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 table 6-2 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 ≤ 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.

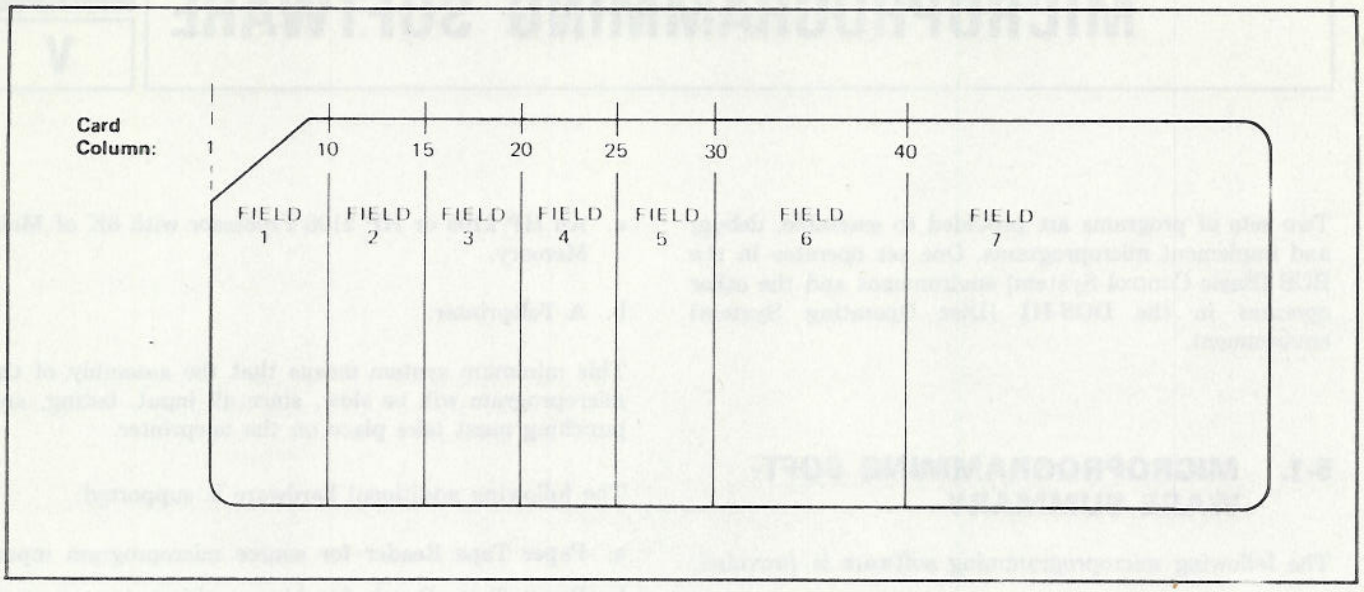


Figure 5-1. Micro-instruction Card Source Record

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
2n words for n micro-instructions (2 words for each micro-instruction)
5+2n words for one Instruction Record

No more than 27 micro-instructions are written into an Instruction Record. Hence the maximum length = 5+(2x27)=59 words. The last object record is a four word End Record. When the microprogram consists of more than 27 micro-instructions, a series of Instruction Records are produced with the last one holding 27 or less micro-instructions. For example, if 57 micro-instructions have been assembled, three Instruction Records and an End Record are required consisting of the following:

a. Instruction Record 1 holds 27 micro-instructions and consists of

5 words of header
54 words for 27 micro-instructions
59 words

b. Instruction Record 2 holds 27 micro-instructions and consists of

5 words of header
54 words for 27 micro-instructions
59 words

c. Instruction Record 3 holds 3 micro-instructions and consists of

5 words of header
6 words for 3 micro-instructions
11 words

d. The End Record consists of

4 words

133 words for the entire microprogram Binary Object.

The Standard Object format is accepted by all programs which accept standard relocatable format. Thus a Standard Object tape can be stored in a DOS-III file using the ":STORE,R,..." directive. However, if the DOS-III user wants the Binary Object stored automatically in a disc file by the micro-assembler, the DOS-III directive "STORE,B,. . ." must have previously been used to reserve a disc file.

The Micro-assembler can also produce a non-standard object as the result of the inclusion of the $RCASE control statement. This optional object is the HP ROM Simulator Object tape. The format of this tape is shown in Appendix A, Figure A-2.

5-8. SYMBOL TABLE LISTING

If the user has a $SYMTAB control statement in his microprogram source input, then the micro-assembler will print a symbol table on the device with logical unit number 6 or on the device defined by the $LIST control statement, if present.

An example of a symbol table is shown in Figure 5-2.

On the left are the symbols or labels in the microprogram. On the right is the value of the symbol; that is the six digit absolute octal address of the symbol. Where X follows the address, the symbol has been defined by a $EXTERNAL control statement.

```
SYMBOL TABLE

MOVE    002412X
GOTO    003421X
RET     002427X
LAST    002717X
OUT     002011
ERR1    002012
```

Figure 5-2. Symbol Table

5-9. MICROASSEMBLY LISTING

Unless suppressed by the $NOLIST control statement, the micro-assembler provides a listing like the one shown in Figure 5-3. This listing is associated with the symbol table illustrated in Figure 5-2.

5-10. MICRO-ASSEMBLER ERROR MESSAGES

During the assembly process the micro-assembler checks each instruction for errors. If an error is detected, an error message of the following general form is printed in the Micro-assembly Listing.

**ERROR eeee IN LINE nnnn

where

eeee
is an Error Code defined in Table 5-1 and

nnnn
is a line number in the Micro-assembly Listing.

Table 5-1 gives the meaning of each error code and the recovery procedure. Note that Figure 5-2 holds examples of two error messages in lines 9 and 11.

5-11. DOS-III OPERATION OF MICRO-ASSEMBLER

Before using the DOS-III version of the Micro-assembler, the following items must be available.

a. A current DOS-III system.

b. A source microprogram, on cards, paper tape, or in a source file on disc.

c. The Micro-assembler program named MICRO stored in the DOS-III user library. If MICRO still is on relocatable object paper tape (HP 12978-160001), it can be loaded in the same way as any other relocatable object program.

For the detailed description of DOS-III operation, see HP 24307B DOS-III Reference Manual (HP 24307-90006).

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**
```

Line Number	ROM Address	Bits 23-16	Bits 15-0	Field 1	Field 2	Field 3	Field 4	Field 5	Field 6	Field 7
		Binary Micro-instruction								

Figure 5-3. Micro-Assembly Listing

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

p2 = list device logical unit number

p3 = paper tape punch device logical unit number

p4 = maximum number of lines-per-page on the list device.

If 99 is entered for any of the above parameters, that parameter and all those that follow are defaulted to "standard" values.

d. The program title

MICRO-ASSEMBLER

is printed and Pass 1 begins. If a $SYMTAB control statement is in the source microprogram, the symbol table is printed at the conclusion of Pass 1. Pass 2 begins immediately (from disc) and the listing and relocatable object tape are output. Micro-assembly is complete.

Note: If Pass 2 fails to begin, check that the paper tape punch is turned on. The micro-assembler will cycle in a loop until the punch is turned on.

5-12. BCS OPERATION OF MICRO-ASSEMBLER

Before proceeding, the following items must be available:

- An absolute BCS binary tape.
- A reloctable object tape of the Micro-assembler program MICRO (HP 12978-160003).
- A source microprogram either on cards or paper tape.

For a detailed description of BCS usage, see the **Basic Control System** manual (HP 02116-9017).

The following procedure need be performed only once. When an absolute binary tape of the Micro-assembler is punched, it is used as described in the procedure "Executing the Micro-assembler."

Making an Absolute Micro-assembler tape:

a. Load the absolute BCS binary tape using the Basic Binary Loader.

b. Set the P-register to 2. Set bit 14 of the Switch Register and clear all other Switch Register bits.

c. Place the MICRO relocatable object tape in the paper tape reader. Check that the paper tape reader and the console device are on. Turn on the paper tape punch. Press PRESET and RUN on the CPU front panel. MICRO reads in and absolute binary tape is punched.

d. The message

*LOAD

is printed and the computer waits. Set Switch Register bits 2 and 14 leaving all others clear. Load BCS Library tape into the paper tape reader. Press RUN.

e. The BCS Library tape reads in and the rest of the absolute binary tape is punched. Linkage information is printed on the console device.

This is the absolute binary tape of MICRO, used for input to the next step.

Executing the Micro-assembler:

a. Load the MICRO absolute binary tape using the Basic Binary Loader.

b. When loading is complete, set P-register to 2. Press PRESET and RUN. The message

MICRO-ASSEMBLER

is printed followed by a request for the logical unit number of the source input device

INPUT=

c. Enter the logical unit number followed by carriage return/line feed. Pass 1 now begins. If a $SYMTAB control statement is in the microprogram source, the symbol table is printed at the conclusion of Pass 1. (See Section 5-5 for a description of the $SYMTAB control statement.)

d. Turn on the paper tape punch.

e. Pass 2 begins immediately. If no $PASS2 control statement was included in the source, the message

RELOAD SOURCE, PRESS RUN

is printed. Reload the source microprogram into the input device and then press RUN on the front panel of the computer.

Note: If Pass 2 fails to begin, check that the paper tape punch is turned on. The micro-assembler will cycle in a loop until the punch is turned on.

If a teletype is used for both listing and punching, the computer halts (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 Section 6 of this manual for installation of the module selection Jumper Assembly and the WCS PCAs.

5-14. HARDWARE ENVIRONMENT

The BCS version requires the following minimum hardware:

a. HP 21MX Series Computer with 8K of Main Memory

b. A console device

c. A paper tape reader

d. One or more WCS PCA's, depending on the size of the microprogram to be debugged.

e. If a ROM program tape is to be punched, a paper tape punch is also required.

The DOS-III version of the MDE requires the same minimum hardware as the DOS-III system.

5-15. INITIALIZATION PROGRAM

When the Micro Debug Editor is to be run for debugging purposes (as opposed to being run merely to punch ROM

```
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.

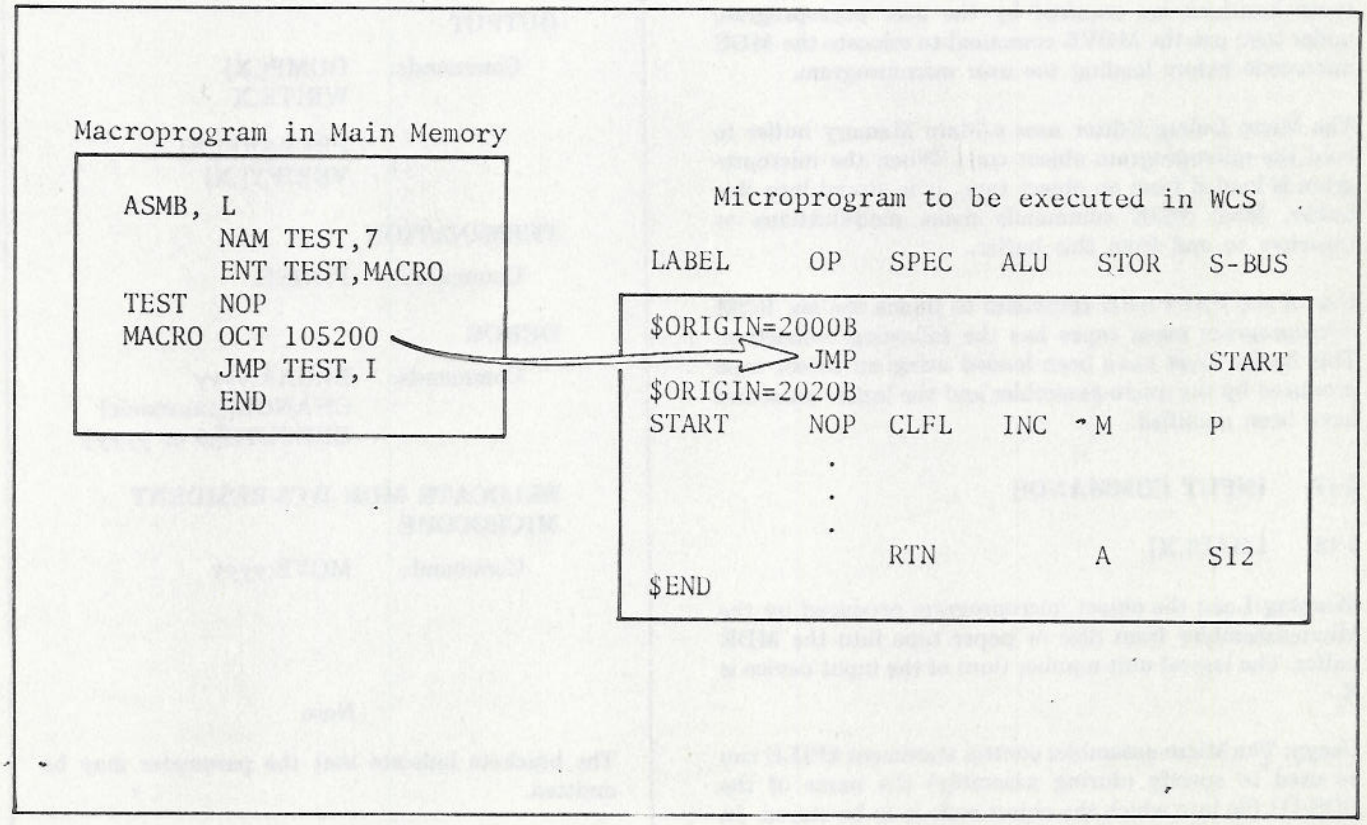


Figure 5-5. Test Program Call to Microprogram

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 nonexistent 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 Micro-assembler.

The following is an example of how the user can proceed. For details on additional DOS-III options, see DOS-III manual (HP 24307-90006).

a. Store the two tapes, MDE and TEST, on the disc using the DOS-III store command

:ST,R,**filename, lun**

where **filename** is any suitable label and **lun** is the logical unit number of the paper tape reader from which the tapes are entered.

b. Make sure the list device is on. At the console device enter

:PR,LOADR,2

DOS-III responds with

ENTER FILE NAMES OR /E

c. Respond as follows:

MDE filename, TEST filename, /E

where **MDE filename** and **TEST filename** are the chosen file names used with the "ST" store command (step A), and /E specifies end of entry.

If MDE is being used only to load WCS with a microprogram, the **TEST filename** may be omitted. The loader then reads the two files into main memory. If the TEST initialization program has been omitted, the message

UNDEFINED EXTS

is printed indicating TEST 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:

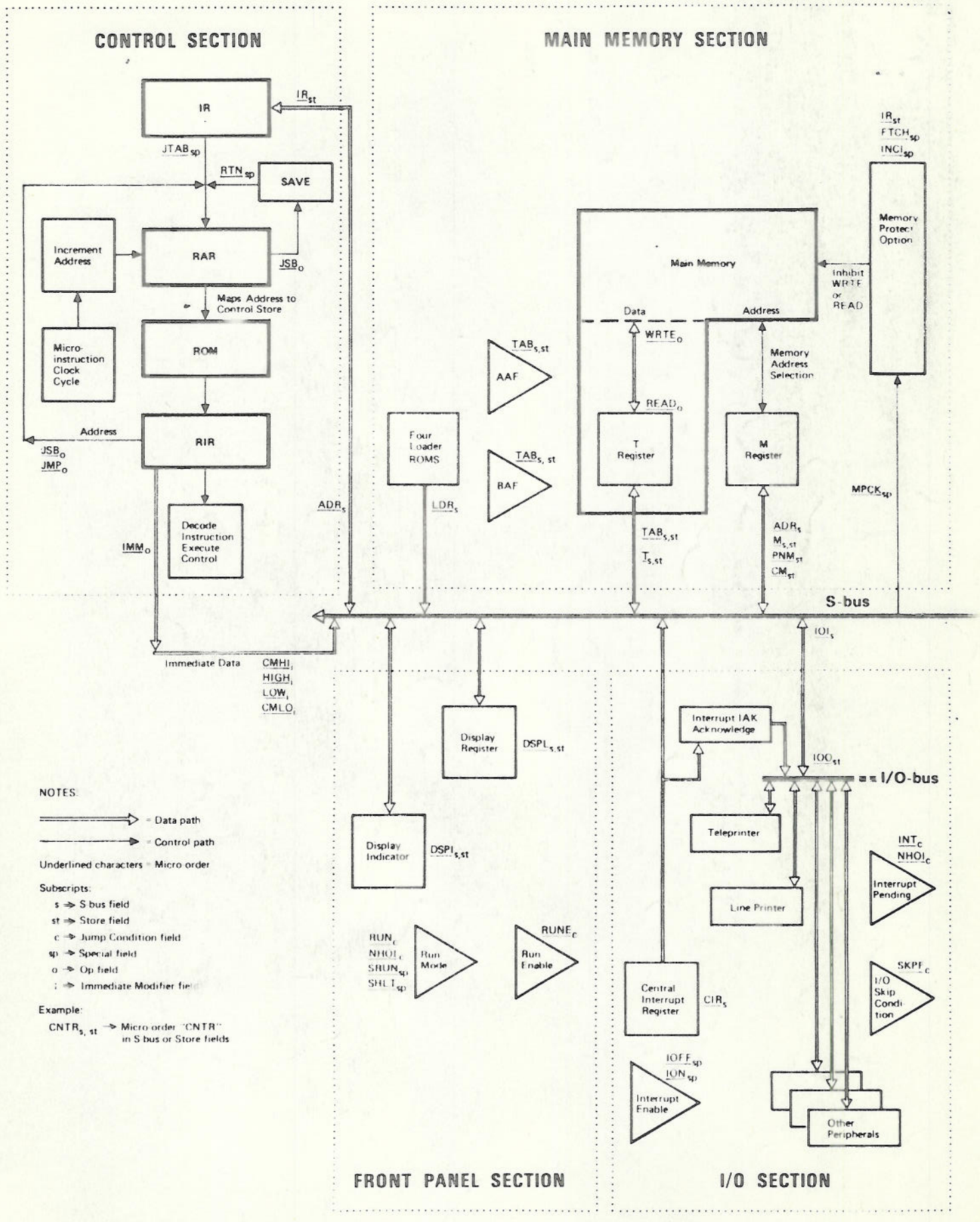
READ

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.



NOTES:

= Data path

= Control path

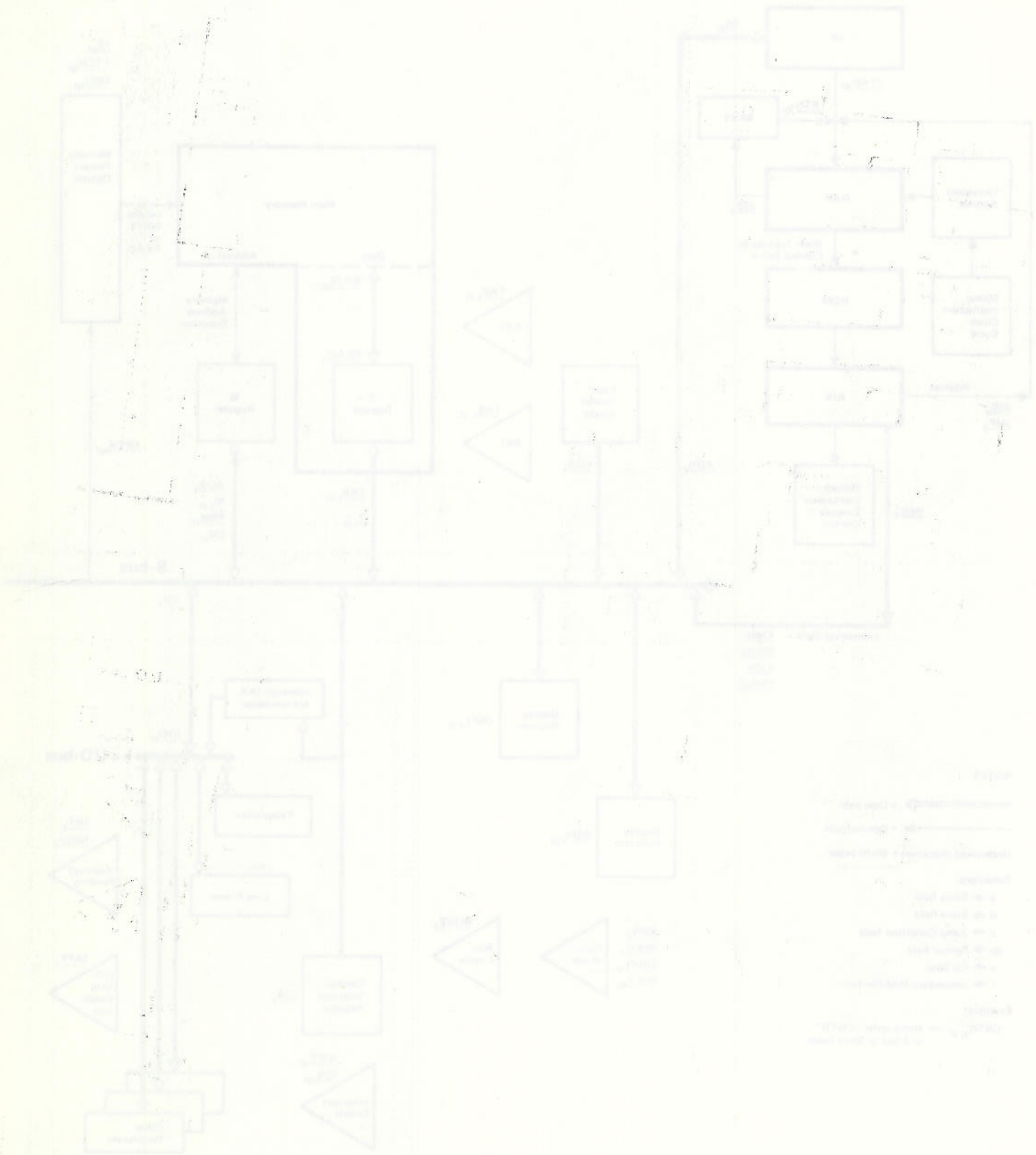
Underlined characters = Micro order

Subscripts:

- s ⇒ S bus field
- st ⇒ Store field
- c ⇒ Jump Condition field
- sp ⇒ Special field
- o ⇒ Op field
- i ⇒ Immediate Modifier field

Example:

$CNTR_{s,\,st}$ ⇒ Micro order "CNTR" in S bus or Store fields



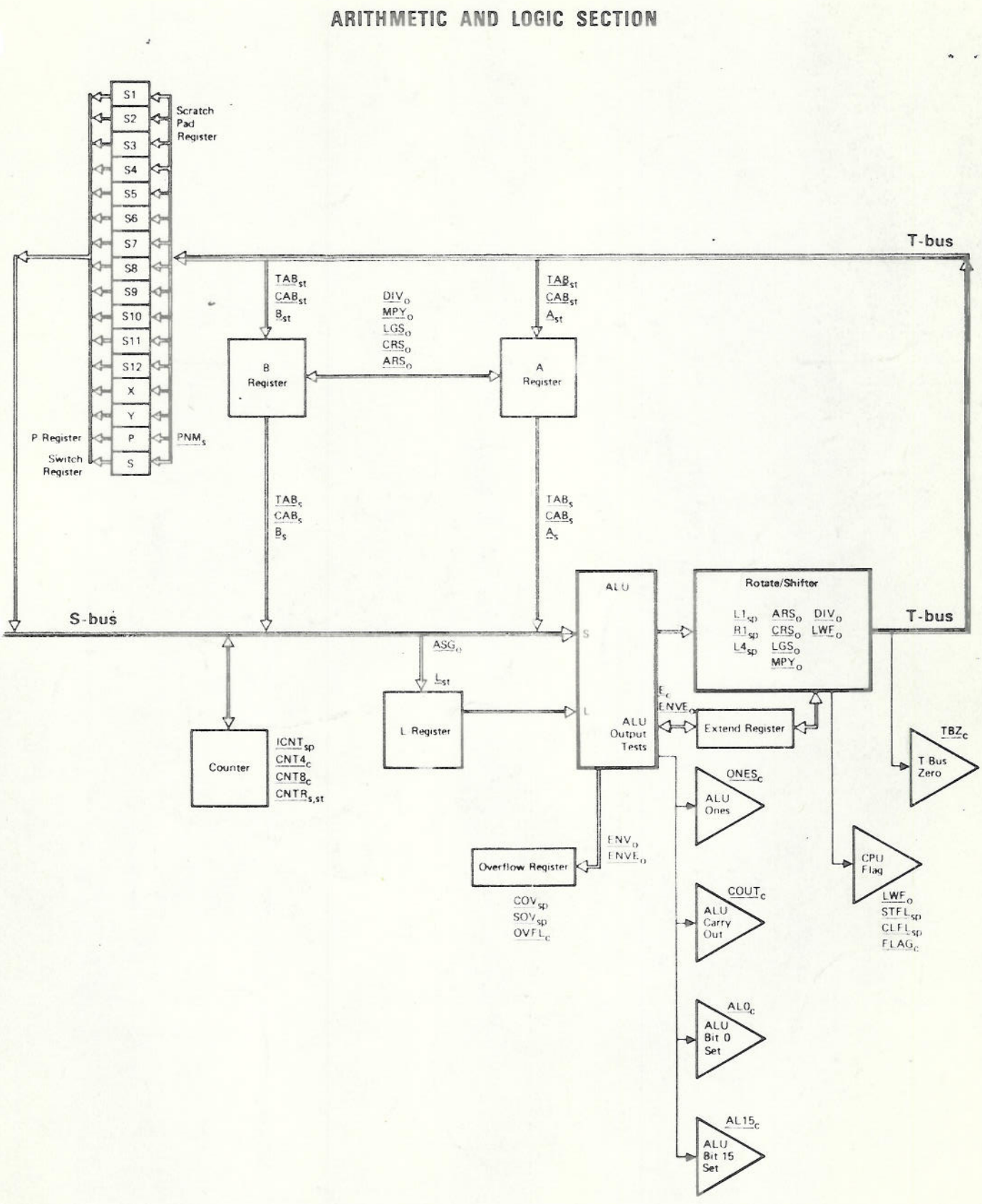


Figure D-1. Functional Block Diagram