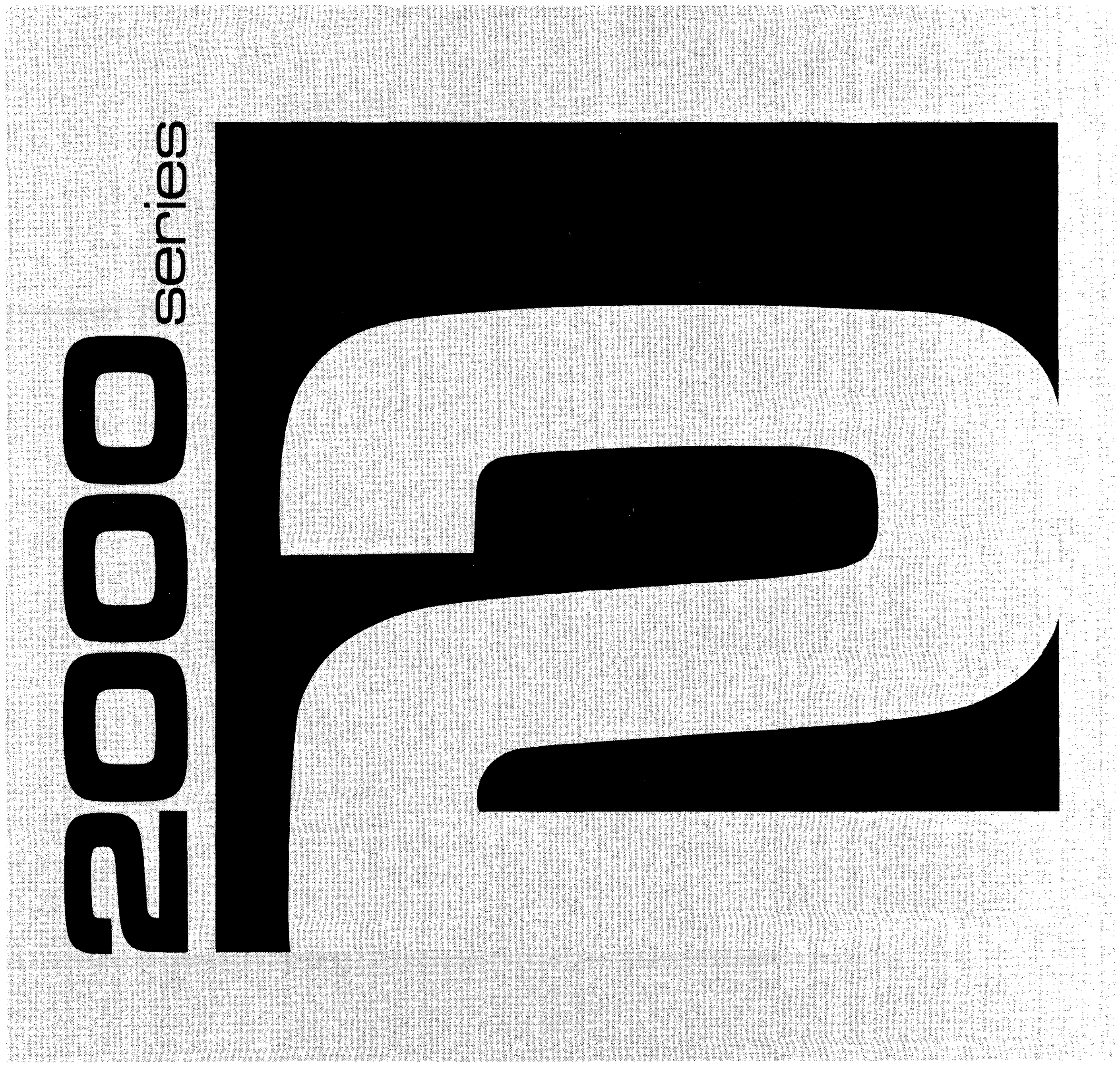
HEWLETT PACKARD

21MX M-Series Computers BCS and DOS Microprogramming

Reference Manual



21MX M-Series Computers BCS and DOS Microprogramming

Reference Manual

(This manual reflects information that is compatible with software revision code 1437.)

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

Library Index Number
21M.340.02108-90008

PART NO. 02108-90008

Printed in U.S.A. 10/77

PUBLICATION NOTICE

Information in this manual describes the use of 21MX M-Series BCS and DOS Microprogramming software. Changes in text to document software updates subsequent to the initial release are supplied in manual update notices and/or complete revisions to the manual. The history of any changes to this edition of the manual is given below under "Publication History." The last change itemized reflects the software currently documented in the manual.

Any changed pages supplied in an update package are identified by a change number adjacent to the page number. Changed information is specifically identified by a vertical line (revision bar) on the outer margin of the page.

PUBLICATION HISTORY

Second Edition .... Oct 77 (Software Rev. Code 1437)

PREFACE

This manual is a complete reference source for microprogramming the Hewlett-Packard 21MX M-Series Computers (HP 2105/2108/2112). With the facilities of the HP 12978A Writable Control Store, the user can expand the already powerful capability of M-Series Computer by adding custom-tailored instructions to the existing set of microprogrammed basic instructions.

The HP 12978A Writable Control Store is provided with options to adapt to the applicable operating system. 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 the *HP 12978A Writable Control Store Reference Manual,* part no. 12978-90007, for a complete description of the options.

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

RELATED DOCUMENTATION

It is assumed that the microprogrammerr has read the ***HP 21MX Computer Series Reference Manual,* part no. 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,* part no. 24307-90006.
- *Basic Control System,* part no. 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,* part no. 24307-90014.

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

- *HP Assembler,* part no. 02116-2014.

The 21MX M-Series Computer is described in:

- *HP 21MX Computer Series Operator's Manual,* part no. 02108-90004.
- *HP 21MX Computer Series Installation and Service Manual,* part no. 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 pROM's is described in:

- *HP 12909B pROM Writer Operating and Reference Manual,* part no. 12909-90009.
- *HP 12909B Programmable ROM Writer Interface Kit Installation and Service Manual,* part no. 12909-90005.

The operation of the HP 12978A (1/4K) Writable Control Interface Kit is described in:

- *HP 12978A Writable Control Store Reference Manual,* part no. 12978-90007.

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 and V so that when the time comes to use the material, it may be found easily. These sections describe the microprogramming language, the Micro-assembler, and the Micro Debug Editor.

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, the microprogram must test for pending interrupts or data may 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

2-1. RELATIONSHIP BETWEEN SECTIONS

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

2-2. 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

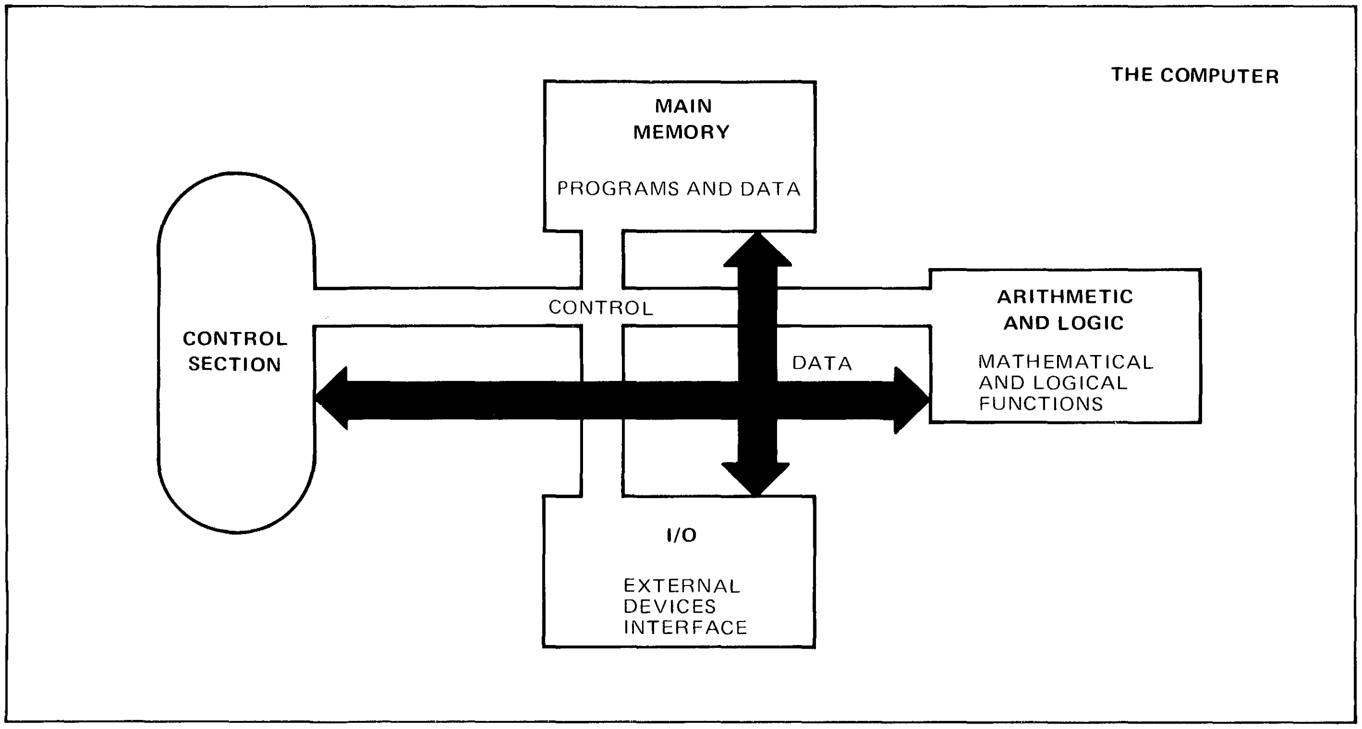


Figure 2-1. Four Major Computer Sections

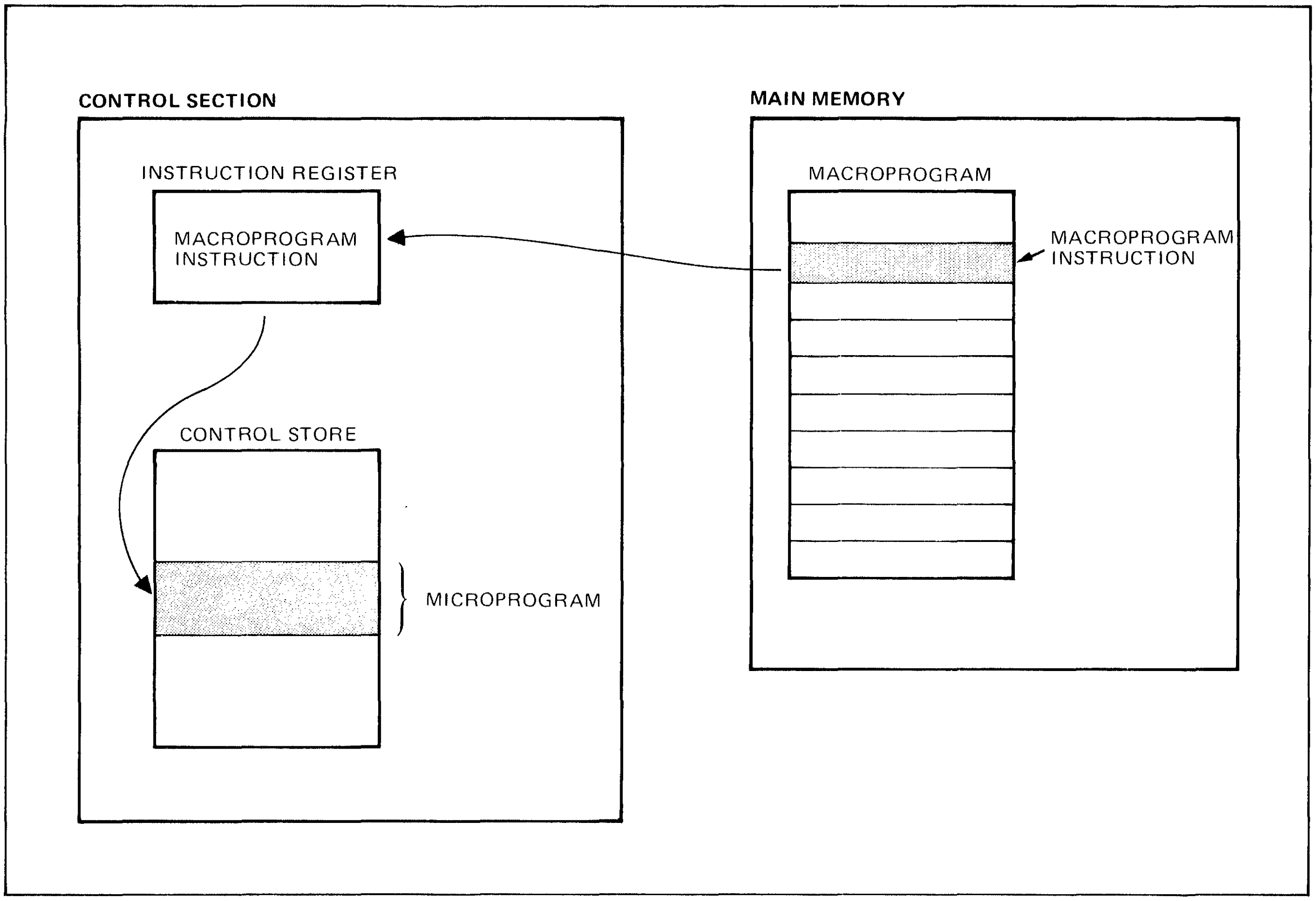


Figure 2-2. A Microprogram Implements **One** Macroprogram Instruction

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

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, Power Fail and Dynamic Mapping System 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.

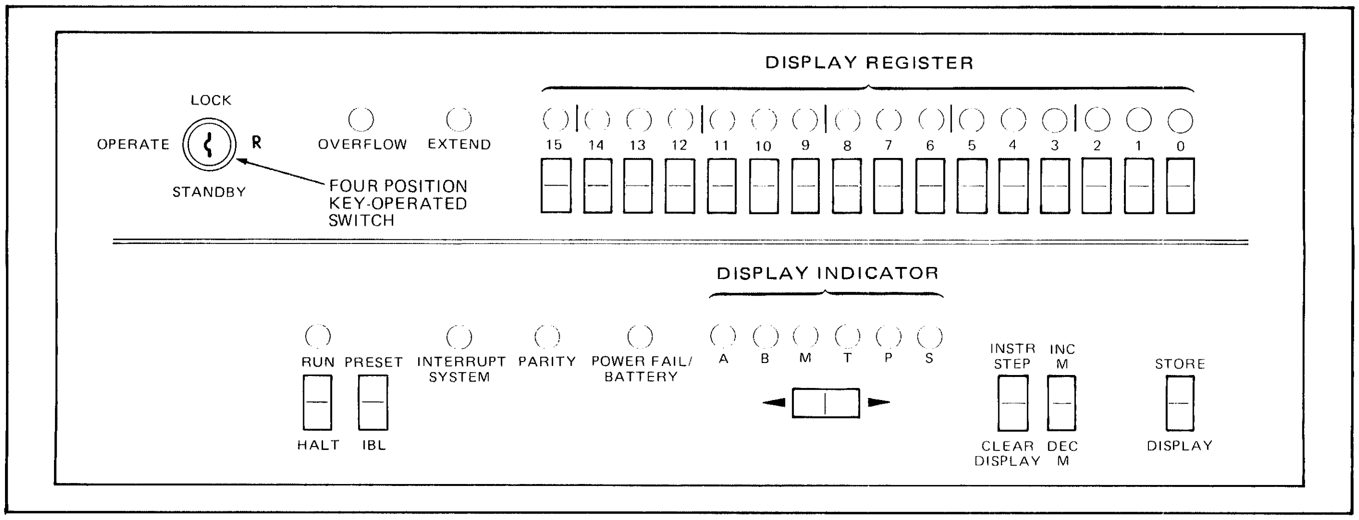


Figure 2-3. Front Panel Displays and Switches

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.

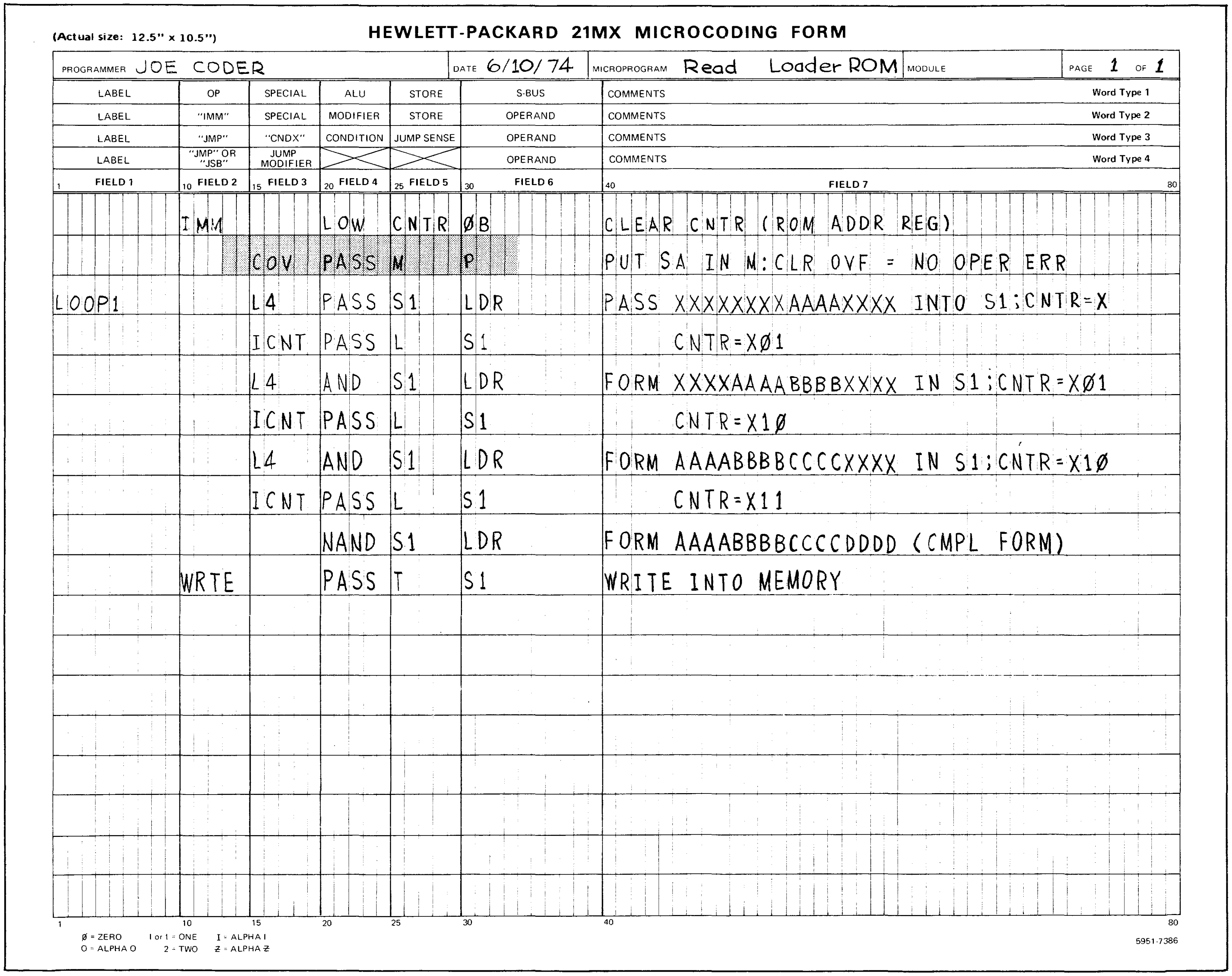
There are four micro-instruction formats. Each format defines a micro-instruction Word Type (Word Type 1, Word Type 2, etc.) and determines a set of micro-orders which may be coded into the format. Micro-instruction Word Types and micro-orders are described in Section IV.

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.



(Actual size: 12.5" x 10.5")

HEWLETT-PACKARD 21MX MICROCODING FORM

PROGRAMMER JOE CODER | DATE 6/10/74 | MICROPROGRAM Read Loader ROM | MODULE | PAGE 1 OF 1

FIELD 1	FIELD 2	FIELD 3	FIELD 4	FIELD 5	FIELD 6	FIELD 7	
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	10	15	20	25	30	40 … 80	

	IMM		LOW	CNTR	ØB	CLEAR CNTR (ROM ADDR REG)
		COV	PASS	M	P	PUT SA IN M:CLR OVF = NO OPER ERR
LOOP1		L4	PASS	S1	LDR	PASS XXXXXXXXAAAAXXXX INTO S1;CNTR=X
		ICNT	PASS	L	S1	CNTR=XØ1
		L4	AND	S1	LDR	FORM XXXXAAAABBBBXXXX IN S1;CNTR=XØ1
		ICNT	PASS	L	S1	CNTR=X1Ø
		L4	AND	S1	LDR	FORM AAAABBBBCCCCXXXX IN S1;CNTR=X1Ø
		ICNT	PASS	L	S1	CNTR=X11
			NAND	S1	LDR	FORM AAAABBBBCCCCDDDD (CMPL FORM)
	WRTE		PASS	T	S1	WRITE INTO MEMORY

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 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 (4.6μs): 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 two micro-instructions (.650μs).

For example:

```
          PASS   A     branch out of code se-
JMP CNDX ALO     OUT   quence if bit 0 of A = 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 form 16 bit constants in 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 at the termination of field 6 (one space must follow the field 6 mnemonic) and contains comments only. Field 7 ends in column 80.

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
		INC		A

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

OP	SPEC	ALU	STORE	S-BUS
	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
	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; an octal -12 is specified in the example (366B):

"IMM"	SPEC	MODIF	STORE	OPERAND
IMM				366B

This is necessary because use of the minus sign (–) is not allowed.

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		366B

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	366B

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	366B

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.

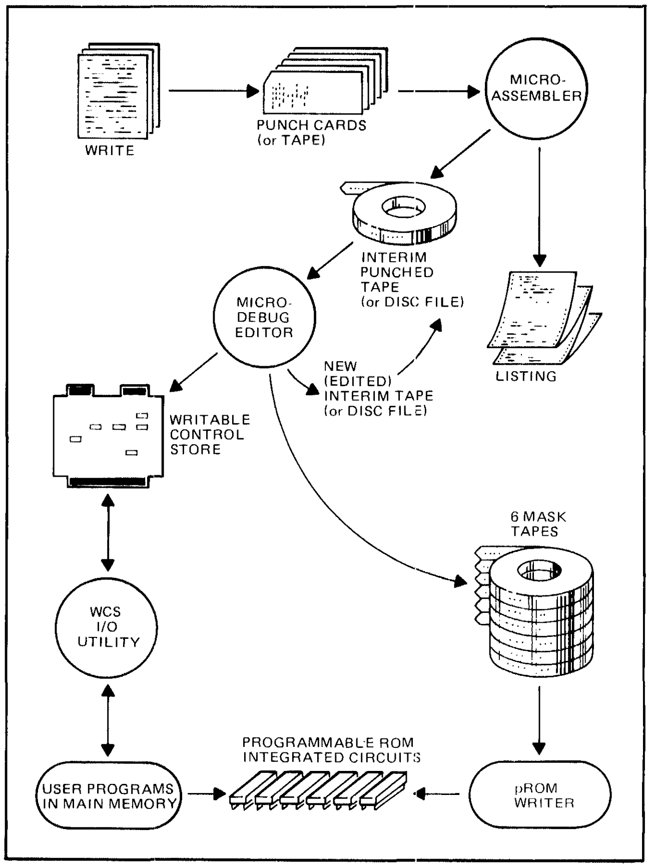

WRITE
PUNCH CARDS (or TAPE)
MICRO-ASSEMBLER
INTERIM PUNCHED TAPE (or DISC FILE)
LISTING
MICRO-DEBUG EDITOR
NEW (EDITED) INTERIM TAPE (or DISC FILE)
WRITABLE CONTROL STORE
6 MASK TAPES
WCS I/O UTILITY
PROGRAMMABLE ROM INTEGRATED CIRCUITS
USER PROGRAMS IN MAIN MEMORY
PROM WRITER

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 microprograms 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.

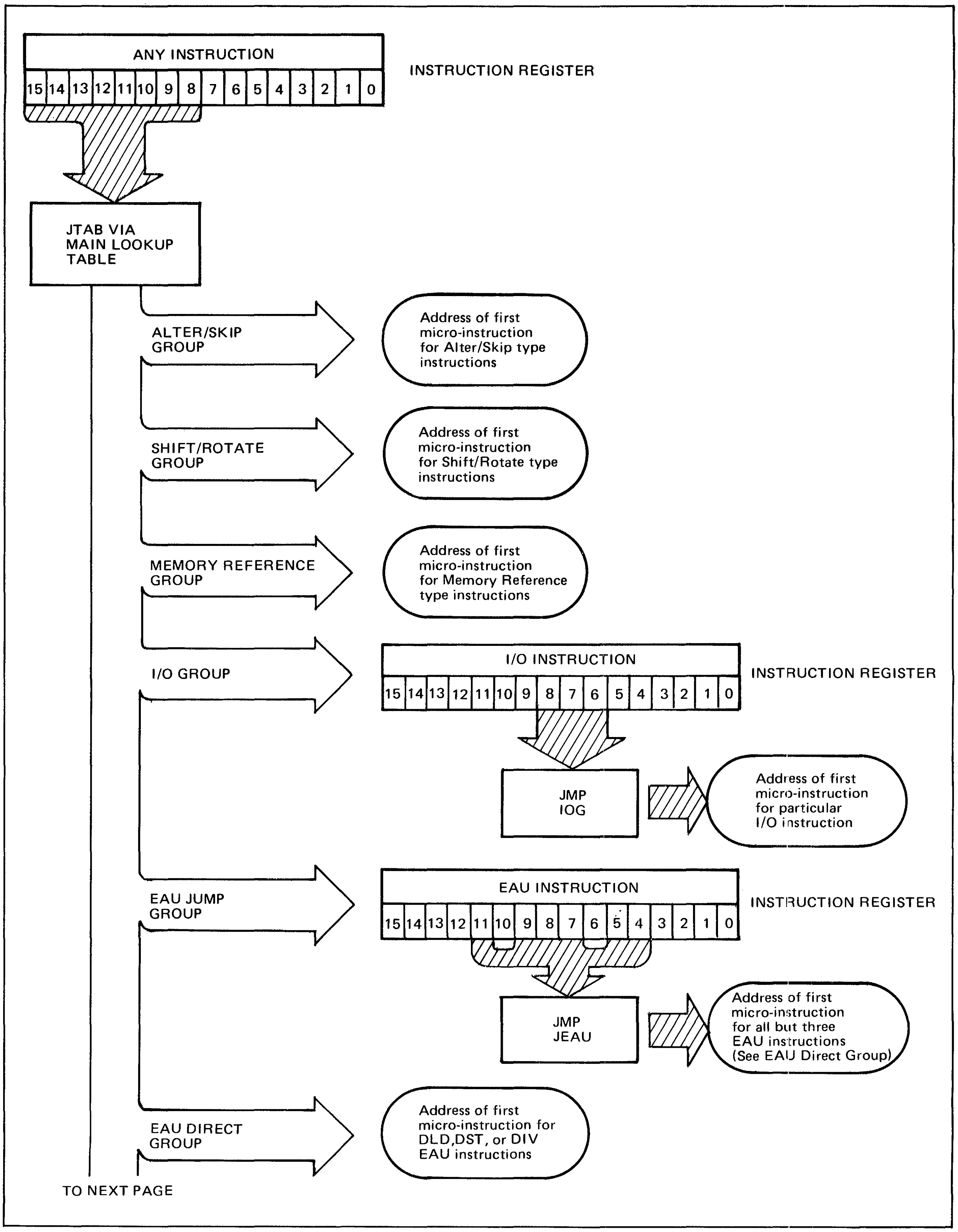

ANY INSTRUCTION
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
INSTRUCTION REGISTER
JTAB VIA MAIN LOOKUP TABLE
ALTER/SKIP GROUP
Address of first micro-instruction for Alter/Skip type instructions
SHIFT/ROTATE GROUP
Address of first micro-instruction for Shift/Rotate type instructions
MEMORY REFERENCE GROUP
Address of first micro-instruction for Memory Reference type instructions
I/O GROUP
I/O INSTRUCTION
INSTRUCTION REGISTER
JMP IOG
Address of first micro-instruction for particular I/O instruction
EAU JUMP GROUP
EAU INSTRUCTION
INSTRUCTION REGISTER
JMP JEAU
Address of first micro-instruction for all but three EAU instructions (See EAU Direct Group)
EAU DIRECT GROUP
Address of first micro-instruction for DLD,DST, or DIV EAU instructions
TO NEXT PAGE

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

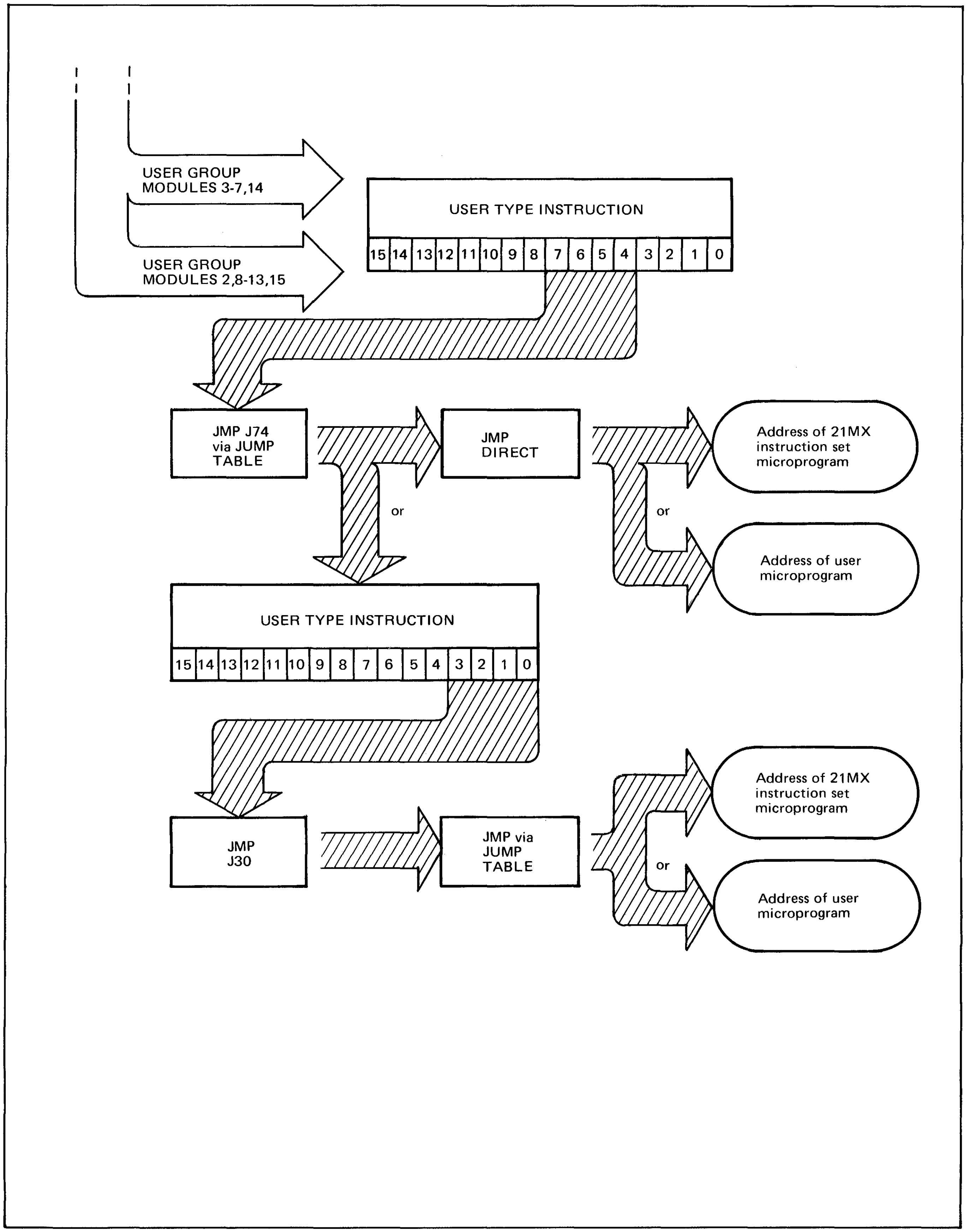


Figure 3-3. Processing the Instruction Register (Sheet 2 of 2)

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.

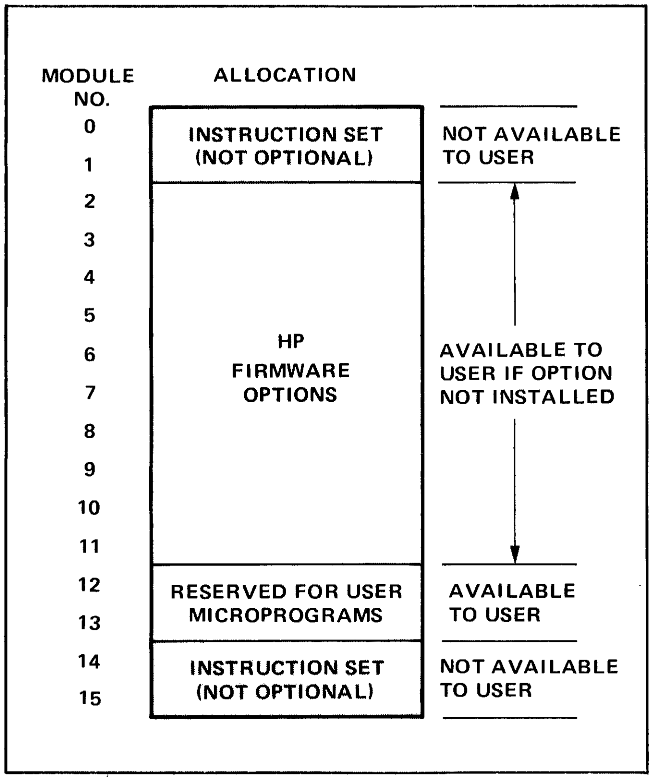

MODULE NO.
ALLOCATION
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
INSTRUCTION SET (NOT OPTIONAL)
NOT AVAILABLE TO USER
HP FIRMWARE OPTIONS
AVAILABLE TO USER IF OPTION NOT INSTALLED
RESERVED FOR USER MICROPROGRAMS
AVAILABLE TO USER
INSTRUCTION SET (NOT OPTIONAL)
NOT AVAILABLE TO USER

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 backplane I/O signals to the I/O devices starting at the next T2 time and according to the contents of the Instruction Register (IR). These are separate and different from micro-orders.

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. If Memory Protect is enabled, the IR must be loaded prior to issuing IOG (see section 3.34).

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. Backplane I/O Signal Generation Determined by IR Bits 11-6

IR*						BACKPLANE I/O SIGNAL	BACKPLANE I/O SIGNAL TIME	GENERAL USE
11	10	9	8	7	6			
x	x	f	0	0	0	none	T3	Turns off the Run Flag on the CPU.
x	x	0	0	0	1	STF	T3	Set device flag.
x	x	1	x	x	x	CLF	T4	Clear device flag.
x	x	0	0	1	0	SFC	T3-T5	SKPF condition is true if and only if the device flag is clear.
x	x	0	0	1	1	SFS	T3-T5	SKPF condition is true if and only if the device flag is set.
x	x	f	1	0	x	IOI	T4	If the corresponding select code *is not* between 1 and 7 (during T4 only), buffer the input data latch on the device onto the I/O-bus.
							T5	Buffer the input data latch on the device onto the I/O-bus.
x	x	f	1	1	x	IOO	T3-T4	Store the I/O-bus into the input data latch on the device.
0	x	f	1	1	1	STC	T4	Set device control flag.
1	x	f	1	1	1	CLC	T4	Clear device control flag.

*Bits marked with x are not significant for the I/O signal specified. If bit 9 is set the device flag is cleared. If bit 9 is not set the device flag is not altered.

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 interrupt 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
HLT

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 IR must specify IOO also. The IOO sends data from the I/O-bus into the input data latch on the device. (Do not confuse this with the IOO backplane I/O signal in table 3-2.) 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 (see IOO store micro-order for register restrictions) 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)		PASS	IOO	CAB
(Time T5)	RTN	PASS	IOO	CAB

3-31. I/O INPUT ROUTINE

This routine is characterized by use of IOI in the S-bus field. The IR must specify IOI also. IOI is used in the I/O cycle during T5 to input data from the I/O device PCA onto the I/O-bus and then onto the S-bus. (Do not confuse this with the IOI backplane I/O signal in table 3-7.) 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. The CIR micro-order should always be used when READ or WRTE micro-orders are used. 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. When MP is enabled:

   1. Only I/O instructions with a select code of 01 are executed.
   2. The IR must be loaded prior to initiating an I/O cycle with the IOG to insure that the signal decoding logic takes effect.

   When MP is not enabled:

   1. No restriction is placed on select codes which are otherwise valid.
   2. The IR may be loaded during the execution of a micro-instruction initiating the I/O cycle with IOG.

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 micro-instruction execution is 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. Do not modify M-register until two micro-instructions after the READ.

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-instructions 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 microprocessor 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 and has not completed its memory reference.

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

j. Load the IR before issuing IOG unless there is no chance that Memory Protect is enabled (no Memory Protect on 2105).

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.

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

$ORIGIN=2000B
$SYNTAB
              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

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.

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=6000
$FILE=FILMV
MOVE                        S1     A         WORD COUNT = 0?
        JMP     CNDX   TBZ         OUT       IF ZERO, THEN GO TO "OUT"
*
        READ           INC  M      P         GET "TO" ADDRESS
                            S2     TAB       PUT IT IN S2
*
LOOP    READ           INC  M      B         READ A DATA WORD
                       INC  B      B         INCREMENT "FROM" ADDRESS
                            S3     TAB       STORE THE WORD IN S3 REG
                       INC  M      S2        GET "TO" ADDRESS
                       INC  S2     S2        INCREMENT "TO" ADDRESS
        WRTE                TAB    S3        WRITE A DATA WORD TO MEMORY
                       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
```

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  —     —            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
      —     —      INC   B    B
      —     —            S2   TAB
```

Get the TO address and store it in Scratch Pad Register 2. Increment the FROM address pointer. 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  —            TAB  S3
```

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

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

Increment the word count. If the word count 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=60178
          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. $\left.\begin{array}{l}\text{S1} = 00050 \\ \text{L} = \text{nn (select code)}\end{array}\right\}$ 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.

```
—      —      ICNT   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
                  ICNT     PASS  L     S1        CNTR=X01
                  L4       AND   S1    LDR       FORM XXXXAAAABBBBXXXX IN S1;CNTR=X01
                  ICNT     PASS  L     S1        CNTR=X10
                  L4       AND   S1    LDR       FORM AAAABBBBCCCCXXXX IN S1;CNTR=X10
                  ICNT     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.

```
  —     —     ICNT   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.

```
  —     —     ICNT   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

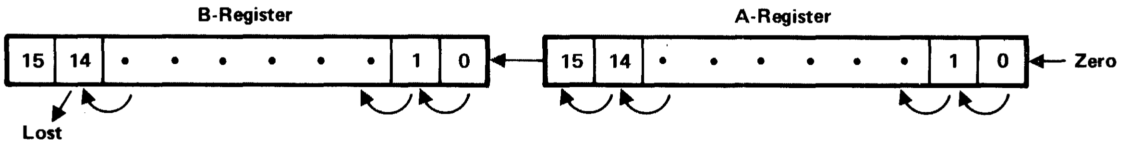
Equivalent micro-instruction binary fields:

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.

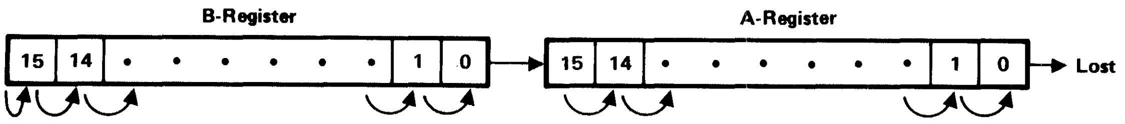
If L1, a 0 is shifted into bit 0 of the A-register; bit 14 of the B-register is lost, but the sign bit remains unchanged. The overflow register bit is set if bits 14 and 15 differ before the shift operation.

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	Complement 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

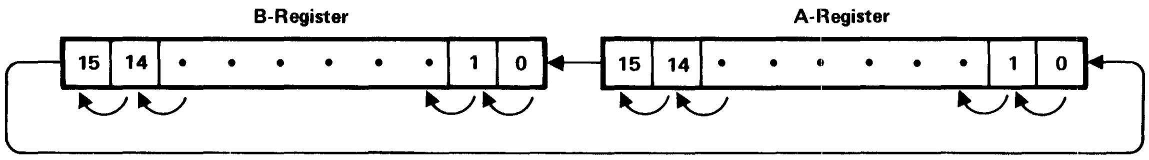
Equivalent micro-instruction binary fields:

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.

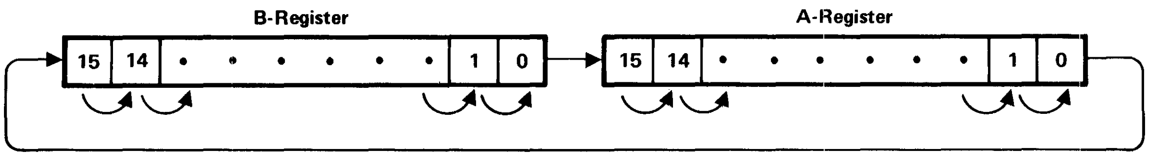
If L1, bit 15 of the B-register is transferred to bit 0 of the A-register.

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. When performing 16 divide steps, an L1 micro-order must be executed before the first divide step for proper bit alignment for the division.

Initial Contents:

B-register		A-register	L-register
Dividend 16 Most Significant bits	. . .	Dividend 16 Least Significant bits	Divisor

After Repeat 16 Times of Divide Step:

Remainder Doubled	16 Bit Quotient of (B,A)/L	Divisor (unchanged)

OP	BIT NO.	23	22	21	20
ENV	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	BIT NO.	23	22	21	20
ENVE	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

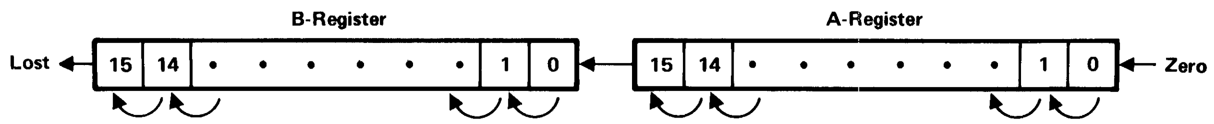
Required micro-instruction binary fields:

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.

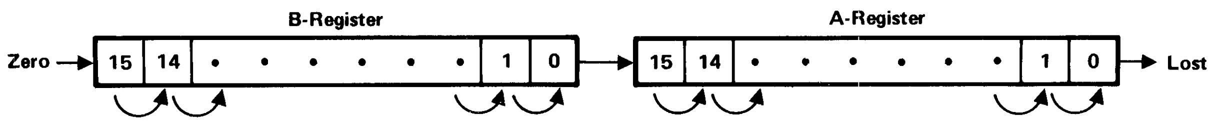
If L1, a 0 is shifted into bit 0 of the A-register and bit 15 of the B-register is lost.

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.

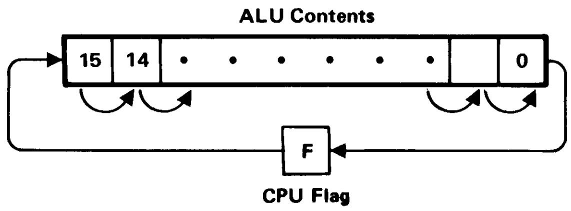
LOGICAL RIGHT SHIFT: SPECIAL=R1



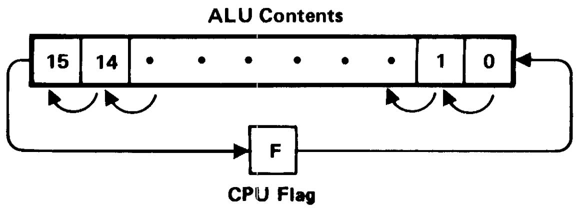
OP	BIT NO.	23	22	21	20
LWF	CONTENT	0	1	1	0

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 clears the flag.

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):

$$(B,A) = [(A x L) + B]$$

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
READ

BIT NO.	23	22	21	20
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 before FTCH or IAK appears in the Special field, 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. 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.**

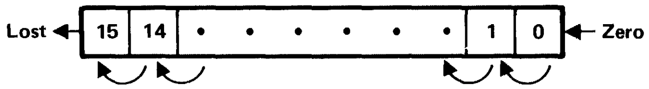
SPECIAL	BIT NO.	4	3	2	1	0
JTAB	CONTENT	1	1	0	1	1

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 3-3).

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

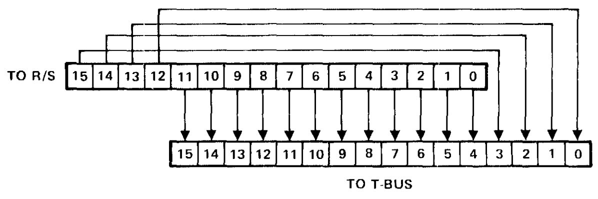
Meaning: Left one bit command to the Rotate/Shifter.



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. The M-register must hold the address to be checked, when the micro-instruction using MPCK executed and this M-register value must not change until the write operation is executed.

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	1	1	1	1

Meaning: No SPECIAL operation is performed.

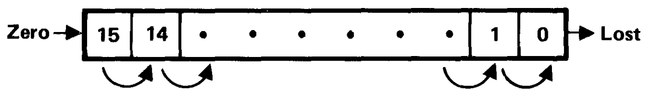
Usage: This is the default operation if none is specified in the SPECIAL field.

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 ($L \cdot S$); 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 ($\bar{S} \cdot L$); 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 ($\bar{S}+L$); 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 $\overline{L}$ 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 (S•$\overline{L}$) 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 $\overline{L}$ 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. (Default for blank ALU field.)

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.

ALU	BIT NO.	19	18	17	16	15
OP13	CONTENT	1	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 third and fourth instructions (T4 and T5) after IOG Special micro-order. When using this micro-order the S-bus is restricted to the A/B registers or the S5-register. 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.

Usage: S-bus field must not contain a CAB micro-order.

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: The ALU micro-order, 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 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 Counter).

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.

STORE		STORE	BIT NO.	9	8	7	6	5
S1	THRU	S12	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 only 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. Always use CIR in conjunction with a READ or WRTE micro-order, even if the location referenced is not used.

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. The upper 10 bits of the S-bus are set to ones.

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.

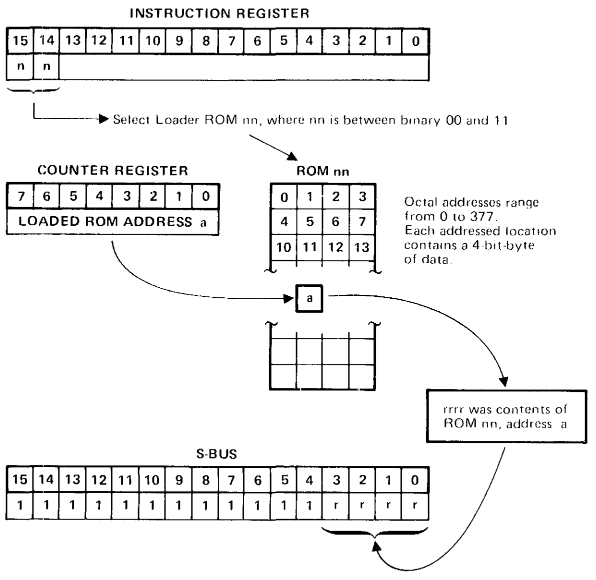
S-BUS	BIT NO.	14	13	12	11	10
IOI	CONTENT	0	0	1	0	0

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.


INSTRUCTION REGISTER
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
n n
Select Loader ROM nn, where nn is between binary 00 and 11
COUNTER REGISTER
7 6 5 4 3 2 1 0
LOADED ROM ADDRESS a
ROM nn
0 1 2 3
4 5 6 7
10 11 12 13
Octal addresses range from 0 to 377. Each addressed location contains a 4-bit-byte of data.
a
rrrr was contents of ROM nn, address a
S-BUS
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
1 1 1 1 1 1 1 1 1 1 1 1 r r r r

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 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. MICROPROGRAMMING INFORMATION FOR DYNAMIC MAPPING

This section will serve to introduce the microprogrammer/hardware designer to the handicaps and versatilities of the MEM when controlled by the CPU or other intelligent system organ.

Use these guidelines:

1. There are three signals generated directly from control store used to control the MEM. In the special field, 'MESP' generates MESP. In the store field, 'MEU' generates MEST. In the S-bus field, 'MEU' generates MEEN.
2. Other signals which directly affect the MEM are MPCK, READ, TEN, IAK (CIREN).
3. Table 4-1 indicates what control line signal (Q_0–Q_7) is enabled by each combination of MEM micro-orders. The three micro-orders are used in a one-of-eight command structure as opposed to have three specific functions.
4. Table 4-2 lists all the functions performed by each of the MEM control signals. These functions are performed only during the microcycle during which they are asserted with the exception of Q_4, port 1.
5. A good feel for the microcode control can be gained by examination of the supplied routines (Appendix E).

For additional control:

1. When issuing a Q_5 command, further information is needed to indicate which utility register you wish to store information into.
2. Since the information has been presented on the S-bus and none of the register require more than 11 bits of information several S-bus bits are reserved for determination of which register is activated.
3. Bit 14 indicates that the State Registers are to be loaded (i.e., enable/disable MEM; select System/User). Bits 9, 8 contain the status information.
4. Bit 13 indicates that the Address Register is to be loaded. Bits 7-0 contain the address information.
5. If a Q_4 has preceded this step by exactly one microcycle (i.e., Q_4, Q_5 in a row), then bit 14 will indicate that the Fence Register is to be loaded. Bits 10-0 contain the fence information.
6. Bit 15 is used to override the Protected Mode, thus allowing these registers (specifically the State Registers) to be altered under microprogram control at any time.

Table 4-1. MEM Signals Invoked by Microcode

LABEL	OP	SPEC	ALU	STORE	S-BUS	MEM SIGNAL
@	@	MESP	@	MEU	MEU	Q_0
@	@	MESP	@	MEU	$	Q_1
@	@	MESP	@	$	MEU	•
@	@	MESP	@	$	$	•
@	@	*	@	MEU	MEU	•
@	@	*	@	MEU	$	
@	@	*	@	$	MEU	
@	@	*	@	$	$	Q_7

@ — Any legal code
* — Any legal code except MESP
$ — Any legal code except MEU

Table 4-2. MEM Microcode Control Signals

Q_0	1. Enable SYS/USR map to S-bus per MEAR bit 5: 0 for SYS, 1 for USR 2. Store S-bus into PORTA/PORTB map per MEAR bit 7: 0 for PORTA, 1 for PORT B 3. Relative map address specified by MEAR bits 4-0
Q_1	1. Store S-bus into maps per MEAR bits 6,5: 00-=SYS, 01=USR, 10=PORTA, 11=PORTB 2. Relative map address specified by MEAR bits 4-0
Q_2	1. Enable maps to S-bus per MEAR bits 6,5: 00-=SYS, 01=USR, 10=PORTA, 11=PORTB 2. S-bus bits 13-10 are always low 3. Relative map address specified by MEAR bits 4-0
Q_3	1. Select opposite program map 2. Can generate DMAFRZ to CPU
Q_4	1. Set "Status Command" flag through next microprocessor cycle 2. Reset to currently selected map (nullifies Q_3)
Q_5	1. Store S-bus into MEM (other than maps) a) MEM State Register = S-bus bits 9,8: bit 9 = 0 = diable MEM, 1 = enable MEM; bit 8 = 0 = select SYS map, 1 = select USR map. b) MEM Fence Register = S-bus bits 10-0 c) MEM Address Register = S-bus bits 6-0 2. Register selected by S-bus bits 15-13: 000 = Fence Register, 001 = Address Register, 010 = State Register. If Base Page Fence is to be selected, Q_5 must be immediately preceded by Q_4.
Q_6	1. Enable MEM data (other than maps) onto S-bus a) Normally enables MEM Violation Register b) If preceded by Q_4, MEM Status Register enabled
Q_7	1. No MEM microcode specified (NOP state for MEM)
Notes:	MEAR is the MEM Address Register MAP bits 9-0 are transferred to/from S-bus bits 9-0 MAP bits 11,10 are transferred to/from S-bus bits 15,14

4-8. WORD TYPE 2 — IMMEDIATE DATA

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	" IMM "	SPECIAL	MODIFIER	STORE	OPERAND	COMMENTS

Figure 4-3. Word Type 2 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	"IMM" op CODE						OPERAND								STORE					SPECIAL				
					MODIFIER																			

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. When using the micro-order to store data in the M-register, the previous micro-instruction must not contain T or TAB in the S-bus field.

4-9. "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.

4-10. 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.

S-Bus	BIT NO.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	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			
CMLO	BIT NO.	19	18
	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.

S-Bus	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							

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			
HIGH	BIT NO.	19	18
	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.

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	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							

4-11. 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

4-12. 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	"JMP" op CODE				CONDITION						OPERAND									"CNDX" SPECIAL CODE				
										JUMP SENSE														

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.

4-13. "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.

4-14. "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.

4-15. 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. (85 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.

4-16. 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.

4-17. 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

4-18. 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.

4-19. "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).

4-20. 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	1

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.

4-21. 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

4-22. PSEUDO INSTRUCTIONS

There are five pseudo instructions recognized by the micro-assembler: DEF, EQU, ONES, SKP, and ZEROES.

4-23. DEF

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

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

4-24. 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 microprogram 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

4-25. 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

4-26. 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

4-27. 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	"ZEROES"	(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	ZEROES	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.

Refer to the HP 12978A Writable Control Store Reference Manual, part no. 12978-90007 for a summary of microprogramming software part numbers.

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.

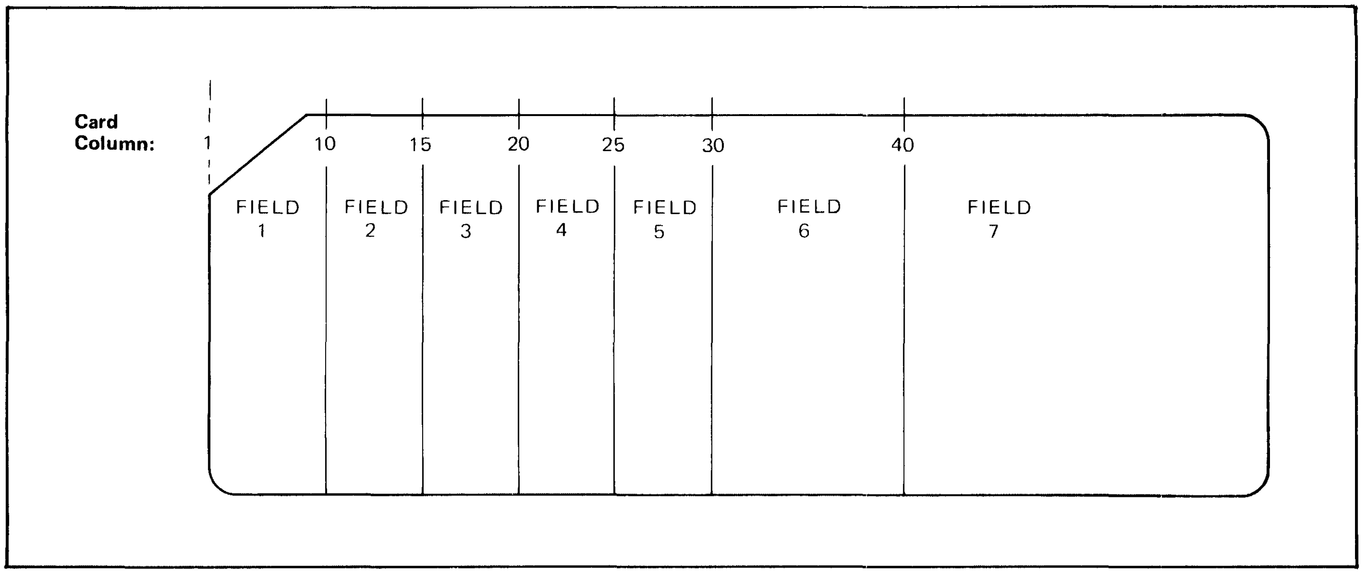


Figure 5-1. Micro-instruction Card Source Record

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*name2*b*address2, *b*. . .namen*b*addressn

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 . . . namen refers to addressn
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. 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

$NOPUNCH

General Form:	$NOPUNCH
Meaning:	Suppress punching of binary object tape.
Purpose:	To perform a micro-assembly for listing and diagnosis only.
Example:	$NOPUNCH

$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 5-3 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.

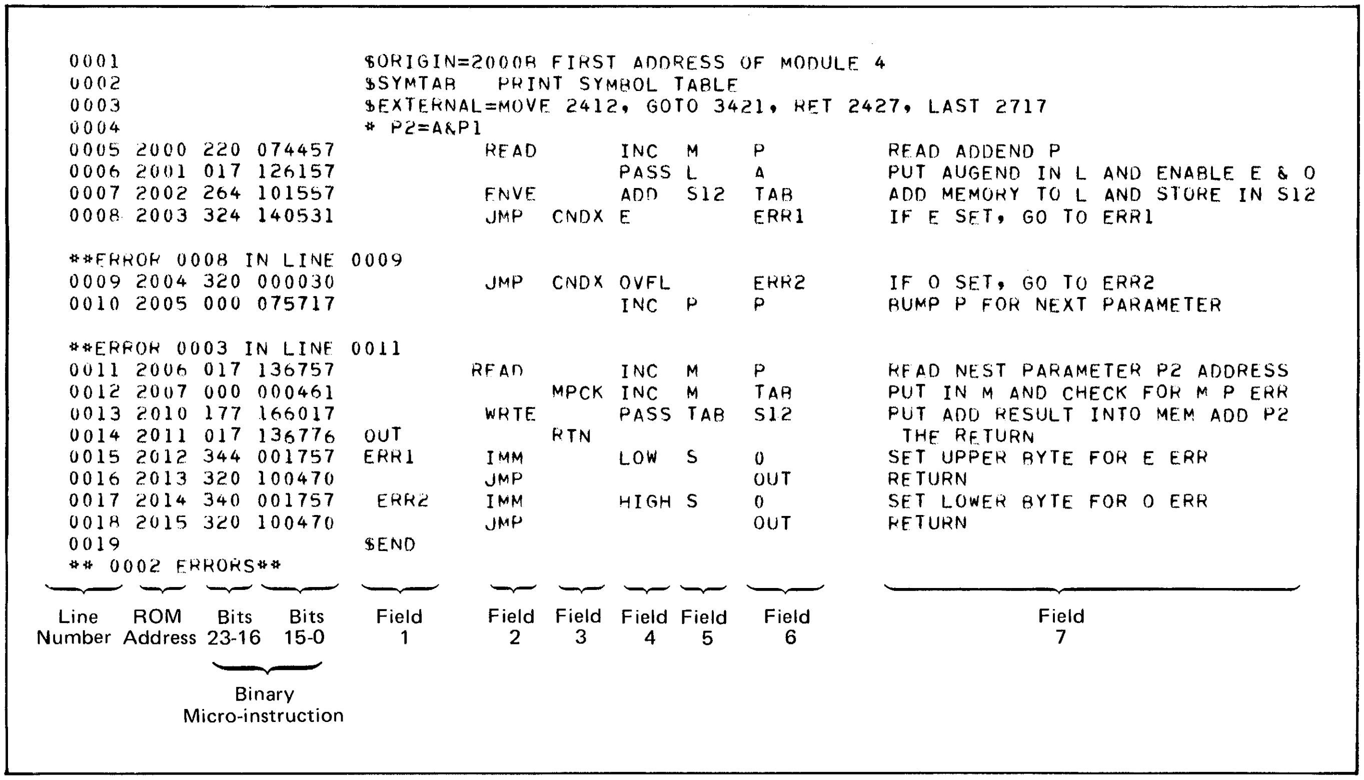
a. A current DOS-III system.

b. A source microprogram, on cards, paper tape, or in a source file on disc. If input is from a disc file, the file must be on the system subchannel.

c. The Micro-assembler program named MICRO stored in the DOS-III user library. If MICRO still is on relocatable object paper tape (HP 12978-16003), 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).

Currently, if MICRO is included in the system area during DOS system generation, base page linking must be



```
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**
```

Figure 5-3. Micro-Assembly Listing

specified. Thus, when generating the system, the answer to the question

ENTER PROG PARAMETERS

must include

MICRO,3,1

where the 1 indicates base page linking. This is necessary because the distributed version of the micro-assembler is set to current page linking and does not executed properly in the system area.

When an executable version of the micro-assembler is properly included in the DOS system, perform the following steps to assemble the microprogram.

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.

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.

b. Place the microprogram source in the input device; turn the device on; turn on the paper tape punch and the list device.

c. Summon the Micro-assembler with statement

   :PR,MICRO,[p1,p2,p3,p4,99]

where

p1 = the input device logical unit number. If input is from a disc file, the file must be on the system subchannel

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 relocatable object tape of the Micro-assembler program MICRO (HP 12978-16003).
- 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.

   ENTER LU NUMBER OF DEVICE

c. Enter $INPUT = <LU #> 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 (T-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 (T-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 the HP 12978A Writable Control Store Reference Manual, part no. 12978-90007 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

```
ASMB,R,B,L,T                           Assembly parameters
      NAM TEXT,7                       Program name
      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

program tapes), the user must supply an initialization program. The initialization program is an assembly language program that prepares the necessary parameters in 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.

An example of a short initialization program is shown in Figure 5-5.

5-16. USING THE MICRO DEBUG EDITOR

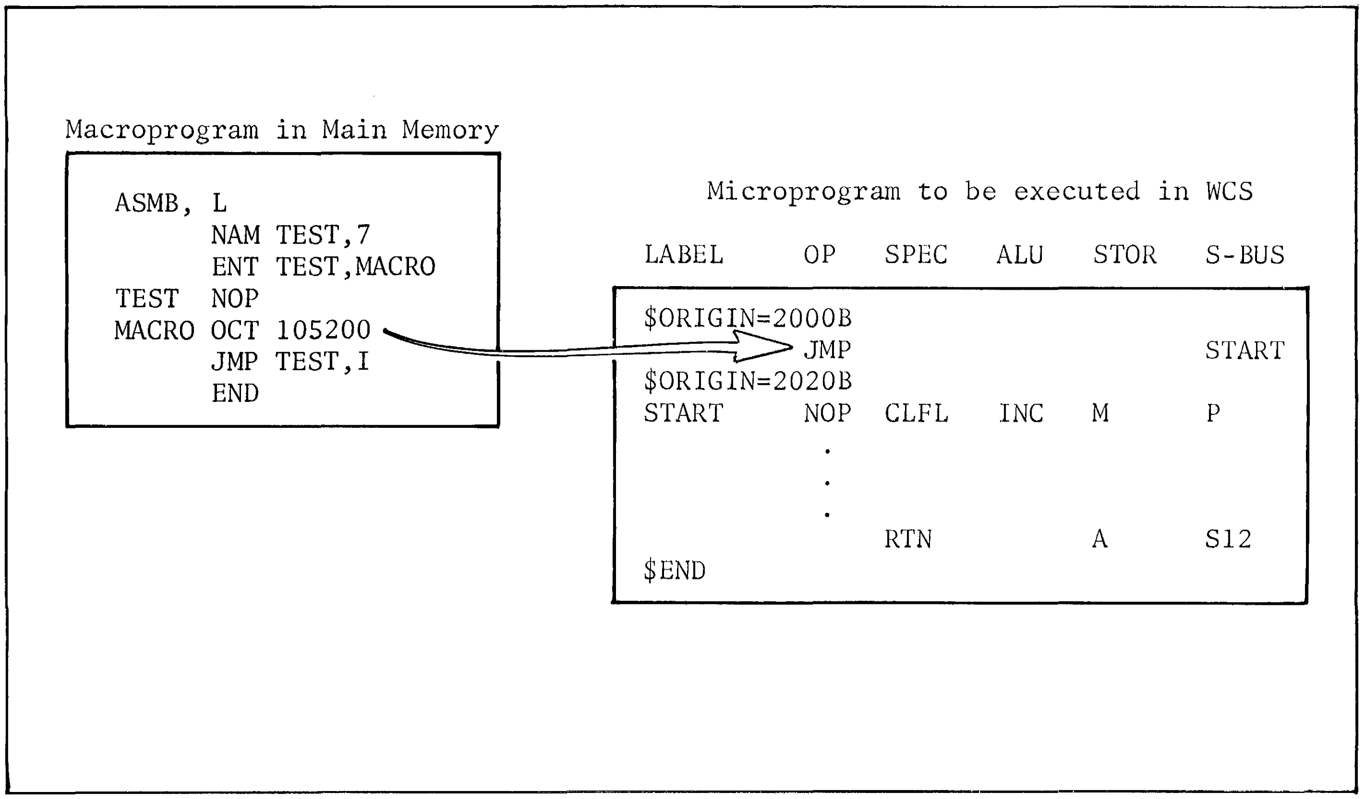
Section 5-37 describes how to execute MDE using the DOS-III operating system. Section 5-38 describes how to execute MDE using the BCS operating system.

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 the WCS Reference manual (12978-90007) 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.


Macroprogram in Main Memory
Microprogram to be executed in WCS

```
ASMB, L
      NAM TEST,7
      ENT TEST,MACRO
TEST  NOP
MACRO OCT 105200
      JMP TEST,I
      END
```

```
LABEL        OP    SPEC   ALU    STOR   S-BUS
$ORIGIN=2000B
             JMP                        START
$ORIGIN=2020B
START        NOP   CLFL   INC    M      P
              .
              .
              .
                   RTN           A      S12
$END
```

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 pROM 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.

Note that a READ requires a prior WRITE command to establish the relationship between logical unit and module number.

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 absolute WCS address and **yyyy** is the ending absolute 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 (underlined characters indicate operator input):

```
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 microcode 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. (Underlined characters indicate operator input.)

   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 **ll** (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 (underlined characters indicate operator entry)

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.

d. at a WRITE micro-order

e. at a READ micro-order if the M-register is not loaded in the same micro-instruction.

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 execution.

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 Microassembler.

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 or MACRO 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 tape output by the micro-assembler. Appendix A illustrates the binary object tape format.

When the microprogram has been stored in the Main Memory program buffer, a WCS I/O Utility calling sequence is used to write the microprogram into the WCS.

To read the contents of the WCS, another 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 WWRIT	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.

OBJECT TAPE FORMATS

APPENDIX A

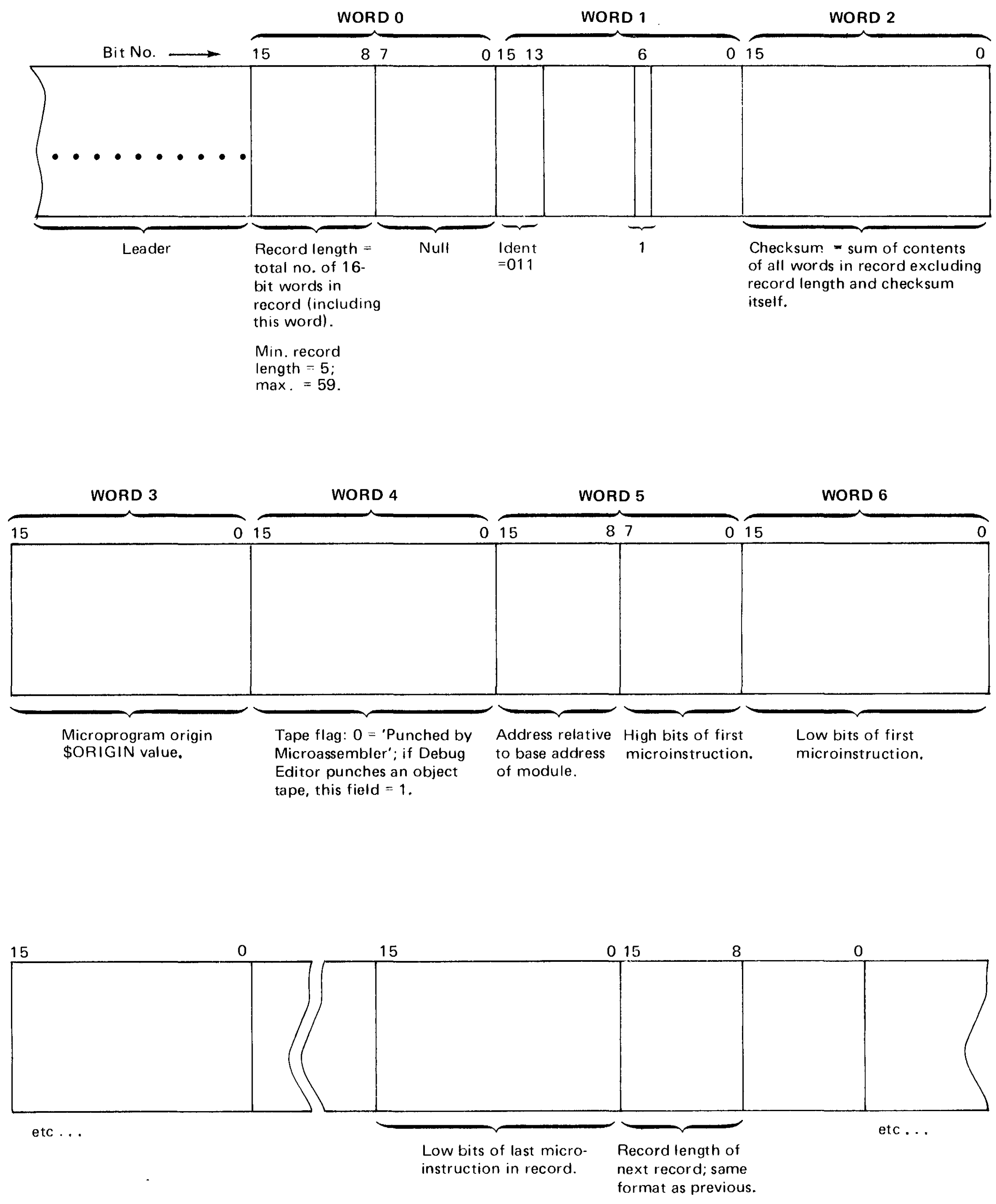

WORD 0
WORD 1
WORD 2
Bit No.
15 8 7 0 15 13 6 0 15 0
Leader
Record length = total no. of 16-bit words in record (including this word).
Min. record length = 5; max. = 59.
Null
Ident =011
1
Checksum = sum of contents of all words in record excluding record length and checksum itself.
WORD 3
WORD 4
WORD 5
WORD 6
15 0 15 0 15 8 7 0 15 0
Microprogram origin $ORIGIN value.
Tape flag: 0 = 'Punched by Microassembler'; if Debug Editor punches an object tape, this field = 1.
Address relative to base address of module.
High bits of first microinstruction.
Low bits of first microinstruction.
15 0 15 0 15 8 0
etc . . .
Low bits of last micro-instruction in record.
Record length of next record; same format as previous.
etc . . .

Figure A-1. Format of Standard Object Tape (Sheet 1 of 2)

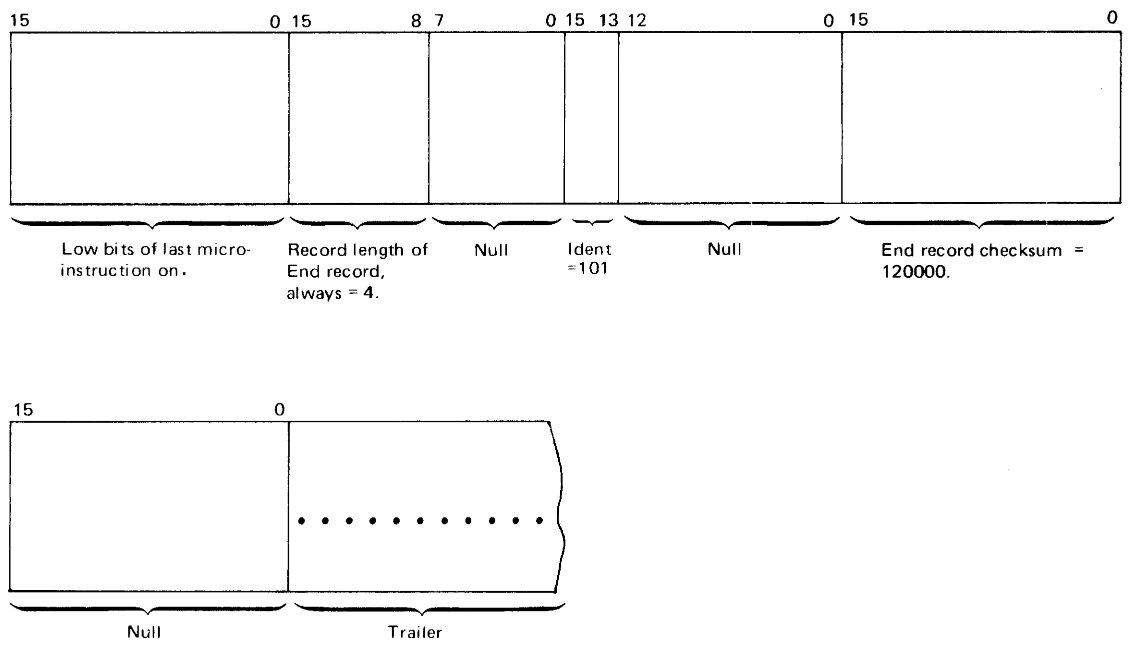

15
0
15
8
7
0
15
13
12
0
15
0
Low bits of last micro-instruction on.
Record length of End record, always = 4.
Null
Ident =101
Null
End record checksum = 120000.
15
0
Null
Trailer

Figure A-1. Format of Standard Object Tape (Sheet 2 of 2)

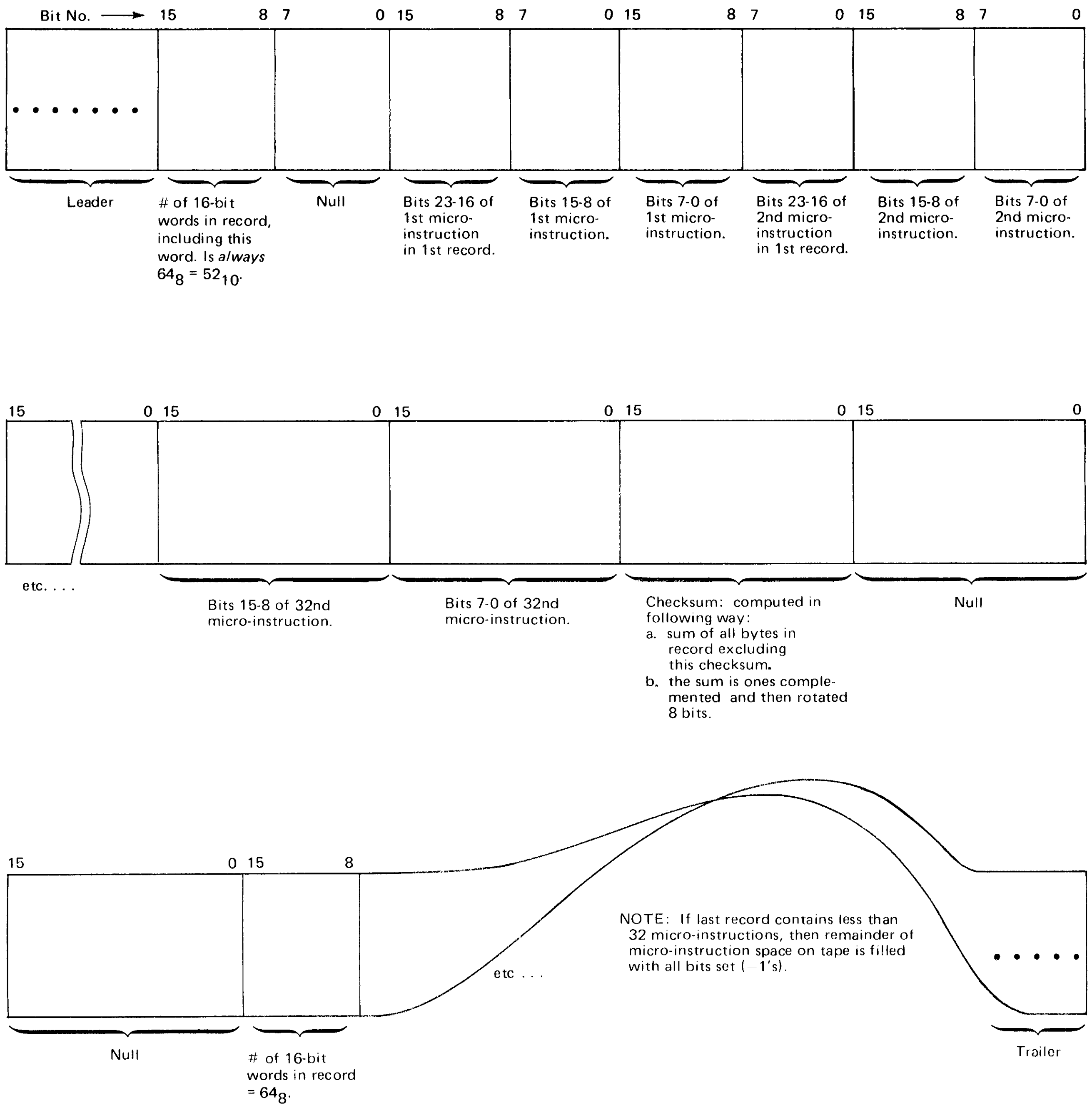
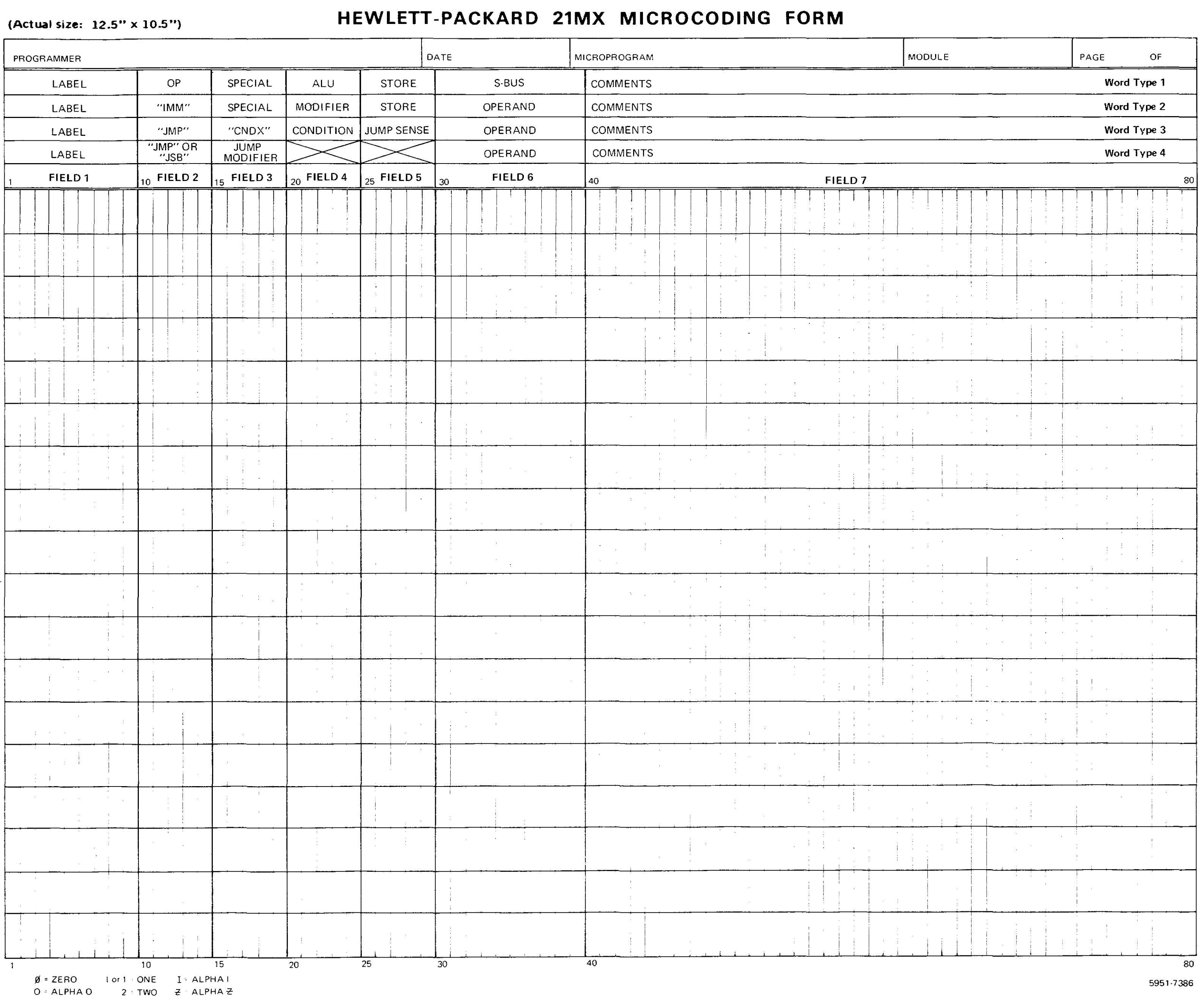

Bit No. → 15 8 7 0 15 8 7 0 15 8 7 0 15 8 7 0
Leader
# of 16-bit words in record, including this word. Is *always* $64_8 = 52_{10}$.
Null
Bits 23-16 of 1st micro-instruction in 1st record.
Bits 15-8 of 1st micro-instruction.
Bits 7-0 of 1st micro-instruction.
Bits 23-16 of 2nd micro-instruction in 1st record.
Bits 15-8 of 2nd micro-instruction.
Bits 7-0 of 2nd micro-instruction.
15 0 15 0 15 0 15 0 15 0
etc. . . .
Bits 15-8 of 32nd micro-instruction.
Bits 7-0 of 32nd micro-instruction.
Checksum: computed in following way:
a. sum of all bytes in record excluding this checksum.
b. the sum is ones complemented and then rotated 8 bits.
Null
15 0 15 8
etc . . .
NOTE: If last record contains less than 32 micro-instructions, then remainder of micro-instruction space on tape is filled with all bits set (–1's).
Null
# of 16-bit words in record = 64_8.
Trailer

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 SOURCE (CARD)	OP	SPECIAL	ALU	JMP COND	IMMEDIATE MODIFIER	STORE	RJS	S-BUS
COLUMN NO.	10	15	20	20	20	25	25	30
BITS (ROM)	23-20	4-0	19-15	19-15	19-18	9-5	14	14-10
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	CMLO	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	RESP		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		*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	OP13	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		JEAU	*PASS	NMEU		S		S

*default micro-order

**JMP default

†If no 'RJS', then bit 14 = 1

░ means not normally used by user microprogrammer.

▓ means included here for completeness only; reserved for exclusive use of system microprogrammers.

FUNCTIONAL BLOCK DIAGRAM

APPENDIX

D

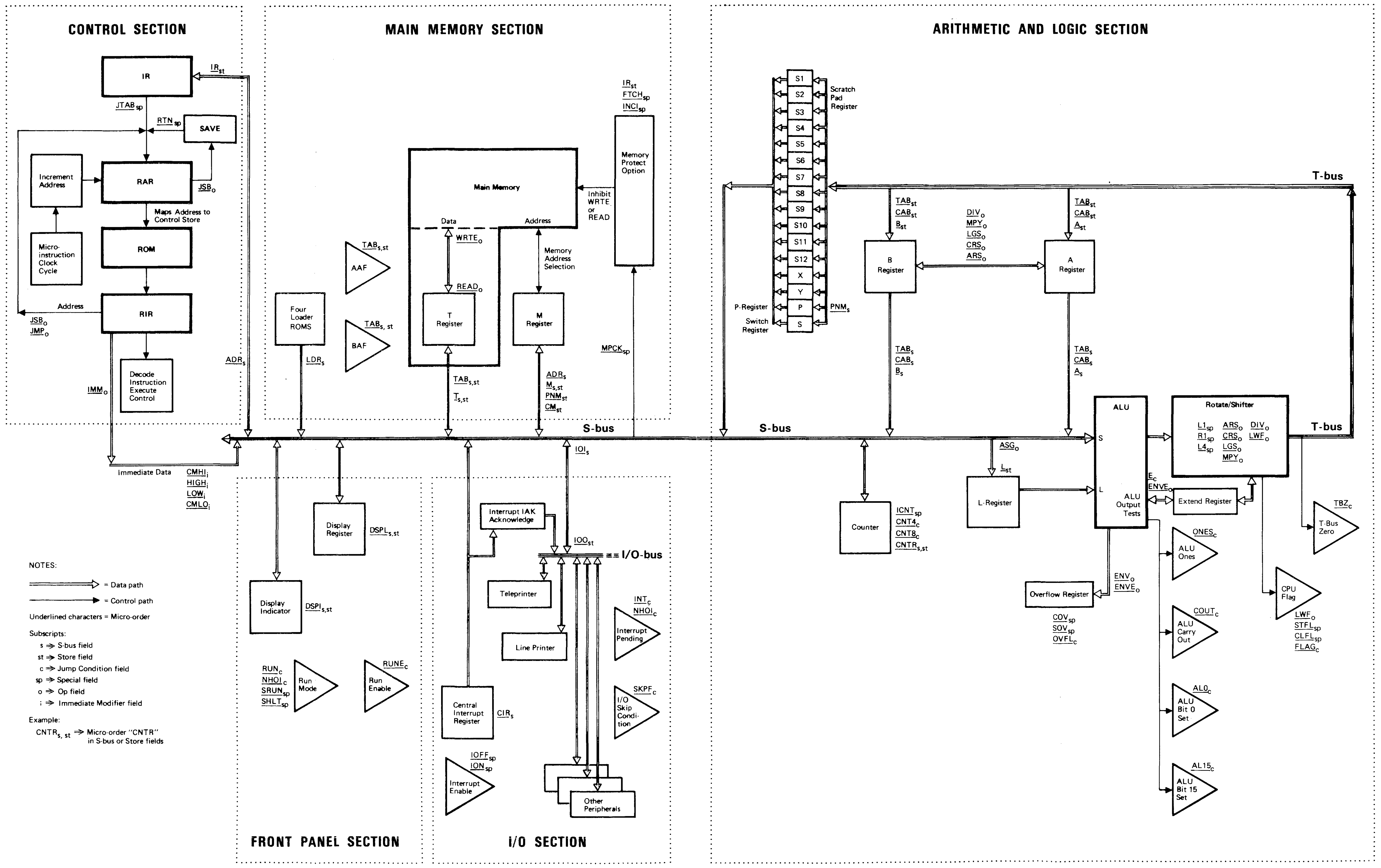


Figure D-1. Functional Block Diagram

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.

```
                            ORG                 0
                   **************************************************************************
                   *
                   *      21MX MICRO-CODE
                   *      MODULE 0
                   *
                   **************************************************************************
                   FADD     EQU                 %7125
                   FSUB     EQU                 %7126
                   FMPY     EQU                 %7221
                   FDIV     EQU                 %7262
                   IFIX     EQU                 %7000
                   FLOAT    EQU                 %7025
                   **************************************************************************
                   *      FETCH ROUTINE
                   **************************************************************************
00000  220 074712  FETCH    READ FTCH INC  PNM  P         M<=P; P<=P+1; READ NEW INSTR.
00001  017 136745                ION                      ENABLE INTERRUPT RECOGNITION
00002  017 100411                CLFL PASS IR   TAB       IR<= T/A/B; CLR FLAG FF
00003  220 020673           READ JTAB INC  CM   ADR       JUMP THRU TABLE; LOAD M IF MRG IN
                   **************************************************************************
00004  325 120031  HORI     JMP  CNDX RUN  RJS  HALT      RUN MODE IMPLIES AN INTERUPT
                   **************************************************************************
                   *      INTERRUPT RESPONSE ROUTINE
                   **************************************************************************
00005  237 106451  INTERUPT READ CLFL PASS M    CIR       M<=CIR; READ TRAP CELL; CLR FLAG
00006  320 000471           JMP  CNDX TBZ  RJS  INTOK     CHECK IF CIR IS VALID
00007  237 106457           READ      PASS M    CIR       M<=CIR; READ TRAP CELL
00010  320 040031           JMP  CNDX TBZ       FETCH     IF NO INT BY NOW, IGNORE
00011  017 104400  INTOK         IOFF PASS IR   T         IR<= TRAP CELL, DISABLE INT RECOG
00012  220 020673           READ JTAB INC  CM   ADR       JUMP THRU TABLE; LOAD M IF MRG IN
                   **************************************************************************
                   *      INDIRECT ROUTINE
                   **************************************************************************
00013  220 022457  INDLEVEL READ      INC  M    M         READ NEXT LEVEL
00014  326 001031           JMP  CNDX NHOI RJS  IND2      HALT OR INTERRUPT?
00015  017 100465  INDIRECT      INCI PASS M    TAB       M<=T/A/B; INCR INDIRECT COUNT
00016  322 040571           JMP  CNDX AL15      INDLEVEL  CHECK FOR ANOTHER LEVEL OF INDIRE
00017  220 022476           READ RTN  INC  M    M         READ EFFECTIVE ADDRESS, RETURN
00020  017 100465  IND2          INCI PASS M    TAB       M<=T/A/B; INCR INDIRECT COUNT
00021  330 100731           JMP  CNDX NSNG RJS  INDIRECT+1JUMP BACK FOR SINGLE INSTRUCTION
00022  007 175717                     DEC  P    P         RESET P
00023  320 000230           JMP                 HORI      HALT OR INTERUPT
```

```
                    ********************************************************************
                    *      ALTER-SKIP GROUP
                    ********************************************************************
00024  017 102757   ASGNOP                  PASS       CAB        SET UP SKIP TEST
00025  327 042431             JMP   CNDX ASGN          ASGNSKP    JUMP IF ASG SKIP NOT MET
00026  200 075717             ASG        INC  P        P          P<=P+1; ENABLE ASG HARDWARE
00027  327 100031             JMP   CNDX IR2  RJS      FETCH      DONE IF NOT INA/B
00030  260 002076             ENVE  RTN  INC  CAB      CAB        A/B <= A/B PLUS 1
                    *
00031  001 136057   ASGCL*                  ZERO CAB              CLEAR  A/B REGISTER
00032  327 042431             JMP   CNDX ASGN          ASGNSKP    JUMP IF ASG SKIP NOT MET
00033  200 075717             ASG        INC  P        P          P<=P+1; ENABLE ASG HARDWARE
00034  327 100031             JMP   CNDX IR2  RJS      FETCH      DONE IF NOT INA/B
00035  260 002076             ENVE  RTN  INC  CAB      CAB        A/B <= A/B PLUS 1
                    *
00036  010 002057   ASGCM*                  CMPS CAB   CAB        A/B <= NOT A/B
00037  327 042431             JMP   CNDX ASGN          ASGNSKP    JUMP IF ASG SKIP NOT MET
00040  200 075717             ASG        INC  P        P          P<=P+1; ENABLE ASG HARDWARE
00041  327 100031             JMP   CNDX IR2  RJS      FETCH      DONE IF NOT INA/B
00042  260 002076             ENVE  RTN  INC  CAB      CAB        A/B <= A/B PLUS 1
                    *
00043  016 036057   ASGCC*                  ONE  CAB              CLR & COMP A/B REGISTER
00044  327 042431             JMP   CNDX ASGN          ASGNSKP    JUMP IF ASG SKIP NOT MET
00045  200 075717             ASG        INC  P        P          P<=P+1; ENABLE ASG HARDWARE
00046  327 100031             JMP   CNDX IR2  RJS      FETCH      DONE IF NOT INA/B
00047  260 002076             ENVE  RTN  INC  CAB      CAB        A/B <= A/B PLUS 1
                    *
00050  217 136757   ASGNSKP   ASG                                 NO SKIP; ENABLE ASG HARDWARE
00051  327 100031             JMP   CNDX IR2  RJS      FETCH      DONE IF NOT INA/B
00052  260 002076             ENVE  RTN  INC  CAB      CAB        A/B <= A/B PLUS 1
                    ********************************************************************
                    *      SHIFT/ROTATE GROUP
                    ********************************************************************
00053  017 102046   SRG             SRG1 PASS CAB      CAB        FIRST SHIFT
00054  017 102056                   SRGE PASS CAB      CAB        CHECK FOR CLEAR E; SET SLA TEST
00055  335 103031             JMP   CNDX SRGL RJS      *+3        SRGL IS SLA TEST
00056  017 102041                   SRG2 PASS CAB      CAB        SECOND SHIFT
00057  000 075736                   RTN  INC  P        P          P<=P+1, WHEN LSB = 0
00060  017 102041                   SRG2 PASS CAB      CAB        SECOND SHIFT
00061  017 136776   RETURN          RTN
```

```
                    ********************************************************************
                    *               I/O GROUP
                    ********************************************************************
00062  017 136757   IOCNTRL         NOP                           ALLOW TIME TO GET SKIP FLAG
00063  326 100031             JMP   CNDX SKPF RJS      FETCH      CHECK SKIP FLAG
00064  000 075736                   RTN  INC  P        P          P <= P + 1
00065  017 136757                   NOP
                    *
00066  017 102757   IO.OT*               PASS          CAB        SET UP S-BUS
00067  017 102217                        PASS IOO      CAB        I/O-BUS <= A/B
00070  017 102236                   RTN  PASS IOO      CAB        HOLD I/O-BUS VALID
00071  017 136757                   NOP
                    *
00072  017 136757   IO.LI*          NOP                           SYNCHRONIZE IOI PULSE
00073  017 136757                   NOP
00074  017 110076                   RTN  PASS CAB      IOI        A/B <= I/O-BUS
00075  017 136757                   NOP
                    *
00076  017 136757   IO.MI*          NOP                           SYNCHRONIZE IOI PULSE
00077  017 102157                        PASS L        CAB        L <= A/B FOR ALU OPERATION
00100  017 010076                   RTN  IOR  CAB      IOI        A/B<= (A/B) + (I/O BUS)
                    *
                    ********************************************************************
                    *      IO GROUP/ EAU GROUP/ MAC GROUP JUMPS
                    ********************************************************************
00101  320 003122   IOG       JMP   IOG                IOCNTRL
00102  320 015437   EAU       JMP   JEAU               EAUTABLE
00103  320 016034   MAC0      JMP   J74                MACTABL0
00104  320 017034   MAC1      JMP   J74                MACTABL1
```

```
                   ************************************************************************
                   *      MEMORY REFERENCE GROUP
                   ************************************************************************
00105  300 000670  AND,I    JSB                 INDIRECT
00106  017 126157  AND                PASS L    A         L <= A
00107  015 100576                RTN  AND  A    TAB       A <= T/A/B AND L
                   *
00110  300 000670  CP*,I    JSB                 INDIRECT
00111  017 102157  CP*                PASS L    CAB       L <= A/B
00112  013 000757                     XOR       TAB       T-BUS <= T/A/B XOR L
00113  320 040031           JMP  CNDX TBZ       FETCH     JUMP TO FETCH IF EQUAL
00114  000 075736                RTN  INC  P    P         P<= P+1 IF NOT EQUAL
                   *
00115  300 000670  XOR,I    JSB                 INDIRECT
00116  017 126157  XOR                PASS L    A         L <= A
00117  013 000576                RTN  XOR  A    TAB       A <= T/A/B XOR L
                   *
00120  300 000670  IOR,I    JSB                 INDIRECT
00121  017 126157  IOR                PASS L    A
00122  017 000576                RTN  IOR  A    TAB       A <= T/A/B IOR L
                   *
00123  300 000670  ST*,I    JSB                 INDIRECT
00124  017 122761  ST*           MPCK PASS      M         MEM PROTECT CHECK OF ADDRESS
00125  177 102036           WRTE RTN  PASS TAB  CAB       T/A/B <= A/B; WRITE
                   *
                   *
00126  300 000670  AD*,I    JSB                 INDIRECT
00127  017 102157  AD*                PASS L    CAB       L <= A/B
00130  264 100076           ENVE RTN  ADD  CAB  TAB       A/B <= T/A/B PLUS L
                   *
00131  300 000640  JSB,I    JSB  IOFF           INDIRECT  DISABLE INTERRUPT RECOGNITION
00132  017 122761  JSB           MPCK PASS      M         MEM PROTECT CHECKS THIS ADDR
00133  177 174017           WRTE      PASS TAB  P         T/A/B <= RETURN ADDRESS; WRITE
00134  000 023736                RTN  INC  P    M         P <= M + 1
                   *
00135  300 000670  ISZ,I    JSB                 INDIRECT
00136  017 122761  ISZ           MPCK PASS      M         MEM PROTECT CHECKS THIS ADDR
00137  000 001017                     INC  S1   TAB       S1 <= T/A/B + 1
00140  177 140017           WRTE      PASS TAB  S1        T <= S1; WRITE
00141  320 000031           JMP  CNDX TBZ  RJS  FETCH     ZERO?  NO, DONE.
00142  000 075736                RTN  INC  P    P         YES, P <= P+1
                   *
00143  300 000670  LD*,I    JSB                 INDIRECT
00144  017 100076  LD*           RTN  PASS CAB  TAB       A/B <= T/A/B


00145  017 136765  JMP,I         INCI                     COUNT ONE INDIRECT LEVEL
00146  017 101000                IOFF PASS S1   TAB       DISABLE INT RECOGNITION;S1<=T/A/B
00147  322 046531           JMP  CNDX AL15      JINDL     JMP IF ANOTHER LEVEL OF INDIRECT
00150  017 140761                MPCK PASS      S1        MEM PROT CHECKS DESTINATION ADDR
00151  017 141736                RTN  PASS P    S1        P <= DESTINATION ADDR
                   *
00152  220 040457  JINDL    READ      INC  M    S1        READ NEXT LEVEL
00153  326 007031           JMP  CNDX NHOI RJS  HORICK    JMP IF HALT OR INT
00154  017 101025                INCI PASS S1   TAB       S1 <= T/A/B; COUNT INDIRECT LEVEL
00155  322 046531           JMP  CNDX AL15      JINDL     JMP IF ANOTHER LEVEL OF INDIRECT
00156  017 140761                MPCK PASS      S1        MEM PROT CHECKS DESTINATION ADDR
00157  017 141736                RTN  PASS P    S1        P <= DESTINATION ADDR
                   *
00160  017 101025  HORICK        INCI PASS S1   TAB       S1 <= T/A/B;  COUNT INDIRECT LEVE
00161  330 106671           JMP  CNDX NSNG RJS  JINDL+3   JUMP BACK FOR SINGLE INSTRUCTION
00162  007 175717                     DEC  P    P         RESET P
00163  320 000230           JMP                 HORI      HALT OR INTERUPT
00164  017 121021  JMP           MPCK PASS S1   ADR       S1<=DESTINATION ADDR; CHECK WITH
00165  017 141736                RTN  PASS P    S1        P <= DESTINATION ADDRESS
```

```
                   ************************************************************************
                   *       EAU MICROPROGRAMS
                   ************************************************************************
00166  010 021017  RRR                CMPS S1   ADR
00167  000 041017                     INC  S1   S1        S1 <= TWO'S COMP OF SHIFTS
00170  017 140255                RPT  PASS CNTR S1        SET UP COUNTER FOR REPEAT
00171  057 124504           CRS  R1   PASS B    B         DOUBLE-WORD SHIFT REPEAT
00172  017 136776                RTN
                   *
00173  010 021017  ASR                CMPS S1   ADR
00174  000 041014                COV  INC  S1   S1        S1 <= TWO'S COMP OF SHIFTS
00175  017 140255                RPT  PASS CNTR S1        SET UP COUNTER FOR REPEAT
00176  037 124504           ARS  R1   PASS B    B         DOUBLE-WORD SHIFT REPEAT
00177  017 136776                RTN
                   *
00200  010 021017  LSR                CMPS S1   ADR
00201  000 041017                     INC  S1   S1        S1 <= TWO'S COMP OF SHIFTS
00202  017 140255                RPT  PASS CNTR S1        SET UP COUNTER FOR REPEAT
00203  077 124504           LGS  R1   PASS B    B         DOUBLE-WORD SHIFT REPEAT
00204  017 136776                RTN
                   *
00205  010 021017  RRL                CMPS S1   ADR
00206  000 041017                     INC  S1   S1        S1 <= TWO'S COMP OF SHIFTS
00207  017 140255                RPT  PASS CNTR S1        SET UP COUNTER FOR REPEAT
00210  057 124502           CRS  L1   PASS B    B         DOUBLE-WORD SHIFT REPEAT
00211  017 136776                RTN
                   *
00212  010 021017  ASL                CMPS S1   ADR
00213  000 041014                COV  INC  S1   S1        S1 <= TWO'S COMP OF SHIFTS
00214  017 140255                RPT  PASS CNTR S1        SET UP COUNTER FOR REPEAT
00215  037 124502           ARS  L1   PASS B    B         DOUBLE-WORD SHIFT REPEAT
00216  017 136776                RTN
                   *
00217  010 021017  LSL                CMPS S1   ADR
00220  000 041017                     INC  S1   S1        S1 <= TWO'S COMP OF SHIFTS
00221  017 140255                RPT  PASS CNTR S1        SET UP COUNTER FOR REPEAT
00222  077 124502           LGS  L1   PASS B    B         DOUBLE-WORD SHIFT REPEAT
00223  017 136776                RTN
                   *
00224  220 074457  DLD      READ      INC  M    P         READ MEMORY ADDRESS
00225  300 000640           JSB  IOFF           INDIRECT  JSB TO GET M<=ADDR OF FIRST WORD
00226  000 023017                     INC  S1   M         S1 <= ADDRESS OF SECOND WORD
00227  017 100557                     PASS A    TAB       A <= FIRST DATA WORD
00230  220 040457           READ      INC  M    S1        M<=ADDR OF SECOND WORD; READ
00231  000 075717                     INC  P    P         P <= P + 1
00232  017 100536                RTN  PASS B    TAB       B <= SECOND DATA WORD
                   *
00233  220 074457  DST      READ      INC  M    P         READ MEMORY ADDRESS
00234  300 000640           JSB  IOFF           INDIRECT  JSB TO GET M <= ADDR OF FIRST WOR
00235  000 023021                MPCK INC  S1   M         MP CHECK FIRST ADDR; S1<=SECOND A
00236  177 126017           WRTE      PASS TAB  A         STORE A INTO FIRST LOCATION
00237  000 040461                MPCK INC  M    S1        MP CHECK S1; M<=S1
00240  177 124017           WRTE      PASS TAB  B         STORE B INTO SECOND LOCATION
00241  000 075736                RTN  INC  P    P         UPDATE P



00242  220 074457  MPY      READ      INC  M    P         M <= P; READ
00243  300 000640           JSB  IOFF           INDIRECT  JSB TO GET M <= ADDR OF OPERAND
00244  000 075717                     INC  P    P         UPDATE P
00245  017 101057                     PASS S2   TAB       S2 <= MULTIPLIER
00246  017 127114  MPYX          COV  PASS S3   A         S3<=A(MULTIPLICAND); CLEAR OVFL
00247  001 136517                     ZERO B              CLEAR B FOR MULTIPLY
00250  017 142157                     PASS L    S2        L <= S2 (MULTIPLIER)
00251  017 124255                RPT  PASS CNTR B         CLEAR COUNTER; SET REPEAT FF
00252  104 124504           MPY  R1   ADD  B    B         MPY STEP (X16); (B,A)<=A TIMES L
00253  017 144757                     PASS      S3        TEST MULTIPLICAND
00254  322 012731           JMP  CNDX AL15 RJS  *+2       JUMP IF POSITIVE
00255  003 024517                     SUB  B    B         UNDO LAST MPY STEP IF NEGATIVE
00256  017 142757                     PASS      S2        TEST MULTIPLIER
00257  322 003071           JMP  CNDX AL15 RJS  RETURN    JUMP IF POSITIVE
00260  017 144157                     PASS L    S3        L <= MULTIPLICAND
00261  003 024536                RTN  SUB  B    B         B<=B MINUS L (CORRECTS FOR NEG. M
                   *
                   *
```

```
00262  220 074457  DIV      READ      INC  M    P         M <= P; READ
00263  300 000640           JSB  IOFF           INDIRECT  JSB TO GET M <= ADDR OF OPERAND
00264  000 075717                     INC  P    P         UPDATE P
00265  010 001157                     CMPS S4   TAB       S4 <= DVSR(CM);;SAVE ORIG SIGN
00266  010 047017                     CMPS S1   S4        S1 <= DVSR
00267  322 013471           JMP  CNDX AL15 RJS  *+2       JMP IF DVSR NEGATIVE
00270  000 047017                     INC  S1   S4        S1 <= DVSR(2CM)
00271  017 140157                     PASS L    S1        L <= ABS VALUE(DVSR)
00272  010 025117                     CMPS S3   B         S3 <= DVNDHI(2CM)
00273  322 054071           JMP  CNDX AL15      DIVS      JMP IF DVND POSITIVE
00274  017 144517                     PASS B    S3          IF DVND IS NEGATIVE...
00275  010 027057                     CMPS S2   A           FORM DVND(2CM)
00276  000 042557                     INC  A    S2          IN B,A-REGISTER
00277  321 014071           JMP  CNDX COUT RJS  DIVS          *
00300  000 044517                     INC  B    S3            *
00301  003 024753  DIVS          SOV  SUB       B         CHECK FOR DVSR TOO SMALL
00302  322 000031           JMP  CNDX AL15 RJS  FETCH     (=DVND TOO LARGE)
00303  077 124502           LGS  L1   PASS B    B         SHIFT OUT SIGN BIT OF FULL WORD
00304  001 137054                COV  ZERO S2             CLEAR OVFL,S2,& CNTR
00305  017 142255                RPT  PASS CNTR S2         AND SET RPTFF
00306  123 024502           DIV  L1   SUB  B    B         DIV(16X); A<=QUO(POS); B<=REM*2
00307  157 144142           LWF  L1   PASS L    S3        L <= FLG <= DVND SIGN(CM)
00310  010 027017                     CMPS S1   A         S1 <= QUO(CM)
00311  320 155071           JMP  CNDX ONES      QZERO     IF QUO=0,THEN NO FURTHER TESTING
00312  013 047057                     XOR  S2   S4        S2(15) <= EXPECTED SIGN OF QUO
00313  322 014671           JMP  CNDX AL15 RJS  *+2       JMP IF POSITIVE WAS EXPECTED
00314  000 040557                     INC  A    S1          ELSE  A <= QUO(2CM)
00315  017 142157                     PASS L    S2        L(15) <= EXPECTED SIGN OF QUO
00316  013 026757                     XOR       A         COMPARE TO FINAL SIGN OF QUO
00317  322 015071           JMP  CNDX AL15 RJS  *+2       JMP IF OK
00320  017 136753                SOV                         ELSE INDICATE OVERFLOW
00321  017 124504  QZERO         R1   PASS B    B         B <= (REM*2)/2
00322  324 040031           JMP  CNDX FLAG      FETCH     CHECK SGN OF DVND
00323  010 024517                     CMPS B    B         IF NEG,THEN FORM 2-COMP OF
00324  000 024536                RTN  INC  B    B         REM & STORE IN B
```

```
                            ORG                 330B
                   *************************************************************************
                   *      EAU TABLE
                   *************************************************************************
00330  320 007330  EAUTABLE JMP                 RRR
00331  320 007570           JMP                 ASR
00332  320 010030           JMP                 LSR
00333  320 000030           JMP                 FETCH     ILLEGAL IR CODE FOR EAU GROUP
00334  320 010270           JMP                 RRL
00335  320 010530           JMP                 ASL
00336  320 010770           JMP                 LSL
00337  320 012130           JMP                 MPY
                   *************************************************************************
                   *      MAC TABLE
                   *************************************************************************
00340  321 145270  MACTABLO JMP                 FADD      / FLOATING POINT
00341  321 145330           JMP                 FSUB      / FLOATING POINT
00342  321 151070           JMP                 FMPY      / FLOATING POINT
00343  321 153130           JMP                 FDIV      / FLOATING POINT
00344  321 140030           JMP                 IFIX      / FLOATING POINT
00345  321 141270           JMP                 FLOAT     / FLOATING POINT
00346  320 060030           JMP                 %1400     *** PROBABLE FUTURE HP USE
00347  320 060035           JMP  J30            %1400     *** PROBABLE FUTURE HP USE
00350  320 100030           JMP                 %2000     *** PROBABLE FUTURE HP USE
00351  320 100035           JMP  J30            %2000     *** PROBABLE FUTURE HP USE
00352  320 120030           JMP                 %2400     *** PROBABLE FUTURE HP USE
00353  320 120035           JMP  J30            %2400     *** PROBABLE FUTURE HP USE
00354  320 140030           JMP                 %3000     *** PROBABLE FUTURE HP USE
00355  320 140035           JMP  J30            %3000     *** PROBABLE FUTURE HP USE
00356  320 160030           JMP                 %3400     *** PROBABLE FUTURE HP USE
00357  320 160035           JMP  J30            %3400     *** PROBABLE FUTURE HP USE
                   *************************************************************************
```

```
00360   321 000030  MACTABL1 JMP                    %4000     *** PROBABLE FUTURE HP USE
00361   321 000035           JMP   J30              %4000     *** PROBABLE FUTURE HP USE
00362   321 020030           JMP                    %4400     *** PROBABLE FUTURE HP USE
00363   321 020035           JMP   J30              %4400     *** PROBABLE FUTURE HP USE
00364   321 040030           JMP                    %5000     *** PROBABLE FUTURE HP USE
00365   321 040035           JMP   J30              %5000     *** PROBABLE FUTURE HP USE
00366   321 060030           JMP                    %5400     *** PROBABLE FUTURE HP USE
00367   321 060035           JMP   J30              %5400     *** PROBABLE FUTURE HP USE
00370   321 100030           JMP                    %6000     +++RESERVED FOR CUSTOMER ONLY
00371   321 100035           JMP   J30              %6000     +++RESERVED FOR CUSTOMER ONLY
00372   321 120030           JMP                    %6400     +++RESERVED FOR CUSTOMER ONLY
00373   321 120035           JMP   J30              %6400     +++RESERVED FOR CUSTOMER ONLY
00374   320 040030           JMP                    %1000     / RESERVED FOR HP USE
00375   320 040035           JMP   J30              %1000     / RESERVED FOR HP USE
00376   321 160035           JMP   J30              %7400     / BASE SET EXTENSION
00377   321 161035           JMP   J30              %7420     / BASE SET EXTENSION
                    **********************************************************************
                             ORG                    400B
                    **********************************************************************
                    *
                    *        21MX MICRO-CODE
                    *        MODULE 1
                    *
                    **********************************************************************
                    DISPLAYA EQU                    %376
                    DISPLAYT EQU                    %367
                    DISPLAYS EQU                    %337
                    **********************************************************************
                    *        MEMORY INITIALIZATION ROUTINE
                    **********************************************************************
00400   322 161171  HALT     JMP   CNDX NMLS        MGOOD     JUMP IF MEMORY NOT LOST
00401   341 004617           IMM        HIGH MEU    %102      ENABLE SYSTEM MAP
00402   347 101017           IMM        LOW  S1     %340      S1 <= 2'S COMP OF 32
00403   001 137057                      ZERO S2               CLR S2 (MAP ADDR)
00404   017 142557                      PASS A      S2        CLR A-REG
00405   017 142517                      PASS B      S2        CLR B-REG
00406   017 142117                      PASS T      S2        CLR T REG
00407   353 077117           IMM        CMHI S3     %337      S3 <= "LOAD ADDR REG" COMMAND
00410   347 076264  LOSTLOOP IMM   SHLT LOW  CNTR   %337      CNTR<=COMP OF 32; CLEAR RUN FF
00411   017 144617                      PASS MEU    S3        LOAD 0 INTO ADDR REG ON MEU
00412   017 142620  MAPLOOP        MESP PASS MEU    S2        LOAD MAP IN MEU
00413   000 043063                 ICNT INC  S2     S2        INC MAP ADDR
00414   323 020531           JMP   CNDX CNT8 RJS    MAPLOOP   LOOP(*32)
00415   001 137717                      ZERO P                CLR P REG
00416   160 074717           WRTE       INC  PNM    P         M<=P; P<=P+1; WRITE ZERO DATA
00417   322 020731           JMP   CNDX AL15 RJS    *-1       LOOP UNTIL M=077777
00420   000 041017                      INC  S1     S1        INC MAP CNTR
00421   320 020431           JMP   CNDX TBZ  RJS    LOSTLOOP  LOOP (*32)
00422   341 000617           IMM        HIGH MEU    %100      DISABLE ALL MAPS NOW...
```

```
                    **********************************************************************
                    *        FRONT PANEL STANDARD SCAN ROUTINES
                    **********************************************************************
00423   334 165531  MGOOD    JMP   CNDX NSFP        CONTFP    JUMP IF   NON-STANDARD FRONT PANE
00424   017 115752                 FTCH PASS S      DSPL      S<=DISPLAY;INITIALIZE MEM. PROTEC
00425   330 121371           JMP   CNDX NSNG RJS    WAIT      JUMP IF "INSTR STEP" PRESSED
00426   347 156357           IMM        LOW  DSPI   DISPLAYT  ACTIVATE "T" INDICATOR IN DSPI
00427   300 024270  WAIT     JSB                    UPDATE    UPDATE DISPLAY WITH PROPER DATA
00430   334 021431           JMP   CNDX NSTB RJS    *         WAIT FOR BUTTON RELEASES
00431   325 164231           JMP   CNDX RUN         RUN
00432   334 061471           JMP   CNDX NSTB        *-1
00433   017 136757  SCAN     NOP                              SCAN FOR SWITCH PRESSED
                    *                                         NOP ONE CYCLE TO SET SWITCH CONDI
00434   332 122571           JMP   CNDX NLT  RJS    LEFT
00435   331 023431           JMP   CNDX NINC RJS    INC.M
00436   331 123531           JMP   CNDX NDEC RJS    DEC.M
00437   333 025471           JMP   CNDX NSTR RJS    STOREX
00440   333 121371           JMP   CNDX NRST RJS    WAIT
00441   332 032231  SCANRT   JMP   CNDX NRT  RJS    RIGHTR    JUMP IF "RIGHT" TO TEST FOR ENTRY
                    *                                         INTO SPECIAL DISPLAY ROUTINE.
```

```
00442  330 026171           JMP  CNDX NLDR RJS  LOADER
00443  325 164231           JMP  CNDX RUN       RUN
00444  330 161431           JMP  CNDX NSNG      WAIT+1    JMP IF "INSTR STEP" NOT PRESSED
00445  335 040271           JMP  CNDX INT       INTERUPT  SERVICE ANY PENDING INTERUPT
00446  017 176317                     PASS DSPL S         DISPLAY <= S
00447  220 074712           READ FTCH INC  PNM  P         DO STANDARD FETCH ROUTINE
00450  017 136745                ION
00451  017 100411                CLFL PASS IR   TAB
00452  220 020673           READ JTAB INC  CM   ADR
                   ************************************************************************
                   *      DISPLAY INDICATOR SHIFT ROUTINES
                   ************************************************************************
00453  017 117004  LEFT          R1   PASS S1   DSPI      S1<=DSPI SHIFTED RIGHT ONE
00454  321 122771           JMP  CNDX ALO  RJS  LEFTA     JUMP IF DSPI WRAP-AROUND REQUIRED
00455  017 140357  LEFTB              PASS DSPI S1        DSPI <= DSPI SHIFTED RIGHT ONE
00456  320 021370           JMP                 WAIT      JUMP TO STANDARD SCAN ROUTINES
00457  347 076357  LEFTA    IMM       LOW  DSPI DISPLAYS  DSPI WRAP-AROUND A TO S
00460  320 021370           JMP                 WAIT      JUMP TO STANDARD SCAN ROUTINES
00461  347 076157  RIGHT    IMM       LOW  L    DISPLAYS
00462  017 016750                STFL IOR       DSPI      SET FLAG; TEST DSPI
00463  320 123331           JMP  CNDX ONES RJS  RIGHTA    JUMP IF WRAP-AROUND OF DSPI REQD
00464  157 117002           LWF  L1   PASS S1   DSPI      S1<=DSPI SHIFTED LEFT ONE
00465  320 022670           JMP                 LEFTB
00466  347 174357  RIGHTA   IMM       LOW  DSPI DISPLAYA  DSPI WRAP-AROUND S TO A
00467  320 021370           JMP                 WAIT      JUMP TO STANDARD SCAN ROUTINES
                   ************************************************************************
                   *      INC M, DEC M ROUTINES
                   ************************************************************************
00470  000 023017  INC.M              INC  S1   M         S1 <= M + 1
00471  320 023570           JMP                 DEC.M+1
00472  007 123017  DEC.M              DEC  S1   M         S1 <= M - 1
00473  017 140457                     PASS M    S1        M <= S1
00474  320 021370           JMP  UNCD           WAIT      JUMP TO STANDARD SCAN ROUTINE
                   ************************************************************************
                   *      SPECIAL TEST TO EXIT SPECIAL DISPLAY LOOP
                   ************************************************************************
00475  017 116417  LEFTR              PASS IR   DSPI      CHECK FOR "M" DSPI
00476  327 122571           JMP  CNDX IR2  RJS  LEFT      JUMP IF "M" TO LEAVE SPECIAL CODE
00477  347 166357           IMM       LOW  DSPI %373      DSPI <= "M" (SHIFT FROM "T")
00500  017 142317                     PASS DSPL S2        SHOW POINTER ON DISPLAY
00501  320 033130           JMP  UNCD           WAITR     WAIT FOR BUTTON RELEASE IN SPECIA


                   ************************************************************************
                   *      STORE AND UPDATE ROUTINES
                   ************************************************************************
                   *                  THE REGISTER INDICATED IN DSPI IS THE BIT POSITION WH
                   *                  LOW. ALL OTHER BITS ARE 1.  THE ORDER (MSB TO LSB) IS
                   *                            S  P  T  M  B  A
                   *                  THE INDICATED REGISTER IS DETERMINED BY LOADING DSPI
                   *                  THE IR, AND JUMPING USING J30 TO GET TO THE APPROPRIA
                   *                  STORE OR UPDATE ROUTINE.  OTHER CODE IS INTERSPERSED
                   *                  FOR MAXIMUM CONTROL STORE EFFICIENCY
00502  017 116417  STORE              PASS IR   DSPI
00503  320 024035           JMP  J30            %0500     JMP TO STORE SELECTED REGISTER
00504  347 076357  RUN      IMM       LOW  DSPI DISPLAYS  DSPI <= "S".   THE SAVE REGISTER
                   *                                      ZERO AT THIS POINT SO THE NEXT RT
                   *                                      WILL INITIATE THE FETCH ROUTINE
                   ************************************************************************
00505  017 116417  UPDATE             PASS IR   DSPI
00506  320 025035           JMP  J30            %520      JMP TO DISPLAY SELECTED REGISTER
                   ************************************************************************
00507  177 114017           WRTE      PASS TAB  DSPL      STORE T
00510  000 023017                     INC  S1   M
00511  000 040457                     INC  M    S1        INCREMENT M, SET TAB LOGIC
00512  320 021430           JMP  UNCD           WAIT+1
00513  000 014476                RTN  INC  M    DSPL      STORE M
00514  017 115776  STORES        RTN  PASS S    DSPL      STORE S
00515  017 114536                RTN  PASS B    DSPL      STORE B
00516  017 114576                RTN  PASS A    DSPL      STORE A
00517  347 136157           IMM       LOW  L    %357      P OR S TO BE DISPLAYED
00520  017 016757                     IOR       DSPI      MASK OUT "S"
00521  320 164631           JMP  CNDX ONES      STORES    JUMP IF "S" INDICATED
00522  017 115736                RTN  PASS P    DSPL      STORE P
```

```
                   *************************************************************************
                   ****** OVFL REG. STORE--PART OF SPECIAL DISPLAY ROUTINES ***************
00523  017 115013  STOROO        SOV  PASS S1   DSPL      CHECK DISPLAY
00524  321 165331           JMP  CNDX ALO       *+2
00525  017 136754                COV                      CLEAR OVERFLOW
00526  017 136776                RTN
                   *************************************************************************
00527  220 022457           READ      INC  M    M         UPDATE T, READ M, SET TAB LOGIC
00530  017 100336                RTN  PASS DSPL TAB       DSPL <= MEM DATA
                   *************************************************************************
00531  300 024130  STOREX   JSB                      STORE     STORE ROUTINES END WITH RTN
00532  320 021370  CONTFP   JMP                      WAIT      JUMP TO STANDARD SCAN ROUTINES
                   *************************************************************************
00533  017 122336                RTN  PASS DSPL M         UPDATE M
00534  017 176336  UPDATES       RTN  PASS DSPL S         UPDATE S
00535  017 124336                RTN  PASS DSPL B         UPDATE B
00536  017 126336                RTN  PASS DSPL A         UPDATE A
00537  347 136157           IMM       LOW  L    357B      P OR S INDICATED
00540  017 016757                     IOR       DSPI      MASK OUT "S"
00541  320 165631           JMP  CNDX ONES      UPDATES
00542  017 174336                RTN  PASS DSPL P         UPDATE P
```

```
                   *************************************************************************
                   *      21MX ROM BOOTSTRAP MEMORY LOADER ROUTINE
                   *************************************************************************
00543  341 177053  LOADER   IMM  SOV  HIGH S2   %177      FORM 0111111111111111 (MAX ADDR)
00544  353 137017           IMM       CMHI S1   %357      FORM 0001000000000000 (10K) IN S1
                   ****** DETERMINE MEMORY SIZE, STARTING ADDR FOR LOADER ****************
00545  347 000157  SIZE     IMM       LOW  L    %300      FORM 1111111111000000 IN L
00546  015 143717                     AND  P    S2        FORM STARTING ADDR IN P
00547  010 075217                     CMPS S5   P         FORM TWO'S COMP
00550  000 051217                     INC  S5   S5        OF SA IN S5
00551  017 142457                     PASS M    S2        PUT LAST ADDR INTO M
00552  320 161371           JMP  CNDX ONES      WAIT      TEST FOR NO READ/WRTE CAPABILITY
00553  177 150117           WRTE      PASS T    S5        PASS INTO T
00554  017 140157                     PASS L    S1        UPDATE LAST ADDR WHILE WAITING
00555  223 043057           READ      SUB  S2   S2        TO RETRIEVE DATA
00556  017 150157                     PASS L    S5        COMPARE WHAT WAS READ FROM MEM.
00557  013 004757                     XOR       T         TO DATA WRITTEN (S3)
00560  320 026271           JMP  CNDX TBZ  RJS  SIZE      IF IT CHECKS, WE HAVE CORRECT STR
                   ****** CHECK SELECT CODE IN  S REG.    ********************************
00561  347 000157           IMM       LOW  L    %300      FORM 1111111111000000 IN L
00562  347 164257           IMM       LOW  CNTR %372      CNTR GETS -6
00563  017 176417                     PASS IR   S         SET UP LOADER SELECT BIT
00564  017 177155                RPT  PASS S4   S         SET UP S-REG FOR SHIFT
00565  017 147144                R1   PASS S4   S4        SHIFT SELECT CODE INTO BITS(0-5)
00566  013 147157                     SANL S4   S4        MASK OFF SEL. CODE
00567  347 160157           IMM       LOW  L    %370      FORM 1111111111111000 (=-10B) IN
00570  004 147153                SOV  ADD  S4   S4        SUB 10B FROM SEL CODE; SAVE IN SJ
00571  322 061371           JMP  CNDX AL15      WAIT      IF NEG RESULT, SCB < 10B; RTN W/
                   ****** PREPARE FOR LOADER TRANSFER ************************************
00572  344 000257           IMM       LOW  CNTR %0        CLEAR CNTR (ROM ADDR REG)
00573  017 174454                COV  PASS M    P         PUT SA IN M;CLR OVF = NO OPER ERR
                   ****** TRANSFER CONTENTS OF LOADER ROM TO MEMORY **********************
00574  017 131003  LOOP1         L4   PASS S1   LDR       PASS XXXXXXXXAAAAXXXX INTO S1;CNT
00575  017 140163                ICNT PASS L    S1        CNTR=X01
00576  015 131003                L4   AND  S1   LDR       FORM XXXXAAAABBBBXXXX IN S1;CNTR=
00577  017 140163                ICNT PASS L    S1        CNTR=X10
00600  015 131003                L4   AND  S1   LDR       FORM AAAABBBBCCCCXXXX IN S1;CNTR=
00601  017 140163                ICNT PASS L    S1        CNTR=X11
00602  012 031017                     NAND S1   LDR       FORM AAAABBBBCCCCDDDD (CMPL FORM)
00603  177 140117           WRTE      PASS T    S1        WRITE INTO MEMORY
00604  000 023063                ICNT INC  S2   M         UPDATE MEM ADDR;CNTR=X00
00605  017 142457                     PASS M    S2        PASS NEW ADDR INTO M
00606  344 000157           IMM       LOW  L    %0        FORM 1111111100000000 IN L
00607  017 022757                     IOR       M         MASK M TO SEE IF LAST WORD OF LDR
00610  320 127631           JMP  CNDX ONES RJS  LOOP1     IF M(0-8)=11111111, DON'TLOOP
```

```
                          ***********************************************************************
00611   347 000257                  IMM         LOW  CNTR %300        SET UP COUNT TO FIND LAST WORD
00612   344 077110                  IMM    STFL LOW  S3   %037
00613   157 145102                  LWF    L1   PASS S3   S3          FORM 1111111000111111 IN S3
00614   017 175017                              PASS S1   P           PASS SA INTO S1
                          ****** CHECK INSTRUCTION IN MEMORY FOR I/O TYPE **********************
00615   237 140457        NUWRD     READ        PASS M    S1          PASS SA INTO M & READ FIRST INSTR
00616   340 026157                  IMM         HIGH L    %013        FORM COMP OF 1111010000000000 IN
00617   017 105057                              PASS S2   T           SAVE WORD IN S2
00620   013 143017                              SANL S1   S2          MASK UPPER BITS FOR I/O TYPE
00621   341 166157                  IMM         HIGH L    %173        FORM 0111101111111111 IN L
00622   013 040757                              XOR       S1          NOW CHECK FOR I/O TYPE
00623   320 171531                  JMP    CNDX ONES      HTST        IF MATCH OCCURS, JUMP OUT OF LOOP
                          ***********************************************************************
00624   000 023023        UPDT             ICNT INC  S1   M           OTHERWISE UPDATE M IN S1
00625   323 030671                  JMP    CNDX CNT8 RJS  NUWRD       LOOP BACK
00626   017 146154                         COV  PASS L    S4          PASS (SCB-10B) INTO L
00627   004 143051                         CLFL ADD  S2   S2          CHNG SC OF DCPC CNTRL WORD
00630   177 142117                  WRTE        PASS T    S2          SAVE IN MEM
00631   320 021370                  JMP                   WAIT        RETURN TO SCAN ROUTINE
                          ****** UPDATE SELECT CODE IN I/O INSTRUCTION *************************
00632   017 144157        HTST                  PASS L    S3          PASS 1111111000111111 INTO L
00633   014 042757                              NSOL      S2          BLEND TO CHECK FOR...000...OF HLT
00634   320 171231                  JMP    CNDX ONES      UPDT        IF FOUND GET NEXT INSTR
00635   347 016157                  IMM         LOW  L    %307        FORM 1111111111000111 IN L
00636   013 142757                              SANL      S2          MASK BITS TO CHECK FOR SC < 10B
00637   320 071231                  JMP    CNDX TBZ       UPDT        IF SO, RTN TO LOOP
00640   017 146157                              PASS L    S4          PASS (SCB-10B) INTO L
00641   004 143057                              ADD  S2   S2          ADD TO SC FROM INSTR
00642   177 142117                  WRTE        PASS T    S2          PASS INTO T AND WRITE INTO MEMORY
00643   320 031230                  JMP                   UPDT        RTN TO LOOP
```

```
                          ***********************************************************************
                          *     SPECIAL DISPLAY ROUTINES
                          ***********************************************************************
00644   017 116417        RIGHTR                PASS IR   DSPI        "RIGHT" PRESSED: IR <= DSPI
00645   327 163071                  JMP    CNDX IR2       RIGHT       JUMP IF M NOT SELECTED BY DSPI
00646   017 115057                              PASS S2   DSPL        S2 <= DSPL (POINTER)
00647   322 023071                  JMP    CNDX AL15 RJS  RIGHT       JMP IF DSPL BIT 15 WASNT SET
00650   347 156357                  IMM         LOW  DSPI %367        DSPI <= "T"
                          ***********************************************************************
00651   006 042157        UPDATR                OP9  L    S2          CHECK DSPL BIT 14, STORE S2 IN L
00652   322 074671                  JMP    CNDX AL15      MEUMAPS     JUMP IF S2 BIT 14 = 1 TO UPDATE M
00653   350 007003                  IMM    L4   CMHI S1   %003        S1 <= MASK FOR REGISTERS = 140017
00654   015 141057                              AND  S2   S1          S2 <= S2 MASK OUT UNUSED BITS
00655   017 142417                              PASS IR   S2          SET REGISTER SELECTION
00656   333 073071                  JMP    CNDX NSTR      READREG     JUMP IF STORE BUTTON NOT PRESSED
00657   300 037035                  JSB    J30            STOREG      SELECTED REGISTER <= DISPLAY
00660   320 033130                  JMP    UNCD           WAITR       WAIT FOR NEXT BUTTON
00661   300 036035        READREG   JSB    J30            DSPLREG     DISPLAY <= SELECTED REGISTER
                          ***********************************************************************
00662   334 033131        WAITR     JMP    CNDX NSTB RJS  *           WAIT FOR BUTTON RELEASE
00663   325 164231                  JMP    CNDX RUN       RUN         JUMP IF RUN INDICATOR LIT
00664   334 073171                  JMP    CNDX NSTB      *-1         JUMP BACK IF NO BUTTON PRESSED
                          *
00665   017 136757                  NOP                               WAIT ONE CYCLE FOR SETTING SWITCH
00666   332 123671                  JMP    CNDX NLT  RJS  LEFTR       JUMP IF "LEFT" PRESSED
00667   331 073531                  JMP    CNDX NINC      NOTINC      JUMP IF "INCM" NOT PRESSED
00670   000 043057                              INC  S2   S2          INCREMENT POINTER
00671   320 034030                  JMP    UNCD           DECMR+1
00672   331 174231        NOTINC    JMP    CNDX NDEC      NOTDEC
00673   340 000157                  IMM         HIGH L    %000        CHECK FOR
00674   015 142757                              AND       S2          DECREMENT OF
00675   320 033771                  JMP    CNDX TBZ  RJS  DECMR       ZERO COUNT
00676   000 143057                              OP1  S2   S2          S2 OR L PLUS 1 (WRAP AROUND COUNT
00677   007 143057        DECMR                 DEC  S2   S2          DECREMENT POINTER
00700   017 116417                              PASS IR   DSPI        IR <= DSPI
00701   327 172471                  JMP    CNDX IR2       UPDATR      JUMP IF M NOT INDICATED
00702   017 142317                              PASS DSPL S2          UPDATE DISPLAY
```

```
00703  320 033130           JMP  UNCD           WAITR     WITH NEW POINTER VALUE AND JUMP
00704  333 134031  NOTDEC   JMP  CNDX NRST RJS  DECMR+1   JUMP IF "DISPLAY" PRESSED
00705  333 074471           JMP  CNDX NSTR      *+4       JUMP IF STORE NOT PRESSED
00706  017 116417                     PASS IR   DSPI
00707  327 125471           JMP  CNDX IR2  RJS  STOREX    JUMP IF M SELECTED. LEAVE SPECIAL
00710  320 032470           JMP  UNCD           UPDATR    M NOT SELECTED
00711  320 022070           JMP  UNCD           SCANRT    JUMP TO STD ROUTINES
                   ************************************************************************
00712  347 172417  STOREE   IMM       LOW  IR   %375      SET UP SRG TYPE ER* SHIFT
00713  017 114741                SRG2 PASS      DSPL      SET E ACCORDING TO DSPL BIT 0
00714  017 136776                RTN
                   ************************************************************************
                   ****** MEU MAP MANIPULATIONS *******************************************
00715  346 000157  MEUMAPS  IMM       LOW  L    %200      S1 <= MASK OF LOW 7 BITS
00716  013 143017                     SANL S1   S2
00717  340 176157           IMM       HIGH L    %077      L <= 037777B
00720  016 141057                     SONL S2   S1        S2 <= MASK OUT BITS 13 TO 8
00721  343 076157           IMM       HIGH L    %337      OR IN BIT 13
00722  016 141017                     SONL S1   S1
00723  017 140617                     PASS MEU  S1        SEND MAP NO. TO MEU
00724  333 075371           JMP  CNDX NSTR      READMAP   JUMP IF STORE NOT PRESSED
00725  017 114620                MESP PASS MEU  DSPL      MEU MAP <= DISPLAY
00726  320 033130           JMP  UNCD           WAITR
00727  017 134320  READMAP       MESP PASS DSPL MEU       DISPLAY <= MEU MAP
00730  320 033130           JMP  UNCD           WAITR
                   ****** SIMULATED LIA 4 I/O INSTRUCTION TO READ CIR ********************
00731  344 010417  DSPLCIR  IMM       LOW  IR   %004      SET UP SEL CODE 4 IN IR
00732  017 136762                IOG                      INITIATE I/O CYCLE AT TIME T2
00733  017 136757           NOP                           WAIT FOR TIME T4
00734  017 110336                RTN  PASS DSPL IOI       CIR TO DISPLAY. DONT ISSUE IAK
```

```
                            ORG                 740B
                   ************************************************************************
                   *      SHORT SUBROUTINES TO STORE/DISPLAY SELECTED REGISTERS
                   ************************************************************************
00740  017 170336  DSPLREG       RTN  PASS DSPL X         PASS REG TO FRONT PANEL AND RETUR
00741  017 172336                RTN  PASS DSPL Y
00742  017 112336                RTN  PASS DSPL CNTR
00743  017 144336                RTN  PASS DSPL S3
00744  017 146336                RTN  PASS DSPL S4
00745  017 150336                RTN  PASS DSPL S5
00746  017 152336                RTN  PASS DSPL S6
00747  017 154336                RTN  PASS DSPL S7
00750  017 156336                RTN  PASS DSPL S8
00751  017 160336                RTN  PASS DSPL S9
00752  017 162336                RTN  PASS DSPL S10
00753  017 164336                RTN  PASS DSPL S11
00754  017 166336                RTN  PASS DSPL S12
00755  320 035470           JMP  UNCD           DSPLCIR
00756  343 176336           IMM  RTN  HIGH DSPL 377B
00757  343 176336           IMM  RTN  HIGH DSPL 377B
                   ************************************************************************
00760  017 115636  STOREG        RTN  PASS X    DSPL      STORE INTO REG FROM FRONT PANEL
00761  017 115676                RTN  PASS Y    DSPL
00762  017 114276                RTN  PASS CNTR DSPL
00763  017 115136                RTN  PASS S3   DSPL
00764  017 115176                RTN  PASS S4   DSPL
00765  017 115236                RTN  PASS S5   DSPL
00766  017 115276                RTN  PASS S6   DSPL
00767  017 115336                RTN  PASS S7   DSPL
00770  017 115376                RTN  PASS S8   DSPL
00771  017 115436                RTN  PASS S9   DSPL
00772  017 115476                RTN  PASS S10  DSPL
00773  017 115536                RTN  PASS S11  DSPL
00774  017 115576                RTN  PASS S12  DSPL
00775  017 106336                RTN  PASS DSPL CIR       LOAD CIR FROM INT. REQUEST LINES
                   *                                      AND ISSUE INTERRUPT ACKNOWLEDGE
00776  320 025170           JMP  UNCD           STOR00
00777  320 034530           JMP  UNCD           STOREE
```

```
                          ************************************************************************
                          *       LOOK UP TABLE USED BY THE JTAB SPECIAL
                          ************************************************************************
01000   000 000053                  DEF                  SRG            1
01001   000 000053                  DEF                  SRG            2
01002   000 000053                  DEF                  SRG            3
01003   000 000024                  DEF                  ASGNOP         4
01004   000 000031                  DEF                  ASGCL*         5
01005   000 000036                  DEF                  ASGCM*         6
01006   000 000043                  DEF                  ASGCC*         7
01007   000 000054                  DEF                  SRG+1         10
01010   000 000053                  DEF                  SRG           11
01011   000 000053                  DEF                  SRG           12
01012   000 000053                  DEF                  SRG           13
01013   000 000024                  DEF                  ASGNOP        14
01014   000 000031                  DEF                  ASGCL*        15
01015   000 000036                  DEF                  ASGCM*        16
01016   000 000043                  DEF                  ASGCC*        17
01017   000 000106                  DEF                  AND           20
01020   000 000106                  DEF                  AND           21
01021   000 000106                  DEF                  AND           22
01022   000 000106                  DEF                  AND           23
01023   000 000106                  DEF                  AND           24
01024   000 000106                  DEF                  AND           25
01025   000 000106                  DEF                  AND           26
01026   000 000106                  DEF                  AND           27
01027   000 000132                  DEF                  JSB           30
01030   000 000132                  DEF                  JSB           31
01031   000 000132                  DEF                  JSB           32
01032   000 000132                  DEF                  JSB           33
01033   000 000132                  DEF                  JSB           34
01034   000 000132                  DEF                  JSB           35
01035   000 000132                  DEF                  JSB           36
01036   000 000132                  DEF                  JSB           37
01037   000 000116                  DEF                  XOR           40
01040   000 000116                  DEF                  XOR           41
01041   000 000116                  DEF                  XOR           42
01042   000 000116                  DEF                  XOR           43
01043   000 000116                  DEF                  XOR           44
01044   000 000116                  DEF                  XOR           45
01045   000 000116                  DEF                  XOR           46
01046   000 000116                  DEF                  XOR           47
01047   000 000164                  DEF                  JMP           50
01050   000 000164                  DEF                  JMP           51
01051   000 000164                  DEF                  JMP           52
01052   000 000164                  DEF                  JMP           53
01053   000 000164                  DEF                  JMP           54
01054   000 000164                  DEF                  JMP           55
01055   000 000164                  DEF                  JMP           56
01056   000 000164                  DEF                  JMP           57
01057   000 000121                  DEF                  IOR           60
01060   000 000121                  DEF                  IOR           61
01061   000 000121                  DEF                  IOR           62
01062   000 000121                  DEF                  IOR           63
01063   000 000121                  DEF                  IOR           64
01064   000 000121                  DEF                  IOR           65
01065   000 000121                  DEF                  IOR           66
01066   000 000121                  DEF                  IOR           67
01067   000 000136                  DEF                  ISZ           70
01070   000 000136                  DEF                  ISZ           71
01071   000 000136                  DEF                  ISZ           72
01072   000 000136                  DEF                  ISZ           73
01073   000 000136                  DEF                  ISZ           74
01074   000 000136                  DEF                  ISZ           75
01075   000 000136                  DEF                  ISZ           76
01076   000 000136                  DEF                  ISZ           77
01077   000 000127                  DEF                  AD*          100
01100   000 000127                  DEF                  AD*          101
01101   000 000127                  DEF                  AD*          102
01102   000 000127                  DEF                  AD*          103
01103   000 000127                  DEF                  AD*          104
01104   000 000127                  DEF                  AD*          105
01105   000 000127                  DEF                  AD*          106
01106   000 000127                  DEF                  AD*          107
01107   000 000127                  DEF                  AD*          110
01110   000 000127                  DEF                  AD*          111
```

```
0753 01111  000 000127          DEF                 AD*         112
0754 01112  000 000127          DEF                 AD*         113
0755 01113  000 000127          DEF                 AD*         114
0756 01114  000 000127          DEF                 AD*         115
0757 01115  000 000127          DEF                 AD*         116
0758 01116  000 000127          DEF                 AD*         117
0759 01117  000 000111          DEF                 CP*         120
0760 01120  000 000111          DEF                 CP*         121
0761 01121  000 000111          DEF                 CP*         122
0762 01122  000 000111          DEF                 CP*         123
0763 01123  000 000111          DEF                 CP*         124
0764 01124  000 000111          DEF                 CP*         125
0765 01125  000 000111          DEF                 CP*         126
0766 01126  000 000111          DEF                 CP*         127
0767 01127  000 000111          DEF                 CP*         130
0768 01130  000 000111          DEF                 CP*         131
0769 01131  000 000111          DEF                 CP*         132
0770 01132  000 000111          DEF                 CP*         133
0771 01133  000 000111          DEF                 CP*         134
0772 01134  000 000111          DEF                 CP*         135
0773 01135  000 000111          DEF                 CP*         136
0774 01136  000 000111          DEF                 CP*         137
0775 01137  000 000144          DEF                 LD*         140
0776 01140  000 000144          DEF                 LD*         141
0777 01141  000 000144          DEF                 LD*         142
0778 01142  000 000144          DEF                 LD*         143
0779 01143  000 000144          DEF                 LD*         144
0780 01144  000 000144          DEF                 LD*         145
0782 01145  000 000144          DEF                 LD*         146
0783 01146  000 000144          DEF                 LD*         147
0784 01147  000 000144          DEF                 LD*         150
0785 01150  000 000144          DEF                 LD*         151
0786 01151  000 000144          DEF                 LD*         152
0787 01152  000 000144          DEF                 LD*         153
0788 01153  000 000144          DEF                 LD*         154
0789 01154  000 000144          DEF                 LD*         155
0790 01155  000 000144          DEF                 LD*         156
0791 01156  000 000144          DEF                 LD*         157
0792 01157  000 000124          DEF                 ST*         160
0793 01160  000 000124          DEF                 ST*         161
0794 01161  000 000124          DEF                 ST*         162
0795 01162  000 000124          DEF                 ST*         163
0796 01163  000 000124          DEF                 ST*         164
0797 01164  000 000124          DEF                 ST*         165
0798 01165  000 000124          DEF                 ST*         166
0799 01166  000 000124          DEF                 ST*         167
0800 01167  000 000124          DEF                 ST*         170
0801 01170  000 000124          DEF                 ST*         171
0802 01171  000 000124          DEF                 ST*         172
0803 01172  000 000124          DEF                 ST*         173
0804 01173  000 000124          DEF                 ST*         174
0805 01174  000 000124          DEF                 ST*         175
0806 01175  000 000124          DEF                 ST*         176
0807 01176  000 000124          DEF                 ST*         177
0808 01177  000 000102          DEF                 EAU         200 MPY,ASL,LSL,RRL
0809 01200  000 000262          DEF                 DIV         201
0810 01201  000 000102          DEF                 EAU         202 ASR,LSR,RRR
0811 01202  000 000104          DEF                 MAC1        203
0812 01203  000 000101          DEF                 IOG         204 HLT,STF,SFC,SFS
0813 01204  000 000101          DEF                 IOG         205 MIA,LIA,OTA,STC
0814 01205  000 000101          DEF                 IOG         206 HLT,CLF
0815 01206  000 000101          DEF                 IOG         207 MIA,LIA,OTA,STC
0816 01207  000 000224          DEF                 DLD         210
0817 01210  000 000233          DEF                 DST         211
0818 01211  000 000103          DEF                 MAC0        212
0819 01212  000 000104          DEF                 MAC1        213
0820 01213  000 000101          DEF                 IOG         214 HLT,STF,SFC,SFS
0821 01214  000 000101          DEF                 IOG         215 MIB,LIB,OTB,CLC
0822 01215  000 000101          DEF                 IOG         216 HLT,CLF
0823 01216  000 000101          DEF                 IOG         217 MIB,LIB,OTB,CLC
0824 01217  000 000105          DEF                 AND,I       220
0825 01220  000 000105          DEF                 AND,I       221
0826 01221  000 000105          DEF                 AND,I       222
0827 01222  000 000105          DEF                 AND,I       223
0828 01223  000 000105          DEF                 AND,I       224
0829 01224  000 000105          DEF                 AND,I       225
```

```
0830 01225  000 000105          DEF                      AND,I      226
0831 01226  000 000105          DEF                      AND,I      227
0833 01227  000 000131          DEF                      JSB,I      230
0834 01230  000 000131          DEF                      JSB,I      231
0835 01231  000 000131          DEF                      JSB,I      232
0836 01232  000 000131          DEF                      JSB,I      233
0837 01233  000 000131          DEF                      JSB,I      234
0838 01234  000 000131          DEF                      JSB,I      235
0839 01235  000 000131          DEF                      JSB,I      236
0840 01236  000 000131          DEF                      JSB,I      237
0841 01237  000 000115          DEF                      XOR,I      240
0842 01240  000 000115          DEF                      XOR,I      241
0843 01241  000 000115          DEF                      XOR,I      242
0844 01242  000 000115          DEF                      XOR,I      243
0845 01243  000 000115          DEF                      XOR,I      244
0846 01244  000 000115          DEF                      XOR,I      245
0847 01245  000 000115          DEF                      XOR,I      246
0848 01246  000 000115          DEF                      XOR,I      247
0849 01247  000 000145          DEF                      JMP,I      250
0850 01250  000 000145          DEF                      JMP,I      251
0851 01251  000 000145          DEF                      JMP,I      252
0852 01252  000 000145          DEF                      JMP,I      253
0853 01253  000 000145          DEF                      JMP,I      254
0854 01254  000 000145          DEF                      JMP,I      255
0855 01255  000 000145          DEF                      JMP,I      256
0856 01256  000 000145          DEF                      JMP,I      257
0857 01257  000 000120          DEF                      IOR,I      260
0858 01260  000 000120          DEF                      IOR,I      261
0859 01261  000 000120          DEF                      IOR,I      262
0860 01262  000 000120          DEF                      IOR,I      263
0861 01263  000 000120          DEF                      IOR,I      264
0862 01264  000 000120          DEF                      IOR,I      265
0863 01265  000 000120          DEF                      IOR,I      266
0864 01266  000 000120          DEF                      IOR,I      267
0865 01267  000 000135          DEF                      ISZ,I      270
0866 01270  000 000135          DEF                      ISZ,I      271
0867 01271  000 000135          DEF                      ISZ,I      272
0868 01272  000 000135          DEF                      ISZ,I      273
0869 01273  000 000135          DEF                      ISZ,I      274
0870 01274  000 000135          DEF                      ISZ,I      275
0871 01275  000 000135          DEF                      ISZ,I      276
0872 01276  000 000135          DEF                      ISZ,I      277
0873 01277  000 000126          DEF                      AD*,I      300
0874 01300  000 000126          DEF                      AD*,I      301
0875 01301  000 000126          DEF                      AD*,I      302
0876 01302  000 000126          DEF                      AD*,I      303
0877 01303  000 000126          DEF                      AD*,I      304
0878 01304  000 000126          DEF                      AD*,I      305
0879 01305  000 000126          DEF                      AD*,I      306
0880 01306  000 000126          DEF                      AD*,I      307
0881 01307  000 000126          DEF                      AD*,I      310
0882 01310  000 000126          DEF                      AD*,I      311
0884 01311  000 000126          DEF                      AD*,I      312
0885 01312  000 000126          DEF                      AD*,I      313
0886 01313  000 000126          DEF                      AD*,I      314
0887 01314  000 000126          DEF                      AD*,I      315
0888 01315  000 000126          DEF                      AD*,I      316
0889 01316  000 000126          DEF                      AD*,I      317
0890 01317  000 000110          DEF                      CP*,I      320
0891 01320  000 000110          DEF                      CP*,I      321
0892 01321  000 000110          DEF                      CP*,I      322
0893 01322  000 000110          DEF                      CP*,I      323
0894 01323  000 000110          DEF                      CP*,I      324
0895 01324  000 000110          DEF                      CP*,I      325
0896 01325  000 000110          DEF                      CP*,I      326
0897 01326  000 000110          DEF                      CP*,I      327
0898 01327  000 000110          DEF                      CP*,I      330
0899 01330  000 000110          DEF                      CP*,I      331
0900 01331  000 000110          DEF                      CP*,I      332
0901 01332  000 000110          DEF                      CP*,I      333
0902 01333  000 000110          DEF                      CP*,I      334
0903 01334  000 000110          DEF                      CP*,I      335
0904 01335  000 000110          DEF                      CP*,I      336
0905 01336  000 000110          DEF                      CP*,I      337
0906 01337  000 000143          DEF                      LD*,I      340
0907 01340  000 000143          DEF                      LD*,I      341
```

```
01341  000 000143          DEF           LD*,I      342
01342  000 000143          DEF           LD*,I      343
01343  000 000143          DEF           LD*,I      344
01344  000 000143          DEF           LD*,I      345
01345  000 000143          DEF           LD*,I      346
01346  000 000143          DEF           LD*,I      347
01347  000 000143          DEF           LD*,I      350
01350  000 000143          DEF           LD*,I      351
01351  000 000143          DEF           LD*,I      352
01352  000 000143          DEF           LD*,I      353
01353  000 000143          DEF           LD*,I      354
01354  000 000143          DEF           LD*,I      355
01355  000 000143          DEF           LD*,I      356
01356  000 000143          DEF           LD*,I      357
01357  000 000123          DEF           ST*,I      360
01360  000 000123          DEF           ST*,I      361
01361  000 000123          DEF           ST*,I      362
01362  000 000123          DEF           ST*,I      363
01363  000 000123          DEF           ST*,I      364
01364  000 000123          DEF           ST*,I      365
01365  000 000123          DEF           ST*,I      366
01366  000 000123          DEF           ST*,I      367
01367  000 000123          DEF           ST*,I      370
01370  000 000123          DEF           ST*,I      371
01371  000 000123          DEF           ST*,I      372
01372  000 000123          DEF           ST*,I      373
01373  000 000123          DEF           ST*,I      374
01374  000 000123          DEF           ST*,I      375
01375  000 000123          DEF           ST*,I      376
01376  000 000123          DEF           ST*,I      377
                           END
```

```
               ORG               1000B
*
*
*
*
*
*
***********************************************************************
*                                                     *
*       MEMORY EXPANSION UNIT MACRO INSTRUCTIONS       *
*       ------ --------- ---- ----- ------------       *
*                                        J.S.ELWARD    *
*  REV  .01 APR 1975                                   *
*******************************************************
*
INDLEVEL EQU                %013
*   A JMP TO INDLEVEL WILL ADD 5 CYCLES + 4 PER ADDITIONAL LEVEL
*
*******************************************************
*
*   REGISTER ASSIGNMENTS
*
*     S3 :: P-REGISTER
*     S4 :: MEM CONTROL WORD; MEM ADDRESS REGISTER
*     S5 :: WORDS AND MAP DATA IN LOOP EXECUTION; MASKS AND CONSTANTS
*     S6 :: GENERAL PURPOSE SCRATCH
*
***********************************************************************
```

```
                        *
                        *
                        *
                        *
                        *
                        *****************************************************************
                        *      ENTRY JUMP TABLE...WE HAVE COMPLETED SIX CYCLES AT ENTRY
                        *****************************************************************
                        *                                                         << LENGTH >>
                        *      MACRO JUMP POINT AND MNEMONIC      BINARY CODE      OCTAL DECIMAL
                        *
01000  320 041035  MEUMACRO JMP  J30            JTABL    1000X01111100XXXX   40   32
01001  320 042010           JMP  STFL           XNM      1000X01111010001    10   08
01002  320 045270           JMP                 XM*      1000X01111010010    16   14
01003  017 136776                RTN                     1000X01111010011    00   00
01004  320 046170           JMP                 XL*      1000X01111010100    05   05
01005  320 046430           JMP                 XS*      1000X01111010101    05   05
01006  320 046670           JMP                 XC*      1000X01111010110    07   07
01007  320 047230           JMP                 LF*      1000X01111010111    04   04
01010  017 134617  RS*                PASS MEU  MEU      1000X01111011000    00   00
01011  017 134076  RV*           RTN  PASS CAB  MEU      1000X01111011001    00   00
01012  320 047430           JMP                 DJP      1000X01111011010    02   02
01013  320 050070           JMP                 DJS      1000X01111011011    02   02
01014  320 047530           JMP                 SJP      1000X01111011100    02   02
01015  320 050170           JMP                 SJS      1000X01111011101    02   02
01016  320 047630           JMP                 UJP      1000X01111011110    05   05
01017  320 050270           JMP                 UJS      1000X01111011111    07   07
01020  320 042030  JTABL    JMP                 XMM      1000X011110X0000    55   45
01021  017 136776                RTN                     1000X01111000001    00   00
01022  320 050670           JMP                 MBI      1000X01111000010    16   14
01023  320 050630           JMP                 MBF      1000X01111000011    01   01
01024  320 051560           JMP  MESP           MBW      1000X01111000100    30   24
01025  320 053170           JMP                 MWI      1000X01111000101    15   13
01026  320 053160           JMP  MESP           MWI      1000X01111000110    00   00
01027  320 054020           JMP  MESP           MWW      1000X01111000111    15   13
01030  320 054670           JMP                 SY*      1000X01111001000    02   02
01031  320 055230           JMP                 US*      1000X01111001001    24   20
01032  320 054770           JMP                 PA*      1000X01111001010    03   03
01033  320 055130           JMP                 PB*      1000X01111001011    02   02
01034  320 056470           JMP                 SSM      1000X01111001100    05   05
01035  320 056730           JMP                 JRS      1000X01111001101    15   13
01036  017 136776                RTN                     1000X01111001110    00   00
01037  017 136776                RTN                     1000X01111001111    00   00
                        *       JSB                 OPGET    *** UTILITY ****    05   05
                        *****************************************************************
                        *
                        *
                        *
                        *
                        *
                        *****************************************************************
01040  017 175117  XMM                PASS S3   P        S3 <= P; SAVE P
01041  010 071217                     CMPS S5   X        S5 <= ONE'S COMP OF COUNT
01042  320 146131           JMP  CNDX ONES      RTN*     TEST FOR ZERO COUNT
01043  346 000157           IMM       LOW  L    %200     L  <= 1111111110000000
01044  013 127157                     SANL S4   A        MASK LOW 7 BITS OF A-REG
01045  343 076157           IMM       HIGH L    %337     L  <= 1101111111111111
01046  016 147157                     SONL S4   S4       ADD CONTROL BIT (13)
01047  017 146617                     PASS MEU  S4       MEM ADDR REG <= S4
01050  017 125717                     PASS P    B        P  <= B(TABLE ADDRESS)
01051  017 150257                     PASS CNTR S5       CNTR <= S5
01052  324 043671           JMP  CNDX FLAG      XMS      TEST FOR XMS INSTRUCTION
01053  322 004271           JMP  CNDX AL15 RJS  READMAP  TEST FOR NEGATIVE COUNT
01054  220 074717           READ      INC  PNM  P        READ FIRST WORD; P <= P+1
                        *
01055  017 101217  MELOOP1            PASS S5   TAB      S5 <= MAP DATA
01056  017 150620                MESP PASS MEU  S5       MAP REG <= DATA
01057  220 074723           READ ICNT INC  PNM  P        READ NEXT WORD; INC CNTR AND P
01060  337 002671           JMP  CNDX CNT4 RJS  MELOOP1  LOOP FOR 16X
01061  323 043271           JMP  CNDX CNT8      XMM.RTN  IS TOTAL LOOP FINSIHED
01062  237 136757           READ                         RESTART THE READ AT SAME ADDR
01063  335 002671           JMP  CNDX INT  RJS  MELOOP1  TEST FOR NO INTERRUPT
01064  007 145117                     DEC  S3   S3       RESET P REGISTER FOR RESTART
                        *       JMP                 XMM.RTN    ELSE SERVICE INTERRUPT
                        *
01065  017 122517  XMM.RTN            PASS B    M        RESET B-REG
01066  000 013257  XMS.RTN            INC  S6   CNTR     S6 <= REMAINING COUNT(2'S COMP)
01067  017 170157                     PASS L    X        L  <= ORIGINAL COUNT(POSITIVE)
01070  010 013617                     CMPS X    CNTR     X  <= REMAINING COUNT(POSITIVE)
01071  004 153257                     ADD  S6   S6       S6 <= ORIGINAL - REMAINING
01072  017 152157                     PASS L    S6       L  <= WORDS COMPLETED
01073  004 126557                     ADD  A    A        A  <= A + TOTAL COMPLETED
01074  017 145736  P.RTN         RTN  PASS P    S3       P  <= NEXT INSTRUCTION
```

```
                   *
                   *
                   *
                   *
                   *
                   *
                   ************************************************************************
01075  322 003631  XMS       JMP  CNDX AL15 RJS  P.RTN     TEST FOR X<0 ... NOP
01076  017 124620  MELOOP2        MESP PASS MEU  B         MAP REG <= DATA
01077  000 024523                 ICNT INC  B    B         B  <= B + 1; INC CNTR
01100  337 003731            JMP  CNDX CNT4 RJS  MELOOP2   LOOP FOR 16X
01101  323 043331            JMP  CNDX CNT8      XMS.RTN   IS TOTAL LOOP FINSIHED
01102  335 003731            JMP  CNDX INT  RJS  MELOOP2   TEST FOR NO INTERRUPT
01103  007 145117                      DEC  S3   S3        RESET P REGISTER FOR RESTART
01104  320 043330            JMP                 XMS.RTN     ELSE SERVICE INTERRUPT
                   *
01105  007 171157  READMAP             DEC  S4   X         S4 <= X-1
01106  017 146257                      PASS CNTR S4        CNTR <= CNT+1(TWO'S COMP)
01107  000 074721  MELOOP3        MPCK INC  PNM  P         M.P.CHECK; P <= P+1
01110  017 135220                 MESP PASS S5   MEU       S5 <= MAP REG
01111  177 150023            WRTE ICNT PASS TAB  S5        WRITE DATA INTO TABLE
01112  337 004371            JMP  CNDX CNT4 RJS  MELOOP3   LOOP FOR 16X
01113  323 044731            JMP  CNDX CNT8      XMM.RTN*
01114  335 004371            JMP  CNDX INT  RJS  MELOOP3   TEST FOR NO INTERRUPT
01115  007 145117                      DEC  S3   S3        RESET P REGISTER FOR RESTART
                   *         JMP                 XMM.RTN*    ELSE SERVICE INTERRUPT
                   *
01116  017 174517  XMM.RTN*            PASS B    P         RESET B-REG
01117  017 170157                      PASS L    X         L  <= ORIGINAL COUNT(NEGATIVE)
01120  000 013617                      INC  X    CNTR      X  <= REMAINING COUNT(2'S COMP)
01121  003 071257                      SUB  S6   X         S6 <= ORIGINAL - REMAINING
01122  017 152157                      PASS L    S6        L  <= WORDS COMPLETED
01123  004 126557                      ADD  A    A         A  <= A + TOTAL COMPLETED
01124  017 145736                 RTN  PASS P    S3        P  <= NEXT INSTRUCTION
                   *
                   *
                   *
                   *
                   *
                   ************************************************************************
01125  353 077157  XM*       IMM       CMHI S4   %337      S4 <= 0010000000000000
01126  157 102742            LWF  L1   PASS      CAB       T-BUS <= A/B; FLAG <= A/B(15)
01127  321 105531  PA.PB     JMP  CNDX ALO  RJS  SY.US     TEST FOR PORT.A MAP
01130  345 176157            IMM       LOW  L    %177      L  <= 1111111101111111
01131  016 147157                      SONL S4   S4        S4 <= 0010000010000000
01132  324 005671  SY.US     JMP  CNDX FLAG RJS  XFER      TEST FOR SYSTEM MAP
01133  347 076157            IMM       LOW  L    %337      L  <= 1111111111011111
01134  016 147157                      SONL S4   S4        S4 <= 00100000X0100000
01135  017 146617  XFER                PASS MEU  S4        MEM ADDR REG <= S4(7-0)
01136  344 000255            IMM  RPT  LOW  CNTR %000      CNTR <= 0; SET REPEAT FF FOR 16X
01137  017 134620                 MESP PASS MEU  MEU       MEM PORT REG <= MEM PROG REG
01140  344 000255            IMM  RPT  LOW  CNTR %000      CNTR <= 0; SET REPEAT FF FOR 16X
01141  017 134620                 MESP PASS MEU  MEU       MEM PORT REG <= MEM PROG REG
01142  017 136776  RTN*           RTN                      RETURN
                   ************************************************************************
```

```
                   *
                   *
                   *
                   *
                   *
                   ************************************************************************
01143  300 057570  XL*       JSB                 OPGET     GET OPERAND ADDR FROM INSTR + 1
01144  000 075720                 MESP INC  P    P
01145  220 022457            READ      INC  M    M         SWITCH MAPS; GET REAL OPERAND
01146  017 134617                      PASS MEU  MEU       RESET MAP STATE
01147  017 100076                 RTN  PASS CAB  TAB
                   ************************************************************************
01150  300 057570  XS*       JSB                 OPGET     GET OPERAND ADDR FROM INSTR + 1
01151  000 075720                 MESP INC  P    P         SWITCH MAP STATE
01152  000 022461                 MPCK INC  M    M
01153  177 102017            WRTE      PASS TAB  CAB
01154  017 134636                 RTN  PASS MEU  MEU       RESET MAP STATE
                   ************************************************************************
```

```
01155  300 057570  XC*      JSB                      OPGET     GET OPERAND ADDR FROM INSTR + 1
01156  017 102160                MESP PASS L    CAB       L  <= A/B; SET ALTERNATE MAP
01157  220 022457           READ      INC  M    M         GET REAL OPERAND
01160  000 075720                MESP INC  P    P         P  <= INSTR + 1; RESET MAP
01161  013 000757                     XOR       TAB       COMPARE A/B WITH MEMORY
01162  320 046131           JMP  CNDX TBZ       RTN*      RTN-DON'T SKIP IF EQUAL
01163  000 075736                RTN  INC  P    P         P  <= INSTR + 2; RETURN
                   **********************************************************************
01164  340 016157  LF*      IMM       HIGH L    %007      L  <= 0000011111111111
01165  015 103157                     AND  S4   CAB       S4 <= A/B(10-0)
01166  017 134617                     PASS MEU  MEU       SEND "FENCE" DIRECTIVE
01167  017 146636                RTN  PASS MEU  S4        MEM FENCE <= S4; RETURN
                   **********************************************************************
                   *
                   *
                   *
                   *
                   *
                   **********************************************************************
01170  341 001157  DJP      IMM       HIGH S4   %100      S4 <= 0100000011111111
01171  320 047670           JMP                      JP*
                   *
01172  341 005157  SJP      IMM       HIGH S4   %102      S4 <= 0100001011111111
01173  320 047670           JMP                      JP*
                   *
01174  341 007157  UJP      IMM       HIGH S4   %103      S4 <= 0100001111111111
01175  300 057540  JP*      JSB  IOFF           OPGET     GET OPERAND ADDR FROM INSTR + 1
01176  017 146617  JMPSTAT            PASS MEU  S4        MEM STATUS IS SET HERE
01177  017 122761                MPCK PASS      M         S-BUS <= ADDRESS; CHECK TARGET
01200  017 123736                RTN  PASS P    M         P  <= TARGET ADDRESS; RETURN
                   **********************************************************************
01201  341 001157  DJS      IMM       HIGH S4   %100      S4 <= 0100000011111111
01202  320 050330           JMP                      JS*
                   *
01203  341 005157  SJS      IMM       HIGH S4   %102      S4 <= 0100001011111111
01204  320 050330           JMP                      JS*
                   *
01205  341 007157  UJS      IMM       HIGH S4   %103      S4 <= 0100001111111111
01206  300 057540  JS*      JSB  IOFF           OPGET     GET OPERAND ADDR FROM INSTR + 1
01207  000 075117                     INC  S3   P         S3 <= RETURN ADDRESS
01210  017 146617                     PASS MEU  S4        MEM STATUS IS SET HERE
01211  017 122761                MPCK PASS      M         S-BUS <= ADDRESS; CHECK TARGET
01212  177 144017           WRTE      PASS TAB  S3        WRITE RETURN ADDR AT TARGET
01213  000 023736                RTN  INC  P    M         P  <= TARGET + 1
                   **********************************************************************
                   *
                   *
                   *
                   *
                   *
                   **********************************************************************
01214  340 000160  MBF      IMM  MESP HIGH L    %000      L  <= 0000000011111111;SET ALT MAP
01215  017 126544  MBI           R1   PASS A    A         A  <= SOURCE WORD ADDRESS
01216  017 124504                R1   PASS B    B         B  <= DESTINATION WORD ADDRESS
01217  157 171604           LWF  R1   PASS X    X         X  <= WORD COUNT; FLAG <= ODD BYTE
01220  300 053230           JSB                      X.LOOP-1  MOVE BYTES IN PAIRS
01221  017 170757                     PASS      X         T-BUS <= X
01222  320 012771           JMP  CNDX TBZ  RJS  B.RESET   TEST FOR INTERRUPTED MOVE
01223  324 013031           JMP  CNDX FLAG RJS  B.RESET+1 TEST FOR NO ODD BYTE
01224  017 120760                MESP PASS      ADR       ALO <= IR(0); SET ALTERNATE MAP
01225  321 151371           JMP  CNDX ALO       *+2       TEST FOR MBF INSTRUCTION
01226  340 000160           IMM  MESP HIGH L    %000      L  <= 0000000011111111;SET ALT MAP
01227  220 026457           READ      INC  M    A         M  <= SOURCE ADDRESS
01230  017 126542                L1   PASS A    A         FORM BYTE ADDRESS IN A
01231  013 101157                     SANL S4   TAB       S4 <= AAAAAAAA00000000
01232  320 052320           JMP  MESP           MB*
                   **********************************************************************
01233  017 126544  MBW           R1   PASS A    A         A  <= SOURCE WORD ADDRESS
01234  017 124504                R1   PASS B    B         B  <= DESTINATION WORD ADDRESS
01235  157 171604           LWF  R1   PASS X    X         X  <= WORD COUNT; FLAG <= ODD BYTE
01236  300 054070           JSB                      W.LOOP-1  MOVE BYTES IN PAIRS
01237  017 170760                MESP PASS      X         T-BUS <= X; SELECT ALTERNATE MAP
01240  320 012771           JMP  CNDX TBZ  RJS  B.RESET   TEST FOR INTERRUPTED MOVE
01241  324 013031           JMP  CNDX FLAG RJS  B.RESET+1 TEST FOR NO ODD BYTE
01242  340 000157           IMM       HIGH L    %000      L  <= 0000000011111111
01243  220 026457           READ      INC  M    A         M  <= SOURCE ADDRESS
01244  017 126542                L1   PASS A    A         FORM BYTE ADDRESS IN A
01245  013 101157                     SANL S4   TAB       S4 <= AAAAAAAA00000000
                   *
```

```
01246  220 024461  MB*      READ MPCK INC  M    B         M  <= DESTINATION ADDRESS
01247  017 124502                 L1   PASS B    B         FORM BYTE ADDRESS IN B
01250  015 101217                      AND  S5   TAB       S5 <= 00000000BBBBBBBB
01251  017 150157                      PASS L    S5        L  <= S5
01252  017 047157                      IOR  S4   S4        S4 <= AAAAAAAABBBBBBBB
01253  177 146017           WRTE       PASS TAB  S4        WRITE DATA INTO DESTINATION
01254  017 134617                      PASS MEU  MEU       RESET SELECTED MAP
01255  000 026557                      INC  A    A         A  <= A + 1
01256  000 024536                 RTN  INC  B    B         B  <= B + 1
                   **********************************************************************
01257  157 171602  B.RESET  LWF  L1   PASS X    X         RESET X IN BYTES
01260  017 126542                 L1   PASS A    A         RESET A FOR EVEN BYTE ADDRESS
01261  017 124502                 L1   PASS B    B         RESET B FOR EVEN BYTE ADDRESS
01262  017 134636                 RTN  PASS MEU  MEU       RESET SELECTED MAP; RETURN
                   **********************************************************************
                   *
                   *
                   *
                   *
                   *
                   **********************************************************************
01263  017 170757  MWI                 PASS      X         T-BUS <= X
01264  320 054631           JMP  CNDX TBZ       MW*       TEST FOR X=0
01265  220 026457  X.LOOP   READ      INC  M    A         READ SOURCE WORD
01266  000 026560                 MESP INC  A    A         INCR. SOURCE ADDR.; SWITCH MAPS
01267  017 101157                      PASS S4   TAB       S4 <= DATA
01270  000 024461                 MPCK INC  M    B         M.P.CHECK; M <= DEST ADDRESS
01271  177 146017           WRTE       PASS TAB  S4        WRITE DATA INTO DESTINATION
01272  000 024517                      INC  B    B         INCREMENT DESTINATION ADDRESS
01273  007 171620                 MESP DEC  X    X         DECREMENT COUNT; SWITCH MAPS
01274  320 054631           JMP  CNDX TBZ       MW*       TEST IF MOVE COMPLETE
01275  335 013271           JMP  CNDX INT  RJS  X.LOOP    TEST FOR NO INTERRUPT
                   *
01276  007 175717                      DEC  P    P         P  <= INSTR ADDR
01277  017 134636                 RTN  PASS MEU  MEU       RESET SELECTED MAP; RETURN
                   **********************************************************************
01300  017 170757  MWW                 PASS      X         SET ALTERNATE MAP; T-BUS <= X
01301  320 054631           JMP  CNDX TBZ       MW*       TEST FOR X=0
01302  220 026457  W.LOOP   READ      INC  M    A         READ SOURCE WORD
01303  000 026557                      INC  A    A         INCREMENT SOURCE ADDRESS
01304  017 101157                      PASS S4   TAB       S4 <= DATA
01305  000 024461                 MPCK INC  M    B         M.P.CHECK; M <= DEST ADDRESS
01306  177 146017           WRTE       PASS TAB  S4        WRITE DATA INTO DESTINATION
01307  000 024517                      INC  B    B         INCREMENT DESTINATION ADDRESS
01310  007 171617                      DEC  X    X         DECREMENT COUNT
01311  320 054631           JMP  CNDX TBZ       MW*       TEST IF MOVE COMPLETE
01312  335 014131           JMP  CNDX INT  RJS  W.LOOP    TEST FOR NO INTERRUPT
                   *
01313  007 175717                      DEC  P    P         P  <= INSTR ADDR
01314  017 134636  MW*           RTN  PASS MEU  MEU       RESET SELECTED MAP; RETURN
                   **********************************************************************
                   *
                   *
                   *
                   *
                   *
                   **********************************************************************
01315  353 077157  SY*      IMM       CMHI S4   %337      S4 <= 0010000000000000
01316  320 055370           JMP                 MAPMOVE
                   **********************************************************************
01317  351 175144  PA*      IMM  R1   CMHI S4   %176      S4 <= 0100000010000000
01320  017 147144                 R1   PASS S4   S4        S4 <= 0010000001000000
01321  320 055370           JMP                 MAPMOVE
                   **********************************************************************
01322  346 077157  PB*      IMM       LOW  S4   %237      S4 <= 1111111110011111
01323  320 055270           JMP                 US*+1     L  <= 1101111111111111
                   *                                       S4 <= 0010000001100000
                   **********************************************************************
01324  347 077157  US*      IMM       LOW  S4   %337      S4 <= 1111111111011111
01325  343 076157           IMM       HIGH L    %337      L  <= 1101111111111111
01326  013 047157                      XOR  S4   S4        S4 <= 0010000000100000
01327  017 146617  MAPMOVE             PASS MEU  S4        MEM ADDR REG <= S4
01330  347 076257           IMM       LOW  CNTR %337      CNTR <= 11011111 (-41B)
01331  017 175117                      PASS S3   P         S3 <= P
01332  017 103717                      PASS P    CAB       P  <= A/B
01333  322 056171           JMP  CNDX AL15      MELOOP5   AL15=1 => READ MAPS
                   *
```

```
01334  220 074717           READ      INC  PNM  P         READ FIRST WORD; P <= P + 1
01335  017 101217  MELOOP4            PASS S5   TAB       S5 <= MAP DATA
01336  017 150620                MESP PASS MEU  S5        MAP REG <= DATA
01337  220 074723           READ ICNT INC  PNM  P         READ NEXT WORD; P <= P + 1
01340  323 015671           JMP  CNDX CNT8 RJS  MELOOP4   LOOP FOR 32K
01341  017 122057                     PASS CAB  M         A/B <= A/B + 32
01342  017 145736                RTN  PASS P    S3        P  <= INSTR + 1
                   *
01343  000 074721  MELOOP5       MPCK INC  PNM  P         M.P.CHECK; P <= P + 1
01344  017 135220                MESP PASS S5   MEU       S5 <= MAP DATA
01345  177 150023           WRTE ICNT PASS TAB  S5        WRITE DATA INTO TABLE
01346  323 016171           JMP  CNDX CNT8 RJS  MELOOP5   LOOP FOR 32K
01347  017 174057                     PASS CAB  P         A/B <= A/B + 32
01350  017 145736                RTN  PASS P    S3        P  <= INSTR + 1
                   ***********************************************************************
                   *
                   *
                   *
                   *
                   *
                   ***********************************************************************
01351  300 057570  SSM      JSB                 OPGET     GET OPERAND ADDR FROM INSTR + 1
01352  017 122761                MPCK PASS      M         M.P. CHECK BEFORE WRITE
01353  017 134617                     PASS MEU  MEU       SEND "STATUS" DIRECTIVE
01354  177 134017           WRTE      PASS TAB  MEU       WRITE STATUS WORD INTO MEMORY
01355  000 075736                RTN  INC  P    P         P <= INSTR + 2; RETURN
                   ***********************************************************************
01356  300 057540  JRS      JSB  IOFF           OPGET     GET OPERAND ADDR FROM INSTR + 1
01357  220 022457           READ      INC  M    M         READ THE STATUS WORD
01360  341 007157           IMM       HIGH S4   %103      S4 <= 0100001111111111
01361  157 101202           LWF  L1   PASS S5   TAB       FLAG <= STAT(15); S5(15) <= STAT(14)
01362  220 044457           READ      INC  M    S3        READ JMP TARGET
01363  300 057670           JSB                 OPGET+2   GET TARGET ADDR FROM INSTR + 2
01364  324 057331  ON.OFF   JMP  CNDX FLAG      SY.USR    TEST IF MEM WAS ON
01365  341 003157           IMM       HIGH S4   %101      IF OFF, S4 <= 0100000111111111
01366  157 151204  SY.USR   LWF  R1   PASS S5   S5        S5 <= STAT; AL15 <= STAT(14)
01367  322 047731           JMP  CNDX AL15      JMPSTAT   TEST STAT(14) FOR USER SELECTED
01370  341 004157           IMM       HIGH L    %102      IF SYS, L  <= 0100001011111111
01371  015 147157                     AND  S4   S4          THEN  S4 <= 010000X011111111
01372  320 047730           JMP                 JMPSTAT   SET STATUS OF MEM; ALSO SET P
                   ***********************************************************************
01373  220 074457  OPGET    READ      INC  M    P
01374  000 075117                     INC  S3   P         S3 <= P + 1; S3 <= INSTR + 2
01375  000 000457                     INC  M    TAB       M  <= NEXT ADDR
01376  322 006131           JMP  CNDX AL15 RJS  RTN*      TEST FOR NO INDIRECT
01377  320 000570           JMP                 INDLEVEL
                   ***********************************************************************
                   *
                   *  LISTING OF INDLEVEL ROUTINE FOR REFERENCE ONLY
                   *
                   *INDLEVEL READ      INC  M    M         READ NEXT LEVEL
                   *        JMP  CNDX NHOI RJS  IND2      HALT OR INTERRUPT?
                   *INDIRECT     INCI PASS M    TAB       M<=T/A/B; INCR INDIRECT COUNTER
                   *        JMP  CNDX AL15      INDLEVEL  CHECK FOR ANOTHER LEVEL OF INDIRECT
                   *        READ RTN  INC  M    M         READ EFFECTIVE ADDRESS, RETURN
                   *IND2         INCI PASS M    TAB       M<=T/A/B; INCR INDIRECT COUNTER
                   *        JMP  CNDX NSNG RJS  INDIRECT+1JUMP BACK FOR SINGLE INSTRUCTION
                   *                   DEC  P    P         RESET P
                   *        JMP                 HORI      HALT OR INTERRUPT
                   *
                   ***********************************************************************
                            END
```

```
                            ORG                 %7000
                   *************************************************************************
                   *
                   *      21MX MICRO-CODE
                   *      MODULE 14: FLOATING POINT INSTRUCTIONS
                   *
                   *************************************************************************
                   INDIRECT EQU                 %0015
                   MPYX     EQU                 %0246
                   *************************************************************************
                   *
```

```
07000  017 125414  IFIX            COV  PASS S9   B         CLEAR THE OVFL AND PUT EXP IN S9
07001  321 100171            JMP   CNDX ALO  RJS  *+2       TEST FOR NEG EXP
07002  001 136576                  RTN  ZERO A              IF EXP<0 WE CAN'T FIX
07003  017 126517                       PASS B    A         PUT HIBITS IN B-REG
07004  344 000157            IMM        LOW  L    %000      PUT 'UP-8' MASK IN L
07005  015 160557                       AND  A    S9        MASK LEAST SIG. 8 BITS INTO A
07006  015 161457                       AND  S10  S9        SAVE BITS FOR ROUND-OFF
07007  013 161404                  R1   SANL S9   S9        MASK EXP INTO S9 WITHOUT SIGN
07010  347 140157            IMM        LOW  L    %360      PUT -20(B8) INTO L
07011  004 161413                  SOV  ADD  S9   S9        CHECK TO SEE IF EXP TOO LARGE
07012  320 140771            JMP   CNDX ONES      NOSHIFT   OR IF NO SHIFT REQUIRED
07013  322 005231            JMP   CNDX AL15 RJS  OVER      IF SO THEN WE CAN'T FIX
07014  000 061417                       INC  S9   S9        START LOOP TO SHIFT DIGITS
07015  017 160255                  RPT  PASS CNTR S9        PASS # OF SHIFTS INTO CNTR
07016  037 124504            ARS   R1   PASS B    B         32-BIT SHIFT
07017  017 126157  NOSHIFT              PASS L    A         HOLD LEFTOVER BITS IN L
07020  017 124554                  COV  PASS A    B         PUT INTEGER INTO A-REG
07021  322 017631            JMP   CNDX AL15 RJS  RTNFP     TEST FOR NEG INTEGER
07022  017 062757                       IOR       S10       IF NEG THEN CHECK FOR TRUNC  BITS
07023  320 057631            JMP   CNDX TBZ       RTNFP     IF ALL ZEROS WE ARE DONE
07024  000 024576                  RTN  INC  A    B         OTHERWISE INC THE INTEGER & RTN
                   *********************************************************************
                   *********************************************************************
07025  017 126517  FLOAT                PASS B    A         PUT INTEGER IN B-REG
07026  001 136557                       ZERO A              CLEAR A-REG
07027  357 141417            IMM        CMLO S9   %360      STORE +15(B10) IN EXP REG
07030  321 142530            JMP                  PACK
                   *
                   *********************************************************************
                   *
07031  017 101317  FLO                  PASS S7   TAB       STORE HIBITS IN S7
07032  000 023017                       INC  S1   M         INC ADDRS FOR NEXT READ
07033  340 000157            IMM        HIGH L    %000      STORE 'LO-8' MASK IN L
07034  220 040457            READ       INC  M    S1        READ SECOND HALF OF WRD
07035  015 125417                       AND  S9   B         MEANWHILE,MASK EXP OF WRD1 INTO S9
07036  017 101217                       PASS S5   TAB       STORE WRD2 LOBITS/EXP IN S5
07037  013 125457                       SANL S10  B         MASK LOBITS OF WRD1 INTO S10
07040  013 151257                       SANL S6   S5        MASK LOBITS OF WRD2 INTO S6
07041  015 151204                  R1   AND  S5   S5        MASK EXP OF WRD2 INTO S1I WITHOUT SGN
07042  321 102271            JMP   CNDX ALO  RJS  *+3       IF SIGN WAS POS, JMF
07043  346 000157            IMM        LOW  L    %200      OTHERWISE PUT -200(B8) INTO L
07044  004 151217                       ADD  S5   S5        ADD TO EXP OF WRD2
07045  017 161404                  R1   PASS S9   S9        MASK EXP OF WRD1 INTO S9 WITHOUT SIGN
07046  321 102471            JMP   CNDX ALO  RJS  *+3       IF SIGN WAS POS, JMP
07047  346 000157            IMM        LOW  L    %200      OTHERWISE PUT -200(B8) INTO L
07050  004 161417                       ADD  S9   S9        ADD TO EXP OF WRD1
07051  017 127536                  RTN  PASS S11  A         PUT HIBITS OF WRD1 INTO S3 & RTN
                   *********************************************************************
                   *********************************************************************
07052  017 126154  PACK            COV  PASS L    A         CLR OVFL AND PUT WRD1 LOBITS INTO L
07053  001 137457                       ZERO S10            CLEAR COUNTER REG
07054  017 024757                       IOR       B         PASS THRU ALU WITH HIBITS
07055  320 057631            JMP   CNDX TBZ       RTNFP     IF A/B IS ZERO,RTN
07056  346 003517            IMM        LOW  S11  %201      STORE -177(B8) IN S11
07057  037 124742  NRMLZ     ARS   L1   PASS      B         TEST IF NUMBER IS NORMALIZED
07060  325 043231            JMP   CNDX OVFL      RND       IF SO, JMP TO ROUNDING ROUTINE
07061  077 124502            LGS   L1   PASS B    B         IF NOT. DO 32-BIT LEFT-SHIFT
07062  000 063457                       INC  S10  S10       INC THE EXP CNTR
07063  321 142770            JMP                  NRMLZ     GO BACK TO CHECK FOR NORMAL NUMBER
07064  322 043331  RND       JMP   CNDX AL15      *+2       SINCE B WAS JUST PASSED THRU ALU
07065  007 165517                       DEC  S11  S11       CHECK SGN & ADJUST ROUND OFF
07066  017 164154                  COV  PASS L    S11       PUT 'ROUND' INTO L
07067  003 026557                       SUB  A    A         ACTUALLY: ADD 200(B8) TO LOBITS
07070  321 004171            JMP   CNDX COUT RJS  XPNT      IF NO COUT FROM LOBITS, OK, JMP
07071  340 000157            IMM        HIGH L    %0        CLR L(15) FOR OVERFLOW
07072  240 024517            ENV        INC  B    B         IF COUT, INC HIBITS AND CHECK FOR OVFL
07073  325 003771            JMP   CNDX OVFL RJS  *+4       IF NO OVFL, OK, JMP
07074  017 124504                  R1   PASS B    B         OVFL IMPLIES B/A= 1000...
07075  000 061414                  COV  INC  S9   S9        SO WE SHIFT B TO FORM 0100 ..&
07076  321 144170            JMP                  XPNT      BUMP EXP, THEN JMP
07077  037 124742            ARS   L1   PASS      B         IF B NEQ 100...,CHECK IF B=111..
07100  325 044171            JMP   CNDX OVFL      XPNT      IF NOT, JMP
07101  077 124502            LGS   L1   PASS B    B         RE-NORMALIZE
07102  000 063457                       INC  S10  S10
07103  017 162153  XPNT            SOV  PASS L    S10       CLR OVFL AND PUT EXP INTO L
07104  003 061417                       SUB  S9   S9        SUB CALC EXP FROM ORIG EXP
07105  346 000157            IMM        LOW  L    %200      PUT -200(B8) INTO L
07106  003 060757                       SUB       S9        TEST FOR EXP UNDERFLO
07107  322 045131            JMP   CNDX AL15      UNFLO     IF SO, JMP
07110  004 160757                       ADD       S9        TEST FOR EXP OVERFLOW
07111  322 017671            JMP   CNDX AL15 RJS  OVFLO     IF SO, JMP (TO 7375)
07112  157 160742            LWF   L1   PASS      S9        PASS EXP SIGN INTO FLAG-REG
```

```
07113  157 161402            LWF   L1   PASS S9   S9         SHIFT EXP WITH SIGN
07114  340 000157            IMM        HIGH L    %000       STORE 'LO-8' MASK IN L
07115  015 161457                       AND  S10  S9         MASK EXP INTO S10
07116  013 127417                       SANL S9   A          MASK LOBITS INTO S9
07117  017 124557                       PASS A    B          PUT HIBITS INTO A-REG
07120  017 160154                  COV  PASS L    S9         PUT LOBITS INTO L
07121  017 062536                  RTN  IOR  B    S10        COMBINE WITH EXP AND STORE IN B-REG
07122  001 136557  UNFLO                ZERO A               CLEAR A-REG; OVFL=1
07123  001 136536                  RTN  ZERO B               NOW CLR B-REG AND RTN
                   *
07124  341 176576  OVER      IMM   RTN  HIGH A    %177       SET UP ERROR CONDITION IN A
                   ************************************************************************
                   ************************************************************************
07125  017 136750  FADD            STFL
                   *
07126  220 074457  FSUB      READ       INC  M    P          PASS P INTO M TO READ ADDR OF WRD2
07127  300 000670            JSB                  INDIRECT   CHECK FOR INDIRECTS
07130  301 141470            JSB                  FLD        UNPACK WRDS INTO SCRATCH REGS
07131  017 154517                       PASS B    S7         CHECK FOR WRD2=0
07132  320 005631            JMP   CNDX TBZ  RJS  *+2        IF NOT,CONTINUE
07133  346 001217            IMM        LOW  S5   %200       IF SO,MAKE EXP MOST NEG (-200,B8)
07134  017 164757                       PASS      S11        CHECK FOR WRD1=0
07135  320 005771            JMP   CNDX TBZ  RJS  *+2        IF NOT,CONTINUE
07136  346 001417            IMM        LOW  S9   %200       IF SO,MAKE EXP MOST NEG (-200,B8)
07137  324 046531            JMP   CNDX FLAG      DIFR       IF DOING ADD,SKIP AHEAD
07140  010 024517                       CMPS B    B          FORM 2-COMP OF HIBITS IN B
07141  010 053257                       CMPS S6   S6         FORM 2-COMP OF
07142  000 053257                       INC  S6   S6         LOBITS OF WRD2
07143  321 006531            JMP   CNDX COUT RJS  DIFR       IF COUT OCCURS
07144  000 024517                       INC  B    B          BUMP HIBITS
07145  322 006531            JMP   CNDX AL15 RJS  DIFR       CHECK SGN; IF POS,JMP
07146  017 124742                  L1   PASS      B          IF NEG,CHECK FOR MOST
07147  320 006531            JMP   CNDX TBZ  RJS  DIFR       NEG #(100. .)
07150  017 124504                  R1   PASS B    B          IF SO,SHIFT BACK (010...)
07151  000 051217                       INC  S5   S5         &BUMP EXP
07152  017 152554  DIFR            COV  PASS A    S6
07153  017 150157                       PASS L    S5         FIND DIFF IN EXPS
07154  003 061351                  CLFL SUB  S8   S9         &STORE IN S8; FLG=0
07155  320 047731            JMP   CNDX TBZ       ADD2       IF DIFF=0,JMP TO ADD STEP
07156  322 047131            JMP   CNDX AL15      RVRS       IF NEG,WRD2>WRD1
07157  010 057357                       CMPS S8   S8         FORM -DIFF
07160  000 057357                       INC  S8   S8         & STORE -DIFF IN S8
07161  321 147430            JMP                  SWAMPCHK
07162  017 124157  RVRS                 PASS L    B          HOLD B IN L
07163  017 164517                       PASS B    S11        WRD1<WRD2; FILL B,A
07164  017 162557                       PASS A    S10        WITH S11,S10
07165  015 037517                       PASL S11             ALSO FILL S11,S10,S9
07166  017 153457                       PASS S10  S6         WITH B,S6,S5
07167  017 151417                       PASS S9   S5
07170  347 120157  SWAMPCHK  IMM        LOW  L    %350       FORM -30(B8) IN L
07171  003 056757                       SUB       S8         IF -DIFF>-31,RTN WITH LARGER #
07172  322 050731            JMP   CNDX AL15      OUT        JMP TO RESTORE A,B
07173  037 124504  SHIFT     ARS   R1   PASS B    B          NOW START SHIFT LOOP
07174  000 057357                       INC  S8   S8         INC COUNTER
07175  320 007571            JMP   CNDX TBZ  RJS  SHIFT      LOOP UNTIL DONE
                   *
                   ***** CONTINUED ON NEXT PAGE ******************************************
                   *
07176  157 164142  ADD2      LWF   L1   PASS L    S11        FLG <= SIGN; L <= HIBITS
07177  244 124517            ENV        ADD  B    B          ADD HIBITS; B <= RESULTS; OVFL?
07200  017 162157                       PASS L    S10        L <= LOBITS
07201  004 126557                       ADD  A    A          ADD LOBITS; A <= RESULTS
07202  321 010271            JMP   CNDX COUT RJS  *+3        CHECK FOR COUT FROM LOBITS
07203  341 176157            IMM        HIGH L    %177       L <= 0111111111111111
07204  240 024517            ENV        INC  B    B          B <= B + 1; OVFL?
07205  325 016271            JMP   CNDX OVFL RJS  PKSUB      IF NO OVERFLOW, RETURN
07206  324 010531            JMP   CNDX FLAG RJS  OFLOW      CHECK IF ADDEND, AUGEND WERE POS
07207  341 176157            IMM        HIGH L    %177       L <= 0111111111111111
07210  013 024757                       XOR       B          TEST FOR B = 1000000000000000
07211  320 156271            JMP   CNDX ONES      PKSUB      IF TRUE, THEN IGNORE OVERFLOW
                   *
07212  157 124504  OFLOW     LWF   R1   PASS B    B          DO FULLWORD SHIFT
07213  157 126544            LWF   R1   PASS A    A             USING FLAG TO INJECT SIGN BIT
07214  000 061417                       INC  S9   S9         BUMP EXPONENT
07215  321 156270            JMP                  PKSUB
                   *
07216  017 164517  OUT                  PASS B    S11        PASS MUCH LARGER WRD INTO B,A
07217  017 162557                       PASS A    S10
07220  321 156270            JMP                  PKSUB
                   ************************************************************************
```

```
                  ************************************************************************
07221 220 074457  FMPY   READ       INC  M    P          PASS P INTO M TO READ ADDR OF WRD2
07222 300 000670         JSB                  INDIRECT   CHECK FOR INDIRECTS
07223 301 141470         JSB                  FLD        STORE ARGS IN SCRATCH REGS
07224 000 061417                    INC  S9   S9
07225 017 150157                    PASS L    S5         FORM EXP1+EXP2+1
07226 004 161417                    ADD  S9   S9         AND SAVE IN S9
07227 017 162544              R1    PASS A    S10        FORM (WRD1 LOBITS)/2 IN A
07230 017 155057                    PASS S2   S7         PASS WRD2 HIBITS INTO S2
07231 300 012330         JSB                  MPYX       JMP TO MPY SUB & RTN WITH
07232 017 125217                    PASS S5   B          HIBITS IN B; SAVE IN S5
07233 017 165057                    PASS S2   S11        PASS WRD1 HIBITS INTO S2
07234 017 127517                    PASS S11  A          LOBITS INTO A; SAVE IN S11
07235 017 152544              R1    PASS A    S6         FORM (WRD2 LOBITS)/2 IN A
07236 300 012330         JSB                  MPYX       JMP TO MPY SUB & RTN WITH
07237 017 126157                    PASS L    A          LOBITS IN A; PASS INTO L
07240 004 164557                    ADD  A    S11        ADD BOTH LOBITS & CHK FOR COUT
07241 321 012171         JMP  CNDX  COUT RJS  *+2        (ELSE TRUNCATE DIGITS)
07242 000 024517                    INC  B    B          IF COUT,BUMP HIBITS
07243 017 124157                    PASS L    B          ADD HIBITS AND SAVE IN S11
07244 004 151517                    ADD  S11  S5
07245 017 154557                    PASS A    S7         PASS WRD2 HIBITS INTO A
07246 300 012330         JSB                  MPYX       JMP TO MPY SUB & RTN WITH
07247 017 126544              R1    PASS A    A          LOBITS IN A; SAVE LOBITS/2
07250 017 126154              COV   PASS L    A          ADD LOBITS/2 TO HIBITS SUM &
07251 244 164542         ENV  L1    ADD  A    S11        SHFT L1 TO REORIENT
07252 322 012671         JMP  CNDX  AL15 RJS  *+3        CHECK FOR CARRY INTO OR
07253 325 052771         JMP  CNDX  OVFL      *+4        BORROW FROM HIBITS &
07254 007 124517                    DEC  B    B          ADJUST ACCORDINGLY
07255 301 142530         JSB                  PACK
07256 000 075736              RTN   INC  P    P
07257 000 024517                    INC  B    B          CAN'T OVFL FROM HIBITS
07260 301 142530         JSB                  PACK
07261 000 075736              RTN   INC  P    P
                  ************************************************************************
                  ************************************************************************
07262 220 074457  FDIV   READ       INC  M    P          PASS P INTO M TO READ ADDR OF WRD2
07263 300 000670         JSB                  INDIRECT   CHECK FOR INDIRECTS
07264 301 141470         JSB                  FLD
07265 010 054554              COV   CMPS A    S7         PASS WRD2 HIBITS & CHECK
07266 320 156131         JMP  CNDX  ONES      DBYZR      FOR DIV BY ZERO
07267 322 053471         JMP  CNDX  AL15      *+2        SINCE WE USE SAME DVSR,MAKE POS
07270 000 027313              SOV   INC  S7   A          NOW & SAVE SGN IN OVFL
07271 017 150157                    PASS L    S5         FORM EXP1-EXP2+1
07272 003 061417                    SUB  S9   S9         & SAVE IN S9
07273 000 061417                    INC  S9   S9
07274 017 162557                    PASS A    S10        FILL B,A WITH WRD1 AS DVND
07275 017 164517                    PASS B    S11        & PRESHIFT TO AVOID OVFL
07276 037 124504         ARS  R1    PASS B    B
07277 301 156370         JSB                  DIVX       JMP TO SPECIAL DIV SUB
07300 017 127217                    PASS S5   A          SAVE QUO1 IN S5
07301 017 124757                    PASS      B          PASS QUO & CHECK FOR ODD/EVEN
07302 321 114231         JMP  CNDX  AL0  RJS  *+2        TO SIMULATE FIRST
07303 007 124517                    DEC  B    B          LEFT SHIFT IN DIV ROUTINE
07304 001 136557                    ZERO A               CLR DVND LOBITS; DVSR SAME
07305 301 156370         JSB                  DIVX       JMP TO SPEC DIV SUB
07306 017 127517                    PASS S11  A          SAVE QUO2 IN S11
07307 017 152504              R1    PASS B    S6         FORM (WRD2 LOBITS)/4 IN
07310 017 124504              R1    PASS B    B          B(=DVND HIBITS)
07311 001 136557                    ZERO A               CLR DVND LOBITS; DVSR SAME
07312 301 156370         JSB                  DIVX       JMP TO SPEC DIV SUB
07313 010 026557                    CMPS A    A          FORM 2-COMP OF QUO3
07314 000 026557                    INC  A    A          AS MPLR
07315 017 151057                    PASS S2   S5         PASS QUO1 AS MCND
07316 300 012330         JSB                  MPYX       JMP TO MPY SUB
07317 017 125317                    PASS S7   B          SAVE PROD HIBITS IN S5
07320 001 136517                    ZERO B               PRE-CLR B
07321 017 164757                    PASS      S11        CHECK SGN OF QUO2
07322 322 015231         JMP  CNDX  AL15 RJS  *+2        & EXTEND AS ALL 0'S(POS)
07323 016 036517                    ONE  B               OR ALL 1'S(NEG)
07324 017 154757                    PASS      S7         CHECK SGN OF -QUO1*QUO3
07325 322 015371         JMP  CNDX  AL15 RJS  *+2        IF NEG,SUB 1 FROM B
07326 007 124517                    DEC  B    B
07327 017 155302              L1    PASS S7   S7         REORIENT PROD (ADJUST EXP,REALLY)
07330 017 154542              L1    PASS A    S7
07331 017 126157                    PASS L    A
07332 004 164557                    ADD  A    S11        ADD TO QUO2
07333 321 015671         JMP  CNDX  COUT RJS  *+2        IF COUT OCCURRED
07334 000 024517                    INC  B    B          BUMP HIBITS OF RESULT
07335 077 124502         LGS  L1    PASS B    B          SHIFT FULLWRD TO ORIENT RESULT
07336 017 150157                    PASS L    S5         ADD QUO1 TO HIBITS
```

```
07337  004 124517                      ADD  B    B
07340  301 142530            JSB                 PACK
07341  000 075736                 RTN  INC  P    P
07342  001 136557  DBYZR               ZERO A              CLR LOBITS
07343  352 000517            IMM       CMHI B    %200      FORM 0111111100000000 IN HIBITS
07344  017 125417                      PASS S9   B         ALSO PASS INTO EXP
07345  301 142530  PKSUB     JSB                 PACK      REPACK A,B REGS
07346  000 075736                 RTN  INC  P    P         INC P AND RETURN
                   ************************************************************************
                   ************************************************************************
                   *
                   *     THIS IS A SPECIAL SUB FOR F.P. DIV
                   *         IT ASSUMES THAT DVSR IS ALWAYS POS
                   *                    THAT ORIG DVSR SGN IS IN OVFL REG
                   *                    THAT YOU HAVE PREVIOUSLY DONE FIRST LEFT SHIFT
                   *                AND THAT NO ERROR COND NEED BE CHECKED
                   *         (BUT IT IS FAST)
                   *
                   ************************************************************************
07347  344 000257  DIVX      IMM       LOW  CNTR %0        CLR CNTR
07350  157 124742            LWF  L1   PASS      B         CHECK FOR NEG DVND & SAVE SGN IN FLG
07351  322 016771            JMP  CNDX AL15 RJS  READY     IF POS, WE ARE READY
07352  010 024517                      CMPS B    B         COMP HIBITS
07353  010 026557                      CMPS A    A         COMP LOBITS
07354  000 026557                      INC  A    A         FORM 2-COMP OF LOBITS
07355  321 016771            JMP  CNDX COUT RJS  READY     IF NO COUT,OK
07356  000 024517                      INC  B    B         ELSE BUMP HIBITS
07357  017 154155  READY          RPT  PASS L    S7        PASS DVSR INTO L; SET RPTFF
07360  123 024502            DIV  L1   SUB  B    B         PERFORM DIV STEP(16X)
07361  017 124504                 R1   PASS B    B         FORM REM IN B
07362  324 017271            JMP  CNDX FLAG RJS  *+3       IF REM SGN IS TO BE NEG
07363  010 024517                      CMPS B    B         (DETERMINED BY DVND),THEN
07364  000 024517                      INC  B    B         FORM 2-COMP IN B
07365  325 057471            JMP  CNDX OVFL      *+4       CHECK ORIG DVSR SGN; IF POS,
07366  324 017631            JMP  CNDX FLAG RJS  RTNFP     LOOK FOR NEG DVND
07367  010 026557                      CMPS A    A         WHICH MEANS FORM
07370  000 026576                 RTN  INC  A    A         NEG QUO IN A & RTN
07371  324 057631            JMP  CNDX FLAG      RTNFP     ELSE IF NEG,LOOK FOR POS DVND
07372  010 026557                      CMPS A    A         WHICH MEANS FORM
07373  000 026576                 RTN  INC  A    A         NEG QUO IN A & RTN
                   ************************************************************************
                   *
07374  017 136776  RTNFP          RTN
                   *
07375  016 036544  OVFLO          R1   ONE  A              PUT MOST POS # AND MOST POS EXP
07376  347 174536            IMM  RTN  LOW  B    %376      INTO A,B-REGS; OVFLO=1
                   ************************************************************************
                             ORG                 %7400
                   ************************************************************************
                   *
                   *      21MX MICRO-CODE
                   *      MODULE 15: EXTENDED INSTRUCTION GROUP
                   *
                   ************************************************************************
                   FETCH     EQU                 %0000
                   ************************************************************************
                   *   JUMP TABLES - ENTERED FROM BASE SET
                   ************************************************************************
07400  321 163170            JMP                 EADRX     SAX/SBX
07401  017 103636                 RTN  PASS X    CAB       CAX/CBX
07402  321 163170            JMP                 EADRX     LAX/LBX
07403  321 163030            JMP                 EADR      STX
07404  017 170076                 RTN  PASS CAB  X         CXA/CXB
07405  321 163030            JMP                 EADR      LDX
07406  321 163030            JMP                 EADR      ADX
07407  321 164070            JMP                 XABX      XAX/XBX
07410  321 163630            JMP                 EADRY     SAY/SBY
07411  017 103676                 RTN  PASS Y    CAB       CAY/CBY
07412  321 163630            JMP                 EADRY     LAY/LBY
07413  321 163030            JMP                 EADR      STY
07414  017 172076                 RTN  PASS CAB  Y         CYA/CYB
07415  321 163030            JMP                 EADR      LDY
07416  321 163030            JMP                 EADR      ADY
07417  321 164230            JMP                 XABY      XAY/XBY
07420  321 164370            JMP                 ISX
07421  321 164670            JMP                 DSX
07422  321 177070            JMP                 JLY
07423  321 172530            JMP                 LBT
07424  321 171630            JMP                 SBT
07425  321 173230            JMP                 MBT
07426  321 175130            JMP                 CBT
07427  321 174030            JMP                 SFB
07430  321 164530            JMP                 ISY
```

```
07431  321 165030            JMP                 DSY
07432  321 177370            JMP                 JPY
07433  321 167330            JMP                 SBSCBS    SBS
07434  321 167330            JMP                 SBSCBS    CPS
07435  321 166630            JMP                 TBS
07436  321 170230            JMP                 CMW
07437  321 171030            JMP                 MVW
                    ********************************************************************************
                    *       INDEX REGISTER INSTRUCTIONS
                    ********************************************************************************
                    *                            DISPLACEMENT FROM FINISH CORREPONDS TO
                    *                            DISPLACEMENT FROM 7400B FOR INSTRS. LISTED
                    *                            IN COMMENT FIELD BELOW.
07440  000 022461   FINISH        MPCK INC  M    M         SAX/SBX
07441  177 102036            WRTE RTN  PASS TAB  CAB
07442  017 100076                 RTN  PASS CAB  TAB       LAX/LBX
07443  000 022461                 MPCK INC  M    M         STX
07444  177 170036            WRTE RTN  PASS TAB  X
07445  017 101636                 RTN  PASS X    TAB       LDX
07446  017 100157                      PASS L    TAB       ADX
07447  264 171636            ENVE RTN  ADD  X    X
07450  000 022461                 MPCK INC  M    M         SAY/SBY
07451  177 102036            WRTE RTN  PASS TAB  CAB
07452  017 100076                 RTN  PASS CAB  TAB       LAY/LBY
07453  000 022461                 MPCK INC  M    M         STY
07454  177 172036            WRTE RTN  PASS TAB  Y
07455  017 101676                 RTN  PASS Y    TAB       LDY
07456  017 100157                      PASS L    TAB       ADY
07457  264 173676            ENVE RTN  ADD  Y    Y
                    ********************************************************************************
                    *                                      EADR IS COMMON TO LD*,ST*,AD*
07460  220 074717   EADR     READ      INC  PNM  P         READ WORD 2 . P<=ADDR OF NEXT INSTR.
07461  301 165630            JSB                 INDBIT    CHECK FOR INDIRECT,GET OPERAND
07462  321 162035            JMP  J30            FINISH    JUMP TO COMPLETE INSTRUCTION
                    ************************************************************************
                    *                                      EADRX DOES EFFECTIVE ADDR FOR
                    *                                      SAX,SBX,LAX,LBX INSTRS.
07463  220 074717   EADRX    READ      INC  PNM  P         READ ADDRESS OF WORD 2
07464  017 170157                      PASS L    X
07465  017 100457                      PASS M    TAB       M<=CONTENTS OF WORD 2.
07466  322 023471            JMP  CNDX AL15 RJS  DIRECT    JUMP IF NO INDIRECT.
                    ************************************************************************
                    *                                      INDIRECT ROUTINE FOR INDEXED INST
07467  220 022457   EADRI    READ      INC  M    M         READ INDIRECT ADDRESS
07470  301 165630            JSB                 INDBIT    JSB TO INDIRECT ROUTINE
                    ************************************************************************
                    *                                      COMPUTE INDEXED ADDRESS THEN JUMP
07471  004 123017   DIRECT             ADD  S1   M         S1<=TARGET ADDR. + X OR Y.
07472  220 040457            READ      INC  M    S1        READ INDEXED ADDRESS.
07473  321 162035            JMP  J30            FINISH    JUMP TO COMPLETE THE INSTRUCTION
                    ************************************************************************
                    *                                      EADRY COMPUTES EFFECTIVE ADDRESS
                    *                                      FOR SAY,SBY,LAY,LBY INSTRS.
07474  220 074717   EADRY    READ      INC  PNM  P
07475  017 172157                      PASS L    Y .
07476  017 100457                      PASS M    TAB       M<= CONTENTS OF WORD 2.
07477  322 023471            JMP  CNDX AL15 RJS  DIRECT    JUMP IF NO INDIRECTS
07500  321 163370            JMP                 EADRI     JUMP TO DO INDIRECT ROUTINE
                    ************************************************************************
07501  017 103017   XABX               PASS S1   CAB       EXCHANGE A/B WITH X
07502  017 170057                      PASS CAB  X
07503  017 141636                 RTN  PASS X    S1
                    ************************************************************************
07504  017 103017   XABY               PASS S1   CAB       EXCHANGE A/B WITH Y
07505  017 172057                      PASS CAB  Y
07506  017 141676                 RTN  PASS Y    S1
                    ************************************************************************
07507  000 071617   ISX                INC  X    X         INCREMENT X, SKIP IF ZERO
07510  320 067271            JMP  CNDX TBZ       SKIP
07511  017 136776   RETURN        RTN
                    ************************************************************************
07512  000 073657   ISY                INC  Y    Y         INCREMENT Y, SKIP IF ZERO.
07513  320 067271            JMP  CNDX TBZ       SKIP
07514  017 136776                 RTN
                    ************************************************************************
07515  007 171617   DSX                DEC  X    X         DECREMENT X, SKIP IF ZERO.
07516  320 067271            JMP  CNDX TBZ       SKIP
07517  017 136776                 RTN
                    ************************************************************************
07520  007 173657   DSY                DEC  Y    Y         DECREMENT Y, SKIP IF ZERO.
07521  320 067271            JMP  CNDX TBZ       SKIP
07522  017 136776                 RTN
```

```
                   *******************************************************************
                   *   GENERAL INDIRECT ROUTINE FOR INDEX BIT INSTR
                   *         COMMON ROUTINES FOR WORD/BYTE INSTRUCTIONS
                   *******************************************************************
                   *                                        INITIALIZATION FOR WORD,BYTE
07523  000 075217  INITCM             INC  S5   P          S5<= ADDRESS OF WORD 3.
07524  220 050457           READ      INC  M    S5         READ ADDRESS OF WORD 3.
07525  000 051157                     INC  S4   S5         S4<= ADDRESS OF NEXT INSTRUCTION.
07526  017 101117                     PASS S3   TAB        S3<= CONTENTS OF WORD 3.
07527  320 065531           JMP  CNDX TBZ       *+3        JUMP IF WORD 3 = 0 (NO INTERRUPT)
07530  000 075717                     INC  P    P          P<=ADDRESS OF WORD 3 (FOR EXIT)
07531  001 155336                RTN  ZERO S7   S7         S7<=0 AND RETURN TO CALLER.
07532  220 074710           READ STFL INC  PNM  P          READ ADDRESS OF WORD 2. P<=P+1.
07533  001 155317                     ZERO S7   S7         S7 <= 0.
                   *******************************************************
                   *                                        COMMON INDIRECT IMBEDDED IN INITCM
07534  017 100457  INDBIT             PASS M    TAB        M <= CONTENTS OF LAST READ ADDRESS.
07535  322 026271           JMP  CNDX AL15 RJS  CONTBIT    JUMP IF NO INDIRECT.
07536  220 022465  INDLBIT  READ INCI INC  M    M          READ ADDRESS IN M
07537  326 065631           JMP  CNDX NHOI      INDBIT     JUMP IF NO HALT OR INTERRUPT PENDING
07540  017 100457  IND2BIT            PASS M    TAB        M<= CONTENTS OF LAST READ ADDRESS
07541  330 125671           JMP  CNDX NSNG RJS  INDBIT+1   JUMP IF SINGLE-INSTRUCT. MODE
07542  007 175717  DEC2               DEC  P    P
07543  007 175717                     DEC  P    P          P <= ADDRESS OF WORD 1.
07544  320 000030           JMP                 FETCH      ATTEMPT JUMP TO FETCH ROUTINE.
07545  324 066371  CONTBIT  JMP  CNDX FLAG      *+2        FLAG IDENTIFIES CALLER TO INDBIT
07546  220 022476           READ RTN  INC  M    M          READ ADDRESS AND RETURN.
                   *******************************************************
07547  220 022451           READ CLFL INC  M    M          CALLER=INITCM--RESET FLAG,READ COUNT
07550  017 101257                     PASS S6   TAB        S6 <= COUNT FOR THIS INSTRUCTION
07551  320 026571           JMP  CNDX TBZ  RJS  RTNCNT     JUMP IF COUNT NOT ZERO.
07552  301 176730           JSB                 EXIT       END THE INSTRUCTION.
07553  017 153136  RTNCNT        RTN  PASS S3   S6         S3 <= COUNT  RETURN TO CALLER.
```

```
                   *******************************************************************
                   *      BIT INSTRUCTIONS
                   *******************************************************************
07554  220 074717  TBS      READ      INC  PNM  P
07555  301 165630           JSB                 INDBIT     GET MASK
07556  017 100157                     PASS L    TAB        L <= MASK.
07557  220 074457           READ      INC  M    P
07560  301 165630           JSB                 INDBIT     GET WORD TO BE TESTED
07561  015 101017                     AND  S1   TAB        LOGICAL AND OF MASK, WORD UNDER TEST
07562  013 041017                     XOR  S1   S1         S1 <= 0 IF ALL MASK BITS SET IN WORD
07563  320 067271           JMP  CNDX TBZ       SKIP       SKIP IF ALL MASK BITS SET IN WORD.
07564  000 075717                     INC  P    P          SKIP NEXT MACHINE INSTRUCTION
07565  000 075736  SKIP          RTN  INC  P    P          ADJUST P, JUMP TO FETCH ROUTINE
                   *******************************************************************
07566  220 074717  SBSCBS   READ      INC  PNM  P
07567  301 165630           JSB                 INDBIT     OBTAIN BIT MASK
07570  017 100157                     PASS L    TAB        L <= BIT MASK
07571  220 074457           READ      INC  M    P
07572  301 165630           JSB                 INDBIT     OBTAIN WORD TO BE OPERATED ON.
07573  327 170031           JMP  CNDX IR2       CBS        JUMP IF INSTRUCTION IS CBS
07574  017 001017                     IOR  S1   TAB        SET BITS IN WORD,PUT IN S1
07575  000 022461                MPCK INC  M    M          MEMORY PROTECT CHECK.
07576  177 140017           WRTE      PASS TAB  S1         REWRITE WORD TO MEMORY  RETURN TO FETCH.
07577  000 075736                RTN  INC  P    P
07600  013 101017  CBS                SANL S1   TAB        S1 <= MEMORY WORD WITH BITS CLEARED
07601  000 022461                MPCK INC  M    M
07602  177 140017           WRTE      PASS TAB  S1         REWRITE WORD TO MEMORY.
07603  000 075736                RTN  INC  P    P          RETURN TO FETCH ROUTINE.
                   *******************************************************************
                   *      WORD INSTRUCTIONS
                   *******************************************************************
07604  301 165170  CMW      JSB                 INITCM     INITIALIZE
07605  220 026457           READ      INC  M    A          READ FROM ARRAY A.
07606  017 100157                     PASS L    TAB        L <= WORD FROM ARRAY A
07607  220 024457           READ      INC  M    B          READ ADDRESS IN ARRAY B.
07610  000 024517                     INC  B    B          INCREMENT ARRAY B POINTER.
07611  003 001017                     SUB  S1   TAB        SUBTRACT ARRAY WORDS B - A.
07612  320 036431           JMP  CNDX TBZ  RJS  CHAL15     JUMP IF UNEQUAL.
07613  000 026557                     INC  A    A          INCREMENT ARRAY A POINTER
07614  007 145117                     DEC  S3   S3         DECREMENT COUNT.
07615  320 076731           JMP  CNDX TBZ       EXIT       JUMP TO EXIT IF COUNT IS ZERO.
07616  335 030271           JMP  CNDX INT  RJS  CMW+1      JUMP IF NOT INTERRUPTED.
07617  321 176130           JMP                 INTPEND
                   *******************************************************************
```

```
07620  301 165170  MVW      JSB                 INITCM    INITIALIZE.
07621  220 026457           READ      INC  M    A         READ FROM ARRAY A
07622  000 026557                     INC  A    A         INCREMENT ARRAY A POINTER.
07623  017 101017                     PASS S1   TAB       S1 <= CONTENTS OF WORD OF ARRAY A
07624  000 024457                     INC  M    B         M <= ADDRESS FROM ARRAY B
07625  017 122761                MPCK PASS      M         MEMORY PROTECT CHECK-- BIT 15 LOW.
07626  177 140017           WRTE      PASS TAB  S1        WRITE WORD INTO ARRAY B.
07627  000 024517                     INC  B    B         ADVANCE ARRAY B POINTER.
07630  007 145117                     DEC  S3   S3        DECREMENT COUNT
07631  320 076731           JMP  CNDX TBZ       EXIT      EXIT IF COUNT IS ZERO.
07632  335 031071           JMP  CNDX INT  RJS  MVW+1     JUMP IF NOT INTERRUPTED.
07633  321 176130           JMP                 INTPEND




                   ********************************************************************************
                   *        BYTE INSTRUCTIONS
                   ********************************************************************************
07634  340 000157  SBT      IMM       HIGH L    %000      L <= 000377B.
07635  015 127017  STBYTE             AND  S1   A         S1 <= RIGHT BYTE OF A REG.
07636  157 125244           LWF  R1   PASS S6   B         S6 <= WORD ADDRESS. FLAG SET IF BYTE ODD
07637  220 052461           READ MPCK INC  M    S6        READ WORD ADDRESS,CHECK FOR MP VIOLATION
07640  324 032171           JMP  CNDX FLAG RJS  STEVEN    JUMP IF STORE TO EVEN BYTE.
07641  013 101417                     SANL S9   TAB       MASK OUT EVEN BYTE OF MEMORY WORD.
07642  321 172330           JMP                 MERGE
07643  015 101417  STEVEN             AND  S9   TAB       MASK OUT ODD BYTE OF MEMORY WORD.
07644  017 141003                L4   PASS S1   S1        EXCHANGE BYTES IN REGISTER CONTAINING
07645  017 141003                L4   PASS S1   S1        BYTE TO BE STORED.
07646  017 160157  MERGE              PASS L    S9        L <= MEMORY WORD WITH TARGET BYTE CLEARED
07647  017 041017                     IOR  S1   S1        S1 <= WORD WITH BYTES MERGED.
07650  000 024517                     INC  B    B         INCREMENT BYTE ADDRESS.
07651  177 140036           WRTE RTN  PASS TAB  S1        WRITE NEW WORD BACK INTO WORD ADDRESS.
                   ********************************************************************************
07652  017 125057  LBT                PASS S2   B         S2 <= BYTE ADDRESS.
07653  000 024517                     INC  B    B         INCREMENT BYTE ADDRESS FOR NEXT INSTRUCT.
07654  340 000151  LDBYTE   IMM  CLFL HIGH L    %000      L <= 000377B.  CLEAR CPU FLAG.
07655  157 143244           LWF  R1   PASS S6   S2        S6 <= WORD ADDRESS OF BYTE. SET FLAG IF O
07656  220 052457           READ      INC  M    S6        ODD BYTE.  READ WORD ADDRESS.
07657  324 073171           JMP  CNDX FLAG      LODD      JUMP IF BYTE IS ODD.
07660  013 100543                L4   SANL A    TAB       MASK OUT EVEN BYTE AND MOVE ODD BYTE
07661  017 126543                L4   PASS A    A         TO EVEN BYTE OF A REG.
07662  017 136776                RTN                      RETURN TO CALLER OR FETCH.
07663  015 100576  LODD          RTN  AND  A    TAB       MASK OUT EVEN BYTE,LOAD INTO A. RETURN.
                   ********************************************************************************
07664  301 165170  MBT      JSB                 INITCM    INITIALIZE.
07665  017 127057                     PASS S2   A         S2 <= ADDRESS START OF ARRAY A.
07666  301 172630           JSB                 LDBYTE    LOAD BYTE FROM ARRAY A.
07667  000 043051                CLFL INC  S2   S2        RESET FLAG, S2 <= NEXT BYTE ADDR IN ARRAY
07670  301 171670           JSB                 STBYTE    STORE BYTE INTO BYTE ADDRESS IN B REG.
07671  007 145117                     DEC  S3   S3        DECREMENT COUNT
07672  320 033671           JMP  CNDX TBZ  RJS  NOTDAN    JUMP IF COUNT NOT ZERO.
07673  017 142557                     PASS A    S2        A <= 1 + LAST ARRAY A BYTE ADDRESS MOVED
07674  321 176730           JMP                 EXIT
07675  335 033331  NOTDAN   JMP  CNDX INT  RJS  MBT+2     JUMP IF NOT INTERRUPTED.
07676  017 142557                     PASS A    S2        A <= 1 + LAST ARRAY A ADDRESS MOVED
07677  321 176130           JMP                 INTPEND
                   ********************************************************************************
07700  340 000157  SFB      IMM       HIGH L    %000      L <= 000377B
07701  015 127117                     AND  S3   A         S3 <= TEST BYTE IN LOW-ORDER BYTE
07702  013 127143                L4   SANL S4   A         S4 <= TERMINATION BYTE IN
07703  017 147143                L4   PASS S4   S4        LOW-ORDER BYTE
07704  017 127357                     PASS S8   A         S8 <= SAVE ORIGINAL CONTENTS OF A REG.
07705  301 172530  CONTSFB  JSB                 LBT       LOAD BYTE INTO A REG FROM ADDRESS IN B.
07706  017 126157                     PASS L    A         L <= BYTE TO TESTED IN LOW BYTE.
07707  017 156557                     PASS A    S8        RESTORE ORIGINAL CONTENTS OF A REG
07710  013 045257                     XOR  S6   S3        COMPARE BYTE TO TEST BYTE.
07711  320 034571           JMP  CNDX TBZ  RJS  NOMATCH   JUMP IF UNEQUAL.
07712  007 124536                RTN  DEC  B    B         B <= BYTE ADDRESS OF MATCH. GO TO FETCH
07713  013 047017  NOMATCH            XOR  S1   S4        COMPARE BYTE TO TERMINATION BYTE.
07714  320 034771           JMP  CNDX TBZ  RJS  INTTST    JUMP IF UNEQUAL.
07715  000 075736                RTN  INC  P    P         SKIP NEXT MACHINE INSTRUCTION AND FETCH.
07716  377 177777           ONES
07717  335 034271  INTTST   JMP  CNDX INT  RJS  CONTSFB   JUMP IF NOT INTERRUPTED.
07720  007 175736                RTN  DEC  P    P         P <= INSTRUCTION ADDRESS, GO TO FETCH.
07721  377 177777           ONES
                   ********************************************************************************
```

```
07722  301 165170  CBT      JSB                 INITCM    INITIALIZE.
07723  017 127357                     PASS S8   A         S8 <= POINTER FOR ARRAY A.
07724  017 157057                     PASS S2   S8        S2 <= NEXT BYTE ADDRESS IN ARRAY A.
07725  301 172630           JSB                 LDBYTE    LOAD ARRAY A BYTE INTO A REG.
07726  017 127417                     PASS S9   A         S9 <= ARRAY A BYTE.
07727  000 043357                     INC  S8   S2        INCREMENT ARRAY A POINTER.
07730  301 172530           JSB                 LBT       A <= BYTE FROM ARRAY B (ADDRESS IN B REG)
07731  017 160157                     PASS L    S9        L <= BYTE FROM ARRAY A
07732  003 027017                     SUB  S1   A         SUBTRACT BYTE FROM ARRAY B - A.
07733  320 036331           JMP  CNDX TBZ  RJS  CHAL15B   JUMP IF BYTES NOT EQUAL.
07734  007 145117                     DEC  S3   S3        DECREMENT THE COUNT.
07735  320 036031           JMP  CNDX TBZ  RJS  CONTCBT   JUMP IF COUNT IS NOT ZERO.
07736  017 156557                     PASS A    S8        EQUAL EXIT... A <= 1+LAST MOVED BYTE ADDR
07737  321 176730           JMP                 EXIT
07740  335 035231  CONTCBT  JMP  CNDX INT  RJS  CBT+2     JUMP IF NOT INTERRUPTED.
07741  017 156557                     PASS A    S8        A <= NEXT BYTE ADDRESS OF ARRAY A TO TEST
```

```
                   *****************************************************************************
                   *        COMMON ROUTINES TO MOVE, COMPARE INSTRUCTIONS
                   *****************************************************************************
                   *                                      INTERRUPT EXIT
07742  000 050457  INTPEND            INC  M    S5        M <= ADDRESS OF WORD 3
07743  177 144017           WRTE      PASS TAB  S3        WRITE REMAINING COUNT INTO WORD 3.
07744  007 175717                     DEC  P    P
07745  007 175736                RTN  DEC  P    P         P <= ADDRESS OF WORD 1, GO TO FETCH.
                   *******************************************************************
                   *                                      EXIT TESTS FOR CBT,CMW
07746  007 156557  CHAL15B            DEC  A    S8        A <= BYTE ADDRESS OF MISMATCH.
07747  017 141017                     PASS S1   S1        CHECK RESULT OF COMPARE
07750  322 036531  CHAL15   JMP  CNDX AL15 RJS  SKIP1     JUMP IF SIGN BIT IS ZERO.
07751  000 047157                     INC  S4   S4        SKIP ONE MACHINE INSTRUCTION.
07752  000 047157  SKIP1              INC  S4   S4        SKIP ONE MACHINE INSTRUCTION.
07753  007 145117                     DEC  S3   S3        DECREMENT THE COUNT.
07754  017 144157                     PASS L    S3        L <= COUNT REMAINING
07755  004 124517                     ADD  B    B         B <= FIRST ADDR. IN ARRAY B + COUNT.
                   *******************************************************************
                   *                                      COMPLETION EXIT
07756  000 050457  EXIT               INC  M    S5        M <= ADDRESS OF WORD 3
07757  177 154017           WRTE      PASS TAB  S7        WORD 3 <= ZERO.
07760  017 147736                RTN  PASS P    S4        P <= NEXT MACHINE INSTR. TO EXECUTE.FETCH
                   *****************************************************************************
                   *     JUMP INSTRUCTIONS
                   *****************************************************************************
07761  220 074717  JLY      READ      INC  PNM  P         READ ADDRESS OF WORD 2
07762  301 165630           JSB                 INDBIT    CHECK FOR INDIRECT,M<= DESTINATION ADDR.
07763  340 120417           IMM       HIGH IR   %050      MACHINE JMP INTO IR TO SET LOW MP BOUNDS
07764  017 122461                MPCK PASS M    M         DO MP CHECK ON JUMP TARGET ADDRESS.
07765  017 175657                     PASS Y    P         Y <= ADDRESS OF FOLLOWING MACHINE INSTR.
07766  017 123736                RTN  PASS P    M         P <= DESTINATION ADDRESS, JUMP TO FETCH.
                   *****************************************************************************
07767  220 074717  JPY      READ      INC  PNM  P         READ ADDRESS OF WORD 2.
07770  017 172157                     PASS L    Y         L <= INDEX REG. Y.
07771  004 101017                     ADD  S1   TAB       S1 <= INDEXED JUMP ADDRESS.
07772  340 120417           IMM       HIGH IR   %050      MACHINE JMP INTO IR TO SET LOW MP BOUNDS
07773  017 140457                     PASS M    S1        M<= INDEXED ADDRESS, WITH BIT 15 LOW
07774  017 122761                MPCK PASS      M         MP CHECK ON 15-BIT DESTINATION ADDRESS.
07775  017 123736                RTN  PASS P    M         P <= DESTINATION ADDRESS. GO TO FETCH.
                   *****************************************************************************
07776  377 177777           ONES
07777  377 177777           ONES
                            END
```

INDEX

HEWLETT hp PACKARD

SALES & SERVICE OFFICES

AFRICA, ASIA, AUSTRALIA

ANGOLA
Telectra
Empresa Técnica de
Equipamentos
Eléctricos, S.A.R.L.
R. Barbosa Rodrigues, 42-I°DT.
Caixa Postal, 6487
Luanda
Tel: 35515/6
Cable: TELECTRA Luanda

AUSTRALIA
Hewlett-Packard Australia
Pty. Ltd.
31-41 Joseph Street
Blackburn, Victoria 3130
P.O. Box 36
Doncaster East, Victoria 3109
Tel: 89-6351
Telex: 31-024
Cable: HEWPARD Melbourne

Hewlett-Packard Australia
Pty. Ltd.
31 Bridge Street
Pymble
New South Wales, 2073
Tel: 449-6566
Telex: 21561
Cable: HEWPARD Sydney

Hewlett-Packard Australia
Pty. Ltd.
153 Greenhill Road
Parkside, S.A., 5063
Tel: 272-5911
Telex: 82536 ADEL
Cable: HEWPARD ADELAID

Hewlett-Packard Australia
Pty.Ltd.
141 Stirling Highway
Nedlands, W.A. 6009
Tel: 86-5455
Telex: 93859 PERTH
Cable: HEWPARD PERTH

Hewlett-Packard Australia
Pty. Ltd.
121 Wollongong Street
Fyshwick, A.C.T. 2609
Tel: 95-2733
Telex: 62650 Canberra
Cable: HEWPARD CANBERRA

Hewlett Packard Australia
Pty. Ltd.
5th Floor
Teachers Union Building
495-499 Boundary Street
Spring Hill, 4000 Queensland
Tel: 229-1544
Cable: HEWPARD Brisbane

GUAM
Medical/Pocket Calculators Only
Guam Medical Supply, Inc.
Jay Ease Building, Room 210
P.O. Box 8947
Tamuning 96911
Tel: 646-4513
Cable: EARMED Guam

HONG KONG
Schmidt & Co.(Hong Kong) Ltd.
P.O. Box 297
Connalight Centre
39th Floor
Connaught Road, Central
Hong Kong
Tel: H-255291-5
Telex: 74766 SCHMC HX
Cable: SCHMIDTCO Hong Kong

INDIA
Blue Star Ltd.
Kasturi Buildings
Jamshedji Tata Rd
Bombay 400 020
Tel: 29 50 21
Telex: 001-2156
Cable BLUEFROST

Blue Star Ltd.
Sahas
414/2 Vir Savarkar Marg
Prabhadevi
Bombay 400 025
Tel: 45 78 87
Telex: 011-4093
Cable: FROSTBLUE

Blue Star Ltd.
Band Box House
Prabhadevi
Bombay 400 025
Tel: 45 73 01
Telex: 011-3751
Cable: BLUESTAR

Blue Star Ltd.
14/40 Civil Lines
Kanpur 208 001
Tel: 6 88 82
Telex: 292
Cable: BLUESTAR

Blue Star Ltd.
7 Hare Street
P.O. Box 506
Calcutta 700 001
Tel: 23-0131
Telex: 021-7655
Cable: BLUESTAR

Blue Star Ltd.
7th & 8th Floor
Bhandari House
91 Nehru Place
New Delhi 110024
Tel: 634770 & 635166
Telex: 031-2463
Cable: BLUESTAR

Blue Star Ltd.
Blue Star House
11/11A Magarath Road
Bangalore 560 025
Tel: 55668
Telex: 043-430
Cable: BLUESTAR

Blue Star Ltd.
Meeakshi Mandiran
xxx/1678 Mahatma Gandhi Rd.
Cochin 682 016
Tel: 32069,32161,32282
Telex: 0885-514
Cable: BLUESTAR

Blue Star Ltd.
1-1-117/1
Sarojini Devi Road
Secunderabad 500 003
Tel: 70126, 70127
Cable: BLUEFROST
Telex: 015-459

Blue Star Ltd.
2/34 Kodambakkam High Road
Madras 600034
Tel: 82056
Telex: 041-379
Cable: BLUESTAR

Blue Star Ltd.
Nataraj Mansions
2nd Floor Bistupur
Jamshedpur 831 001
Tel: 7383
Cable: BLUESTAR
Telex: 240

INDONESIA
BERCA Indonesia P.T.
P.O. Box 496/Jkt.
JLN●Abdul Muis 62
Jakarta
Tel: 40369, 49886,49255,356038
JKT.42895
Cable: BERCACON

BERCA Indonesia P.t.
63 JL. Raya Gubeng
Surabaya
Tel: 44309

ISRAEL
Electronics & Engineering Div.
of Motorola Israel Ltd.
17, Kremenetski Street
P.O. Box 25016
Tel-Aviv
Tel: 38973
Telex: 33569
Cable: BASTEL Tel-Aviv

JAPAN
Yokogawa-Hewlett-Packard Ltd.
Ohashi Building
59-1 Yoyogi 1-Chome
Shibuya-ku, **Tokyo** 151
Tel: 03-370-2281/92
Telex: 232-2024YHP
Cable: YHPMARKET TOK 23-724

Yokogawa-Hewlett-Packard Ltd.
Chuo Bldg., 4th Floor
4-20, Nishinakajima 5-chome
Yodogawa-ku, Osaka-shi
Osaka,532
Tel: 06-304-6021

Yokogawa-Hewlett-Packard Ltd.
Nakamo Building
24 Kami Sasajima-cho
Nakamura-ku, **Nagoya** , 450
Tel: (052) 571-5171

Yokogawa-Hewlett-Packard Ltd.
Tanigawa Building
2-24-1 Tsuruya-cho
Kanagawa-ku
Yokohama, 221
Tel: 045-312-1252
Telex: 382-3204 YHP YOK

Yokogawa-Hewlett-Packard Ltd.
Mito Mitsu Building
105, Chome-1,San-no-maru
Mito, Ibaragi 310
Tel:: 0292-25-7470

Yokogawa-Hewlett-Packard Ltd.
Inoue Building
1348-3, Asahi-cho, 1-chome
Atsugi, Kanagawa 243
Tel: 0462-24-0452

Yokogawa-Hewlett-Packard Ltd.
Kumagaya Asahi
Hachijuni Building
4th Floor
3-4, Tsukuba
Kumagaya, Saitama 360
Tel: 0485-24-6563

KENYA
Technical Engineering
Services(E.A.)Ltd.,
P.O. Box 18311
Nairobi
Tel: 557726/556762
Cable: PROTON

Medical Only
International Aeradio(E.A.)Ltd.,
P.O. Box 19012
Nairobi Airport
Nairobi
Tel: 336055/56
Telex: 22201/22301
Cable: INTAERIO Nairobi

KOREA
Samsung Electronics Co., Ltd.
20th Fl. Dongbang Bldg. 250, 2-KA
C.P.O. Box 2775
Taepyung-Ro, Chung-Ku
Seoul
Tel: (23) 6811
Telex: 22575
Cable: ELEKSTAR Seoul

MALAYSIA
Teknik Mutu Sdn. Bhd.
2 Lorong 13/6A
Section 13
Petaling Jaya,**Selangor**
Tel: 54994/54916
Telex: MA 37605

Protel Engineering
P.O. Box 1917
Lot 259, Satok Road
Kuching, **Sarawak**
Tel: 2400
Cable: PROTEL ENG

MOZAMBIQUE
A.N. Goncalves, Lta.
162, 1 Apt. 14 Av. D. Luis
Caixa Postal 107
Lourenco Marques
Tel: 27091, 27114
Telex: 6-203 NEGON Mo
Cable: NEGON

NEW ZEALAND
Hewlett-Packard (N.Z.) Ltd.
P.O. Box 9443
Courtenay Place
Wellington
Tel: 877-199
Telex: NZ 3839
Cable: HEWPACK Wellington

Hewlett-Packard (N.Z.) Ltd.
Pakuranga Professional Centre
267 Pakuranga Highway
Box 51092
Pakuranga
Tel: 569-651
Cable: HEWPACK,Auckland

Analytical/Medical Only
Medical Supplies N.Z. Ltd.
Scientific Division
79 Carlton Gore Rd., Newmarket
P.O. Box 1234
Auckland
Tel: 75-289
Cable:DENTAL Auckland

Analytical/Medical Only
Medical Supplies N.Z. Ltd.
P.O. Box 1994
147-161 Tory St.
Wellington
Tel: 850-799
Telex: 3858
Cable: DENTAL, Wellington

Analytical/Medical Only
Medical Supplies N.Z. Ltd.
P.O. Box 309
239 Stanmore Road
Christchurch
Tel: 892-019
Cable: DENTAL, Christchurch

Analytical/Medical Only
Medical Supplies N.Z. Ltd.
303 Great King Street
P.O. Box 233
Dunedin
Tel: 88-817
Cable: DENTAL, Dunedin

NIGERIA
The Electronics
Instrumentations Ltd.
N6B/770 Oyo Road
Oluseun House
P.M.B. 5402
Ibadan
Tel: 61577
Telex: 31231 TEIL Nigeria
Cable: THETEIL Ibadan

The Electronics Instrumenta-
tions Ltd.
144 Agege Motor Road, Mushin
P.O. Box 6645
Lagos
Cable: THETEIL Lagos

PAKISTAN
Mushko & Company, Ltd.
Oosman Chambers
Abdullah Haroon Road
Karachi-3
Tel: 511027, 512927
Telex: 2894
Cable: COOPERATOR Karachi

Mushko & Company, Ltd.
38B, Satellite Town
Rawalpindi
Tel: 41924
Cable: FEMUS Rawalpindi

PHILIPPINES
The Online Advanced
Systems Corporation
8th Floor, Filcapital Bldg.
Ayala Avenue
Makati, Metro**Manila**
Tel: 85-35-81, 85-34-91
Telex: 3274 ONLINE

RHODESIA
Field Technical Sales
45 Kelvin Road North
P.O. Box 3458
Salisbury
Tel: 705231 (5 lines)
Telex: RH 4122

SINGAPORE
Hewlett-Packard Singapore
(Pte.) Ltd.
1150 Depot Road
Alexandra P.O. Box 58
Singapore 4
Tel: 270-2355
Telex: HPSG RS 21486
Cable: HEWPACK, Singapore

SOUTH AFRICA
Hewlett-Packard South Africa
(Pty.), Ltd.
Private Bag Wendywood
Sandton, Transvaal 2144
Hewlett-Packard Centre
Daphne Street, Wendywood,
Sandton, Transvaal 2144
Tel: 802-10408
Telex: 8-4782
Cable: HEWPACK JOHANNESBURG

Service Department
Hewlett-Packard South Africa
(Pty.), Ltd.
P.O. Box 39325
Gramley, Sandton, 2018
451 Wynberg Extension 3,
Sandton, 2001
Tel: 636-8188/9
Telex: 8-2391

Hewlett-Packard South Africa
(Pty.), Ltd.
P.O. Box 120
Howard Place, Cape Province, 7450
Pine Park Centre, Forest Drive,
Pinelands, Cape Province, 7405
Tel: 53-7955 thu 9
Telex: 57-0006

Service Department
Hewlett-Packard South Africa
(Pty.), Ltd.
P.O. Box 37099
Overport, Durban 4067
Braby House
641 Ridge Road
Durban, 4001
Tel: 88-7478
Telex: 6-7954

TAIWAN
Hewlett-Packard Far East Ltd.,
Taiwan Branch
39 Chung Hsiao West Road
Sec. 1, 7th Floor
Taipei
Tel: 3819160-4
Cable: HEWPACK TAIPEI

Hewlett-Packard Far East Ltd.
Taiwan Branch
68-2, Chung Cheng 3rd. Road
Kaohsiung
Tel: (07) 242318-Kaohsiung

Analytical Only
San Kwang Instruments Co., Ltd.,
No. 20, Yung Sui Road
Taipei
Tel: 3715171-4 (5 lines)
Telex: 22894 SANKWANG
Cable: SANKWANG TAIPEI

TANZANIA
Medical Only
International Aeradio (E.A.), Ltd.
P.O. Box 861
Dar es Salaam
Tel: 21251 Ext. 265
Telex: 41030

THAILAND
UNIMESA Co., Ltd.
Elcom Research Building
2538 Sukumvit Ave
Bangkok
Tel: 3932387, 3930338
Cable: UNIMESA Bangkok

UGANDA
Medical Only
International Aeradio(E.A.), Ltd.,
P.O. Box 2577
Kampala
Tel: 54388
Cable: INTAERIO Kampala

ZAMBIA
R.J. Tilbury (Zambia) Ltd.
P.O. Box 2792
Lusaka
Tel: 73793
Cable: ARJAYTEE, Lusaka

OTHER AREAS NOT LISTED, CONTACT:
Hewlett-Packard Intercontinental
3200 Hillview Ave.
Palo Alto, California 94304
Tel: (415) 493-1501
TWX: 910-373-1267
Cable: HEWPACK Palo Alto
Telex: 034-8300, 034-8493

CANADA

ALBERTA
Hewlett-Packard (Canada) Ltd.
11620A - 168th Street
EdmontonT5M 3T9
Tel: (403) 452-3670
TWX: 610-831-2431 EDTH

Hewlett-Packard (Canada) Ltd.
210,7220 Fisher St. S.E.
Calgary T2H 2H8
Tel: (403) 253-2713
Twx: 610-821-6141

BRITISH COLUMBIA
Hewlett-Packard (Canada) Ltd.
837 E. Cordova Street
Vancouver V6A 3R2
Tel: (604) 254-0531
TWX: 610-922-5059 VCR

MANITOBA
Hewlett-Packard (Canada) Ltd.
513 Century St.
St. James
Winnipeg R3H 0L8
Tel: (204) 786-7581
TWX: 610-671-3531

NOVA SCOTIA
Hewlett-Packard (Canada) Ltd.
800 Windmill Road
Dartmouth B3B 1L1
Tel: (902) 469-7820
TWX: 610-271-4482 HFX

ONTARIO
Hewlett-Packard (Canada) Ltd.
1020 Morrison Dr.
Ottawa K2H 8K7
Tel: (613) 820-6483
TWX: 610-563-1636

Hewlett-Packard (Canada) Ltd.
6877 Goreway Drive
Mississauga L4V 1M8
Tel: (416) 678-9430
TWX: 610-492-4246

QUEBEC
Hewlett-Packard (Canada) Ltd.
275 Hymus Blvd.
Pointe Claire H9R 1G7
Tel: (514) 697-4232
TWX: 610-422-3022
TLX: 05-821521 HPCL

FOR CANADIAN AREAS NOT LISTED:
Contact Hewlett-Packard (Canada)
Ltd. in Mississauga.

CENTRAL AND SOUTH AMERICA

ARGENTINA
Hewlett-Packard Argentina
S.A.
Av. Leandro N. Alem 822 - 12
1001**Buenos Aires**
Tel: 31-6063,4,5,6 and 7
Telex: Public Booth N°9
Cable: HEWPACK ARG

BOLIVIA
Casa Kavlin S.A.
Calle Potosi' 1130
P.O. Box 500
La Paz
Tel: 41530,53221
Telex: CWC BX 5298,ITT 3560082
Cable: KAVLIN

BRAZIL
Hewlett-Packard do Brasil
I.e.C. Ltda.
Avenida Rio Negro, 980
Alphaville
06400**Barueri** SP
Tel: 429-2148/9;429-2118/9

Hewlett-Packard do Brasil
I.e.C. Ltda.
Rua Padre Chagas, 32
90000-**Pôrto Alegre**-RS
Tel: (0512) 22-2998, 22-5621
Cable: HEWPACK Pôrto Alegre

Hewlett-Packard do Brasil
I.E.C. Ltda.
Rua Siqueira Campos, 53
Copacabana
20000-**Rio de Janeiro**
Tel: 257-80-94-DDD (021)
Telex: 391-212-1905 HEWP-BR
Cable: HEWPACK
Rio de Janeiro

CHILE
Calcagni y Metcalfe Ltda.
Alameda 580-Of. 807
Casilla 2118
Santiago, 1
Tel: 398613
Telex: 3520001 CALMET
Cable: CALMET Santiago

COLOMBIA
Instrumentación
Henrik A. Langebaek & Kier S.A.
Carrera 7 No. 48-75
Apartado Aéreo 6287
Bogotá, I.D.E.
Tel: 69-88-77
Cable: AARIS Bogotá
Telex: 044-400

COSTA RICA
Cientifica Costarricense S.A.
Calle Central, Avenidas 1 y 3
Apartado 10159
San José
Tel: 21-86-13
Cable: GALGUR San José

ECUADOR
Medical Only
A.F. Vizcaino Compañia Ltda.
Av. Rio Amazonas No. 239
P.O. Box 2925
Quito
Tel: 242-150,247-033/034
Cable: Astor Quito

Calculators Only
Computadoras y Equipos
Electrónicos
P.O. Box 6423 CCI
Eloy Alfaro #1824,3 Piso
Quito
Tel: 453482
Telex: 02-2113 Sagita Ed
Cable: Sagita-Quito

EL SALVADOR
Instrumentacion y Procesamiento
Electronico de el Salvador
Bulevar de los Heroes II-48
San Salvador
Tel: 252787

GUATEMALA
IPESA
Avenida La Reforma 3-48,
Zona 9
Guatemala City
Tel: 63627, 64786
Telex: 4192 Teletro Gu

MEXICO
Hewlett-Packard Mexicana,
S.A. de C.V.
Torres Adalid No. 21, 11 Piso
Col. del Valle
Mexico 12, D.F.
Tel: (905) 543-42-32
Telex: 017-74-507

Hewlett-Packard Mexicana,
S.A. de C.V.
Ave. Constitución No. 2184
Monterrey, N.L.
Tel: 48-71-32, 48-71-84
Telex: 038-843

NICARAGUA
Roberto Terán G.
Apartado Postal 689
Edificio Terán
Managua
Tel: 25114, 23412,23454
Cable: ROTERAN Managua
Calculators Only
Cientifice Costarricewse S.A.
Ciudad Jardin D-1
Managua
Tel: 24108

PANAMA
Electrónico Balboa, S.A.
P.O. Box 4929
Calle Samuel Lewis
Cuidad de Panama
Tel: 64-2700
Telex: 3431103 Curunda,
Canal Zone
Cable: ELECTRON Panama

PARAGUAY
Z.J. Melamed S.R.L.
División: Aparatos y Equipos
Médicos
División: Aparatos y Equipos
Cientificos y de Investigación
P.O.Box 676
Chile-482, Edificio Victoria
Asunción
Tel: 91-271, 91-272
Cable: RAMEL

PERU
Compañia Electro Médica S.A.
Los Flamencos 145
San Isidro Casilla 1030
Lima 1
Tel: 41-4325
Cable: ELMED Lima

PUERTO RICO
Hewlett-Packard Inter-Americas
Puerto Rico Branch Office
Calle 272,
No. 203 Urb. Country Club
Carolina 00924
Tel: (809) 762-7255
Telex: 345 0514

URUGUAY
Pablo Ferrando S.A.
Comercial e Industrial
Avenida Italia 2877
Casilla de Correo 370
Montevideo
Tel: 40-3102
Cable: RADIUM Montevideo

VENEZUELA
Hewlett-Packard de Venezuela
C.A.
P.O. Box 50933
Caracas 105
Los Ruices Norte
3a Transversal
Edificio Segre
Caracas 107
Tel: 35-00-11 (20 lines)
Telex: 25146 HEWPACK
Cable: HEWPACK Caracas

FOR AREAS NOT LISTED, CONTACT:
Hewlett-Packard
Inter-Americas
3200 Hillview Ave.
Palo Alto, California 94304
Tel: (415) 493-1501
TWX: 910-373-1260
Cable: HEWPACK Palo Alto
Telex: 034-8300, 034-8493

EUROPE, NORTH AFRICA AND MIDDLE EAST

AUSTRIA
Hewlett-Packard Ges.m.b.H
Handelskai 52
P.O. box 7
A-1205 **Vienna**
Tel: (0222) 351621 to 27
cable: HEWPAK Vienna
Telex: 75923 hewpak a

BELGIUM
Hewlett-Packard Benelux
S.A./N.V.
Avenue de Col-Vert. 1.
(Groenkraaglaan)
B-1170 **Brussels**
Tel: (02) 672 22 40
Cable: PALOBEN Brussels
Telex: 23 494 paloben bru

CYPRUS
Kypronics
19, Gregorios & Xenopoulos Rd.
P.O. Box 1152
CY-**Nicosia**
Tel: 45628/29
Cable: KYPRONICS PANDEHIS
Telex: 3018

CZECHOSLOVAKIA
Vyvojova a Provozni Zakladna
Vyzkumnych Ustavu v Bechovicich
CSSR-25097 **Bechovice u Prahy**
Tel: 89 93 41
Telex: 121333

Institute of Medical Bionics
Vyskumny Ustav Lekarskej Bioniky
Jedlova 6
CS-88346 **Bratislava-Kramare**
Tel: 44-551/45-541

DDR
Entwicklungslabor der TU Dresden
Forschungsinstitut Meinsberg
DDR-7305
Waldheim/Meinsberg
Tel: 37 667
Telex: 112145

Export Contact AG Zuerich
Guenther Forgber
Schlegelstrasse 15
1040 **Berlin**
Tel: 42-74-12
Telex: 111889

DENMARK
Hewlett-packard A/S
Datavej 52
DK-3460 **Birkerød**
Tel: (02) 81 66 40
Cable: HEWPACK AS
Telex: 166 40 hpas

Hewlett-Packard A/S
Navervej 1
DK-8600 **Silkeborg**
Tel: (06) 82 71 66
Telex: 166 40 hpas
Cable: HEWPACK AS

FINLAND
Hewlett-Packard OY
Nahkahousuntie 5
P.O. Box 6
SF-00211 **Helsinki** 21
Tel: (90) 6923031
Cable: HEWPACKOY Helsinki
Telex: 12-1563 HEWPA SF

FRANCE
Hewlett-Packard France
Quartier de Courtaboeuf
Boite Postale No. 6
F-91401 **Orsay** Cédex
Tel: (1) 907 78 25
Cable: HEWPACK Orsay
Telex: 600048

Hewlett-Packard France
Agency Régionale
"Le Saquin"
Chemin des Mouilles
B.P. 162
F-69130 **Ecully**
Tel: (78) 33 81 25.
Cable: HEWPACK Eculy
Telex: 31 06 17

Hewlett-Packard France
Agence Régionale
Péricentre de la Cépière
Chemin de la Cépière, 20
F-31300 **Toulouse-Le Mirail**
Tel:(61) 40 11 12
Cable: HEWPACK 51957
Telex 510957

Hewlett-Packard France
Agence Régionale
Aéroport principal de
Marseille-Marignane
F-13721**Marignane**
Tel: (91) 89 12 36
Cable: HEWPACK MARGN
Telex 410770

Hewlett-Packard France
Agence Régionale
63. Avenue de Rochester
B.P. 1124
F-35014 **Rennes** Cédex
Tel: (99) 36 33 21
Cable: HEWPACK 74912
Telex 740912

Hewlett-Packard France
Agence Régionale
74. Allée de la Robertsau
F-67000 **Strasbourg**
Tel: (88) 35 23 20/21
Telex. 890141
Cable: HEWPACK STRBG

Hewlett-Packard France
Agence Régionale
Centre Vauban
201, rue Colbert
Entrée A2
F-59000 **Lille**
Tel: (20) 51 44 14
Telex. 820744

Hewlett-Packard France
Centre d' Affaires Paris-Nord
Bâtiment Ampère
Rue de La Commune de Paris
B.P. 300
F-93153 **Le Blanc Mesnil Cédex**
Tel: (01) 931 88 50

GERMAN FEDERAL REPUBLIC
Hewlett-Packard GmbH
Vertriebszentrale Frankfurt
Bernerstrasse 117
Postfach 560 140
D-6000 **Frankfurt** 56
Tel: (0611) 50 04-1
Cable: HEWPACKSA Frankfurt
Tel: (0611) 50 04-1
Cable: HEWPACKSA Frankfurt
Telex: 04 13249 hpffmd

Hewlett-Packard GmbH
Technisches Buero Böblingen
Herrenbergerstrasse 110
D-7030 **Böblingen**, Württemberg
Tel: (07031) 667-1
Cable: HEPAK Böblingen
Telex: 07265739 bbn

Hewlett-Packard GmbH
Technisches Buero Düsseldorf
Emanuel-Leutze-Str.1 (Seestern)
D-4000 **Düsseldorf** 11
Tel: (0211) 59711
Telex: 085/86 533 hpdd d

Hewlett-Packard GmbH
Technisches Buero Hamburg
Wendenstrasse 23
D-2000 **Hamburg** 1
Tel: (040) 24 13 93
Cable: HEWPACKSA Hamburg
Telex: 21 63 032 hphh d

Hewlett-Packard GmbH
Technisches Buero Hannover
Am Grossmarkt 6
D-3000 **Hannover-Kleefeld 91**
Tel: (0511) 46 60 01
Telex: 092 3259

Hewlett-Packard GmbH
Werk Groetzingen
Ohmstrasse 6
D-7500 **Karlsruke 41**
Tel: (0721) 69 40 06
Telex: 07-825707

Hewlett-Packard GmbH
Technisches Buero Nuremberg
Neumeyer Str. 90
D-8500 **Nuremberg**
Tel: (0911) 56 30 83/85
Telex: 0623 860

Hewlett-Packard GmbH
Technisches Buero Munchen
Unterhachinger Strasse 28
ISAR Center
D-8012 **Ottobrunn**
Tel: (089) 601 30 61/7
Cable HEWPACKSA Munchen
Telex. 0524985

Hewlett-Packard GmbH
Technisches Buero Berlin
Keith Strasse 2-4
D-1000 **Berlin** 30
Tel. (030) 24 90 86
Telex 18 3405 hpbln d

GREECE
Kostas Karayannis
18, Ermou Street
GR-**Athens** 126
Tel: 3237731
Cable: RAKAR Athens
Telex: 21 59 62 rkar gr

Analytical Only
"INTECO" G. Papathanassiou & Co
Marni 17
GR - **Athens** 103
Tel: 522 1915
Cable: INTEKNIKA Athens
Telex: 21 5329 INTE GR

Medical Only
Technomed Hellas Ltd.
52.Skoufa Street
GR - **Athens** 135
Tel: 362 6972. 363 3830
Cable:etalak athens
Telex: 21-4693 ETAL GR

HUNGARY
MTA
Müszerügyi és Méréstechnikai
Szolgalata
Lenin Krt. 67
Tel: 42 03 38
Telex. 22 51 14

ICELAND
Medical Only
Elding Trading Company Inc
Hafnarhvoli - Tryggvatotu
IS-**Reykjavik**
Tel: 1 58 20
Cable: ELDING Reykjavik

IRAN
Hewlett-Packard Iran Ltd.
No 13, Fourteenth St.
Miremad Avenue
P.O. Box 41/2419
IR-**Tehran**
Tel: 851082-7
Telex: 213405 HEWP IR

IRAQ
Hewlett-Packard Trading Co.
Mansoor City
Baghdad
Tel: 5517827
Telex: 2455 Hepairaq ik
Cable: HEWPACDAD.
Baghdad Iraq

IRELAND
Hewlett-Packard Ltd.
King Street Lane
GB-**Winnersh**,Wokingham
Berks, RG11 5AR
Tel. (0734) 78 47 74
Telex: 847178/848179

ITALY
Hewlett-Packard Italiana S.p.A.
Via Amerigo Vespucci 2
Casella postale 3645
I-20100 **Milano**
Tel: (2) 6251 (10 lines)
Cable: HEWPACKIT Milano
Telex: 32046

Hewlett-Packard Italiana S.p.A.
Via Pietro Maroncelli 40
(ang. Via Visentin)
I-35100 **Padova**
Tel: (49) 66 48 88
Telex: 41612 Hewpacki

Medical only
Hewlett-Packard Italiana S.p.A
Via d'Aghiardi, 7
I-56100 **Pisa**
Tel: (050) 2 32 04
Telex: 32046 via Milano

Hewlett-Packard Italiana S.p.A.
Via G. Armellini 10
I-00143 **Roma**
Tel: (06) 54 69 61
Telex: 61514
Cable: HEWPACKIT Roma

Hewlett-Packard Italiana S.p.A.
Corso Giovanni Lanza 94
I-10130 **Torino**
Tel (011) 682245/659308

Medical/Calculators Only
Hewlett-Packard Italiana S.p.A.
Via Principe Nicola 43 G/C
I-95126 **Catania**
Tel:(095) 37 05 04

Hewlett-Packard Italiana S.p.A
Via Amerigo Vespucci, 9
I-80142 **Napoli**
Tel: (081) 33 77 11

Hewlett-Packard Italiana S.p.A.
Via E. Masi, 9/B
I-40137 **Bologna**
Tel: (051) 30 78 87

KUWAIT
Al-Khaldiya Trading &
Contracting Co.
P.O. Box 830
Kuwait
Tel: 42 49 10
Cable. VISCOUNT

LUXEMBURG
Hewlett-Packard Benelux
S.A./N.V.
Avenue du Col-Vert. 1.
(Groenkraaglaan)
B-1170 **Brussels**
Tel: (02) 672 22 40
Cable: PALOBEN Brussels
Telex. 23 494

MOROCCO
Gerep
190. Blvd. Brahim Roudani
Casablanca
Tel 25-16-76/25-90-99
Cable: Gerep-Casa
Telex: 23739

NETHERLANDS
Hewlett-Packard Benelux N.V.
Van Heuven Goedhartlaan 121
P.O. Box 667
NL-1134 **Amstelveen**
Tel: (020) 47 20 21
Cable: PALOBEN Amsterdam
Telex: 13 216 hepa nl

NORWAY
Hewlett-Packard Norge A/S
Nesveien 13
Box 149
N-1344 **Haslum**
Tel: (02) 53 83 60
Telex: 16621 hpnas n

POLAND
Biuro Informacji Technicznej
Hewlett-Packard
U1 Stawki 2, 6P
00-950**Warszawa**
Tel: 395962/395187
Telex: 81 24 53 hepa pl

UNIPAN
Zaklad Doswiadczalny
Budowy Aparatury Naukowej
U1. Krajowej Rady Narodowej 51/55
00-800 **Warszawa**
Tel: 36190
Telex: 81 46 48

Zaklady Naprawcze Sprzetu
Medycznego
Plac Komuny Paryskiej 6
90-007 **Lódź**
Tel: 334-41. 337-83

PORTUGAL
Telectra-Empresa Técnica de
Equipamentos Eléctricos S.a.r.l.
Rua Rodrigo da Fonseca 103
P.O. Box 2531
P-**Lisbon** 1
Tel: (19) 68 60 72
Cable: TELECTRA Lisbon
Telex: 12598

Medical only
Mundinter
Intercambio Mundial de Comércio
S.a.r.l.
Av.A.A.de Aguiar 138
P.O. Box 2761
P - **Lisbon**
Tel: (19) 53 21 31/7
Cable: INTERCAMBIO Lisbon

RUMANIA
Hewlett-Packard Reprezentanta
Bd.N. Balcescu 16
Bucharest
Tel: 158023/138885
Telex: 10440

I.I.R.U.C.
Intreprinderea Pentru
Intretinerea
Si Repararea Utilajelor de Calcul
B-dul prof. Dimitrie Pompei 6
Bucharest-Sectorul 2
Tel: 12 64 30
Telex: 11716

SAUDI ARABIA
Modern Electronic Establishment
King Abdul Aziz str.(Head office)
P.O. Box 1228
Jeddah
Tel: 31173-332201
Cable: ELECTRA

P.O. Box 2728 (Service center)
Riyadh
Tel: 62596-66232
Cable: RAOUFCO

SPAIN
Hewlett-Packard Española, S.A.
Jerez, Calle 3
E-**Madrid** 16
Tel:(1) 458 26 00 (10 lines)
Telex: 23515 hpe

Hewlett-Packard Española, S.A.
Milanesado 21-23
E-**Barcelona** 17
Tel: (3) 203 6200 (5 lines)
Telex: 52603 hpbe e

Hewlett-Packard Española, S.A.
Av Ramón y Cajal, 1
Edificio Sevilla, planta 9°,
E-**Seville** 5
Tel: 64 44 54/58

Hewlett-Packard Española S.A.
Edificio Albia II 7° B
E-**Bilbao**-1
Tel: 23 83 06/23 82 06

Calculators Only
Hewlett-Packard Española S.A.
Gran Via Fernando El Católico, 67
E-**Valencia**-8
Tel: 326 67 28/326 85 55

SWEDEN
Hewlett-Packard Sverige AB
Enighetsvägen 1-3
Fack
S-161 20 **Bromma** 20
Tel: (08) 730 05 50
Cable: MEASUREMENTS
Stockholm
Telex: 10721

Hewlett-Packard Sverige AB
Ostra Vintergatan 22
S-702 40 **Orebro**
Tel: (019) 14 07 20

Hewlett-Packard Sverige AB
Frötallsgatan 30
S-421 32 **Västra Frölunda**
Tel: (031) 49 09 50
Telex: 10721 Via Bromma Office

SWITZERLAND
Hewlett-Packard (Schweiz) AG
Zürcherstrasse 20
P.O. Box 307
CH-8952 **Schlieren-Zurich**
Tel: (01) 730 52 40/730 18 21
Cable: HPAG CH
Telex: 53933 hpag ch

Hewlett-Packard (schweiz) AG
Château Bloc 19
CH-1219 **Le Lignon-Geneva**
Tel: (022) 96 03 22
Cable: HEWPACKAG Geneva
Telex: 27 333 hpag ch

SYRIA
Medical/Calculator only
Sawah & Co.
Place Azmé
B.P. 2308
SYR-**Damascus**
Tel: 16367, 19697, 14268
Cable: SAWAH, Damascus

TURKEY
Telekom Engineering Bureau
P.O. Box 437
Beyoğlu
TR-**Istanbul**
Tel: 49 40 40
Cable: TELEMATION Istanbul
Telex: 23609

Medical only
E.M.A.
Muhendislik Kollektif Sirketi
Adakale Sokak 41/6
TR-**Ankara**
Tel: 175622

Analytical only
Yilmaz Ozyurek
Milli Mudafaa Cad No. 16/6
Kizilay
TR-**Ankara**
Tel: 25 03 09
Telex: 42576 Ozek tr

UNITED KINGDOM
Hewlett-Packard Ltd.
King Street Lane
GB-**Winnersh,** Wokingham
Berks. RG11 5AR
Tel: (0734) 78 47 74
Cable: Hewpie London
Telex:847178/9

Hewlett-Packard Ltd.
Trafalger House,
Navagation Road
Altrincham
Cheshire WA14 1NU
Tel: (061) 928 6422
Cable: Hewpie Manchester
Telex: 668068

Hewlett-Packard Ltd.
Lygon Court
Hereward Rise
Dudley Road
Halesowen,
West Midlands B62 8SD
Tel: (021) 550 9911
Telex: 339105

Hewlett-Packard Ltd.
Wedge House
799, London Road
GB-**Thornton Heath**
Surrey CR4 6XL
Tel: (01) 684 0103/8
Telex: 946825

Hewlett-Packard Ltd.
c/o Makro
South Serviceholesale Centre
Wear Industrial Estate
Washington
GB-**New Town**, County Durham
Tel: Washington 464001 ext. 57/58

Hewlett-Packard Ltd
10, Wesley St.
GB-**Castleford**
West Yorkshire WF10 1AE
Tel: (09775) 50402
Telex: 557355

Hewlett-Packard Ltd
1, Wallace Way
GB-**Hitchin**
Herts
Tel: (0462) 52824/56704
Telex: 825981

USSR
Hewlett-Packard
Representative Office USSR
Pokrovsky Boulevard 4/17-KW 12
Moscow 101000
Tel:294-2024
Telex: 7825 hewpak su

YUGOSLAVIA
Iskra-standard/Hewlett-Packard
Miklosiceva 38/VII
61000 **Ljubljana**
Tel: 31 58 79/32 16 74
Telex: 31583

SOCIALIST COUNTRIES NOT SHOWN PLEASE CONTACT:
Hewlett-Packard Ges.m.b.H
P.O. Box 7
A-1205 **Vienna**, Austria
Tel: (0222) 35 16 21 to 27
Cable: HEWPAK Vienna
Telex: 75923 hewpak a

MEDITERRANEAN AND MIDDLE EAST COUNTRIES NOT SHOWN PLEASE CONTACT:
Hewlett-Packard S.A.
Mediterranean and Middle
East Operations
35, Kolokotroni Street
Platia Kefallariou
GR-Kifissia-**Athens**, Greece
Tel: 8080337/359/429
Telex: 21-6588
Cable: HEWPACKSA Athens

FOR OTHER AREAS NOT LISTED CONTACT
Hewlett-Packard S.A.
7, rue du Bois-du-Lan
P.O. Box
CH-1217 Meyrin 2 - **Geneva**
Switzerland
Tel: (022) 82 70 00
Cable: HEWPACKSA Geneva
Telex: 2 24 86

UNITED STATES

ALABAMA
8290 Whitesburg Dr., S.E.
P.O. Box 4207
Huntsville 35802
Tel: (205) 881-4591

Medical Only
228 W. Valley Ave.,
Room 220
Birmingham 35209
Tel: (205) 942-2081/2

ARIZONA
2336 E. Magnolia St.
Phoenix 85034
Tel: (602) 244-1361

2424 East Aragon Rd.
Tucson 85706
Tel: (602) 294-3148

***ARKANSAS**
Medical Service Only
P.O. Box 5646
Brady Station
Little Rock 72215
Tel: (501) 376-1844

CALIFORNIA
1430 East Orangethorpe Ave.
Fullerton 92631
Tel: (714) 870-1000

3939 Lankershim Boulevard
North Hollywood 91604
Tel: (213) 877-1282
TWX: 910-499-2671

6305 Arizona Place
Los Angeles 90045
Tel: (213) 649-2511
TWX: 910-328-6147

***Los Angeles**
Tel: (213) 776-7500

3003 Scott Boulevard
Santa Clara 95050
Tel: (408) 249-7000
TWX: 910-338-0518

***Ridgecrest**
Tel: (714) 446-6165

646 W. North Market Blvd
Sacramento 95834
Tel: (916) 929-7222

9606 Aero Drive
P.O. Box 23333
San Diego 92123
Tel: (714) 279-3200

COLORADO
5600 South Ulster Parkway
Englewood 80110
Tel: (303) 771-3455

CONNECTICUT
12 Lunar Drive
New Haven 06525
Tel: (203) 389-6551
TWX: 710-465-2029

FLORIDA
P.O. Box 24210
2806 W. Oakland Park Blvd.
Ft. Lauderdale 33311
Tel: (305) 731-2020

***Jacksonville**
Medical Service only
Tel: (904) 398-0663

P.O. Box 13910
6177 Lake Ellenor Dr.
Orlando 32809
Tel: (305) 859-2900

P.O. Box 12826
Pensacola 32575
Tel: (904) 476-8422

GEORGIA
P.O. Box 105005
Atlanta 30348
Tel: (404) 955-1500
TWX:810-766-4890

Medical Service Only
***Augusta** 30903
Tel: (404) 736-0592

P.O Box 2103
Warner Robins 31098
Tel: (912) 922-0449

HAWAII
2875 So. King Street
Honolulu 96814
Tel: (808) 955-4455
Telex: 723-705

ILLINOIS
5201 Tollview Dr.
Rolling meadows 60008
Tel: (312) 255-9800
TWX: 910-687-2260

INDIANA
7301 North Shadeland Ave.
Indianapolis46250
Tel: (317)842-1000
TWX: 810-260-1797

IOWA
1902 Broadway
Iowa City 52240
Tel: (319) 338-9466

KENTUCKY
Medical Only
Atkinson Square
3901 Atkinson Dr.
Suite 207
Louisville 40218
Tel: (502) 456-1573

LOUISIANA
P.O. Box 840
3229-39 Williams Boulevard
Kenner 70063
Tel: (504) 443-6201

MARYLAND
6707 Whitestone Road
Baltimore 21207
Tel: (301) 944-5400
TWX: 710-862-9157

2 Choke Cherry Road
Rockville 20850
Tel: (301) 948-6370
TWX: 710-828-9684

MASSACHUSETTS
32 Hartwell Ave.
Lexington 02173
Tel: (617) 861-8960
TWX: 710-326-6904

MICHIGAN
23855 Research Drive
Farmington Hills 48024
Tel: (313) 476-6400

MINNESOTA
2400 N. Prior Ave.
St. Paul 55113
Tel (612) 636-0700

MISSISSIPPI
***Jackson**
Medical Service only
Tel: (601) 982-9363

MISSOURI
11131 Colorado Ave.
Kansas City 64137
Tel (816) 763-8000
TWX: 910-771-2087

1024 Executive Parkway
St. Louis 63141
Tel: (314) 878-0200

NEBRASKA
Medical Only
7171 Mercy Road
Suite 110
Omaha 68106
Tel: (402) 392-0948

NEW JERSEY
W. 120 Century Rd.
Paramus 07652
Tel: (201) 265-5000
TWX: 710-990-4951

Crystal Brook Professional
Building
Eatontown 07724
Tel:(201) 542-1384

NEW MEXICO
P.O. Box 11634
Station E
11300 Lomas Blvd., N.E.
Albuquerque 87123
Tel: (505) 292-1330
TWX. 910-989-1185

156 Wyatt Drive
Las Cruces 88001
Tel: (505) 526-2484
TWX: 910-9983-0550

NEW YORK
6 Automation Lane
Computer Park
Albany 12205
Tel. (518) 458-1550

201 South Avenue
Poughkeepsie 12601
Tel: (914) 454-7330
TWX: 510-253-5981

650 Perinton Hill Office Park
Fairport 14450
Tel: (716) 223-9950

5858 East Molloy Road
Syracuse 13211
Tel: (315) 454-2486
TWX: 710-541-0482

1 Crossways Park West
Woodbury 11797
Tel: (516) 921-0300
TWX: 710-990-4951

NORTH CAROLINA
P.O. Box 5188
1923 North Main Street
High Point 27262
Tel: (919) 885-8101

OHIO
16500 Sprague Road
Cleveland 44130
Tel: (216) 243-7300
TWX: 810-423-9430

330 Progress Rd.
Dayton 45449
Tel: (513) 859-8202

1041 Kingsmill Parkway
Columbus 43229
Tel: (614) 436-1041

OKLAHOMA
P.O. Box 32008
Oklahoma City 73132
Tel: (405) 721-0200

OREGON
17890 SW Lower Boones
Ferry Road
Tualatin 97062
Tel: (503) 620-3350

PENNSYLVANIA
111 Zeta Drive
Pittsburgh 15238
Tel: (412) 782-0400

1021 8th Avenue
King of Prussia Industrial Park
King of Prussia 19406
Tel: (215) 265-7000
TWX: 510-660-2670

SOUTH CAROLINA
6941-0 N. Trenholm Road
Columbia 29260
Tel: (803) 782-6493

TENNESSEE
***Knoxville**
Medical Service only
Tel: (615) 523-5022

3027 Vanguard Dr.
Director's Plaza
Memphis 38131
Tel: (901) 346-8370

Nashville
Medical Service only
Tel: (615) 244-5448

TEXAS
P.O. Box 1270
201 E. Arapaho Rd.
Richardson 75080
Tel: (214) 231-6101

10535 Harwin Dr.
Houston 77036
Tel: (713) 776-6400

205 Billy Mitchell Road
San Antonio 78226
Tel: (512) 434-8241

UTAH
2160 South 3270 West Street
Salt Lake City 84119
Tel: (801) 972-4711

VIRGINA
P.O. Box 12778
No. 7 Koger Exec. Center
Suite 212
Norfolk 23502
Tel:(804) 461-4025/6

P.O.Box 9669
2914 Hungary Springs Road
Richmond 23228
Tel: (804) 285-3431

WASHINGTON
Bellefield Office Pk.
1203-114th Ave. S.E.
Bellevue 98004
Tel: (206) 454-3971
TWX: 910-443-2446

***WEST VIRGINIA**
Medical/Analytical Only
Charleston
Tel: (304) 345-1640

WISCONSIN
9004 West Lincoln Ave.
West Allis 53227
Tel: (414) 541-0550

FOR U.S. AREAS NOT LISTED:
Contact the regional office nearest you: Atlanta, Georgia... North Hollywood, California... Rockville, Maryland...Rolling Meadows, Illinois.Their complete addresses are listed above.

***Service Only**

8/77

PART NO. 02108-90008
Printed in U.S.A. **10/77**