button can be set to OFF. Press RUN on the computer front panel.

f. Pass 2 completes micro-assembly. The microprogram object tape is complete. To assemble another microprogram proceed from step b.

5-13. MICRO DEBUG EDITOR

The Micro Debug Editor (MDE) makes it possible to load the object microprograms output from the Micro-assembler into a Writable Control Store module. It also provides the ability to debug microcode stored in the WCS and to "burn" microprograms into ROM chips.

Before using the Micro Debug Editor to debug microprograms, the Writable Control Store PCAs must be set to the required control store module numbers. This is accomplished by the installation of a module selection Jumper Assembly (HP Part Number 5060-8342). Refer to Section 6 of this manual for installation of the module selection Jumper Assembly and the WCS PCAs.

5-14. HARDWARE ENVIRONMENT

The BCS version requires the following minimum hardware:

a. HP 21MX Series Computer with 8K of Main Memory

b. A console device

c. A paper tape reader

d. One or more WCS PCA's, depending on the size of the microprogram to be debugged.

e. If a ROM program tape is to be punched, a paper tape punch is also required.

The DOS-III version of the MDE requires the same minimum hardware as the DOS-III system.

5-15. INITIALIZATION PROGRAM

When the Micro Debug Editor is to be run for debugging purposes (as opposed to being run merely to punch ROM program tapes), the user must supply an initialization program. The initialization program is an assembly language program that prepares the necessary parameters in

```
ASMB,R,B,L,T          Assembly parameters
      NAM TEXT,6      Program name (DOS-III)
      ENT TEST,MACRO  Entry points
TEST  NOP
        .             Any initialization procedure re-
        .             quired by the microprogram
        .
MACRO OCT 105xxx      (or 101xxx) Instruction that calls
                      the user microprogram
      DEF P1
      DEF P2          Parameter addresses required by
         .            the microprogram
         .
         .
      DEF Px
      JMP TEST,I      Return to calling program (MDE)
P1    (parameter 1 value)
P2    (parameter 2 value)
        .             Parameter values
        .
        .
Px    (parameter x value)
      END
```

Figure 5-4. General Format of the Initialization Program

Main Memory and then executes a 101xxx or 105xxx macro-instruction.

The name of the initialization program must be TEST (required in BCS systems, is a NAM TEST statement; in DOS-III systems a NAM TEST, 6 statement). The program must also have the symbol "MACRO" declared as an entry point where MACRO is the symbolic address (label) of the macro-instruction (101xxx or 105xxx) which calls the microprogram under test. Note that there must only be one such macro-instruction in the TEST initialization program.

Figure 5-4 holds the general structure of the initialization program.

This initialization program is called as a relocatable subroutine by MDE. Thus, its name is one of the references that must be satisfied when loading MDE.

A note of caution: a microprogram cannot be debugged using MDE unless the microprogram has:

a. An entry point which is a "JMP" micro-instruction of Word Type 4 (described in Section 4-17).

b. The micro-instruction jumped to by the JMP at the entry point must not contain a "READ" micro-order.

An example of a short initialization program is shown in Figure 5-5.

5-16. USING THE MICRO DEBUG EDITOR

Section 5-37 describes how to execute MDE using the DOS-III operating system. Section 5-38 describes how to execute MDE using the BCS operating system.

Before using the Micro Debug Editor to debug a microprogram, the Writable Control Store PCAs must have the correct terminal board plugged in, to establish the Control Store module number. Refer to Section VI of this manual for a description of setting module numbers in a Writable Control Store PCA.

When the module number has been set in the Writable Control Store PCA and it is plugged into the correct I/O slot, the user loads the microprogram object tape (produced by the Micro-assembler) using the Micro Debug Editor LOAD command. The microprogram is then output to the Writable Control Store using the WRITE command.

When the user is ready to execute his microprogram, the EXECUTE command is used. For the microprogram to execute properly, the following conditions must hold:

a. The module that the microprogram was written into matches the range of addresses used by the microprogram. For example, a microprogram whose addresses are in the octal range 2400 to 2777 must be stored in a Writable Control Store PCA which has been set to module 5.

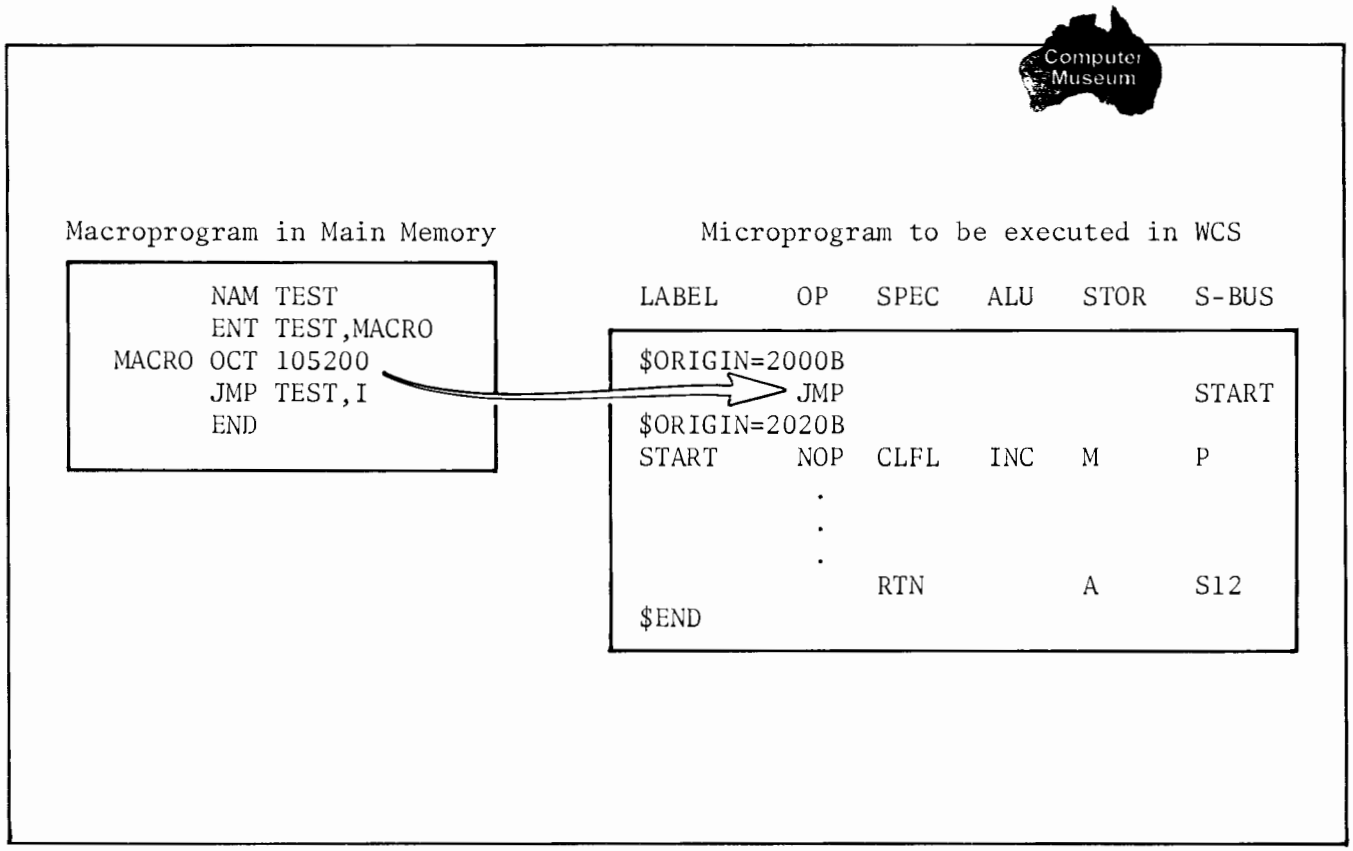


Figure 5-5. Test Program Call to Microprogram

b. The macro-instruction in the TEST program must initiate entry into Control Store at the proper address of the microprogram to be tested.

Micro Debug Editor results are unpredictable if either of the above conditions are not met.

When MDE is executed, it prints the input prompt

COMMAND?

on the system teleprinter.

Respond by entering one of the input, edit, output, or debug commands described in Table 5-2 and the following pages. In most cases, the first letter of the command is sufficient to specify it to MDE. The two commands, "MOVE" and "MODIFY", require at least three letters to identify the command. After MDE has performed the specified operation, it again prints COMMAND? to repeat the cycle.

Terminate an MDE run by entering the FINISH command.

There are 13 MDE commands which are summarized in Table 5-2. A detailed description of each command follows. Whenever a logical unit number (lun) is called for, it must be entered in octal.

Note that the last octal 45 words of the lowest numbered WCS module loaded with a microprogram are used by Micro Debug Editor for its own resident microcode. If these locations are required by the user microprogram under test, use the MOVE command to relocate the MDE microcode before loading the user microprogram.

The Micro Debug Editor uses a Main Memory buffer to hold the microprogram object code. When the microprogram is loaded from an object tape, it is stored into this buffer. Most MDE commands make modifications or transfers to and from this buffer.

Use of the PREPARE command to punch the six ROM microprogram mask tapes has the following restriction. This buffer must have been loaded using an object tape produced by the micro-assembler and the buffer must not have been modified.

5-17. INPUT COMMANDS

5-18. LOAD[,X]

Meaning: Load the object microprogram produced by the Micro-assembler from disc or paper tape into the MDE buffer. The logical unit number (lun) of the input device is **X**.

Usage: The Micro-assembler control statement $FILE can be used to specify (during assembly) the name of the DOS-III file into which the object code is to be stored. In the DOS-III version of MDE, if the logical unit number entered is that of the disc, MDE will respond with a request for the name of the file in which the object code is to be stored:

FILENAME?

Enter the file name given to the object code by the $FILE control statement.

Note: When loading the object microprogram for output to WCS (instead of punching ROM tapes), the LOAD command must be followed immediately by a WRITE command to the appropriate WCS PCA. No intervening commands are allowed. This allows the Micro Debug Editor to build a table relating microprogram addresses to WCS logical unit numbers.

Table 5-2. Micro Debug Editor Commands

INPUT	
Commands:	LOAD[,X] READ,X
EDIT	
Commands:	SHOW,xxxx[,yyyy] MODIFY,xxxx[,yyyy]
OUTPUT	
Commands:	DUMP[,X] WRITE,X
	PREPARE[,X] VERIFY[,X]
TERMINATION	
Command:	FINISH
DEBUG	
Commands:	BREAK,yyyy CHANGE[,mnemonic] EXECUTE[,0 or yyyy]
RELOCATE MDE WCS-RESIDENT MICROCODE	
Command:	MOVE,yyyy

Note

The brackets indicate that the parameter may be omitted.

5-19. READ,X

Meaning: Read the contents of a WCS into the Micro Debug Editor buffer. **X** is the logical unit number of the WCS.

Usage: If no WCS is on the specified logical unit, the MDE buffer is unchanged. **No** notification is made to the user that the buffer is unchanged or that no WCS is on the logical unit specified. Thus, if READ or SHOW is being used to insure that a previous WRITE executed properly to the same (non-WCS) logical unit, the MDE buffer will still hold the data that was assumed to be written to that logical unit. The user could incorrectly assume that the non-existent WCS holds the proper data.

5-20. EDIT COMMANDS

5-21. SHOW,xxxx[,yyyy]

Meaning: Display the WCS contents on the console device, where **xxxx** is the beginning address and **yyyy** is the ending address. Only the contents of the address **xxxx** are displayed, if **yyyy** is omitted.

Usage: See Usage under 5-19, READ,**X**.

The display format of each 24-bit word is:

```
aaa  mmm  nnnnnn
```

where aaa is the control store address of the location being displayed, mmm is the octal representation of bits 23-16 of the location, and nnnnnn is the octal representation of bits 15-0 of the location.

5-22. MODIFY,xxxx[,yyyy]

Meaning: Change the contents of the MDE buffer and the WCS where **xxxx** is the beginning WCS address and **yyyy** is the ending WCS address. Change WCS address **xxxx** if **yyyy** is omitted.

Usage: See Usage under 5-25, WRITE **X**.

"MOD" is the minimum input required to initiate the modify command. **xxxx** and **yyyy** must be absolute WCS addresses in a single WCS module. One at a time, the contents of each location are printed on the console device in the same format as the SHOW command above. Following the location contents, the operator enters the new location contents followed by a CARRIAGE RETURN and LINE FEED.

If fewer than 3 digits are entered for **mmm** or fewer than 6 digits are entered for **nnnnnn**, the number entered is right justified with zeros automatically filled to the left. To specify that no change is to be made, enter an asterisk (*), instead of **mmm** or **nnnnnn**.

Example:

```
MOD,4000,4003
4000  123  456777  *,123456
```

leaves bits 23-16 unchanged and sets bits 15-0 to 123456 in WCS location 4000.

```
4001  123  456777  6,123
```

is equivalent to entering 006,000123; bits 23-16 are set to 006 and bits 15-0 are set to 000123 in location 4001.

```
4002  123  456777  123,*
```

sets bits 23-16 to 123 and leaves bits 15-0 unchanged in location 4002.

```
4003  123  456777  *,*
```

makes no change to location 4003.

5-23. OUTPUT COMMANDS

5-24. DUMP[,X]

Meaning: Punch the entire contents of the MDE buffer on the paper tape punch. **X** is the logical unit number of the paper tape punch. If **X** is omitted, it is assumed to be 4.

Usage: The DUMP command must be preceded by a READ or LOAD command to fill the MDE buffer. The tape produced is in the same format as the object tape produced by the Micro-assembler. If the tape is reloaded into the MDE buffer, the buffer cannot be used to punch (PREPARE command) a set of six pROM mask tapes. The primary use of this tape is to enable the user to save the results of a microprogram debug session for resumption later.

5-25. WRITE,X

Meaning: Write the contents of the MDE buffer into the WCS. **X** is the logical unit number of the WCS.

Usage: Since the Micro Debug Editor addresses the WCS by logical unit number, it is the responsibility of the user to insure that a WCS is installed with logical unit number **X** and that it is set to the proper module for the micro-code to be stored. If no WCS is on the specified logical unit, no notification is given to the user that a WRITE or MODIFY command failed to transmit data to the non-existent WCS.

5-26. PREPARE[,X]

Meaning: Punch a set of six pROM mask tapes each headed by three lines of I.D. and a checksum on the paper tape punch. **X** is the logical unit number of the device. If **X** is omited, it is assumed to be 4.

Usage: Following entry of the PREPARE command, a cycle of dialogue is initiated between the operator and the console device. In the following procedure, the underlined characters indicate operator input is required at the console device. Each entry must be followed by a CARRIAGE RETURN and LINE FEED.

a. Turn on the paper tape punch. The message cycle starts with:

GENERATION OF MASK BITS 23-20

where 23-20 represents the 4 bit range of bits to be punched into the first mask tape.

ENTER 3 LINES OF I.D. INFORMATION

LINE 1 — **key in first line of tape I.D.**

LINE 2 — **key in second line of tape I.D.**

LINE 3 — **key in third line of tape I.D.**

Enter up to 72 characters of identification information in each line.

b. Following entry of the third I.D. line, the mask tape is punched for mask bits 23 to 20. This is for ROM chip number 6. The following cycle of dialogue is repeated for each of the remaining five mask tapes:

GENERATION OF MASK BITS UU-LL

UU - LL is the range of bits to be punched.

ANY CHANGE OF I.D. INFO IN LINE 1? **key in N (no) or Y(yes) and new line 1 I.D.**

LINE 2? **key in N or Y and new line 2 I.D.**

LINE 3? **key in N or Y and new line 3 I.D.**

c. The next mask tape is punched. When all six mask tapes have been punched, the following message is output:

GENERATION OF TAPES COMPLETED

The six mask tapes have the following characteristics:

UU-LL	Punch Sequence	For Module ROM Chip No.
23-20	First tape	6
19-16	Second tape	5
15-12	Third tape	4
11-08	Fourth tape	3
07-04	Fifth tape	2
03-00	Sixth tape	1

Conventions: Line 1 I.D. holds module number, ROM chip number, number of bits (4), ROM size, and other I.D. information.

For example:

LINE 1-1,005, 4, 1025 REENTRY FACTOR

Line 2 I.D. holds part number or other central reference number. For example:

LINE 2-MT 38-0226 REVISION C

Line 3 I.D. holds date and any other I.D. information. For example:

LINE 3-04/01/75 PVT. D.M. BULMAN

5-27. VERIFY[,X]

Meaning: Compare the contents of the pROM mask tapes to the contents of the MDE buffer. The logical unit number of the paper tape reader is **X.**

Usage: Following entry of the command, the console device requests the range of bits in the pROM mask tape to be compared to the MDE buffer (underlined characters indicate operator entry).

TAPE NUMBER: **uull**

Enter CARRIAGE RETURN and LINE FEED after the bit range uu (upperlimit) and 11 (lowerlimit). Refer to 5-26 PREPARE[,X] for valid bit ranges.

For example, the entry "2320" specifies verification of bits 23 to 20. The paper tape then reads the mask tape and compares its contents to the specified bits in the MDE buffer. As the tape is being read, the three lines of I.D. (see PREPARE command) and checksum are printed on the console device.

Note: If the DOS-III operating system is being used, and no errors were encountered, an I/O "error" message is printed at the console device:

I/O ERR ET EQT **#n**

Where **n** is the EQT number of the paper tape reader. This message notes a characteristic of the mask tape that DOS-III normally interprets as an error condition, but the message in fact, connotes **no** error.

If no errors were detected, the message

TAPE VERIFIED

is printed. Enter another bit range as before. The VERIFY command completes only after the bit range 03 ot 00 has been entered and verified.

Errors: If errors are detected, dialogue between the console device and the operator is initiated. Follow each operator entry with CARRIAGE RETURN and LINE FEED.

a. The message CHECKSUM ERROR OR BAD MASK TAPE is printed followed by a tape repunch request:

   DO YOU WANT TO REPUNCH THIS TAPE?
   enter Y or N

b. If N is entered, another bit range request with the message

   TAPE NUMBER?

   Enter another bit range as before. The VERIFY command completes only after the bit range 03 to 00 has been entered and verified.

c. If Y is entered, the following request is made:

   ENTER PUNCH LOGICAL UNIT # **enter octal logical unit number of paper tape punch**

   The message

   ENTER THREE LINES OF I.D. INFORMATION

   is printed.

   Enter up to 3 lines of tape I.D. information according to the procedure given in 5-26, PREPARE[,X]. The new mask tape is punched, headed by the I.D. information.

Special DOS-III operation: When a series of bit ranges are being verified, specification of each successive range at the console device (as a result of the message TAPE NUMBER?) will bring about the prompt character "@". To verify the specified bit range on paper tape:

a. Enter the following command

   :UP,**n**

   where **n** is the EQT number of the paper tape reader.

b. Then enter:

   :GO

   The next tape to be verified will read in as above.

Verify sequence: The mask tapes may be verified in any order with exception that the last tape verified must have the bit range 03 to 00.

5-28. TERMINATION COMMAND

5-29. FINISH

Meaning: Terminate the current MDE run.

5-30. DEBUG COMMANDS

5-31. BREAK,yyyy

Meaning: Set a Breakpoint at location **yyyy** and clear the previous one. If yyyy = 0, no breakpoint is set and the previous one is cleared.

Usage: Microcode execution is initiated by an EXECUTE command. When the Breakpoint address **yyyy** is reached,

REG'S?

is printed and microprogram execution ceases (breaks). Enter the mnemonics of the flags or registers that are to be displayed, separated by commas. The mnemonics are described under the CHANGE command. The entry is of the form

REG'S? **m1,m2,m3, . . . mn**

where **m1** through **mn** are register and flag mnemonics. The resulting display is of the form

m1 = c1, m2 = c2, m3 = c3, ..., mn = cn

when c1 through cn are octal contents of the requested registers and flags.

Example of a display request:

REG'S **A,B,1,2,3,4,14**

The resulting display:

```
A = 00004,    B = 103005,    1 = 000447,
2 = 00012,    3 = 00000,     4 = 00000,
14 = 034716
```

Enter "!" to display all registers and flags. Enter "/" to return to command entry mode.

Restrictions: Do not set a breakpoint

a. in the WCS entry point address of the microprogram

b. in a microprogram subroutine (within the JSB . . . RTN code limits)

c. in an address where the micro-instruction passes information to or from the T-register immediately following a WRITE or READ micro-order.

5-32. CHANGE[,m]

Meaning: Alter the contents of one or more registers and flags. If the mnemonic **m** is specified, alter the contents of the register or flag which it specifies. It not specified, all registers and flags are displayed in sequence to prompt the user to make required changes.

Mnemonics: The list of register and flag mnemonics follows:

Mnemonic	Stands For	Mnemonic	Stands For
A	A-register	9	S9-register
B	B-register	10	S10-register
S	S-register	11	S11-register
P	P-register	12	S12-register
1	*S1-register	X	X-register
2	S2-register	Y	Y-register
3	S3-register	O	Overflow Register bit
4	S4-register	E	Extend Register bit
5	S5-register	F	CPU Flag bit
6	S6-register	CN	Counter Register
7	S7-register	L	L-register
8	S8-register		

*Scratch Pad Register 1; similarly for S2, S3, etc.

Usage: Upon entry of the command, the message

m xxxxxx =

is printed, where **m** is the register or flag mnemonic and xxxxxx is the octal representation of the contents. Enter the new contents or an asterisk (*) if no change is to be made.

Example of a CHANGE request:

CHANGE,6
6 173777 = **173770**

This is a request for a change to S6-register (Scratch Pad Register 6). The original contents were octal 173777. The new contents are octal 173770.

5-33. EXECUTE[,yyyy]

Meaning: Execute microprogram.

If **yyyy** = 0, the TEST initialization program is run, which carries execution to the microcode in WCS. This is the normal mode of initiating microcode.

Note: If the entire system goes dead after entering an EXECUTE,0, the reason may be that the WCS with the correct module number is not plugged into the correct slot.

If **yyyy** = an absolute WCS address, execution of microcode begins at that address.

If **yyyy** is omitted, execution resumes from the last breakpoint with registers and flags set

a. according to their setting when the breakpoint was encountered, or

b. modified by the CHANGE command.

Usage: Execution will continue until a breakpoint is encountered or until the microprogram is completed. When complete, the command entry mode is repeated.

Before initiating a microprogram execute (other than EXECUTE,0), make sure that all registers and flags are preset using the CHANGE command, if necessary.

5-34. RELOCATE MDE WCS-RESIDENT MICROCODE

5-35. MOVE,yyyy

Meaning: Move the octal 45 word WCS-resident microprogram portion of MDE from the usually resident locations to locations beginning with **yyyy**.

Usage: "MOV" is the minimum input required to initiate the move operation. MDE requires a portion of WCS for register dump and register restore microprograms. These MDE microprograms are initially stored in relative octal locations 333 to 377 of the first WCS loaded. If the user requires these locations in Writable Control Store, he can move this resident MDE microcode elsewhere.

No check is made to see if a portion of the user microcode has been overlayed. The reason is that the user may actually want to situate the dump and restore microprograms on top of his own microcode as he debugs another portion of his code.

The actual relocation of the MDE microcode does not occur until the EXECUTE command is given.

5-36. MDE ERROR MESSAGES

During the use of MDE, commands, parameters, and processing functions are monitored. If an error condition is detected, an appropriate message is printed. Table 5-3 holds the list of MDE error messages plus their meaning and the recovery procedure.

5-37. DOS-III OPERATION OF MDE

Before using the DOS-III version of the Micro Debug Editor (MDE), the following items must be available.

a. A current DOS-III system

b. A relocatable object tape of MDE (HP 12978-16002).

c. A relocatable object tape of the TEST initialization program if a debug run is to be made.

d. A microprogram object tape output by the Micro-assembler.

The following is an example of how the user can proceed. For details on additional DOS-III options, see DOS-III manual (HP 24307-90006).

a. Store the two tapes, MDE and TEST, on the disc using the DOS-III store command

:ST,R,**filename, lun**

where **filename** is any suitable label and **lun** is the logical unit number of the paper tape reader from which the tapes are entered.

b. Make sure the list device is on. At the console device enter

:PR,LOADR,2

DOS-III responds with

ENTER FILE NAMES OR /E

c. Respond as follows:

MDE filename, TEST filename, /E

where MDE **filename** and TEST **filename** are the chosen file names used with the "ST" store command (step A), and /E specifies end of entry.

If MDE is being used only to load WCS with a microprogram, the TEST **filename** may be omitted. The loader then reads the two files into main memory. If the TEST initialization program has been omitted, the message

UNDEFINED EXTS

is printed indicating TEST is an undefined external to the MDE program.

To proceed, enter

:GO,1

When loading is finished, the message

LOADER COMPLETE

is printed.

Table 5-3. Alphabetical List of MDE Error Messages

Message	Meaning/Recovery
CAN'T FILL MORE THAN 16 MODULES!	User has tried to write microprograms to more than the maximum of 16 WCS modules. The user can debug no more than 16 WCS modules at a time.
ILLEGAL COMMAND	Command just entered is not an MDE command; re-enter command.
ILLEGAL DIGIT	An "8" or "9" was entered in the previous command that called for an octal digit; re-issue the entire command.
ILLEGAL PARAMETER	An unacceptable parameter was entered in the previous command; re-issue command.
ILLEGAL REG. MNEMONIC	Register or flag mnemonic just entered is not one of those listed under the CHANGE command (section 5-32); enter correct mnemonic.
ILLEGAL TAPE #	Bit range entered is not one of those listed under PREPARE command (section 5-26).
MISSING PARAMETER	A required parameter was omitted from the previous command; re-issue command.
NO BREAKPOINT HAS BEEN SET!	An EXECUTE-from-breakpoint command was given without having set a breakpoint logically beyond the execute address.
WCS NOT LOADED	The Writable Control Store PCA corresponding to the logical unit specified in the command just entered, has not been loaded with a microprogram during this MDE session; load the WCS.

d. Save the loaded MDE program with

:ST,P

To summon MDE from now on, enter

:PR,MDE

e. The program title is then printed followed by command request:

MICRO-DEBUG EDITOR
COMMAND?

Now enter the MDE commands required as described beginning in Section 5-16.

5-38. BCS OPERATION OF MDE

Before proceeding, the following items must be available:

a. An absolute BCS binary tape.

b. A relocatable object tape of MDE (HP 12978-16004).

c. A relocatable object tape of the TEST initialization program, if a debug run is to be made.

d. A microprogram object tape.

e. A BCS Library tape (HP 24145-60001), Revision B.

The following is an example of how the user can proceed. For details on additional BCS options, see the **Basic Control System** manual (HP 02116-9017).

a. Load the absolute BCS binary tape using the Basic Binary Loader.

b. Set the P-register to 2. Set bit 14 of the Switch Register and clear all other Switch Register bits.

c. Place MDE relocatable object tape in the paper tape reader and insure that the paper tape reader and the console device are on. Turn on paper tape punch. Press PRESET and RUN on the CPU Front Panel.

The MDE tape is read and an absolute binary tape is punched.

d. The message

*LOAD

is printed on the console device and the program halts.

If required, load the relocatable TEST Initialization Program tape into the paper tape reader. Press RUN.

The TEST tape is read and another absolute binary tape is punched.

e. The message

*LOAD

is printed on the teleprinter and the program halts.

Set Switch Register bits 2 and 14 leaving all others clear. Load BCS Library tape into the paper tape reader. Press RUN.

f. Library tape is read and more absolute binary tape is punched.

Linkage information is printed on the Teleprinter. Remove paper tape from punch. This is the complete absolute binary tape of the Micro Debug Editor including the TEST Initialization Program.

g. Load this tape using the Basic Binary Loader.

h. When loading is complete, set P-register to 2. Press PRESET and RUN. The message

MICRO-DEBUG EDITOR
COMMAND?

is printed.

i. Now enter the required MDE commands as described beginning in Section 5-16.

5-39. WCS I/O UTILITY SUBROUTINE

This library subroutine provides the capability of writing a microprogram into and reading a microprogram from a WCS using a buffer in an Assembly Language, FORTRAN, or ALGOL program and operating in a BCS or DOS-III environment. This avoids the necessity of running MDE every time it is necessary to access a WCS. This subroutine is in the standard BCS and DOS-III libraries for 21MX Series Computers.

Unlike a ROM chip, whenever the computer power is turned off, the WCS contents are lost. Thus the WCS must be loaded before access can be made to microprograms. This WCS I/O utility has been provided to serve that purpose.

Besides the calling sequence, a buffer is required in the calling program large enough to hold the number of microinstructions being transferred in or out.

Initially, the microprogram is stored on an object paper tape, in an object file on disc, or as octal data stored in the Main Memory program. In the case where the microprogram is in the form of octal data in the Main Memory program, the octal data area serves as the buffer when the WCS I/O Utility is used to write the microprogram into the WCS.

In the case where the microprogram resides on disc or paper tape, the control system (BCS or DOS-III) must be used to read the tape or disc file into a buffer in the Main Memory program. It must be remembered that the microprogram object contains header and end record information that must be deleted before storing the microprogram in the buffer. (Header and end record information must not be written into the WCS.)

Refer to Section 5-7 for a description of the Binary object output by the micro-assembler. Appendix A illustrates the binary object format.

When the microprogram has been stored in the Main Memory program buffer, a WCS I/O Utility WRITE calling sequence is used to write the microprogram into the WCS.

To read the contents of the WCS, a WCS I/O Utility READ calling sequence is used.

The assembly language calling sequences are the following:

READ

JSB WREAD	Branch to WCS read subroutine
DEF *+5	Return address
DEF lun	Logical unit number of WCS
DEF BUFF	Address of microprogram buffer
DEF LENGTH	Number of words of transfer
DEF ADRS	WCS relative address

WRITE

JSB WWRITE	Branch to the WCS write subroutine
DEF *+4	Return address
DEF lun	Logical unit number of WCS
DEF BUFF	Address of microprogram buffer
DEF LENGTH	Number of words of transfer

Where lun contains the logical unit number of the WCS being accessed and BUFF contains the first word of a word pair that holds a micro-instruction. LENGTH contains the octal number of words in the transfer; if LENGTH is positive, the number of 24 bit words is specified; if LENGTH is negative, the number of 16 bit words is specified. ADRS contains the WCS relative address (between octal addresses 0 and 377) of where to start reading.

WRITABLE CONTROL STORE

SECTION VI

This section covers general information, installation, programming, and general theory of operation for the HP 12978A Writable Control Store Interface Kit. Options 001 and 002 for the interface kit are also covered in this section.

6-1. GENERAL INFORMATION

The Hewlett-Packard 12978A Writable Control Store Interface Kit provides the HP 21MX Computers with the necessary logic to dynamically change the instruction set of the computer. The printed-circuit assembly and flat cable assembly contained in the interface kits are shown in figure 6-1 and listed in table 6-1.

6-2. IDENTIFICATION

Hewlett-Packard uses five digits and a letter (12978A) for standard kit designations. If the designation of your kit does not agree with this number, there are differences between your kit and the kit described in this manual.

6-3. INTERFACE KIT CONTENTS

Table 6-1. Interface Kit Contents

INTERFACE KIT	CONTENTS	HP PART NO.
12978A	Writable Control Store PCA	12908-60006
	Flat Cable Assembly 5 Connectors	5060-8393
	Microprogramming 21MX Computers	02108-90008
	Diagnostic Paper Tape	24359-16001
	Diagnostic Manual	12908-90009

6-4. CONTENTS OF INTERFACE KIT OPTIONS

There are two 12978A Interface Kit Options. They contain material in addition to that contained in the basic interface kit. Option 001 provides all the software required for use of the writable control store in the DOS-III system. Option 002 provides all the software required for the use in the BCS system.

Table 6-2. Additional Material for Interface Options

OPTION	ADDITIONAL MATERIAL	HP PART NO.
12978A-001	DOS-III WCS Driver	24278-60001
	DOS-III WCS I/O Utility	24333-60001
	DOS-III Micro-assembler	12978-16001
	Dos-III Micro Debug Editor	12978-16002
12978A-002	BCS WCS Driver	24277-60001
	BCS WCS I/O Utility	24283-60001
	BCS Micro-assembler	12978-16003
	BCS Micro Debug Editor	12978-16004

6-5. SPECIFICATIONS

Table 6-3 lists the characteristics and specifications of the writable control store PCA.

6-6. INSTALLATION

6-7. UNPACKING AND INSPECTION

If the shipping carton is damaged upon receipt, request that the carrier's agent be present when the kit is unpacked. Inspect the kit for damage (cracked, broken parts, etc.). If the kit is damaged and fails to meet specifications, notify the carrier and the nearest HP Sales and Service Office immediately. (Sales and Service Offices are listed at the back of this manual.) Retain the shipping container and the packing material for the carrier's inspection. The HP Sales and Service Office will arrange for the repair or replacement of the damaged item without waiting for any claims against the carrier to be settled.

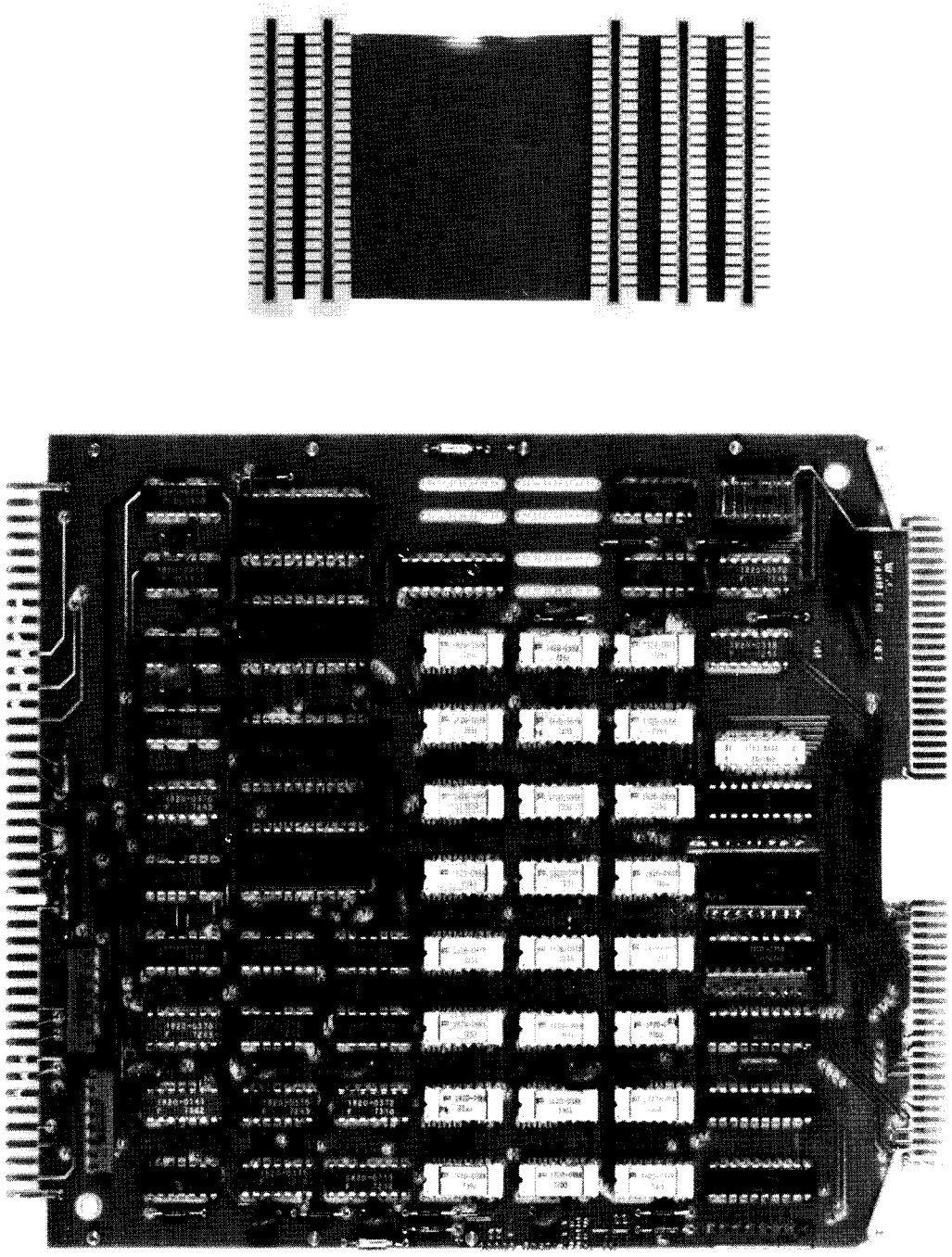


Figure 6-1. Writable Control Store Interface Kit

Table 6-3. Writable Control Store PCA Specifications

CAPACITY

Words Available: 256 per module

Maximum WCS Modules: one per HP 2105; two per HP 2108

Word Size: 24 bits

MICRO-INSTRUCTION TIME

Access: 162 ns.

Full Micro-instruction Cycle: 325 ns.

INSTALLATION

One writable control store PCA requires the use of one Input/Output slot (slot 10). Writable control store may be used as any module, except module 0.

DATA STORAGE

Input/Output Group instructions or an HP 21MX Dual Channel Port Controller are used to load the WCS.

DATA READBACK

Input/Output Group instructions only are used to read data from the WCS.

INTERFACE CURRENT SUPPLIED BY COMPUTER

0.15A (−2V supply); 4.6A (+5V supply)

PCA DIMENSIONS

Width: 7-3/4 inches (196.8 mm)

Height: 8-11/16 inches (220.7 mm)

PCA WEIGHT

Net Weight: 18 oz (511.2 gm) (card and cable only)

Shipping Weight: 4 lb (2.27 kg)

PCA INPUT LEVELS

"1" state: 1.9 volts minimum

"0" state: 1.1 volts maximum

PCA OUTPUT LEVELS

"1" state: 2.4 volts minimum

"0" state: 0.7 volts maximum

6-8. INSTALLATION

Install the writable control store kit as follows:

a. Ensure that the computer operates properly prior to installing the writable control store interface kit.

b. Turn off power at the computer.

c. Remove the bottom and back access covers from the computer.

d. On the writable control store remove the appropriate jumper wires from TB1 to select the desired module number (see figure 6-2 for pin number configuration). Refer to table 6-4 for the desired module number and jumper removal.

e. On the writable control store PCA place the WCS module 0 enable switch S1 in the OFF position.

f. Place the first writable control store PCA in slot number 10 (select code 10) of the I/O section of the computer. Any additional writable control store PCAs should be placed in slot 11.

Note: When WCS PCAs are installed, computer software must be reconfigured because of the changed I/O slot usage. If adding WCS PCA(s) will overburden the Power Supply of the computer, it may be necessary to move some I/O PCAs to an I/O Extender, HP 2156A.

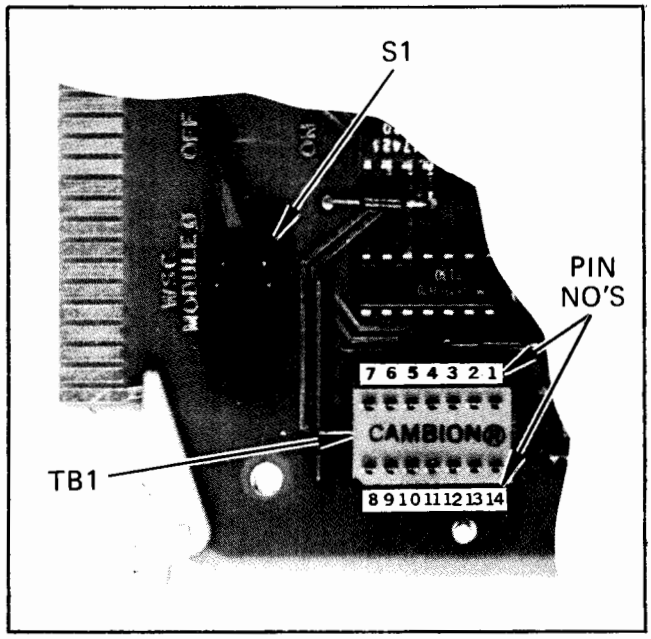


Figure 6-2. WCS Terminal Board for Selecting Module Number Position

Table 6-4. WCS PCA Jumper Removal on Terminal Board for Various Module Selections

MODULE	JUMPERS TO BE REMOVED
0	None
1	Pins 6,9
2	Pins 5,10
3	Pins 6,9; 5,10
4	Pins 4,11
5	Pins 6,9; 4,11
6	Pins 5,10; 4,11
7	Pins 6,9; 5,10; 4,11
8	Pins 3,12
9	Pins 6,9; 3,12
10	Pins 5,10; 3,12
11	Pins 6,9; 5,10;, 3,12
12	Pins 4,11; 3,12
13	Pins 6,9; 4,11; 3,12
14	Pins 5,10; 4,11; 3,12
15	Pins 6,9; 5,10; 4,11; 3,12

g. Remove the ROM-CPU Interconnect assembly, part no. 5060-8344. Install the connectors of the flat cable assembly, part no. 5060-8393

1. on J1 of the ROM Control PCA 1, A7
2. on J2 of the CPU A1
3. on J1 of each WCS PCA

as shown in sideview on figure 6-3.

Note: If an I/O PCA is installed immediately above the WCS (refer to figure 6-3) that requires a cable (hood) connector on the back, then it may be necessary to double the flat cable assembly back or cut it to make room for the I/O cable connector.

h. Replace the bottom and back access covers on the computer.

i. Turn on power at the computer and perform the diagnostic test as outlined in the Diagnostic Program Procedures (part no. 12908-90009) shipped with the 12978A Interface Kit. If the diagnostic program is completed without error, the PCA is installed and operating properly. If the diagnostic program indicates errors, halt the computer, turn off power, and recheck all of the above installation procedures. Correct where necessary, then recheck and repeat the operating procedures of the diagnostic.

6-9. RESHIPMENT

If an item of the kit is to be shipped to Hewlett-Packard for service or repair, attach a tag to the item identifying the owner and indicating the service or repair to be accomplished. Include the model number of the kit. Package the item in the original factory packaging material, if available. If the original material is not available, standard factory packaging material can be obtained from a local Hewlett-Packard Sales and Service Office. If standard factory packaging material is not used, wrap the item in Air Cap TH-240 Cushioning (or equivalent) manufactured by Sealed Air Corp., Hawthorne, N.J. and place in a corrugated carton (200 pound test material). Seal the shipping carton securely and mark it "FRAGILE" to ensure careful handling.

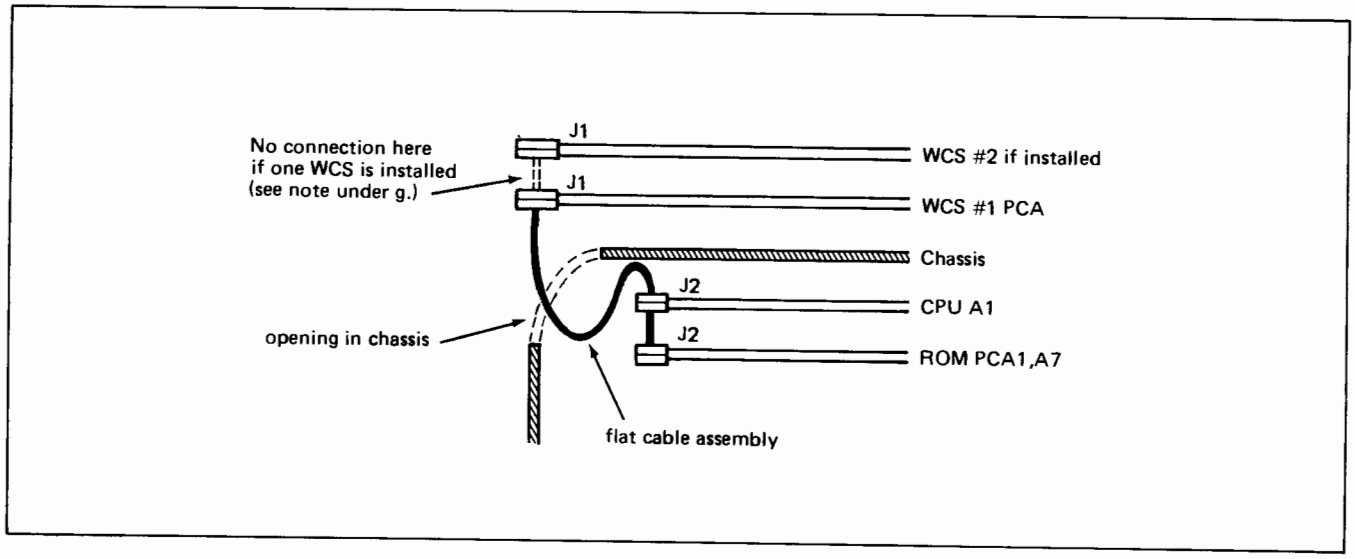


Figure 6-3. Installation of Flat Cable Assembly

Note: In any correspondence identify the kit by model number. Refer any questions to the nearest Hewlett-Packard Sales and Service Office.

6-10. PROGRAMMING

Two methods exist for writing data into (loading) a WCS module: under program control and under control of the Dual Channel Port Controller (DCPC). Under program control, prior to initiating the load routine, the data to be loaded must be stored in the computer memory. This requires a block of up to 512 words per module. The load routine will send two words from memory (32 bits which are mapped into an 8 bit address and a 24 bit micro-instruction) to the WCS module, issue a write command to that module and cause the data to be stored there. The load routine will repeat this process until the desired number of words have been stored in the WCS module.

Once loaded, the contents of the WCS module may be read back under program control via the I/O bus and compared with their counterpart in memory.

Timing sequences for flags used in the following examples are shown in figure 6-4.

6-11. PROGRAM EXAMPLE: LOADING WCS

The following is an example of the program sequence necessary for loading a WCS under program control. This example does not include block pointers, counters, etc., which are necessary for proper control.

Note: "SC" indicates select code of the WCS PCA.

STF	SC	Initializes the Direction FF (flip-flop or flag)
OTA	SC	Loads the first computer word into first WCS buffer and toggles the Direction FF. This word comprises the 8 bit address and the 8 most significant bits of the micro-instruction.
OTB	SC	Loads the second computer word into the second WCS buffer and toggles the Direction FF. This word comprises the 16 least significant bits of the micro-instruction.
STC	SC	Provides the write pulse to load the WCS buffers into the RAM.

The OTA, OTB, and STC instructions are normally in a loop that is repeated until the desired number of micro-instructions have been stored. OTA/OTB was chosen as an example; any combination of these instructions is allowable.

6-12. PROGRAM EXAMPLE: READING WCS

An example of reading from WCS under program control via the I/O bus is shown below. This example is shown without regard to the block pointers, counters, etc., which are necessary for proper control.

STF	SC	Initializes the Direction FF.
OTA	SC	Sends the 8 bit address to the WCS module from the 8 most significant bits of the A-register. (B-register could be used, as well).
STF	SC	Re-initializes the Direction FF.
LIA	SC	Places eight zeros into the 8 most significant bit positions of the A-register and places the eight most significant bits of the micro-instruction into the eight least significant bit positions of the A-register.
LIB	SC	Places the 16 least significant bits of the micro-instruction into the B-register.

The STF, OTA, STF, LIA, and LIB sequence is normally in a loop that is repeated until the desired number of micro-instructions have been read in from WCS. LIA/LIB was chosen as an example; any combination of these instructions is allowable.

6-13. PROGRAM EXAMPLE: LOADING WCS BY DUAL CHANNEL PORT CONTROLLER

Under Dual Channel Port Controller (DCPC) control, the load routine must send only the three DCPC control words to the selected channel. When the channel is turned on, DCPC will utilize every I/O cycle until the entire block of data is sent to the WCS module (maximum of 512 cycles). DCPC will transfer these words at a rate of 1.62 μs/word (512 words will take 830 μs to transfer).

The following is an example of the program sequence necessary for loading WCS via DCPC. This example does not include block pointers, counters, etc., which are necessary for proper control.

```
        LDA   CW1     Get the first DCPC control word.
        OTA   6       Send the first DCPC control word
                      to the selected DCPC channel
                      (DCPC channel 1 has been
                      selected here for demonstration
                      purposes only).
        CLC   2       Prepare the selected DCPC
                      channel to receive the second
                      DCPC control word.
        LDA   CW2     Get the second DCPC control
                      word.
        OTA   2       Send the second DCPC control
                      word to the selected DCPC
                      channel.
        STC   2       Prepare the selected DCPC chan-
                      nel to receive the third DCPC
                      word.
        LDA   CW3     Get the third DCPC control word.
        OTA   2       Send the third DCPC control
                      word to the selected DCPC
                      channel.
        STC   6,C     Turn on the selected DCPC
                      channel.
        STF   SC      Initialize the Direction FF.
        CLF   SC      Start DCPC transfer.
        SFS   6       Test for the completion of the
                      transfer.
        JMP   *-1     Loop until done.
CW1     OCT   12000SC
CW2     OCT   (Starting address of the block to be
              transferred)
CW3     OCT   (Two's complement of the number of com-
              puter words to be transferred.)
```

6-14. GENERAL THEORY OF OPERATION

Writable Control Store (WCS) consists of a bipolar semiconductor Random Access Memory (RAM) containing 24 integrated circuit (IC) packages mounted on a 2100-size printed-circuit assembly (PCA). Also included is the flat jumper cable assembly necessary for complete mechanization within the HP 21MX Computer. The WCS PCA should be installed only in slots 10 (standard) and 11 of the computer I/O slots. Each IC package is configured in 256 bits and organized as one bit per word. Thus one module of WCS is capable of storing 256 words of 24 bits each.

For the purpose of execution of WCS instructions, WCS can be configured to be addressed as any one of the computer's ROM modules except module 0. One WCS module can be installed on an HP 2105 Computer. Two WCS modules can be installed on an HP 2108 Computer.

6-15. WCS MODULE IDENTIFICATION

For proper addressing of WCS, an integrated circuit comparator and terminal board (with jumpers) on the WCS PCA is used to identify the PCA as a particular module of Control Store. For example, if the terminal board is configured for module 2, the PCA will be enabled when the ROM Address Register (RAR) contains the pattern "0010" in its four most significant bits (11-8), and disabled otherwise. When enabled, the word in WCS addressed by RAR bits 0 through 7 will be sent to the ROM Instruction Register (RIR) as signals ROM0 through ROM23. The computer will then execute this word (micro-instruction) as though it came from a standard ROM PCA. The access time of data from WCS (162 ns.) allows the computer to operate at its normal clock rate. If it is desired to replace any module already existing in ROM with a WCS module, that ROM module must be removed in order to prevent unwanted "or" conditions on the data lines.

Note: The ON position of switch S1 (figure 6-2). is not intended for use in the 21MX computers. All Control Store is disabled, if S1 is set to ON.

6-16. WCS CONNECTION

WCS is connected to the computer central processor through the I/O structure (for loading and checking), and also through a 50 wire flat cable connector. It is this connector that enables WCS to be used as an extension of the computer's basic control store. The cable connects one or two WCS PCAs to ROM control PCA 1,A7 and to CPU A1. The ROM address register on the CPU sends a 12 bit address to the WCS PCA or PCAs through this cable, and the addressed PCA then sends its data (micro-instruction) from that address back through this cable, where it is merged with the outputs of ROM. From there the data is sent to the ROM instruction register as though it was from ROM.

6-17. WCS ADDRESSING

To load the WCS RAM circuits, the WCS PCA must be addressed through the I/O interface structure of the computer. A 32 bit format is necessary and requires that a 2 word transfer be used in the loading procedure through the computer A- and/or B-registers. Two computer words and thus two transfer operations are required for one WCS word. The eight most significant bits of the first computer word transferred is the WCS RAM circuit address. The remaining eight bits of the first computer word and all 16 bits of the second computer word (total of 24 bits) are stored in WCS at the address specified.

Once loaded, WCS becomes an extension of the ROM. Thus the WCS may be used to alter the computer instruc-

tion set while the computer is in an operating condition. This feature permits dynamic expansion of the computer instruction set.

6-18. WCS Loading Timing diagram

Figure 6-4 illustrates the WCS timing.

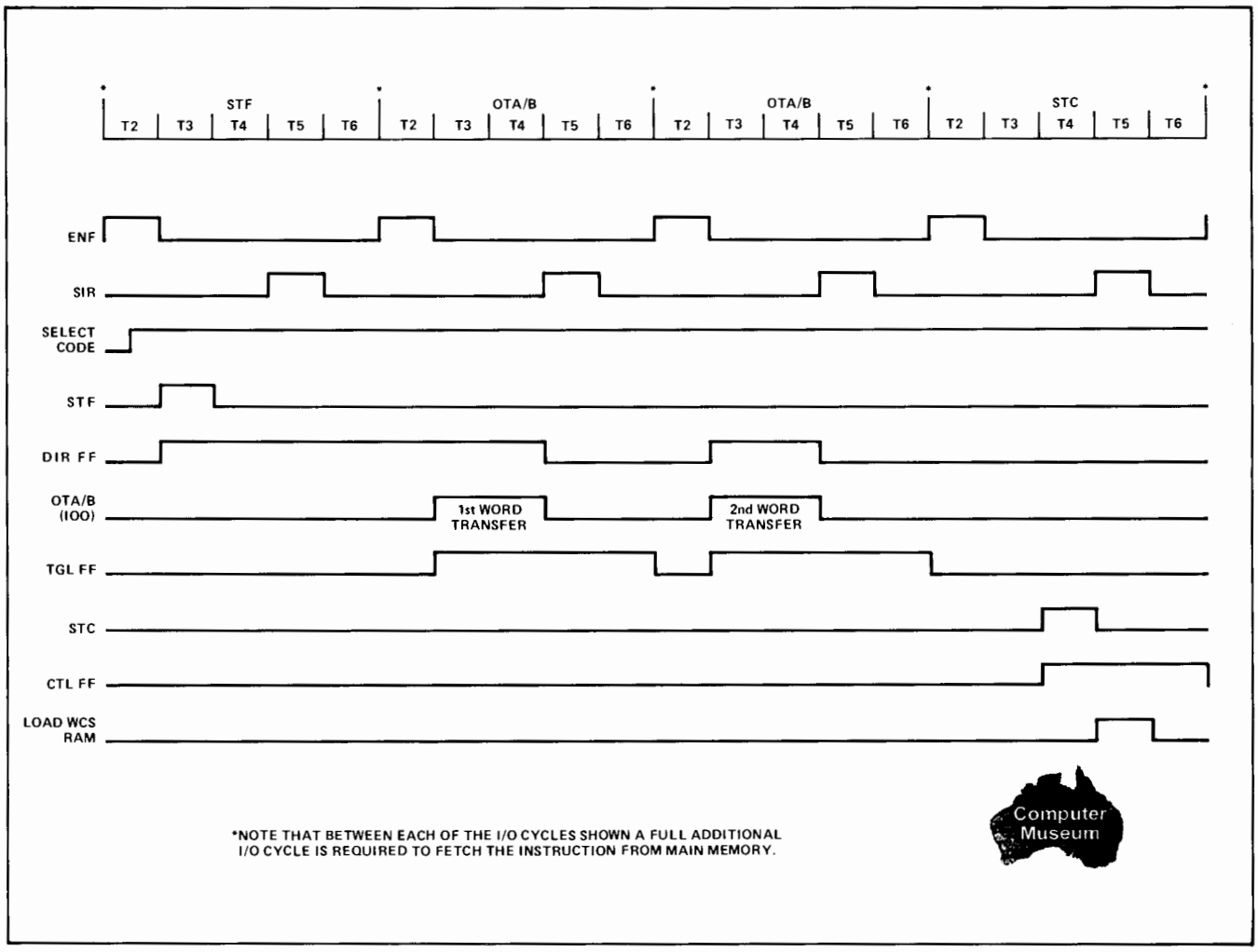


Figure 6-4. WCS Loading Timing Diagram

OBJECT TAPE FORMATS

APPENDIX A

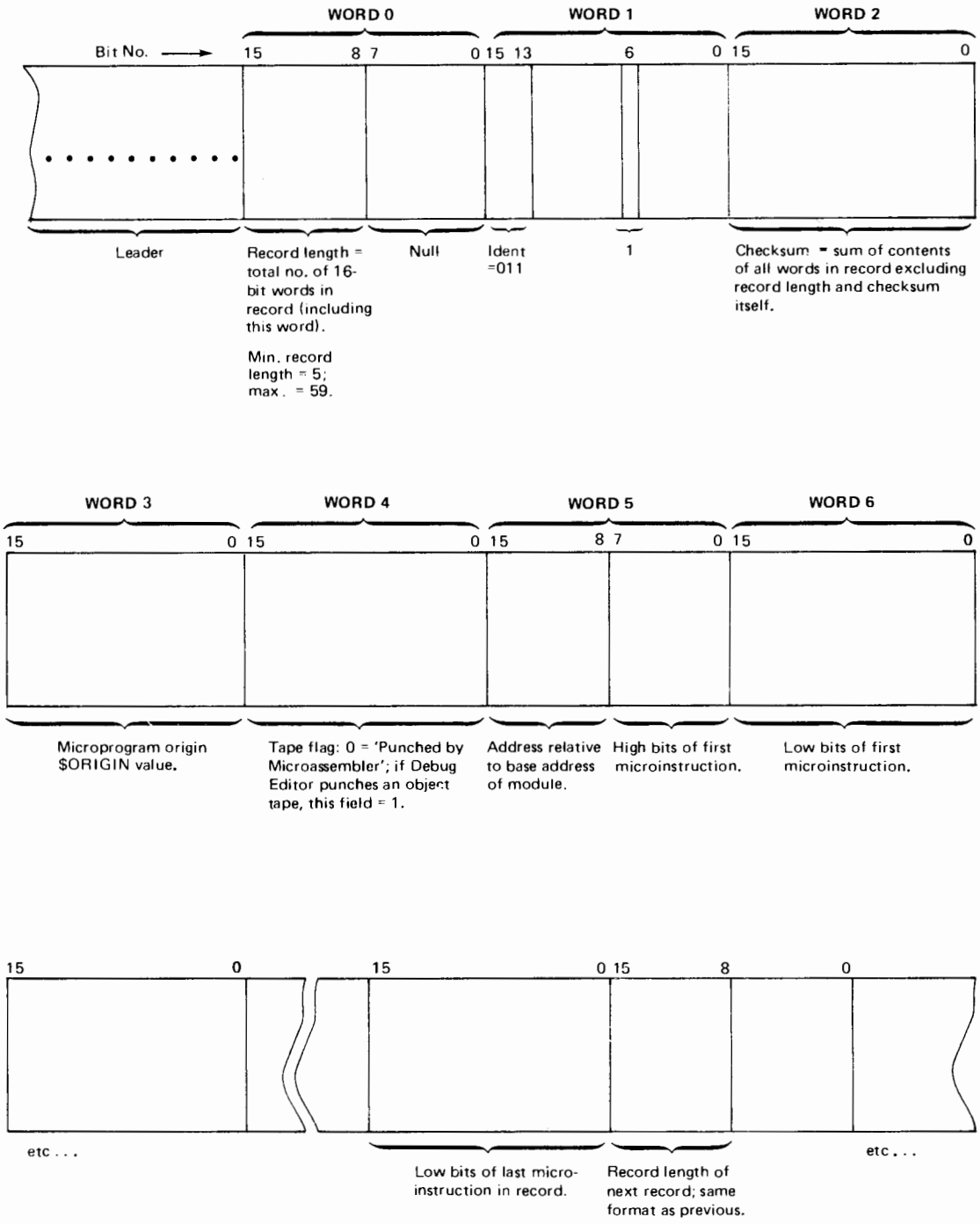


Figure A-1. Format of Standard Object Tape (Sheet 1 of 2)

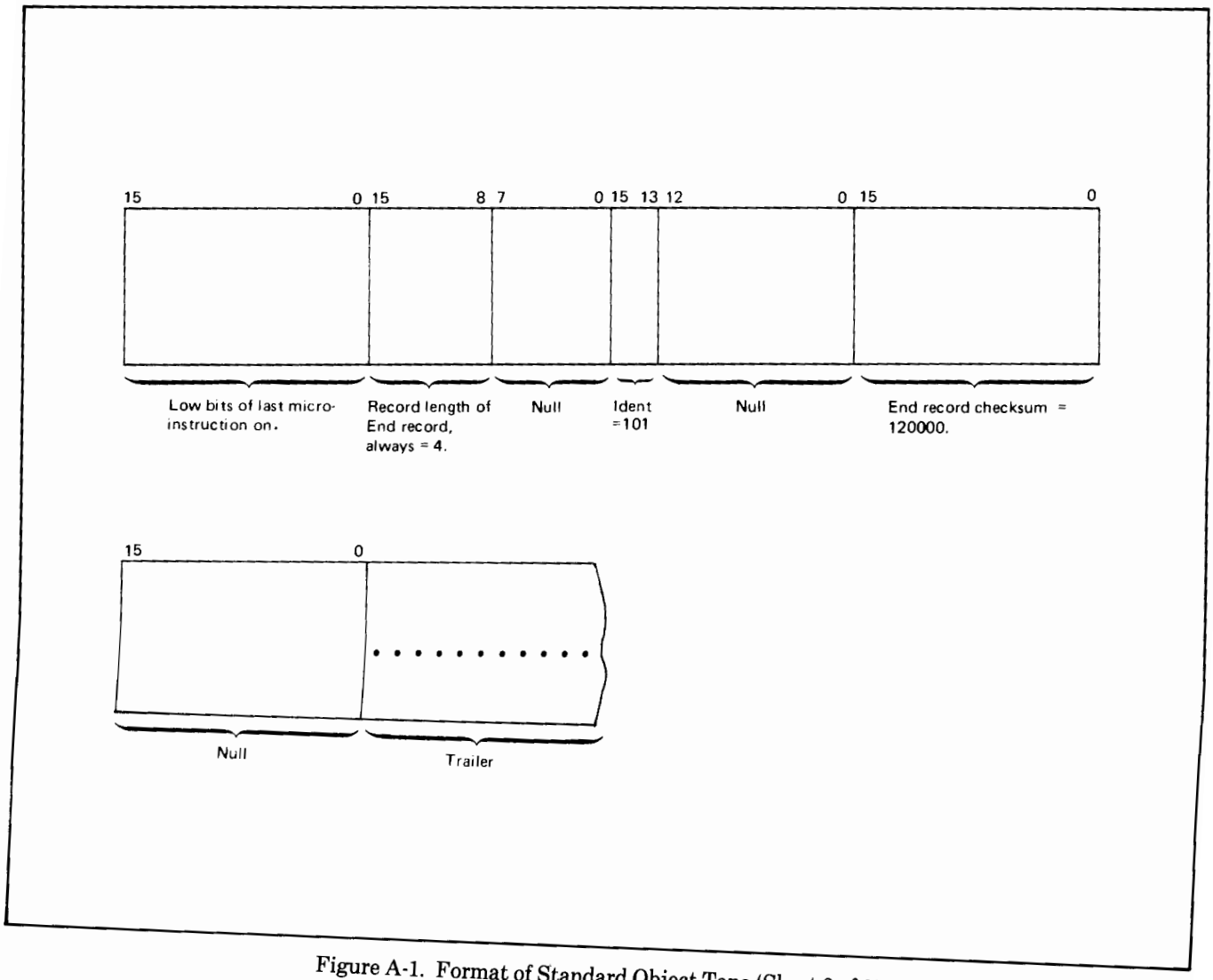


Figure A-1. Format of Standard Object Tape (Sheet 2 of 2)

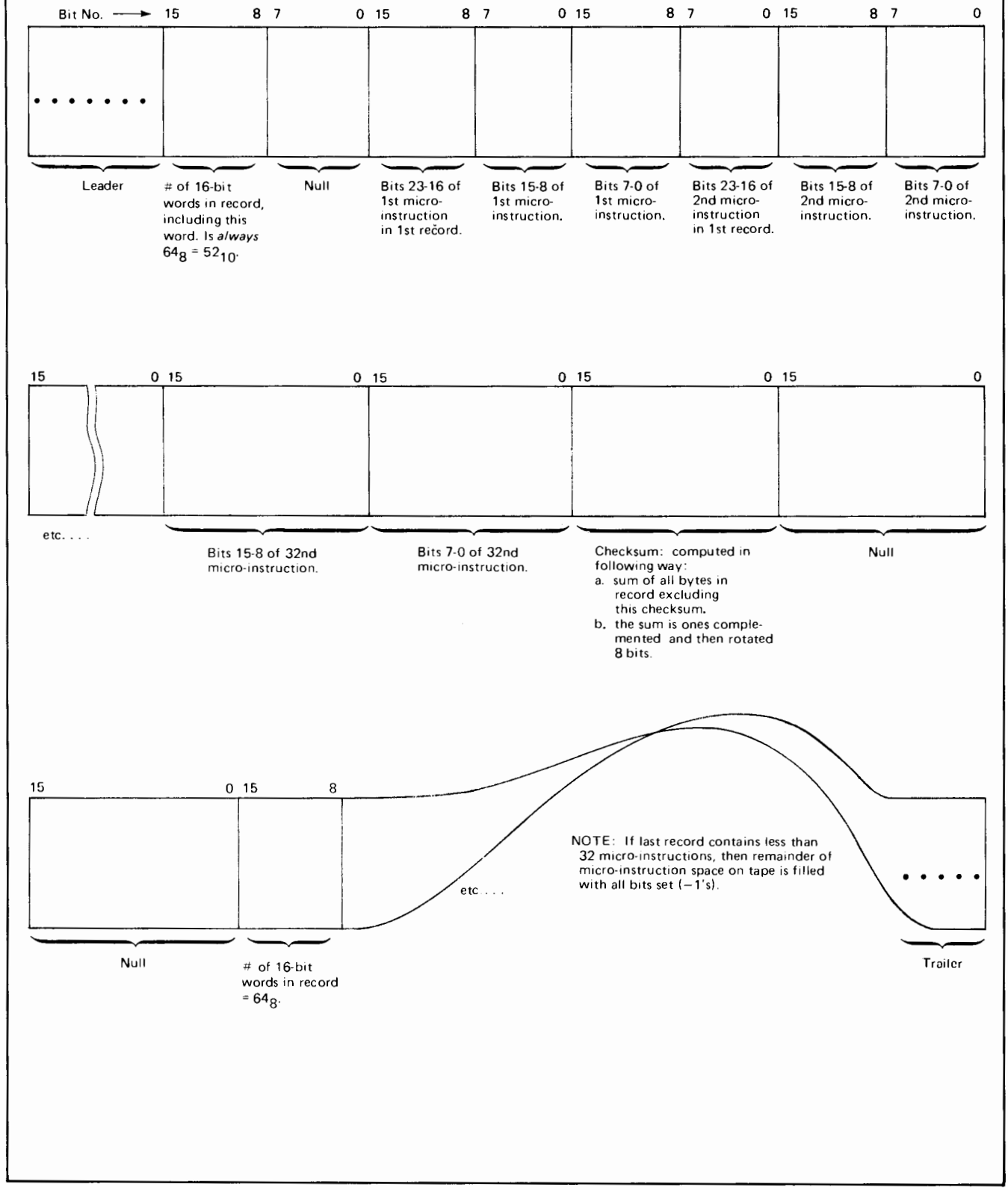


Figure A-2. Format of the $RCASE Object Tape

(Actual size: 12.5" x 10.5")

HEWLETT-PACKARD 21MX MICROCODING FORM

PROGRAMMER				DATE		MICROPROGRAM	MODULE	PAGE OF
LABEL	OP	SPECIAL	ALU	STORE	S-BUS	COMMENTS		Word Type 1
LABEL	"IMM"	SPECIAL	MODIFIER	STORE	OPERAND	COMMENTS		Word Type 2
LABEL	"JMP"	"CNDX"	CONDITION	JUMP SENSE	OPERAND	COMMENTS		Word Type 3
LABEL	"JMP" OR "JSB"	JUMP MODIFIER			OPERAND	COMMENTS		Word Type 4
1 FIELD 1	10 FIELD 2	15 FIELD 3	20 FIELD 4	25 FIELD 5	30 FIELD 6	40 FIELD 7		80

1 10 15 20 25 30 40 80

Ø = ZERO I or 1 = ONE I = ALPHA I
O = ALPHA O 2 = TWO Ƶ = ALPHA Ƶ

5951-7386

Figure B-1. Microcoding Form

MICRO-ORDER SUMMARY

APPENDIX

C

Table C-1. Summary of User Micro-orders

MICRO-ASSEMBLER →	OP	SPECIAL	ALU	JMP COND	IMMEDIATE MODIFIER	STORE	RJS	S-BUS
CARD COLUMN →	10	15	20	20	20	25	25	30
BITS (ROM) →	20-23	0-4	15-19	15-19	18-19	10-14	14	5-9
Corresponding Bit Pattern								
00000	*NOP	IOFF	INC	TBZ	HIGH	TAB	RJS	TAB
00001	ARS	SRG2	OP1	ONES	LOW	CAB		CAB
00010	CRS	L1	OP2	COUT	CMHI	T		T
00011	LGS	L4	ZERO	AL0	CML0	L		CIR
00100	MPY	R1	OP3	AL15		IOO		IOI
00101	DIV	ION	OP4	NMLS		CNTR		CNTR
00110	LWF	SRG1	SUB	CNT8		DSPL		DSPL
00111	WRTE	RES2	OP5	FPSP		DSPI		DSPI
01000	ASG	STFL	OP6	FLAG		IR		ADR
01001	READ	CLFL	ADD	E		M		M
01010	ENV	FTCH	OP7	OVFL		B		B
01011	ENVE	SOV	OP8	RUN		A		A
01100	JSB	COV	OP9	NHOI		MEU		LDR
01101	JMP	RPT	OP10	SKPF		CM		RES2
01110	IMM	SRGE	OP11	ASGN		PNM		MEU
01111	(not used)	*NOP	DEC	IR2		*NOP		*NOP
10000		MESP	CMPS	NLDR		S1		S1
10001		MPCK	NOR	NSNG		S2		S2
10010		IOG	NSAL	NINC		S3		S3
10011		ICNT	OP12	NDEC		S4		S4
10100		SHLT	NAND	NRT		S5		S5
10101		INCI	CMPL	NLT		S6		S6
10110		RES1	XOR	NSTR		S7		S7
10111		SRUN	SANL	NRST		S8		S8
11000		**UNCD	NSOL	NSTB		S9		S9
11001		CNDX	XNOR	NSFP		S10		S10
11010		JIO	PASL	INT		S11		S11
11011		JTAB	AND	SRGL		S12		S12
11100		J74	ONE	RUNE		X		X
11101		J30	SONL	*NOP		Y		Y
11110		RTN	IOR	CNT4		P		P
11111	(not used)	JEAU	*PASS	NMEU		S		S

*default micro-order

**JMP default

(10% screen) not normally used by user microprogrammer.

(20% screen) included here for completness only; reserved for exclusive use of system microprogrammers.

FUNCTIONAL BLOCK DIAGRAM

APPENDIX

D

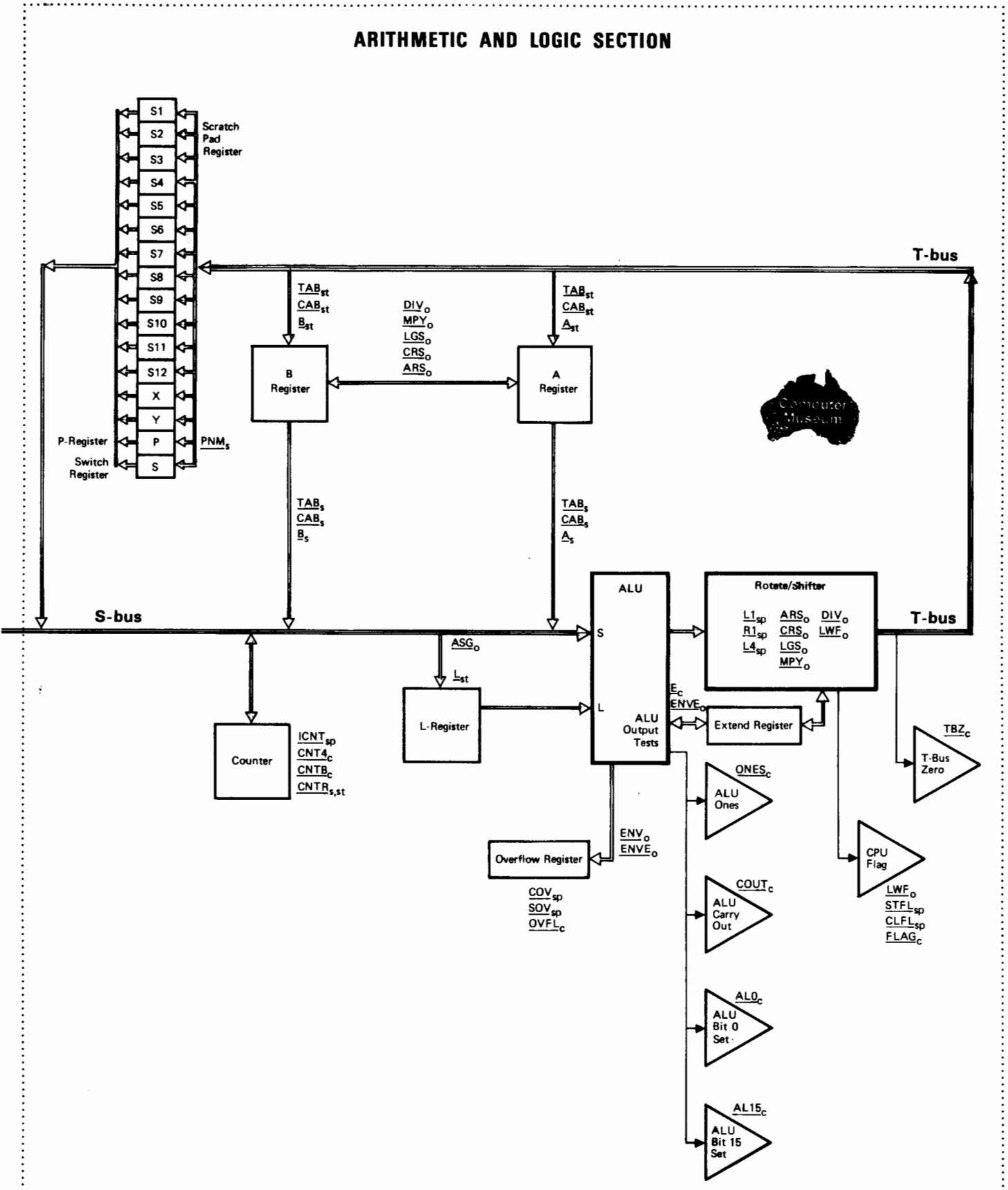
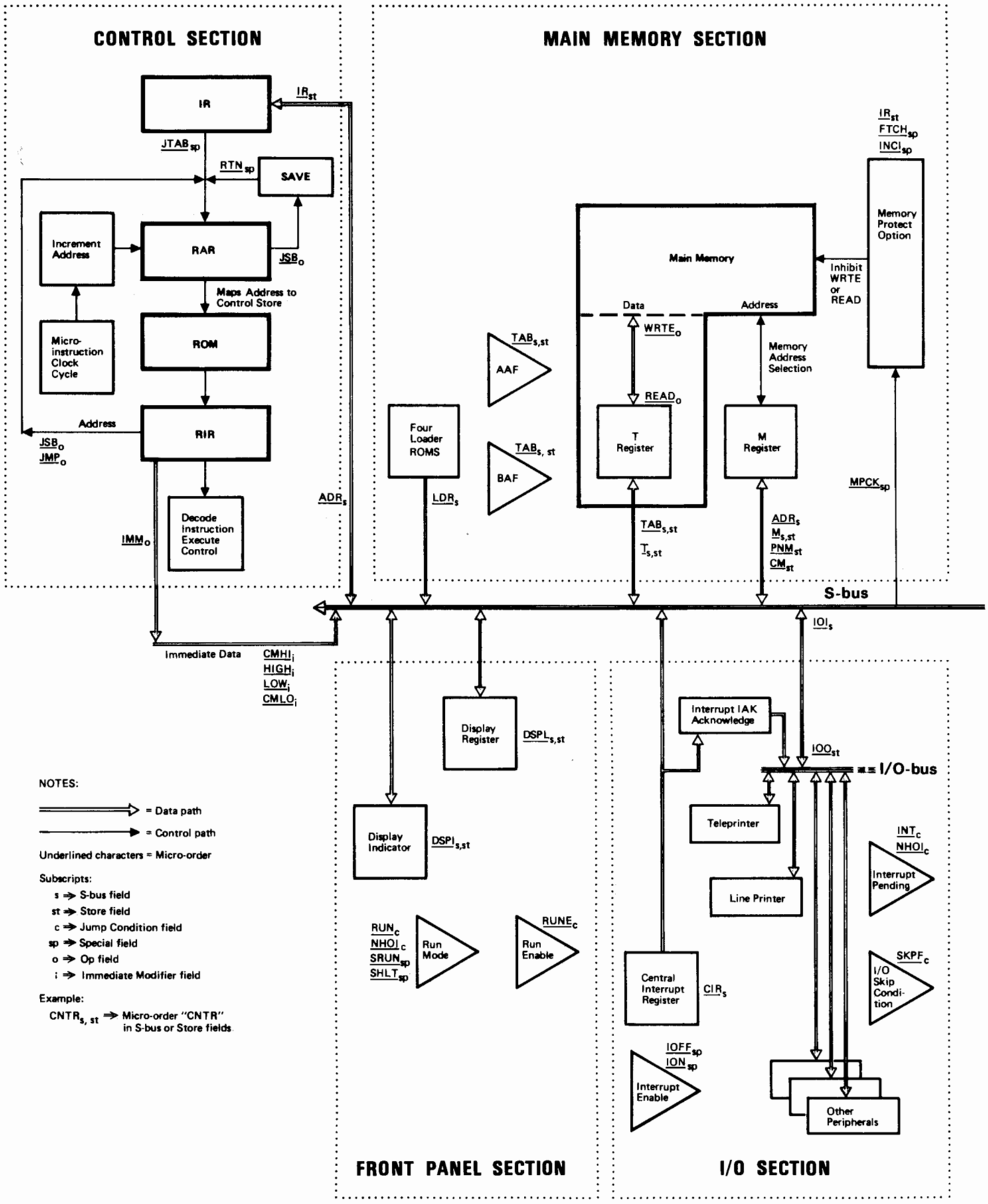


Figure D-1. Functional Block Diagram



NOTES:

= Data path

= Control path

Underlined characters = Micro-order

Subscripts:

- s ⇒ S-bus field
- st ⇒ Store field
- c ⇒ Jump Condition field
- sp ⇒ Special field
- o ⇒ Op field
- i ⇒ Immediate Modifier field

Example:

$CNTR_{s,\,st}$ ⇒ Micro-order "CNTR" in S-bus or Store fields

BASIC INSTRUCTION SET MICROPROGRAM LISTING

APPENDIX E

This appendix holds a full micro-assembly listing of the 21MX Computer Basic Instruction Set microprogram. Due to the size of this microprogram, a special micro-assembler was used. Minor differences can be seen between this micro-assembly listing and a listing produced by the micro-assembler described in this manual. The major difference to be noted is that octal numbers are preceded by a "%" symbol in this listing. Other differences are self explanatory.

```
0001                    $ORIGIN=0
0002                    ********************************************************************************
0003                    *
0004                    *       21MX MICRO-CODE
0005                    *       MODULE 0
0006                    *
0007                    ********************************************************************************
0008                    HALT    EQU                     %0400
0009                    FADD    EQU                     %7125
0010                    FSUB    EQU                     %7126
0011                    FMPY    EQU                     %7221
0012                    FDIV    EQU                     %7262
0013                    IFIX    EQU                     %7000
0014                    FLOAT   EQU                     %7025
0015                    ********************************************************************************
0016                    *       FETCH ROUTINE
0017                    ********************************************************************************
0018 0000 220 074712    FETCH   READ FTCH INC   PNM   P         M<=P; P<=P+1; READ NEW INSTR
0019 0001 017 136745            IEN                             ENABLE INTERRUPT RECOGNITION
0020 0002 017 100411            CLFL PASS IR    TAB             IR<= T/A/B; CLR FLAG FF
0021 0003 220 020673            READ JTAB INC   CM    ADR       JUMP THRU TABLE; LOAD M IF MRG INSTR
0022                    ********************************************************************************
0023 0004 325 120031    HORI    JMP  ONDX RUN   RJS   HALT      RUN MODE IMPLIES AN INTERUPT
0024                    ********************************************************************************
0025                    *       INTERRUPT RESPONSE ROUTINE
0026                    ********************************************************************************
0027 0005 237 106451    INTERUPT READ CLFL PASS M     CIR       M<=CIR; READ TRAP CELL; CLR FLAG FF
0028 0006 320 000471            JMP  ONDX TBZ   RJS   INTOK     CHECK IF CIR IS VALID
0029 0007 237 106457            READ       PASS M     CIR       M<=CIR; READ TRAP CELL
0030 0010 320 040031            JMP  ONDX TBZ         FETCH     IF NO INT BY NOW, IGNORE
0031 0011 017 104400    INTOK           IOFF PASS IR    T       IR<= TRAP CELL, DISABLE INT RECOG.
0032 0012 220 020673            READ JTAB INC   CM    ADR       JUMP THRU TABLE; LOAD M IF MRG INSTR.
0033                    ********************************************************************************
0034                    *       INDIRECT ROUTINE
0035                    ********************************************************************************
0036 0013 220 022457    INDLEVEL READ       INC   M     M       READ NEXT LEVEL
0037 0014 326 001031            JMP  ONDX NHB1 RJS   IND2      HALT OR INTERRUPT?
0038 0015 017 100465    INDIRECT        INCI PASS M     TAB     M<=T/A/B; INCR INDIRECT COUNT
0039 0016 322 040571            JMP  ONDX AL15        INDLEVEL  CHECK FOR ANOTHER LEVEL OF INDIRECT
0040 0017 220 022476            READ RTN  INC   M     M       READ EFFECTIVE ADDRESS, RETURN
0041 0020 017 100465    IND2            INCI PASS M     TAB     M<=T/A/B; INCR INDIRECT COUNT
0042 0021 330 100731            JMP  ONDX NSNG RJS   INDIRECT+1JUMP BACK FOR SINGLE INSTRUCTION
0043 0022 007 175717                    DEC  P     P       RESET P
0044 0023 320 000230            JMP                 HORI    HALT OR INTERUPT
```

```
0045                    ********************************************************************
0046                    *       ALTER-SKIP GROUP
0047                    ********************************************************************
0048 0024 017 102757    ASGNSP              PASS        CAB         SET UP SKIP TEST
0049 0025 327 042431              JMP   CNDX ASGN        ASGNSKP     JUMP IF ASG SKIP NOT MET
0050 0026 200 075717              ASG         INC   P     P           P<=P+1; ENABLE ASG HARDWARE
0051 0027 327 100031              JMP   CNDX IR2   RJS   FETCH       DONE IF NOT INA/B
0052 0030 260 002076              ENVE  RTN  INC   CA3   CAB         A/B <= A/B PLUS 1
0053                    *
0054 0031 001 136057    ASGCL*              ZERO CAB                  CLEAR  A/B REGISTER
0055 0032 327 042431              JMP   CNDX ASGN        ASGNSKP     JUMP IF ASG SKIP NOT MET
0056 0033 200 075717              ASG         INC   P     P           P<=P+1; ENABLE ASG HARDWARE
0057 0034 327 100031              JMP   CNDX IR2   RJS   FETCH       DONE IF NOT INA/B
0058 0035 260 002076              ENVE  RTN  INC   CA3   CAB         A/B <= A/B PLUS 1
0059                    *
0060 0036 010 002057    ASGCM*              CMPS CAB   CAB           A/B <= NOT A/B
0061 0037 327 042431              JMP   CNDX ASGN        ASGNSKP     JUMP IF ASG SKIP NOT MET
0062 0040 200 075717              ASG         INC   P     P           P<=P+1; ENABLE ASG HARDWARE
0063 0041 327 100031              JMP   CNDX IR2   RJS   FETCH       DONE IF NOT INA/B
0064 0042 260 002076              ENVE  RTN  INC   CA3   CAB         A/B <= A/B PLUS 1
0065                    *
0066 0043 016 036057    ASGCC*              ONE  CA3                  CLR & COMP A/B REGISTER
0067 0044 327 042431              JMP   CNDX ASGN        ASGNSKP     JUMP IF ASG SKIP NOT MET
0068 0045 200 075717              ASG         INC   P     P           P<=P+1; ENABLE ASG HARDWARE
0069 0046 327 100031              JMP   CNDX IR2   RJS   FETCH       DONE IF NOT INA/B
0070 0047 260 002076              ENVE  RTN  INC   CA3   CAB         A/B <= A/B PLUS 1
0071                    *
0072 0050 217 136757    ASGNSKP   ASG                               NO SKIP; ENABLE ASG HARDWARE
0073 0051 327 100031              JMP   CNDX IR2   RJS   FETCH       DONE IF NOT INA/B
0074 0052 260 002076              ENVE  RTN  INC   CA3   CAB         A/B <= A/B PLUS 1
0075                    ********************************************************************
0076                    *       SHIFT/ROTATE GROUP
0077                    ********************************************************************
0078 0053 017 102046    SRG               SRG1 PASS CA3   CAB         FIRST SHIFT
0079 0054 017 102056                      SRGE PASS CA3   CAB         CHECK FOR CLEAR E; SET SLA TEST
0080 0055 335 103031              JMP   CNDX SRGL RJS   *+3          SRGL IS SLA TEST
0081 0056 017 102041                      SRG2 PASS CA3   CAB         SECOND SHIFT
0082 0057 000 075736                      RTN  INC   P    P           P<=P+1, WHEN LSB = 0
0083 0060 017 102041                      SRG2 PASS CA3   CAB         SECOND SHIFT
0084 0061 017 136776    RETURN            RTN
```

```
0085                    ********************************************************************
0086                    *                 I/O GROUP
0087                    ********************************************************************
0088 0062 017 136757    IOCNTRL           NOP                       ALLOW TIME TO GET SKIP FLAG
0089 0063 326 100031              JMP   CNDX SKPF RJS   FETCH       CHECK SKIP FLAG
0090 0064 000 075736                      RTN  INC   P    P           P <= P + 1
0091 0065 017 136757                      NOP
0092                    *
0093 0066 017 102757    IO.OT*              PASS        CAB         SET UP S-BUS
0094 0067 017 102217                      PASS IOO    CAB         I/O-BUS <= A/B
0095 0070 017 102236                      RTN  PASS IOO   CAB         HOLD I/O-BUS VALID
0096 0071 017 136757                      NOP
0097                    *
0098 0072 017 136757    IO.LI*            NOP                       SYNCHRONIZE IOI PULSE
0099 0073 017 136757                      NOP
0100 0074 017 110076                      RTN  PASS CAB   IOI         A/B <= I/O-BUS
0101 0075 017 136757                      NOP
0102                    *
0103 0076 017 136757    IO.MI*            NOP                       SYNCHRONIZE IOI PULSE
0104 0077 017 102157                      PASS L      CAB         L <= A/B FOR ALU OPERATION
0105 0100 017 010076                      RTN  IOR    CAB   IOI       A/B<= (A/B) + (I/O BUS)
0106                    *
0107                    ********************************************************************
0108                    *       IO GROUP/ EAU GROUP/ MAC GROUP JUMPS
0109                    ********************************************************************
0110 0101 320 003122    IOG       JMP   IOG             IOCNTRL
0111 0102 320 015437    EAU       JMP   JEAU            EAUTABLE
0112 0103 320 016034    MAC0      JMP   J74             MACTABL0
0113 0104 320 017034    MAC1      JMP   J74             MACTABL1
```

```
0114                    ********************************************************************************
0115                    *       MEMORY REFERENCE GROUP
0116                    ********************************************************************************
0117 0105 300 000670    AND,I     JSB                           INDIRECT
0118 0106 017 126157    AND                 PASS L      A       L <= A
0119 0107 015 100576                RTN     AND  A      TAB     A <= T/A/B AND L
0120                    *
0121 0110 300 000670    CP*,I     JSB                           INDIRECT
0122 0111 017 102157    CP*                 PASS L      CAB     L <= A/B
0123 0112 013 000757                        XOR         TAB     T-BUS <= T/A/B XOR L
0124 0113 320 040031                JMP     CNDX TBZ            FETCH   JUMP TO FETCH IF EQUAL
0125 0114 000 075736                RTN     INC  P      P       P<= P+1 IF NOT EQUAL
0126                    *
0127 0115 300 000670    XOR,I     JSB                           INDIRECT
0128 0116 017 126157    XOR                 PASS L      A       L <= A
0129 0117 013 000576                RTN     XOR  A      TAB     A <= T/A/B XOR L
0130                    *
0131 0120 300 000670    IOR,I     JSB                           INDIRECT
0132 0121 017 126157    IOR                 PASS L      A
0133 0122 017 000576                RTN     IOR  A      TAB     A <= T/A/B IOR L
0134                    *
0135 0123 300 000670    ST*,I     JSB                           INDIRECT
0136 0124 017 122761    ST*               MPCK PASS     M       MEM PROTECT CHECK OF ADDRESS
0137 0125 177 102036              WRTE RTN  PASS TAB    CAB     T/A/B <= A/B; WRITE
0138                    *
0139                    *
0140 0126 300 000670    AD*,I     JSB                           INDIRECT
0141 0127 017 102157    AD*                 PASS L      CAB     L <= A/B
0142 0130 264 100076              ENVE RTN  ADD  CAB    TAB     A/B <= T/A/B PLUS L
0143                    *
0144 0131 300 000640    JSB,I     JSB  IOFF                     INDIRECT  DISABLE INTERRUPT RECOGNITION
0145 0132 017 122761    JSB               MPCK PASS     M       MEM PROTECT CHECKS THIS ADDR
0146 0133 177 174017              WRTE      PASS TAB    P       T/A/B <= RETURN ADDRESS; WRITE
0147 0134 000 023736                RTN     INC  P      M       P <= M + 1
0148                    *
0149 0135 300 000670    ISZ,I     JSB                           INDIRECT
0150 0136 017 122761    ISZ               MPCK PASS     M       MEM PROTECT CHECKS THIS ADDR
0151 0137 000 001017                        INC  S1     TAB     S1 <= T/A/B + 1
0152 0140 177 140017              WRTE      PASS TAB    S1      T <= S1; WRITE
0153 0141 320 000031              JMP   CNDX TBZ  RJS   FETCH   ZERO?  NO, DONE.
0154 0142 000 075736                RTN     INC  P      P       YES, P <= P+1
0155                    *
0156 0143 300 000670    LD*,I     JSB                           INDIRECT
0157 0144 017 100076    LD*               RTN   PASS CAB  TAB     A/B <= T/A/B
```

```
0158 0145 017 136765    JMP,I           INCI                          COUNT ONE INDIRECT LEVEL
0159 0146 017 101090                    IOFF PASS S1    TAB           DISABLE INT RECOGNITION;S1<=T/A/B
0160 0147 322 046531            JMP     CNDX AL15       JINDL         JMP IF ANOTHER LEVEL OF INDIRECT
0161 0150 017 140761                    MPCK PASS       S1            MEM PROT CHECKS DESTINATION ADDR
0162 0151 017 141736                    RTN  PASS P     S1            P <= DESTINATION ADDR
0163                    *
0164 0152 220 040457    JINDL   READ            INC  M  S1            READ NEXT LEVEL
0165 0153 326 007031            JMP     CNDX NHOI RJS   HORICK        JMP IF HALT OR INT
0166 0154 017 101025                    INCI PASS S1    TAB           S1 <= T/A/B; COUNT INDIRECT LEVEL
0167 0155 322 046531            JMP     CNDX AL15       JINDL         JMP IF ANOTHER LEVEL OF INDIRECT
0168 0156 017 140761                    MPCK PASS       S1            MEM PROT CHECKS DESTINATION ADDR
0169 0157 017 141736                    RTN  PASS P     S1            P <= DESTINATION ADDR
0170                    *
0171 0160 017 101025    HORICK          INCI PASS S1    TAB           S1 <= T/A/B;  COUNT INDIRECT LEVEL
0172 0161 330 106671            JMP     CNDX NSNG RJS   JINDL+3      JUMP BACK FOR SINGLE INSTRUCTION
0173 0162 007 175717                    DEC  P          P             RESET P
0174 0163 320 000230            JMP                     HORI          HALT OR INTERUPT
0175 0164 017 121021    JMP             MPCK PASS S1    ADR           S1<=DESTINATION ADDR; CHECK WITH M.P.
0176 0165 017 141736                    RTN  PASS P     S1            P <= DESTINATION ADDRESS
```

```
0177                    ********************************************************************************
0178                    *       EAU MICROPROGRAMS
0179                    ********************************************************************************
0180 0166 010 021017    RRR                 CMPS S1    ADR
0181 0167 000 041017                        INC  S1    S1          S1 <= TWO'S COMP OF SHIFTS
0182 0170 017 140255                  RPT   PASS CNTR S1          SET UP COUNTER FOR REPEAT
0183 0171 057 124504                  CRS   R1   PASS B    B     DOUBLE-WORD SHIFT REPEAT
0184 0172 017 136776                        RTN
0185                    *
0186 0173 010 021017    ASR                 CMPS S1    ADR
0187 0174 000 041014                  COV   INC  S1    S1          S1 <= TWO'S COMP OF SHIFTS
0188 0175 017 140255                  RPT   PASS CNTR S1          SET UP COUNTER FOR REPEAT
0189 0176 037 124504                  ARS   R1   PASS B    B     DOUBLE-WORD SHIFT REPEAT
0190 0177 017 136776                        RTN
0191                    *
0192 0200 010 021017    LSR                 CMPS S1    ADR
0193 0201 000 041017                        INC  S1    S1          S1 <= TWO'S COMP OF SHIFTS
0194 0202 017 140255                  RPT   PASS CNTR S1          SET UP COUNTER FOR REPEAT
0195 0203 077 124504                  LGS   R1   PASS B    B     DOUBLE-WORD SHIFT REPEAT
0196 0204 017 136776                        RTN
0197                    *
0198 0205 010 021017    RRL                 CMPS S1    ADR
0199 0206 000 041017                        INC  S1    S1          S1 <= TWO'S COMP OF SHIFTS
0200 0207 017 140255                  RPT   PASS CNTR S1          SET UP COUNTER FOR REPEAT
0201 0210 057 124502                  CRS   L1   PASS B    B     DOUBLE-WORD SHIFT REPEAT
0202 0211 017 136776                        RTN
0203                    *
0204 0212 010 021017    ASL                 CMPS S1    ADR
0205 0213 000 041014                  COV   INC  S1    S1          S1 <= TWO'S COMP OF SHIFTS
0206 0214 017 140255                  RPT   PASS CNTR S1          SET UP COUNTER FOR REPEAT
0207 0215 037 124502                  ARS   L1   PASS B    B     DOUBLE-WORD SHIFT REPEAT
0208 0216 017 136776                        RTN
0209                    *
0210 0217 010 021017    LSL                 CMPS S1    ADR
0211 0220 000 041017                        INC  S1    S1          S1 <= TWO'S COMP OF SHIFTS
0212 0221 017 140255                  RPT   PASS CNTR S1          SET UP COUNTER FOR REPEAT
0213 0222 077 124502                  LGS   L1   PASS B    B     DOUBLE-WORD SHIFT REPEAT
0214 0223 017 136776                        RTN
0215                    *
0216 0224 220 074457    DLD         READ          INC  M    P     READ MEMORY ADDRESS
0217 0225 300 000640                JSB   IOFF                INDIRECT  JSB TO GET M<=ADDR OF FIRST WORD
0218 0226 000 023017                              INC  S1   M     S1 <= ADDRESS OF SECOND WORD
0219 0227 017 100557                              PASS A    TAB   A <= FIRST DATA WORD
0220 0230 220 040457                READ          INC  M    S1    M<=ADDR OF SECOND WORD; READ
0221 0231 000 075717                              INC  P    P     P <= P + 1
0222 0232 017 100536                        RTN   PASS B    TAB   B <= SECOND DATA WORD
0223                    *
0224 0233 220 074457    DST         READ          INC  M    P     READ MEMORY ADDRESS
0225 0234 300 000640                JSB   IOFF                INDIRECT  JSB TO GET M <= ADDR OF FIRST WORD
0226 0235 000 023021                      MPCK    INC  S1   M     MP CHECK FIRST ADDR; S1<=SECOND ADDR
0227 0236 177 126017                WRTE          PASS TAB  A     STORE A INTO FIRST LOCATION
0228 0237 000 040461                      MPCK    INC  M    S1    MP CHECK S1; M<=S1
0229 0240 177 124017                WRTE          PASS TAB  B     STORE B INTO SECOND LOCATION
0230 0241 000 075736                        RTN   INC  P    P     UPDATE P
```

```
0231 0242 220 074457    MPY         READ          INC  M    P     M <= P; READ
0232 0243 300 000640                JSB   IOFF                INDIRECT  JSB TO GET M <= ADDR OF OPERAND
0233 0244 000 075717                              INC  P    P     UPDATE P
0234 0245 017 101057                              PASS S2   TAB   S2 <= MULTIPLIER
0235 0246 017 127114    MPYX                COV   PASS S3   A     S3<=A(MULTIPLICAND); CLEAR OVFL
0236 0247 001 136517                              ZERO B          CLEAR B FOR MULTIPLY
0237 0250 017 142157                              PASS L    S2    L <= S2 (MULTIPLIER)
0238 0251 017 124255                  RPT   PASS CNTR B         CLEAR COUNTER; SET REPEAT FF
0239 0252 104 124504                  MPY   R1   ADD  B    B     MPY STEP (X16); (B,A)<=A TIMES L PLUS B
0240 0253 017 144757                              PASS      S3    TEST MULTIPLICAND
0241 0254 322 012731                  JMP   CNDX AL15 RJS  *+2   JUMP IF POSITIVE
0242 0255 003 024517                              SUB  B    B     UNDO LAST MPY STEP IF NEGATIVE
0243 0256 017 142757                              PASS      S2    TEST MULTIPLIER
0244 0257 322 003071                  JMP   CNDX AL15 RJS  RETURN  JUMP IF POSITIVE
0245 0260 017 144157                              PASS L    S3    L <= MULTIPLICAND
0246 0261 003 024536                        RTN   SUB  B    B     B<=B MINUS L (CORRECTS FOR NEG. MULT)
0247                    *
0248                    *
```

```
0249 0262 220 074457  DIV      READ       INC  M     P          M <= P; READ
0250 0263 300 000640           JSB  IOFF             INDIRECT   JSB TO GET M <= ADDR OF OPERAND
0251 0264 000 075717                      INC  P     P          UPDATE P
0252 0265 010 001157                      CMPS S4    TAB        S4 <= DVSR(CM)::SAVE ORIG SIGN
0253 0266 010 047017                      CMPS S1    S4         S1 <= DVSR
0254 0267 322 013471           JMP  CNDX  AL15 RJS   *+2       JMP IF DVSR NEGATIVE
0255 0270 000 047017                      INC  S1    S4         S1 <= DVSR(2CM)
0256 0271 017 140157                      PASS L     S1         L <= ABS VALUE(DVSR)
0257 0272 010 025117                      CMPS S3    B          S3 <= DVNDHI(2CM)
0258 0273 322 054071           JMP  CNDX  AL15       DIVS       JMP IF DVND POSITIVE
0259 0274 017 144517                      PASS B     S3           IF DVND IS NEGATIVE...
0260 0275 010 027057                      CMPS S2    A            FORM DVND(2CM)
0261 0276 000 042557                      INC  A     S2           IN B,A-REGISTER
0262 0277 321 014071           JMP  CNDX  COUT RJS   DIVS         *
0263 0300 000 044517                      INC  B     S3           *
0264 0301 003 024753  DIVS       SOV  SUB         B          CHECK FOR DVSR TOO SMALL
0265 0302 322 000031           JMP  CNDX  AL15 RJS   FETCH      (=DVND TOO LARGE)
0266 0303 077 124502           LGS  L1    PASS B     B          SHIFT OUT SIGN BIT OF FULL WORD
0267 0304 001 137054                 COV  ZERO S2               CLEAR OVFL,S2,& CNTR
0268 0305 017 142255                 RPT  PASS CNTR S2          AND SET RPTFF
0269 0306 123 024502           DIV  L1    SUB  B     B          DIV(16X); A(=QUO(POS); B(=REM*2
0270 0307 157 144142           LWF  L1    PASS L     S3         L <= FLG <= DVND SIGN(CM)
0271 0310 010 027017                      CMPS S1    A          S1 <= QUO(CM)
0272 0311 320 155071           JMP  CNDX  ONES       QZERO      IF QUO=0,THEN NO FURTHER TESTING
0273 0312 013 047057                      XOR  S2    S4         S2(15) <= EXPECTED SIGN OF QUO
0274 0313 322 014671           JMP  CNDX  AL15 RJS   *+2        JMP IF POSITIVE WAS EXPECTED
0275 0314 000 040557                      INC  A     S1           ELSE  A <= QUO(2CM)
0276 0315 017 142157                      PASS L     S2         L(15) <= EXPECTED SIGN OF QUO
0277 0316 013 026757                      XOR        A          COMPARE TO FINAL SIGN OF QUO
0278 0317 322 015071           JMP  CNDX  AL15 RJS   *+2        JMP IF OK
0279 0320 017 136753                 SOV                        ELSE INDICATE OVERFLOW
0280 0321 017 124504  QZERO      R1   PASS B     B          B <= (REM*2)/2
0281 0322 324 040031           JMP  CNDX  FLAG       FETCH      CHECK SGN OF DVND
0282 0323 010 024517                      CMPS B     B          IF NEG,THEN FORM 2-COMP OF
0283 0324 000 024536                 RTN  INC  B     B          REM & STORE IN B
```


```
0284                  $ORIGIN=3308
0285                  ********************************************************************************
0286                  *       EAU TABLE
0287                  ********************************************************************************
0288 0330 320 007330  EAUTABLE JMP                    RRR
0289 0331 320 007570           JMP                    ASR
0290 0332 320 010030           JMP                    LSR
0291 0333 320 000030           JMP                    FETCH      ILLEGAL IR CODE FOR EAU GROUP
0292 0334 320 010270           JMP                    RRL
0293 0335 320 010530           JMP                    ASL
0294 0336 320 010770           JMP                    LSL
0295 0337 320 012130           JMP                    MPY
0296                  ********************************************************************************
0297                  *       MAC TABLE
0298                  ********************************************************************************
0299 0340 321 145270  MACTABLO JMP                    FADD       / FLOATING POINT
0300 0341 321 145330           JMP                    FSUB       / FLOATING POINT
0301 0342 321 151070           JMP                    FMPY       / FLOATING POINT
0302 0343 321 153130           JMP                    FDIV       / FLOATING POINT
0303 0344 321 140030           JMP                    IFIX       / FLOATING POINT
0304 0345 321 141270           JMP                    FLOAT      / FLOATING POINT
0305 0346 320 060030           JMP                    %1400      *** PROBABLE FUTURE HP USE
0306 0347 320 060035           JMP  J30               %1400      *** PROBABLE FUTURE HP USE
0307 0350 320 100030           JMP                    %2000      *** PROBABLE FUTURE HP USE
0308 0351 320 100035           JMP  J30               %2000      *** PROBABLE FUTURE HP USE
0309 0352 320 120030           JMP                    %2400      *** PROBABLE FUTURE HP USE
0310 0353 320 120035           JMP  J30               %2400      *** PROBABLE FUTURE HP USE
0311 0354 320 140030           JMP                    %3000      *** PROBABLE FUTURE HP USE
0312 0355 320 140035           JMP  J30               %3000      *** PROBABLE FUTURE HP USE
0313 0356 320 160030           JMP                    %3400      *** PROBABLE FUTURE HP USE
0314 0357 320 160035           JMP  J30               %3400      *** PROBABLE FUTURE HP USE
0315                  ********************************************************************************
```

```
0316 0360 321 000030  MACTABL1 JMP                    %4000     *** PROBABLE FUTURE HP USE
0317 0361 321 000035           JMP   J30              %4000     *** PROBABLE FUTURE HP USE
0318 0362 321 020030           JMP                    %4400     *** PROBABLE FUTURE HP USE
0319 0363 321 020035           JMP   J30              %4400     *** PROBABLE FUTURE HP USE
0320 0364 321 040030           JMP                    %5000     *** PROBABLE FUTURE HP USE
0321 0365 321 040035           JMP   J30              %5000     *** PROBABLE FUTURE HP USE
0322 0366 321 060030           JMP                    %5400     *** PROBABLE FUTURE HP USE
0323 0367 321 060035           JMP   J30              %5400     *** PROBABLE FUTURE HP USE
0324 0370 321 100030           JMP                    %6000     +++RESERVED FOR CUSTOMER ONLY
0325 0371 321 100035           JMP   J30              %6000     +++RESERVED FOR CUSTOMER ONLY
0326 0372 321 120030           JMP                    %6400     +++RESERVED FOR CUSTOMER ONLY
0327 0373 321 120035           JMP   J30              %6400     +++RESERVED FOR CUSTOMER ONLY
0328 0374 320 040030           JMP                    %1000     / RESERVED FOR HP USE
0329 0375 320 040035           JMP   J30              %1000     / RESERVED FOR HP USE
0330 0376 321 160035           JMP   J30              %7400     / BASE SET EXTENSION
0331 0377 321 161035           JMP   J30              %7420     / BASE SET EXTENSION
0332                  ********************************************************************************
0333                  $END
** NO ERRORS**
```

```
0001                  $ORIGIN=4008
0002                  ********************************************************************************
0003                  *
0004                  *       21MX MICRO-CODE
0005                  *       MODULE 1
0006                  *
0007                  ********************************************************************************
0008                  DISPLAYA EQU                  %376
0009                  DISPLAYT EQU                  %367
0010                  DISPLAYS EQU                  %337
0011                  INTERUPT EQU                  %0005
0012                  ********************************************************************************
0013                  *       MEMORY INITIALIZATION ROUTINE
0014                  ********************************************************************************
0015 0400 322 161171  HALT     JMP   CNDX MMLS       MGOOD     JUMP IF MEMORY NOT LOST
0016 0401 341 004617           IMM        HIGH MEU   %102      ENABLE SYSTEM MAP
0017 0402 347 101017           IMM        LOW  S1    %340      S1 <= 2'S COMP OF 32
0018 0403 001 137057                      ZERO S2               CLR S2 (MAP ADDR)
0019 0404 017 142357                      PASS A     S2         CLR A-REG
0020 0405 017 142517                      PASS B     S2         CLR B-REG
0021 0406 017 142117                      PASS T     S2         CLR T REG
0022 0407 353 077117           IMM        CMM1 S3    %337      S3 <= "LOAD ADDR REG" COMMAND
0023 0410 347 076264  LOSTLOOP IMM   SHLT LOW  CNTR  %337      CNTR<=COMP OF 32; CLEAR RUN FF
0024 0411 017 144617                      PASS MEU   S3         LOAD 0 INTO ADDR REG ON MEU
0025 0412 017 142620  MAPLOOP        MESP PASS MEU   S2         LOAD MAP IN MEU
0026 0413 000 043063                 ICNT INC  S2    S2        INC MAP ADDR
0027 0414 323 020531           JMP   CNDX CNTB RJS   MAPLOOP   LOOP(*32)
0028 0415 001 137717                      ZERO P                CLR P REG
0029 0416 160 074717           WRTE       INC  PHM   P         M(=P; P(=P+1; WRITE ZERO DATA
0030 0417 322 020731           JMP   CNDX AL15 RJS   *-1       LOOP UNTIL M=077777
0031 0420 000 041017                 INC  S1    S1        INC MAP CNTR
0032 0421 320 020431           JMP   CNDX TBZ  RJS   LOSTLOOP  LOOP (*32)
0033 0422 341 000617           IMM        HIGH MEU   %100      DISABLE ALL MAPS NOW...
0034                  ********************************************************************************
0035                  *       FRONT PANEL STANDARD SCAN ROUTINES
0036                  ********************************************************************************
0037 0423 334 165531  MGOOD    JMP   CNDX NSFP       CONTFP    JUMP IF   NON-STANDARD FRONT PANEL
0038 0424 017 115752                 FTCH PASS S     DSPL      S(=DISPLAY;INITIALIZE MEM. PROTECT
0039 0425 330 121371           JMP   CNDX NSNG RJS   WAIT      JUMP IF "INSTR STEP" PRESSED
0040 0426 347 156357           IMM        LOW  DSPI  DISPLAYT  ACTIVATE "T" INDICATOR IN DSPI
0041 0427 300 024270  WAIT     JSB                   UPDATE    UPDATE DISPLAY WITH PROPER DATA
0042 0430 334 021431           JMP   CNDX NSTB RJS   *         WAIT FOR BUTTON RELEASES
0043 0431 325 164231           JMP   CNDX RUN        RUN
0044 0432 334 061471           JMP   CNDX NSTB       *-1
0045 0433 017 136757  SCAN     NOP                             SCAN FOR SWITCH PRESSED
0046                  *                                       NOP ONE CYCLE TO SET SWITCH CONDITIONS
0047 0434 332 122571           JMP   CNDX NLT  RJS   LEFT
0048 0435 331 023431           JMP   CNDX NINC RJS   INC.M
0049 0436 331 123531           JMP   CNDX NDEC RJS   DEC.M
0050 0437 333 025471           JMP   CNDX NSTR RJS   STOREX
0051 0440 333 121371           JMP   CNDX NRST RJS   WAIT
0052 0441 332 032231  SCANRT   JMP   CNDX NRT  RJS   RIGHTR    JUMP IF "RIGHT" TO TEST FOR ENTRY
0053                  *                                       INTO SPECIAL DISPLAY ROUTINE.
```

```
0054 0442 330 026171          JMP   CNDX NLDR RJS   LOADER
0055 0443 325 164231          JMP   CNDX RUN        RUN
0056 0444 330 161431          JMP   CNDX NSNG       WAIT+1     JMP IF "INSTR STEP" NOT PRESSED
0057 0445 335 040271          JMP   CNDX INT        INTERUPT   SERVICE ANY PENDING INTERRUPT
0058 0446 017 176317                PASS DSPL S     S          DISPLAY <= S
0059 0447 220 074712          READ  FTCH INC  PHM   P          DO STANDARD FETCH ROUTINE
0060 0450 017 136745                ION
0061 0451 017 100411                CLFL PASS IR    TAB
0062 0452 220 020673          READ  JTAB INC  CM    ADR
0063                    ********************************************************************
0064                    *       DISPLAY INDICATOR SHIFT ROUTINES
0065                    ********************************************************************
0066 0453 017 117004  LEFT          R1    PASS S1   DSPI       S1<=DSPI SHIFTED RIGHT ONE
0067 0454 321 122771          JMP   CNDX ALO  RJS   LEFTA      JUMP IF DSPI WRAP-AROUND REQUIRED
0068 0455 017 140357  LEFTB               PASS DSPI S1         DSPI <= DSPI SHIFTED RIGHT ONE
0069 0456 320 021370          JMP                   WAIT       JUMP TO STANDARD SCAN ROUTINES
0070 0457 347 076357  LEFTA   IMM         LOW   DSPI DISPLAYS  DSPI WRAP-AROUND A TO S
0071 0460 320 021370          JMP                   WAIT       JUMP TO STANDARD SCAN ROUTINES
0072 0461 347 076157  RIGHT   IMM         LOW   L    DISPLAYS
0073 0462 017 016750                STFL IOR        DSPI       SET FLAG; TEST DSPI
0074 0463 320 123331          JMP   CNDX ONES RJS   RIGHTA    JUMP IF WRAP-AROUND OF DSPI READ
0075 0464 157 117002          LWF   L1    PASS S1   DSPI       S1<=DSPI SHIFTED LEFT ONE
0076 0465 320 022670          JMP                   LEFTB
0077 0466 347 174357  RIGHTA   IMM        LOW   DSPI DISPLAYA  DSPI WRAP-AROUND S TO A
0078 0467 320 021370          JMP                   WAIT       JUMP TO STANDARD SCAN ROUTINES
0079                    ********************************************************************
0080                    *       INC M, DEC M ROUTINES
0081                    ********************************************************************
0082 0470 000 023017  INC.M               INC   S1   M          S1 <= M + 1
0083 0471 320 023570          JMP                   DEC.M+1
0084 0472 007 123017  DEC.M               DEC   S1   M          S1 <= M - 1
0085 0473 017 140457                PASS M    S1         M <= S1
0086 0474 320 021370          JMP   UNCD            WAIT       JUMP TO STANDARD SCAN ROUTINE
0087                    ********************************************************************
0088                    *       SPECIAL TEST TO EXIT SPECIAL DISPLAY LOOP
0089                    ********************************************************************
0090 0475 017 116417  LEFTR               PASS IR   DSPI       CHECK FOR "M" DSPI
0091 0476 327 122571          JMP   CNDX IR2  RJS   LEFT       JUMP IF "M" TO LEAVE SPECIAL CODE.
0092 0477 347 166357          IMM         LOW   DSPI %373      DSPI <= "M" (SHIFT FROM "T")
0093 0500 017 142317                PASS DSPL S2        SHOW POINTER ON DISPLAY
0094 0501 320 033130          JMP   UNCD            WAITR      WAIT FOR BUTTON RELEASE IN SPECIAL CODE
```

```
0095                    ********************************************************************
0096                    *       STORE AND UPDATE ROUTINES
0097                    ********************************************************************
0098                    *                     THE REGISTER INDICATED IN DSPI IS THE BIT POSITION WHICH IS
0099                    *                     LOW. ALL OTHER BITS ARE 1.  THE ORDER (MSB TO LSB) IS
0100                    *                                  S  P  T  M  B  A
0101                    *                     THE INDICATED REGISTER IS DETERMINED BY LOADING DSPI INTO
0102                    *                     THE IR, AND JUMPING USING J3D TO GET TO THE APPROPRIATE
0103                    *                     STORE OR UPDATE ROUTINE.  OTHER CODE IS INTERSPERSED
0104                    *                     FOR MAXIMUM CONTROL STORE EFFICIENCY
0105 0502 017 116417  STORE               PASS IR   DSPI
0106 0503 320 024035          JMP   J3D             %0500      JMP TO STORE SELECTED REGISTER
0107 0504 347 076357  RUN     IMM         LOW   DSPI DISPLAYS  DSPI <= "S".   THE SAVE REGISTER IS
0108                    *                                      ZERO AT THIS POINT SO THE NEXT RTN
0109                    *                                      WILL INITIATE THE FETCH ROUTINE
0110                    ********************************************************************
0111 0505 017 116417  UPDATE              PASS IR   DSPI
0112 0506 320 025035          JMP   J3D             %520       JMP TO DISPLAY SELECTED REGISTER
0113                    ********************************************************************
0114 0507 177 114017          WRTE        PASS TAB  DSPL       STORE T
0115 0510 000 023017                      INC   S1  M
0116 0511 000 040457                      INC   M   S1         INCREMENT M, SET TAB LOGIC
0117 0512 320 021430          JMP   UNCD            WAIT+1
0118 0513 000 014476                RTN   INC   M   DSPL       STORE M
0119 0514 017 115776  STORES        RTN   PASS S    DSPL       STORE S
0120 0515 017 114536                RTN   PASS B    DSPL       STORE B
0121 0516 017 114576                RTN   PASS A    DSPL       STORE A
0122 0517 347 136157          IMM         LOW   L   %357       P OR S TO BE DISPLAYED
0123 0520 017 016757                      IOR       DSPI       MASK OUT "S"
0124 0521 320 164631          JMP   CNDX ONES       STORES     JUMP IF "S" INDICATED
0125 0522 017 115736                RTN   PASS P    DSPL       STORE P
```

```
0126                    ********************************************************************************
0127                    ****** OVFL REG. STORE--PART OF SPECIAL DISPLAY ROUTINES ***************
0128 0523 017 115013    STORDO      SOV  PASS S1   DSPL        CHECK DISPLAY
0129 0524 321 165331            JMP  CNDX AL0       *+2
0130 0525 017 136754             COV                           CLEAR OVERFLOW
0131 0526 017 136776             RTN
0132                    ********************************************************************************
0133 0527 220 022457            READ       INC  M     M        UPDATE T, READ M, SET TAB LOGIC
0134 0530 017 100336                  RTN  PASS DSPL TAB       DSPL <= MEM DATA
0135                    ********************************************************************************
0136 0531 300 024130    STOREX  JSB                  STORE     STORE ROUTINES END WITH RTN
0137 0532 320 021370    CONTFP  JMP                  WAIT      JUMP TO STANDARD SCAN ROUTINES
0138                    ********************************************************************************
0139 0533 017 122336                  RTN  PASS DSPL M         UPDATE M
0140 0534 017 176336    UPDATES       RTN  PASS DSPL S         UPDATE S
0141 0535 017 124336                  RTN  PASS DSPL B         UPDATE B
0142 0536 017 126336                  RTN  PASS DSPL A         UPDATE A
0143 0537 347 136157            IMM        LOW  L     357B      P OR S INDICATED
0144 0540 017 016757                       IOR        DSPI      MASK OUT "S"
0145 0541 320 165631            JMP  CNDX ONES       UPDATES
0146 0542 017 174336                  RTN  PASS DSPL P         UPDATE P
```

```
0147                    ********************************************************************************
0148                    *       21MX ROM BOOTSTRAP MEMORY LOADER ROUTINE                               *
0149                    ********************************************************************************
0150 0543 341 177053    LOADER  IMM  SOV  HIGH S2   %177      FORM 0111111111111111 (MAX ADDR)
0151 0544 353 137017            IMM       CMMI S1   %357      FORM 0001000000000000 (10K) IN S1
0152                    ****** DETERMINE MEMORY SIZE, STARTING ADDR FOR LOADER ******************
0153 0545 347 000157    SIZE    IMM       LOW  L    %300      FORM 1111111111000000 IN L
0154 0546 015 143717                      AND  P    S2        FORM STARTING ADDR IN P
0155 0547 010 075217                      CMPS S5   P         FORM TWO'S COMP
0156 0550 000 051217                      INC  S5   S5        OF SA IN S5
0157 0551 017 142457                      PASS M    S2        PUT LAST ADDR INTO M
0158 0552 320 161371                 JMP  CNDX ONES  WAIT     TEST FOR NO READ/WRTE CAPABILITY
0159 0553 177 150117                 WRTE      PASS T    S5    PASS INTO T
0160 0554 017 140157                      PASS L    S1        UPDATE LAST ADDR WHILE WAITING
0161 0555 223 043057                 READ      SUB  S2   S2    TO RETRIEVE DATA
0162 0556 017 150157                      PASS L    S5        COMPARE WHAT WAS READ FROM MEM.
0163 0557 013 004757                      XOR       T         TO DATA WRITTEN (S3)
0164 0560 320 026271                 JMP  CNDX TBZ  RJS  SIZE  IF IT CHECKS, WE HAVE CORRECT STRT ADDR
0165                    ****** CHECK SELECT CODE IN  S REG.     **********************************
0166 0561 347 000157            IMM       LOW  L    %300      FORM 1111111111000000 IN L
0167 0562 347 164257            IMM       LOW  CNTR %372      CNTR GETS -6
0168 0563 017 176417                      PASS IR   S         SET UP LOADER SELECT BIT
0169 0564 017 177155                 RPT  PASS S4   S         SET UP S-REG FOR SHIFT
0170 0565 017 147144                 R1   PASS S4   S4        SHIFT SELECT CODE INTO BITS(0-5)
0171 0566 013 147157                      3ANL S4   S4        MASK OFF SEL. CODE
0172 0567 347 160157            IMM       LOW  L    %370      FORM 1111111111111000 (=-10B) IN L
0173 0570 004 147153                 SOV  ADD  S4   S4        SUB 10B FROM SEL CODE; SAVE IN SJ
0174 0571 322 061371            JMP  CNDX AL15  WAIT         IF NEG RESULT, SCB < 10B; RTN W/ OVF ON
0175                    ****** PREPARE FOR LOADER TRANSFER **************************************
0176 0572 344 000257            IMM       LOW  CNTR %0        CLEAR CNTR (ROM ADDR REG)
0177 0573 017 174454                 COV  PASS M    P         PUT SA IN M;CLR OVF = NO OPER ERR
0178                    ****** TRANSFER CONTENTS OF LOADER ROM TO MEMORY ************************
0179 0574 017 131003    LOOP1        L4   PASS S1   LDR       PASS XXXXXXXXAAAAXXXX INTO S1;CNTR=X00
0180 0575 017 140163                 ICNT PASS L    S1        CNTR=X01
0181 0576 015 131003                 L4   AND  S1   LDR       FORM XXXXAAAABBBBXXXX IN S1;CNTR=X01
0182 0577 017 140163                 ICNT PASS L    S1        CNTR=X10
0183 0600 015 131003                 L4   AND  S1   LDR       FORM AAAABBBBCCCCXXXX IN S1;CNTR=X10
0184 0601 017 140163                 ICNT PASS L    S1        CNTR=X11
0185 0602 012 031017                      NAND S1   LDR       FORM AAAABBBBCCCCDDDD (CMPL FORM)
0186 0603 177 140117            WRTE      PASS T    S1        WRITE INTO MEMORY
0187 0604 000 023063                 ICNT INC  S2   M         UPDATE MEM ADDR;CNTR=X00
0188 0605 017 142457                      PASS M    S2        PASS NEW ADDR INTO M
0189 0606 344 000157            IMM       LOW  L    %0        FORM 1111111100000000 IN L
0190 0607 017 022757                      IOR       M         MASK M TO SEE IF LAST WORD OF LDR
0191 0610 320 127631            JMP  CNDX ONES RJS  LOOP1     IF M(0-8)=11111111, DON'T LOOP
```

```
0192                    ********************************************************************************
0193 0611 347 000257            IMM       LOW  CNTR %300      SET UP COUNT TO FIND LAST WORD
0194 0612 344 077110            IMM  STFL LOW  S3   %037
0195 0613 157 145102            LWF  L1   PASS S3   S3        FORM 1111111000111111 IN S3
0196 0614 017 175017                      PASS S1   P         PASS SA INTO S1
0197                    ****** CHECK INSTRUCTION IN MEMORY FOR I/O TYPE ************************
0198 0615 237 140457 NUWRD      READ      PASS M    S1        PASS SA INTO M & READ FIRST INSTR
0199 0616 340 026157            IMM       HIGH L    %013      FORM COMP OF 1111010000000000 IN L
0200 0617 017 105057                      PASS S2   T         SAVE WORD IN S2
0201 0620 013 143017                      SANL S1   S2        MASK UPPER BITS FOR I/O TYPE
0202 0621 341 166157            IMM       HIGH L    %173      FORM 0111101111111111 IN L
0203 0622 013 040757                      XOR       S1        NOW CHECK FOR I/O TYPE
0204 0623 320 171531            JMP  CNDX ONES      HTST      IF MATCH OCCURS, JUMP OUT OF LOOP
0205                    ********************************************************************************
0206 0624 000 023023 UPDT           ICNT INC  S1    M         OTHERWISE UPDATE M IN S1
0207 0625 323 030671            JMP  CNDX CNT8 RJS  NUWRD     LOOP BACK
0208 0626 017 146154                 COV  PASS L    S4        PASS (SCB-10B) INTO L
0209 0627 004 143051                 CLFL ADD  S2   S2        CHNG SC OF DCPC CNTRL WORD
0210 0630 177 142117            WRTE      PASS T    S2        SAVE IN MEM
0211 0631 320 021370            JMP                 WAIT      RETURN TO SCAN ROUTINE
0212                    ****** UPDATE SELECT CODE IN I/O INSTRUCTION ********************************
0213 0632 017 144157 HTST               PASS L    S3        PASS 1111111000111111 INTO L
0214 0633 014 042757                    NSOL      S2        BLEND TO CHECK FOR...000...OF HLT
0215 0634 320 171231            JMP  CNDX ONES      UPDT      IF FOUND GET NEXT INSTR
0216 0635 347 016157            IMM       LOW  L    %307      FORM 1111111111000111 IN L
0217 0636 013 142757                      SANL      S2        MASK BITS TO CHECK FOR SC < 10B
0218 0637 320 071231            JMP  CNDX TBZ       UPDT      IF SO, RTN TO LOOP
0219 0640 017 146157                      PASS L    S4        PASS (SCB-10B) INTO L
0220 0641 004 143057                      ADD  S2   S2        ADD TO SC FROM INSTR
0221 0642 177 142117            WRTE      PASS T    S2        PASS INTO T AND WRITE INTO MEMORY
0222 0643 320 031230            JMP                 UPDT      RTN TO LOOP
```

```
0223                    ********************************************************************************
0224                    *       SPECIAL DISPLAY ROUTINES
0225                    ********************************************************************************
0226 0644 017 116417 RIGHTR             PASS IR   DSPI      "RIGHT" PRESSED: IR <= DSPI
0227 0645 327 163071            JMP  CNDX IR2       RIGHT     JUMP IF M NOT SELECTED BY DSPI
0228 0646 017 115057                      PASS S2   DSPL      S2 <= DSPL (POINTER)
0229 0647 322 023071            JMP  CNDX AL15 RJS  RIGHT     JMP IF DSPL BIT 15 WASNT SET
0230 0650 347 156357            IMM       LOW  DSPI %367      DSPI <= "T"
0231                    ********************************************************************************
0232 0651 006 042157 UPDATR             OP9  L    S2        CHECK DSPL BIT 14, STORE S2 IN L
0233 0652 322 074671            JMP  CNDX AL15      NEWMAPS   JUMP IF S2 BIT 14 = 1 TO UPDATE NEW
0234 0653 350 007003            IMM  L4   CMHI S1   %003      S1 <= MASK FOR REGISTERS = 140017B
0235 0654 015 141057                      AND  S2   S1        S2 <= S2 MASK OUT UNUSED BITS
0236 0655 017 142417                      PASS IR   S2        SET REGISTER SELECTION
0237 0656 333 073071            JMP  CNDX NSTR      READREG   JUMP IF STORE BUTTON NOT PRESSED
0238 0657 300 037035            JSB  J30            STOREG    SELECTED REGISTER <= DISPLAY
0239 0660 320 033130            JMP  UNCD           WAITR     WAIT FOR NEXT BUTTON
0240 0661 300 036035 READREG    JSB  J30            DSPLREG   DISPLAY <= SELECTED REGISTER
0241                    ********************************************************************************
0242 0662 334 033131 WAITR      JMP  CNDX NSTB RJS  *         WAIT FOR BUTTON RELEASE
0243 0663 325 164231            JMP  CNDX RUN       RUN       JUMP IF RUN INDICATOR LIT
0244 0664 334 073171            JMP  CNDX NSTB      *-1       JUMP BACK IF NO BUTTON PRESSED
0245                    *
0246 0665 017 136757            NOP                           WAIT ONE CYCLE FOR SETTING SWITCH CONDIT
0247 0666 332 123671            JMP  CNDX NLT  RJS  LEFTR     JUMP IF "LEFT" PRESSED
0248 0667 331 073531            JMP  CNDX NINC      NOTINC    JUMP IF "INCM" NOT PRESSED
0249 0670 000 043057                 INC  S2   S2   S2        INCREMENT POINTER
0250 0671 320 034030            JMP  UNCD           DECMR+1
0251 0672 331 174231 NOTINC     JMP  CNDX NDEC      NOTDEC
0252 0673 340 000157            IMM       HIGH L    %000      CHECK FOR
0253 0674 015 142757                      AND       S2        DECREMENT OF
0254 0675 320 033771            JMP  CNDX TBZ  RJS  DECMR     ZERO COUNT
0255 0676 000 143057                      OPI  S2   S2        S2 OR L PLUS 1 (WRAP AROUND COUNT + 1)
0256 0677 007 143057 DECMR                DEC  S2   S2        DECREMENT POINTER
0257 0700 017 116417                      PASS IR   DSPI      IR <= DSPI
0258 0701 327 172471            JMP  CNDX IR2       UPDATR    JUMP IF M NOT INDICATED
0259 0702 017 142317                      PASS DSPL S2        UPDATE DISPLAY
0260 0703 320 033130            JMP  UNCD           WAITR     WITH NEW POINTER VALUE AND JUMP
0261 0704 333 134031 NOTDEC     JMP  CNDX NRST RJS  DECMR+1   JUMP IF "DISPLAY" PRESSED
0262 0705 333 074471            JMP  CNDX NSTR      *+4       JUMP IF STORE NOT PRESSED
0263 0706 017 116417                      PASS IR   DSPI
0264 0707 327 125471            JMP  CNDX IR2  RJS  STOREX    JUMP IF M SELECTED. LEAVE SPECIAL MODE
0265 0710 320 032470            JMP  UNCD           UPDATR    M NOT SELECTED
0266 0711 320 022070            JMP  UNCD           SCANRT    JUMP TO STD ROUTINES
```

```
0267                    ********************************************************************************
0268 0712 347 172417    STOREE   IMM        LOW  IR    %375       SET UP SRC TYPE ER* SHIFT
0269 0713 017 114741             SRC2 PASS        DSPL       SET E ACCORDING TO DSPL BIT 0
0270 0714 017 136776             RTN
0271                    ********************************************************************************
0272                    ****** MEU MAP MANIPULATIONS ************************************************
0273 0715 346 000157    MEUMAPS  IMM        LOW  L     %200       S1 <= MASK OF LOW 7 BITS
0274 0716 013 143017                        SANL S1    S2
0275 0717 340 176157             IMM        HIGH L     %077       L <= 037777B
0276 0720 016 141057                        SONL S2    S1         S2 <= MASK OUT BITS 13 TO 8
0277 0721 343 076157             IMM        HIGH L     %337       OR IN BIT 13
0278 0722 016 141017                        SONL S1    S1
0279 0723 017 140617                        PASS MEU   S1         SEND MAP NO. TO MEU
0280 0724 333 075371             JMP  CNDX MSTR        READMAP    JUMP IF STORE NOT PRESSED
0281 0725 017 114620                  MESP PASS MEU     DSPL       MEU MAP <= DISPLAY
0282 0726 320 033130             JMP  UNCD             WAITR
0283 0727 017 134320    READMAP       MESP PASS DSPL    MEU        DISPLAY <= MEU MAP
0284 0730 320 033130             JMP  UNCD             WAITR
0285                    ****** SIMULATED LIA 4 I/O INSTRUCTION TO READ CIR **********************
0286 0731 344 010417    DSPLCIR  IMM        LOW  IR    %004       SET UP SEL CODE 4 IN IR
0287 0732 017 136762             IOG                               INITIATE I/O CYCLE AT TIME T2
0288 0733 017 136757             NOP                               WAIT FOR TIME T4
0289 0734 017 110336                  RTN  PASS DSPL IOI          CIR TO DISPLAY. DONT ISSUE IAK

0290                    $ORIGIN=7408
0291                    ********************************************************************************
0292                    *       SHORT SUBROUTINES TO STORE/DISPLAY SELECTED REGISTERS
0293                    ********************************************************************************
0294 0740 017 170336    DSPLREG       RTN  PASS DSPL X            PASS REG TO FRONT PANEL AND RETURN
0295 0741 017 172336                  RTN  PASS DSPL Y
0296 0742 017 112336                  RTN  PASS DSPL CNTR
0297 0743 017 144336                  RTN  PASS DSPL S3
0298 0744 017 146336                  RTN  PASS DSPL S4
0299 0745 017 150336                  RTN  PASS DSPL S5
0300 0746 017 152336                  RTN  PASS DSPL S6
0301 0747 017 154336                  RTN  PASS DSPL S7
0302 0750 017 156336                  RTN  PASS DSPL S8
0303 0751 017 160336                  RTN  PASS DSPL S9
0304 0752 017 162336                  RTN  PASS DSPL S10
0305 0753 017 164336                  RTN  PASS DSPL S11
0306 0754 017 166336                  RTN  PASS DSPL S12
0307 0755 320 035470             JMP  UNCD             DSPLCIR
0308 0756 343 176336             IMM  RTN  HIGH DSPL 377B
0309 0757 343 176336             IMM  RTN  HIGH DSPL 377B
0310                    ********************************************************************************
0311 0760 017 115636    STOREG        RTN  PASS X    DSPL       STORE INTO REG FROM FRONT PANEL
0312 0761 017 115676                  RTN  PASS Y    DSPL
0313 0762 017 114276                  RTN  PASS CNTR DSPL
0314 0763 017 115136                  RTN  PASS S3   DSPL
0315 0764 017 115176                  RTN  PASS S4   DSPL
0316 0765 017 115236                  RTN  PASS S5   DSPL
0317 0766 017 115276                  RTN  PASS S6   DSPL
0318 0767 017 115336                  RTN  PASS S7   DSPL
0319 0770 017 115376                  RTN  PASS S8   DSPL
0320 0771 017 115436                  RTN  PASS S9   DSPL
0321 0772 017 115476                  RTN  PASS S10  DSPL
0322 0773 017 115536                  RTN  PASS S11  DSPL
0323 0774 017 115576                  RTN  PASS S12  DSPL
0324 0775 017 106336                  RTN  PASS DSPL CIR          LOAD CIR FROM IHT. REQUEST LINES
0325                    *                                         AND ISSUE INTERRUPT ACKNOWLEDGE
0326 0776 320 025170             JMP  UNCD             STOR00
0327 0777 320 034530             JMP  UNCD             STOREE
0328                    ********************************************************************************
0329                    $END
** NO ERRORS**
```

```
0001                    $ORIGIN=7000B
0002                    ********************************************************************************
0003                    *
0004                    *       21MX MICRO-CODE
0005                    *       MODULE 14: FLOATING POINT INSTRUCTIONS
0006                    *
0007                    ********************************************************************************
0008                    INDIRECT EQU                    %0015
0009                    MPYX     EQU                    %0246
0010                    ********************************************************************************
0011                    *
0012 7000 017 125414    IFIX            COV   PASS S9   B          CLEAR THE OVFL AND PUT EXP IN S9
0013 7001 321 100171            JMP     CNDX ALO  RJS  *+2         TEST FOR NEG EXP
0014 7002 001 136576            RTN     ZERO A                     IF EXP<0 WE CAN'T FIX
0015 7003 017 126517                    PASS B    A                PUT HIBITS IN B-REG
0016 7004 344 000157            IMM     LOW   L     %000           PUT 'UP-B' MASK IN L
0017 7005 015 160557                    AND   A     S9             MASK LEAST SIG. B BITS INTO A
0018 7006 015 161457                    AND   S10   S9             SAVE BITS FOR ROUND-OFF
0019 7007 013 161404            R1      SANL  S9    S9             MASK EXP INTO S9 WITHOUT SIGN
0020 7010 347 140157            IMM     LOW   L     %360           PUT -20(B8) INTO L
0021 7011 004 161413                    SOV   ADD   S9    S9       CHECK TO SEE IF EXP TOO LARGE
0022 7012 320 140771            JMP     CNDX ONES         NOSHIFT  OR IF NO SHIFT REQUIRED
0023 7013 322 005231            JMP     CNDX AL15 RJS     OVER     IF SO THEN WE CAN'T FIX
0024 7014 000 061417                    INC   S9    S9             START LOOP TO SHIFT DIGITS
0025 7015 017 160255            RPT     PASS CNTR S9               PASS B OF SHIFTS INTO CNTR
0026 7016 037 124504            ARS     R1   PASS B    B          32-BIT SHIFT
0027 7017 017 126157    NOSHIFT         PASS L    A                HOLD LEFTOVER BITS IN L
0028 7020 017 124554            COV     PASS A    B                PUT INTEGER INTO A-REG
0029 7021 322 017631            JMP     CNDX AL15 RJS     RTNFP    TEST FOR NEG INTEGER
0030 7022 017 062757                    IOR           S10          IF NEG THEN CHECK FOR TRUNC. BITS
0031 7023 320 057631            JMP     CNDX TBZ          RTNFP    IF ALL ZEROS WE ARE DONE
0032 7024 000 024576            RTN     INC   A     B              OTHERWISE INC THE INTEGER & RTN
0033                    ********************************************************************************
0034                    ********************************************************************************
0035 7025 017 126517    FLOAT           PASS B    A                PUT INTEGER IN B-REG
0036 7026 001 136557            ZERO    A                          CLEAR A-REG
0037 7027 357 141417            IMM     CMLO S9   %360             STORE +15(B10) IN EXP REG
0038 7030 321 142530            JMP                       PACK
0039                    *
0040                    ********************************************************************************
0041                    *
0042 7031 017 101317    FLD             PASS S7   TAB              STORE HIBITS IN S7
0043 7032 000 023017                    INC  S1   M                INC ADDRS FOR NEXT READ
0044 7033 340 000157            IMM     HIGH L    %000             STORE 'LO-B' MASK IN L
0045 7034 220 040457            READ    INC  M    S1               READ SECOND HALF OF WRD
0046 7035 015 125417                    AND  B9   B                MEANWHILE,MASK EXP OF WRD1 INTO S9
0047 7036 017 101217                    PASS S5   TAB              STORE WRD2 LOBITS/EXP IN S5
0048 7037 013 125457                    SANL S10  B                MASK LOBITS OF WRD1 INTO S10
0049 7040 013 151257                    SANL S6   S5               MASK LOBITS OF WRD2 INTO S6
0050 7041 015 151204            R1      AND  S5   S5               MASK EXP OF WRD2 INTO S11 WITHOUT SGN
0051 7042 321 102271            JMP     CNDX ALO  RJS  *+3         IF SIGN WAS POS, JMP
0052 7043 346 000157            IMM     LOW  L    %200             OTHERWISE PUT -200(B8) INTO L
0053 7044 004 151217                    ADD  S5   S5               ADD TO EXP OF WRD2
0054 7045 017 161404            R1      PASS S9   S9               MASK EXP OF WRD1 INTO S9 WITHOUT SIGN
0055 7046 321 102471            JMP     CNDX ALO  RJS  *+3         IF SIGN WAS POS, JMP
0056 7047 346 000157            IMM     LOW  L    %200             OTHERWISE PUT -200(B8) INTO L
0057 7050 004 161417                    ADD  S9   S9               ADD TO EXP OF WRD1
0058 7051 017 127536            RTN     PASS S11  A                PUT HIBITS OF WRD1 INTO S3 & RTN
0059                    ********************************************************************************
0060                    ********************************************************************************
0061 7052 017 126154    PACK            COV  PASS L    A           CLR OVFL AND PUT WRD1 LOBITS INTO L
0062 7053 001 137457                    ZERO S10                   CLEAR COUNTER REG
0063 7054 017 024757                    IOR          B             PASS THRU ALU WITH HIBITS
0064 7055 320 057631            JMP     CNDX TBZ          RTNFP    IF A/B IS ZERO,RTN
0065 7056 346 003517            IMM     LOW  S11  %201             STORE -177(B8) IN S11
0066 7057 037 124742    NRMLZ   ARS     L1   PASS         B        TEST IF NUMBER IS NORMALIZED
0067 7060 325 043231            JMP     CNDX OVFL         RND      IF SO, JMP TO ROUNDING ROUTINE
0068 7061 077 124502            LGS     L1   PASS B       B        IF NOT, DO 32-BIT LEFT-SHIFT
0069 7062 000 063457                    INC  S10  S10              INC THE EXP CNTR
0070 7063 321 142770            JMP                       NRMLZ    GO BACK TO CHECK FOR NORMAL NUMBER
0071 7064 322 043331    RND     JMP     CNDX AL15         *+2      SINCE B WAS JUST PASSED THRU ALU
0072 7065 007 165517                    DEC  S11  S11              CHECK SGN & ADJUST ROUND OFF
0073 7066 017 164154                    COV  PASS L    S11         PUT 'ROUND' INTO L
0074 7067 003 026557                    SUB  A    A                ACTUALLY; ADD 200(B8) TO LOBITS
0075 7070 321 004171            JMP     CNDX COUT RJS     XPNT     IF NO COUT FROM LOBITS, OK, JMP
0076 7071 340 000157            IMM     HIGH L    %0               CLR L(15) FOR OVERFLOW
0077 7072 240 024517            ENV     INC  B    B                IF COUT, INC HIBITS AND CHECK FOR OVFL
0078 7073 325 003771            JMP     CNDX OVFL RJS     *+4      IF NO OVFL, OK, JMP
0079 7074 017 124504                    R1   PASS B    B          OVFL IMPLIES B/A= 1000...
0080 7075 000 061414                    COV  INC  S9   S9         SO WE SHIFT B TO FORM 0100...&
```

```
0081 7076 321 144170          JMP                    XPNT      BUMP EXP, THEN JMP
0082 7077 037 124742          ARS   L1   PASS        8         IF B NEG 100..,CHECK IF B=111..
0083 7100 325 044171          JMP   CNDX OVFL        XPNT      IF NOT, JMP
0084 7101 077 124502          LGS   L1   PASS B      B         RE-NORMALIZE
0085 7102 000 063457                     INC  S10     S10
0086 7103 017 162153 XPNT             SOV  PASS L     S10       CLR OVFL AND PUT EXP INTO L
0087 7104 003 061417                     SUB  S9      S9        SUB CALC EXP FROM ORIG EXP
0088 7105 346 000157          IMM         LOW  L     %200      PUT -200(B8) INTO L
0089 7106 003 060757                     SUB          S9        TEST FOR EXP UNDERFLO
0090 7107 322 045131          JMP   CNDX AL15        UNFLO     IF SO, JMP
0091 7110 004 160757                     ADD          S9        TEST FOR EXP OVERFLOW
0092 7111 322 017671          JMP   CNDX AL15 RJS    OVFLO     IF SO, JMP (TO 7375)
0093 7112 157 160742          LWF   L1   PASS        S9        PASS EXP SIGN INTO FLAG-REG
0094 7113 157 161402          LWF   L1   PASS S9     S9        SHIFT EXP WITH SIGN
0095 7114 340 000157          IMM         HIGH L     %000      STORE 'LO-8' MASK IN L
0096 7115 015 161457                     AND  S10     S9        MASK EXP INTO S10
0097 7116 013 127417                     SANL S9      A         MASK LOBITS INTO S9
0098 7117 017 124557                     PASS A       B         PUT HIBITS INTO A-REG
0099 7120 017 160154                SOV  PASS L      S9        PUT LOBITS INTO L
0100 7121 017 062536                RTN  IOR  B      S10       COMBINE WITH EXP AND STORE IN B-REG
0101 7122 001 136557 UNFLO                ZERO A               CLEAR A-REG; OVFL=1
0102 7123 001 136536                RTN  ZERO B               NOW CLR B-REG AND RTN
0103                          *
0104 7124 341 176576 OVER     IMM   RTN  HIGH A     %177      SET UP ERROR CONDITION IN A
0105                          ********************************************************************************
```

```
0106                          ********************************************************************************
0107 7125 017 136750 FADD           STFL
0108                          *
0109 7126 220 074457 FSUB     READ        INC  M     P         PASS P INTO M TO READ ADDR OF WRD2
0110 7127 300 000670          JSB                    INDIRECT  CHECK FOR INDIRECTS
0111 7130 301 141470          JSB                    FLD       UNPACK WRDS INTO SCRATCH REGS
0112 7131 017 154517                     PASS B       37        CHECK FOR WRD2=0
0113 7132 320 005631          JMP   CNDX TBZ  RJS     *+2       IF NOT,CONTINUE
0114 7133 346 001217          IMM         LOW  S5     %200      IF SO,MAKE EXP MOST NEG (-200,B8)
0115 7134 017 164757                     PASS         S11       CHECK FOR WRD1=0
0116 7135 320 005771          JMP   CNDX TBZ  RJS     *+2       IF NOT,CONTINUE
0117 7136 346 001417          IMM         LOW  S9     %200      IF SO,MAKE EXP MOST NEG (-200,B8)
0118 7137 324 046531          JMP   CNDX FLAG         DIFR      IF DOING ADD,SKIP AHEAD
0119 7140 010 024517                     CMPS B       B         FORM 2-COMP OF HIBITS IN B
0120 7141 010 053257                     CMPS S6      S6        FORM 2-COMP OF
0121 7142 000 053257                     INC  S6      S6        LOBITS OF WRD2
0122 7143 321 006531          JMP   CNDX COUT RJS     DIFR      IF COUT OCCURS
0123 7144 000 024517                     INC  B       B         BUMP HIBITS
0124 7145 322 006531          JMP   CNDX AL15 RJS     DIFR      CHECK SGN; IF POS,JMP
0125 7146 017 124742               L1   PASS         B         IF NEG,CHECK FOR MOST
0126 7147 320 006531          JMP   CNDX TBZ  RJS     DIFR      NEG #(100...)
0127 7150 017 124504               R1   PASS B       B         IF SO,SHIFT BACK (010...)
0128 7151 000 051217                     INC  S5      S5        &BUMP EXP
0129 7152 017 152557 DIFR                PASS A       S6
0130 7153 017 150157                     PASS L       S5        FIND DIFF IN EXPS
0131 7154 003 061351               CLFL SUB  S8       S9        &STORE IN S8; FLG=0
0132 7155 320 047731          JMP   CNDX TBZ          ADD2      IF DIFF=0,JMP TO ADD STEP
0133 7156 322 047131          JMP   CNDX AL15         RVRS      IF NEG,WRD2>WRD1
0134 7157 010 057357                     CMPS S8      S8        FORM -DIFF
0135 7160 000 057357                     INC  S8      S8        & STORE -DIFF IN S8
0136 7161 321 147430          JMP                    SWAMPCHK
0137 7162 017 124157 RVRS                PASS L       B         HOLD B IN L
0138 7163 017 164517                     PASS B       S11       WRD1<WRD2; FILL B,A
0139 7164 017 162557                     PASS A       S10       WITH S11,S10
0140 7165 015 037517                     PASL S11               ALSO FILL S11,S10,S9
0141 7166 017 153457                     PASS S10     S6        WITH B,S6,S5
0142 7167 017 151417                     PASS S9      S5
0143 7170 347 120157 SWAMPCHK IMM        LOW  L     %350      FORM -30(B8) IN L
0144 7171 003 056757                     SUB          S8        IF -DIFF>-31,RTN WITH LARGER #
0145 7172 322 050671          JMP   CNDX AL15         OUT       JMP TO RESTORE A,B
0146 7173 037 124504 SHIFT    ARS   R1   PASS B       B         NOW START SHIFT LOOP
0147 7174 000 057357                     INC  S8      S8        INC COUNTER
0148 7175 320 007571          JMP   CNDX TBZ  RJS     SHIFT     LOOP UNTIL DONE
0149                          *
0150                          ***** CONTINUED ON NEXT PAGE ************************************************
```

```
0151                          *
0152 7176 017 162154  ADD2          COV  PASS L    S10       PASS LOBITS INTO L
0153 7177 004 126557                ADD  A    A              ADD & CHECK FOR COUT
0154 7200 321 010171          JMP   CNDX COUT RJS  *+3       IF NOT JMP
0155 7201 340 000157          IMM         HIGH L   %0        CLR L(15) FOR OVFL
0156 7202 240 024517          ENV         INC  B   B         IF SO,INC HIBITS & ENABLE OVFL
0157 7203 017 164151                CLFL PASS L    S11       FLG=0
0158 7204 244 124517          ENV         ADD  B   B         ADD HIBITS AND ENABLE OVFL
0159 7205 325 010571          JMP   CNDX OVFL RJS  PKSUB     IF NO OVFL,RETURN
0160 7206 322 050431          JMP   CNDX AL15      *+2       OVFL IMPLIES SGN CHNG
0161 7207 017 136750                STFL                      SO FLG=0 IF ALU15=0
0162 7210 157 124504          LWF   R1   PASS B    B         DO FULLWRD SHIFT
0163 7211 157 126544          LWF   R1   PASS A    A         USING FLG REG TO INJECT SGN
0164 7212 000 061417                INC  S9   S9    S9        BUMP EXP
0165 7213 301 142530  PKSUB  JSB                    PACK      REPACK A,B REGS
0166 7214 000 075736                RTN  INC  P    P         INC P AND RETURN
0167 7215 017 164517  OUT                PASS B    S11       PASS MUCH LARGER WRD INTO B,A
0168 7216 017 162557                     PASS A    S10
0169 7217 301 142530          JSB                    PACK
0170 7220 000 075736                RTN  INC  P    P
0171                          ********************************************************************

0172                          ********************************************************************
0173 7221 220 074457  FMPY   READ        INC  M    P         PASS P INTO M TO READ ADDR OF WRD2
0174 7222 300 000670          JSB                    INDIRECT  CHECK FOR INDIRECTS
0175 7223 301 141470          JSB                    FLD       STORE ARGS IN SCRATCH REGS
0176 7224 000 061417                INC  S9   S9    S9
0177 7225 017 150157                PASS L    S5    S5        FORM EXP1+EXP2+1
0178 7226 004 161417                ADD  S9   S9    S9        AND SAVE IN S9
0179 7227 017 162544          R1    PASS A    S10   S10       FORM (WRD1 LOBITS)/2 IN A
0180 7230 017 155057                PASS S2   S7    S7        PASS WRD2 HIBITS INTO S2
0181 7231 300 012330          JSB                    MPYX      JMP TO MPY SUB & RTN WITH
0182 7232 017 125217                PASS S5   B     B         HIBITS IN B; SAVE IN S5
0183 7233 017 165057                PASS S2   S11   S11       PASS WRD1 HIBITS INTO S2
0184 7234 017 127517                PASS S11  A     A         LOBITS INTO A; SAVE IN S11
0185 7235 017 152544          R1    PASS A    S6    S6        FORM (WRD2 LOBITS)/2 IN A
0186 7236 300 012330          JSB                    MPYX      JMP TO MPY SUB & RTN WITH
0187 7237 017 126157                PASS L    A     A         LOBITS IN A; PASS INTO L
0188 7240 004 164557                ADD  A    S11   S11       ADD BOTH LOBITS & CHK FOR COUT
0189 7241 321 012171          JMP   CNDX COUT RJS  *+2       (ELSE TRUNCATE DIGITS)
0190 7242 000 024517                INC  B    B     B         IF COUT,BUMP HIBITS
0191 7243 017 124157                PASS L    B     B         ADD HIBITS AND SAVE IN S11
0192 7244 004 151517                ADD  S11  S5    S5
0193 7245 017 154557                PASS A    S7    S7        PASS WRD2 HIBITS INTO A
0194 7246 300 012330          JSB                    MPYX      JMP TO MPY SUB & RTN WITH
0195 7247 017 126544                R1   PASS A    A         LOBITS IN A; SAVE LOBITS/2
0196 7250 017 126154                COV  PASS L    A         ADD LOBITS/2 TO HIBITS SUM &
0197 7251 244 164542          ENV   L1   ADD  A    S11       SHFT L1 TO REORIENT
0198 7252 322 012671          JMP   CNDX AL15 RJS  *+3       CHECK FOR CARRY INTO OR
0199 7253 325 052771          JMP   CNDX OVFL      *+4       BORROW FROM HIBITS &
0200 7254 007 124517                DEC  B    B     B         ADJUST ACCORDINGLY
0201 7255 301 142530          JSB                    PACK
0202 7256 000 075736                RTN  INC  P    P
0203 7257 000 024517                INC  B    B     B         CAN'T OVFL FROM HIBITS
0204 7260 301 142530          JSB                    PACK
0205 7261 000 075736                RTN  INC  P    P
0206                          ********************************************************************
```

```
0207                          ********************************************************************
0208 7262 220 074457  FDIV   READ        INC  M    P         PASS P INTO M TO READ ADDR OF WRD2
0209 7263 300 000670          JSB                    INDIRECT  CHECK FOR INDIRECTS
0210 7264 301 141470          JSB                    FLD
0211 7265 010 054554                COV  CMPS A    S7        PASS WRD2 HIBITS & CHECK
0212 7266 320 156131          JMP   CNDX ONES      DBYZR     FOR DIV BY ZERO
0213 7267 322 053471          JMP   CNDX AL15      *+2       SINCE WE USE SAME DVSR,MAKE POS
0214 7270 000 027313                SOV  INC  S7   A         NOW & SAVE SGN IN OVFL
0215 7271 017 150157                PASS L    S5    S5        FORM EXP1-EXP2+1
0216 7272 003 061417                SUB  S9   S9    S9        & SAVE IN S9
0217 7273 000 061417                INC  S9   S9    S9
0218 7274 017 162557                PASS A    S10   S10       FILL B,A WITH WRD1 AS DVND
0219 7275 017 164517                PASS B    S11   S11       & PRESHIFT TO AVOID OVFL
0220 7276 037 124504          ARS   R1   PASS B    B
0221 7277 301 156370          JSB                    DIVX      JMP TO SPECIAL DIV SUB
0222 7300 017 127217                PASS S5   A     A         SAVE QUO1 IN S5
0223 7301 017 124757                PASS           B         PASS QUO & CHECK FOR ODD/EVEN
0224 7302 321 114231          JMP   CNDX ALO  RJS  *+2       TO SIMULATE FIRST
0225 7303 007 124517                DEC  B    B     B         LEFT SHIFT IN DIV ROUTINE
0226 7304 001 136557                ZERO A                    CLR DVND LOBITS; DVSR SAME
```

```
0227 7305 301 156370          JSB             DIVX    JMP TO SPEC DIV SUB
0228 7306 017 127517                  PASS S11  A     SAVE QUO2 IN S11
0229 7307 017 152504          R1      PASS B    S6    FORM (WRD2 LOBITS)/4 IN
0230 7310 017 124504          R1      PASS B    B     B(=DVND HIBITS)
0231 7311 001 136557                  ZERO A          CLR DVND LOBITS; DVSR SAME
0232 7312 301 156370          JSB             DIVX    JMP TO SPEC DIV SUB
0233 7313 010 026557                  CMPS A    A     FORM 2-COMP OF QUO3
0234 7314 000 026557                  INC  A    A     AS MPLR
0235 7315 017 151057                  PASS S2   S5    PASS QUO1 AS MCND
0236 7316 300 012330          JSB             MPYX    JMP TO MPY SUB
0237 7317 017 125317                  PASS S7   B     SAVE PROD HIBITS IN S5
0238 7320 001 136517                  ZERO B          PRE-CLR B
0239 7321 017 164757                  PASS      S11   CHECK SGN OF QUO2
0240 7322 322 015231          JMP  CNDX AL15 RJS  *+2   & EXTEND AS ALL 1'S(POS)
0241 7323 016 036517                  ONE  B          OR ALL 1'S(NEG)
0242 7324 017 154757                  PASS      S7    CHECK SGN OF -QUO1*QUO3
0243 7325 322 015371          JMP  CNDX AL15 RJS  *+2   IF NEG,SUB 1 FROM B
0244 7326 007 124517                  DEC  B    B
0245 7327 017 155302          L1      PASS S7   S7    REORIENT PROD (ADJUST EXP,REALLY)
0246 7330 017 154542          L1      PASS A    S7
0247 7331 017 126157                  PASS L    A
0248 7332 004 164557                  ADD  A    S11   ADD TO QUO2
0249 7333 321 015671          JMP  CNDX COUT RJS  *+2   IF COUT OCCURRED
0250 7334 000 024517                  INC  B    B     BUMP HIBITS OF RESULT
0251 7335 077 124502          LGS  L1   PASS B    B     SHIFT FULLWRD TO ORIENT RESULT
0252 7336 017 150157                  PASS L    S5    ADD QUO1 TO HIBITS
0253 7337 004 124517                  ADD  B    B
0254 7340 301 142530          JSB             PACK
0255 7341 000 075736               RTN  INC  P    P
0256 7342 001 136557                  ZERO A          CLR LOBITS
0257 7343 352 000517          DBYZR   CMHI B    %200  FORM 0111111100000000 IN HIBITS
0258 7344 017 125417          IMM     PASS S9   B     ALSO PASS INTO EXP
0259 7345 301 142530          JSB             PACK
0260 7346 000 075736               RTN  INC  P    P
0261
```

```
0262                  ********************************************************************
0263                  *
0264                  *     THIS IS A SPECIAL SUB FOR F.P. DIV
0265                  *        IT ASSUMES THAT DVSR IS ALWAYS POS
0266                  *                   THAT ORIG DVSR SGN IS IN OVFL REG
0267                  *                   THAT YOU HAVE PREVIOUSLY DONE FIRST LEFT SHIFT
0268                  *                AND THAT NO ERROR COND NEED BE CHECKED
0269                  *        (BUT IT IS FAST)
0270                  *
0271                  ********************************************************************
0272 7347 344 000257 DIVX    IMM        LOW  CNTR %0        CLR CNTR
0273 7350 157 124742         LWF  L1    PASS       B        CHECK FOR NEG DVND & SAVE SGN IN FLG
0274 7351 322 016771         JMP  CNDX  AL15 RJS   READY    IF POS, WE ARE READY
0275 7352 010 024517                    CMPS B     B        COMP HIBITS
0276 7353 010 026557                    CMPS A     A        COMP LOBITS
0277 7354 000 026557                    INC  A     A        FORM 2-COMP OF LOBITS
0278 7355 321 016771         JMP  CNDX  COUT RJS   READY    IF NO COUT,OK
0279 7356 000 024517                    INC  B     B        ELSE BUMP HIBITS
0280 7357 017 154155 READY         RPT  PASS L     S7       PASS DVSR INTO L; SET RPTFF
0281 7360 123 024502         DIV  L1    SUB  B     B        PERFORM DIV STEP(16X)
0282 7361 017 124504              R1    PASS B     B        FORM REM IN B
0283 7362 324 017271         JMP  CNDX  FLAG RJS   *+3      IF REM SGN IS TO BE NEG
0284 7363 010 024517                    CMPS B     B        (DETERMINED BY DVND),THEN
0285 7364 000 024517                    INC  B     B        FORM 2-COMP IN B
0286 7365 325 057471         JMP  CNDX  OVFL       *+4      CHECK ORIG DVSR SGN; IF POS.
0287 7366 324 017631         JMP  CNDX  FLAG RJS   RTNFP    LOOK FOR NEG DVND
0288 7367 010 026557                    CMPS A     A        WHICH MEANS FORM
0289 7370 000 026576              RTN   INC  A     A        NEG QUO IN A & RTN
0290 7371 324 057631         JMP  CNDX  FLAG       RTNFP    ELSE IF NEG,LOOK FOR POS DVND
0291 7372 010 026557                    CMPS A     A        WHICH MEANS FORM
0292 7373 000 026576              RTN   INC  A     A        NEG QUO IN A & RTN
0293                  ********************************************************************
0294                  *
0295 7374 017 136776 RTNFP         RTN
0296                  *
0297 7375 016 036544 OVFLO         R1    ONE  A             PUT MOST POS # AND MOST POS EXP
0298 7376 347 174536         IMM  RTN   LOW  B     %376     INTO A,B-REGS; OVFLO=1
0299                  ********************************************************************
0300                  $END
** NO ERRORS**
```

```
0001                    $ORIGIN=7400
0002                    ********************************************************************************
0003                    *
0004                    *       21MX MICRO-CODE
0005                    *       MODULE 15: EXTENDED INSTRUCTION GROUP
0006                    *
0007                    ********************************************************************************
0008                    FETCH     EQU                   %0000
0009                    ********************************************************************************
0010                    *   JUMP TABLES - ENTERED FROM BASE SET
0011                    ********************************************************************************
0012 7400 321 163170            JMP                     EADRX       SAX/SBX
0013 7401 017 103636                 RTN   PASS X       CAB         CAX/CBX
0014 7402 321 163170            JMP                     EADRX       LAX/LBX
0015 7403 321 163030            JMP                     EADR        STX
0016 7404 017 170076                 RTN   PASS CAB     X           CXA/CXB
0017 7405 321 163030            JMP                     EADR        LDX
0018 7406 321 163030            JMP                     EADR        ADX
0019 7407 321 164070            JMP                     XABX        XAX/XBX
0020 7410 321 163630            JMP                     EADRY       SAY/SBY
0021 7411 017 103676                 RTN   PASS Y       CAB         CAY/CBY
0022 7412 321 163630            JMP                     EADRY       LAY/LBY
0023 7413 321 163030            JMP                     EADR        STY
0024 7414 017 172076                 RTN   PASS CAB     Y           CYA/CYB
0025 7415 321 163030            JMP                     EADR        LDY
0026 7416 321 163030            JMP                     EADR        ADY
0027 7417 321 164230            JMP                     XABY        XAY/XBY
0028 7420 321 164370            JMP                     ISX
0029 7421 321 164670            JMP                     DSX
0030 7422 321 177070            JMP                     JLY
0031 7423 321 172530            JMP                     LBT
0032 7424 321 171630            JMP                     SBT
0033 7425 321 173230            JMP                     MBT
0034 7426 321 175130            JMP                     CBT
0035 7427 321 174030            JMP                     SFB
0036 7430 321 164530            JMP                     ISY
0037 7431 321 165030            JMP                     DSY
0038 7432 321 177370            JMP                     JPY
0039 7433 321 167330            JMP                     SBSCBS      SBS
0040 7434 321 167330            JMP                     SBSCBS      CBS
0041 7435 321 166630            JMP                     TBS
0042 7436 321 170230            JMP                     CMW
0043 7437 321 171030            JMP                     MVW


0044                    ********************************************************************************
0045                    *       INDEX REGISTER INSTRUCTIONS
0046                    ********************************************************************************
0047                    *                               DISPLACEMENT FROM FINISH CORREPONDS TO
0048                    *                               DISPLACEMENT FROM 7400B FOR INSTRS. LISTED
0049                    *                               IN COMMENT FIELD BELOW.
0050 7440 000 022461    FINISH          MPCK INC  M     M           SAX/SBX
0051 7441 177 102036            WRTE RTN  PASS TAB  CAB
0052 7442 017 100076                 RTN  PASS CAB  TAB         LAX/LBX
0053 7443 000 022461                 MPCK INC  M    M           STX
0054 7444 177 170036            WRTE RTN  PASS TAB  X
0055 7445 017 101636                 RTN  PASS X    TAB         LDX
0056 7446 017 100157                      PASS L    TAB         ADX
0057 7447 264 171636            ENVE RTN  ADD  X    X
0058 7450 000 022461                 MPCK INC  M    M           SAY/SBY
0059 7451 177 102036            WRTE RTN  PASS TAB  CAB
0060 7452 017 100076                 RTN  PASS CAB  TAB         LAY/LBY
0061 7453 000 022461                 MPCK INC  M    M           STY
0062 7454 177 172036            WRTE RTN  PASS TAB  Y
0063 7455 017 101676                 RTN  PASS Y    TAB         LDY
0064 7456 017 100157                      PASS L    TAB         ADY
0065 7457 264 173676            ENVE RTN  ADD  Y    Y
0066                    ********************************************************************************
0067                    *                                                           EADR IS COMMON TO LD*,ST*,AD*
0068 7460 220 074717    EADR    READ            INC  PNM  P     READ WORD 2   P<=ADDR OF NEXT INSTR.
0069 7461 301 165630            JSB                       INDBIT    CHECK FOR INDIRECT,GET OPERAND
0070 7462 321 162035            JMP  J3B                  FINISH    JUMP TO COMPLETE INSTRUCTION
0071                    ********************************************************************************
0072                    *                                                           EADRX DOES EFFECTIVE ADDR FOR
0073                    *                                                           SAX,SBX,LAX,LBX INSTRS.
0074 7463 220 074717    EADRX   READ            INC  PNM  P     READ ADDRESS OF WORD 2
0075 7464 017 170157                      PASS L    X
0076 7465 017 100457                      PASS M    TAB       M<=CONTENTS OF WORD 2.
0077 7466 322 023471            JMP  CNDX AL15 RJS  DIRECT    JUMP IF NO INDIRECT.
```

```
0078                    ********************************************************************************
0079                    *                                       INDIRECT ROUTINE FOR INDEXED INST
0080 7467 220 022457    EADRI     READ        INC   M     M       READ INDIRECT ADDRESS
0081 7470 301 165630              JSB                     INDBIT  JSB TO INDIRECT ROUTINE
0082                    ********************************************************************************
0083                    *                                       COMPUTE INDEXED ADDRESS THEN JUMP
0084 7471 004 123017    DIRECT                ADD   S1    M       S1<=TARGET ADDR. + X OR Y.
0085 7472 220 040457              READ        INC   M     S1      READ INDEXED ADDRESS.
0086 7473 321 162035              JMP   J3D                 FINISH  JUMP TO COMPLETE THE INSTRUCTION
0087                    ********************************************************************************
0088                    *                                       EADRY COMPUTES EFFECTIVE ADDRESS
0089                    *                                       FOR SAY,SBY,LAY,LBY INSTRS.
0090 7474 220 074717    EADRY     READ        INC   PNM   P
0091 7475 017 172157                          PASS  L     Y
0092 7476 017 100457                          PASS  M     TAB     M<= CONTENTS OF WORD 2.
0093 7477 322 023471              JMP   CNDX  AL15  RJS   DIRECT  JUMP IF NO INDIRECTS.
0094 7500 321 163370              JMP                     EADRI   JUMP TO DO INDIRECT ROUTINE
0095                    ********************************************************************************
0096 7501 017 103017    XABX                  PASS  S1    CAB     EXCHANGE A/B WITH X
0097 7502 017 170057                          PASS  CAB   X
0098 7503 017 141636                RTN       PASS  X     S1
0099                    ********************************************************************************
0100 7504 017 103017    XABY                  PASS  S1    CAB     EXCHANGE A/B WITH Y
0101 7505 017 172057                          PASS  CAB   Y
0102 7506 017 141676                RTN       PASS  Y     S1
0103                    ********************************************************************************
0104 7507 000 071617    ISX                   INC   X     X       INCREMENT X, SKIP IF ZERO
0105 7510 320 067271              JMP   CNDX  TBZ               SKIP
0106 7511 017 136776    RETURN    RTN
0107                    ********************************************************************************
0108 7512 000 073657    ISY                   INC   Y     Y       INCREMENT Y, SKIP IF ZERO.
0109 7513 320 067271              JMP   CNDX  TBZ               SKIP
0110 7514 017 136776              RTN
0111                    ********************************************************************************
0112 7515 007 171617    DSX                   DEC   X     X       DECREMENT X, SKIP IF ZERO.
0113 7516 320 067271              JMP   CNDX  TBZ               SKIP
0114 7517 017 136776              RTN
0115                    ********************************************************************************
0116 7520 007 173657    DSY                   DEC   Y     Y       DECREMENT Y, SKIP IF ZERO.
0117 7521 320 067271              JMP   CNDX  TBZ               SKIP
0118 7522 017 136776              RTN
```

```
0119                    ********************************************************************************
0120                    *   GENERAL INDIRECT ROUTINE FOR INDEX BIT INSTR
0121                    *       COMMON ROUTINES FOR WORD/BYTE INSTRUCTIONS
0122                    ********************************************************************************
0123                    *                                       INITIALIZATION FOR WORD,BYTE
0124 7523 000 075217    INITCM                INC   S5    P       S5<= ADDRESS OF WORD 3.
0125 7524 220 050457              READ        INC   M     S5      READ ADDRESS OF WORD 3.
0126 7525 000 051157                          INC   S4    S5      S4<= ADDRESS OF NEXT INSTRUCTION.
0127 7526 017 101117                          PASS  S3    TAB     S3<= CONTENTS OF WORD 3.
0128 7527 320 065531              JMP   CNDX  TBZ               JUMP IF WORD 3 = 0 (NO INTERRUPT)
0129 7530 000 075717                          INC   P     P       P<=ADDRESS OF WORD 3 (FOR EXIT)
0130 7531 001 155336                RTN       ZERO  S7    S7      S7<=0 AND RETURN TO CALLER.
0131 7532 220 074710              READ  STFL  INC   PNM   P       READ ADDRESS OF WORD 2. P<=P+1.
0132 7533 001 155317                          ZERO  S7    S7      S7 <= 0.
0133                    ********************************************************************************
0134                    *                                       COMMON INDIRECT IMBEDDED IN INITCM
0135 7534 017 100457    INDBIT                PASS  M     TAB     M <= CONTENTS OF LAST READ ADDRESS.
0136 7535 322 026271              JMP   CNDX  AL15  RJS   CONTBIT  JUMP IF NO INDIRECT.
0137 7536 220 022465    INDLBIT   READ  INCI  INC   M     M       READ ADDRESS IN M
0138 7537 326 065631              JMP   CNDX  NHOI              INDBIT  JUMP IF NO HALT OR INTERRUPT PENDING
0139 7540 017 100457    IND2BIT               PASS  M     TAB     M<= CONTENTS OF LAST READ ADDRESS
0140 7541 330 125671              JMP   CNDX  NSNG  RJS   INDBIT+1  JUMP IF SINGLE-INSTRUCT. MODE
0141 7542 007 175717    DEC2                  DEC   P     P
0142 7543 007 175717                          DEC   P     P       P <= ADDRESS OF WORD 1.
0143 7544 320 000030              JMP                     FETCH   ATTEMPT JUMP TO FETCH ROUTINE.
0144 7545 324 066371    CONTBIT   JMP   CNDX  FLAG            *+2     FLAG IDENTIFIES CALLER TO INDBIT
0145 7546 220 022476              READ  RTN   INC   M     M       READ ADDRESS AND RETURN.
0146                    ********************************************************************************
0147 7547 220 022451              READ  CLFL  INC   M     M       CALLER=INITCM--RESET FLAG,READ COUNT
0148 7550 017 101257                          PASS  S6    TAB     S6 <= COUNT FOR THIS INSTRUCTION
0149 7551 320 026571              JMP   CNDX  TBZ   RJS   RTNCNT  JUMP IF COUNT NOT ZERO.
0150 7552 301 176730              JSB                     EXIT    END THE INSTRUCTION.
0151 7553 017 153136    RTNCNT          RTN   PASS  S3    S6      S3 <= COUNT, RETURN TO CALLER.
```

```
0152                    ********************************************************************************
0153                    *       BIT INSTRUCTIONS
0154                    ********************************************************************************
0155 7554 220 074717 TBS     READ      INC  PNM  P
0156 7555 301 165630         JSB                  INDBIT    GET MASK
0157 7556 017 100157                   PASS L     TAB       L <= MASK.
0158 7557 220 074457         READ      INC  M    P
0159 7560 301 165630         JSB                  INDBIT    GET WORD TO BE TESTED
0160 7561 015 101017                   AND  S1    TAB       LOGICAL AND OF MASK, WORD UNDER TEST.
0161 7562 013 041017                   XOR  S1    S1        S1 <= 0 IF ALL MASK BITS SET IN WORD
0162 7563 320 067271         JMP  CNDX TBZ        SKIP      SKIP IF ALL MASK BITS SET IN WORD.
0163 7564 000 075717                   INC  P     P         SKIP NEXT MACHINE INSTRUCTION.
0164 7565 000 075736 SKIP         RTN  INC  P     P         ADJUST P, JUMP TO FETCH ROUTINE.
0165                    ********************************************************************************
0166 7566 220 074717 SBSCBS  READ      INC  PNM  P
0167 7567 301 165630         JSB                  INDBIT    OBTAIN BIT MASK
0168 7570 017 100157                   PASS L     TAB       L <= BIT MASK
0169 7571 220 074457         READ      INC  M    P
0170 7572 301 165630         JSB                  INDBIT    OBTAIN WORD TO BE OPERATED ON.
0171 7573 327 170031         JMP  CNDX IR2        CBS       JUMP IF INSTRUCTION IS CBS.
0172 7574 017 001017                   IOR  S1    TAB       SET BITS IN WORD,PUT IN S1
0173 7575 000 022461              MPCK INC  M     M         MEMORY PROTECT CHECK.
0174 7576 177 140017         WRTE      PASS TAB   S1        REWRITE WORD TO MEMORY. RETURN TO FETCH.
0175 7577 000 075736              RTN  INC  P     P
0176 7600 013 101017 CBS              SANL S1    TAB       S1 <= MEMORY WORD WITH BITS CLEARED
0177 7601 000 022461              MPCK INC  M     M
0178 7602 177 140017         WRTE      PASS TAB   S1        REWRITE WORD TO MEMORY.
0179 7603 000 075736              RTN  INC  P     P         RETURN TO FETCH ROUTINE.
0180                    ********************************************************************************
0181                    *       WORD INSTRUCTIONS
0182                    ********************************************************************************
0183 7604 301 165170 CMW      JSB                  INITCM    INITIALIZE
0184 7605 220 026457         READ      INC  M     A         READ FROM ARRAY A.
0185 7606 017 100157                   PASS L     TAB       L <= WORD FROM ARRAY A
0186 7607 220 024457         READ      INC  M     B         READ ADDRESS IN ARRAY B.
0187 7610 000 024517                   INC  B     B         INCREMENT ARRAY B POINTER.
0188 7611 003 001017                   SUB  S1    TAB       SUBTRACT ARRAY WORDS B - A.
0189 7612 320 036431         JMP  CNDX TBZ  RJS   CHAL15    JUMP IF UNEQUAL.
0190 7613 000 026557                   INC  A     A         INCREMENT ARRAY A POINTER.
0191 7614 007 145117                   DEC  S3    S3        DECREMENT COUNT.
0192 7615 320 076731         JMP  CNDX TBZ        EXIT      JUMP TO EXIT IF COUNT IS ZERO.
0193 7616 335 030271         JMP  CNDX INT  RJS   CMW+1     JUMP IF NOT INTERRUPTED.
0194 7617 321 176130         JMP                  INTPEND
0195                    ********************************************************************************
0196 7620 301 165170 MVW      JSB                  INITCM    INITIALIZE.
0197 7621 220 026457         READ      INC  M     A         READ FROM ARRAY A
0198 7622 000 026557                   INC  A     A         INCREMENT ARRAY A POINTER.
0199 7623 017 101017                   PASS S1    TAB       S1 <= CONTENTS OF WORD OF ARRAY A
0200 7624 000 024457                   INC  M     B         M <= ADDRESS FROM ARRAY B
0201 7625 017 122761              MPCK PASS        M         MEMORY PROTECT CHECK-- BIT 15 LOW.
0202 7626 177 140017         WRTE      PASS TAB   S1        WRITE WORD INTO ARRAY B.
0203 7627 000 024517                   INC  B     B         ADVANCE ARRAY B POINTER.
0204 7630 007 145117                   DEC  S3    S3        DECREMENT COUNT.
0205 7631 320 076731         JMP  CNDX TBZ        EXIT      EXIT IF COUNT IS ZERO.
0206 7632 335 031071         JMP  CNDX INT  RJS   MVW+1     JUMP IF NOT INTERRUPTED.
0207 7633 321 176130         JMP                  INTPEND
```

```
0208                    ********************************************************************************
0209                    *       BYTE INSTRUCTIONS
0210                    ********************************************************************************
0211 7634 340 000157 SBT      IMM       HIGH L     %000      L <= 000377B.
0212 7635 015 127017 STBYTE               AND  S1    A         S1 <= RIGHT BYTE OF A REG.
0213 7636 157 125244              LWF  R1  PASS S6    B         S6 <= WORD ADDRESS. FLAG SET IF BYTE ODD
0214 7637 220 052461         READ      MPCK INC  M     S6        READ WORD ADDRESS,CHECK FOR MP VIOLATION
0215 7640 324 032171         JMP  CNDX FLAG RJS   STEVEN    JUMP IF STORE TO EVEN BYTE.
0216 7641 013 101417                   SANL S9    TAB       MASK OUT EVEN BYTE OF MEMORY WORD.
0217 7642 321 172330         JMP                  MERGE
0218 7643 015 101417 STEVEN            AND  S9    TAB       MASK OUT ODD BYTE OF MEMORY WORD.
0219 7644 017 141093                L4 PASS S1    S1        EXCHANGE BYTES IN REGISTER CONTAINING
0220 7645 017 141093                L4 PASS S1    S1        BYTE TO BE STORED.
0221 7646 017 160157 MERGE             PASS L     S9        L <= MEMORY WORD WITH TARGET BYTE CLEARED
0222 7647 017 041017                   IOR  S1    S1        S1 <= WORD WITH BYTES MERGED.
0223 7650 000 024517                   INC  B     B         INCREMENT BYTE ADDRESS.
0224 7651 177 140036         WRTE RTN  PASS TAB   S1        WRITE NEW WORD BACK INTO WORD ADDRESS.
```

Appendix E

```
0225                    ********************************************************************************
0226 7652 017 125057 LBT                PASS S2   B       S2 <= BYTE ADDRESS.
0227 7653 000 024517                    INC  B    B       INCREMENT BYTE ADDRESS FOR NEXT INSTRUCT.
0228 7654 340 000151 LDBYTE   IMM  CLFL HIGH L   %000    L <= 000377B.  CLEAR CPU FLAG.
0229 7655 157 143244          LWF  R1   PASS S6   S2      S6 <= WORD ADDRESS OF BYTE. SET FLAG IF O
0230 7656 220 052457          READ      INC  M    S6      DDD BYTE.  READ WORD ADDRESS.
0231 7657 324 073171          JMP  CNDX FLAG      LODD    JUMP IF BYTE IS ODD.
0232 7660 013 100543               L4   SANL A    TAB     MASK OUT EVEN BYTE AND MOVE ODD BYTE
0233 7661 017 126543               L4   PASS A    A       TO EVEN BYTE OF A REG.
0234 7662 017 136776               RTN                     RETURN TO CALLER OR FETCH.
0235 7663 015 100776 LODD          RTN  AND A    TAB     MASK OUT EVEN BYTE,LOAD INTO A. RETURN.
0236                    ********************************************************************************
0237 7664 301 165170 MBT       JSB                INITCM  INITIALIZE.
0238 7665 017 127057                    PASS S2   A       S2 <= ADDRESS START OF ARRAY A.
0239 7666 301 172630           JSB                LDBYTE  LOAD BYTE FROM ARRAY A.
0240 7667 000 043051                CLFL INC  S2   S2      RESET FLAG, S2 <= NEXT BYTE ADDR IN ARRAY
0241 7670 301 171670           JSB                STBYTE  STORE BYTE INTO BYTE ADDRESS IN B REG.
0242 7671 007 145117                    DEC  S3   S3      DECREMENT COUNT
0243 7672 320 033671           JMP  CNDX TBZ  RJS  NOTDAN  JUMP IF COUNT NOT ZERO.
0244 7673 017 142557                    PASS A    S2      A <= 1 + LAST ARRAY A BYTE ADDRESS MOVED
0245 7674 321 176730           JMP                EXIT
0246 7675 335 033331 NOTDAN    JMP  CNDX INT  RJS  MBT+2   JUMP IF NOT INTERRUPTED.
0247 7676 017 142557                    PASS A    S2      A <= 1 + LAST ARRAY A ADDRESS MOVED
0248 7677 321 176130           JMP                INTPEND
```

```
0249                    ********************************************************************************
0250 7700 340 000157 SFB       IMM        HIGH L   %000    L <= 000377B.
0251 7701 015 127117                      AND  S3   A       S3 <= TEST BYTE IN LOW-ORDER BYTE
0252 7702 013 127143                 L4   SANL S4   A       S4 <= TERMINATION BYTE IN
0253 7703 017 147143                 L4   PASS S4   S4      LOW-ORDER BYTE
0254 7704 017 127357                      PASS S8   A       S8 <= SAVE ORIGINAL CONTENTS OF A REG.
0255 7705 301 172530 CONTSFB   JSB                LBT     LOAD BYTE INTO A REG FROM ADDRESS IN B.
0256 7706 017 126157                      PASS L    A       L <= BYTE TO TESTED IN LOW BYTE.
0257 7707 013 045257                      XOR  S6   S3      COMPARE BYTE TO TEST BYTE.
0258 7710 320 034571           JMP  CNDX TBZ  RJS  NOMATCH  JUMP IF UNEQUAL.
0259 7711 017 156557                      PASS A    S8      RESTORE ORIGINAL CONTENTS OF A REG
0260 7712 007 124536                RTN   DEC  B    B       B <= BYTE ADDRESS OF MATCH. GO TO FETCH
0261 7713 013 047017 NOMATCH              XOR  S1   S4      COMPARE BYTE TO TERMINATION BYTE.
0262 7714 320 034771           JMP  CNDX TBZ  RJS  INTTST   JUMP IF UNEQUAL.
0263 7715 017 156557                      PASS A    S8      RESTORE ORIGINAL CONTENTS OF A REG.
0264 7716 000 075736                RTN   INC  P    P       SKIP NEXT MACHINE INSTRUCTION AND FETCH.
0265 7717 335 034271 INTTST    JMP  CNDX INT  RJS  CONTSFB  JUMP IF NOT INTERRUPTED.
0266 7720 017 156557                      PASS A    S8      RESTORE ORIGINAL CONTENTS OF A REG.
0267 7721 007 175736                RTN   DEC  P    P       P <= INSTRUCTION ADDRESS, GO TO FETCH.
0268                    ********************************************************************************
0269 7722 301 165170 CBT       JSB                INITCM  INITIALIZE.
0270 7723 017 127357                      PASS S8   A       S8 <= POINTER FOR ARRAY A.
0271 7724 017 157057                      PASS S2   S8      S2 <= NEXT BYTE ADDRESS IN ARRAY A.
0272 7725 301 172630           JSB                LDBYTE  LOAD ARRAY A BYTE INTO A REG.
0273 7726 017 127417                      PASS S9   A       S9 <= ARRAY A BYTE.
0274 7727 000 043357                      INC  S8   S2      INCREMENT ARRAY A POINTER.
0275 7730 301 172530           JSB                LBT     A <= BYTE FROM ARRAY B (ADDRESS IN B REG)
0276 7731 017 160157                      PASS L    S9      L <= BYTE FROM ARRAY A
0277 7732 003 027017                      SUB  S1   A       SUBTRACT BYTE FROM ARRAY B - A.
0278 7733 320 036331           JMP  CNDX TBZ  RJS  CHAL15B  JUMP IF BYTES NOT EQUAL.
0279 7734 007 145117                      DEC  S3   S3      DECREMENT THE COUNT.
0280 7735 320 036031           JMP  CNDX TBZ  RJS  CONTCBT  JUMP IF COUNT IS NOT ZERO.
0281 7736 017 156557                      PASS A    S8      EQUAL EXIT... A <= 1+LAST MOVED BYTE ADDR
0282 7737 321 176730           JMP                EXIT
0283 7740 335 035231 CONTCBT   JMP  CNDX INT  RJS  CBT+2   JUMP IF NOT INTERRUPTED.
0284 7741 017 156557                      PASS A    S8      A <= NEXT BYTE ADDRESS OF ARRAY A TO TEST
```

```
0285                    ********************************************************************************
0286                    *         COMMON ROUTINES TO MOVE, COMPARE INSTRUCTIONS
0287                    ********************************************************************************
0288                    *                                           INTERRUPT EXIT
0289 7742 000 050457    INTPEND             INC  M     S5         M <= ADDRESS OF WORD 3
0290 7743 177 144017            WRTE        PASS TAB   S3         WRITE REMAINING COUNT INTO WORD 3.
0291 7744 007 175717                        DEC  P     P
0292 7745 007 175736                 RTN    DEC  P     P          P <= ADDRESS OF WORD 1, GO TO FETCH.
0293                    ********************************************************************
0294                    *                                           EXIT TESTS FOR CBT,CMW
0295 7746 007 156557    CHAL158             DEC  A     S8         A <= BYTE ADDRESS OF MISMATCH.
0296 7747 017 141017                        PASS S1    S1         CHECK RESULT OF COMPARE
0297 7750 322 036531    CHAL15   JMP  CNDX  AL15 RJS   SKIP1      JUMP IF SIGN BIT IS ZERO.
0298 7751 000 047157                        INC  S4    S4         SKIP ONE MACHINE INSTRUCTION.
0299 7752 000 047157    SKIP1               INC  S4    S4         SKIP ONE MACHINE INSTRUCTION.
0300 7753 007 145117                        DEC  S3    S3         DECREMENT THE COUNT.
0301 7754 017 144157                        PASS L     S3         L <= COUNT REMAINING
0302 7755 004 124517                        ADD  B     B          B <= FIRST ADDR. IN ARRAY B + COUNT.
0303                    ********************************************************************
0304                    *                                           COMPLETION EXIT
0305 7756 000 050457    EXIT                INC  M     S5         M <= ADDRESS OF WORD 3
0306 7757 177 154017            WRTE        PASS TAB   S7         WORD 3 <= ZERO.
0307 7760 017 147736                 RTN    PASS P     S4         P <= NEXT MACHINE INSTR. TO EXECUTE.FETCH
0308                    ********************************************************************************
0309                    *     JUMP INSTRUCTIONS
0310                    ********************************************************************************
0311 7761 220 074717    JLY         READ    INC  PNM   P          READ ADDRESS OF WORD 2.
0312 7762 301 165630            JS8         JS8        INDBIT     CHECK FOR INDIRECT,M<= DESTINATION ADDR.
0313 7763 340 120417            IMM         HIGH IR    %050       MACHINE JMP INTO IR TO SET LOW MP BOUNDS
0314 7764 017 122461                 MPCK   PASS M     M          DO MP CHECK ON JUMP TARGET ADDRESS.
0315 7765 017 175657                        PASS Y     P          Y <= ADDRESS OF FOLLOWING MACHINE INSTR.
0316 7766 017 123736                 RTN    PASS P     M          P <= DESTINATION ADDRESS, JUMP TO FETCH.
0317                    ********************************************************************************
0318 7767 220 074717    JPY         READ    INC  PNM   P          READ ADDRESS OF WORD 2.
0319 7770 017 172157                        PASS L     Y          L <= INDEX REG. Y.
0320 7771 004 101017                        ADD  S1    TAB        S1 <= INDEXED JUMP ADDRESS.
0321 7772 340 120417            IMM         HIGH IR    %050       MACHINE JMP INTO IR TO SET LOW MP BOUNDS
0322 7773 017 140457                        PASS M     S1         M<= INDEXED ADDRESS, WITH BIT 15 LOW
0323 7774 017 122761                 MPCK   PASS       M          MP CHECK ON 15-BIT DESTINATION ADDRESS.
0324 7775 017 123736                 RTN    PASS P     M          P <= DESTINATION ADDRESS. GO TO FETCH.
0325                    ********************************************************************************
0326 7776 377 177777            ONES
0327 7777 377 177777            ONES
0328                    $END
**  NO  ERRORS**
```

INDEX

HEWLETT hp PACKARD

MANUAL PART NO. 02108-90008
MICROFICHE PART NO. 02108-90009

Printed: AUG 1974
Printed in U.S.A.