ENKAA-2.1

VAX 11/730
MICRO-MONITOR

EP-ENKAA-DL-2.1
FICHE 1 OF 1

digital

IDENTIFICATION

Product code: ZZ-ENKAA VERSION 2.10

Product name: MICRO-DIAGNOSTIC MONITOR FOR VAX-11/730

Product date: 14-JUN-1983

Maintainer: BASE SYSTEMS DIAGNOSTIC ENGINEERING

Table of Contents

1.0 ABSTRACT

This program is the heart of the VAX-11/730 micro-diagnostics package. It provides the code to load, control and monitor each micro-diagnostic, regardless of whether the diagnostic is to reside in 8085 RAM or in WCS RAM. It also contains the code necessary to communicate and run under APT.

The micro-monitor is responsible for parsing and executing commands typed in at the terminal or sent by APT. It will load micro-diagnostics from the TU58 cassette into the proper area when requested. Under APT, it will downline load micro-diagnostics. The monitor is also responsible for reporting any errors that occur.

2.0 HARDWARE REQUIREMENTS

This program needs only a WCS module to run. Of course, as such, it would not be very useful since all of the micro-diagnostics that it will load and run use at least a portion of the CPU module, while some micro-diagnostics use ather modules in the machine.

There are no restrictions as to what other hardware may be added to the machine as far as the micro-monitor is concerned.

3.0 SOFTWARE REQUIREMENTS

The micro-monitor runs as a standalone program residing in 8085 RAM where the console software is normally present. It neither requires nor uses any system micro-code. It will use WCS micro-routines that are loaded with a WCS based micro-diagnostic.

4.0 PREREQUISITES

The only prerequisites are that the ROM based selftest, that executes on powerup, completes succesfully, and that the TU58 drive containing the micro-diagnostic cassette is operational.

5.0 OPERATING INSTRUCTIONS

The following section describes the commands that can be executed via the micro-monitor:

Note: Square brackets ([]) denote optional letters that may be typed if desired. All commands, except DIR and those with a /, can be abbreviated to 2 letters.

All numerical inputs (addresses or data) must be hex values.

DIAGNOSE

The diagnose command is used to execute micro-diagnostics. The execution of these diagnostics may be controlled by using combinations of the following keywords:

BO[ARD]	SE[CTION]	TE[ST]
CO[NTINUE]	PA[SS]	SH[ORTEN]

The following are some examples of diagnose commands:

```
MIC>DI
```

Execute all non-optional sections that MICMON has listed in it's internal tables.

SECTION keyword

This function is incompatable with the BOARD keyword.

```
MIC>DI SE XXXXXX
```

Load section XXXXXX from the tape and execute. MICMON looks in its internal tables to find this section name and then loads the section in WCS or in 8085 RAM. If MICMON can not find the name in the table it assumes that the section is to be loaded in WCS.

BOARD keyword

This function is incompatable with the following keywords:

SECTION TEST

NOTE: If an error occurs and the SHORTEN keyword was used this will cause MICMON to exit from it's board mode and

perform the shorten function on the section that is currently loaded.

MIC>DI BO YYY

Load and execute all non-optional sections that are associated with board YYY (YYY is WCS, DAP or CPU, FPA, MCT, or IDC).

TEST keyword

This function is incompatable with the BOARD keyword.

MIC>DI TE 2

Execute test 2 of the section that is currently loaded.

MIC>DI TE 2 5

Execute tests 2 through 5 inclusive of the section that is currently loaded.

CONTINUE keyword

Note: This keyword may only appear after the TEST Keyword.

MIC>DI TE 2 CO

Execute from test 2 to end of the currently loaded section (i.e. through to the last test).

PASS keyword

MIC>DI SE XXXXXX TE 2 5 PA 3

Load section XXXXXX from the tape and execute tests 2 thru 5 inclusive and repeat 3 times.

MIC>DI BO YYY PA -1

Load and execute all non-optional sections that are associated with board YYY and repeat forever.

SHORTEN keyword

MIC>DI TE 1 15 SH

Execute tests 1 thru 15 inclusive. If an error occurs,

shorten the loop to execute tests 1 through the test that is causing the error.

SHOW

The show command prints the names of the flags that are in their asserted state.

```
HALT            Halt on error
LOOP            Loop on error
NER             No error report
BELL            Bell on error
SER             Single error report or loop on sizing test
TRACE           Print test numbers
SOMM address    Stop on micro-match, current address
```

```
MIC>SH
```

SET/CLEAR

The SET/CLEAR command is used to enable functions used in the execution of diagnostics. The CL function usually disables what SE enabled. When the CL functionality is not obvious, it will be discussed under the SE description. The following functions are available.

```
HA[LT]          LO[OP]          NE[R]
BE[LL]          SE[R]           TR[ACE]
SO[MM]          DE[FAULT]       PA[RITY]
ST[EP]*         BR[EAK]
```

* = ONLY SET IS AVAILABLE FOR THIS COMMAND

HALT

HALT ON ERROR. Setting halt on error will cause MICMON to return to command level after reporting an error.

```
MIC>SE HA
```

LOOP

Loop on error. Setting loop on error will cause MICMON to loop on the smallest piece of test code needed to reproduce the error. When WCS based tests are executed, MICMON will set a flag which is read by the error routine in the WCS

diagnostic. Loop on error continues to loop even if the error goes away. This is useful for intermittant errors.

Note: HALT has a higher priority than loop. Thus if both are set halt will occur first.

MIC>SE LO

NER

No error reports. Setting no error reports will cause MICMON to skip the printout portion of error reporting.

MIC>SE NE

BELL

Bell on error. Setting bell on error will cause MICMON to ring the terminal bell each time there is an error reported.

Note: This function is independant of NER.

MIC>SE BE

SER

Enable Single Bit Errors to be reported as normal errors. With this flag cleared, only the number of single bit errors detected is printed. This function is sometimes used to force looping on some of the sizing routines. See test documentation for use in these routines.

MIC>SE SE

TRACE

Trace execution. Setting trace will cause MICMON to print the test number before starting each test.

MIC>SE TR

SOMM

Stop on micro match. Setting stop on micro match will cause MICMON to write bad parity to the WCS location specified. When execution of this word occurs and if stall on parity error is not disabled (See set parity), MICMON will trap the interrupt caused by the parity error. If the address matches the last somm address set, MICMON will print "SOMM"

and the UPC, and return to command mode. If the parity error is not the SOMM address a parity error message will be issued. If a SET SOMM is issued, and a SOMM address is already in effect, the previous SOMM address will be cleared.

Note: This command is affected by SET PARITY and CLEAR PARITY in 2 ways. In either case SOMM will not work properly.

1) Stall on parity error must be enabled. Thus if SE PA is enabled, a CL PA must be issued before starting test execution.

2) It is possible to clear the parity error (and thus the SOMM) with CLEAR PARITY somm-address. In this case, the SHOW command will still think that SOMM is set.

MIC>SE SO AAAA
where AAAA = WCS address.

DEFAULT

Set flags to default value. This will cause micmon to SET or CLEAR specific flags to their default value. The default is SOMM disabled and all flags clear except halt on error.

MIC>SE DE

PARITY

Set bad parity. This command will cause MICMON to write bad parity to the address given and to disable stall on parity errors. This command is useful to provide a scope sync at a specific micro-word. If no address is given, only disable stall on parity error will occur.

Clear parity is used to remove the parity error and reenable stall on parity error. If no address is given, only enable stall on parity error will occur. This command is also useful to write good parity at a location that has been deposited into to assure that the parity is correct.

MIC>SE PA

MIC>SE PA AAAA
where AAAA = WCS address.

STEP

Set single step. This will cause MICMON to single step the

CPU n times. If no value for single stepping is given, the CPU will single step once for each space bar typed. Any other character will cause MICMON to exit single step mode and return to command level.

Note: The character typed to exit single step mode will become the first character of the next command. If a clean start on the next command is desired, ^C can be typed to exit single step mode.

MIC>SE ST

MIC>SE ST NNNN
where NNNN = number of times to single step CPU.

BREAK

Break point in 8085 code. Setting break will cause MICMON to write a branch instuction to the break routine at the address specified. This branch instruction is 3 bytes long. Caution: This will over-write another routine's instructions. If the instruction is written to the program area of MICMON, a checksum error will be printed, but this will not prevent the user from executing MICMON. The instruction replaced by the breakpoint is NOT restored until a CL BR or another SE BR to a new address is issued. Clear break will replace the JMP instruction with the original three bytes at the RAM address where the set break was inserted.

MIC>SE BR AAAA
where AAAA = 8085 address.

MIC>CL BR

SPECIAL SELECT AND CLEAR COMMANDS

The IDC uses three special commands in addition to the Examine and Deposit commands listed below for setting up IDC registers. These commands are listed below.

SElect A Fifo

This command executes code in WCS that selects Fifo A to supply the address used when examine or deposit of the DBUF is executed. It is also used prior to a CL FI to select which address register will be cleared.

NOTE: This command requires that a WCS based micro diagnostic be loaded in WCS RAM.

```
MIC>SE AF
```

SElect B Fifo

This command executes code in WCS that selects Fifo B to supply the address used when examine or deposit of the DBUF is executed. It is also used prior to a CL FI to select which address register will be cleared.

NOTE: This command requires that a WCS based micro diagnostic be loaded in WCS RAM.

```
MIC>SE BF
```

CLear FIfo

This command executes code in WCS that clears the FIFO address reg currently selected. It should be preceeded by the SE AF or SE BF command above if it matters which FIFO is cleared by this command.

NOTE: This command requires that a WCS based micro diagnostic be loaded in WCS RAM.

```
MIC>CL FI
```

EXAMINE/DEPOSIT

The examine and deposit commands are used to read or write data into various registers or locations. The following examine and deposit commands are available (also see SE AF, SE BF, and CL FI for special IDC commands):

RA[M] or /U	UP[C]	UB[S]#
CS[R]	WC[S] or /C	MC[T]#
WR[K]	LS	TB#
MM#	OS	IC[SR]#
ID[AR]#	DB[UF]#	PA[TT]*#
PO[SIT]*#		

* = ONLY EXAMINE AVAILABLE FOR THIS COMMAND
\# = THIS COMMAND REQUIRES THAT A WCS BASED DIAGNOSTIC BE LOADED IN WCS RAM

RAM
/U

Examine/deposit console (8085) RAM. The examine command will print 4 bytes of information at the address specified. The deposit will write 1 byte of data at the address given.

```
        MIC>EX RA AAAA
        MIC>E/U AAAA
           where AAAA is an 8085 address.

        MIC>DE RA AAAA DDDD
        MIC>D/U AAAA DDDD
           where DDDD is data.


    UPC

        Examine/deposit UPC.  This command will  read  or  write  15
        bits of data into the UPC register.

        MIC>EX UP

        MIC>DE UP DDDD
           where DDDD is data.


    UBS

        Examine/deposit UBS.  This command will read or  write  data
        into the UNIBUS map portion of the translation buffer.

        NOTE:  This  command  requires  that  a  WCS  based  micro
        diagnostic be loaded in WCS RAM.

        MIC>EX UB AAAA
           where AAAA is UNIBUS  translation  buffer  (map)  address
        range is 200-3FF.

        MIC>DE UB AAAA DDDD
           where DDDD is data and bits 31-23 and  bit  0  cannot  be
        written into.


CSR

        Examine/deposit CSR.  This command will  read  or  write  24
        bits of data into the CPU CSR (control store register).

        MIC>EX CS
        MIC>DE CS DDDD
           where DDDD is data.


WCS
/C

        Examine/deposit WCS.  This command will read or  write  data
        into WCS (without parity calculation).

        Note:  CLEAR PARITY AAAA can be used to  make  parity  valid
```

after depositing data.

```
MIC>EX WC AAAA
MIC>E/C AAAA
   where AAAA is WCS address.
MIC>DE WC AAAA DDDD
MIC>D/C AAAA DDDD
   where DDDD is data.
```

MCT

Examine/deposit MCT. This command will read or write data into the memory controler's CSR registers.

NOTE: This command requires that a WCS based micro diagnostic be loaded in WCS RAM.

```
MIC>EX MC AAAA
   where AAAA is MCT register address (0,1, or 2).

MIC>DE MC AAAA DDDD
   where DDDD is data (only CSR 1, bits 29-25 are writable).
```

WRK

Examine/deposit WRK. This command will read or write data into the 2901 working registers.

```
MIC>EX WR AAAA
   where AAAA is 2901 register address.

MIC>DE WR AAAA DDDD
   where DDDD is data.
```

LS

Examine/deposit LS. This command will read or write data into the local store locations.

```
MIC>EX LS AAAA
   where AAAA is an LS address.

MIC>DE LS AAAA DDDD
   where DDDD is data.
```

TB

Examine/deposit TB. This command will read or write data into the memory controller's translation buffer.

NOTE: This command requires that a WCS based micro diagnostic be loaded in WCS RAM.

MIC>EX TB AAAA
where AAAA is translation buffer address range is 0-7F.

MIC>DE TB AAAA DDDD
where DDDD is data and bits 31-23 and bit 0 cannot be written into.

MM

Examine/deposit MM. This command will read or write data into the CPU's main memory.

NOTE: This command requires that a WCS based micro diagnostic be loaded in WCS RAM.

MIC>EX MM AAAA
where AAAA is a main memory address.

MIC>DE MM AAAA DDDD
where DDDD is data.

OS

Examine/deposit OS. This command will read or write 8 bits of data into the CPU OS register.

MIC>EX OS
MIC>DE OS DDDD
where DDDD is data.

ICSR

Examine/deposit ICSR. This command will read or write data into the IDC's CSR register.

NOTE: This command requires that a WCS based micro diagnostic be loaded in WCS RAM.

MIC>EX IC
MIC>DE IC DDDD
where DDDD is data and bits 0,4,5,10-21,26,29-31 cannot be written into.

IDAR

Examine/deposit IDAR. This command will read or write data into the IDC's Disk Address Register.

NOTE: This command requires that a WCS based micro diagnostic be loaded in WCS RAM.

```
MIC>EX ID
MIC>DE ID DDDD
   where DDDD is data and bits 20-31 cannot be written into.
```

DBUF

Examine/deposit DBUF. This command will get or put data into the IDC's DATA BUFFER. Note that the DBUF is addressable. The address examined or deposited into comes from the currently selected FIFO (see SE AF and SE BF above). Also note that the address in the selected FIFO register is automatically incremented after an examine or deposit command. This feature can not be suppressed. Thus to deposit and examine the same location requires that the address reg first be cleared (see CL FI above) before the deposit and then cleared again before the examine. Any address can be accessed by repeated EX or DE which increments the address after execution.

NOTE: This command requires that a WCS based micro diagnostic be loaded in WCS RAM.

```
MIC>EX DB
MIC>DE DB DDDD
   where DDDD is data.
```

PATT

Examine PATT. This command will get data from the IDC's ECC pattern register.

** ONLY EXAMINE AVAILABLE FOR THIS COMMAND **
NOTE: This command requires that a WCS based micro diagnostic be loaded in WCS RAM.

```
MIC>EX PA
```

POSIT

Examine POSIT. This command will get data from the IDC's ECC position register.

** ONLY EXAMINE AVAILABLE FOR THIS COMMAND **
NOTE: This command requires that a WCS based micro diagnostic be loaded in WCS RAM.

```
MIC>EX PO
```

CONTINUE

This command causes the diagnostic to continue execution from the last error halt, SOMM, single step termination, ^C, or ^P.

```
MIC>CO
MIC>C
```

DIR

This command prints a directory of the TU58 being currently used on the console terminal.

```
MIC>DIR
```

S/U

This command starts the 8085 at the RAM or ROM address specified. It can be used to initiate a power up or for certain debug techniques (such as starting after a breakpoint). APT will use this command to start test execution.

```
MIC>S/U AAAA
   where AAAA is the ROM or RAM address to start at.
```

T/E

This command operates like a RETURN or S/U 0 in that the self test is executed due to a power-up.

```
MIC>T/E
```

RETURN

This command is used to return control to the standard console. It forces the 8085 to start at ROM address 0.

R
-

This command is used to repeat a command, such as examine or deposit. It must be included with the original command and can only be typed in as "R". If "RE" is typed, a RETURN will be executed. The repeat can be stopped by C or P. It forces the 8085 to start at ROM address 0.

```
MIC>R EX RA 0
```

LOOP

This command is used to loop on error after halt on error has occured. It sets the NER and LOOP flags, and clears the HALT and TRACE flags. LOOP typed after a halt on error has occurred provides a slightly shorter error loop than setting the flags individually and executing the test. It is also more convenient for the operator to use. Disable by typing SE DE (SEt DEfault) after exiting the loop via a ^C or ^P.

LD
--

Load section (diagnostic) from tape. MICMON looks in it's internal tables to find the section name and then loads the section in WCS or in 8085 RAM as indicated in the table. If MICMON can not find the name in the table it assumes that the section is to be loaded in WCS.

```
MIC>LD XXXXX
   where XXXXX is the filename (without the extension).
```

INIT

This command is used to initialize a WCS section. It loads constants used by the WCS tests into LS and main memory (if required). It must be executed before a WCS based micro-diagnostic is executed the first time (after a LD command) or whenever it is suspected that main memory or LS has been clobbered. It is automatically executed on a DI SE or DI command.

```
MIC>IN
MIC>I
```

ABORT

This command is for APT use. It will do any housekeeping necessary to be ready for a new load via the X command.

X/C or X/U

This command is only used by APT/RD to downline laod a file into 8085 RAM or WCS RAM. It is never executed from a console terminal. See the Midrange Console Spec or the APT Interface spec for more information.

MICMON ERROR CODE LISTING

HEX 01 = BOARD NAME NOT FOUND
HEX 02 = NO TEST NUMBER FOUND
HEX 03 = NO PASS COUNT FOUND
HEX 04 = CONTINUE NOT AVAILABLE AT THIS POINT

HEX 10 = NO ADDRESS OR NO DATA INPUT WITH COMMAND
HEX 11 = NO WCS IMAGE FOR EXAM OR DEPOSIT SUBROUTINE

HEX 22 = EXAMINE OR DEPOSIT MAIN MEMORY REPORTS ERROR SUM**
HEX 23 = SEQUENCE ERROR IN WCS TESTING OR NONEXISTENT TEST
HEX 24 = TIMEOUT ERROR IN WCS TESTING
HEX 25 = TU58 ERROR
HEX 26 = PARITY ERROR IN WCS
HEX 27 = CHECKSUM ERROR IN MICMON
HEX 28 = DI TE ATTEMPTED WITH POSSIBLY INVALID TEST CODE LOADED
(DO DI SE OR DI BO)
HEX 29 = UPC DOES NOT MATCH ON VERIFY OF 32 BIT DATA WRITE
HEX 2A = INIT DONE WITHOUT FILE LOADED

HEX 30 = CHECKSUM ERROR IN X COMMAND
HEX 31 = NO X COMMAND ADDRESS FOUND
HEX 32 = NO X COMMAND COUNT FOUND

**WILL RESULT IF MEM REQ CANNOT BE ASSERTED. RUN ENKCC, THE MCT MICRO.

5.1 Options

Not applicable

5.2 Event Flags

Not applicable

5.3 APT Setup

The micro-monitor knows when APT is present by the position of the keyswitch and the state of a line connected to the APT-RD port. It sets a flag in the CPU Local Store which can be read by diagnostics if necessary.

Under APT, all diagnostics will halt on error and report error information to APT via the mailbox in 8085 RAM. The monitor handles this information and puts it in the proper location.

6.0 PROGRAM FUNCTIONAL DESCRIPTION

6.1 Program Overview

On startup MICMON sets the stack and calculates a checksum on itself. This checksum is recalculated and verified each time the prompt is given. If the checksum is not the same as it was on power up, then an error message is given. The version is then printed and flags are initialized. MICMON then goes to the parser.

The parser gets an input line and dispatches from a table to the various subroutines according to the command typed. If one of the subroutines sets the flag called EXECUTE then MICMON drops out of the parser and into the mainline.

The mainline loops on itself until the EXECUTE flag is cleared. The mainline accesses a table of section names (micro-diagnostic filenames) to perform its work. There are three modes of operation. The first mode is not to use the table at all. The second mode is to select only the entries in the table that belong to a specified board (module). The third mode is to select all non-optional sections in the table.

Once a section is selected it is then loaded into the proper address space (See loading). After the section is loaded, the section is started. If the section was loaded into the 8085 address space, control is transfered to this section when the diagnostic is started. If the section was loaded into the WCS address space, MICMON goes to a subroutine that monitors the execution of this section.

When a section is started it may be started at a specific test number. In the case of the WCS tests, MICMON starts the WCS micro code at the address that is equal to the test number. For example, start at address 5 to execute test 5. This indexes into a table stored in the WCS. In the case of an 8085 based test MICMON branches to the first test. The begining of test logic skips each test until the starting test is reached.

When a section name is specified to be loaded, MICMON checks to see if this section is in the internal table. If the section is in the table then MICMON uses the information there to determine where the section is to be loaded. If the section is not found MICMON assumes it is to be loaded into the WCS area.

When a WCS section is loaded, MICMON single steps the CPU through a routine that shifts the working regs left and compares each address returned with an internal table of addresses. If ALL addresses compare correctly, then a flag is set stating that this WCS subroutine can be used to load constants or data into the Local Store or main memory. If the addresses do not compare, then MICMON shifts instructions into the CSR one at a

time and writes the data without the of this subroutine. This method is referred to as a SLOW WRITE TO LS and takes 6 times longer to complete than using the WCS based subroutine.

6.2 Program Size

This program is about 10K bytes in length and loads at RAM address 4000(H). It must not extend above address 7000(H), as this area is reserved for 8085 based micro-diagnostics.

6.3 Program Run Times

Not Applicable.

6.4 Run-time Dynamics

Not Applicable.

6.5 Fault Detection

The monitor will report errors with the following header:

```
SECT  TST ERR   EXP   REC  OTHER  MSK  MODULE
```

1. SECT is the program name currently being executed.
2. TST is the test number that failed.
3. ERR is the error number that failed.
4. EXP is the expected (correct) data.
5. REC is the received (actual) data.
6. OTHER is any other pertinent data (such as an address). This field is not always used. N/A will be printed under OTHER when not in use. The ERRORS section in the micro-diagnostic listing will describe the contents of this field when it is used.
7. MASK is the error mask used to check the result. Bits set to a 1 in this mask correspond to bits in the result that are not checked.
8. MODULE is the suspected module that caused the failure.

6.6 Performance During Hardware Failures

A power fail will cause a rebooting of the system unless a battery backup is installed. Other failures, such as a timeout or a WCS parity error, will print an error message and return to the MIC> prompt. The operator can then issue other commands, including a continue if desired.

6.7 Program Applications

This program must be used to run any micro-diagnostics. Its application depends on the reason for executing the micro-diagnostics. This includes detection and isolation of hardware faults, and verifying that the machine hardware is operational after installation or repair. It can also be used to verify the "goodness" of the machine on a periodic or as needed basis.

6.8 Test Descriptions

Not Applicable.

7.0 MAINTENANCE HISTORY

DATE	VERSION	DESCRIPTION
8-MAR-82	01.00	Initial release.
6-JUL-82	02.00	Changes to parser, and changes for IDC part II.
24-MAR-83	02.10	Added new IDC sizer, ENKCH. Added ENKCC to DI BO WCS command.

Object Module Synopsis

Module Name	Ident	Bytes	File	Creation Date	Creator
.MAIN.	0	8968	DRB0:[GREEN.CRD]ENKAA.OBJ;3	14-JUN-1983 10:27	VAX-11 Macro V03-00

Program Section Synopsis

Psect Name	Module Name	Base	End	Length	Align	Attributes
MICMON		00004C00	00006F07	00002308 (8968.)	BYTE 0	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC
	.MAIN.	00004C00	00006F07	00002308 (8968.)	BYTE 0	

Symbols By Name

Symbol	Value	Symbol	Value	Symbol	Value	Symbol	Value
APT	00004C38-R	GCVECT	00004129	READ_LS	00004D02-R		
APT_PASS_CNT	00004C18-R	HEX_3	00000003	READ_WCS	00004CE7-R		
CC_ADDR	000000FE	HEX_BUF	00004C5B-R	SET_FOR_CONT	00004D2C-R		
CLEAR_CPU_ATT	00004D2F-R	INC_WORD	00004D08-R	SHIFTER	00004CE1-R		
CNTLC_TYPED	00004C39-R	LD_OPC	00004D17-R	SHIFTER1	00004D32-R		
COMPARE	00004CCF-R	LF	0000000A	SHIFT_L_2901	00004CF6-R		
CONTINUE	00004C3A-R	LOAD_CSR	00004CF0-R	SHIFT_PNT	00004C68-R		
CON_ADD_DAT	00004C3B-R	LOAD_UPC	00004CF3-R	SHIFT_R_2901	00004CF9-R		
CON_BEGIN_TEST	00004D23-R	MAKE_XD	00004D0B-R	SKIP_TEST	00004C6A-R		
CON_ERROR	00004D20-R	MCPU_A	00004D40-R	SPACE	00004CDB-R		
CON_ERR_NUM	00004C3F-R	MICRO_STEP_CPU	00004D38-R	UPC_VALUE	00004CC2-R		
CON_EXP_ADD	00004C44-R	MOVER	00004CD2-R	WCS_ADDRESS	00004C6C-R		
CON_EXP_DAT	00004C40-R	MOVER_3	00004D1A-R	WCS_BAD_PARITY	00004C6E-R		
CON_MOD_DAT	00004C46-R	MOVER_4	00004D11-R	WCS_READ	00004D26-R		
CON_MSK_DAT	00004C48-R	MSK_ERROR	00004D1D-R	WCS_SHIFT	00004C6B-R		
CON_REC_ADD	00004C53-R	MWCS_A	00004D3B-R	WCS_VALUE	00004CB4-R		
CON_REC_DAT	00004C4F-R	NA_EXP_REC	00004C5F-R	WCS_VAL_SHIFT	00004CB3-R		
CR	0000000D	NA_OTHER	00004C60-R	WCS_WRITE	00004D29-R		
CRLF	00004CDE-R	NOP_CSR	00004CE4-R	WRITE_DATA_32	00004CFF-R		
CSR_SHIFT	00004CB7-R	NOP_INST	00004C72-R	WRITE_LS	00004D05-R		
CSR_VALUE	00004CB8-R	OS_ADD	0000007C	WRITE_TAB_PNT	00004C6F-R		
DATA0	00004CA9-R	OVERLAY_RAM	00004C03-R	WRITE_WCS	00004CEA-R		
DATA1	00004CAA-R	OVERLAY_WCS	00004C06-R	XD_ADDRS	00004C71-R		
DATA2	00004CAB-R	PARITY_JUMP	00004C61-R	X_CLR_PAGE	00004C7D-R		
DATA3	00004CAC-R	PARITY_ON	00004C64-R	X_CLR_WR0	00004CA1-R		
DATA_SHIFT	00004CA8-R	PERFORM_CSR	00004CED-R	X_COM_WR0	00004CA5-R		
DECR_WORD	00004D35-R	POWER_BRANCH	00004C65-R	X_MOV_WRWR	00004C9E-R		
DOLOOP	00004C55-R	PRINT	00004CD5-R	X_ROT_L	00004C79-R		
EOS_FLG	00004C56-R	PRINT_HEX	00004CD8-R	X_SET_PAGE	00004C81-R		
ERROR_CON	00004C57-R	PRINT_HEX_1	00004D14-R	X_SHIFT_L	00004C75-R		
FLAGS	00004C5A-R	PRINT_STRING	00004D0E-R	X_SHIFT_R	00004C85-R		
F_WORD	00004C58-R	READ_DATA_32	00004CFC-R				

Symbol	Value	Symbol	Value	Symbol	Value	Symbol	Value

Key for special characters above:

*	- Undefined
U	- Universal
R	- Relocatable
X	- External

Image Synopsis

```
Virtual memory allocated:                   00004C00 00006FFF 00002400 (9216. bytes, 18. pages)
Stack size:                                        C. pages
Image binary virtual block limits:                 1.       18. (   18. blocks)
Image name and identification:              ENKAA 0
Number of files:                                   1.
Number of modules:                                 1.
Number of program sections:                        3.
Number of global symbols:                         95.
Number of image sections:                          1.
Image type:                                 SYSTEM.
Map format:                                 DEFAULT in file DRB0:[GREEN.CRD]ENKAA.MAP;3
Estimated map length:                       15. blocks
```

Link Run Statistics

Performance Indicators	Page Faults	CPU Time	Elapsed Time
Command processing:	38	00:00:00.09	00:00:00.88
Pass 1:	34	00:00:00.24	00:00:00.78
Allocation/Relocation:	5	00:00:00.07	00:00:01.66
Pass 2:	19	00:00:02.26	00:00:09.44
Map data after object module synopsis:	13	00:00:00.24	00:00:00.24
Symbol table output:	1	00:00:00.05	00:00:00.54
Total run values:	110	00:00:02.95	00:00:13.54

```
Using a working set limited to 300 pages and 30 pages of data storage (excluding image)

Total number object records read (both passes):   154
    of which 0 were in libraries and 2 were DEBUG data records containing 61 bytes

Number of modules extracted explicitly              = 0
    with 0 extracted to resolve undefined symbols

0 library searches were for symbols not in the library searched

A total of 7 global symbol table records was written

/SYST=%X4C00/EXE=ENKAA.EXE/SYMBOL=ENKAA ENKAA.OBJ
```

```
ZZ-ENKAA-2.1    4C00    4
4C00     3
4C00     4
4C00     5
4C00     6
4C00     7
4C00     8
4C00     9
4C00    10
4C00    11
4C00    12
4C00    13
4C00    14
4C00    15
4C00    16
4C00    17
4C00    18
4C00    19
4C00    20
4C00    21
4C00    22
4C00    23
4C00    24
4C00    25
4C00    26
4C00    27
4C00    28
4C00    29
4C00    30
4C00    31
4C00    32
4C00    33
4C00    34
4C00    35
4C00    36
4C00    37
4C00    38
4C00    39
4C00    40
4C00    41
4C00    42
4C00    43
4C00    44
4C00    45
4C00    46
4C00    47
4C00    48
4C00    49
4C00    55
4C00    56
4C00    57
4C00    58
4C00    59
4C00    60
4C00    61
4C00    62
4C00    63
4C00    64
4C00    65
4C00    66
4C00    67
```

```
.TITLE  'ENKAA  8085 based micro-monitor for micro-diagnostics Rev 02.10'
;
;
;                       COPYRIGHT (c) 1982 BY
;             DIGITAL EQUIPMENT CORPORATION, MAYNARD,
;              MASSACHUSETTS.  ALL RIGHTS RESERVED.
;
; THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY  BE  USED  AND  COPIED
; ONLY  IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE INCLUSION
; OF THE ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE OR ANY OTHER COPIES THEREOF
; MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY OTHER  PERSON.  NO
; TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY TRANSFERRED.
;
; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE, AND
; SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT CORPORATION.
;
; DIGITAL  ASSUMES  NO  RESPONSIBILITY  FOR  THE  USE OR RELIABILITY OF ITS
; SOFTWARE ON EQUIPMENT THAT IS NOT SUPPLIED BY DIGITAL.
;
;++
; FACILITY:     VAX-11/730 MICRO-DIAGNOSTIC MONITOR
;
; ABSTRACT:     This 8085 based program monitors and controls the execution
;               of both 8085 based micro-diagnostics and WCS based micro-
;               diagnostics.
;
; ENVIRONMENT:  Standalone
;
; AUTHOR:       Ernest Preisig          1-OCT-81
;
; MODIFIED BY:  David Mayo              8-MAR-82        VERSION 01.00
;               Initial release.
;
;               David Mayo              6-JUL-82        VERSION 02.00
;               Changes for IDC Part II micro diagnostic.
;               RET_VECTOR established as the return vector to the Console.
;               Modified parsing tables and parser to save space.
;               Added compacting routine to INPUT_LINE so that parser will
;               accept multiple spaces/special characters between commands.
;
;               Peter Green             14-JUN-83       VERSION 02.10
;               Added new diagnostic, ENKCH.
;               ENKCC added on DI BO WCS command.
;
;--
```

```
4C00   68
4C00   69
4C00   70
4C00   71
4C00   72
4C00   73
4C00   74
4C00   75
4C00   76
4C00   77
4C00   78
4C00   79
4C00   80
4C00   81
4C00   82
4C00   83
4C00   84
4C00   85
4C00   86
4C00   87
4C00   88
4C00   89
4C00   90
4C00   91
4C00   92
4C00   93
4C00   94
4C00   95
4C00   96
4C00   97
4C00   98
4C00   99
4C00  100
4C00  101
4C00  102
4C00  103
4C00  104
4C00  105
4C00  106
4C00  107
4C00  108
4C00  109
4C00  110
4C00  111
4C00  112
4C00  113
4C00  114
4C00  115
4C00  116
4C00  117
4C00  118
4C00  119
4C00  120
4C00  121
4C00  122
4C00  123
4C00  124
4C00  125
4C00  126
4C00  127
```

;8085 SOURCE CODE MICRO DIAGNOSTIC MONITOR

```
;*****************************************************************************
;
;       NOTE:  THE LAST ADDRESS OF THIS PROGRAM MUST NOT EXCEED 6FFF.  THE
;       8085 MICRO-DIAGNOSTIC OVERLAYS LOAD AT ADDRESS 7000 HEX!!!
;
;*****************************************************************************


;*****************************************************************************
;
;       THIS PROGRAM WAS DESIGNED USING PROGRAM DESIGN LANGUAGE (PDL).
;  SINCE PDL IS A HIGH LEVEL LANGUAGE THE DECISION TO WRITE THE MONITOR
;  USING MACROS WAS MADE.  THE MACROS USE MANY OF THE REGISTERS, SO USE
;  OF REGISTERS FOR PASSING PARAMETERS IS DANGEROUS AT BEST.  ALSO THERE ARE
;  DIFFERENCES BETWEEN THE PDL AND THE PROGRAM.  PDL IS MEANT AS A DESIGNERS
;  LANGUAGE AND IS USED TO PASS INFORMATION TO THE IMPLIMENTOR.  THE USE OF
;  THE MACROS HELPS KEEP THE TWO SIMILAR, BUT IMPLEMENTATION IN A SLIGHTLY
;  DIFFERENT MANNER IS SOMETIMES NECESSARY.
;
;*****************************************************************************

;*****************************************************************************
;  IMPLEMENTATION NOTES:
;
;       THIS PROGRAM IS CALLED MICMON (MICRO MONITOR).  MICMON  IS USED TO
; RUN MICRO DIAGNOSTICS AND PERFORM OPERATING SYSTEM TYPE FUNCTIONS FOR
; THE MICRO DIAGNOSTICS.  MICMON RUNS TWO DIFFERENT FORMS OF DIAGNOSTICS
;       A) DIAGNOSTICS WRITTEN IN CPU MICRO CODE.
;       B) DIAGNOSTICS WRITTEN IN 8085 CODE.
;
;       MICMON PASSES CONTROL TO 8085 DIAGNOSTICS BY CALLING THEM
; AS IF THEY WERE SUBROUTINES OF THE MICRO MONITOR.
;
;       MICMON CAUSES CPU MICRO CODE DIAGNOSTICS TO BE EXECUTED. THIS IS
; DONE BY STARTING THE CPU AT SPECIFIC ADDRESSES. (THESE VARIOUS ADDRESSES
; ARE CONTAINED IN THE CONSTANTS LISTED IN THE CONSTANT SECTION) THE CPU
; MICRO DIAGNOSTICS RUN CONCURRENT TO THE 8085.  THE CPU MICRO CODED
; DIAGNOSTIC COMMUNICATES WITH MICMON BY SETTING THE CPU ATTENTION
; SIGNAL (HARDWARE) AND CONVEYING IT'S MESSAGE THROUGH THE USE OF A
; 32 BIT COMMAND AND STATUS WORD.
;
;  VARIABLE NAMES ENDING IN _A ARE ASCII TEXT
;*****************************************************************************


;*****************************************************************************
;
;  MICMON ERROR CODE LISTING
;
;       HEX 01  =  BOARD NAME NOT FOUND
;       HEX 02  =  NO TEST NUMBER FOUND
;       HEX 03  =  NO PASS COUNT FOUND
;       HEX 04  =  CONTINUE NOT AVAILABLE AT THIS POINT
```

```
4C00  128
4C00  129                       ;       HEX 10  =  NO ADDRESS OR NO DATA INPUT WITH COMMAND
4C00  130                       ;       HEX 11  =  NO WCS IMAGE FOR EXAM OR DEPOSIT SUBROUTINE
4C00  131                       ;
4C00  132                       ;       HEX 22  =  EXAMINE OR DEPOSIT MAIN MEMORY REPORTS ERROR SUM**
4C00  133                       ;       HEX 23  =  SEQUENCE ERROR IN WCS TESTING OR NONEXISTENT TEST
4C00  134                       ;       HEX 24  =  TIMEOUT ERROR IN WCS TESTING
4C00  135                       ;       HEX 25  =  TU58 ERROR
4C00  136                       ;       HEX 26  =  PARITY ERROR IN WCS
4C00  137                       ;       HEX 27  =  CHECKSUM ERROR IN MICMON
4C00  138                       ;       HEX 28  =  DI TE ATTEMPTED WITH POSSIBLY INVALID TEST CODE LOADED
4C00  139                       ;                  (DO DI SE OR DI BO)
4C00  140                       ;       HEX 29  =  UPC DOES NOT MATCH ON VERIFY OF 32 BIT DATA WRITE
4C00  141                       ;       HEX 2A  =  INIT DONE WITHOUT FILE LOADED
4C00  142                       ;
4C00  143                       ;       HEX 30  =  CHECKSUM ERROR IN X COMMAND
4C00  144                       ;       HEX 31  =  NO X COMMAND ADDRESS FOUND
4C00  145                       ;       HEX 32  =  NO X COMMAND COUNT FOUND
4C00  146                       ;
4C00  147                       ;       ** WILL RESULT IF MEM REQ CANNOT BE ASSERTED.  RUN ENKCC, THE MCT MICRO
4C00  148                       ;*************************************************************************
4C00  149
4C00  150                       ;*************************************************************************
4C00  151                       ;
4C00  152                       ; DATA SECTION
4C00  153                       ;
4C00  154                       ;*************************************************************************
4C00  155
4C00  156                                       .PSECT  MICMON
4C00  157 000
4C00  158
4C00  159 00000000              HEAD            = 0                     ;CALCULATE NO OFFSET IN ASSEMBLY
4C00  160
4C00  161                       ;ENTRY POINT TO THE PROGRAM IS FIRST ADDRESS OF OVERLAY
4C00  162
4C00  163 1F6D'C3                               JMP     START_UP
4C03  164
4C03  165
4C03  166 00000000                              EQUATE                  ;REGISTER EQUATE MACRO
          00000002
          00000004
          00000006
          00000006
          00000001
          00000003
          00000005
          00000007 00000006
4C03  167
4C03  168
4C03  169                       ;*************************************************************************
4C03  170                       ;
4C03  171                       ;  THE FOLLOWING BLOCK OF MACROS DEFINE THE HARDWARE INTERFACE TO
4C03  172                       ;  THE 8085 SYSTEM.
4C03  173                       ;
4C03  174                       ;*************************************************************************
4C03  175
4C03  176
4C03  177 00000000              UPC14A          =<^X00>                 ;(RO) UPC BIT 14 ASSERTED HIGH <MSB>
4C03  178
4C03  179 00000001              MISC2           =<^X01>
```

```
4C03  180 00000001                BOOTF       =<^X01>     ;(RO) BOOT FLAG. ASSERTED LOW. <LSB>
4C03  181
4C03  182 00000002                DCLO        =<^X02>     ;(RO) DC LOW FLAG ASSERTED HIGH<MSB>
4C03  183 00000002                HLTFLG      =<^X02>     ;(RO) HALT FLAG ASSERTED LOW   <LSB>
4C03  184
4C03  185 00000003                READJ1      =<^X03>     ;(RO) JUMPER J1 FLAG ASS. HIGH <MSB>
4C03  186 00000003                ALLOWP      =<^X03>     ;(RO) ENABLE CONTROL P ASS LOW <LSB>
4C03  187
4C03  188 00000004                READJ2      =<^X04>     ;(RO) BIT 07 EQUALS J2 <MSB>
4C03  189 00000004                APTFLG      =<^X04>     ;(RO) APT PRESENT FLAG <LSB>
4C03  190
4C03  191 00000005                AUTO        =<^X05>     ;(RO) AUTO TEST FLAG ASS LOW <LSB>
4C03  192
4C03  193 00000006                REMOTE      =<^X06>     ;(RO) REMOTE FLAG ASS LOW <LSB>
4C03  194
4C03  195 00000007                OKV15       =<^X07>     ;(RO) 15 VOLTS OK FLAG ASS LOW <MSB>
4C03  196 00000007                BOOTEN      =<^X07>     ;(RO) BOOT ENABLE FLAG ASS LOW <LSB>
4C03  197
4C03  198 00000008                WCSB0       =<^X08>     ;(WO) WCS WRITE DATA REG, BYTE 0
4C03  199 00000009                WCSB1       =<^X09>     ;(WO) WCS WRITE DATA REG, BYTE 1
4C03  200 0000000A                WCSB2       =<^X0A>     ;(WO) WCS WRITE DATA REG, BYTE 2
4C03  201
4C03  202 0000001C                ITDB        =<^X1C>     ;(R/W) TIMER DATA REGISTER
4C03  203 0000001D                ITCSR       =<^X1D>     ;(R/W) TIMER STATUS REGISTER
4C03  204
4C03  205 00000020                CLRCLK      =<^X20>     ;(WO) CLEAR CPU CLOCK RUN
4C03  206 00000021                SETCLK      =<^X21>     ;(WO) SET CPU CLOCK RUN
4C03  207
4C03  208 00000022                CLRSS       =<^X22>     ;(WO) CLEAR CLOCK SINGLE STEP
4C03  209 00000023                SETSS       =<^X23>     ;(WO) SET CLOCK SINGLE STEP
4C03  210
4C03  211 00000024                CLRCSK      =<^X24>     ;(WO) CLEAR CSR CLOCK
4C03  212 00000025                SETCSK      =<^X25>     ;(WO) SET CSR CLOCK
4C03  213
4C03  214 00000026                CLRUPK      =<^X26>     ;(WO) CLEAR THE UPC CLOCK
4C03  215 00000027                SETUPK      =<^X27>     ;(WO) SET THE UPC CLOCK
4C03  216
4C03  217 00000028                SETMCI      =<^X28>     ;(WO) SET MEMORY CONTROL INIT
4C03  218 00000029                CLRMCI      =<^X29>     ;(WO) CLEAR MEMORY CONTROL INIT
4C03  219
4C03  220 0000002A                SETMCK      =<^X2A>     ;(WO) SET MEMORY CONTROL CLOCK
4C03  221 0000002B                CLRMCK      =<^X2B>     ;(WO) CLEAR MEMORY CONTROL CLOCK
4C03  222
4C03  223 0000002C                CLRTMR      =<^X2C>     ;(WO) STOP TIMER
4C03  224 0000002D                SETTMR      =<^X2D>     ;(WO) START TIMER COUNTING
4C03  225
4C03  226 0000002E                CLRBSY      =<^X2E>     ;(WO) CLEAR UNIBUS BUS BUSY
4C03  227 0000002F                SETBSY      =<^X2F>     ;(WO) SET UNIBUS BUS BUSY
4C03  228
4C03  229 00000030                SETDCL      =<^X30>     ;(WO) SET DC LOW
4C03  230 00000031                CLRDCL      =<^X31>     ;(WO) CLEAR DC LOW
4C03  231
4C03  232 00000032                SETACL      =<^X32>     ;(WO) SET AC LOW
4C03  233 00000033                CLRACL      =<^X33>     ;(WO) CLEAR AC LOW
4C03  234
4C03  235 00000034                INITL       =<^X34>     ;(WO) CLEAR UNIBUS INIT
4C03  236 00000035                INITH       =<^X35>     ;(WO) SET UNIBUS INIT
4C03  237
4C03  238 00000036                CLRWCK      =<^X36>     ;(WO) CLEAR WCS WRITE CLOCK
4C03  239 00000037                SETWCK      =<^X37>     ;(WO) SET WCS WRITE CLOCK
```

```
4C03   240
4C03   241 00000038                    CLRCSR          =<^X38>         ;(WO) CLEAR CSR SERIAL INPUT
4C03   242 00000039                    SETCSR          =<^X39>         ;(WO) SET CSR SERIAL INPUT
4C03   243
4C03   244 0000003C                    CLRLIT          =<^X3C>         ;(WO) CLEAR THE RUN LIGHT
4C03   245 0000003D                    SETLIT          =<^X3D>         ;(WO) SET THE RUN LIGHT
4C03   246
4C03   247 00000044                    TUDB            =<^X44>         ;(R/W) TU-58 DATA BUFFER
4C03   248 00000045                    TUSR            =<^X45>         ;(RO)  TU-58 STATUS REGISTER
4C03   249 00000046                    TUMODE          =<^X46>         ;(R/W) TU-58 MODE REGISTER 1 + 2
4C03   250                                                             ;FIRST ACCESS GET MR1,SECOND GETS MR2
4C03   251                                                             ;READING THE CR FORCES THE INTERNAL
4C03   252                                                             ;POINTER BACK TO MR1
4C03   253 00000047                    TUCR            =<^X47>         ;(R/W) TU-58 COMMAND REGISTER
4C03   254
4C03   255 00000048                    TTDB            =<^X48>         ;(R/W) TERMINAL DATA BUFFER
4C03   256 00000049                    TTSR            =<^X49>         ;(RO)  TERMINAL STATUS REGISTER
4C03   257 0000004A                    TTMODE          =<^X4A>         ;(R/W) TERMINAL MODE REGISTER 1 + 2
4C03   258                                                             ;FIRST ACCESS GET MR1,SECOND GETS MR2
4C03   259                                                             ;READING THE CR FORCES THE INTERNAL
4C03   260                                                             ;POINTER BACK TO MR1
4C03   261 0000004B                    TTCR            =<^X4B>         ;(R/W) TERMINAL COMMAND REGISTER
4C03   262
4C03   263 00000080                    READ            =<^X80>         ;(RO) CONSOLE READ REGISTER
4C03   264
4C03   265 00000082                    CPUACK          =<^X82>         ;(RO) ACKNOWLEDGE FROM CPU <LSB>
4C03   266
4C03   267 00000083                    CPATTN          =<^X83>         ;(RO) ATTENTION FROM CPU <LSB>
4C03   268
4C03   269 00000084                    UPC14           =<^X84>         ;(RO) UPC BIT 14        <LSB>
4C03   270
4C03   271 00000085                    CSR07           =<^X85>         ;(RO) CSR BIT 7         <LSB>
4C03   272
4C03   273 00000086                    CSR15           =<^X86>         ;(RO) CSR BIT 15        <LSB>
4C03   274
4C03   275 00000087                    CSR23           =<^X87>         ;(RO) CSR BIT 23        <LSB>
4C03   276
4C03   277 000000A0                    ENBPAR          =<^XA0>         ;(WO) ENABLE STALL ON PARITY ERROR
4C03   278 000000A1                    DISPAR          =<^XA1>         ;(WO) DISABLE STALL ON PARITY ERROR
4C03   279
4C03   280 000000A2                    VSHIFT          =<^XA2>         ;(WO) SET THE V-BUS TO SHIFT MODE
4C03   281 000000A3                    VNORML          =<^XA3>         ;(WO) SET THE V-BUS TO NORMAL MODE
4C03   282
4C03   283 000000A4                    SETTIM          =<^XA4>         ;(WO) SET INTERVAL TIMER INTERRUPT
4C03   284 000000A5                    CLRTIM          =<^XA5>         ;(WO) CLEAR INTERVAL TIMER INTRPT.
4C03   285
4C03   286 000000A6                    ENBMEM          =<^XA6>         ;(WO) ENABLE MEMORY REFFRENCES
4C03   287 000000A7                    DISMEM          =<^XA7>         ;(WO) DISABLE MEMORY REFERENCES
4C03   288
4C03   289 000000A8                    SETACK          =<^XA8>         ;(WO) SET CONSOLE ACKNOWLEDGE
4C03   290 000000A9                    CLRACK          =<^XA9>         ;(WO) CLEAR CONSOLE ACKNOWLEDGE
4C03   291
4C03   292                             ;NOTE:  THE CONSOLE ACKNOWLEDGE SIGNAL IS ALSO USED AS THE UPC
4C03   293                             ;SERIAL INPUT.  WHEN USED THIS WAY THE SENSE IS INVERTED.  TO SIMPLIFY
4C03   294                             ;MATTERS THE NEXT TWO EQUATES ARE INCLUDED
4C03   295
4C03   296 000000A8                    CLRUPC          =<^XA8>         ;(WO) CLEAR THE V-BUS SERIAL INPUT
4C03   297 000000A9                    SETUPC          =<^XA9>         ;(WO) SET THE V-BUS SERIAL INPUT
4C03   298
4C03   299 000000AA                    SETATN          =<^XAA>         ;(WO) SET CONSOLE ATTENTION
```

```
4C03  300 000000AB                CLRATN          =<^XAB>             ;(WO) CLEAR CONSOLE ATTENTION
4C03  301
4C03  302 000000AC                SETPFI          =<^XAC>             ;(WO) SET POWER FAIL INTERRUPT
4C03  303 000000AD                CLRPFI          =<^XAD>             ;(WO) CLEAR POWER FAIL INTERRUPT
4C03  304
4C03  305 000000AE                SETHLT          =<^XAE>             ;(WO) SET THE CONSOLE HALT BIT
4C03  306 000000AF                CLRHLT          =<^XAF>             ;(WO) CLEAR THE CONSOLE HALT BIT
4C03  307
4C03  308 000000CC                WRITE           =<^XCC>             ;(WO) THE CONSOLE WRITE REGISTER
4C03  309
4C03  310 000000EC                YBUSRD          =<^XEC>             ;(RO) Y-BUS READ
4C03  311
4C03  312
4C03  313                         ;*************************************************************************
4C03  314                         ;
4C03  315                         ;  THE FOLLOWING EQUATES ARE USED AS PROGRAM CONSTANTS
4C03  316                         ;  THIS IS DONE WHERE EVER POSSIBLE TO CONTROL THE USAGE OF THESE
4C03  317                         ;  CONSTANTS -- SO THEY ARE DEFINED ONLY IN ONE PLACE --
4C03  318                         ;
4C03  319                         ;*************************************************************************
4C03  320
4C03  321 00000001                ADDR            = <^X1>             ;BIT 24 COMMAND + STATUS WORD
4C03  322 00000049                APT_PARAM       = <^X49>            ;LS LOCATION FOR APT PARAMETERS
4C03  323 00000080                BEG_TST         = <^X80>            ;BIT 15 COMMAND + STATUS WORD
4C03  324 00004082                BYTSUM          = <^X4082>          ;ADDRESS OF CHECKSUM IN BOOT BLOCK
4C03  325 000000FE                CC_ADDR         = <^XFE>            ;LS ADDR OF ALU CC REG
4C03  326 00000008                CLEARPAR_ERR    = <^X8>             ;BIT 27 COMMAND + STATUS WORD
4C03  327 00000010                CONATT          = <^X10>            ;BIT 20 COMMAND + STATUS WORD
4C03  328 00000020                CONACK          = <^X20>            ;BIT 21 COMMAND + STATUS WORD
4C03  329 00000002                CONHLT          = <^X2>             ;BIT 17 COMMAND + STATUS WORD
4C03  330 00000004                CON_LOOP        = <^X4>             ;BIT 10 COMMAND + STATUS WORD
4C03  331 00000080                CON_RAM         = <^X80>            ;BIT 7 ON = SECT FOR CONSOLE RAM
4C03  332 00007003                CON_VER_ADD     = <^X7003>          ;ADDRESS IN 8085 BASED TEST WHERE
4C03  333                                                             ; VERSION NUMBER IS FOUND (2 BYTES)
4C03  334 00007005                CON_NAME_ADD    = <^X7005>          ;ADDRESS IN 8085 BASED TEST WHERE
4C03  335                                                             ; FILE NAME IS FOUND (LAST 2 BYTES)
4C03  336 0000000D                CR              = <^X0D>            ;HEX VALUE OF CR
4C03  337 000040FE                CTLDIS          = <^X40FE>          ;BOOT BLOCK DISABLE CNTL CHAR FLAG
4C03  338 0000411A                DIR_VECTOR      = <^X411A>          ;ADDRESS OF DIR COMMAND IN BOOT BLOCK
4C03  339 0000002E                DOT_A           = <^X2E>            ;ASCII FOR DOT
4C03  340 00000040                EMEMREF         = <^X40>            ;BIT 22 COMMAND + STATUS WORD
4C03  341 00000040                EOS             = <^X40>            ;BIT 14 COMMAND + STATUS WORD
4C03  342 00007000                EXEC_CONTESTS   = <^X7000>          ;BEGINNING OF TESTS = RAM LOAD ADDRESS
4C03  343 0000008A                EXPECT_LAB      = <^X8A>            ;LS LOCATION OF FIRST BYTE OF EXPECTED
4C03  344                                                             ; RL02 PACK LABEL
4C03  345 00000048                FLAG_LOC        = <^X48>            ;LS LOCATION OF FLAG INFO TO WRITE
4C03  346 00000010                FPA_PRES        = <^X10>            ;BIT 12 COMMAND + STATUS WORD
4C03  347 00004129                GCVECT          = <^X4129>          ;ADDR OF GET CHAR ROUTINE IN BOOT BLOCK
4C03  348 00004123                GLVECT          = <^X4123>          ;ADDR OF GET LINE ROUTINE IN BOOT BLOCK
4C03  349 00004145                GSVECT          = <^X4145>          ;ADDR OF GET SILO ROUTINE IN BOOT BLOCK
4C03  350 00000001                HEX_1           = <^X1>             ;HEX VALUE OF 1
4C03  351 00000003                HEX_3           = <^X3>             ;HEX VALUE OF 3
4C03  352 00000004                HEX_4           = <^X4>             ;HEX VALUE OF 4
4C03  353 00000007                HEX_7           = <^X7>             ;HEX VALUE OF 7
4C03  354 00000008                HEX_8           = <^X8>             ;HEX VALUE OF 8
4C03  355 00000010                HEX_10          = <^X10>            ;HEX VALUE OF 10
4C03  356 0000000D                HEX_0D          = <^X0D>            ;HEX VALUE OF D
4C03  357 0000001F                HEX_1F          = <^X1F>            ;HEX VALUE OF 1F
4C03  358 00000020                HEX_20          = <^X20>            ;HEX VALUE OF 20
4C03  359 00000021                HEX_21          = <^X21>            ;HEX VALUE OF 21
```

```
4C03  360 0000002D  HEX_2D          = <^X2D>          ;HEX VALUE OF 2D
4C03  361 00000030  HEX_30          = <^X30>          ;HEX VALUE OF 30
4C03  362 0000003A  HEX_3A          = <^X3A>          ;HEX VALUE OF 3A
4C03  363 0000003F  HEX_3F          = <^X3F>          ;HEX VALUE OF 3F
4C03  364 00000040  HEX_40          = <^X40>          ;HEX VALUE OF 40
4C03  365 00000041  HEX_41          = <^X41>          ;HEX VALUE OF 41
4C03  366 00000050  HEX_50          = <^X50>          ;HEX VALUE OF 50
4C03  367 0000007F  HEX_7F          = <^X7F>          ;HEX VALUE OF 7F
4C03  368 00000080  HEX_80          = <^X80>          ;HEX VALUE OF 80
4C03  369 000000FF  HEX_FF          = <^XFF>          ;HEX VALUE OF FF
4C03  370 00000080  IDC_PRES        = <^X80>          ;BIT 31 COMMAND + STATUS WORD
4C03  371 00000008  INVTIM          = <^X8>           ;BIT 19  COMMAND + STATUS WORD
4C03  372 0000000A  LF              = <^XA>           ;HEX VALUE OF LINE FEED
4C03  373 00000005  LS_5            = <^X5>           ;LS LOCATION 5 (ADDRESS)
4C03  374 00000006  LS_6            = <^X6>           ;LS LOCATION 6 (DATA)
4C03  375 00000007  LS_7            = <^X7>           ;LS LOCATION 7 (XFER POINTER)
4C03  376 0000000F  LS_F            = <^XF>           ;LS LOCATION F (T15)
4C03  377 00000047  LOOP_CNT_ADD    = <^X47>          ;ADDRESS OF LOOP COUNT
4C03  378 00000002  LOOPING         = <^X2>           ;BIT 25 COMMAND + STATUS WORD
4C03  379 00000001  MM_OR_LS        = <^X1>           ;MSB = MM XFER
4C03  380 00000008  MM_SIZE         = <^X8>           ;BIT 11 COMMAND + STATUS WORD
4C03  381 00000001  MOD_PRES_FLG    = <^X1>           ;BIT 1 ON = FPA, R80 OR IDC PRESENT
4C03  382 00000080  NA              = <^X80>          ;BIT 23 COMMAND + STATUS WORD
4C03  383 00000080  NEG_CNT         = <^X80>          ;MSB ON FOR NEGATIVE X COM COUNT
4C03  384 000040FF  NOCHK           = <^X40FF>        ;FLAG IN BOOT BLOCK TO IGNORE <CR>
4C03  385 000040BB  OFLAG           = <^X40BB>        ;CONTROL O FLAG IN BOOT BLOCK
4C03  386 00000040  OPTIONAL_TST    = <^X40>          ;BIT 6 OF SECTION FLAG
4C03  387 0000007C  OS_ADD          = <^X7C>          ;LS ADDRESS OF OS REGISTER
4C03  388 0000001B  POWER_SEQ       = <^X1B>          ;ADDRESS OF POWER UP SEQ IN ROM
4C03  389 00004080  PRI_STACK       = <^X4080>        ;STACK AREA IN BOOT BLOCK (128 BYTES)
4C03  390 00000004  PWRFAIL         = <^X4>           ;BIT 18 COMMAND + STATUS WORD
4C03  391 00000080  R80_PRES        = <^X80>          ;BIT 7 COMMAND + STATUS WORD
4C03  392 00007000  RAM_LOAD_ADD    = EXEC_CONTESTS   ;ADDRESS RAM OVERLAYS LOAD AT.
4C03  393 0000008E  RECEIVE_CAB     = <^X8E>          ;LS ADDRESS OF FIRST BYTE OF RECEIVED
                                                      ; RL02 PACK LABEL
4C03  394
4C03  395 0000414D  RET_VECTOR      = <^X414D>        ;RETURN VECTOR TO CONSOLE
4C03  396 0000000B  ROM_X_ADDRS     = <^XB>           ;ROM ADDRESS OF X COMMAND
4C03  397 0000413D  RSVECT          = <^X413D>        ;ADDR OF READ SILO ROUTINE IN BOOT BLOCK
4C03  398 00000006  SEC_LEN         = 6               ;LENGTH OF A SECTION NAME
4C03  399 00004138  SCVECT          = <^X4138>        ;ADDR OF SEND CHAR ROUTINE IN BOOT BLOCK
4C03  400 00000002  SETPAR_ERR      = <^X02>          ;BIT 9 COMMAND + STATUS WORD
4C03  401 00000004  SINGLE_ERR      = <^X4>           ;BIT 26 COMMAND + STATUS WORD
4C03  402 00000001  SIGNAL          = <^X1>           ;BIT 16 COMMAND + STATUS WORD
4C03  403 0000412E  SLVECT          = <^X412E>        ;ADDR OF SEND LINE ROUTINE IN BOOT BLOCK
4C03  404 00004133  SLVEC1          = <^X4133>        ;ADDR OF SPECIAL SEND LINE ROUTINE
4C03  405 00000020  SPACE_A         = <^X20>          ;HEX VALUE OF SPACE
4C03  406 00004086  SPBUFF          = <^X4086>        ;ADDRESS IN BOOT BLOCK WHERE SP STORED
4C03  407 00000000  STNDRD_CON      = 0               ;BRANCH ADDRESS TO LOAD STNDARD CONSOLE
4C03  408 00000040  STATUS          = <^X40>          ;ADDRESS OF COMMAND + STATUS WORD
4C03  409 00004117  TU58_DRIVER     = <^X4117>        ;ADDRESS OF TU58 DRIVER
4C03  410 00000040  UBE_PRES        = <^X40>          ;BIT 6 COMMAND + STATUS WORD
4C03  411 00000040  UBS_BUSY        = <^X40>          ;BIT 30 COMMAND + STATUS WORD
4C03  412 00000020  UBS_DC_LO       = <^X20>          ;BIT 29 COMMAND + STATUS WORD
4C03  413 00000010  UBS_INIT        = <^X10>          ;BIT 28 COMMAND + STATUS WORD
4C03  414 00000001  WCSERR          = <^X1>           ;BIT  8 COMMAND + STATUS WORD
4C03  415 00000044  WCS_ADDR_DAT    = <^X44>          ;ADDRESS OF ADDRESS DATA WORD
4C03  416 00000062  WCS_CL_FIFO     = <^X62>          ;ADDRESS OF CLEAR IDC FIFO ADDR
                                                      ; SUBROUTINE
4C03  417
4C03  418 00000600  WCS_DAT_32_ADD  = <^X600>         ;ADDRESS OF DATA 32 XFER ROUTINE
4C03  419 0000005F  WCS_DE_TCSR     = <^X5F>          ;ADDRESS OF DEPOSIT IDC CSR SUBROUTINE
```

```
4C03  420 00000060   WCS_DE_IDAR     = <^X60>      ;ADDRESS OF DEPOSIT IDC DAR SUBROUTINE
4C03  421 00000061   WCS_DE_DBUF     = <^X61>      ;ADDRESS OF DEPOSIT IDC DBUF SUBROUTINE
4C03  422 00000050   WCS_DE_MCT      = <^X50>      ;ADDRESS OF DEPOSIT MCT SUBROUTINE
4C03  423 00000054   WCS_DE_MM       = <^X54>      ;ADDRESS OF DEPOSIT MM SUBROUTINE
4C03  424 00000052   WCS_DE_TB       = <^X52>      ;ADDRESS OF DEPOSIT TB SUBROUTINE
4C03  425 00000058   WCS_DE_UBS      = <^X58>      ;ADDRESS OF DEPOSIT UBS SUBROUTINE
4C03  426 00000041   WCS_ERR_NUM     = <^X41>      ;ADDRESS OF ERROR NUMBER DATA WORD
4C03  427 0000005C   WCS_EX_DBUF     = <^X5C>      ;ADDRESS OF EXAMINE IDC DATA BUF
4C03  428 0000005A   WCS_EX_ICSR     = <^X5A>      ;ADDRESS OF EXAMINE IDC CSR
4C03  429 0000005B   WCS_EX_IDAR     = <^X5B>      ;ADDRESS OF EXAMINE IDC DAR
4C03  430 00000051   WCS_EX_MCT      = <^X51>      ;ADDRESS OF EXAMINE MCT SUBROUTINE
4C03  431 00000055   WCS_EX_MM       = <^X55>      ;ADDRESS OF EXAMINE MM SUBROUTINE
4C03  432 0000005D   WCS_EX_PATT     = <^X5D>      ;ADDRESS OF EXAMINE IDC ECC PATTERN
4C03  433 0000005E   WCS_EX_POSIT    = <^X5E>      ;ADDRESS OF EXAMINE IDC ECC POSITION
4C03  434 00000053   WCS_EX_TB       = <^X53>      ;ADDRESS OF EXAMINE TB SUBROUTINE
4C03  435 00000059   WCS_EX_UBS      = <^X59>      ;ADDRESS OF EXAMINE UBS SUBROUTINE
4C03  436 00000042   WCS_EXP_DAT     = <^X42>      ;ADDRESS OF EXPECTED DATA WORD
4C03  437 00000000   WCS_LOAD_ADD    = <^X0>       ;ADDRESS WCS OVERLAYS LOAD AT.
4C03  438 00000046   WCS_MOD_DAT     = <^X46>      ;ADDRESS OF MODULE NUMBER DATA WORD
4C03  439 00000045   WCS_MSK_DAT     = <^X45>      ;ADDRESS OF ERROR MASK DATA WORD
4C03  440 00000043   WCS_REC_DAT     = <^X43>      ;ADDRESS OF RECEIVED DATA WORD
4C03  441 00000057   WCS_REST_WR     = <^X57>      ;RESTORE 2901 WORKING REGS SUBR ADDR
4C03  442 00000056   WCS_SAVE_WR     = <^X56>      ;SAVE 2901 WORKING REGS SUBR ADDR
4C03  443 00000063   WCS_SEL_FIFOA   = <^X63>      ;ADDRESS OF SELECT IDC FIFO A
4C03  444                                          ; SUBROUTINE
4C03  445 00000064   WCS_SEL_FIFOB   = <^X64>      ;ADDRESS OF SELECT IDC FIFO B
4C03  446                                          ; SUBROUTINE
4C03  447 00000606   WCS_VER_ADD     = <^X606>     ;ADDRESS OF WCS BASED VERSION NUMBER
4C03  448 00000607   WCS_FILE_NAME   = <^X607>     ;ADDRESS OF WCS BASED FILE NAME
4C03  449 00000020   WRONG_LAB       = <^X20>      ;BIT 5 COMMAND + STATUS WORD
4C03  450 00000020   XFER            = <^X20>      ;BIT 13 COMMAND + STATUS WORD
4C03  451
4C03  452            ;*************************************************************************
4C03  453            ;
4C03  454            ;  THE FOLLOWING SET OF EQUATES DEFINE THE BITS USED IN
4C03  455            ;  THE COMMAND AND STATUS WORD
4C03  456            ;
4C03  457            ;*************************************************************************
4C03  458
4C03  459            ;FLAG EQUATES
4C03  460 00000001   LOOP_S_DEF      = <^X1>       ;MASK USED TO SET LOOP ON ERROR
4C03  461 00000040   LOOP_COM_S_DEF  = <^X40>      ;MASK USED TO SET LOOP COMAND BIT
4C03  462 00000080   SPECIAL_S_DEF   = <^X80>      ;MASK USED TO SET SPECIAL BIT (USED
4C03  463                                          ; TO LOOP ON SIZING TESTS)
4C03  464 00000002   NER_S_DEF       = <^X2>       ;MASK USED TO SET NO ERROR REPORTS
4C03  465 00000004   BELL_S_DEF      = <^X4>       ;MASK USED TO SET BELL ON ERROR
4C03  466 00000008   HALT_S_DEF      = <^X8>       ;MASK USED TO SET HALT ON ERROR
4C03  467 00000010   SER_S_DEF       = <^X10>      ;MASK USED TO SET SINGLE ERR. REP.
4C03  468 00000020   APT_S_DEF       = <^X20>      ;MASK USED TO SET APT PRESENT
4C03  469                                          ; INDICATOR (NOT SETABLE BY SET
4C03  470                                          ; COMMAND)
4C03  471
4C03  472 000000BE   LOOP_C_DEF      = <^XBE>      ;MASK USED TO CLEAR LOOP ON ERROR
4C03  473                                          ; AND LOOP COMMAND BITS
4C03  474 000000BF   LOOP_COM_C_DEF  = <^XBF>      ;MASK USED TO CLEAR LOOP COMAND BIT
4C03  475 0000007F   SPECIAL_C_DEF   = <^X7F>      ;MASK USED TO CLEAR SPECIAL BIT (USED
4C03  476                                          ; TO LOOP ON SIZING TESTS)
4C03  477 000000BD   NER_C_DEF       = <^XBD>      ;MASK USED TO CLEAR NO ERROR REPORT
4C03  478                                          ; AND LOOP COMMAND BITS
4C03  479 000000FB   BELL_C_DEF      = <^XFB>      ;MASK USED TO CLEAR BELL ON ERROR
```

```
4C03   480 000000F7                HALT_C_DEF      = <^XF7>        ;MASK USED TO CLEAR HALT ON ERROR
4C03   481 000000EF                SER_C_DEF       = <^XEF>        ;MASK USED TO CLEAR SINGLE ERR. REP.
4C03   482 000000DF                APT_C_DEF       = <^XDF>        ;MASK USED TO CLEAR APT PRESENT
4C03   483                                                         ; INDICATOR (NOT CLEARABLE BY CLEAR
4C03   484                                                         ; COMMAND)
4C03   485
4C03   486                         ;*****************************************************************************
4C03   487                         ;
4C03   488                         ;  PARAMETER BLOCK FOR TU58 DRIVER
4C03   489                         ;
4C03   490                         ;*****************************************************************************
4C03   491
4C03   492                                                         ;TU58 DRIVER PARAMETER BLOCK
4C03   493 00004106                SEC_PARMB_UNIT  = <^X4106>      ;UNIT NUMBER
4C03   494 00004107                SEC_PARMB_NAM   = <^X4107>      ;FILENAME
4C03   495 0000410D                SEC_PARMB_EXT   = <^X410D>      ;DOT PLUS FILE EXTENTION
4C03   496 00004111                SEC_PARMB_ADD   = <^X4111>      ;LOAD ADDRESS
4C03   497 00004115                SEC_PARMB_DST   = <^X4115>      ;DESTINATION 6=RAM 7=WCS
4C03   498 00004116                SEC_PARMB_ERR   = <^X4116>      ;ERROR RETURN BYTE
4C03   499 00004150                SEC_PARMB_DEF   = <^X4150>      ;DEFAULT DRIVE (WHERE MICMON WAS LOADED
4C03   500                                                         ; FROM)
4C03   501
4C03   502                         ;*****************************************************************************
4C03   503                         ;
4C03   504                         ;  EXTERNAL REFERENCES
4C03   505                         ;  ALL EXTERNAL REFERENCES MUST BE LISTED HERE
4C03   506                         ;
4C03   507                         ;*****************************************************************************
4C03   508
4C03   509                                 .GLOBAL COMPARE,EOS_FLG,FLAGS,HEX_BUF
4C03   510                                 .GLOBAL MOVER,PRINT,PRINT_HEX,SPACE
4C03   511                                 .GLOBAL MWCS_A,MCPU_A,CR,LF,HEX_3,CC_ADDR,CRLF
4C03   512                                 .GLOBAL SHIFT_PNT,F_WORD,SHIFTER,DOLOOP
4C03   513                                 .GLOBAL NOP_CSR,READ_WCS,WCS_ADDRESS,WCS_VALUE
4C03   514                                 .GLOBAL WRITE_WCS,X_SET_PAGE,PERFORM_CSR
4C03   515                                 .GLOBAL X_CLR_PAGE,WCS_VAL_SHIFT
4C03   516                                 .GLOBAL CSR_VALUE,LOAD_CSR,UPC_VALUE,LOAD_UPC
4C03   517                                 .GLOBAL PARITY_JUMP,PARITY_ON,X_MOV_WRWR
4C03   518                                 .GLOBAL SHIFT_L_2901,SHIFT_R_2901,DATA0
4C03   519                                 .GLOBAL READ_DATA_32,WRITE_DATA_32,X_CLR_WR0
4C03   520                                 .GLOBAL DATA1,DATA2,DATA3,DATA_SHIFT
4C03   521                                 .GLOBAL READ_LS,WRITE_LS,INC_WORD,OS_ADD
4C03   522                                 .GLOBAL NOP_INST,CSR_SHIFT,X_COM_WR0
4C03   523                                 .GLOBAL MAKE_XD,XD_ADDRS,X_ROT_L,X_SHIFT_L
4C03   524                                 .GLOBAL WCS_BAD_PARITY,POWER_BRANCH,WCS_SHIFT
4C03   525
4C03   526                                 .GLOBAL PRINT_STRING,MOVER_4,PRINT_HEX_1,LD_UPC
4C03   527                                 .GLOBAL CON_ADD_DAT,CON_ERR_NUM,CON_EXP_DAT
4C03   528                                 .GLOBAL CON_EXP_ADD,CON_MOD_DAT,CON_MSK_DAT
4C03   529                                 .GLOBAL CON_REC_DAT,CON_REC_ADD,ERROR_CON,MOVER_3
4C03   530                                 .GLOBAL NA_EXP_REC,NA_OTHER,SKIP_TEST,MSK_ERROR
4C03   531                                 .GLOBAL CON_ERROR,CON_BEGIN_TEST,GCVECT,WCS_READ
4C03   532                                 .GLOBAL WCS_WRITE,CONTINUE,CNTLC_TYPED,SET_FOR_CONT
4C03   533                                 .GLOBAL CLEAR_CPU_ATT,SHIFTER1,X_SHIFT_R,DECR_WORD
4C03   534                                 .GLOBAL MICRO_STEP_CPU,WRITE_TAB_PNT
4C03   535                                 .GLOBAL APT,APT_PASS_CNT
4C03   536
4C03   537                         ;*****************************************************************************
4C03   538                         ;
4C03   539                         ;  APT INFORMATION.... THE ADDRESS OF THESE VARIABLES MUST NOT CHANGE
```

```
4C03   540
4C03   541
4C03   542
4C03   543
4C03   544 06E4'C3                  ;  IN OFFSET FROM THE BEGINNING OF THIS PROGRAM.
4C06   545                          ;
4C06   546 06D6'C3                  ;*****************************************************************************
4C09   547
4C09   548 7000                     OVERLAY_RAM::   JMP     APT_CON_TEST    ;STARTING ADDRESS FOR RAM TESTS
4C0B   549 00000010                                                         ;  STARTED BY APT
4C10   550                          OVERLAY_WCS::   JMP     APT_WCS_TEST    ;STARTING ADDRESS FOR WCS TESTS
4C10   551 0000                                                             ;  STARTED BY APT
4C12   552 0000                     APT_RAM_LD_ADD: .WORD   EXEC_CONTESTS   ;POINTER TO RAM OVERLAY LOAD ADDRESS
4C14   553 0000                                     .BLKB   5
4C16   554 0000                     APT_MAIL_BOX:
4C18   555 0000                     APT_MESSAGE_CODE:.WORD  0               ;FLAG 1=TEST START  2=ERROR
4C1A   556 0000001C                 APT_ERROR_NUMBER:.WORD  0               ;DIAGNOSTIC ERROR NUMBER
4C1C   557 00                       APT_SUBTEST:    .WORD   0               ;SUBTEST NUMBER FILLER
4C1D   558 00                       APT_TEST_NUMBER:.WORD   0               ;FAILING TEST NUMBER
4C1E   559 00                       APT_PASS_CNT:   .WORD   0               ;# PASSES EXECUTED
4C1F   560 00000022                                 .BLKB   2               ;RESERVED FILLER
4C22   561                          APT_MEM_SIZE:   .BYTE   0               ;APT MEMORY SIZE
4C22   562                          APT_HARD_FLG:   .BYTE   0               ;APT HARDWARE INFORMATION
4C22   563                          APT_SOFT_FLG:   .BYTE   0               ;APT SOFTWARE INFORMATION
4C22   564                                          .BLKB   3               ;RESERVED FILLER
4C22   565
4C22   566                          ;*****************************************************************************
4C22   567                          ;
4C22   568                          ;  RETURN VECTORS - THE FOLLOWING FOUR INSTRUCTIONS ARE USED AS THE RETURN
4C22   569                          ;  POINT FOR SOME BOOT BLOCK ROUTINES.  THE NEXT BYTE IS A FLAG READ BY THE
4C22   570                          ;  BOOT BLOCK TO DETERMINE IF MICMON IS LOADED.  THE NEXT TWO INSTRUCTIONS
4C22   571                          ;  ARE ALSO USED AS VECTORS FROM THE BOOT BLOCK.  THEY MUST NOT CHANGE IN
4C22   572 1BEE'C3                  ;  OFFSET FROM THE BEGINNING OF THIS PROGRAM.
4C25   573 1F55'C3                  ;
4C28   574 1322'C3                  ;*****************************************************************************
4C2B   575 1333'C3
4C2E   576 01                                       JMP     PARITY_ERR      ;RETURN FROM PARITY ERROR
4C2F   577 1F6C'C3                                  JMP     POWER_REC       ;RETURN FROM POWER FAIL RECOVER
4C32   578 00000038                                 JMP     CNTLC_VEC       ;RETURN FROM CNTL C TYPED
4C38   579                                          JMP     CNTLP_VEC       ;RETURN FROM CNTL P TYPED
4C38   580                          MICFLAG:        .BYTE           1       ;FLAG SO BOOT BLOCK KNOWS MIC RESIDENT
4C38   581                                          JMP     TIMER_VEC       ;RETURN FROM INTERVAL TIMER INTERRUPT
4C38   582                                          .BLKB   6               ;RESERVED FILLER FOR NEW RETURN VECTORS
4C38   583
4C38   584                          ;*****************************************************************************
4C38   585                          ;
4C38   586                          ;       THE FOLLOWING DECLARATIONS ARE FOR GLOBAL VARIABLES.
4C38   587                          ;       DO NOT DISPLACE FROM THEIR CURRENT LOCATIONS.
4C38   588 00                       ;       ADD NEW VARIABLES AFTER ALL GLOBAL DECLARATIONS.
4C39   589 00                       ;
4C3A   590 00                       ;*****************************************************************************
4C3B   591 0000003F
4C3F   592 01                       APT:            .BYTE           0       ;FLAG 1=APT PRESENT
4C40   593 00000044                 CNTLC_TYPED:    .BYTE           0       ;FLAG 1=CONTROL C WAS TYPED
4C44   594 00000046                 CONTINUE:       .BYTE           0       ;FLAG 1=CONTINUE MODE ENABLED
4C46   595 0000004B                 CON_ADD_DAT:    .BLKB           4       ;OTHER DATA
4C4B   596 0000004F                 CON_ERR_NUM:    .BYTE           1       ;CONSOLE BASED ERROR NUMBER
4C4F   597 00000053                 CON_EXP_DAT:    .BLKB           4       ;EXPECTED DATA
4C53   598 00000055                 CON_EXP_ADD:    .BLKB           2       ;EXP DAT ADDR USED BY MSK_ERR
4C55   599 00                       CON_MOD_DAT:    .BLKB           5       ;ASCII MODULE NAME-FILLED IN
                                    CON_MSK_DAT:    .BLKB           4       ;MASK DATA
                                    CON_REC_DAT:    .BLKB           4       ;RECEIVED DATA
                                    CON_REC_ADD:    .BLKB           2       ;REC DAT ADDR USED BY MSK_ERR
                                    DOLOOP:         .BYTE           0       ;DO MACRO LOOP COUNTER
```

```
4C56  600 00                  EOS_FLG:          .BYTE       0               ;FLAG 1=END OF SECTION
4C57  601 00                  ERROR_CON:        .BYTE       0               ;FLAG 1=ERROR EXISTS
4C58  602 FFFF                F_WORD:           .WORD       <^XFFFF>
4C5A  603 00                  FLAGS:            .BYTE       0               ;HALT,LOOP,NER,BELL,SER
4C5B  604 0000005F            HEX_BUF:          .BLKB       4               ;HEX PRINT OUT BUFFER
4C5F  605 00                  NA_EXP_REC:       .BYTE       0               ;FLAG 1=N/A FOR EXP AND REC DA
4C60  606 00                  NA_OTHER:         .BYTE       0               ;FLAG 1=N/A FOR OTHER DATA
4C61  607 1BEE'C3             PARITY_JUMP:      JMP         PARITY_ERR
4C64  608 01                  PARITY_ON:        .BYTE       1               ;FLAG 1=CALCULATE PARITY
4C65  609 1F55'C3             POWER_BRANCH:     JMP         POWER_REC
4C68  610 0000                SHIFT_PNT:        .WORD       0               ;SHIFT ROUTINE DATA POINTER
4C6A  611 00                  SKIP_TEST:        .BYTE       0               ;FLAG 1=SKIP TEST # IS TO LOW
4C6B  612 02                  WCS_SHIFT:        .BYTE       2               ;TWO BYTES OF SHIFT DATA FOLLOW
4C6C  613 0000                WCS_ADDRESS:      .WORD       0               ;WCS ADDRESS TO READ OR WRITE
4C6E  614 00                  WCS_BAD_PARITY:   .BYTE       0               ;FLAG 1=WRITE BAD PARITY
4C6F  615 0000                WRITE_TAB_PNT:    .WORD       0               ;POINTER TO UPC ADDRES TABLE
4C71  616 00                  XD_ADDRS:         .BYTE       0               ;XD ADDRS FIELD OF CPU INST
4C72  617
4C72  618                     ;GLOBAL CSR EXECUTABLE CODE
4C72  619
4C72  620 DB                  NOP_INST:         .BYTE       <^XDB>          ;NOP INST FOR CPU CSR
4C73  621 00                                    .BYTE       <^X00>
4C74  622 15                                    .BYTE       <^X15>
4C75  623 01                  X_SHIFT_L:        .BYTE       1
4C76  624 23                                    .BYTE       <^X23>          ;CLR WR[1]
4C77  625 D0                                    .BYTE       <^XD0>
4C78  626 15                                    .BYTE       <^X15>
4C79  627
4C79  628 01                  X_ROT_L:          .BYTE       1
4C7A  629 A3                                    .BYTE       <^XA3>          ;ROL WR[0]
4C7B  630 C0                                    .BYTE       <^XC0>          ;SHIFT LEFT 1 PLACE
4C7C  631 15                                    .BYTE       <^X15>
4C7D  632
4C7D  633 01                  X_CLR_PAGE:       .BYTE       1
4C7E  634 90                                    .BYTE       <^X90>          ;MISC [CLR.HIGH.PAGE]
4C7F  635 A0                                    .BYTE       <^XA0>
4C80  636 15                                    .BYTE       <^X15>
4C81  637
4C81  638 01                  X_SET_PAGE:       .BYTE       1
4C82  639 10                                    .BYTE       <^X10>          ;MISC [SET.HIGH.PAGE]
4C83  640 B0                                    .BYTE       <^XB0>
4C84  641 15                                    .BYTE       <^X15>
4C85  642
4C85  643
4C85  644                     ;  This subroutine shifts the contents of WR[0] right 8 bits and leaves the
4C85  645                     ;  result on the Y-bus.  The monitor can clock the console read register to
4C85  646                     ;  read the result.  To be executed 3 times by the monitor for a 32 bit WR
4C85  647                     ;  read.  ROR will have to have been checked by the console.
4C85  648
4C85  649 08                  X_SHIFT_R:        .BYTE       8
4C86  650 23                                    .BYTE       <^X23>          ;ROR
4C87  651 40                                    .BYTE       <^X40>
4C88  652 15                                    .BYTE       <^X15>
4C89  653
4C89  654 23                                    .BYTE       <^X23>          ;ROR
4C8A  655 40                                    .BYTE       <^X40>
4C8B  656 15                                    .BYTE       <^X15>
4C8C  657
4C8C  658 23                                    .BYTE       <^X23>          ;ROR
4C8D  659 40                                    .BYTE       <^X40>
```

```
4C8E   660 15                                    .BYTE       <^X15>
4C8F   661
4C8F   662 23                                    .BYTE       <^X23>          ;ROR
4C90   663 40                                    .BYTE       <^X40>
4C91   664 15                                    .BYTE       <^X15>
4C92   665
4C92   666 23                                    .BYTE       <^X23>          ;ROR
4C93   667 40                                    .BYTE       <^X40>
4C94   668 15                                    .BYTE       <^X15>
4C95   669
4C95   670 23                                    .BYTE       <^X23>          ;ROR
4C96   671 40                                    .BYTE       <^X40>
4C97   672 15                                    .BYTE       <^X15>
4C98   673
4C98   674 23                                    .BYTE       <^X23>          ;ROR
4C99   675 40                                    .BYTE       <^X40>
4C9A   676 15                                    .BYTE       <^X15>
4C9B   677
4C9B   678 23                                    .BYTE       <^X23>          ;ROR
4C9C   679 40                                    .BYTE       <^X40>
4C9D   680 15                                    .BYTE       <^X15>
4C9E   681
4C9E   682 A0                        X_MOV_WRWR: .BYTE       <^XA0>          ;MOV WR[0] TO WR[0]
4C9F   683 00                                    .BYTE       <^X00>          ;MOVE DATA TO Y-BUS
4CA0   684 15                                    .BYTE       <^X15>          ;TO BE READ
4CA1   685
4CA1   686
4CA1   687
4CA1   688 02                        X_CLR_WR0:  .BYTE       2
4CA2   689 2F                                    .BYTE       <^X2F>          ;CLR WR[0]
4CA3   690 80                                    .BYTE       <^X80>
4CA4   691 15                                    .BYTE       <^X15>
4CA5   692
4CA5   693 A0                        X_COM_WR0:  .BYTE       <^XA0>          ;COM WR[0]
4CA6   694 C0                                    .BYTE       <^XC0>
4CA7   695 15                                    .BYTE       <^X15>
4CA8   696
4CA8   697                           ;*****************************************************************************
4CA8   698                           ;
4CA8   699                           ;  DATA AREA FOR READS AND WRITES TO CPU (32 BITS)
4CA8   700                           ;
4CA8   701                           ;*****************************************************************************
4CA8   702
4CA8   703 04                        DATA_SHIFT: .BYTE       4               ;FOUR BYTE OF SHIFT DATA
4CA9   704 00                        DATA0:      .BYTE       0               ;MSB OF 32 BIT DATA AREA
4CAA   705 00                        DATA1:      .BYTE       0
4CAB   706 00                        DATA2:      .BYTE       0
4CAC   707 00                        DATA3:      .BYTE       0               ;LSB OF 32 BIT DATA AREA
4CAD   708 000000AF                              .BLKB       2               ;EXTENDED DATA FOR 3 BYTE MOVES
4CAF   709
4CAF   710 00                        COMMAND0:   .BYTE       0               ;MSB OF 32 BIT COMMAND+STATUS
4CB0   711 00                        COMMAND1:   .BYTE       0               ;WORD
4CB1   712 00                        COMMAND2:   .BYTE       0
4CB2   713 00                        COMMAND3:   .BYTE       0               ;LSB OF 32 BIT COMMAND+STATUS
4CB3   714
4CB3   715 03                        WCS_VAL_SHIFT: .BYTE    3               ;SHIFT 3 BYTES OF DATA
4CB4   716 000000B7                  WCS_VAL0E:  .BLKB       3               ;WCS WORD TO BE WRITTEN
4CB7   717
4CB7   718
4CB7   719                           ;*****************************************************************************
```

```
4CB7  720
4CB7  721                       ;
4CB7  722                       ;  CSR DATA BUFFER
4CB7  723                       ;
4CB7  724                       ;*****************************************************************************
4CB7  725 03                    CSR_SHIFT:      .BYTE           3               ;3 BYTES OF DATA
4CB8  726 000000BB              CSR_VALUE:      .BLKB           3
4CBB  727 000000BE              SAVED_CSR:      .BLKB           3
4CBE  728 000000C1              CONT_SAVE_CSR:  .BLKB           3
4CC1  729
4CC1  730                       ;*****************************************************************************
4CC1  731                       ;
4CC1  732                       ; UPC DATA BUFFER
4CC1  733                       ;
4CC1  734                       ;*****************************************************************************
4CC1  735
4CC1  736 02                    UPC_SHIFT:      .BYTE           2               ;2 BYTES OF DATA
4CC2  737 000000C4              UPC_VALUE:      .BLKB           2
4CC4  738 000000C6              SAVED_UPC:      .BLKB           2
4CC6  739 000000C8              UPC_SOB:        .BLKB           2
4CC8  740 000000CA              CONT_SAVE_UPC:  .BLKB           2
4CCA  741 000000CC              UPC_COMPARE:    .BLKB           2
4CCC  742 000000CF              ATT_UPC_SAVE:   .BLKB           3
4CCF  743
4CCF  744                       ;JUMPS TO GLOBAL SUBROUTINES
4CCF  745 1184'C3               COMPARE:        JMP             COMPARE_R
4CD2  746 11A4'C3               MOVER:          JMP             MOVER_R
4CD5  747 1265'C3               PRINT:          JMP             PRINT_R
4CD8  748 11C4'C3               PRINT_HEX:      JMP             PRINT_HEX_R
4CDB  749 1260'C3               SPACE:          JMP             SPACE_R
4CDE  750 125A'C3               CRLF:           JMP             CRLF_R
4CE1  751 10E7'C3               SHIFTER:        JMP             SHIFTER_R
4CE4  752 10F9'C3               NOP_CSR:        JMP             NOP_CSR_R
4CE7  753 109D'C3               READ_WCS:       JMP             READ_WCS_R
4CEA  754 10B9'C3               WRITE_WCS:      JMP             WRITE_WCS_R
4CED  755 15C2'C3               PERFORM_CSR:    JMP             PERFORM_CSR_R
4CF0  756 1116'C3               LOAD_CSR:       JMP             LOAD_CSR_R
4CF3  757 1148'C3               LOAD_UPC:       JMP             LOAD_UPC_R
4CF6  758 1515'C3               SHIFT_L_2901:   JMP             SHIFT_L_2901_R
4CF9  759 151B'C3               SHIFT_R_2901:   JMP             SHIFT_R_2901_R
4CFC  760 1481'C3               READ_DATA_32:   JMP             READ_DATA_32_R
4CFF  761 149E'C3               WRITE_DATA_32:  JMP             WRITE_DATA_32_R
4D02  762 1435'C3               READ_LS:        JMP             READ_LS_R
4D05  763 1415'C3               WRITE_LS:       JMP             WRITE_LS_R
4D08  764 1581'C3               INC_WORD:       JMP             INC_WORD_R
4D0B  765 1450'C3               MAKE_XD:        JMP             MAKE_XD_R
4D0E  766 124A'C3               PRINT_STRING:   JMP             PRINT_STRING_R
4D11  767 11A2'C3               MOVER_4:        JMP             MOVER_4_R
4D14  768 11C2'C3               PRINT_HEX_1:    JMP             PRINT_HEX_1_R
4D17  769 22D7'C3               LD_UPC:         JMP             LD_UPC_R
4D1A  770 118E'C3               MOVER_3:        JMP             MOVER_3_R
4D1D  771 223C'C3               MSK_ERROR:      JMP             MSK_ERROR_R
4D20  772 221F'C3               CON_ERROR:      JMP             CON_ERROR_R
4D23  773 21CD'C3               CON_BEGIN_TEST: JMP             CON_BEGIN_TEST_R
4D26  774 228B'C3               WCS_READ:       JMP             WCS_READ_R
4D29  775 22C1'C3               WCS_WRITE:      JMP             WCS_WRITE_R
4D2C  776 1349'C3               SET_FOR_CONT:   JMP             SET_FOR_CONT_R
4D2F  777 1D43'C3               CLEAR_CPU_ATT:  JMP             CLEAR_CPU_ATT_R
4D32  778 10F0'C3               SHIFTER1:       JMP             SHIFTER1_R
4D35  779 158C'C3               DECR_WORD:      JMP             DECR_WORD_R
```

```
4D38  780 1570'C3                  MICRO_STEP_CPU: JMP          MICRO_STEP_CPU_R
4D3B  781
4D3B  782                          ;*****************************************************************************
4D3B  783                          ;
4D3B  784                          ;  MODULE LITERALS
4D3B  785                          ;
4D3B  786                          ;*****************************************************************************
4D3B  787
4D3B  788                          PRT_MOD_A:
4D3B  789 34 39 33 38 4D           MWCS_A:        .ASCII        'M8394''        ;WCS
4D40  790 30 39 33 38 4D           MCPU_A:        .ASCII        'M8390''        ;CPU
4D45  791 31 39 33 38 4D           MMCT_A:        .ASCII        'M8391''        ;MCT
4D4A  792 39 38 33 38 4D           MFPA_A:        .ASCII        'M8389''        ;FPA
4D4F  793 38 32 37 38 4D           MMEM_A:        .ASCII        'M8728''        ;MEMORY ARRAY CARD
4D54  794 38 38 33 38 4D           MIDC_A:        .ASCII        'M8388''        ;IDC
4D59  795 20 32 30 4C 52           MRL02_A:       .ASCII        'RL02 ''        ;RL02
4D5E  796 20 20 30 38 52           MR80_A:        .ASCII        'R80  ''        ;R80
4D63  797
4D63  798                          ;*****************************************************************************
4D63  799                          ;
4D63  800                          ;  INPUT BUFFER AREA
4D63  801                          ;
4D63  802                          ;*****************************************************************************
4D63  803
4D63  804 00                       TTBUF_CNT:     .BYTE         0               ;CHAR CNT OF INPUT LINE PUT
4D64  805                                                                        ; HERE BY GET LINE ROUTINE
4D64  806 000001B4                 TTBUF:         .BLKB         80              ;TTY BUFFER
4DB4  807
4DB4  808                          ;*****************************************************************************
4DB4  809                          ;
4DB4  810                          ;     ALL PRECEDING DECLARATIONS ARE FOR GLOBAL VARIABLES.
4DB4  811                          ;     DO NOT DISPLACE ANY OF THESE FROM THEIR CURRENT LOCATIONS.
4DB4  812                          ;     ALL NEW VARIABLES MUST BE ADDED BELOW THIS POINT.
4DB4  813                          ;
4DB4  814                          ;*****************************************************************************
4DB4  815
4DB4  816
4DB4  817                          ;VARIABLES
4DB4  818 00                       A_ADDRS:       .BYTE         0               ;A ADDRS FIELD OF CPU INST
4DB5  819 00                       ACK_SAVE:      .BYTE         0               ;SAVE CPU ACK FOR SUBROUTINES
4DB6  820 45 54 20 49 44           APT_DI_COMMAND: .ASCII "DI TE 1 CO"<CR>      ;ASCII COMMAND STRING TO START
          0D 4F 43 20 31 20
4DC1  821 00                                      .BYTE         0               ; WCS TESTS UNDER APT
4DC2  822 09                       ARRAY_NUM_A:   .BYTE         9               ;CHAR COUNT
4DC3  823 52 52 41 20 20                          .ASCII        "  ARRAY #"
          23 20 59 41
4DCC  824 00                       B_ADDRS:       .BYTE         0               ;B ADDRS FIELD OF CPU INST
4DCD  825 20                       BLANK_A:       .ASCII        " "             ;ONE ASCII SPACE
4DCE  826 000001D4                 BOARD_NAME:    .BLKB         6               ;BOARD NAME AREA
4DD4  827 000001DA                 BOARD_BUF:     .BLKB         6               ;BOARD NAME AREA
4DDA  828 00                       BOARD_NUM:     .BYTE         0               ;NUMBER OF BOARD INPUT
4DDB  829 0000                     BOARD_PNT:     .WORD         0               ;POINTER TO BOARD TABLE
4DDD  830 20                       BREAK_A:       .BYTE         32              ;CHAR COUNT
4DDE  831 41 45 52 42 0D                          .ASCII <CR>"BREAK"<CR>" A  C  BC   DE   HL   SP"<CR>
          20 41 20 0D 4B
          42 20 20 43 20
          44 20 20 20 43
          48 20 20 20 45
          53 20 20 20 4C
          0D 50
```

```
ZZ-ENKAA-2.1    4C00    4
4DFE   832 0000
4E00   833 1EFA'C3
4E03   834 00
4E04   835 00
4E05   836 00
4E06   837 0000
4E08   838 00
4E09   839 00
4E0A   840 00
4E0B   841 01
4E0C   842 0D
4E0D   843 04
4E0E   844 20 52 53 43
4E12   845 00
4E13   846 00
4E14   847 20 46 55 42 44
4E19   848 07
4E1A   849 56 49 52 44 20
           20 45
4E21   850 4B 4E 45
4E24   851 00000226
4E26   852
4E26   853 00
4E27   854 05
4E28   855 20 50 4F 45 0D
4E2D   856 03
4E2E   857 20 3D 20
4E31   858 00
4E32   859 06
4E33   860 4F 52 52 45 20
           52
4E39   861 0000
4E3B   862 00
4E3C   863 45 58 45 2E
4E40   864 00
4E41   865 00
4E42   866 04
4E43   867 41 50 46 0D
4E47   868 1D
4E48   869 0D
4E49   870 20 54 43 45 53
           20 54 53 54 20
           20 20 52 52 45
           20 50 58 45 20
           20 20 20 20 20
           43 45 52
4E65   871 1D
4E66   872 20 20 20 20 20
           45 48 54 4F 20
           20 20 20 20 52
           20 20 4B 53 4D
           44 4F 4D 20 20
           0D 45 4C 55
4E83   873 00
4E84   874 20 52 53 43 49
4E89   875 20 52 41 44 49
4E8E   876 04
4E8F   877 43 44 49 0D
4E93   878 00
4E94   879 03
```

```
                                B 4

BREAK_ADDRESS:      .WORD       0               ;ADDRESS OF LAST SET BREAK
BREAK_BRANCH:       JMP         BREAK
BREAK_FLG:          .BYTE       0               ;FLAG 1=SET BREAK PERFORMED
C_EXECUTE:          .BYTE       0               ;TEMP FLAG HOLDING FOR CONTINUE
C_RUNNING:          .BYTE       0               ;TEMP FLAG HOLDING FOR CONTINUE
CALC_ADD:           .WORD       0               ;ADDRESS TO CALC PARITY ON
CNT_PAR:            .BYTE       0               ;COUNT OF PAR ERRORS
CONSOLE_TEST:       .BYTE       0               ;FLAG 1=CONSOLE 0=WCS TEST
CONT_DONE:          .BYTE       0               ;FLAG 1=CONTINUE IS DONE DO RET
CRLF_A:             .BYTE       1               ;BYTE COUNT
                    .ASCII      <CR>            ;CRLF FOR CALL CRLF ROUTINE
CSR_A:              .BYTE       4               ;CHAR COUNT
                    .ASCII      "CSR "
CSR_CNT:            .BYTE       0               ;COUNT OF CSR INST TO PERFORM
DATA_XFER_FLG:      .BYTE       0               ;FLAG 1=XFER WAS DONE
DBUF_A:             .ASCII      "DBUF "
DRIVE_A:            .BYTE       7               ;CHAR COUNT
                    .ASCII      " DRIVE "

ENK_A:              .ASCII      "ENK"
ENK_END:            .BLKB       2               ;STORAGE FOR LAST 2 CHARS OF
                                                ; SECTION NAME READ FROM FILE
                    .BYTE       0               ;LAST CHAR IS NULL
EOP_A:              .BYTE       5               ;CHAR COUNT
                    .ASCII      <CR>"EOP "
EQUALS_A:           .BYTE       3               ;CHAR COUNT
                    .ASCII      " = "
ERROR:              .BYTE       0               ;FLAG
ERROR_A:            .BYTE       6               ;CHAR COUNT
                    .ASCII      " ERROR"

EXAM_TEMP:          .WORD       0               ;TEMP STORAGE
EXECUTE:            .BYTE       0               ;FLAG 1=EXECUTE MODE
EXT_A:              .ASCII      ".EXE"          ;ALL OVERLAYS ARE .EXE !!!!!!
FAST_WRITE:         .BYTE       0               ;FLAG 1=USE FAST WRITE ROUTINE
FILE_LOADED:        .BYTE       0               ;SECTION LOADED FROM TAPE
FPA_A:              .BYTE       4               ;CHAR COUNT
                    .ASCII      <CR>"FPA"
HEADER_A:           .BYTE       29              ;CHAR COUNT
                    .ASCII      <CR>
                    .ASCII      "SECT  TST ERR   EXP      REC"

HEADER_B:           .BYTE       29              ;CHAR COUNT
                    .ASCII      "      OTHER    MSK    MODULE"<CR>

HAVE_CNT:           .BYTE       0               ;FLAG 1=WCS LOOP COUNT READ
ICSR_A:             .ASCII      "ICSR "
IDAR_A:             .ASCII      "IDAR "
IDC_A:              .BYTE       4               ;CHAR COUNT
                    .ASCII      <CR>"IDC"
LOOP_COUNTER:       .BYTE       0               ;LOOP COUNTER
LS_A:               .BYTE       3               ;CHAR COUNT
```

Fiche 1 Frame B4 Sequence 40

```
ZZ-ENKAA-2.1     4C00     4                    Fiche 1  Frame C4          Sequence 41
4E95   880 20 53 4C                  .ASCII    "LS "
4E98   881 00                        .BYTE     0                ;PARITY ERROR SET BY CPU ATTN
4E99   882 00                        .BYTE     0                ;PAR ERR ACTIVE (BEING PROCESSED)
4E9A   883 20 20 54 43 4D            .ASCII    "MCT  "
4E9F   884 00                        .BYTE     0                ;FLAG 1=MEM REQ TO BE ENABLED
4EA0   885 26                        .BYTE     38               ;CHAR COUNT
4EA1   886 4F 4D 45 4D 0D            .ASCII    <CR>"MEMORY SIZE IN 256K BLOCKS (IN HEX) = "
       887 49 53 20 59 52
       888 4E 49 20 45 5A
       889 4B 36 35 32 20
       890 43 4F 4C 42 20
       891 49 28 20 53 4B
       892 58 45 48 20 4E
       893 20 3D 20 29
4EC8   894 00                        .BYTE     0                ;FLAG NON 0=MICRO STEP MODE
4EC9   895 00                        .BYTE     0                ;FLAG 1=NEG NUMBER TYPED
4ECA   896 20 20 20 4D 4D            .ASCII    "MM   "
4ECF   897 0C                        .BYTE     12               ;CHAR COUNT
4ED0   898 20 54 4F 4E 20            .ASCII    " NOT PRESENT"
       899 45 53 45 52 50
       900 54 4E
4EDC   901 08                        .BYTE     8                ;CHAR COUNT
4EDD   902 53 45 52 50 20            .ASCII    " PRESENT"
       903 54 4E 45
4EE5   904 00                        .BYTE     0                ;FLAG 1=MORE DATA TO XFER
4EE6   905 09                        .BYTE     9                ;CHAR COUNT
4EE7   906 41 2F 4E 20 20            .ASCII    "  N/A    "
       907 20 20 20 20
4EF0   908 00                        .BYTE     0                ;NEW CHECKSUM VALUE
4EF1   909 00                        .BYTE     0                ;FLAG 1=NO ADDRESS ON EXAM+DEP
4EF2   910 00                        .BYTE     0                ;FLAG 1=NO INIT OF WCS (TEST0)
4EF3   911 00                        .BYTE     0                ;FLAG 1=NO RETURN AFTER MICMON
4EF4   912                                                      ; ERROR
4EF4   913 00                        .BYTE     0                ;FLAG 1= DON'T SAVE UPC OR CSR
4EF5   914 00                        .BYTE     0                ;FLAG 1=NO SPACE AFTER PR HEX
4EF6   915 00                        .BYTE     0                ;FLAG 1=INPUT PARSED NOT FOUND
4EF7   916 000002FA                  .BLKB     3                ;STORAGE FOR ORIGINAL CONTENTS
4EFA   917                                                      ; OF A SET BREAK LOCATION
4EFA   918 00000304                  .BLKB     10               ;BACKUP TEMP BUFF FOR ERRORS
4F04   919 0000                      .WORD     0                ;BACKUP POINTER TO TTBUF
4F06   920 00                        .BYTE     0                ;OLD CHECKSUM VALUE
4F07   921 03                        .BYTE     3                ;CHAR COUNT
4F08   922 20 53 4F                  .ASCII    "OS "
4F0B   923 00                        .BYTE     0                ;FLAG 1=WRONG RL02 PACK FOUND
4F0C   924 00                        .BYTE     0                ;FLAG 1=PARITY TO BE ENABLED
4F0D   925 0000                      .WORD     0                ;PARSER ADDRESS DEST ADDRESS
4F0F   926 00                        .BYTE     0                ;FLAG 1=END OF INPUT LINE
4F10   927 0000                      .WORD     0                ;POINTER TO PARSE BUFFER
4F12   928 0000                      .WORD     0                ;POINTER TO PARSE TABLE
4F14   929 0000                      .WORD     0                ;PASS COUNTER
4F16   930 20 54 54 41 50            .ASCII    "PATT "
4F1B   931 20 20 53 4F 50            .ASCII    "POS  "
4F20   932 00                        .BYTE     0                ;NUMBER OF CHARS TO PRINT
4F21   933 0000                      .WORD     0                ;POINTER TO CHARS TO BE PRINTED
4F23   934 05                        .BYTE     5                ;CHAR COUNT
4F24   935 3E 43 49 4D 0D           .ASCII    <CR>"MIC>"
4F29   936 04                        .BYTE     4                ;CHAR COUNT
4F2A   937 30 38 52 0D               .ASCII    <CR>"R80"
4F2E   938 04                        .BYTE     4                ;CHAR COUNT
4F2F   939 20 4D 41 52               .ASCII    "RAM "
```

```
4F33  929 00                  REPEAT:          .BYTE    0              ;REPEAT COMMAND FLAG
4F34  930 00                  RESTART:         .BYTE    0              ;FLAG 1=RESTART PARSER
4F35  931 00                  RUNNING:         .BYTE    0              ;FLAG 1=CPU RUNNING MODE
4F36  932 00                  SAVE_A:          .BYTE    0              ;BREAK ROUTINE SAVE A REG
4F37  933 0000                SAVE_BC:         .WORD    0              ;BREAK ROUTINE SAVE B+C REGS
4F39  934 0000                SAVE_C:          .WORD    0              ;BREAK ROUTINE SAVE CARRY
4F3B  935 0000                SAVE_DE:         .WORD    0              ;BREAK ROUTINE SAVE D+E REGS
4F3D  936 0000                SAVE_HL:         .WORD    0              ;BREAK ROUTINE SAVE H+L REGS
4F3F  937 0000                SAVE_SP:         .WORD    0              ;BREAK ROUTINE SAVE SP REG
4F41  938 0000                SAVE_STACK:      .WORD    0
4F43  939 00                  SECTION_B_NUM:   .BYTE    0              ;BOARD NUMBER FROM SEC. TABLE
4F44  940 00                  SETACK_FLG:      .BYTE    0              ;FLAG 1=SET CONSOLE ACKNOWLEDGE
4F45  941 00                  SHORTEN:         .BYTE    0              ;FLAG 1=SHORTEN TEST LOOP
4F46  942 06                  SHOWFLAG_A:      .BYTE    6              ;CHAR COUNT
4F47  943 53 47 41 4C 46                       .ASCII   "FLAGS:"
          3A
4F4D  944 2D                  SINGLE_A:        .BYTE    45             ;CHAR COUNT
4F4E  945 47 4E 49 53 0D                       .ASCII   <CR>"SINGLE BIT ERRORS DETECTED ON 256 KB BLOCK #"
          49 42 20 45 4C
          52 52 45 20 54
          44 20 53 52 4F
          54 43 45 54 45
          4E 4F 20 44 45
          20 36 35 32 20
          4C 42 20 42 4B
          23 20 4B 43 4F
4F7B  946 00                  SKIP_CNT:        .BYTE    0              ;NUMBER OF CHARS SPANED
4F7C  947 00                  SLASH_U:         .BYTE    0              ;FLAG 1= X/U 0=X/C TYPED
4F7D  948 19                  SLOW_WRITE_A:    .BYTE    25             ;CHAR COUNT
4F7E  949 4C 4C 49 57 0D                       .ASCII   <CR>"WILL DO SLOW WRITE TO LS"
          53 20 4F 44 20
          57 20 57 4F 4C
          20 45 54 49 52
          53 4C 20 4F 54
4F97  950 05                  SOMM_A:          .BYTE    5              ;CHAR COUNT
4F98  951 4D 4D 4F 53 0D                       .ASCII   <CR>"SOMM"
4F9D  952 0000                SOMM_ADDRESS:    .WORD    0              ;STOP ON MICRO MATCH ADDR
4F9F  953 00                  SOMM_FLG:        .BYTE    0              ;FLAG 1=STOP ON MICRO MATCH
4FA0  954 00                  SPACE_FOUND:     .BYTE    0              ;FLAG 1=SPACE FOUND IN TTBUF
4FA1  955 0000                STARTING_UPC:    .WORD    0              ;STARTING UPC ADDRESS
4FA3  956 0000                STEP_CNT:        .WORD    0              ;MICRO STEP COUNTER
4FA5  957 0D                  SYNTAX_A:        .BYTE    13             ;CHAR COUNT
4FA6  958 54 4E 59 53 0D                       .ASCII   <CR>"SYNTAX ERROR"
          52 45 20 58 41
          52 4F 52
4FB3  959 00                  TABLE_MODE:      .BYTE    0              ;TABLE MODE INDICATOR
4FB4  960 0000                TABLE_PNT:       .WORD    0              ;SECTION TABLE POINTER
4FB6  961 20 20 20 42 54      TB_A:            .ASCII   "TB   "
4FBB  962 000003C5            TEMP_BUF:        .BLKB    10             ;TEMP CHAR BUFFER
4FC5  963 00                  TEMP_BYTE:       .BYTE    0              ;TEMP STORAGE BYTE
4FC6  964 0000                TEMP_WORD:       .WORD    0              ;TEMP STORAGE WORD
4FC8  965 0000                TEMP_WORD1:      .WORD    0              ;TEMP STORAGE WORD
4FCA  966 0000                TEMP_WORD2:      .WORD    0              ;TEMP STORAGE WORD
4FCC  967 00                  TEMP_FLAG:       .BYTE    0              ;TEMP STORAGE OF SECTION FLAG
4FCD  968 000003D3            TEMP_SECTION:    .BLKB    6              ;TEMP STORAGE OF SECTION NAME
4FD3  969 05                  TEST_A:          .BYTE    5              ;CHAR COUNT
4FD4  970 20 54 53 45 54                       .ASCII   "TEST "
4FD9  971 00                  TEST_NUM:        .BYTE    0              ;CURRENT TEST NUMBER
4FDA  972 00                  TEST_NUM1:       .BYTE    0              ;STARTING TEST NUMBER
4FDB  973 00                  TEST_NUM2:       .BYTE    0              ;ENDING TEST NUMBER
```

```
ZZ-ENKAA-2.1    4C00    4
4FDC   974 00                  TEST_ZERO:      .BYTE           0               ;FLAG 1=EXECUTING TEST 0
4FDD   975 0000                TIMOUT_CNT:     .WORD           0               ;TIME OUT COUNTER
4FDF   976 00                  TRACE:          .BYTE           0               ;FLAG 1=TRACE MODE
4FE0   977 0000                TTBUF_PNT:      .WORD           0               ;POINTER TO TTY INPUT BUFFER
4FE2   978 04                  UBE_A:          .BYTE           4               ;CHAR COUNT
4FE3   979 45 42 55 0D                         .ASCII          <CR>''UBE''
4FE7   980 20 20 53 42 55      UBS_A:          .ASCII          ''UBS  ''
4FEC   981 05                  UPC_A:          .BYTE           5               ;CHAR COUNT
4FED   982 20 43 50 55 0D                      .ASCII          <CR>''UPC ''
4FF2   983 02                  VER_A:          .BYTE           2               ;CHAR COUNT
4FF3   984 56 2D                               .ASCII          ''-V''
4FF5   985 0B                  VERSION:        .BYTE           11              ;CHAR COUNT
4FF6   986 20 52 45 56 0D                      .ASCII          <CR>''VER 02.10''<CR>
       0D 30 31 2E 32 30
5001   987 00                  WARM_RESTART:   .BYTE           0               ;FLAG 1=POWER FAIL RESTART
5002   988 04                  WCS_A:          .BYTE           4               ;CHAR COUNT
5003   989 20 53 43 57                         .ASCII          ''WCS ''
5007   990 00                  WCS_CALC:       .BYTE           0               ;FLAG 1=WCS PARITY CALCULATION
5008   991 00                  WCS_LOADED:     .BYTE           0               ;FLAG 1=WCS OVERLAY LOADED
5009   992 00                  WCS_PNT:        .BYTE           0               ;COUNT FOR WCS 3 BYTE LOADS
500A   993 04                  WRK_A:          .BYTE           4               ;CHAR COUNT
500B   994 20 4B 52 57                         .ASCII          ''WRK ''
500F   995 00000413            X_COM_ADDRS:    .BLKB           4
5013   996 0000                XFER_CNT:       .WORD           0               ;LONG WORD XFER COUNTER
5015   997 00000418            XFER_DEST:      .BLKB           3               ;ADDR DATA XFER TO
5018   998 00                  XFER_MM:        .BYTE           0               ;FLAG FOR MM OR LS XFER
5019   999 0000                XFER_PNT:       .WORD           0
501B  1000 0000041F            ZERO_WORD:      .BLKB           4
501F  1001
501F  1002                     ;CSR EXECUTABLE CODE
501F  1003                     ;NOTE:  THE INSTRUCTIONS THAT ARE PAIRED WITH S_XXXX ARE DYNAMICLY MODIFIED
501F  1004                     ;       DURING EXECUTION.  THUS THE ADDRESS 0 WAS ASSEMBLED INTO THEM. THIS
501F  1005                     ;       ADDRESS MY BE MODIFYED
501F  1006
501F  1007 01                  X_READ_LS:      .BYTE           1
5020  1008 36                                  .BYTE           <^X36>          ;MOV LS[0] TO WR[0]
5021  1009 00                                  .BYTE           <^X00>
5022  1010 15                                  .BYTE           <^X15>
5023  1011
5023  1012 36                  S_READ_LS:      .BYTE           <^X36>          ;MOV LS[0] TO WR[0]
5024  1013 00                                  .BYTE           <^X00>
5025  1014 15                                  .BYTE           <^X15>
5026  1015
5026  1016
5026  1017 01                  X_CLR_CPU_ATT:  .BYTE           1               ;INSTRUCTION COUNT
5027  1018 90                                  .BYTE           <^X90>          ;MISC [CLR.CP.ATTN.AND.ACK]
5028  1019 C0                                  .BYTE           <^XC0>
5029  1020 15                                  .BYTE           <^X15>
502A  1021
502A  1022
502A  1023 01                  X_ED_WRK:       .BYTE           1               ;INSTRUCTION COUNT
502B  1024 A0                                  .BYTE           <^XA0>          ;MOV WR[0] TO WR[0]
502C  1025 00                                  .BYTE           <^X00>
502D  1026 15                                  .BYTE           <^X15>
502E  1027
502E  1028 A0                  S_ED_WRK:       .BYTE           <^XA0>          ;MOV WR[0] TO WR[0]
502F  1029 00                                  .BYTE           <^X00>
5030  1030 15                                  .BYTE           <^X15>
5031  1031
5031  1032
```

```
5031 1033 01                    X_WRITE_LS:     .BYTE           1               ;INSTRUCTION COUNT
5032 1034 BE                                    .BYTE           <^XBE>          ;MOV WR[0] TO LS[0]
5033 1035 00                                    .BYTE           <^X00>
5034 1036 15                                    .BYTE           <^X15>
5035 1037
5035 1038 BE                    S_WRITE_LS:     .BYTE           <^XBE>          ;MOV WR[0] TO LS[0]
5036 1039 00                                    .BYTE           <^X00>
5037 1040 15                                    .BYTE           <^X15>
5038 1041
5038 1042
5038 1043
5038 1044                       ;*****************************************************************************
5038 1045                       ;
5038 1046                       ;  TEMP CSR AREA FOR SHIFTING DATA
5038 1047                       ;
5038 1048                       ;*****************************************************************************
5038 1049
5038 1050 03                    TEMP_SHIFT:     .BYTE           3               ;3 BYTES OF DATA IN CSR
5039 1051 0000043C              TEMP_CSR:       .BLKB           3               ;TEMP CSR STORAGE
503C 1052
503C 1053                       ;*****************************************************************************
503C 1054                       ;
503C 1055                       ;  INPUT HEX NUMBER AREA
503C 1056                       ;
503C 1057                       ;*****************************************************************************
503C 1058
503C 1059 04                    NUM_SHIFT:      .BYTE           4               ;4 BYTES OF HEX DIGITS
503D 1060 00000441              INPUT_NUM:      .BLKB           4
5041 1061
5041 1062                       ;*****************************************************************************
5041 1063                       ;
5041 1064                       ;  EXPECTED AND RECEIVED RL02 LABEL AREA
5041 1065                       ;
5041 1066                       ;*****************************************************************************
5041 1067
5041 1068 0C                    EXP_LABEL:      .BYTE           12              ;LABEL IS 12 BYTES LONG
5042 1069 0000044E              EXP_LAB_DAT:    .BLKB           12              ;STORAGE FOR EACH BYTE
504E 1070
504E 1071 0C                    REC_LABEL:      .BYTE           12              ;CHAR COUNT
504F 1072 0000045B              REC_LAB_DAT:    .BLKB           12              ;STORAGE FOR EACH BYTE
505B 1073
505B 1074
505B 1075                       ;*****************************************************************************
505B 1076                       ;
505B 1077                       ;  BOARD NAME TABLE
505B 1078                       ;
505B 1079                       ;*****************************************************************************
505B 1080
505B 1081 01                    BOARD_TABLE:    .BYTE           1               ;BOARD NUMBER
505C 1082 53 43 57                              .ASCII          "WCS"
505F 1083 02                                    .BYTE           2               ;BOARD NUMBER
5060 1084 50 41 44                              .ASCII          "DAP"
5063 1085 02                                    .BYTE           2               ;BOARD NUMBER
5064 1086 55 50 43                              .ASCII          "CPU"
5067 1087 04                                    .BYTE           4               ;BOARD NUMBER
5068 1088 54 43 4D                              .ASCII          "MCT"
506B 1089 08                                    .BYTE           8               ;BOARD NUMBER
506C 1090 41 50 46                              .ASCII          "FPA"
506F 1091 10                                    .BYTE           <^X10>          ;BOARD NUMBER
5070 1092 43 44 49                              .ASCII          "IDC"
```

```
5073 1093 FF                        .BYTE       <^XFF>          ;END OF BOARD TABLE
5074 1094
5074 1095
5074 1096
5074 1097
5074 1098
5074 1099
5074 1100
5074 1101
5074 1102
5074 1103
5074 1104
5074 1105
5074 1106
5074 1107
5074 1108
5074 1109
```

```
;*****************************************************************************
;
;  SECTION NAME TABLE (NOTE: ALL ENTRIES MUST BE THE SAME NAME LENGTH (6))
;  AND ALL FILL CHARS MUST BE ZERO'S (BINARY ZEROS)
;  THE BYTE BEFORE EACH NAME IS USED TO SET SECTION FLAG AS FOLLOWS
;
;  SECTION_FLAG              FF = END OF TABLE
;                  BIT 7      0 = NAME IS SECTION NAME FOR WCS
;                                1 = NAME IS SECTION NAME FOR CONSOLE RAM
;                  BIT 6      1 = OPTIONAL TEST
;              BITS 0 - 5    XX = CPU CLUSTER BOARD NUMBER
;
;*****************************************************************************
```

```
5074 1110 83                SEC_TABLE:  .BYTE       <^X83>          ;8085 BASED CPU TESTS 1-8
5075 1111 41 42 4B 4E 45                .ASCII      "ENKBA"
507A 1112 00                            .BYTE       0
507B 1113
507B 1114 83                            .BYTE       <^X83>          ;8085 BASED CPU TESTS 9-F
507C 1115 42 42 4B 4E 45                .ASCII      "ENKBB"
5081 1116 00                            .BYTE       0
5082 1117
5082 1118 83                            .BYTE       <^X83>          ;8085 BASED CPU TESTS 10-1D
5083 1119 43 42 4B 4E 45                .ASCII      "ENKBC"
5088 1120 00                            .BYTE       0
5089 1121
5089 1122 81                            .BYTE       <^X81>          ;8085 BASED CPU TESTS 1E-26
508A 1123 44 42 4B 4E 45                .ASCII      "ENKBD"
508F 1124 00                            .BYTE       0
5090 1125
5090 1126 81                            .BYTE       <^X81>          ;8085 BASED CPU TESTS 27-2E
5091 1127 45 42 4B 4E 45                .ASCII      "ENKBE"
5096 1128 00                            .BYTE       0
5097 1129
5097 1130 C0                            .BYTE       <^XC0>          ;8085 BASED WCS MARCH TEST.
5098 1131 46 42 4B 4E 45                .ASCII      "ENKBF"         ; (OPTIONAL)
509D 1132 00                            .BYTE       0
509E 1133
509E 1134 C0                            .BYTE       <^XC0>          ;8085 BASED APT UART TEST
509F 1135 47 42 4B 4E 45                .ASCII      "ENKBG"         ; (OPTIONAL)
50A4 1136 00                            .BYTE       0
50A5 1137
50A5 1138 02                            .BYTE       <^X02>          ;WCS BASED CPU TEST
50A6 1139 41 43 4B 4E 45                .ASCII      "ENKCA"
50AB 1140 00                            .BYTE       0
50AC 1141
50AC 1142 02                            .BYTE       <^X02>          ;WCS BASED CPU TEST
50AD 1143 42 43 4B 4E 45                .ASCII      "ENKCB"
50B2 1144 00                            .BYTE       0
50B3 1145
50B3 1146 05                            .BYTE       <^X05>          ;WCS BASED MCT TEST
50B4 1147 43 43 4B 4E 45                .ASCII      "ENKCC"
50B9 1148 00                            .BYTE       0
50BA 1149
50BA 1150 02                            .BYTE       <^X02>          ;WCS BASED CPU TEST
50BB 1151 44 43 4B 4E 45                .ASCII      "ENKCD"
50C0 1152 00                            .BYTE       0
```

```
50C1 1153
50C1 1154 08                    .BYTE   <^X08>          ;WCS BASED FPA TEST
50C2 1155 45 43 4B 4E 45        .ASCII  "ENKCE"
50C7 1156 00                    .BYTE   0
50C8 1157
50C8 1158 10                    .BYTE   <^X10>          ;WCS BASED IDC TEST
50C9 1159 46 43 4B 4E 45        .ASCII  "ENKCF"
50CE 1160 00                    .BYTE   0
50CF 1161
50CF 1162 50                    .BYTE   <^X50>          ;WCS BASED IDC TEST
50D0 1163 47 43 4B 4E 45        .ASCII  "ENKCG"         ; (OPTIONAL)
50D5 1164 00                    .BYTE   0
50D6 1165
50D6 1166 10                    .BYTE   <^X10>          ;WCS BASED IDC TEST
50D7 1167 48 43 4B 4E 45        .ASCII  "ENKCH"         ; (OPTIONAL)
50DC 1168 00                    .BYTE   0
50DD 1169
50DD 1170 FF                    .BYTE   <^XFF>          ;END OF TABLE
50DE 1171
50DE 1172 00            SECTION_FLAG:   .BYTE   0       ;SECTION TYPE AND BOARD NUM
50DF 1173 000004E5      SECTION_NAME:   .BLKB   SEC_LEN ;SAVE BYTES FOR NAME
50E5 1174
50E5 1175               ;*****************************************************************************
50E5 1176               ;
50E5 1177               ;  UPC VERIFY ADDRESS TABLE
50E5 1178               ;
50E5 1179               ;  THIS TABLE OF VALUES IS COMPARED TO ACTUAL UPC VALUES TO VERIFY
50E5 1180               ;  THE WRITE OF DATA TO THE CPU IS WORKING CORRECTLY
50E5 1181               ;
50E5 1182               ;*****************************************************************************
50E5 1183
50E5 1184 0601          WRITE_TAB:      .WORD   <^X601>
50E7 1185 0602                          .WORD   <^X602>
50E9 1186 0603                          .WORD   <^X603>
50EB 1187 0604                          .WORD   <^X604>
50ED 1188 0606                          .WORD   <^X606>
50EF 1189 0603                          .WORD   <^X603>
50F1 1190 0605                          .WORD   <^X605>
50F3 1191 0606                          .WORD   <^X606>
50F5 1192               ;*****************************************************************************
50F5 1193               ;
50F5 1194               ;  CHAR STRINGS AND JUMP ADDRESSES FOR PARSING AT COMMAND LEVEL
50F5 1195               ;
50F5 1196               ;*****************************************************************************
50F5 1197
50F5 1198 03            MAIN_TAB:       .BYTE   3
50F6 1199 55 2F 58                      .ASCII  "X/U"           ;X COMMAND
50F9 1200 0D66'                         .WORD   XU
50FB 1201
50FB 1202 03                            .BYTE   3
50FC 1203 43 2F 58                      .ASCII  "X/C"           ;X COMMAND
50FF 1204 0D6E'                         .WORD   XC
5101 1205
5101 1206 03                            .BYTE   3
5102 1207 52 49 44                      .ASCII  "DIR"           ;DIR MUST COME B4 DI!!!!!
5105 1208 1DF1'                         .WORD   DIR_COMMAND     ;DIRECTORY COMMAND
5107 1209
5107 1210 02                            .BYTE   2
5108 1211 49 44                         .ASCII  "DI"            ;DI MUST COME B4 D!!!!!
510A 1212 070C'                         .WORD   DIAG_COMMAND    ;DIAGNOSE COMMAND
```

```
ZZ-ENKAA-2.1    4C00    4
510C 1213
510C 1214 02                .BYTE   2
510D 1215 48 53             .ASCII  "SH"
510F 1216 08D5'             .WORD   SHOW_COMMAND    ;SHOW COMMAND
5111 1217
5111 1218 02                .BYTE   2               ;CL MUST COME BEFORE C!!!!!
5112 1219 4C 43             .ASCII  "CL"
5114 1220 0A5A'             .WORD   CLEAR_COMMAND   ;CLEAR COMMAND
5116 1221
5116 1222 01                .BYTE   1
5117 1223 43                .ASCII  "C"
5118 1224 0946'             .WORD   CONT_COMMAND    ;CONTINUE COMMAND
511A 1225
511A 1226 02                .BYTE   2
511B 1227 45 52             .ASCII  "RE"            ;RE MUST COME BEFORE R!!!!
511D 1228 0705'             .WORD   STNDRD_CON_COM  ;RETURN TO STANDARD CONSOLE
511F 1229
511F 1230 01                .BYTE   1
5120 1231 52                .ASCII  "R"
5121 1232 0AD6'             .WORD   REPEAT_COMMAND  ;REPEAT COMMAND
5123 1233
5123 1234 02                .BYTE   2
5124 1235 45 53             .ASCII  "SE"
5126 1236 097D'             .WORD   SET_COMMAND     ;SET COMMAND
5128 1237
5128 1238 02                .BYTE   2
5129 1239 4F 4C             .ASCII  "LO"
512B 1240 0AC2'             .WORD   LOOP_COMMAND    ;LOOP COMMAND
512D 1241
512D 1242 03                .BYTE   3
512E 1243 45 2F 54          .ASCII  "T/E"
5131 1244 0705'             .WORD   STNDRD_CON_COM  ;RETURN TO STANDARD CONSOLE
5133 1245
5133 1246 02                .BYTE   2
5134 1247 58 45             .ASCII  "EX"            ;EX MUST COME B4 E!!!!!
5136 1248 0AE1'             .WORD   EXAM_COMMAND    ;EXAMINE COMMAND
5138 1249
5138 1250 01                .BYTE   1
5139 1251 45                .ASCII  "E"
513A 1252 0AE1'             .WORD   EXAM_COMMAND    ;EXAMINE COMPATABLE WITH STANDARD
513C 1253
513C 1254 02                .BYTE   2
513D 1255 45 44             .ASCII  "DE"            ;DE MUST COM B4 D!!!!!!
513F 1256 0C57'             .WORD   DEP_COMMAND     ;DEPOSIT COMMAND
5141 1257
5141 1258 01                .BYTE   1
5142 1259 44                .ASCII  "D"
5143 1260 0C57'             .WORD   DEP_COMMAND     ;DEPOSIT COMPATABLE WITH STANDARD
5145 1261
5145 1262 03                .BYTE   3
5146 1263 55 2F 53          .ASCII  "S/U"           ;START AT 8085 ADDRESS
5149 1264 0D4C'             .WORD   START
514B 1265
514B 1266 02                .BYTE   2
514C 1267 44 4C             .ASCII  "LD"            ;LOAD SECTION (NO EXECUTE)
514E 1268 0939'             .WORD   LOADER
5150 1269
5150 1270 01                .BYTE   1
5151 1271 49                .ASCII  "I"             ;INIT (EXEC WCS TEST 0)
5152 1272 1E8A'             .WORD   EXEC_TEST0
```

```
5154 1273
5154 1274 02                        .BYTE   2
5155 1275 42 41                     .ASCII  "AB"            ;ABORT (FUTURE EXPANSION)
5157 1276 1F6D'                     .WORD   START_UP
5159 1277
5159 1278 01                        .BYTE   1
515A 1279 0D                        .ASCII  <CR>            ;END ON LINE HAS <CR>
515B 1280 055E'                     .WORD   RETURN
515D 1281
515D 1282 FF                        .BYTE   <^XFF>          ;END OF TABLE
515E 1283
515E 1284 C9                RETURN: RET                     ;RETURN INSTRUCTION FOR <CR> CASE
515F 1285
515F 1286                   ;*******************************************************************************
515F 1287                   ;
515F 1288                   ;  CHAR STRINGS AND JUMP ADDRESSES FOR PARSING THE DIAGNOSE COMMAND
515F 1289                   ;
515F 1290                   ;*******************************************************************************
515F 1291
515F 1292 02                DIAG_TAB:       .BYTE   2
5160 1293 4F 42                             .ASCII  "BO"
5162 1294 079D'                             .WORD   DIAG_BOARD      ;DIAGNOSE BOARD
5164 1295
5164 1296 02                                .BYTE   2
5165 1297 45 53                             .ASCII  "SE"
5167 1298 07EB'                             .WORD   DIAG_SECTION    ;DIAGNOSE SECTION
5169 1299
5169 1300 02                                .BYTE   2
516A 1301 45 54                             .ASCII  "TE"
516C 1302 0858'                             .WORD   DIAG_TEST       ;DIAGNOSE TEST
516E 1303
516E 1304 02                                .BYTE   2
516F 1305 4F 43                             .ASCII  "CO"
5171 1306 089C'                             .WORD   DIAG_CONTIN     ;DIAGNOSE CONTINUE
5173 1307
5173 1308 02                                .BYTE   2
5174 1309 41 50                             .ASCII  "PA"
5176 1310 08A2'                             .WORD   DIAG_PASS       ;DIAGNOSE PASS
5178 1311
5178 1312 02                                .BYTE   2
5179 1313 48 53                             .ASCII  "SH"
517B 1314 08C7'                             .WORD   DIAG_SHORTEN    ;DIAGNOSE SHORTEN
517D 1315
517D 1316 01                                .BYTE   1
517E 1317 00                                .BYTE   0               ;END OF LINE HAS 0 AFTER <CR>
517F 1318 1EF4'                             .WORD   SET_PARDONE
5181 1319
5181 1320 FF                                .BYTE   <^XFF>          ;END OF TABLE
5182 1321
5182 1322                   ;*******************************************************************************
5182 1323                   ;
5182 1324                   ;  CHAR STRINGS AND JUMP ADDRESSES FOR THE CLEAR COMMAND
5182 1325                   ;
5182 1326                   ;*******************************************************************************
5182 1327
5182 1328 02                CLEAR_TAB:      .BYTE   2
5183 1329 41 48                             .ASCII  "HA"
5185 1330 0A61'                             .WORD   CLEAR_HALT      ;CLEAR HALT ON ERROR
5187 1331
5187 1332 02                                .BYTE   2
```

```
5188 1333 4F 4C
518A 1334 0A6B'                 .ASCII  "LO"
518C 1335                       .WORD   CLEAR_LOOP      ;CLEAR LOOP ON ERROR
518C 1336 02
518D 1337 45 4E                 .BYTE   2
518F 1338 0A74'                 .ASCII  "NE"
5191 1339                       .WORD   CLEAR_NER       ;CLEAR NO ERROR REPORTS
5191 1340 02
5192 1341 45 42                 .BYTE   2
5194 1342 0A79'                 .ASCII  "BE"
5196 1343                       .WORD   CLEAR_BELL      ;CLEAR BELL ON ERROR
5196 1344 02
5197 1345 45 53                 .BYTE   2
5199 1346 0A7E'                 .ASCII  "SE"
519B 1347                       .WORD   CLEAR_SER       ;CLEAR SINGLE BIT ERROR REPORTS
519B 1348 02
519C 1349 50 53                 .BYTE   2
519E 1350 0A83'                 .ASCII  "SP"
51A0 1351                       .WORD   CLEAR_SP        ;CLEAR SPECIAL BIT (FOR LOOPING ON
51A0 1352                                               ; SIZING TESTS)
51A0 1353 02
51A1 1354 52 54                 .BYTE   2
51A3 1355 0A95'                 .ASCII  "TR"
51A5 1356                       .WORD   CLEAR_TRACE     ;CLEAR TRACE MODE
51A5 1357 02
51A6 1358 4F 53                 .BYTE   2
51A8 1359 0A88'                 .ASCII  "SO"
51AA 1360                       .WORD   CLEAR_SOMM      ;CLEAR STOP ON MICRO MATCH
51AA 1361 02
51AB 1362 45 44                 .BYTE   2
51AD 1363 0A9A'                 .ASCII  "DE"
51AF 1364                       .WORD   CLEAR_DEFAULT   ;CLEAR FLAGS TO DEFAULT VALUES
51AF 1365 02
51B0 1366 41 50                 .BYTE   2
51B2 1367 0AAF'                 .ASCII  "PA"
51B4 1368                       .WORD   CLEAR_PAR       ;CLEAR PARITY COMMAND
51B4 1369 02
51B5 1370 49 46                 .BYTE   2
51B7 1371 0A43'                 .ASCII  "FI"            ;CLEAR IDC FIFO ADDRESS
51B9 1372                       .WORD   CLEAR_FIFO
51B9 1373 02
51BA 1374 52 42                 .BYTE   2
51BC 1375 0A28'                 .ASCII  "BR"            ;CLEAR IDC FIFO ADDRESS
51BE 1376                       .WORD   CLEAR_BREAK
51BE 1377 FF                    .BYTE   <^XFF>          ;END OF TABLE
51BF 1378
51BF 1379       ;*****************************************************************************
51BF 1380       ;
51BF 1381       ;  CHAR STRINGS AND JUMP ADDRESSES FOR THE SET COMMAND
51BF 1382       ;
51BF 1383       ;*****************************************************************************
51BF 1384
51BF 1385 02    SET_TAB:        .BYTE   2
51C0 1386 41 48                 .ASCII  "HA"
51C2 1387 0984'                 .WORD   SET_HALT        ;SET HALT ON ERROR
51C4 1388
51C4 1389 02                    .BYTE   2
51C5 1390 4F 4C                 .ASCII  "LO"
51C7 1391 098C'                 .WORD   SET_LOOP        ;SET LOOP ON ERROR
51C9 1392
```

```
51C9 1393 02
51CA 1394 45 4E
51CC 1395 0991'
51CE 1396
51CE 1397 02
51CF 1398 45 42
51D1 1399 0996'
51D3 1400
51D3 1401 02
51D4 1402 45 53
51D6 1403 099E'
51D8 1404
51D8 1405 02
51D9 1406 50 53
51DB 1407 09A3'
51DD 1408
51DD 1409
51DD 1410 02
51DE 1411 52 54
51E0 1412 09E3'
51E2 1413
51E2 1414 02
51E3 1415 4F 53
51E5 1416 09A8'
51E7 1417
51E7 1418 02
51E8 1419 45 44
51EA 1420 0A9A'
51EC 1421
51EC 1422 02
51ED 1423 41 50
51EF 1424 09E9'
51F1 1425
51F1 1426 02
51F2 1427 54 53
51F4 1428 09C6'
51F6 1429
51F6 1430 02
51F7 1431 52 42
51F9 1432 0A00'
51FB 1433
51FB 1434 02
51FC 1435 46 41
51FE 1436 0A37'
5200 1437
5200 1438 02
5201 1439 46 42
5203 1440 0A3D'
5205 1441
5205 1442 FF
5206 1443
5206 1444
5206 1445
5206 1446
5206 1447
5206 1448
5206 1449
5206 1450 02
5207 1451 41 52
5209 1452 0AEF'
```

```
                .BYTE   2
                .ASCII  "NE"
                .WORD   SET_NER         ;SET NO ERROR REPORTS

                .BYTE   2
                .ASCII  "BE"
                .WORD   SET_BELL        ;SET BELL ON ERROR

                .BYTE   2
                .ASCII  "SE"
                .WORD   SET_SER         ;SET SINGLE BIT ERROR REPORTS

                .BYTE   2
                .ASCII  "SP"
                .WORD   SET_SP          ;SET SPECIAL BIT (FOR LOOPING ON
                                        ; SIZING TESTS)

                .BYTE   2
                .ASCII  "TR"
                .WORD   SET_TRACE       ;SET TRACE MODE

                .BYTE   2
                .ASCII  "SO"
                .WORD   SET_SOMM        ;SET STOP ON MICRO MATCH

                .BYTE   2
                .ASCII  "DE"
                .WORD   CLEAR_DEFAULT   ;CLEAR FLAGS TO DEFAULT VALUE

                .BYTE   2
                .ASCII  "PA"
                .WORD   SET_PAR         ;SET PARITY COMMAND

                .BYTE   2
                .ASCII  "ST"
                .WORD   SET_STEP        ;SINGLE STEP

                .BYTE   2
                .ASCII  "BR"            ;SET BREAK POINT
                .WORD   SET_BREAK

                .BYTE   2
                .ASCII  "AF"            ;SELECT IDC FIFO A
                .WORD   SEL_FIFO_A

                .BYTE   2
                .ASCII  "BF"            ;SELECT IDC FIFO B
                .WORD   SEL_FIFO_B

                .BYTE   <^XFF>          ;END OF TABLE

;*****************************************************************************
;
;  CHAR STRINGS AND JUMP ADDRESSES FOR THE EXAMINE COMMAND
;
;*****************************************************************************

EXAM_TAB:       .BYTE   2
                .ASCII  "RA"
                .WORD   EXAM_RAM        ;EXAMINE 8085 RAM
```

```
520B 1453
520B 1454 02                .BYTE   2
520C 1455 50 55             .ASCII  "UP"            ;UP MUST COME BEFORE U!!!!!
520E 1456 0B2C'             .WORD   EXAM_UPC        ;EXAMINE UPC REGISTER
5210 1457
5210 1458 02                .BYTE   2
5211 1459 42 55             .ASCII  "UB"            ;UB MUST COME BEFORE U!!!!!
5213 1460 0B9B'             .WORD   EXAM_UBS
5215 1461
5215 1462 01                .BYTE   1
5216 1463 55                .ASCII  "U"
5217 1464 0AEF'             .WORD   EXAM_RAM        ;E/U FOR COMPATABILITY WITH STANDARD
5219 1465
5219 1466 02                .BYTE   2
521A 1467 53 43             .ASCII  "CS"            ;CS MUST COME BEFORE C
521C 1468 0B19'             .WORD   EXAM_CSR        ;EXAMINE CSR REGISTER
521E 1469
521E 1470 01                .BYTE   1
521F 1471 43                .ASCII  "C"
5220 1472 0BCA'             .WORD   EXAM_WCS        ;E/C FOR COMPATABILITY WITH STANDARD
5222 1473
5222 1474 02                .BYTE   2
5223 1475 43 4D             .ASCII  "MC"
5225 1476 0B92'             .WORD   EXAM_MCT        ;EXAMINE MEMORY CONTROL + STATUS
5227 1477
5227 1478 02                .BYTE   2
5228 1479 52 57             .ASCII  "WR"
522A 1480 0B4E'             .WORD   EXAM_WRK        ;EXAMINE 2901 WORKING REGS
522C 1481
522C 1482 02                .BYTE   2
522D 1483 53 4C             .ASCII  "LS"
522F 1484 0B7C'             .WORD   EXAM_LS         ;EXAMINE LS LOCATIONS
5231 1485
5231 1486 02                .BYTE   2
5232 1487 42 54             .ASCII  "TB"
5234 1488 0BA4'             .WORD   EXAM_TB         ;EXAMINE TRANSLATION BUFFER
5236 1489
5236 1490 02                .BYTE   2
5237 1491 4D 4D             .ASCII  "MM"
5239 1492 0BAD'             .WORD   EXAM_MM         ;EXAMINE MAIN MEMORY
523B 1493
523B 1494 02                .BYTE   2
523C 1495 53 4F             .ASCII  "OS"
523E 1496 0BB6'             .WORD   EXAM_OS         ;EXAMINE OS REGISTER
5240 1497
5240 1498 02                .BYTE   2
5241 1499 43 57             .ASCII  "WC"
5243 1500 0BCA'             .WORD   EXAM_WCS        ;EXAMINE WCS MEMORY
5245 1501
5245 1502 02                .BYTE   2
5246 1503 43 49             .ASCII  "IC"
5248 1504 0BE0'             .WORD   EXAM_ICSR       ;EXAMINE IDC CSR
524A 1505
524A 1506 02                .BYTE   2
524B 1507 44 49             .ASCII  "ID"
524D 1508 0BE9'             .WORD   EXAM_IDAR       ;EXAMINE IDC DISK ADD REG
524F 1509
524F 1510 02                .BYTE   2
5250 1511 42 44             .ASCII  "DB"
5252 1512 0BF2'             .WORD   EXAM_DBUF       ;EXAMINE IDC DATA BUFFER
```

```
5254 1513
5254 1514 02                    .BYTE   2
5255 1515 41 50                 .ASCII  "PA"
5257 1516 0BFB'                 .WORD   EXAM_PATT       ;EXAMINE IDC ECC PATTERN
5259 1517
5259 1518 02                    .BYTE   2
525A 1519 4F 50                 .ASCII  "PO"
525C 1520 0C04'                 .WORD   EXAM_POSIT      ;EXAMINE IDC ECC POSITION
525E 1521
525E 1522 FF                    .BYTE   <^XFF>          ;END OF TABLE
525F 1523
525F 1524
525F 1525              ;*****************************************************************
525F 1526              ;
525F 1527              ;  CHAR STRINGS AND JUMP ADDRESSES FOR THE DEPOSIT COMMAND
525F 1528              ;
525F 1529              ;*****************************************************************
525F 1530
525F 1531 02           DEP_TAB:         .BYTE   2
5260 1532 41 52                         .ASCII  "RA"
5262 1533 0C5D'                         .WORD   DEP_RAM         ;DEPOSIT 8085 RAM
5264 1534
5264 1535 02                            .BYTE   2               ;UP MUST COME BEFORE U!!!!!!
5265 1536 50 55                         .ASCII  "UP"
5267 1537 0C89'                         .WORD   DEP_UPC         ;DEPOSIT UPC REGISTER
5269 1538
5269 1539 02                            .BYTE   2
526A 1540 42 55                         .ASCII  "UB"            ;UB MUST COME BEFORE U!!!!!
526C 1541 0D13'                         .WORD   DEP_UBS
526E 1542
526E 1543 01                            .BYTE   1
526F 1544 55                            .ASCII  "U"
5270 1545 0C5D'                         .WORD   DEP_RAM         ;D/U FOR COMPATABILITY WITH STANDARD
5272 1546
5272 1547 02                            .BYTE   2
5273 1548 53 43                         .ASCII  "CS"            ;CS MUST COME BEFORE C!!!!!
5275 1549 0C76'                         .WORD   DEP_CSR         ;DEPOSIT CSR REGISTER
5277 1550
5277 1551 01                            .BYTE   1
5278 1552 43                            .ASCII  "C"
5279 1553 0CC2'                         .WORD   DEP_WCS         ;D/C FOR COMPATABILITY WITH STANDARD
527B 1554
527B 1555 02                            .BYTE   2
527C 1556 43 4D                         .ASCII  "MC"
527E 1557 0D01'                         .WORD   DEP_MCT         ;DEPOSIT MEMORY CONTROL + STATUS
5280 1558
5280 1559 02                            .BYTE   2
5281 1560 52 57                         .ASCII  "WR"
5283 1561 0C95'                         .WORD   DEP_WRK         ;DEPOSIT 2901 WORKING REGS
5285 1562
5285 1563 02                            .BYTE   2
5286 1564 53 4C                         .ASCII  "LS"
5288 1565 0CEC'                         .WORD   DEP_LS          ;DEPOSIT LS LOCATIONS
528A 1566
528A 1567 02                            .BYTE   2
528B 1568 42 54                         .ASCII  "TB"
528D 1569 0D0D'                         .WORD   DEP_TB          ;DEPOSIT TRANSLATION BUFFER
528F 1570
528F 1571 02                            .BYTE   2
5290 1572 4D 4D                         .ASCII  "MM"
```

```
5292 1573 0D07'                         .WORD   DEP_MM          ;DEPOSIT MAIN MEMORY
5294 1574
5294 1575 02                            .BYTE   2
5295 1576 53 4F                         .ASCII  "OS"
5297 1577 0CDE'                         .WORD   DEP_OS          ;DEPOSIT OS REGISTER
5299 1578
5299 1579 02                            .BYTE   2
529A 1580 43 57                         .ASCII  "WC"
529C 1581 0CC2'                         .WORD   DEP_WCS         ;DEPOSIT WCS MEMORY
529E 1582
529E 1583 02                            .BYTE   2
529F 1584 43 49                         .ASCII  "IC"
52A1 1585 0D19'                         .WORD   DEP_ICSR        ;DEPOSIT IDC CSR
52A3 1586
52A3 1587 02                            .BYTE   2
52A4 1588 44 49                         .ASCII  "ID"
52A6 1589 0D1F'                         .WORD   DEP_IDAR        ;DEPOSIT IDC DISK ADD REG
52A8 1590
52A8 1591 02                            .BYTE   2
52A9 1592 42 44                         .ASCII  "DB"
52AB 1593 0D25'                         .WORD   DEP_DBUF        ;DEPOSIT IDC DATA BUFFER
52AD 1594
52AD 1595 FF                            .BYTE   <^XFF>          ;END OF TABLE
52AE 1596
52AE 1597
52AE 1598                       ;*******************************************************************************
52AE 1599                       ;
52AE 1600                       ;  FLAG LITERALS
52AE 1601                       ;
52AE 1602                       ;*******************************************************************************
52AE 1603
52AE 1604 20 50 4F 4F 4C        PRT_FLAGS_A:
52B3 1605 20 20 52 45 4E                        .ASCII          "LOOP "
52B8 1606 20 4C 4C 45 42                        .ASCII          "NER  "
52BD 1607 20 54 4C 41 48                        .ASCII          "BELL "
52C2 1608 20 20 52 45 53                        .ASCII          "HALT "
52C7 1609 20 45 43 52 54                        .ASCII          "SER  "
52CC 1610 20 4D 4D 4F 53                        .ASCII          "TRCE "
52D1 1611 20 43 45 50 53                        .ASCII          "SOMM "
52D6 1612                                       .ASCII          "SPEC "
52D6 1613
52D6 1614                       BEGIN_CHECKSUM:
52D6 1615                       ;*******************************************************************************
52D6 1616                       ;
52D6 1617                       ;  COMMAND SUBROUTINES
52D6 1618                       ;
52D6 1619                       ;*******************************************************************************
52D6 1620
52D6 1621                       ;*******************************************************************************
52D6 1622                       ;
52D6 1623                       ;  APT_WCS_TEST
52D6 1624                       ;  THIS ROUTINE IS ONLY EXECUTED WHEN APT IS RUNNING THE SHOW.  AFTER THE
52D6 1625                       ;  LOAD OF A WCS DIAGNOSTIC VIA THE 'X' COMMAND, THIS ROUTINE IS ENTERED
52D6 1626                       ;  VIA A 'S/U 4C06' COMMAND.  IT IS RESPONSIBLE FOR STARTING THE WCS
52D6 1627                       ;  DIAGNOSTIC WITHOUT A 'DI' COMMAND OF ANY KIND.
52D6 1628                       ;
52D6 1629                       ;*******************************************************************************
52D6 1630 AF                    APT_WCS_TEST:   XRA     A
52D7 1631 0209'32                               STA     CONSOLE_TEST            ;CLEAR CONSOLE TEST FLAG
52DA 1632 3C                                    INR     A                       ;SET WCS_LOADED FLAG TO
```

```
52DB 1633 0408'32                       STA     WCS_LOADED              ; INDICATE WCS CODE IS LOADED
52DE 1634
52DE 1635 1E8A'CD                       CALL    EXEC_TEST0              ;LOAD LS CONSTANTS AND MAIN
52E1 1636                                                               ; MEMORY CONSTANTS, IF NECES-
52E1 1637                                                               ; SARY BY RUNNING WCS TEST 0
52E1 1638 06EC'C3                       JMP     APT_START               ;LOAD BUFFER AND EXECUTE
52E4 1639
52E4 1640       ;*******************************************************************************
52E4 1641       ;
52E4 1642       ;  APT_CON_TEST
52E4 1643       ;  THIS ROUTINE IS ONLY EXECUTED WHEN APT IS RUNNING THE SHOW.  AFTER THE
52E4 1644       ;  LOAD OF AN 8085 DIAGNOSTIC VIA THE 'X' COMMAND, THIS ROUTINE IS ENTERED
52E4 1645       ;  VIA A 'S/U 4C03' COMMAND.  IT IS RESPONSIBLE FOR STARTING THE 8085
52E4 1646       ;  DIAGNOSTIC WITHOUT A 'DI' COMMAND OF ANY KIND.
52E4 1647       ;
52E4 1648       ;*******************************************************************************
52E4 1649
52E4 1650 AF    APT_CON_TEST:   XRA     A                               ;CLEAR WCS_LOADED FLAG
52E5 1651 0408'32               STA     WCS_LOADED                      ;INDICATE NO WCS CODE LOADED
52E8 1652 3C                    INR     A                               ;SET CONSOLE TEST FLAG
52E9 1653 0209'32               STA     CONSOLE_TEST
52EC 1654
52EC 1655
52EC 1656 01'B6'21      APT_START:      LXI     H,APT_DI_COMMAND        ;POINT TO 'DI TE 1 C0' ASCII
52EF 1657 01'64'11              LXI     D,TTBUF                         ;POINT TO INPUT BUFFER
52F2 1658 0C 0E                 MVI     C,12
52F4 1659 11A4'CD               CALL    MOVER_R                         ;MOV DI COMMAND TO INPUT BUFFER
52F7 1660 01'64'21              LXI     H,TTBUF                         ;HL GETS ADDRESS OF BUFFER
52FA 1661 03E0'22               SHLD    TTBUF_PNT                       ;SAVE IN MEM
52FD 1662 0334'32 3C AF         SET     RESTART                         ;ALLOW PARSE WITHOUT INPUT LINE
5302 1663 2022'C3               JMP     PARSER                          ;PARSE COMMAND LINE
5305 1664
5305 1665
5305 1666       ;*******************************************************************************
5305 1667       ;
5305 1668       ;       STNDRD_CON_COM
5305 1669       ;       THIS ROUTINE IS ENTERED IF A RE, T/E, OR S/U 0 COMMAND IS ISSUED
5305 1670       ;       FROM THE CONSOLE DEVICE.  IT WILL INITIALIZE THE INTERVAL TIMER
5305 1671       ;       BY DOING A MASTER RESET TO IT AND THEN GO TO THE CONSOLE RETURN
5305 1672       ;       ROUTINE AT ADDRESS 414D OF THE BOOT BLOCK.
5305 1673       ;
5305 1674       ;*******************************************************************************
5305 1675
5305 1676 FF 3E         STNDRD_CON_COM: MVI     A,HEX_FF                ;LOAD ACCUM WITH FF
5307 1677 1D D3                 OUT     ITCSR                           ;SEND FF TO INTERVAL TIMER (MASTER
5309 1678                                                               ; RESET)
5309 1679 414D C3               JMP     RET_VECTOR                      ;RETURN TO STANDARD CONSOLE
530C 1680
530C 1681
530C 1682       ;*******************************************************************************
530C 1683       ;
530C 1684       ;  DIAG_COMMAND
530C 1685       ;  THE DIAGNOSE COMMAND CAUSES EXECUTION OF DIAGNOSTICS IN ONE OF THE
530C 1686       ;  FOLLOWING MODES
530C 1687       ;       1) BY THE DIAGNOSTIC NAME
530C 1688       ;       2) BY THE MODULE NAME
530C 1689       ;       3) ALL NON-OPTIONAL DIAGNOSTICS
530C 1690       ;
530C 1691       ;*******************************************************************************
530C 1692
```

```
530C 1693 0F 003A'3A            ****************** DIAG_COMMAND:   IF      CONTINUE
          0722'D2               *
5313 1694                       *
5313 1695 AF                    *                                   XRA     A
5314 1696 003A'32               *                                   STA     CONTINUE            ;THIS COMMAND IS ILLEGAL IN
5317 1697 3C                    *                                   INR     A                   ; CONT MODE SO PUT BACK TO
5318 1698 0334'32               *                                   STA     RESTART             ; NORMAL MODE AND RESTART PROG
531B 1699                       *
531B 1700 01'64'21              *                                   LXI     H,TTBUF             ; BACK AT PARSER SO STACK
531E 1701 03E0'22               *                                   SHLD    TTBUF_PNT           ; WILL REFLECT A NORMAL ENTRY
5321 1702                       *
5321 1703 C9                    *                                   RET
5322 1704                       *
5322 1705                       ******************                  ENDIF
5322 1706
5322 1707 0F 0038'3A            ******************                  IF      APT                 ;IF APT, SET FLAG IN LS
          0731'D2               *
5329 1708 20 06                 *                                   MVI     B,APT_S_DEF         ;GET BIT FOR APT PRESENT
532B 1709 159E'CD               *                                   CALL    SET_LS_STAT         ;WRITE IN LS ERROR CONTROL
532E 1710 0736'C3               ******************                  ELSE
5331 1711 DF 06                 *                                   MVI     B,APT_C_DEF         ;GET BIT TO CLEAR FOR NO APT
5333 1712 15A8'CD               *                                   CALL    CLEAR_LS_STAT       ;CLEAR LS ERROR CONTROL BIT
5336 1713                       ******************                  ENDIF
5336 1714
5336 1715 001C'3A                                                   LDA     APT_MEM_SIZE        ;GET MEM SIZE THAT APT SENT .
5339 1716 00AC'32                                                   STA     DATA3               ;PUT IN BYTE 0 STORAGE
533C 1717 001D'3A                                                   LDA     APT_HARD_FLG        ;GET APT HARDWARE CONFIG
533F 1718 00AB'32                                                   STA     DATA2               ;PUT IN BYTE 1 STORAGE
5342 1719 001E'3A                                                   LDA     APT_SOFT_FLG        ;GET APT SOFTWARE CONFIG
5345 1720 00AA'32                                                   STA     DATA1               ;PUT IN BYTE 2 STORAGE
5348 1721 001F'3A                                                   LDA     APT_SOFT_FLG+1      ;GET APT RESERVED BYTE
534B 1722 00A9'32                                                   STA     DATA0               ;PUT IN BYTE 3 STORAGE
534E 1723
534E 1724 49 3E                                                     MVI     A,APT_PARAM         ;LS LOCATION TO WRITE
5350 1725 1415'CD                                                   CALL    WRITE_LS_R          ;WRITE INTO LS
5353 1726
5353 1727 023B'32 3C AF                                             SET     EXECUTE             ;CAUSES EXIT OF PARSER
5358 1728
5358 1729 02 3E                                                     MVI     A,2
535A 1730 03B3'32                                                   STA     TABLE_MODE          ;SET TO ALL SECTION MODE(D)
535D 1731
535D 1732 04'74'21                                                  LXI     H,SEC_TABLE
5360 1733 03B4'22                                                   SHLD    TABLE_PNT           ;RESET TABLE POINTER
5363 1734
5363 1735 2134'CD                                                   CALL    SET_ALL_TESTS       ;SET UP TEST NUM FOR ALL
5366 1736                                                                                       ; TESTS AND CLEAR EOS FLG
5366 1737 AF                                                        XRA     A
5367 1738 0314'32                                                   STA     PASS_CNT
536A 1739 3C                                                        INR     A                   ;A NOW CONTAINS 1
536B 1740 0315'32                                                   STA     PASS_CNT+1          ;SET TO 1 PASS (DEFAULT)
536E 1741
536E 1742 0F 0038'3A            ******************                  IF      APT
          077D'D2               *
5375 1743                       *
5375 1744 FF 3E                 *                                   MVI     A,HEX_FF
5377 1745 0314'32               *                                   STA     PASS_CNT            ;SET TO INFINITE PASSES
537A 1746 0315'32               *                                   STA     PASS_CNT+1
537D 1747                       *
537D 1748                       ******************                  ENDIF
537D 1749
```

```
537D 1750 AF                                        XRA     A
537E 1751 0018'32                                   STA     APT_PASS_CNT            ;INIT MAILBOX PASS CNT
5381 1752 0019'32                                   STA     APT_PASS_CNT+1          ;TO 0
5384 1753 030F'32                                   STA     PARDONE                 ;CLEAR RE-ENTRY PARSING FLAG
5387 1754 02C8'32                                   STA     MICRO_STEP              ;SET TO NO MICRO STEPPING
538A 1755
538A 1756 05'5F'21                          1$:     LXI     H,DIAG_TAB              ;PARSE DIAG COMMAND LINE
538D 1757 1FDB'CD                                   CALL    PAR_TEMP_BUF            ;UNTIL ERROR OR <CR>
5390 1758
5390 1759 030F'3A                                   LDA     PARDONE                 ;GET FLAG
5393 1760 B7                                        ORA     A                       ;SET CONDITION CODES
5394 1761 C0                                        RNZ                             ;RETURN IF PARDONE (<CR>)
5395 1762
5395 1763 02F6'3A                                   LDA     NOT_FOUND               ;GET NOT FOUND FLAG
5398 1764 B7                                        ORA     A                       ;SET CONDITION CODES
5399 1765 C0                                        RNZ                             ;RET IF NOT FOUND
539A 1766
539A 1767 078A'C3                                   JMP     1$
539D 1768
539D 1769
539D 1770
539D 1771
539D 1772
539D 1773
539D 1774
539D 1775
539D 1776
539D 1777
539D 1778
539D 1779
539D 1780
539D 1781
539D 1782
539D 1783
539D 1784
539D 1785
539D 1786
539D 1787
539D 1788
```

```
;*******************************************************************************
;
;  DIAG_BOARD
;  THIS ROUTINE INPUTS A BOARD NAME WHILE PARSING THE DIAGNOSE COMMAND
;  THIS BOARD NAME IS THEN LOOKED UP ON A TABLE AND A BOARD NUMBER IS
;  FOUND. TABLE_MODE WILL BE SET SO THAT ONLY SECTION NAMES WITH MATCHING
;  BOARD NUMBERS WILL BE USED.  THE TABLE WILL BE AS FOLLOWS
;
;       NUMBER  NAME
;
;       1       WCS (M8394)
;       2       DAP OR CPU (M8390)
;       4       MCT (M8391)
;       8       FPA (M8389)
;       10      IDC (M8388)
;
;        FF (TERMINATOR)
;
;*******************************************************************************
```

```
539D 1789 03E0'2A                           DIAG_BOARD: LHLD    TTBUF_PNT
53A0 1790 03C6'22                                       SHLD    TEMP_WORD               ;SAVE POINTER TO BOARD NAME
53A3 1791 1063'CD                                       CALL    SKIP_TO_SPEC            ;SKIP OVER IT TO GET COUNT AND
53A6 1792 03C6'2A                                       LHLD    TEMP_WORD               ;MOVE BOARD NAME TO SAVE AREA
53A9 1793 01'CE'11                                      LXI     D,BOARD_NAME
53AC 1794 118E'CD                                       CALL    MOVER_3_R
53AF 1795
53AF 1796 04'5B'21                                      LXI     H,BOARD_TABLE
53B2 1797 01DB'22                                       SHLD    BOARD_PNT               ;SET BOARD TABLE POINTER
53B5 1798
53B5 1799 03B3'32 3C AF                                 SET     TABLE_MODE              ;BOARD MODE
53BA 1800
53BA 1801 01DB'2A                           1$:         LHLD    BOARD_PNT               ;MOVE ENTRY IN BOARD TABLE
53BD 1802 01'D4'11                                      LXI     D,BOARD_BUF             ;TO BUFFER AREA
53C0 1803 11A2'CD                                       CALL    MOVER_4_R
53C3 1804 01DB'22                                       SHLD    BOARD_PNT               ;UPDATE BOARD TABLE POINTER
53C6 1805
53C6 1806 FF FE 01D4'3A     ******************          IFEQI   BOARD_BUF,HEX_FF        ;END OF TABLE?
          07D3'C2           *
53CE 1807 01 06             *                           MVI     B,<^X01>                ;BOARD NOT FOUND
53D0 1808 0F79'C3           *                           JMP     ERROR_ROUTINE_NO_RET    ;REPORT ERROR, DON'T RETURN
```

```
53D3 1809                ******************
53D3 1810                *
53D3 1811 01'CE'21       *                      IFEQ    BOARD_NAME,BOARD_BUF+1,3 ;COMPARE BOARD TO NAME GIVEN
          03 0E 01'D5'11 *
          07E8'C2 00CF'CD *
53E1 1812                *
53E1 1813 01D4'3A        *                      LDA     BOARD_BUF                ;EQUAL SAVE BOARD #
53E4 1814 01DA'32        *                      STA     BOARD_NUM
53E7 1815 C9             *                      RET                              ;DONE
53E8 1816                *
53E8 1817                ******************     ENDIF
53E8 1818
53E8 1819 07BA'C3                               JMP     1$                       ;LOOP UNTIL ERROR OR BOARD
53EB 1820                                                                        ; FOUND
53EB 1821
53EB 1822
53EB 1823
53EB 1824
53EB 1825
53EB 1826
53EB 1827
53EB 1828
53EB 1829
53EB 1830
53EB 1831
53EB 1832
53EB 1833 03B3'32 AF            DIAG_SECTION:   CLEAR   TABLE_MODE               ;SET FOR NO TABLE USAGE
53EF 1834
53EF 1835 06 0E 03'CD'21                        ZERO    TEMP_SECTION,SEC_LEN     ;START WITH CLEAR FIELD
          0D 23 77 AF
          07F5'C2
53FB 1836
53FB 1837 03E0'2A                               LHLD    TTBUF_PNT
53FE 1838 03C6'22                               SHLD    TEMP_WORD                ;SAVE POINTER TO SECTION
5401 1839 1063'CD                               CALL    SKIP_TO_SPEC             ;SKIP OVER IT TO GET COUNT
5404 1840 03C6'2A                               LHLD    TEMP_WORD                ;MOVE NAME TO SAVE AREA
5407 1841 03'CD'11                              LXI     D,TEMP_SECTION
540A 1842 037B'3A                               LDA     SKIP_CNT
540D 1843 4F                                    MOV     C,A                      ;SECT NAME LENGTH
540E 1844 11A4'CD                               CALL    MOVER_R
5411 1845
5411 1846 04'74'21                              LXI     H,SEC_TABLE
5414 1847 03B4'22                               SHLD    TABLE_PNT                ;RESET TABLE POINTER
5417 1848
5417 1849 03B4'2A                       1$:     LHLD    TABLE_PNT                ;MOVE ENTRY IN SEC TABLE TO
541A 1850 04'DE'11                              LXI     D,SECTION_FLAG           ;BUFFER AREA
541D 1851 07 0E                                 MVI     C,SEC_LEN+1              ;COUNT
541F 1852 11A4'CD                               CALL    MOVER_R
5422 1853 03B4'22                               SHLD    TABLE_PNT                ;UPDATE SECTION TABLE POINTER
5425 1854
5425 1855 FF FE 04DE'3A  ******************     IFEQI   SECTION_FLAG,HEX_FF      ;END OF TABLE
          0843'C2        *
542D 1856                *
542D 1857 04DE'32 AF     *                      CLEAR   SECTION_FLAG             ;MAKE WCS DEFAULT AREA
5431 1858 03'CD'21       *                      LXI     H,TEMP_SECTION           ;NAME IS NOT FOUND IN TABLE
5434 1859 04'DF'11       *                      LXI     D,SECTION_NAME           ;SO LOAD IT INTO WCS IF IT
5437 1860 06 0E          *                      MVI     C,SEC_LEN                ;IS FOUND ON THE TAPE
5439 1861 11A4'CD        *                      CALL    MOVER_R
543C 1862                *
543C 1863 1D6C'CD        *                      CALL    LOAD_SECTION
```

;**********
;
; DIAG_SECTION
; THIS SUBROUTINE GETS THE SECTION PARAMETER OF A DIAGNOSE COMMAND
; AND LOOKS IT UP IN A TABLE TO SEE WHETHER IT IS A CPU MICRO CODE
; OR AN 8085 DIAGNOSTIC. IF THE NAME IS NOT FOUND IN THE TABLE IT
; IS ASSUMED TO BE A CPU MICRO CODE DIAGNOSTIC. THE SECTION IS THEN
; LOADED INTO THE APPROPRIATE AREA.
;
;**********

```
543F 1864 C9                    *                       RET                             ;EXIT FROM SEARCH LOOP
5440 1865                       *
5440 1866 0855'C3               ******************      ELSE
5443 1867                       *
5443 1868 03'CD'21              * ****************      IFEQ    TEMP_SECTION,SECTION_NAME,SEC_LEN
          06 0E 04'DF'11        * *
          0855'C2 00CF'CD       * *
5451 1869                       * *
5451 1870 1D6C'CD               * *                     CALL    LOAD_SECTION
5454 1871 C9                    * *                     RET                             ;EXIT FROM SEARCH LOOP
5455 1872                       * *
5455 1873                       * ****************      ENDIF
5455 1874                       *
5455 1875                       ******************      ENDIF
5455 1876
5455 1877 0817'C3                                       JMP     1$                      ;LOOP UNTIL SECTION FOUND
5458 1878
5458 1879                               ;*******************************************************************************
5458 1880                               ;
5458 1881                               ;  DIAG_TEST
5458 1882                               ;  THIS SUBROUTINE INPUTS THE TEST PARAMETERS OF THE DIAGNOSE KEYWORD
5458 1883                               ;  ONE TEST NUMBER MEANS EXECUTE THAT TEST ONLY
5458 1884                               ;  TWO TEST NUMBERS MEANS EXECUTE THE TESTS BETWEEN THAT RANGE INCLUSIVE
5458 1885                               ;
5458 1886                               ;*******************************************************************************
5458 1887
5458 1888 0F 0209'3A            ****************** DIAG_TEST:    IFN     CONSOLE_TEST            ;IF CONSOLE TEST, SKIP THIS
          086F'DA               *
545F 1889                       *
545F 1890 0F 0408'3A            * ****************      IFN     WCS_LOADED              ;ELSE SEE IF WCS LOADED
          086F'DA               * *
5466 1891                       * *                                                     ; IF NOT, REPORT ERROR
5466 1892 023B'32 AF            * *                     CLEAR   EXECUTE                 ;CLEAR EXECUTE FLAG TO PROMPT
546A 1893 28 06                 * *                     MVI     B,<^X28>                ;CODE FOR NO TEST CODE ERROR
546C 1894 0F79'C3               * *                     JMP     ERROR_ROUTINE_NO_RET    ;REPORT ERROR
546F 1895                       * ****************      ENDIF
546F 1896                       *
546F 1897                       ******************      ENDIF
546F 1898
546F 1899 03B3'32 AF                                    CLEAR   TABLE_MODE              ;SET TO NO TABLE MODE
5473 1900
5473 1901 0FCF'CD                                       CALL    INPUT_NUM_IF            ;LOOK FOR FIRST TEST NUMBER
5476 1902
5476 1903 02 06                                         MVI     B,<^X02>                ;NO FIRST NUMBER - ERROR
5478 1904 0231'3A                                       LDA     ERROR                   ;GET ERROR FLAG
547B 1905 B7                                            ORA     A                       ;SET CONDITION CODES
547C 1906 0F79'C2                                       JNZ     ERROR_ROUTINE_NO_RET    ;REPORT IF ERROR.  DON'T RETURN
547F 1907
547F 1908 0440'3A                                       LDA     INPUT_NUM+3             ;FIRST NUMBER IS FIRST TEST
5482 1909 03DA'32                                       STA     TEST_NUM1               ;SAVE IT
5485 1910
5485 1911 0FCF'CD                                       CALL    INPUT_NUM_IF            ;GET SECOND NUMBER IF THERE
5488 1912
5488 1913 0F 0231'3A            ******************      IF      ERROR
          0895'D2               *
548F 1914 03DA'3A               *                       LDA     TEST_NUM1               ;NO SECOND NUM - USE FIRST TEST
5492 1915                       *
5492 1916 0898'C3               ******************      ELSE
5495 1917                       *
5495 1918 0440'3A               *                       LDA     INPUT_NUM+3             ;SAVE ENDING TEST NUMBER
```

```
5498 1919                          ******************
5498 1920
5498 1921 03DB'32                                       STA     TEST_NUM2
549B 1922
549B 1923 C9                                            RET
549C 1924
549C 1925
549C 1926                       ;*************************************************************************
549C 1927                       ;
549C 1928                       ;  DIAG_CONTIN
549C 1929                       ;  THIS SUBROUTINE HANDLES THE DIAGNOSE CONTINUE KEYWORD
549C 1930                       ;  A CONTINUE WILL CAUSE THE TESTING TO GO TO THE END OF THE SECTION
549C 1931                       ;
549C 1932                       ;  NOTE: - - TO BE EFFECTIVE THIS KEYWORD MUST BE GIVEN AFTER THE TEST
549C 1933                       ;            KEYWORD AND THE TEST KEYWORD MUST HAVE ONLY 1 PARAMETER
549C 1934                       ;            IF THE TEST PARAMETER IS GIVEN AFTER THE CONTINUE KEYWORD
549C 1935                       ;            IT WILL BE CANCELED OUT.
549C 1936                       ;
549C 1937                       ;*************************************************************************
549C 1938
549C 1939 50 3E                 DIAG_CONTIN:    MVI     A,HEX_50                ;MOVE LAST POSSIBLE TEST
549E 1940 03DB'32                               STA     TEST_NUM2               ;TO ENDING TEST NUMBER
54A1 1941
54A1 1942 C9                                    RET
54A2 1943
54A2 1944
54A2 1945                       ;*************************************************************************
54A2 1946                       ;
54A2 1947                       ;  DIAG_PASS
54A2 1948                       ;  THIS SUBROUTINE INPUTS THE DIAGNOSE PASS COUNT.
54A2 1949                       ;  IF NOT SPECIFIED THE PASS COUNT WILL DEFAULT TO ONE.
54A2 1950                       ;  IF THE PASS COUNT IS SPECIFED AS -1 THIS INDICATES A CONTINUOUS LOOP
54A2 1951                       ;
54A2 1952                       ;*************************************************************************
54A2 1953
54A2 1954 0FCF'CD               DIAG_PASS:      CALL    INPUT_NUM_IF            ;GET THE PASS COUNT
54A5 1955
54A5 1956 03 06                                 MVI     B,<^X03>                ;PASS COUNT ERROR
54A7 1957 0231'3A                               LDA     ERROR                   ;GET ERROR FLAG
54AA 1958 B7                                    ORA     A                       ;SET CONDITION CODES
54AB 1959 0F79'C2                               JNZ     ERROR_ROUTINE_NO_RET    ;REPORT IF ERROR.  DON'T RETURN
54AE 1960
54AE 1961 0F 02C9'3A                ******************  IF      MINUS
          08C0'D2                   *
54B5 1962                           *
54B5 1963 FF 3E                     *                   MVI     A,HEX_FF
54B7 1964 0314'32                   *                   STA     PASS_CNT            ;SET TO INFINITE PASSES
54BA 1965 0315'32                   *                   STA     PASS_CNT+1
54BD 1966                           *
54BD 1967 08C6'C3                   ******************  ELSE
54C0 1968                           *
54C0 1969 043F'2A                   *                   LHLD    INPUT_NUM+2
54C3 1970 0314'22                   *                   SHLD    PASS_CNT            ;MOVE 2 BYTES PASS COUNT
54C6 1971                           *
54C6 1972                           ******************  ENDIF
54C6 1973
54C6 1974 C9                                    RET
54C7 1975
54C7 1976
54C7 1977                       ;*************************************************************************
```

```
54C7 1978
54C7 1979
54C7 1980
54C7 1981
54C7 1982
54C7 1983
54C7 1984
54C7 1985
54C7 1986
54C7 1987
54C7 1988 FF 3E                                ;*************************************************************************
                                               ;
                                               ;  DIAG_SHORTEN
                                               ;  THIS SUBROUTINE HANDLES THE DIAGNOSE SHORTEN KEYWORD
                                               ;  THE SHORTEN COMMAND CHANGES THE ENDING TEST NUMBER TO
                                               ;  CAUSE THE SHORTEST LOOP FROM THE BEGINING OR THE STARTING
                                               ;  TEST NUMBER TO THE LOWEST FAILING TEST NUMBER
                                               ;  THE PASS COUNT IS ALSO SET TO INFINITE PASSES
                                               ;
                                               ;*************************************************************************

54C7 1988 FF 3E                DIAG_SHORTEN:   MVI     A,HEX_FF                ;SET TO INFINITE PASSES
54C9 1989 0314'32                              STA     PASS_CNT
54CC 1990 0315'32                              STA     PASS_CNT+1
54CF 1991
54CF 1992 0345'32 3C AF                        SET     SHORTEN                 ;SET SHORTEN MODE FLAG
54D4 1993
54D4 1994 C9                                   RET
54D5 1995
54D5 1996                                      ;*************************************************************************
54D5 1997                                      ;
54D5 1998                                      ;  SHOW_COMMAND
54D5 1999                                      ;  THIS SUBROUTINE HANDLES THE COMMAND SHOW
54D5 2000                                      ;  IT PRINTS THE STATE OF THE FLAGS HALT,LOOP,NER,BELL,SOMM AND THE
54D5 2001                                      ;  SOMM ADDRESS IF THE SOMM FLAG IS SET
54D5 2002                                      ;
54D5 2003                                      ;*************************************************************************
54D5 2004
54D5 2005
54D5 2006 03'46'21             SHOW_COMMAND:   LXI     H,SHOWFLAG_A
54D8 2007 124A'CD                              CALL    PRINT_STRING_R
54DB 2008
54DB 2009 01 3E                                MVI     A,1                     ;SET UP ROTATING ONE TO MASK
54DD 2010                                                                      ;FLAGS WITH
54DD 2011 03C5'32                              STA     TEMP_BYTE
54E0 2012
54E0 2013 06'AE'21                             LXI     H,PRT_FLAGS_A           ;SET UP POINTER TO ASCII
54E3 2014 03C6'22                              SHLD    TEMP_WORD               ;NAMES OF FLAGS
54E6 2015
54E6 2016 C5 46 00'55'21       ******************      DO      5               ;5 FLAGS TO PRINT IN BYTE
          77 05 3E             *
54EE 2017                      *
54EE 2018 47 03C5'3A           * ****************      IFMB    FLAGS,TEMP_BYTE ;IS FLAG SET FOR THIS BIT
          A0 005A'3A           * *
          08FC'CA              * *
54F9 2019                      * *
54F9 2020 11AD'CD              * *                     CALL    PRINT_FLG       ;PRINT FLAG NAME IF SET
54FC 2021                      * *
54FC 2022                      * ****************      ENDIF
54FC 2023                      *
54FC 2024 1575'CD              *                       CALL    INC_TEMPW_5     ;INC TEMP WORD TO NEXT FLAG
54FF 2025 03'C5'21             *                       LXI     H,TEMP_BYTE     ;MOVE MASK ONE BIT LEFT
5502 2026 7E                   *                       MOV     A,M
5503 2027 17                   *                       RAL
5504 2028 77                   *                       MOV     M,A
5505 2029                      *
5505 2030 35 00'55'21          ******************      ENDDO
          70 C1 08EE'C2
550E 2031
550E 2032 03DF'3A                              LDA     TRACE                   ;GET TRACE FLAG
5511 2033 B7                                   ORA     A                       ;SET CONDITION CODES
```

```
5512 2034 11AD'C4                                   CNZ     PRINT_FLG               ;PRINT OUT TRACE FLAG IF SET
5515 2035
5515 2036 1575'CD                                   CALL    INC_TEMPW_5
5518 2037
5518 2038 0F 039F'3A                ******************  IF      SOMM_FLG                ;PRINT OUT SOMM FLAG IF SET
          092B'D2                   *
551F 2039                           *
551F 2040 11AD'CD                   *                   CALL    PRINT_FLG
5522 2041                           *
5522 2042 039D'2A                   *                   LHLD    SOMM_ADDRESS            ;PRINT SOMM ADDRESS WITH FLAG
5525 2043 005B'22                   *                   SHLD    HEX_BUF
5528 2044 11BD'CD                   *                   CALL    PRINT_HEX_2
552B 2045                           *
552B 2046                           ******************  ENDIF
552B 2047
552B 2048 1575'CD                                   CALL    INC_TEMPW_5             ;POINT TO NEXT FLAG NAME
552E 2049
552E 2050 005A'3A                                   LDA     FLAGS                   ;GET FLAGS INDICATOR
5531 2051 47                                        MOV     B,A                     ;PUT BYTE IN B
5532 2052 80 3E                                     MVI     A,HEX_80                ;PUT AN 80 IN A
5534 2053 A0                                        ANA     B                       ;AND 80 WITH FLAGS INDICATOR
5535 2054 11AD'C4                                   CNZ     PRINT_FLG               ;PRINT FLAG NAME IF SET
5538 2055
5538 2056 C9                                        RET
5539 2057
5539 2058                           ;*****************************************************************************
5539 2059                           ;
5539 2060                           ;  LOADER
5539 2061                           ;  THIS ROUTINE LOADS A IMAGE INTO WCS OR THE CONSOLE BUT DOES
5539 2062                           ;  NOT START IT AND DOES NOT INITIALIZE IT
5539 2063                           ;
5539 2064                           ;*****************************************************************************
5539 2065
5539 2066 02F2'32 3C AF             LOADER:         SET     NOINIT
553E 2067 07EB'CD                                   CALL    DIAG_SECTION
5541 2068 02F2'32 AF                                CLEAR   NOINIT
5545 2069 C9                                        RET
5546 2070
5546 2071
5546 2072                           ;*****************************************************************************
5546 2073                           ;
5546 2074                           ;  CONT_COMMAND
5546 2075                           ;  THE CONTINUE COMMAND ALLOWS CONTINUATION OF TEST EXECUTION AFTER
5546 2076                           ;  TESTING HAS BEEN HALTED
5546 2077                           ;
5546 2078                           ;*****************************************************************************
5546 2079
5546 2080 04 06                     CONT_COMMAND:   MVI     B,<^X04>                ;ERROR CODE IF CONTINUE CLEAR
5548 2081
5548 2082 003A'3A                                   LDA     CONTINUE                ;GET CONTINUE FLAG
554B 2083 B7                                        ORA     A                       ;SET CONDITION CODES
554C 2084 0F7E'CA                                   JZ      ERROR_ROUTINE           ;NO CONTINUE AT THIS POINT IF
554F 2085                                                                           ; CONTINUE FLAG IS CLEAR
554F 2086                                                                           ; ELSE CONTINUE ON
554F 2087 0F 0209'3A                ******************  IFN     CONSOLE_TEST            ;DO WR RESTORE IF 8085
          0959'DA                   *
5556 2088 0F41'CD                   *                   CALL    RESTORE_WR              ; TESTS ARE NOT EXECUTING
5559 2089                           ******************  ENDIF
5559 2090
5559 2091 3D D3                                     OUT     SETLIT                  ;TURN RUN LITE BACK ON
```

```
555B 2092
555B 2093 0204'3A                       LDA     C_EXECUTE               ;RESTORE FLAGS
555E 2094 023B'32                       STA     EXECUTE
5561 2095
5561 2096 00'BE'21                      LXI     H,CONT_SAVE_CSR         ;RESTORE CSR
5564 2097 00'BB'11                      LXI     D,SAVED_CSR
5567 2098 118E'CD                       CALL    MOVER_3_R
556A 2099
556A 2100 00C8'2A                       LHLD    CONT_SAVE_UPC           ;RESTORE UPC
556D 2101 00C4'22                       SHLD    SAVED_UPC
5570 2102
5570 2103 0205'3A                       LDA     C_RUNNING               ;GET OLD RUNNING FLAG
5573 2104 B7                            ORA     A                       ;SET CONDITION CODES
5574 2105 13DD'C4                       CNZ     RESTART_CPU             ;RESTART IF SET
5577 2106
5577 2107 020A'32 3C AF                 SET     CONT_DONE
557C 2108
557C 2109 C9                            RET
557D 2110
557D 2111               ;*****************************************************************************
557D 2112               ;
557D 2113               ;  SET_COMMAND
557D 2114               ;  THE SET COMMAND IS USED TO SET THE STATE OF FLAGS USED TO CONTROL
557D 2115               ;  THE EXECUTION OF DIAGNOSTICS
557D 2116               ;
557D 2117               ;*****************************************************************************
557D 2118
557D 2119 05'BF'21      SET_COMMAND:    LXI     H,SET_TAB               ;POINTER TO PARSE TABLE
5580 2120 1FDB'CD                       CALL    PAR_TEMP_BUF            ;PARSE SET COMMANDS
5583 2121
5583 2122 C9                            RET
5584 2123
5584 2124
5584 2125
5584 2126               ;*****************************************************************************
5584 2127               ;
5584 2128               ;  THIS SUBROUTINE HANDLES THE SET HALT COMMAND
5584 2129               ;  SET HALT IS FOR THE HALT ON ERROR FUNCTION
5584 2130               ;  THIS SET COMMAND SETS A LOCAL FLAG AND WRITES THE FLAG
5584 2131               ;  STATUS WORD IN LS
5584 2132               ;
5584 2133               ;*****************************************************************************
5584 2134
5584 2135 0A66'CD       SET_HALT:       CALL    CLEAR_LOOP_COM          ;CLEAR LOOP COMMAND BIT
5587 2136 08 06                         MVI     B,HALT_S_DEF            ;BIT TO SET IN B
5589 2137 159E'C3                       JMP     SET_LS_STAT
558C 2138
558C 2139               ;*****************************************************************************
558C 2140               ;
558C 2141               ;  SET_LOOP
558C 2142               ;  THIS SUBROUTINE HANDLES THE SET LOOP COMMAND
558C 2143               ;  SET LOOP IS FOR LOOP ON ERROR FUNCTION
558C 2144               ;  THIS SET COMMAND SETS A LOCAL FLAG AND WRITES THE FLAG
558C 2145               ;  STATUS WORD IN LS
558C 2146               ;
558C 2147               ;*****************************************************************************
558C 2148
558C 2149 01 06         SET_LOOP:       MVI     B,LOOP_S_DEF            ;BIT TO SET IN B
558E 2150 159E'C3                       JMP     SET_LS_STAT
5591 2151
```

```
5591 2152
5591 2153
5591 2154                       ;*****************************************************************************
5591 2155                       ;
5591 2156                       ;  SET_NER
5591 2157                       ;  THIS SUBROUTINE HANDLES THE SET NER COMMAND
5591 2158                       ;  SET NER IS FOR "NO ERROR REPORT" FUNCTION
5591 2159                       ;  THIS SET COMMAND SETS A LOCAL FLAG AND WRITES THE FLAG
5591 2160                       ;  STATUS WORD IN LS
5591 2161                       ;
5591 2162                       ;*****************************************************************************
5591 2163
5591 2164 02 06                 SET_NER:        MVI     B,NER_S_DEF             ;BIT TO SET IN B
5593 2165 159E'C3                               JMP     SET_LS_STAT
5596 2166
5596 2167
5596 2168
5596 2169                       ;*****************************************************************************
5596 2170                       ;
5596 2171                       ;  SET_BELL
5596 2172                       ;  THIS SUBROUTINE HANDLES THE SET BELL COMMAND
5596 2173                       ;  SET BELL RINGS THE TERMINAL BELL ON ERROR
5596 2174                       ;  THIS SET COMMAND SETS A LOCAL FLAG AND WRITES THE FLAG
5596 2175                       ;  STATUS WORD IN LS
5596 2176                       ;
5596 2177                       ;*****************************************************************************
5596 2178
5596 2179 0A66'CD               SET_BELL:       CALL    CLEAR_LOOP_COM          ;CLEAR LOOP COMMAND BIT
5599 2180 04 06                                 MVI     B,BELL_S_DEF            ;BIT TO SET IN B
559B 2181 159E'C3                               JMP     SET_LS_STAT
559E 2182
559E 2183
559E 2184                       ;*****************************************************************************
559E 2185                       ;
559E 2186                       ;  SET_SER
559E 2187                       ;  THIS SUBROUTINE HANDLES THE 'SET SINGLE BIT ERROR REPORTS' COMMAND.
559E 2188                       ;  SET SER CAUSES MEMORY SINGLE BIT ERRORS TO BE REPORTED AS ERRORS
559E 2189                       ;  THIS SET COMMAND SETS A LOCAL FLAG AND WRITES THE FLAG
559E 2190                       ;  STATUS WORD IN LS
559E 2191                       ;
559E 2192                       ;*****************************************************************************
559E 2193
559E 2194 10 06                 SET_SER:        MVI     B,SER_S_DEF             ;BIT TO SET IN B
55A0 2195 159E'C3                               JMP     SET_LS_STAT             ;WRITE IN LS ERROR CONTROL
55A3 2196
55A3 2197
55A3 2198                       ;*****************************************************************************
55A3 2199                       ;
55A3 2200                       ;  SET_SP
55A3 2201                       ;  THIS SUBROUTINE HANDLES THE 'SET SPECIAL BIT' COMMAND.  SET SP CAUSES
55A3 2202                       ;  SIZING TESTS TO LOOP REGARDLESS OF THERE RESULTS.  IT IS USED TO LOOP
55A3 2203                       ;  ON A SIZING TEST THAT DOESN'T FIND A DEVICE THAT IS PRESENT OR VICE
55A3 2204                       ;  VERSA.  TESTS THAT USE IT WILL EXPLAIN IT'S OPERATION IN THE TEST
55A3 2205                       ;  DOCUMENTATION.
55A3 2206                       ;
55A3 2207                       ;*****************************************************************************
55A3 2208
55A3 2209 80 06                 SET_SP:         MVI     B,SPECIAL_S_DEF         ;BIT TO SET IN B
55A5 2210 159E'C3                               JMP     SET_LS_STAT             ;WRITE IN LS ERROR CONTROL
55A8 2211
```

```
55A8 2212
55A8 2213
55A8 2214
55A8 2215
55A8 2216
55A8 2217
55A8 2218
55A8 2219
55A8 2220
55A8 2221
55A8 2222
55A8 2223 039F'3A                ;**********************************************************************
55A8 2224                        ;
55A8 2225                        ;  SET_SOMM
55A8 2226                        ;  THIS ROUTINE SETS THE STOP ON MICRO MATCH FLAG AND WRITES BAD PARITY
55A8 2227                        ;  AT THE GIVEN ADDRESS.  WHEN A PARITY ERROR OCCURS THIS ADDRESS IS
55A8 2228                        ;  CHECKED TO SEE IF IT WAS PUT THERE INTENTIONALY.  IF A SET SOMM ADDRESS
55A8 2229 0F6B'CD                ;  IS GIVEN WHEN THERE IS ONE ALREADY IN EFFECT THE PARTIY IS CORRECTED
55B2 2230                        ;  AT THE OLD SOMM ADDRESS BEFORE THE NEW ONE IS WRITTEN
55B2 2231                        ;
55B2 2232                        ;**********************************************************************
55B2 2233 AF                     SET_SOMM:       LDA     SOMM_FLG                ;GET SOMM FLAG
55B3 2234 3C                                     ORA     A                       ;SET CONDITION CODES
55B4 2235 039F'32                                CNZ     CLEAR_SOMM              ;IF SOMM SET WHEN SET IS DONE
55B7 2236 030C'32                                                                ; THE OLD SOMM ADDRESS IS
55BA 2237 006E'32                                                                ; CLEARED FIRST
55BD 2238
55BD 2239 043F'2A                                CALL    INPUT_NUM_CHECK         ;CHECK FOR INPUT,REPORT ERR
55C0 2240 039D'22                                                                ; IF NO INPUT, ELSE LOAD
55C3 2241 0F52'C3                                                                ; INPUT_NUM WITH SOMM ADDRESS
55C6 2242
55C6 2243                                        XRA     A
55C6 2244                                        INR     A
55C6 2245                                        STA     SOMM_FLG                ;SET SOMM FLAG
55C6 2246                                        STA     PAR_ON                  ;SET PAR_ON FLAG
55C6 2247                                        STA     WCS_BAD_PARITY          ;SET WCS_BAD_PARITY FLAG
55C6 2248
55C6 2249                                        LHLD    INPUT_NUM+2
55C6 2250                                        SHLD    SOMM_ADDRESS            ;SAVE THE SOMM ADDRESS
55C6 2251                                        JMP     CHANGE_WCS              ;CHANGE PARITY TO BAD
55C6 2252
55C6 2253                        ;**********************************************************************
55C6 2254                        ;
55C6 2255                        ;  SET_STEP
55C6 2256                        ;  THIS ROUTINE SETS MICRO STEPPING MODE
55C6 2257 0FCF'CD                ;  MICRO_STEP  0 = NO MICRO STEP
55C9 2258                        ;              1 = SPACE BAR MODE
55C9 2259 0F 0231'3A             ;              3 = COUNT MODE
          09D5'D2                ;
55D0 2260 01 3E                  ;
55D2 2261                        ;
55D2 2262 09DD'C3                ;  MICRO STEPPING WILL BE DONE AS PART OF THE IDLE-ATTENTION
55D5 2263 043F'2A                ;  LOOPS USED IN THE PROGRAM
55D8 2264 03A3'22                ;
55DB 2265 03 3E                  ;**********************************************************************
55DD 2266
55DD 2267
55DD 2268 02C8'32                SET_STEP:       CALL    INPUT_NUM_IF
55E0 2269 0946'C3
55E3 2270
```

```
                                 IF      ERROR                   ;ERROR SET IF NO NUMBER
                                 MVI     A,1                     ;THEN SET TO SPACE BAR MODE
                                 ELSE
                                 LHLD    INPUT_NUM+2             ;SAVE NUMBER OF COUNTS
                                 SHLD    STEP_CNT
                                 MVI     A,3                     ;SET TO COUNT MODE
                                 ENDIF

                                 STA     MICRO_STEP              ;SAVE MODE IN MICRO_STEP
                                 JMP     CONT_COMMAND
```

```
55E3 2271
55E3 2272                       ;*****************************************************************************
55E3 2273                       ;
55E3 2274                       ;  SET_TRACE
55E3 2275                       ;  THIS SUBROUTINE SETS THE TRACE FLAG
55E3 2276                       ;  WHEN THE TRACE FLAG IS SET THE TEST NUMBERS WILL BE PRINTED AS THEY
55E3 2277                       ;  BEGIN
55E3 2278                       ;
55E3 2279                       ;*****************************************************************************
55E3 2280
55E3 2281 03DF'32 3C AF         SET_TRACE:      SET     TRACE                   ;SET TO TRACE MODE
55E8 2282
55E8 2283 C9                                    RET
55E9 2284
55E9 2285
55E9 2286                       ;*****************************************************************************
55E9 2287                       ;
55E9 2288                       ;  SET_PAR
55E9 2289                       ;  THIS ROUTINE SETS A PARITY ERROR IN WCS AT THE GIVEN ADDRESS
55E9 2290                       ;  IF NO ADDRESS IS GIVEN THEN ONLY A DISABLE HALT ON PARITY ERRORS
55E9 2291                       ;  WILL BE DONE
55E9 2292                       ;
55E9 2293                       ;*****************************************************************************
55E9 2294
55E9 2295 0FCF'CD               SET_PAR:        CALL    INPUT_NUM_IF            ;CHECK FOR PARITY ERROR ADDR
55EC 2296
55EC 2297 0F 0231'3A                    ********************
          09FB'DA               *
55F3 2298 006E'32 3C AF         *               SET     WCS_BAD_PARITY          ;SET FLAG
55F8 2299 0F52'CD               *               CALL    CHANGE_WCS              ;CHANGE PARITY TO BAD
55FB 2300                       ********************    ENDIF
55FB 2301
55FB 2302 030C'32 AF                            CLEAR   PAR_ON                  ;SET FOR NO HALT ON PAR ERR
55FF 2303
55FF 2304 C9                                    RET
5600 2305
5600 2306
5600 2307                       ;*****************************************************************************
5600 2308                       ;
5600 2309                       ;  SET_BREAK
5600 2310                       ;  THIS SUBROUTINE SETS A BRANCH IN RAM AT THE ADDRESS GIVEN
5600 2311                       ;  THE BRANCH GOES TO A ROUTINE THAT WILL PRINT OUT INFO AND THEN
5600 2312                       ;  RETURN TO THE PARSER SO EXAMINES CAN BE MADE.  THE LOCATION SELECTED
5600 2313                       ;  AND ITS CONTENTS ARE SAVED AND RESTORE IF A BREAK IS SET AT ANOTHER
5600 2314                       ;  ADDRESS, OR A CL BR IS ISSUED.
5600 2315                       ;
5600 2316                       ;  CAUTION: THIS USES 3 BYTES AT THE ADDRESS GIVEN, UNTIL CLEARED OR
5600 2317                       ;  CHANGED TO ANOTHER LOCATION.
5600 2318                       ;
5600 2319                       ;*****************************************************************************
5600 2320
5600 2321 0F6B'CD               SET_BREAK:      CALL    INPUT_NUM_CHECK         ;CHECK FOR INPUT,REPORT ERR
5603 2322                                                                       ; IF NO INPUT, ELSE LOAD
5603 2323                                                                       ; INPUT_NUM WITH BREAK ADDRESS
5603 2324
5603 2325 0203'3A                               LDA     BREAK_FLG               ;SEE IF BREAK WAS ISSUED BEFORE
5606 2326 B7                                    ORA     A                       ;SET CONDITION CODES
5607 2327 0A28'C4                               CNZ     CLEAR_BREAK             ;CLEAR OLD BREAK ADDRESS
560A 2328
560A 2329 0203'32 3C AF                         SET     BREAK_FLG               ;SET FLAG
```

```
560F 2330 043F'2A                       LHLD    INPUT_NUM+2             ;ELSE GET ADDRESS
5612 2331 7C                            MOV     A,H
5613 2332 65                            MOV     H,L
5614 2333 6F                            MOV     L,A                     ;SWITCH H&L
5615 2334 01FE'22                       SHLD    BREAK_ADDRESS           ;SAVE BREAK ADDRESS
5618 2335 02'F7'11                      LXI     D,OLD_INSTRUCTION       ;SAVE OLD CONTENTS OF BREAK
561B 2336                                                               ; LOCATION
561B 2337 118E'CD                       CALL    MOVER_3_R               ;MOVE IT
561E 2338
561E 2339 01FE'2A                       LHLD    BREAK_ADDRESS           ;GET BACK POINTER
5621 2340 EB                            XCHG                            ;MAKE BREAK ADDRESS DEST
5622 2341 02'00'21                      LXI     H,BREAK_BRANCH
5625 2342 118E'C3                       JMP     MOVER_3_R
5628 2343
5628 2344
5628 2345                       ;*******************************************************************************
5628 2346                       ;
5628 2347                       ;  CLEAR_BREAK
5628 2348                       ;  THIS SUBROUTINE CLEARS THE BRANCH IN RAM DEPOSITED ON THE SET_BREAK.
5628 2349                       ;  IT REPLACES THE OLD INSTRUCTION AT THE LAST BREAK LOCATION.
5628 2350                       ;
5628 2351                       ;*******************************************************************************
5628 2352
5628 2353 02'F7'11              CLEAR_BREAK:    LXI     D,OLD_INSTRUCTION       ;SOURCE IS ORIGINAL DATA
562B 2354 01FE'2A                               LHLD    BREAK_ADDRESS           ;DEST IS BREAK LOCATION
562E 2355 EB                                    XCHG                            ;SWITCH HL&DE
562F 2356 118E'CD                               CALL    MOVER_3_R               ;REPLACE OLD INSTRUCTION
5632 2357 0203'32 AF                            CLEAR   BREAK_FLG               ;CLEAR FLAG
5636 2358 C9                                    RET                             ;DONE
5637 2359
5637 2360                       ;*******************************************************************************
5637 2361                       ;
5637 2362                       ;  SEL_FIFO_A
5637 2363                       ;  THIS SUBROUTINE EXECUTES A SUBROUTINE IN WCS WHICH SELECTS FIFO A ON
5637 2364                       ;  THE IDC MODULE IN PREPERATION FOR AN EXAMINE.
5637 2365                       ;
5637 2366                       ;*******************************************************************************
5637 2367
5637 2368 00 63 21              SEL_FIFO_A:     LXI     H,WCS_SEL_FIFOA         ;ADDRESS OF WCS SUBROUTINE
563A 2369 0A49'C3                               JMP     EXEC_SUB                ;EXECUTE SUBROUTINE
563D 2370
563D 2371
563D 2372                       ;*******************************************************************************
563D 2373                       ;
563D 2374                       ;  SEL_FIFO_B
563D 2375                       ;  THIS SUBROUTINE EXECUTES A SUBROUTINE IN WCS WHICH SELECTS FIFO B ON
563D 2376                       ;  THE IDC MODULE IN PREPERATION FOR AN EXAMINE.
563D 2377                       ;
563D 2378                       ;*******************************************************************************
563D 2379
563D 2380 00 64 21              SEL_FIFO_B:     LXI     H,WCS_SEL_FIFOB         ;ADDRESS OF WCS SUBROUTINE
5640 2381 0A49'C3                               JMP     EXEC_SUB                ;EXECUTE SUBROUTINE
5643 2382
5643 2383
5643 2384                       ;*******************************************************************************
5643 2385                       ;
5643 2386                       ;  CLEAR_FIFO
5643 2387                       ;  THIS SUBROUTINE EXECUTES A SUBROUTINE IN WCS WHICH SELECTS CLEARS THE
5643 2388                       ;  FIFO ADDRESS ON THE IDC MODULE IN PREPERATION FOR AN EXAMINE.
5643 2389                       ;
```

```
5643 2390
5643 2391
5643 2392 00 62 21                  CLEAR_FIFO:     LXI     H,WCS_CL_FIFO           ;ADDRESS OF WCS SUBROUTINE
5646 2393 0A49'C3                                   JMP     EXEC_SUB                ;EXECUTE SUBROUTINE
5649 2394
5649 2395
5649 2396                           ;*****************************************************************************
5649 2397                           ;
5649 2398                           ;  EXEC_SUB
5649 2399                           ;
5649 2400                           ;  THIS ROUTINE EXECUTES A SUBROUTINE IN WCS WHICH PERFORMS A SPECIFIC
5649 2401                           ;  FUNCTION.
5649 2402                           ;
5649 2403                           ;  INPUT CONDITIONS
5649 2404                           ;       H+L  = ADDRESS OF WCS SUBROUTINE
5649 2405                           ;
5649 2406                           ;*****************************************************************************
5649 2407
5649 2408 11 06                     EXEC_SUB:       MVI     B,<^X11>                ;ERROR CODE IF WCS NOT LOADED
564B 2409 0F5E'CD                                   CALL    CHECK_WCS_LOADED        ;CHECK FOR WCS LOADED SET
564E 2410                                                                           ; RETURN IF OK
564E 2411
564E 2412 7C                                        MOV     A,H                     ;GET SUBROUTINE ADDR
564F 2413 65                                        MOV     H,L
5650 2414 6F                                        MOV     L,A
5651 2415 03A1'22                                   SHLD    STARTING_UPC
5654 2416
5654 2417 13CE'CD                                   CALL    START_CPU               ;START THE SUBROUTINE
5657 2418 1A7A'C3                                   JMP     SUB_ATTENTION           ;WAIT FOR EXIT
565A 2419
565A 2420
565A 2421                           ;*****************************************************************************
565A 2422                           ;
565A 2423                           ;  CLEAR_COMMAND
565A 2424                           ;  THE CLEAR COMMAND IS USED TO CLEAR THE FLAGS USED TO CONTROL
565A 2425                           ;  THE EXECUTION OF DIAGNOSTICS
565A 2426                           ;
565A 2427                           ;*****************************************************************************
565A 2428
565A 2429 05'82'21                  CLEAR_COMMAND:  LXI     H,CLEAR_TAB             ;POINTER TO PARSE TABLE
565D 2430 1FDB'CD                                   CALL    PAR_TEMP_BUF            ;PARSE THE COMMAND
5660 2431
5660 2432 C9                                        RET
5661 2433
5661 2434
5661 2435
5661 2436                           ;*****************************************************************************
5661 2437                           ;
5661 2438                           ;  CLEAR_HALT
5661 2439                           ;  THIS SUBROUTINE HANDLES THE CLEAR HALT COMMAND
5661 2440                           ;  CLEAR HALT IS TO DISABLE HALT ON ERROR
5661 2441                           ;  THIS CLEAR COMMAND CLEARS A LOCAL FLAG AND WRITES THE FLAG
5661 2442                           ;  STATUS WORD IN LS
5661 2443                           ;
5661 2444                           ;*****************************************************************************
5661 2445
5661 2446 F7 06                     CLEAR_HALT:     MVI     B,HALT_C_DEF            ;BIT TO CLEAR IN B
5663 2447 15A8'C3                                   JMP     CLEAR_LS_STAT
5666 2448
5666 2449
```

```
5666 2450
5666 2451
5666 2452
5666 2453
5666 2454
5666 2455
5666 2456
5666 2457
5666 2458
5666 2459
5666 2460 BF 06                  CLEAR_LOOP_COM: MVI     B,LOOP_COM_C_DEF        ;BIT TO CLEAR LOOP COMMAND
5668 2461 15A8'C3                                 JMP     CLEAR_LS_STAT
566B 2462
566B 2463
566B 2464
566B 2465
566B 2466
566B 2467
566B 2468
566B 2469
566B 2470
566B 2471
566B 2472
566B 2473
566B 2474
566B 2475
566B 2476 0057'32 AF              CLEAR_LOOP:     CLEAR   ERROR_CON               ;CLEAR 8085 ERROR LOOP FLAG
566F 2477 BE 06                                   MVI     B,LOOP_C_DEF            ;BITS TO CLEAR IN B (LOOP
5671 2478                                                                         ; AND LOOP COMMAND BITS)
5671 2479 15A8'C3                                 JMP     CLEAR_LS_STAT
5674 2480
5674 2481
5674 2482
5674 2483
5674 2484
5674 2485
5674 2486
5674 2487
5674 2488
5674 2489
5674 2490
5674 2491
5674 2492
5674 2493
5674 2494
5674 2495 BD 06                  CLEAR_NER:      MVI     B,NER_C_DEF             ;BITS TO CLEAR IN B (NER
5676 2496                                                                         ; AND LOOP COMMAND BITS)
5676 2497 15A8'C3                                 JMP     CLEAR_LS_STAT
5679 2498
5679 2499
5679 2500
5679 2501
5679 2502
5679 2503
5679 2504
5679 2505
5679 2506
5679 2507
5679 2508
5679 2509
```

```
;*****************************************************************************
;
;  CLEAR_LOOP_COM
;  THIS SUBROUTINE HANDLES THE CLEAR LOOP COMMAND COMMAND.  THIS FLAG
;  IS SET WHEN THE COMMAND "LOOP" IS ISSUED AFTER A HALT ON ERROR.
;  THIS FLAG IS TRANSPARENT TO THE USER AND IS CLEARED ONLY WHEN ONE OF
;  THE FLAGS SET UP BY THE LOOP COMMAND IS CHANGED.
;
;*****************************************************************************

;*****************************************************************************
;
;  CLEAR_LOOP
;  THIS SUBROUTINE HANDLES THE CLEAR LOOP COMMAND
;  CLEAR LOOP IS TO DISABLE LOOP ON ERROR
;  THIS CLEAR COMMAND CLEARS A LOCAL FLAG AND WRITES THE FLAG
;  STATUS WORD IN LS
;
;*****************************************************************************

;*****************************************************************************
;
;  CLEAR_NER
;  THIS SUBROUTINE HANDLES THE CLEAR NER COMMAND
;  CLEAR NER INHIBITS  "NO ERROR REPORT" THUS ERRORS ARE REPORTED
;  THIS CLEAR COMMAND CLEARS A LOCAL FLAG AND WRITES THE FLAG
;  STATUS WORD IN LS
;
;*****************************************************************************

;*****************************************************************************
;
;  CLEAR_BELL
;  THIS SUBROUTINE HANDLES THE CLEAR BELL COMMAND
;  CLEAR BELL INHIBITS RINGING THE TERMINAL BELL ON ERROR
;  THIS CLEAR COMMAND CLEARS A LOCAL FLAG AND WRITES THE FLAG
;  STATUS WORD IN LS
;
```

```
5679 2510
5679 2511
5679 2512
5679 2513 FB 06                 CLEAR_BELL:     MVI     B,BELL_C_DEF            ;LOAD B REG WITH BIT TO CLR
567B 2514 15A8'C3                               JMP     CLEAR_LS_STAT
567E 2515
567E 2516
567E 2517
567E 2518                       ;*****************************************************************************
567E 2519                       ;
567E 2520                       ;  CLEAR_SER
567E 2521                       ;  THIS SUBROUTINE HANDLES THE CLEAR SINGLE BIT ERROR REPORTS COMMAND
567E 2522                       ;  CLEAR SER CAUSES MEMORY SINGLE BIT ERRORS NOT TO BE REPORTED AS ERRORS
567E 2523                       ;  THIS CLEAR COMMAND CLEARS A LOCAL FLAG AND WRITES THE FLAG
567E 2524                       ;  STATUS WORD IN LS
567E 2525                       ;
567E 2526                       ;*****************************************************************************
567E 2527
567E 2528 EF 06                 CLEAR_SER:      MVI     B,SER_C_DEF             ;LOAD B REG WITH BIT TO CLR
5680 2529 15A8'C3                               JMP     CLEAR_LS_STAT
5683 2530
5683 2531
5683 2532                       ;*****************************************************************************
5683 2533                       ;
5683 2534                       ;  CLEAR_SP
5683 2535                       ;  THIS SUBROUTINE CLEARS THE SPECIAL BIT.  SEE SET SP FOR AN EXPLANATION
5683 2536                       ;  OF THE SPECIAL BIT.
5683 2537                       ;
5683 2538                       ;*****************************************************************************
5683 2539
5683 2540 7F 06                 CLEAR_SP:       MVI     B,SPECIAL_C_DEF         ;BIT TO CLEAR IN B
5685 2541 15A8'C3                               JMP     CLEAR_LS_STAT           ;WRITE IN LS ERROR CONTROL
5688 2542
5688 2543                       ;*****************************************************************************
5688 2544                       ;
5688 2545                       ;  CLEAR_SOMM
5688 2546                       ;  THIS ROUTINE CLEARS THE STOP ON MICRO MATCH FLAG AND WRITES GOOD
5688 2547                       ;  PARITY AT THE MATCH ADDRESS.
5688 2548                       ;
5688 2549                       ;*****************************************************************************
5688 2550
5688 2551
5688 2552 039F'32 AF            CLEAR_SOMM:     CLEAR   SOMM_FLG
568C 2553
568C 2554 039D'2A                               LHLD    SOMM_ADDRESS            ;GET SOMM ADDRESS TO CLEAR
568F 2555 043F'22                               SHLD    INPUT_NUM+2
5692 2556
5692 2557 0F52'C3                               JMP     CHANGE_WCS              ;SET PARITY GOOD ON ADDRESS
5695 2558
5695 2559
5695 2560                       ;*****************************************************************************
5695 2561                       ;
5695 2562                       ;  CLEAR_TRACE
5695 2563                       ;  THIS SUBROUTINE CLEARS THE TRACE FLAG
5695 2564                       ;  WHEN THE TRACE FLAG IS CLEARED THE TEST NUMBERS WILL NOT BE PRINTED
5695 2565                       ;  AS THEY BEGIN
5695 2566                       ;
5695 2567                       ;*****************************************************************************
5695 2568
5695 2569 03DF'32 AF            CLEAR_TRACE:    CLEAR   TRACE                   ;CLEAR TRACE FLAG
```

```
5699 2570
5699 2571 C9                                    RET
569A 2572
569A 2573
569A 2574
569A 2575
569A 2576
569A 2577
569A 2578
569A 2579
569A 2580 0A79'CD                               CLEAR_DEFAULT:  CALL    CLEAR_BELL
569D 2581 0A74'CD                                               CALL    CLEAR_NER
56A0 2582 0984'CD                                               CALL    SET_HALT
56A3 2583 0A6B'CD                                               CALL    CLEAR_LOOP
56A6 2584 0A95'CD                                               CALL    CLEAR_TRACE
56A9 2585 0A7E'CD                                               CALL    CLEAR_SER
56AC 2586 0A83'C3                                               JMP     CLEAR_SP
56AF 2587
56AF 2588
56AF 2589
56AF 2590
56AF 2591
56AF 2592
56AF 2593
56AF 2594
56AF 2595
56AF 2596
56AF 2597
56AF 2598 0FCF'CD                               CLEAR_PAR:      CALL    INPUT_NUM_IF            ;GET CLEAR ADDRESS
56B2 2599
56B2 2600 0AB5'C3                                               JMP     INT_CLEAR_PAR           ;CALL INTERNAL SUBROUTINE
56B5 2601
56B5 2602
56B5 2603
56B5 2604
56B5 2605
56B5 2606
56B5 2607
56B5 2608
56B5 2609
56B5 2610
56B5 2611
56B5 2612
56B5 2613
56B5 2614 0231'3A                               INT_CLEAR_PAR:  LDA     ERROR                   ;GET ERROR FLAG
56B8 2615 B7                                                    ORA     A                       ;SET CONDITION CODES
56B9 2616 0F52'CC                                               CZ      CHANGE_WCS              ;CHANGE PARITY TO GOOD IF
56BC 2617                                                                                       ; ADDRESS GIVEN
56BC 2618
56BC 2619 030C'32 3C AF                                         SET     PAR_ON                  ;SET FOR HALT ON PAR ERR
56C1 2620
56C1 2621 C9                                                    RET
56C2 2622
56C2 2623
56C2 2624
56C2 2625
56C2 2626
56C2 2627
56C2 2628
56C2 2629
```

```
;*****************************************************************************
;
;  CLEAR_DEFAULT
;  THIS ROUTINE SETS THE FLAGS TO THEIR DEFAULT VALUES
;
;*****************************************************************************

;*****************************************************************************
;
;  CLEAR_PAR
;  THIS ROUTINE CLEARS A PARITY ERROR IN WCS AT THE GIVEN ADDRESS
;  IF NO ADDRESS IS GIVEN THEN ONLY AN ENABLE HALT PARITY ERRORS
;  WILL BE DONE
;
;*****************************************************************************

;*****************************************************************************
;
;  INT_CLEAR_PAR
;  THIS SUBROUTINE IS CALLED FROM BOTH THE COMMAND CLEAR_PARITY AND FROM
;  THE INTERNAL CLEAR PARITY REQUEST
;
;  INPUT_CONDITION:
;       ERROR IS SET IF NO ADDRESS IS FOUND
;
;*****************************************************************************

;*****************************************************************************
;
;  LOOP_COMMAND
;  THIS SUBROUTINE HANDLES THE LOOP COMMAND
;  THE LOOP COMMAND IS USED TO MAKE A TEST LOOP AFTER IT HAS HALTED ON
;  ERROR.  WHEN LOOP IS USED THERE WILL BE NO ERROR PRINTOUTS
```

```
56C2 2630
56C2 2631
56C2 2632
56C2 2633
56C2 2634 0A79'CD                ;  THIS COMMAND ALSO WRITES THE LS STATUS WORD
56C2 2635                        ;
56C2 2636                        ;*********************************************************************
56C2 2637 0991'CD                LOOP_COMMAND:   CALL    CLEAR_BELL
56C5 2638 0A61'CD                                CALL    CLEAR_HALT
56C8 2639 098C'CD                                CALL    SET_NER
56CB 2640 40 06                                  CALL    SET_LOOP
56CE 2641 159E'CD                                MVI     B,LOOP_COM_S_DEF        ;BIT TO SET FOR LOOP COMMAND
56D0 2642                                        CALL    SET_LS_STAT
56D3 2643 0946'C3                                JMP     CONT_COMMAND
56D6 2644
56D6 2645                        ;*********************************************************************
56D6 2646                        ;
56D6 2647                        ;  REPEAT_COMMAND
56D6 2648                        ;
56D6 2649                        ;*********************************************************************
56D6 2650 0333'32 3C AF          REPEAT_COMMAND: SET     REPEAT                  ;SET REPEAT FLAG
56DB 2651 04'F5'21                               LXI     H,MAIN_TAB              ;POINTER TO MAIN TABLE
56DE 2652 1FDB'C3                                JMP     PAR_TEMP_BUF
56E1 2653
56E1 2654
56E1 2655                        ;*********************************************************************
56E1 2656                        ;
56E1 2657                        ;  EXAM_COMMAND
56E1 2658                        ;
56E1 2659                        ;*********************************************************************
56E1 2660
56E1 2661 06'06'21               EXAM_COMMAND:   LXI     H,EXAM_TAB              ;POINTER TO PARSE TABLE
56E4 2662 1FDB'CD                                CALL    PAR_TEMP_BUF
56E7 2663
56E7 2664 0333'3A                                LDA     REPEAT                  ;GET REPEAT FLAG
56EA 2665 B7                                     ORA     A                       ;SET CONDITION CODES
56EB 2666 125A'C4                                CNZ     CRLF_R                  ;DO <CR><LF> IF REPEAT
56EE 2667 C9                                     RET
56EF 2668
56EF 2669
56EF 2670
56EF 2671                        ;*********************************************************************
56EF 2672                        ;
56EF 2673                        ;  EXAM_RAM
56EF 2674                        ;  THIS ROUTINE EXAMINES THE RAM OR ROM OF THE 8085 CONSOLE AT THE
56EF 2675                        ;  ADDRESS GIVEN
56EF 2676                        ;
56EF 2677                        ;*********************************************************************
56EF 2678
56EF 2679 0F6B'CD                EXAM_RAM:       CALL    INPUT_NUM_CHECK         ;CHECK FOR INPUT,REPORT ERR
56F2 2680                                                                        ;  IF NO INPUT, ELSE LOAD
56F2 2681                                                                        ;  INPUT_NUM WITH RAM ADDR
56F2 2682
56F2 2683 043F'2A                                LHLD    INPUT_NUM+2             ;ELSE GET ADDRESS
56F5 2684 005B'22                                SHLD    HEX_BUF                 ;MOVE ADD TO PRINT AREA
56F8 2685
56F8 2686 03'2E'21                               LXI     H,RAM_A
56FB 2687 124A'CD                                CALL    PRINT_STRING_R
56FE 2688 11BD'CD                                CALL    PRINT_HEX_2             ;PRINT ADDRESS
5701 2689
```

```
5701 2690 043F'3A                       LDA     INPUT_NUM+2             ;PUT ADDRESS IN H+L TO READ RAM
5704 2691 67                            MOV     H,A
5705 2692 0440'3A                       LDA     INPUT_NUM+3
5708 2693 6F                            MOV     L,A
5709 2694
5709 2695 00'5E'11                      LXI     D,HEX_BUF+3             ;MOVE 4 BYTES TO BE PRINTED
570C 2696 04 0E                         MVI     C,4                     ;COUNTER IN C
570E 2697
570E 2698 7E                    1$:     MOV     A,M                     ;GET BYTE
570F 2699 12                            STAX    D                       ;STORE AT HEX_BUF
5710 2700 23                            INX     H                       ;POINT TO NEXT BYTE
5711 2701 1B                            DCX     D                       ;BACKUP HEX BUF
5712 2702 0D                            DCR     C                       ;DECREMENT COUNTER
5713 2703 0B0E'C2                       JNZ     1$                      ;REPEAT UNTIL 4 BYTES LOADED
5716 2704
5716 2705 11B8'C3                       JMP     PRINT_HEX_4             ;PRINT 4 BYTES AT RAM ADDRESS
5719 2706
5719 2707
5719 2708               ;*******************************************************************************
5719 2709               ;
5719 2710               ;  EXAM_CSR
5719 2711               ;  THIS ROUTINE EXAMINES THE CSR
5719 2712               ;
5719 2713               ;*******************************************************************************
5719 2714
5719 2715 1116'CD       EXAM_CSR:       CALL    LOAD_CSR_R              ;GET THE CSR
571C 2716
571C 2717 02'0D'21                      LXI     H,CSR_A                 ;PRINT "CSR "
571F 2718 124A'CD                       CALL    PRINT_STRING_R
5722 2719
5722 2720 0FC1'CD                       CALL    PRINT_CSR_VAL
5725 2721
5725 2722 0064'32 AF                    CLEAR   PARITY_ON               ;DONT CALC PARITY ON REWRITE
5729 2723
5729 2724 1116'C3                       JMP     LOAD_CSR_R              ;RETURN THE CSR
572C 2725
572C 2726
572C 2727               ;*******************************************************************************
572C 2728               ;
572C 2729               ;  EXAM_UPC
572C 2730               ;  THIS ROUTINE EXAMINES THE UPC
572C 2731               ;
572C 2732               ;*******************************************************************************
572C 2733
572C 2734 1148'CD       EXAM_UPC:       CALL    LOAD_UPC_R              ;GET THE UPC
572F 2735 00C2'2A                       LHLD    UPC_VALUE
5732 2736 005B'22                       SHLD    HEX_BUF                 ;SAVE THE VALUE
5735 2737
5735 2738 0F 02C8'3A    ******************      IF      MICRO_STEP              ;SEE IF MICRO_STEP MODE
          0B42'D2       *
573C 2739 00'5B'21      *                       LXI     H,HEX_BUF               ;POINT TO STORED UPC
573F 2740 158C'CD       *                       CALL    DECR_WORD_R             ;BACK UP UPC TO REFLECT
5742 2741               *                                                       ; ACTUAL INSTRUCTION IN
5742 2742               *                                                       ; PRINTOUT
5742 2743               ******************      ENDIF
5742 2744
5742 2745 1148'CD                       CALL    LOAD_UPC_R              ;RETURN THE UPC
5745 2746
5745 2747 03'EC'21      PRINT_UPC:      LXI     H,UPC_A                 ;PRINT "UPC "
5748 2748 124A'CD                       CALL    PRINT_STRING_R
```

```
574B 2749 11BD'C3                       JMP     PRINT_HEX_2
574E 2750
574E 2751
574E 2752
574E 2753
574E 2754
574E 2755
574E 2756
574E 2757
574E 2758
574E 2759 0F6B'CD       EXAM_WRK:       CALL    INPUT_NUM_CHECK         ;CHECK FOR INPUT,REPORT ERR
5751 2760                                                               ; IF NO INPUT, ELSE LOAD
5751 2761                                                               ; INPUT_NUM WITH WR NUMBER
5751 2762
5751 2763 04'2E'21                      LXI     H,S_ED_WRK              ;RESET THE INSTRUCTION TO
5754 2764 04'2B'11                      LXI     D,X_ED_WRK+1            ;NO MODIFIED BITS
5757 2765 118E'CD                       CALL    MOVER_3_R
575A 2766
575A 2767 0440'3A                       LDA     INPUT_NUM+3             ;MAKE MOVE WR[X] TO WR [0]
575D 2768 01B4'32                       STA     A_ADDRS
5760 2769 04'2B'21                      LXI     H,X_ED_WRK+1
5763 2770 1478'CD                       CALL    MAKE_A
5766 2771
5766 2772 04'2A'21                      LXI     H,X_ED_WRK
5769 2773 15C2'CD                       CALL    PERFORM_CSR_R           ;MOV REG TO READ REG
576C 2774 1481'CD                       CALL    READ_DATA_32_R          ;READ 32 BITS OF DATA
576F 2775
576F 2776 119C'CD                       CALL    MOV_DATA0_HEXBUF        ;MOVE DATA TO HEX PRINT AREA
5772 2777
5772 2778 04'0A'21                      LXI     H,WRK_A
5775 2779 124A'CD                       CALL    PRINT_STRING_R
5778 2780 11B8'CD                       CALL    PRINT_HEX_4             ;PRINT THE DATA
577B 2781
577B 2782 C9                            RET
577C 2783
577C 2784
577C 2785
577C 2786
577C 2787
577C 2788
577C 2789
577C 2790
577C 2791
577C 2792
577C 2793 0F6B'CD       EXAM_LS:        CALL    INPUT_NUM_CHECK         ;CHECK FOR INPUT,REPORT ERR
577F 2794                                                               ; IF NO INPUT, ELSE LOAD
577F 2795                                                               ; INPUT_NUM WITH LS ADDR
577F 2796
577F 2797 0440'3A                       LDA     INPUT_NUM+3             ;ELSE GET ADDRESS
5782 2798 1435'CD                       CALL    READ_LS_R               ;READ THE LS LOCATION
5785 2799
5785 2800 119C'CD                       CALL    MOV_DATA0_HEXBUF        ;MOVE DATA TO HEX PRINT AREA
5788 2801
5788 2802 02'94'21                      LXI     H,LS_A
578B 2803 124A'CD                       CALL    PRINT_STRING_R          ;PRINT STRING
578E 2804 11B8'CD                       CALL    PRINT_HEX_4             ;PRINT THE DATA
5791 2805
5791 2806 C9                            RET
5792 2807
5792 2808
```

```
5792 2809
5792 2810
5792 2811
5792 2812
5792 2813
5792 2814
5792 2815
5792 2816 00 51 21          ;*****************************************************************************
5795 2817 02'9A'11          ;
5798 2818 0C12'C3           ;  EXAM_MCT
579B 2819                   ;  THIS ROUTINE EXAMINES MEMORY CONTROL + STATUS WORDS
579B 2820                   ;
579B 2821                   ;*****************************************************************************
579B 2822
579B 2823
579B 2824
579B 2825
579B 2826
579B 2827
579B 2828
579B 2829 00 59 21          ;*****************************************************************************
579E 2830 03'E7'11          ;
57A1 2831 0C12'C3           ;  EXAM_UBS
57A4 2832                   ;  THIS ROUTINE EXAMINES UNIBUS TRANSLATION BUFFER
57A4 2833                   ;
57A4 2834                   ;*****************************************************************************
57A4 2835
57A4 2836
57A4 2837
57A4 2838
57A4 2839
57A4 2840
57A4 2841 00 53 21          ;*****************************************************************************
57A7 2842 03'B6'11          ;
57AA 2843 0C12'C3           ;  EXAM_TB
57AD 2844                   ;  THIS ROUTINE EXAMINES TRANSLATION BUFFER WORDS
57AD 2845                   ;
57AD 2846                   ;*****************************************************************************
57AD 2847
57AD 2848
57AD 2849
57AD 2850
57AD 2851
57AD 2852
57AD 2853 00 55 21          ;*****************************************************************************
57B0 2854 02'CA'11          ;
57B3 2855 0C12'C3           ;  EXAM_MM
57B6 2856                   ;  THIS ROUTINE EXAMINES MAIN MEMORY
57B6 2857                   ;
57B6 2858                   ;*****************************************************************************
57B6 2859
57B6 2860
57B6 2861
57B6 2862
57B6 2863
57B6 2864
57B6 2865 7C 3E             ;*****************************************************************************
57B8 2866 1435'CD           ;
57BB 2867                   ;  EXAM_OS
57BB 2868 00AC'3A           ;  THIS ROUTINE EXAMINES OS CPU REG
                            ;
                            ;*****************************************************************************

EXAM_MCT:   LXI   H,WCS_EX_MCT
            LXI   D,MCT_A
            JMP   EXAM_SUB

EXAM_UBS:   LXI   H,WCS_EX_UBS
            LXI   D,UBS_A
            JMP   EXAM_SUB

EXAM_TB:    LXI   H,WCS_EX_TB
            LXI   D,TB_A
            JMP   EXAM_SUB

EXAM_MM:    LXI   H,WCS_EX_MM
            LXI   D,MM_A
            JMP   EXAM_SUB

EXAM_OS:    MVI   A,OS_ADD
            CALL  READ_LS_R             ;OS IS AT AN LS ADDRESS
            LDA   DATA3                 ;GET DATA TO PRINT
```

```
57BE 2869 005B'32                       STA     HEX_BUF
57C1 2870
57C1 2871 03'07'21                      LXI     H,OS_A
57C4 2872 124A'CD                       CALL    PRINT_STRING_R
57C7 2873 11C2'C3                       JMP     PRINT_HEX_1_R           ;PRINT THE DATA
57CA 2874
57CA 2875
57CA 2876               ;*****************************************************************************
57CA 2877               ;
57CA 2878               ;  EXAM_WCS
57CA 2879               ;  THIS ROUTINE EXAMINES  WCS LOCATIONS
57CA 2880               ;
57CA 2881               ;*****************************************************************************
57CA 2882
57CA 2883 0F6B'CD       EXAM_WCS:       CALL    INPUT_NUM_CHECK         ;CHECK FOR INPUT,REPORT ERR
57CD 2884                                                               ; IF NO INPUT, ELSE LOAD
57CD 2885                                                               ; INPUT_NUM WITH WCS ADDR
57CD 2886
57CD 2887 043F'2A                       LHLD    INPUT_NUM+2             ;ELSE GET ADDRESS
57D0 2888 006C'22                       SHLD    WCS_ADDRESS
57D3 2889 109D'CD                       CALL    READ_WCS_R              ;READ THE DATA
57D6 2890
57D6 2891 04'02'21                      LXI     H,WCS_A
57D9 2892 124A'CD                       CALL    PRINT_STRING_R
57DC 2893
57DC 2894 0FC1'CD                       CALL    PRINT_CSR_VAL           ;PRINT THE DATA
57DF 2895
57DF 2896 C9                            RET
57E0 2897
57E0 2898
57E0 2899
57E0 2900               ;*****************************************************************************
57E0 2901               ;
57E0 2902               ;  EXAM_ICSR
57E0 2903               ;  THIS ROUTINE EXAMINES THE IDC CSR
57E0 2904               ;
57E0 2905               ;*****************************************************************************
57E0 2906
57E0 2907 00 5A 21      EXAM_ICSR:      LXI     H,WCS_EX_ICSR
57E3 2908 02'84'11                      LXI     D,ICSR_A
57E6 2909 0C0D'C3                       JMP     EXAM_SOB_NOADD
57E9 2910
57E9 2911
57E9 2912
57E9 2913               ;*****************************************************************************
57E9 2914               ;
57E9 2915               ;  EXAM_IDAR
57E9 2916               ;  THIS ROUTINE EXAMINES THE IDC DISK ADDRESS REGISTER
57E9 2917               ;
57E9 2918               ;*****************************************************************************
57E9 2919
57E9 2920 00 5B 21      EXAM_IDAR:      LXI     H,WCS_EX_IDAR
57EC 2921 02'89'11                      LXI     D,IDAR_A
57EF 2922 0C0D'C3                       JMP     EXAM_SOB_NOADD
57F2 2923
57F2 2924
57F2 2925
57F2 2926               ;*****************************************************************************
57F2 2927               ;
57F2 2928               ;  EXAM_DBUF
```

```
57F2 2929
57F2 2930                ;  THIS ROUTINE EXAMINES THE IDC DATA BUFFER
57F2 2931                ;
57F2 2932                ;*****************************************************************************
57F2 2933 00 5C 21       EXAM_DBUF:      LXI     H,WCS_EX_DBUF
57F5 2934 02'14'11               LXI     D,DBUF_A
57F8 2935 0C0D'C3                JMP     EXAM_SUB_NOADD
57FB 2936
57FB 2937
57FB 2938
57FB 2939                ;*****************************************************************************
57FB 2940                ;
57FB 2941                ;  EXAM_PATT
57FB 2942                ;  THIS ROUTINE EXAMINES THE IDC ECC PATTERN REGISTER
57FB 2943                ;
57FB 2944                ;*****************************************************************************
57FB 2945
57FB 2946 00 5D 21       EXAM_PATT:      LXI     H,WCS_EX_PATT
57FE 2947 03'16'11               LXI     D,PATT_A
5801 2948 0C0D'C3                JMP     EXAM_SUB_NOADD
5804 2949
5804 2950
5804 2951
5804 2952                ;*****************************************************************************
5804 2953                ;
5804 2954                ;  EXAM_POSIT
5804 2955                ;  THIS ROUTINE EXAMINES THE IDC ECC POSITION REGISTER
5804 2956                ;
5804 2957                ;*****************************************************************************
5804 2958
5804 2959 00 5E 21       EXAM_POSIT:     LXI     H,WCS_EX_POSIT
5807 2960 03'1B'11               LXI     D,POSIT_A
580A 2961 0C0D'C3                JMP     EXAM_SUB_NOADD
580D 2962
580D 2963
580D 2964
580D 2965                ;*****************************************************************************
580D 2966                ;
580D 2967                ;  EXAM_SUB
580D 2968                ;  THIS ROUTINE IS CALLED BY EXAMINE ROUTINES THAT WANT TO CALL A
580D 2969                ;  SUBROUTINE IN THE WCS WITH PARAMETER OF ADDRESS IN LS 5.  THE
580D 2970                ;  EXAM_SUB_NOADD ENTRY IS FOR EXAMINES THAT DO NOT USE AN ADDRESS
580D 2971                ;
580D 2972                ;  INPUT CONDITIONS
580D 2973                ;       H+L   = ADDRESS OF WCS SUBROUTINE
580D 2974                ;       D+E   = ADDRESS OF STRING PRINTOUT (IMPLIED 4 CHAR LENGTH)
580D 2975                ;       INPUT_NUM = ADDRESS TO EXAMINE (IF APPLICABLE)
580D 2976                ;
580D 2977                ;*****************************************************************************
580D 2978
580D 2979 02F1'32 3C AF  EXAM_SUB_NOADD: SET     NOADD_FLG              ;SET FLAG, NO ADDRESS REQUIRED
5812 2980
5812 2981 11 06          EXAM_SUB:       MVI     B,<^X11>               ;ERROR CODE IF WCS NOT LOADED
5814 2982 0F5E'CD                CALL    CHECK_WCS_LOADED       ;CHECK FOR WCS LOADED SET
5817 2983                                                       ; RETURN IF OK
5817 2984
5817 2985 0C36'CD                CALL    EXAM_DEP_SUB           ;SETUP AND ERROR CHECK FOR
581A 2986                                                       ; EXAM AND DEP SUBROUTINES
581A 2987 0F6E'CD                CALL    CHECK_ERR_10           ;SEE IF NO ADDRESS ERROR
581D 2988 0239'2A                LHLD    EXAM_TEMP
```

```
5820 2989 05 0E                                 MVI     C,5
5822 2990 1243'CD                               CALL    PRINTER
5825 2991 13CE'CD                               CALL    START_CPU               ;START THE SUBROUTINE
5828 2992 1A7A'CD                               CALL    SUB_ATTENTION           ;WAIT FOR EXIT
582B 2993 06 3E                                 MVI     A,LS_6                  ;READ RETURNED DATA FROM LS 6
582D 2994 1435'CD                               CALL    READ_LS_R
5830 2995 119C'CD                               CALL    MOV_DATA0_HEXBUF        ;MOVE DATA TO HEX PRINT AREA
5833 2996 11B8'C3                               JMP     PRINT_HEX_4             ;PRINT THE DATA
5836 2997
5836 2998       ;*****************************************************************************
5836 2999       ;
5836 3000       ;  EXAM_DEP_SUB
5836 3001       ;
5836 3002       ;  THIS ROUTINE SETS UP THE ADDRESS FOR EXAMINE AND DEPOSIT ROUTINES THAT
5836 3003       ;  USE WCS SUBROUTINES.  IT ALSO CHECKS FOR ERRORS IN THE COMMAND.
5836 3004       ;
5836 3005       ;*****************************************************************************
5836 3006
5836 3007
5836 3008 7C                    EXAM_DEP_SUB:   MOV     A,H                     ;GET STARTING ADDRESS
5837 3009 65                                    MOV     H,L
5838 3010 6F                                    MOV     L,A
5839 3011 03A1'22                               SHLD    STARTING_UPC
583C 3012
583C 3013 EB                                    XCHG
583D 3014 0239'22                               SHLD    EXAM_TEMP               ;GET LABEL IF EXAMINE
5840 3015
5840 3016 0F 02F1'3A            ****************IFN     NOADD_FLG               ;SEE IF NEED ADDRESS
          0C52'DA               *
5847 3017 0FCF'CD               *               CALL    INPUT_NUM_IF            ;GET ADDRESS INPUT, SET
584A 3018                       *                                               ; ERROR FLG IF NO INPUT
584A 3019                       *
584A 3020 1193'CD               *               CALL    MOV_INPUT_DATA0         ;MOVE 4 BYTES OF INPUT_NUM
584D 3021                       *                                               ; TO DATA0 THROUGH DATA3
584D 3022 05 3E                 *               MVI     A,LS_5
584F 3023 1415'CD               *               CALL    WRITE_LS_R              ;PUT ADDRESS IN LS 5
5852 3024                       ****************ENDIF
5852 3025
5852 3026 02F1'32 AF                            CLEAR   NOADD_FLG               ;CLEAR FLAG IF SET
5856 3027 C9                                    RET                             ;DONE
5857 3028
5857 3029
5857 3030       ;*****************************************************************************
5857 3031       ;
5857 3032       ;  DEP_COMMAND
5857 3033       ;
5657 3034       ;*****************************************************************************
5857 3035
5857 3036 06'5F'21              DEP_COMMAND:    LXI     H,DEP_TAB               ;POINTER TO PARSE TABLE
585A 3037 1FDB'C3                               JMP     PAR_TEMP_BUF
585D 3038
585D 3039
585D 3040       ;*****************************************************************************
585D 3041       ;
585D 3042       ;  DEP_RAM
585D 3043       ;  THIS ROUTINE DEPOSITS THE RAM OF THE 8085 CONSOLE WITH THE DATA AT THE
585D 3044       ;  ADDRESS GIVEN
585D 3045       ;
585D 3046       ;*****************************************************************************
585D 3047
```

```
585D 3048 0F6B'CD                DEP_RAM:    CALL    INPUT_NUM_CHECK         ;CHECK FOR INPUT,REPORT ERR
5860 3049                                                                    ; IF NO INPUT, ELSE LOAD
5860 3050                                                                    ; INPUT_NUM WITH RAM ADDR
5860 3051
5860 3052 043F'3A                            LDA     INPUT_NUM+2             ;NUMBER IS DEP ADDRESS
5863 3053 67                                 MOV     H,A
5864 3054 0440'3A                            LDA     INPUT_NUM+3
5867 3055 6F                                 MOV     L,A
5868 3056 03C6'22                            SHLD    TEMP_WORD
586B 3057
586B 3058 0F6B'CD                            CALL    INPUT_NUM_CHECK         ;CHECK FOR INPUT,REPORT ERR
586E 3059                                                                    ; IF NO INPUT, ELSE LOAD
586E 3060                                                                    ; INPUT_NUM WITH RAM DATA
586E 3061
586E 3062 03C6'2A                            LHLD    TEMP_WORD               ;POINT TO TEMP WORD
5871 3063 0440'3A                            LDA     INPUT_NUM+3             ;GET DATA
5874 3064 77                                 MOV     M,A                     ;DEP DATA
5875 3065
5875 3066 C9                                 RET
5876 3067
5876 3068
5876 3069
5876 3070                        ;*******************************************************************************
5876 3071                        ;
5876 3072                        ;  DEP_CSR
5876 3073                        ;  THIS ROUTINE DEPOSITS THE CSR.
5876 3074                        ;
5876 3075                        ;*******************************************************************************
5876 3076
5876 3077 0F6B'CD                DEP_CSR:    CALL    INPUT_NUM_CHECK         ;CHECK FOR INPUT,REPORT ERR
5879 3078                                                                    ; IF NO INPUT, ELSE LOAD
5879 3079                                                                    ; INPUT_NUM WITH CSR DATA
5879 3080
5879 3081 04'3E'21                           LXI     H,INPUT_NUM+1           ;MOVE TO CSR LOAD AREA
587C 3082 00'B8'11                           LXI     D,CSR_VALUE
587F 3083 118E'CD                            CALL    MOVER_3_R
5882 3084
5882 3085 0064'32 AF                         CLEAR   PARITY_ON               ;DONT WRITE PARITY BIT
5886 3086                                                                    ;ON DEPOSITS
5886 3087 1116'C3                            JMP     LOAD_CSR_R              ;PUT THE VALUE IN CSR
5889 3088
5889 3089
5889 3090                        ;*******************************************************************************
5889 3091                        ;
5889 3092                        ;  DEP_UPC
5889 3093                        ;  THIS ROUTINE DEPOSITS THE UPC.
5889 3094                        ;
5889 3095                        ;*******************************************************************************
5889 3096
5889 3097 0F6B'CD                DEP_UPC:    CALL    INPUT_NUM_CHECK         ;CHECK FOR INPUT,REPORT ERR
588C 3098                                                                    ; IF NO INPUT, ELSE LOAD
588C 3099                                                                    ; INPUT_NUM WITH UPC DATA
588C 3100
588C 3101 043F'2A                            LHLD    INPUT_NUM+2             ;POINT TO DATA
588F 3102 00C2'22                            SHLD    UPC_VALUE               ;MOVE VALUE TO CSR LOAD AREA
5892 3103
5892 3104 1148'C3                            JMP     LOAD_UPC_R              ;PUT VALUE IN CSR
5895 3105
5895 3106                        ;*******************************************************************************
5895 3107                        ;
```

```
5895 3108
5895 3109
5895 3110
5895 3111
5895 3112
5895 3113 0F6B'CD                ;  DEP_WRK
                                 ;  THIS ROUTINE DEPOSITS 2901 WORKING REGS
                                 ;
                                 ;*****************************************************************************

                                 DEP_WRK:        CALL    INPUT_NUM_CHECK         ;CHECK FOR INPUT,REPORT ERR
5898 3114                                                                        ; IF NO INPUT, ELSE LOAD
5898 3115                                                                        ; INPUT_NUM WITH WR NUMBER
5898 3116
5898 3117 0440'3A                                LDA     INPUT_NUM+3             ;GET WRK REG NUMBER
589B 3118 03C5'32                                STA     TEMP_BYTE               ;SAVE WORKING REG TO DEP
589E 3119
589E 3120 0F6B'CD                                CALL    INPUT_NUM_CHECK         ;CHECK FOR INPUT,REPORT ERR
58A1 3121                                                                        ; IF NO INPUT, ELSE LOAD
58A1 3122                                                                        ; INPUT_NUM WITH WR DATA
58A1 3123
58A1 3124 04'2E'21                               LXI     H,S_ED_WRK              ;RESET THE INSTRUCTION TO
58A4 3125 04'2B'11                               LXI     D,X_ED_WRK+1            ;NO MODIFIED BITS
58A7 3126 118E'CD                                CALL    MOVER_3_R
58AA 3127
58AA 3128 1193'CD                                CALL    MOV_INPUT_DATA0         ;MOVE 4 BYTES OF INPUT_NUM
58AD 3129                                                                        ; TO DATA0 THROUGH DATA3
58AD 3130
58AD 3131 03C5'3A                                LDA     TEMP_BYTE
58B0 3132 01CC'32                                STA     B_ADDRS                 ;MOV WORKING REG 0 TO REG X
58B3 3133 04'2B'21                               LXI     H,X_ED_WRK+1
58B6 3134 1462'CD                                CALL    MAKE_B                  ;MODIFY INST TO BE MOVE TO X0
58B9 3135
58B9 3136 149E'CD                                CALL    WRITE_DATA_32_R         ;WRITE THE DATA TO 2901
58BC 3137 04'2A'21                               LXI     H,X_ED_WRK
58BF 3138 15C2'C3                                JMP     PERFORM_CSR_R           ;MOV REG TO READ REG


                                 ;*****************************************************************************
                                 ;
                                 ;  DEP_WCS
                                 ;  THIS ROUTINE DEPOSITS WCS LOCATIONS
                                 ;
                                 ;*****************************************************************************

58C2 3148 0F6B'CD                DEP_WCS:        CALL    INPUT_NUM_CHECK         ;CHECK FOR INPUT,REPORT ERR
58C5 3149                                                                        ; IF NO INPUT, ELSE LOAD
58C5 3150                                                                        ; INPUT_NUM WITH WCS ADDRESS
58C5 3151
58C5 3152 043F'2A                                LHLD    INPUT_NUM+2             ;MOVE FIRST NUM TO WCS
58C8 3153                                                                        ; ADDRESS
58C8 3154 006C'22                                SHLD    WCS_ADDRESS
58CB 3155
58CB 3156 0F6B'CD                                CALL    INPUT_NUM_CHECK         ;CHECK FOR INPUT,REPORT ERR
58CE 3157                                                                        ; IF NO INPUT, ELSE LOAD
58CE 3158                                                                        ; INPUT_NUM WITH WCS DATA
58CE 3159
58CE 3160 04'3E'21                               LXI     H,INPUT_NUM+1           ;MOVE SECOND NUMBER TO
58D1 3161 00'B4'11                               LXI     D,WCS_VALUE             ;WCS_VALUE
58D4 3162 118E'CD                                CALL    MOVER_3_R
58D7 3163
58D7 3164 0064'32 AF                             CLEAR   PARITY_ON               ;DON'T CALC PARITY ON WRITE
58DB 3165 10B9'C3                                JMP     WRITE_WCS_R             ;WRITE SECOND NUM TO WCS
58DE 3166
58DE 3167                        ;*****************************************************************************
```

```
58DE 3168
58DE 3169
58DE 3170
58DE 3171
58DE 3172
58DE 3173
58DE 3174 0F6B'CD                ;
                                 ;  DEP_OS
                                 ;  THIS ROUTINE DEPOSITS THE GIVEN DATA INTO OS
                                 ;
                                 ;*********************************************************************
                                 DEP_OS:         CALL    INPUT_NUM_CHECK         ;CHECK FOR INPUT,REPORT ERR
58E1 3175                                                                        ; IF NO INPUT, ELSE LOAD
58E1 3176                                                                        ; INPUT_NUM WITH OS DATA
58E1 3177
58E1 3178 0440'3A                                LDA     INPUT_NUM+3             ;GET 8 BITS AND MOVE TO
58E4 3179 00AC'32                                STA     DATA3                   ;WRITE AREA
58E7 3180
58E7 3181 7C 3E                                  MVI     A,OS_ADD                ;OS ADDRESS IN LS ADDRESS SPACE
58E9 3182 1415'C3                                JMP     WRITE_LS_R
58EC 3183
58EC 3184
58EC 3185                        ;*********************************************************************
58EC 3186                        ;
58EC 3187                        ;  DEP_LS
58EC 3188                        ;  THIS ROUTINE DEPOSITS DATA INTO LS AT THE GIVEN ADDRESS
58EC 3189                        ;
58EC 3190                        ;*********************************************************************
58EC 3191
58EC 3192 0F6B'CD                DEP_LS:         CALL    INPUT_NUM_CHECK         ;CHECK FOR INPUT,REPORT ERR
58EF 3193                                                                        ; IF NO INPUT, ELSE LOAD
58EF 3194                                                                        ; INPUT_NUM WITH LS ADDRESS
58EF 3195
58EF 3196 0440'3A                                LDA     INPUT_NUM+3             ;GET LS ADDRESS AND SAVE
58F2 3197 03C5'32                                STA     TEMP_BYTE
58F5 3198
58F5 3199 0F6B'CD                                CALL    INPUT_NUM_CHECK         ;CHECK FOR INPUT,REPORT ERR
58F8 3200                                                                        ; IF NO INPUT, ELSE LOAD
58F8 3201                                                                        ; INPUT_NUM WITH LS DATA
58F8 3202
58F8 3203 1193'CD                                CALL    MOV_INPUT_DATA0         ;MOVE 4 BYTES OF INPUT_NUM
58FB 3204                                                                        ; TO DATA0 THROUGH DATA3
58FB 3205
58FB 3206 03C5'3A                                LDA     TEMP_BYTE               ;GET ADDRESS AGAIN
58FE 3207 1415'C3                                JMP     WRITE_LS_R              ;AND PERFORM WRITE
5901 3208
5901 3209
5901 3210                        ;*********************************************************************
5901 3211                        ;
5901 3212                        ;  DEP_MCT
5901 3213                        ;  THIS ROUTINE DEPOSITS MEMORY CONTROL + STATUS
5901 3214                        ;
5901 3215                        ;*********************************************************************
5901 3216
5901 3217 00 50 21               DEP_MCT:        LXI     H,WCS_DE_MCT
5904 3218 0D30'C3                                JMP     DEP_SUB
5907 3219
5907 3220
5907 3221                        ;*********************************************************************
5907 3222                        ;
5907 3223                        ;  DEP_MM
5907 3224                        ;  THIS ROUTINE DEPOSITS MAIN MEMORY
5907 3225                        ;
5907 3226                        ;*********************************************************************
5907 3227
```

```
5907 3228 00 54 21          DEP_MM:         LXI     H,WCS_DE_MM
590A 3229 0D30'C3                           JMP     DEP_SOB
590D 3230
590D 3231
590D 3232                   ;*****************************************************************************
590D 3233                   ;
590D 3234                   ;  DEP_TB
590D 3235                   ;  THIS ROUTINE DEPOSITS TRANSLATION BUFFER LOCATIONS
590D 3236                   ;
590D 3237                   ;*****************************************************************************
590D 3238
590D 3239 00 52 21          DEP_TB:         LXI     H,WCS_DE_TB
5910 3240 0D30'C3                           JMP     DEP_SOB
5913 3241
5913 3242
5913 3243                   ;*****************************************************************************
5913 3244                   ;
5913 3245                   ;  DEP_UBS
5913 3246                   ;  THIS ROUTINE DEPOSITS UNIBUS TRANSLATION BUFFER LOCATIONS
5913 3247                   ;
5913 3248                   ;*****************************************************************************
5913 3249
5913 3250 00 58 21          DEP_UBS:        LXI     H,WCS_DE_UBS
5916 3251 0D30'C3                           JMP     DEP_SOB
5919 3252
5919 3253
5919 3254                   ;*****************************************************************************
5919 3255                   ;
5919 3256                   ;  DEP_ICSR
5919 3257                   ;  THIS ROUTINE DEPOSITS THE IDC CSR
5919 3258                   ;
5919 3259                   ;*****************************************************************************
5919 3260
5919 3261 00 5F 21          DEP_ICSR:       LXI     H,WCS_DE_ICSR
591C 3262 0D2B'C3                           JMP     DEP_SOB_NOADD
591F 3263
591F 3264
591F 3265                   ;*****************************************************************************
591F 3266                   ;
591F 3267                   ;  DEP_IDAR
591F 3268                   ;  THIS ROUTINE DEPOSITS THE IDC DISK ADDRESS REGISTER
591F 3269                   ;
591F 3270                   ;*****************************************************************************
591F 3271
591F 3272 00 60 21          DEP_IDAR:       LXI     H,WCS_DE_IDAR
5922 3273 0D2B'C3                           JMP     DEP_SOB_NOADD
5925 3274
5925 3275
5925 3276                   ;*****************************************************************************
5925 3277                   ;
5925 3278                   ;  DEP_DBUF
5925 3279                   ;  THIS ROUTINE DEPOSITS THEN IDC DATA BUFFER
5925 3280                   ;
5925 3281                   ;*****************************************************************************
5925 3282
5925 3283 00 61 21          DEP_DBUF:       LXI     H,WCS_DE_DBUF
5928 3284 0D2B'C3                           JMP     DEP_SOB_NOADD
592B 3285
592B 3286
592B 3287
```

```
592B 3288
592B 3289
592B 3290
592B 3291
592B 3292
592B 3293
592B 3294
592B 3295
592B 3296
592B 3297
592B 3298
592B 3299
592B 3300
592B 3301
592B 3302
592B 3303 02F1'32 3C AF
5930 3304
5930 3305 11 06
5932 3306 0F5E'CD
5935 3307
5935 3308
5935 3309 0C36'CD
5938 3310
5938 3311
5938 3312 0F6E'CD
593B 3313 0F6B'CD
593E 3314
593E 3315
593E 3316
593E 3317 1193'CD
5941 3318
5941 3319
5941 3320 06 3E
5943 3321 1415'CD
5946 3322
5946 3323 13CE'CD
5949 3324 1A7A'C3
594C 3325
594C 3326
594C 3327
594C 3328
594C 3329
594C 3330
594C 3331
594C 3332
594C 3333
594C 3334 1063'CD
594F 3335
594F 3336 0F6B'CD
5952 3337
5952 3338
5952 3339
5952 3340 043F'3A
5955 3341
5955 3342 67
5956 3343 B7
5957 3344 0D61'C2
595A 3345
595A 3346 0440'3A
595D 3347 B7
```

```
;*****************************************************************************
;
;  DEP_SUB
;  THIS ROUTINE IS CALLED BY DEPOSIT ROUTINES THAT WANT TO CALL A
;  SUBROUTINE IN THE WCS WITH PARAMETER OF ADDRESS IN LS 5 AND DATA
;  IN LS 6.  THIS ROUTINE WILL INPUT THE NUMBERS,PUT THEM IN LS AND
;  CALL THE WCS SUBROUTINE.  IF NO ADDRESS IS REQUIRED, THE ENTRY POINT
;  DEP_SUB_NOADD IS USED.
;
;  INPUT CONDITIONS
;       H+L  = ADDRESS OF WCS SUBROUTINE
;       ADDRESS (IF NEEDED) AND DATA ARE IN TERMINAL INPUT STREAM
;
;*****************************************************************************

DEP_SUB_NOADD:  SET     NOADD_FLG               ;SET FLAG, NO ADDRESS USED

DEP_SUB:        MVI     B,<^X11>                ;ERROR CODE IF WCS NOT LOADED
                CALL    CHECK_WCS_LOADED        ;CHECK FOR WCS LOADED SET
                                                ; RETURN IF OK

                CALL    EXAM_DEP_SUB            ;SETUP AND ERROR CHECK FOR
                                                ; EXAM AND DEP SUBROUTINES

                CALL    CHECK_ERR_10            ;SEE IF NO ADDRESS ERROR
                CALL    INPUT_NUM_CHECK         ;CHECK FOR INPUT,REPORT ERR
                                                ; IF NO INPUT, ELSE LOAD
                                                ; INPUT_NUM WITH DEP DATA

                CALL    MOV_INPUT_DATA0         ;MOVE 4 BYTES OF INPUT_NUM
                                                ; TO DATA0 THROUGH DATA3

                MVI     A,LS_6
                CALL    WRITE_LS_R              ;PUT  DATA IN LS 6
                                                ;FOR SUBROUTINE TO READ
                CALL    START_CPU               ;START THE SUBROUTINE
                JMP     SUB_ATTENTION           ;WAIT FOR EXIT

;*****************************************************************************
;
;  START
;  THIS COMMAND MAY BE USED TO START THE 8085 CPU AT THE GIVEN ADDRESS
;
;*****************************************************************************

START:          CALL    SKIP_TO_SPEC            ;SKIP OVER "/U" PART OF COMMAND

                CALL    INPUT_NUM_CHECK         ;CHECK FOR INPUT,REPORT ERR
                                                ; IF NO INPUT, ELSE LOAD
                                                ; INPUT_NUM WITH START ADDR

                LDA     INPUT_NUM+2             ;GET THE INPUT NUMBER
                                                ; INTO H+L
                MOV     H,A                     ;H GETS HIGH BYTE OF ADDRESS
                ORA     A                       ;SET CONDITION CODES
                JNZ     1$                      ;JUMP IF NOT S/U 0

                LDA     INPUT_NUM+3             ;GET LOW BYTE OF ADDRESS
                ORA     A                       ;SET CONDITION CODES
```

```
595E 3348 0705'CA                       JZ      STNDRD_CON_COM          ;GO TO ROUTINE THAT RESETS
5961 3349                                                               ; INTERVAL TIMER AND GOES TO
5961 3350                                                               ; ADDRESS 0 IF S/U 0
5961 3351 0440'3A               1$:     LDA     INPUT_NUM+3
5964 3352 6F                            MOV     L,A
5965 3353 E9                            PCHL                            ;MOVE NUMBER TO PC(BRANCH)
5966 3354
5966 3355
5966 3356               ;*******************************************************************************
5966 3357               ;
5966 3358               ;  X COMMAND
5966 3359               ;  THIS COMMAND WILL INPUT OR OUTPUT BINARY DATA FROM CONSOLE RAM OR
5966 3360               ;  THE WCS. THE SPECIFICATION FOR THIS COMMAND IS IN THE MID RANGE
5966 3361               ;  CONSOLE SPEC...
5966 3362               ;  NOTE: FOR PURPOSES OF THE MICRO MONITOR IF THE ADDRESS 4100 IS GIVEN
5966 3363               ;  AND A POSITIVE NUMBER (MEANING DEPOSIT TO CONSOLE RAM) THE PROGRAM
5966 3364               ;  WILL BRANCH TO THE ROM TO HAVE THE COMMAND EXECUTED
5966 3365               ;
5966 3366               ;*******************************************************************************
5966 3367
5966 3368 037C'32 3C AF XU:             SET     SLASH_U                 ;SLASH U COMMAND TYPED
596B 3369 0D72'C3                       JMP     X_COMMAND
596E 3370
596E 3371 037C'32 AF            XC:     CLEAR   SLASH_U                 ;SLASH P COMMAND TYPED
5972 3372
5972 3373 02F3'32 3C AF         X_COMMAND:      SET     NO_RETURN       ;DON'T RETURN IF ERROR
5977 3374 1063'CD                       CALL    SKIP_TO_SPEC            ;SKIP TO ADDRESS FIELD
597A 3375
597A 3376 0FCF'CD                       CALL    INPUT_NUM_IF            ;GET ADDRESS
597D 3377
597D 3378 31 06                         MVI     B,<^X31>                ;ERROR CODE IF ERROR SET
597F 3379                                                               ; (NO X COMMAND ADDRESS)
597F 3380 0F70'CD                       CALL    CHECK_ERR_FLG           ;SEE IF ERROR, RETURN IF NOT
5982 3381
5982 3382 04'3D'21                      LXI     H,INPUT_NUM
5985 3383 04'0F'11                      LXI     D,X_COM_ADDRS
5988 3384 11A2'CD                       CALL    MOVER_4_R               ;SAVE ADDRESS FIELD
5988 3385
5988 3386 0FCF'CD                       CALL    INPUT_NUM_IF            ;GET COUNT
598E 3387
598E 3388 32 06                         MVI     B,<^X32>                ;ERROR CODE IF ERROR SET
5990 3389                                                               ; (NO X COMMAND COUNT)
5990 3390 0F70'CD                       CALL    CHECK_ERR_FLG           ;SEE IF ERROR, RETURN IF NOT
5993 3391
5993 3392 40FE 32 3C AF                SET     CTLDIS                  ;DISABLE CONTROL CHARACTER
5998 3393                                                               ; CHECK
5998 3394 131B'CD                       CALL    INPUT_CHAR_IF           ;GET THE CHECKSUM (WAITS IN
5998 3395                                                               ; ROUTINE TIL CHAR RECEIVED)
5998 3396
5998 3397 30 06                         MVI     B,<^X30>                ;ERROR CODE IF CHECKSUM ERR
599D 3398 4082 3A                       LDA     BYTSUM                  ;GET CHECKSUM OF LINE
59A0 3399 B7                            ORA     A                       ;SET CONDITION CODES
59A1 3400 0F7E'C2                       JNZ     ERROR_ROUTINE           ;REPORT CHECKSUM ERROR IF
59A4 3401                                                               ; BYTSUM IS NOT 0
59A4 3402
59A4 3403 80 E6 043D'3A         ********************    IFMBI   INPUT_NUM,NEG_CNT       ;ELSE SEE IF WRITE OR READ
          0DB2'CA               *
59AC 3404 0E75'CD               *                   CALL    OUT_BIN_STRING          ;COUNT IS NEG ..OUTPUT DATA
59AF 3405                       *
59AF 3406 0DB5'C3               ********************    ELSE
```

```
59B2 3407                    *
59B2 3408 0DC0'CD            *                          CALL    IN_BIN_STRING          ;COUNT IS POS ..INPUT DATA
59B5 3409                    ******************         ENDIF
59B5 3410
59B5 3411 AF                                            XRA     A                      ;CLEAR FLAGS
59B6 3412 40FE 32                                       STA     CTLDIS                 ;REENABLE CNTL CHAR CHECK
59B9 3413 40FF 32                                       STA     NOCHK                  ;REENABLE CR ON OUTPUT TO TERM
59BC 3414 02F3'32                                       STA     NO_RETURN              ;INIT NO RETURN ON ERROR FLAG
59BF 3415 C9                                            RET
59C0 3416
59C0 3417
59C0 3418                                       ;*******************************************************************************
59C0 3419                                       ;
59C0 3420                                       ;  IN_BIN_STRING
59C0 3421                                       ;  THIS ROUTINE IS USED TO INPUT A BINARY STRING FOR THE X COMMAND
59C0 3422                                       ;  THE ADDRESS MUST BE IN X_COM_ADDRS
59C0 3423                                       ;  THE COUNT MUST BE IN INPUT_NUM
59C0 3424                                       ;
59C0 3425                                       ;*******************************************************************************
59C0 3426
59C0 3427 41 FE 0411'3A      ****************** IN_BIN_STRING:  IFEQI   X_COM_ADDRS+2,HEX_41   ;SEE IF DESTINATION ADDRESS IS
          0DD7'C2            *
59C8 3428                    *                                                              ; 4100(H)
59C8 3429 00 FE 0412'3A      * ****************                IFEQI   X_COM_ADDRS+3,0
          0DD7'C2            * *
59D0 3430 037C'3A            * *                             LDA     SLASH_U                ;SEE IF DESTINATION IS 8085 RAM
59D3 3431 B7                 * *                             ORA     A                      ;SET CONDITION CODES
59D4 3432 000B C2            * *                             JNZ     ROM_X_ADDRS            ;GO TO ROM CODE IF DEST IS 8085
59D7 3433                    * *                                                            ; RAM SINCE LOAD WILL OVERLAY
59D7 3434                    * *                                                            ; MICMON
59D7 3435                    * ****************              ENDIF
59D7 3436                    ******************             ENDIF
59D7 3437
59D7 3438 0411'2A                                           LHLD    X_COM_ADDRS+2          ;GET ADDRESS TO WRITE
59DA 3439 00C2'22                                           SHLD    UPC_VALUE
59DD 3440 1148'CD                                           CALL    LOAD_UPC_R             ;LOAD STARTING UPC, IN CASE
59E0 3441                                                                                  ; THIS IS A WCS LOAD
59E0 3442 10F9'CD                                           CALL    NOP_CSR_R              ;KEEP MACHINE IDLE.
59E3 3443
59E3 3444 0409'32 3C AF                                     SET     WCS_PNT                ;SET TO FIRST BYTE OF 3
59E8 3445                                                                                  ;BYTES OF WCS WORDS
59E8 3446 1208'CD                                           CALL    PRINT_PROMPT           ;ELSE PRINT PROMPT FOR DATA
59EB 3447 4082 32 AF                                        CLEAR   BYTSUM                 ;START NEW CHECKSUM
59EF 3448
59EF 3449 0F 037C'3A         ******************             IF      SLASH_U                ;DO DEPOSIT TO RAM IF X/U
          0E16'D2            *
59F6 3450                    *
59F6 3451 4145 CD            *                      1$:     CALL    GSVECT                 ;GET NEXT CHAR
59F9 3452 0DF6'D2            *                              JNC     1$                     ;LOOP UNTIL CHAR RECEIVED
59FC 3453                    *
59FC 3454 04'11'21           *                              LXI     H,X_COM_ADDRS+2        ;POINT TO RAM ADDRESS
59FF 3455 56                 *                              MOV     D,M                    ;STORE HIGH BYTE IN D
5A00 3456 23                 *                              INX     H                      ;POINT TO LOW BYTE
5A01 3457 5E                 *                              MOV     E,M                    ;STORE LOW BYTE IN E
5A02 3458 EB                 *                              XCHG                           ;LOAD HL WITH DE
5A03 3459                    *
5A03 3460 77                 *                              MOV     M,A                    ;WRITE CHAR INTO RAM
5A04 3461                    *
5A04 3462 04'11'21           *                              LXI     H,X_COM_ADDRS+2        ;INCREMENT ADDRESS
5A07 3463 1581'CD            *                              CALL    INC_WORD_R
```

```
5A0A 3464                       *
5A0A 3465 04'3F'21              *                                   LXI     H,INPUT_NUM+2
5A0D 3466 158C'CD               *                                   CALL    DECR_WORD_R             ;DECREMENT COUNT
5A10 3467 0DF6'C2               *                                   JNZ     1$                      ;LOOP UNTIL COUNT=0
5A13 3468                       *
5A13 3469 0E65'C3               ******************                  ELSE                            ;DO DEPOSIT TO WCS IF X/C
5A16 3470                       *
5A16 3471 03 FE 0409'3A         * ***************** 2$:               IFEQI   WCS_PNT,3
          0E35'C2               * *
5A1E 3472 4145 CD               * *                   7$:           CALL    GSVECT                  ;GET NEXT CHAR
5A21 3473 0E1E'D2               * *                                 JNC     7$                      ;LOOP UNTIL CHAR RECEIVED
5A24 3474 0A D3                 * *                                 OUT     WCSB2                   ;WRITE THIRD BYTE IN REG
5A26 3475                       * *
5A26 3476 37 D3                 * *                                 OUT     SETWCK                  ;WRITE WCS
5A28 3477 36 D3                 * *                                 OUT     CLRWCK
5A2A 3478                       * *
5A2A 3479 27 D3                 * *                                 OUT     SETUPK                  ;INCREMENT UPC TO NEXT ADDRESS
5A2C 3480 26 D3                 * *                                 OUT     CLRUPK
5A2E 3481                       * *
5A2E 3482 0409'32 AF            * *                                 CLEAR   WCS_PNT                 ;RESET TO FIRST BYTE
5A32 3483 0E58'C3               * *                                 JMP     3$                      ;SKIP OTHER CHECKS
5A35 3484                       * *****************                 ENDIF
5A35 3485                       *
5A35 3486 01 FE 0409'3A         * *****************                 IFEQI   WCS_PNT,1
          0E48'C2               * *
5A3D 3487 4145 CD               * *                   6$:           CALL    GSVECT                  ;GET NEXT CHAR
5A40 3488 0E3D'D2               * *                                 JNC     6$                      ;LOOP UNTIL CHAR RECEIVED
5A43 3489 08 D3                 * *                                 OUT     WCSB0                   ;WRITE FIRST BYTE IN REG
5A45 3490 0E58'C3               * *                                 JMP     3$                      ;SKIP OTHER CHECKS
5A48 3491                       * *****************                 ENDIF
5A48 3492                       *
5A48 3493 02 FE 0409'3A         * *****************                 IFEQI   WCS_PNT,2
          0E58'C2               * *
5A50 3494 4145 CD               * *                   5$:           CALL    GSVECT                  ;GET NEXT CHAR
5A53 3495 0E50'D2               * *                                 JNC     5$                      ;LOOP UNTIL CHAR RECEIVED
5A56 3496 09 D3                 * *                                 OUT     WCSB1                   ;WRITE SECOND BYTE IN REG
5A58 3497                       * *****************                 ENDIF
5A58 3498                       *
5A58 3499 04'09'21              *                   3$:           LXI     H,WCS_PNT               ;INC POINTER TO NEXT BYTE
5A5B 3500 34                    *                                 INR     M
5A5C 3501                       *
5A5C 3502 04'3F'21              *                                 LXI     H,INPUT_NUM+2
5A5F 3503 158C'CD               *                                 CALL    DECR_WORD_R             ;DECREMENT COUNT
5A62 3504 0E16'C2               *                                 JNZ     2$                      ;LOOP UNTIL COUNT=0
5A65 3505                       *
5A65 3506                       ******************                ENDIF
5A65 3507
5A65 3508 4145 CD                                   4$:           CALL    GSVECT                  ;GET NEXT CHAR (CHECKSUM)
5A68 3509 0E65'D2                                                 JNC     4$                      ;LOOP UNTIL CHAR RECEIVED
5A6B 3510
5A6B 3511 30 06                                                   MVI     B,<^X30>                ;ERROR CODE IF CHECKSUM ERR
5A6D 3512 4082 3A                                                 LDA     BYTSUM                  ;GET CHECKSUM OF DATA
5A70 3513 B7                                                      ORA     A                       ;SET CONDITION CODES
5A71 3514 0F7E'C2                                                 JNZ     ERROR_ROUTINE           ;REPORT CHECKSUM ERROR IF
5A74 3515                                                                                         ; BYTSUM IS NOT 0
5A74 3516
5A74 3517 C9                                                      RET
5A75 3518
5A75 3519
5A75 3520                       ;*************************************************************************************
```

```
5A75 3521
5A75 3522                       ;  OUT_BIN_STRING
5A75 3523                       ;  THIS ROOTINE OUTPUTS A BINARY STRING FOR THE X COMMAND
5A75 3524                       ;  THE ADDRESS MUST BE IN X_COM_ADDRS
5A75 3525                       ;  THE COUNT MUST BE IN INPOT_NOM
5A75 3526                       ;
5A75 3527                       ;*****************************************************************************
5A75 3528
5A75 3529 1208'CD               OUT_BIN_STRING: CALL    PRINT_PROMPT            ;PRINT PROMPT
5A78 3530
5A78 3531 AF                            XRA     A                       ;CLERA FLAGS
5A79 3532 4082 32                       STA     BYTSUM                  ;CLEAR OUTPUT CHECKSUM
5A7C 3533 40FE 32                       STA     CTLDIS                  ;CLEAR CONTROL CHAR DISABLE
5A7F 3534
5A7F 3535 3C                            INR     A                       ;SET FLAGS
5A80 3536 40FF 32                       STA     NOCHK                   ;IGNORE <CR>
5A83 3537 0409'32                       STA     WCS_PNT                 ;SET TO FIRST BYTE OF 3
5A86 3538                                                               ;BYTES OF WCS WORDS
5A86 3539
5A86 3540 0F 037C'3A            ********************* 1$:             IF      SLASH_U
          0E93'D2               *
5A8D 3541 0EA7'CD               *                               CALL    X_EX_U          ;GET OUTPUT FROM RAM
5A90 3542                       *
5A90 3543 0E96'C3               *********************           ELSE
5A93 3544                       *
5A93 3545 0EB9'CD               *                               CALL    X_EX_C          ;GET OUTPUT FROM WCS
5A96 3546                       *********************           ENDIF
5A96 3547
5A96 3548 04'3F'21                      LXI     H,INPUT_NUM+2
5A99 3549 158C'CD                       CALL    DECR_WORD_R             ;DECREMENT NEG COUNT UNTIL 0
5A9C 3550 0E86'C2                       JNZ     1$                      ;LOOP UNTIL COUNT=0
5A9F 3551
5A9F 3552 4082 3A                       LDA     BYTSUM                  ;CALCULATE CHECKSUM TO ADD
5AA2 3553 2F                            CMA                             ;UP TO ZERO WITH BINARY
5AA3 3554 3C                            INR     A                       ;STRING SENT
5AA4 3555
5AA4 3556 1265'C3                       JMP     TYPE_CHAR               ;SEND CHECKSUM CHAR
5AA7 3557
5AA7 3558
5AA7 3559                       ;*****************************************************************************
5AA7 3560                       ;
5AA7 3561                       ;  X_EX_U
5AA7 3562                       ;  THIS ROUTINE EXAMINES THE CONSOLE RAM FOR THE X COMMMAND
5AA7 3563                       ;  THE RECIVED BYTE IS SENT TO APT AND THE ADDRESS IS INCREMENTED
5AA7 3564                       ;
5AA7 3565                       ;*****************************************************************************
5AA7 3566
5AA7 3567 0411'3A               X_EX_U:         LDA     X_COM_ADDRS+2           ;PUT ADDRESS IN H+L
5AAA 3568 67                            MOV     H,A
5AAB 3569 0412'3A                       LDA     X_COM_ADDRS+3
5AAE 3570 6F                            MOV     L,A
5AAF 3571 7E                            MOV     A,M                     ;GET CHAR FROM MEMORY
5AB0 3572 1265'CD                       CALL    TYPE_CHAR               ;OUTPUT TO APT
5AB3 3573
5AB3 3574 04'11'21                      LXI     H,X_COM_ADDRS+2         ;INCREMENT ADDRESS
5AB6 3575 1581'C3                       JMP     INC_WORD_R
5AB9 3576
5AB9 3577
5AB9 3578                       ;*****************************************************************************
5AB9 3579                       ;
```

```
5AB9 3580                               ;  X_EX_C
5AB9 3581                               ;  THIS ROUTINE EXAMINES THE WCS FOR THE X_COMMAND
5AB9 3582                               ;  THE ROUTINE READ THE DATA FROM THE WCS 24 BITS AT A TIME AND GIVES
5AB9 3583                               ;  ONE BYTE IN PRINT_CHAR EACH TIME IT IS CALLED.  THE ADDRESS IS
5AB9 3584                               ;  INCREMENTED AFTER THE 24 BITS OF DATA (3 BYTES) IS PASSED
5AB9 3585                               ;
5AB9 3586                               ;*************************************************************************
5AB9 3587
5AB9 3588 01 FE 0409'3A      ********** X_EX_C:          IFEQI   WCS_PNT,1
          0ED4'C2            *
5AC1 3589                    *
5AC1 3590 0411'2A            *                          LHLD    X_COM_ADDRS+2          ;GET ADDRESS TO READ
5AC4 3591 006C'22            *                          SHLD    WCS_ADDRESS
5AC7 3592                    *
5AC7 3593 0064'32 Af         *                          CLEAR   PARITY_ON              ;DON'T CALC PARITY ON READ
5ACB 3594 109D'CD            *                          CALL    READ_WCS_R
5ACE 3595                    *
5ACE 3596 00B4'3A            *                          LDA     WCS_VALUE              ;GET FIRST BYTE OF WORD
5AD1 3597 1265'CD            *                          CALL    TYPE_CHAR              ;SEND TO APT
5AD4 3598                    **********                 ENDIF
5AD4 3599
5AD4 3600 02 FE 0409'3A      **********                 IFEQI   WCS_PNT,2
          0EE2'C2            *
5ADC 3601                    *
5ADC 3602 00B5'3A            *                          LDA     WCS_VALUE+1            ;GET SECOND BYTE OF WORD
5ADF 3603 1265'CD            *                          CALL    TYPE_CHAR              ;SEND TO APT
5AE2 3604                    **********                 ENDIF
5AE2 3605
5AE2 3606 03 FE 0409'3A      **********                 IFEQI   WCS_PNT,3
          0EFA'C2            *
5AEA 3607                    *
5AEA 3608 00B6'3A            *                          LDA     WCS_VALUE+2            ;GET THIRD BYTE OF WORD
5AED 3609 1265'CD            *                          CALL    TYPE_CHAR              ;SEND TO APT
5AF0 3610                    *
5AF0 3611 0409'32 AF         *                          CLEAR   WCS_PNT                ;RESET TO FIRST BYTE
5AF4 3612                    *
5AF4 3613 04'11'21           *                          LXI     H,X_COM_ADDRS+2        ;INCREMENT ADDRESS
5AF7 3614 1581'CD            *                          CALL    INC_WORD_R
5AFA 3615                    *
5AFA 3616                    **********                 ENDIF
5AFA 3617
5AFA 3618 04'09'21                                      LXI     H,WCS_PNT              ;INC POINTER TO NEXT BYTE
5AFD 3619 34                                            INR     M
5AFE 3620
5AFE 3621 C9                                            RET
5AFF 3623
5AFF 3624                               ;*************************************************************************
5AFF 3625                               ;
5AFF 3626                               ;  MISC SUBROUTINES
5AFF 3627                               ;
5AFF 3628                               ;*************************************************************************
5AFF 3629                               ;*************************************************************************
5AFF 3630                               ;
5AFF 3631                               ;  PARITY_CALC
5AFF 3632                               ;  THIS ROUTINE CALCULATES PARITY FOR THE WCS WORDS
5AFF 3633                               ;  AND PLACES IT IN BIT 23 OF WCS_WORD IF THE PARITY SELECTOR IS SET
5AFF 3634                               ;  TO BAD PARITY AS USED IN STOP ON MICRO MATCH THE PARITY WILL BE
5AFF 3635                               ;  CHANGED TO BAD PARITY IN BIT 23 OF CSR_VALUE
5AFF 3636                               ;
5AFF 3637                               ;  INPUT CONDITIONS:
```

```
5AFF 3638                       ;       H+L             ADDRESS OF 24 BIT WORD TO CALC PARITY ON
5AFF 3639                       ;       PARITY_ON       PARITY CALCULATING ENABLE FLAG
5AFF 3640                       ;       WCS_BAD_PARITY  INDICATES THAT THIS WCS WORD IS TO HAVE BAD PARITY
5AFF 3641                       ;
5AFF 3642                       ;  OUTPUT CONDITION:
5AFF 3643                       ;       WCS_BAD_PARITY  IS CLEARED IF BAD PARITY AND WCS CALC WERE SET
5AFF 3644                       ;       PARITY_ON       SET FOR PARITY CALCULATION
5AFF 3645                       ;
5AFF 3646                       ;*************************************************************************
5AFF 3647
5AFF 3648 0206'22               PARITY_CALC:    SHLD    CALC_ADD                ;SAVE INPUT POINTER
5B02 3649
5B02 3650 0F 0064'3A            ****************        IF      PARITY_ON               ;IF PARITY CALC ENABLED
          0F3B'D2               *
5B09 3651                       *
5B09 3652 0206'2A               *                       LHLD    CALC_ADD
5B0C 3653 7E                    *                       MOV     A,M                     ;GET FIRST BYTE
5B0D 3654 7F E6                 *                       ANI     HEX_7F                  ;AND OUT CURRENT PARITY BIT
5B0F 3655 23                    *                       INX     H
5B10 3656 AE                    *                       XRA     M                       ;XOR WITH SECOND AND THIRD
5B11 3657 23                    *                       INX     H                       ;BYTES OF 24 BIT WORD
5B12 3658 AE                    *                       XRA     M
5B13 3659                       *
5B13 3660 2B                    *                       DCX     H                       ;MOVE POINTER BACK AND
5B14 3661 2B                    *                       DCX     H                       ;GET THE HIGH BYTE
5B15 3662 7E                    *                       MOV     A,M
5B16 3663                       *
5B16 3664 0F1E'EA               * ****************      IFPO                            ;IF PARITY ODD
5B19 3665                       * *
5B19 3666 7F E6                 * *                     ANI     HEX_7F                  ;AND OUT PARITY BIT
5B1B 3667                       * *
5B1B 3668 0F20'C3               * ****************      ELSE
5B1E 3669                       * *
5B1E 3670 80 F6                 * *                     ORI     HEX_80                  ;OR IN PARITY BIT
5B20 3671                       * *
5B20 3672                       * ****************      ENDIF
5B20 3673                       *
5B20 3674 77                    *                       MOV     M,A                     ;PUT DATA BACK IN CSR_VALUE
5B21 3675                       *                                                       ; OR WCS VALUE
5B21 3676 0F 0407'3A            * ****************      IF      WCS_CALC                ;THIS ROOTINE WHEN WCS PARITY
          0F3B'D2               * *
5B28 3677                       * *                                                     ; IS CALCULATED
5B28 3678 0F 006E'3A            * * ***************     IF      WCS_BAD_PARITY
          0F3B'D2               * * *
5B2F 3679 0206'2A               * * *                   LHLD    CALC_ADD
5B32 3680 7E                    * * *                   MOV     A,M
5B33 3681 17                    * * *                   RAL
5B34 3682 3F                    * * *                   CMC                             ;FLIP PARITY BIT IN CARRY
5B35 3683 1F                    * * *                   RAR                             ;TO MAKE BAD PARITY
5B36 3684 77                    * * *                   MOV     M,A
5B37 3685 006E'32 AF            * * *                   CLEAR   WCS_BAD_PARITY          ;CLEAR BAD PARITY FLAG
5B3B 3686                       * * ***************     ENDIF
5B3B 3687                       * ****************      ENDIF
5B3B 3688                       *
5B3B 3689                       ******************      ENDIF
5B3B 3690
5B3B 3691 0064'32 3C AF                                 SET     PARITY_ON               ;TURN PARITY CALC BACK ON
5B40 3692
5B40 3693 C9                                            RET
5B41 3694
```

```
5B41 3695
5B41 3696          ;*****************************************************************************
5B41 3697          ;
5B41 3698          ;  RESTORE_WR
5B41 3699          ;  THIS ROUTINE CALLS A WCS SUBROUTINE THAT RESTORES THE WORKING REGS
5B41 3700          ;  OF THE 2901 FROM A SAVED AREA
5B41 3701          ;
5B41 3702          ;*****************************************************************************
5B41 3703
5B41 3704 00 57 21 RESTORE_WR:     LXI     H,WCS_REST_WR           ;ADDRESS OF SUBROUTINE
5B44 3705 0A49'C3                  JMP     EXEC_SUB                ;DO THE SUBROUTINE
5B47 3706
5B47 3707          ;*****************************************************************************
5B47 3708          ;
5B47 3709          ;  SAVE_WR
5B47 3710          ;  THIS ROUTINE CALLS A WCS SUBROUTINE THAT SAVES THE CONTENTS OF THE
5B47 3711          ;  2901 IN A SPECIAL SAVE AREA SO THEY MAY BE RESTORED LATER.
5B47 3712          ;
5B47 3713          ;*****************************************************************************
5B47 3714
5B47 3715
5B47 3716 02F4'32 3C AF SAVE_WR:   SET     NO_SAVE_UPC_CSR         ;SET FLAG SO SAVED CSR AND
5B4C 3717                                                          ; UPC ARE NOT DESTROYED
5B4C 3718 00 56 21                 LXI     H,WCS_SAVE_WR           ;ADDRESS OF SUBROUTINE
5B4F 3719 0A49'C3                  JMP     EXEC_SUB                ;DO THE SUBROUTINE
5B52 3720
5B52 3721          ;*****************************************************************************
5B52 3722          ;
5B52 3723          ;  CHANGE_WCS
5B52 3724          ;  THIS ROUTINE READS A WCS WORD, THEN CALCULATES PARITY ON IT AND
5B52 3725          ;  WRITES IT BACK.  THIS IS USED FOR SETTING AND CLEARING SOMM
5B52 3726          ;  ADDRESSES AND SETING BAD PARITY
5B52 3727          ;
5B52 3728          ;  INPUT CONDITIONS:
5B52 3729          ;       WCS_BAD_PARITY MUST BE SET OR CLEARED
5B52 3730          ;       BECAUSE IT IS USED IN A CALLED ROUTINE
5B52 3731          ;
5B52 3732          ;       INPUT_NUM MUST HAVE THE ADDRESS TO BE CHANGED
5B52 3733          ;
5B52 3734          ;*****************************************************************************
5B52 3735
5B52 3736
5B52 3737 043F'2A  CHANGE_WCS:     LHLD    INPUT_NUM+2
5B55 3738 006C'22                  SHLD    WCS_ADDRESS             ;PUT ADDRESS IN UPC TO READ
5B58 3739 109D'CD                  CALL    READ_WCS_R              ;FROM WCS
5B5B 3740
5B5B 3741 10B9'C3                  JMP     WRITE_WCS_R             ;WRITE WORD BACK WITH
5B5E 3742                                                          ;PARITY
5B5E 3743
5B5E 3744          ;*****************************************************************************
5B5E 3745          ;
5B5E 3746          ;  CHECK_WCS LOADED
5B5E 3747          ;
5B5E 3748          ;  THIS ROUTINE CHECKS THE WCS LOADED FLAG AND REPORTS AN ERROR IF SET.  IT
5B5E 3749          ;  WILL POP THE STACK IF ERROR IS SET, SO THAT THE RETURN IN THE ERROR ROUTINE
5B5E 3750          ;  WILL SKIP THE REST OF THE ROUTINE THAT HAD THE ERROR.
5B5E 3751          ;
5B5E 3752          ;*****************************************************************************
5B5E 3753
5B5E 3754          CHECK_WCS_LOADED:
```

```
5B5E 3755 0408'3A                       LDA     WCS_LOADED              ;GET WCS LOADED FLAG
5B61 3756 B7                            ORA     A                       ;SET CONDITION CODES
5B62 3757 C0                            RNZ                             ;RETURN IF WCS LOADED
5B63 3758
5B63 3759 02F1'32 AF                    CLEAR   NOADD_FLG               ;ELSE CLEAR FLAG IF SET
5B67 3760 F1                            POP     $PSW                    ;POP THE STACK TO REMOVE
5B68 3761                                                               ; RETURN ADDRESS TO CALLING
5B68 3762                                                               ; ROUTINE
5B68 3763 0F7E'C3                       JMP     ERROR_ROUTINE           ;GO REPORT ERROR BELOW
5B6B 3764
5B6B 3765 ;*****************************************************************************
5B6B 3766 ;
5B6B 3767 ;  CHECK_ERR_FLG
5B6B 3768 ;
5B6B 3769 ;  THIS ROUTINE CHECKS THE ERROR FLAG AND REPORTS AN ERROR IF SET.  IT WILL
5B6B 3770 ;  POP THE STACK IF ERROR IS SET, SO THAT THE RETURN IN THE ERROR ROUTINE
5B6B 3771 ;  WILL SKIP THE REST OF THE ROUTINE THAT HAD THE ERROR.
5B6B 3772 ;
5B6B 3773 ;*****************************************************************************
5B6B 3774
5B6B 3775 INPUT_NUM_CHECK:
5B6B 3776 0FCF'CD                       CALL    INPUT_NUM_IF            ;GET INPUT DATA
5B6E 3777
5B6E 3778 10 06                 CHECK_ERR_10:   MVI     B,<^X10>                ;ERROR CODE IF ERROR SET
5B70 3779                                                               ; (NO INPUT ADDRESS OR DATA)
5B70 3780
5B70 3781 0231'3A               CHECK_ERR_FLG:  LDA     ERROR                   ;GET ERROR FLAG
5B73 3782 B7                            ORA     A                       ;SET CONDITION CODES
5B74 3783 C8                            RZ                              ;RETURN IF NO ERROR
5B75 3784
5B75 3785 F1                            POP     $PSW                    ;POP THE STACK TO REMOVE
5B76 3786                                                               ; RETURN ADDRESS TO CALLING
5B76 3787                                                               ; ROUTINE
5B76 3788 0F7E'C3                       JMP     ERROR_ROUTINE           ;GO REPORT ERROR BELOW
5B79 3789
5B79 3790 ;*****************************************************************************
5B79 3791 ;
5B79 3792 ;  ERROR_ROUTINE
5B79 3793 ;  THIS ROUTINE WILL REPORT AN ERROR AND ITS ERROR NUMBER
5B79 3794 ;
5B79 3795 ;  INPUT CONDITIONS:
5B79 3796 ;       B = ERROR CODE
5B79 3797 ;
5B79 3798 ;*****************************************************************************
5B79 3799
5B79 3800 ERROR_ROUTINE_NO_RET:
5B79 3801 02F3'32 3C AF                 SET     NO_RETURN               ;SET FLAG, JUMP PARSER AFTER
5B7E 3802                                                               ; ERROR REPORTED
5B7E 3803 78                    ERROR_ROUTINE:  MOV     A,B                     ;ERROR CODE FROM B TO AC
5B7F 3804 005B'32                       STA     HEX_BUF                 ;SAVE ERROR CODE
5B82 3805 0013'32                       STA     APT_ERROR_NUMBER+1      ;SAVE IN APT MAILBOX
5B85 3806 01 3E                         MVI     A,1                     ;1 IN ACCUM
5B87 3807 0012'32                       STA     APT_ERROR_NUMBER        ;1 IN HIGH BYTE TO INDICATE
5B8A 3808                                                               ; MONITOR ERROR
5B8A 3809
5B8A 3810 AF                            XRA     A                       ;CLEAR FLAGS
5B8B 3811 0231'32                       STA     ERROR                   ;RESET ERROR CONDITON
5B8E 3812 02F1'32                       STA     NOADD_FLG               ;MAKE SURE EXAM FLAG IS CLEAR
5B91 3813 40FE 32                       STA     CTLDIS                  ;ENABLE CONTROL CHAR CHECK
5B94 3814                                                               ; IF DISABLED
```

```
ZZ-ENKAA-2.1    4C00    4                                   Fiche 1  Frame N7          Sequence 91
5B94  3815 02 3E                    MVI     A,2                     ;INDICATE ERROR
5B96  3816 0011'32                  STA     APT_MESSAGE_CODE+1      ;STORE IN MAILBOX
5B99  3817 02'0B'21                 LXI     H,CRLF_A                ;POINT TO CR
5B9C  3818 1254'CD                  CALL    PRINT_STRING_APT        ;PRINT CRLF
5B9F  3819 3F 3E                    MVI     A,HEX_3F                ;QUESTION MARK
5BA1  3820 1265'CD                  CALL    TYPE_CHAR
5BA4  3821
5BA4  3822 01 0E                    MVI     C,1                     ;PRINT 1 BYTE (2 DIGITS)
5BA6  3823 11C9'CD                  CALL    PRINT_HEX_APT           ;PRINT ERROR CODE IN HEX BUF
5BA9  3824
5BA9  3825 02'32'21                 LXI     H,ERROR_A               ;PRINT '' ERROR''
5BAC  3826 1254'CD                  CALL    PRINT_STRING_APT
5BAF  3827
5BAF  3828 02F3'3A                  LDA     NO_RETURN               ;SEE IF CAN'T RETURN
5BB2  3829 B7                       ORA     A                       ;SET CONDITION CODES
5BB3  3830 C8                       RZ                              ;RETURN IF FLAG CLEAR
5BB4  3831
5BB4  3832 AF                       XRA     A
5BB5  3833 02F3'32                  STA     NO_RETURN               ;ELSE CLEAR FLAG
5BB8  3834 003A'32                  STA     CONTINUE                ;NO CONTINUE ALLOWED
5BBB  3835 40 80 31                 LXI     $SP,PRI_STACK           ;RESET STACK POINTER
5BBE  3836 2022'C3                  JMP     PARSER                  ;GO TO PARSER
5BC1  3837
5BC1  3838
5BC1  3839
5BC1  3840
5BC1  3841
5BC1  3842
5BC1  3843
5BC1  3844
5BC1  3845 00'B8'21                 LXI     H,CSR_VALUE
5BC4  3846 00'5B'11                 LXI     D,HEX_BUF               ;MOVE CSR_VALUE TO HEX NUMBER
5BC7  3847 118E'CD                  CALL    MOVER_3_R               ;PRINT AREA
5BCA  3848
5BCA  3849 03 0E                    MVI     C,3
5BCC  3850 11C4'C3                  JMP     PRINT_HEX_R             ;PRINT 3 BYTES OF CSR
5BCF  3851
5BCF  3852
5BCF  3853
5BCF  3854
5BCF  3855
5BCF  3856
5BCF  3857
5BCF  3858
5BCF  3859
5BCF  3860
5BCF  3861
5BCF  3862
5BCF  3863
5BCF  3864
5BCF  3865
5BCF  3866
5BCF  3867 0231'32 AF
5BD3  3868
5BD3  3869 2D FE 03BB'3A            ******************
           0FE9'C2                  *
5BDB  3870                          *
5BDB  3871 02C9'32 3C AF            *
5BE0  3872 1063'CD                  *
5BE3  3873 1063'CD                  *
```

```
;*****************************************************************************
;
;  PRINT_CSR_VALUE
;  THIS ROUTINE PRINTS THE VALUE IN THE FIELD CSR_VALUE
;
;*****************************************************************************

PRINT_CSR_VAL:  LXI     H,CSR_VALUE
                LXI     D,HEX_BUF               ;MOVE CSR_VALUE TO HEX NUMBER
                CALL    MOVER_3_R               ;PRINT AREA

                MVI     C,3
                JMP     PRINT_HEX_R             ;PRINT 3 BYTES OF CSR

;*****************************************************************************
;
;  INPUT_NUM_IF
;  THIS ROUTINE TAKES A NUMBER FROM THE INPUT LINE
;
;  INPUT CONDITIONS
;       TTBUF_PNT = POINTER TO INPUT STRING
;
;  OUTPUT CONDITIONS
;       MINUS = 1 IF MINUS SIGN FOUND,0 IF NOT
;       ERROR IS SET IF AN ILLEGAL HEX CHAR IS FOUND
;
;*****************************************************************************

INPUT_NUM_IF:   CLEAR   ERROR

                IFEQI   TEMP_BUF,HEX_2D         ;IF MINUS SIGN

                SET     MINUS                   ;MINUS = INFINITE LOOP
                CALL    SKIP_TO_SPEC            ;SKIP - SIGN AND THEN
                CALL    SKIP_TO_SPEC            ;SKIP THE NUMBER
```

```
5BE6 3874                       *
5BE6 3875 102C'C3               ******************
5BE9 3876                       *
5BE9 3877 04'3C'21              *
5BEC 3878 0068'22               *
5BEF 3879 02C9'32 AF            *
5BF3 3880 04 0E 04'3D'21        *
          0D 23 77 AF           *
          0FF9'C2               *
5BFF 3881                       *
5BFF 3882 1063'CD               *
5C02 3883                       *
5C02 3884 0304'2A               *
5C05 3885 03C8'22               *
5C08 3886                       *
5C08 3887 B7                    *                1$:
5C09 3888                       *
5C09 3889 10E7'CD               *
5C0C 3890 10E7'CD               *
5C0F 3891 10E7'CD               *
5C12 3892 10E7'CD               *
5C15 3893                       *
5C15 3894 03C8'2A               *
5C18 3895 7E                    *
5C19 3896 23                    *
5C1A 3897 03C8'22               *
5C1D 3898                       *
5C1D 3899 1045'CD               *
5C20 3900                       *
5C20 3901 04'40'21              *
5C23 3902 B6                    *
5C24 3903 77                    *
5C25 3904                       *
5C25 3905 03'7B'21              *
5C28 3906 35                    *
5C29 3907 1008'C2               *
5C2C 3908                       *
5C2C 3909                       ******************
5C2C 3910
5C2C 3911 0F 0231'3A            ******************
          1044'D2               *
5C33 3912                       *
5C33 3913 0304'2A               *
5C36 3914 03E0'22               *
5C39 3915                       *
5C39 3916 02'FA'21              *
5C3C 3917 03'BB'11              *
5C3F 3918 0A 0E                 *
5C41 3919 11A4'CD               *
5C44 3920                       *
5C44 3921                       ******************
5C44 3922
5C44 3923 C9
5C45 3924
5C45 3925
5C45 3926
5C45 3927
5C45 3928
5C45 3929
5C45 3930
```

```
ELSE

LXI     H,NUM_SHIFT             ;SET UP SHIFT POINTER
SHLD    SHIFT_PNT               ;TO NUMBER BUFFER
CLEAR   MINUS                   ;NO MINUS SIGN FOUND
ZERO    INPUT_NUM,4             ;CLEAR INPUT NUMBER AREA


CALL    SKIP_TO_SPEC            ;SPAN AND GET COUNT

LHLD    OLD_TTBUF
SHLD    TEMP_WORD1

ORA     A                       ;CLEAR THE CARRY SO SHIFT IN
                                ;WILL BE ZERO'S
CALL    SHIFTER_R               ;SHIFT NUMBER BUFF LEFT NIBBLE
CALL    SHIFTER_R
CALL    SHIFTER_R
CALL    SHIFTER_R

LHLD    TEMP_WORD1
MOV     A,M                     ;GET A CHAR AND BUMP POINTER
INX     H
SHLD    TEMP_WORD1

CALL    ASCII_TO_BIN            ;CONVERT TO BINARY

LXI     H,INPUT_NUM+3           ;OR BINARY VALUE AND NIBBLE
ORA     M
MOV     M,A

LXI     H,SKIP_CNT
DCR     M                       ;DEC CHAR COUNT
JNZ     1$                      ;END?

ENDIF

IF      ERROR

LHLD    OLD_TTBUF               ;NO NUMBERS FOUND MOVE POINTER
SHLD    TTBUF_PNT               ;BACK TO WHERE IT WAS ON ENTRY

LXI     H,OLD_TEMP_BUF
LXI     D,TEMP_BUF
MVI     C,10
CALL    MOVER_R

ENDIF

RET

;*****************************************************************************
;
;  ASCII_TO_BIN
;  THIS ROUTINE CONVERTS AN ASCII CHAR TO A BINARY NUMBER
;  IF THE CHAR IS OUT OF RANGE ERROR IS SET
```

```
5C45 3931
5C45 3932
5C45 3933
5C45 3934 30 FE
5C47 3935 104F'F2
5C4A 3936 0231'32 3C AF
5C4F 3937 30 D6
5C51 3938 0A FE
5C53 3939 1062'FA
5C56 3940 07 D6
5C58 3941 10 FE
5C5A 3942 1062'FA
5C5D 3943 0231'32 3C AF
5C62 3944 C9
5C63 3945
5C63 3946
5C63 3947
5C63 3948
5C63 3949
5C63 3950
5C63 3951
5C63 3952
5C63 3953
5C63 3954
5C63 3955
5C63 3956
5C63 3957
5C63 3958
5C63 3959
5C63 3960
5C63 3961
5C63 3962
5C63 3963
5C63 3964
5C63 3965
5C63 3966
5C63 3967
5C63 3968 037B'32 AF
5C67 3969
5C67 3970 03E0'2A
5C6A 3971 0304'22
5C6D 3972
5C6D 3973 03'BB'21
5C70 3974 02'FA'11
5C73 3975 0A 0E
5C75 3976 11A4'CD
5C78 3977
5C78 3978 03E0'2A
5C7B 3979 7E
5C7C 3980 30 FE
5C7E 3981
5C7E 3982 23
5C7F 3983 03E0'22
5C82 3984
5C82 3985
5C82 3986 1091'FA          ******************
5C85 3987 3A FE             *
5C87 3988 1091'CA           *
5C8A 3989 03'7B'21          *
5C8D 3990 34                *
```

```
;*****************************************************************************

ASCII_TO_BIN:   CPI     <^X30>
                JP      1$
                SET     ERROR                   ;CHAR BELOW RANGE
1$:             SUI     <^X30>
                CPI     <^XA>
                JM      3$
                SUI     7
                CPI     <^X10>
                JM      3$
                SET     ERROR                   ;CHAR ABOVE RANGE
3$:             RET

;*****************************************************************************
;
;  SKIP_TO_SPEC
;  THIS ROUTINE SKIPS TO THE NEXT SPECIAL CHAR IN THE INPUT STRING
;  THEN MOVES ONE MORE.(WHICH SHOULD BE THE BEGINING OF THE NEXT WORD.)
;  IT ALSO LEAVES A COUNT IN SKIP_CNT FOR THE NUMBER OF CHARS SPANNED
;  AT END IT MOVES 10 CHARS FROM THE NOW POINTER INTO TEMP_BUF
;
;  INPUT CONDITIONS:
;       TTBUF_PNT          = POINTS TO CHARS IN THE INPUT LINE
;
;  OUTPUT CONDITIONS:
;       SKIP_CNT           = # CHARS SPANNED  (SEE NOTE)
;       TEMP_BUF           = 10 CHARS FROM POINTER
;       TTBUF_PNT          = NEW POINTER
;       OLD_TTBUF          = CHAR POINTER ON ENTRY (FOR RESET WHEN NO # FOUND)
;
;  NOTE:  SPANNED CHARS COUNT DOES NOT INCLUDE THE SPECIAL CHAR !!!
;
;*****************************************************************************

SKIP_TO_SPEC:   CLEAR   SKIP_CNT

                LHLD    TTBUF_PNT               ;POINTER TO INPUT CHARS
                SHLD    OLD_TTBUF               ;SAVE OLD POINTER FOR ERRORS

                LXI     H,TEMP_BUF              ;SAVE OLD BUFFER FOR ERRORS
                LXI     D,OLD_TEMP_BUF
                MVI     C,10
                CALL    MOVER_R

1$:             LHLD    TTBUF_PNT
                MOV     A,M
                CPI     HEX_30                  ;SPEC CHAR ARE <30 HEX

                INX     H                       ;MOVE TO NEXT INPUT CHAR
                SHLD    TTBUF_PNT               ;STORE NEW POINTER.  NOTE:
                                                ; THESE 2 INSTRUCTIONS DO
                                                ; NOT AFFECT CONDITION CODES
                IFPOS                           ;IF NOT SPECIAL CHAR
                CPI     HEX_3A                  ;SEE IF : (ALSO SPECIAL CHAR)
                JZ      2$                      ;JUMP IF :
                LXI     H,SKIP_CNT
                INR     M                       ;INCREMENT SPANNED CHAR CNT
```

```
5C8E 3991                       *
5C8E 3992 109A'C3               ******************              ELSE                                    ;IF SPECIAL CHARACTER
5C91 3993 0A 0E                 *                       2$:     MVI     C,10
5C93 3994 03'BB'11              *                               LXI     D,TEMP_BUF
5C96 3995 11A4'CD               *                               CALL    MOVER_R                         ;PUT 10 CHARS INTO TEMP_BUF
5C99 3996 C9                    *                               RET                                     ;GET OUT OF LOOP
5C9A 3997                       *
5C9A 3998                       ******************              ENDIF
5C9A 3999
5C9A 4000 1078'C3                                               JMP     1$                              ;LOOP UNTIL SPECIAL CHAR
5C9D 4001
5C9D 4002
5C9D 4003                       ;*************************************************************************************
5C9D 4004                       ;
5C9D 4005                       ;  READ_WCS_R
5C9D 4006                       ;  THIS ROUTINE READS 3 BYTES OF DATA FROM THE WCS TO CSR_VALUE USING
5C9D 4007                       ;  THE ADDRESS IN WCS_ADDRESS. ALSO A NOP IS PLACED IN THE CSR TO PREVENT
5C9D 4008                       ;  THE MACHINE FROM DOING NASTY THINGS
5C9D 4009                       ;
5C9D 4010                       ;  INPUT CONDITIONS:
5C9D 4011                       ;       WCS_ADDRESS HAS WCS ADDRESS TO READ
5C9D 4012                       ;
5C9D 4013                       ;  OUTPUT CONDITIONS:
5C9D 4014                       ;       CSR_VALUE AND WCS_VALUE CONTAIN THE VALUE OF THE WORD READ
5C9D 4015                       ;
5C9D 4016                       ;*************************************************************************************
5C9D 4017
5C9D 4018 006C'2A               READ_WCS_R:     LHLD    WCS_ADDRESS                     ;GET ADDRESS TO READ
5CA0 4019 00C2'22                               SHLD    UPC_VALUE
5CA3 4020 1148'CD                               CALL    LOAD_UPC_R
5CA6 4021
5CA6 4022 10F9'CD                               CALL    NOP_CSR_R
5CA9 4023
5CA9 4024 23 D3                                 OUT     SETSS                           ;SING STEP MACHINE TO GET
5CAB 4025 22 D3                                 OUT     CLRSS                           ;WCS VALUE INTO CSR
5CAD 4026
5CAD 4027 1116'CD                               CALL    LOAD_CSR_R                      ;READ IT OUT
5CB0 4028
5CB0 4029 00'B8'21                              LXI     H,CSR_VALUE
5CB3 4030 00'B4'11                              LXI     D,WCS_VALUE
5CB6 4031 118E'C3                               JMP     MOVER_3_R
5CB9 4032
5CB9 4033                       ;*************************************************************************************
5CB9 4034                       ;
5CB9 4035                       ;  WRITE_WCS_R
5CB9 4036                       ;  THIS ROUTINE WRITES 3 BYTES OF DATA TO THE WCS FROM WCS_VALUE
5CB9 4037                       ;  BEFORE THE WRITE OCCURES PARITY IS CALCULATED ON THAT 3 BYTE WCS WORD
5CB9 4038                       ;  AND PLACED IN BIT 23.
5CB9 4039                       ;
5CB9 4040                       ;  INPUT CONDITIONS:
5CB9 4041                       ;       WCS_ADDRESS HAS WCS ADDRESS TO READ
5CB9 4042                       ;       WCS_BAD_PARITY = 0 FOR GOOD PARITY, 1 FOR BAD PARITY
5CB9 4043                       ;       PARITY_ON = 0 FOR NO PARITY CALC, 1 TO CALCULATE PARITY
5CB9 4044                       ;
5CB9 4045                       ;*************************************************************************************
5CB9 4046
5CB9 4047 006C'2A               WRITE_WCS_R:    LHLD    WCS_ADDRESS                     ;GET ADDRESS TO WRITE
5CBC 4048 00C2'22                               SHLD    UPC_VALUE
5CBF 4049 1148'CD                               CALL    LOAD_UPC_R
5CC2 4050
```

```
ZZ-ENKAA-2.1    4C00    4
5CC2 4051 10F9'CD                   CALL    NOP_CSR_R               ;KEEP MACHINE IDLE.
5CC5 4052
5CC5 4053 0407'32 3C AF             SET     WCS_CALC                ;PARITY CALC ON WCS
5CCA 4054 00'B4'21                  LXI     H,WCS_VALUE             ;CALC PARITY ON WCS WORD
5CCD 4055 0EFF'CD                   CALL    PARITY_CALC
5CD0 4056 0407'32 AF                CLEAR   WCS_CALC                ;CLEAR FLAG
5CD4 4057
5CD4 4058 00'B6'21                  LXI     H,WCS_VALUE+2
5CD7 4059 7E                        MOV     A,M                     ;GET LOW BYTE
5CD8 4060
5CD8 4061 08 D3                     OUT     WCSB0                   ;CLOCK INTO BYTE 0 OF LATCH
5CDA 4062 2B                        DCX     H                       ;INC POINTER TO NEXT BYTE
5CDB 4063
5CDB 4064 7E                        MOV     A,M                     ;GET MIDDLE BYTE
5CDC 4065 09 D3                     OUT     WCSB1                   ;CLOCK INTO BYTE 1 OF LATCH
5CDE 4066 2B                        DCX     H                       ;INC POINTER TO NEXT BYTE
5CDF 4067
5CDF 4068 7E                        MOV     A,M                     ;GET HIGH BYTE
5CE0 4069 0A D3                     OUT     WCSB2                   ;CLOCK INTO BYTE 2 OF LATCH
5CE2 4070
5CE2 4071 37 D3                     OUT     SETWCK                  ;CLOCK 24 BITS INTO WCS
5CE4 4072 36 D3                     OUT     CLRWCK                  ;CLEAR WCS CLOCK
5CE6 4073 C9                        RET
5CE7 4074
5CE7 4075
5CE7 4076
5CE7 4077
5CE7 4078
5CE7 4079
5CE7 4080
5CE7 4081
5CE7 4082
5CE7 4083
5CE7 4084
5CE7 4085
5CE7 4086
5CE7 4087
5CE7 4088
5CE7 4089
5CE7 4090
5CE7 4091 F5                        SHIFTER_R:      PUSH    $PSW            ;SAVE CARRY
5CE8 4092 0068'2A                                   LHLD    SHIFT_PNT       ;POINTER TO SHIFT DATA
5CEB 4093 4E                                        MOV     C,M             ;GET COUNT IN BYTES
5CEC 4094 00 06                                     MVI     B,0             ;MAKE COUNT IN REG PAIR
5CEE 4095 09                                        DAD     B               ;ADD COUNT , GOTO LAST BYTE
5CEF 4096 F1                                        POP     $PSW
5CF0 4097
5CF0 4098 7E                        SHIFTER1_R:     MOV     A,M             ;MOVE BYTE 1 LEFT
5CF1 4099 17                                        RAL                     ;CARRY = ROTATE IN BIT NEXT BTYE
5CF2 4100 77                                        MOV     M,A
5CF3 4101 0D                                        DCR     C               ;DEC BYTE COUNT
5CF4 4102 C8                                        RZ                      ;RETURN IF ZERO
5CF5 4103 2B                                        DCX     H               ;MOVE POINTER TO NEXT BYTE
5CF6 4104
5CF6 4105 10F0'C3                                   JMP     SHIFTER1_R      ;REPEAT UNTIL BYTE COUNT=0
5CF9 4106
5CF9 4107
5CF9 4108
5CF9 4109
5CF9 4110
```

```
;*****************************************************************************
;
;  SHIFTER_R
;  THIS ROUTINE SHIFTS N BYTES OF DATA LEFT 1 BIT AT THE ADDRESS POINTED
;  TO BY  SHIFT_PNT.
;
;  INPUT CONDITIONS:
;       SHIFT_PNT = POINTS TO COUNT OF NUMBER OF BYTES TO SHIFT
;                   DATA TO BE SHIFTED FOLLOWS IN NEXT N BYTES
;
;  CARRY BIT GETS MSB, CARRY MAY ALSO BE USED AS
;  INPUT TO THE SHIFT
;
;*****************************************************************************

;*****************************************************************************
;
;  NOP_CSR_R:
;  THIS ROUTINE LOAD A NOP INTO THE CSR
```

```
5CF9 4111
5CF9 4112                ;*************************************************************************
5CF9 4113
5CF9 4114 00'72'21       NOP_CSR_R:      LXI     H,NOP_INST
5CFC 4115 00'B8'11               LXI     D,CSR_VALUE
5CFF 4116 118E'CD                CALL    MOVER3_R                ;MOVE NOP TO CSR SAVE AREA
5D02 4117 0064'3A                LDA     PARITY_ON               ;SAVE PARITY ON FLAG
5D05 4118 03C5'32                STA     TEMP_BYTE               ;STORE IN TEMP BYTE
5D08 4119 0064'32 AF             CLEAR   PARITY_ON               ;CALC NO PARITY ON NOP
5D0C 4120
5D0C 4121 1116'CD                CALL    LOAD_CSR_R              ;LOAD IT
5D0F 4122
5D0F 4123 03C5'3A                LDA     TEMP_BYTE               ;GET OLD PARITY_ON FLAG
5D12 4124 0064'32                STA     PARITY_ON               ;RESTORE FLAG
5D15 4125
5D15 4126 C9                     RET
5D16 4127
5D16 4128
5D16 4129                ;*************************************************************************
5D16 4130                ;
5D16 4131                ;  LOAD_CSR_R
5D16 4132                ;  THIS ROUTINE LOADS THE CSR FROM THE CSR_VALUE AREA,
5D16 4133                ;  ALSO CSR_VALUE AREA GETS THE VALUE OF THE CSR
5D16 4134                ;
5D16 4135                ;*************************************************************************
5D16 4136
5D16 4137 A2 D3          LOAD_CSR_R:     OUT     VSHIFT                  ;SET V-BUS SHIFT MODE
5D18 4138
5D18 4139 00'B8'21               LXI     H,CSR_VALUE             ;CALC PARITY ON NEW CSR VALUE
5D1B 4140 0EFF'CD                CALL    PARITY_CALC
5D1E 4141
5D1E 4142 C5 46 00'55'21 *****************       DO      24                      ;SHIFT 24 BITS
          77 18 3E       *
5D26 4143                *
5D26 4144 87 DB          *                      IN      CSR23                   ;GET OLD CSR DATA
5D28 4145 1F             *                      RAR                             ;PUT CSR 23 IN CARRY
5D29 4146 00'BA'21       *                      LXI     H,CSR_VALUE+2           ;POINTER TO LAST CSR DATA BYTE
5D2C 4147 03 0E          *                      MVI     C,3                     ;BYTES TO SHIFT
5D2E 4148 10F0'CD        *                      CALL    SHIFTER1_R              ;CARRY = MSB OF NEW CSR DATA ON
5D31 4149                *                                                      ; RETURN
5D31 4150                *
5D31 4151 38 D3          *                      OUT     CLRCSR                  ;CLEAR CSR INPUT
5D33 4152 1138'D2        *                      JNC     1$                      ;DON'T SET INPUT IF CARRY CLEAR
5D36 4153 39 D3          *                      OUT     SETCSR                  ;ELSE SET CSR INPUT
5D38 4154                *
5D38 4155 25 D3          *              1$:     OUT     SETCSK                  ;SET UPC CLOCK
5D3A 4156 24 D3          *                      OUT     CLRCSK                  ;CLEAR UPC CLOCK
5D3C 4157                *
5D3C 4158 35 00'55'21    *****************       ENDDO
          70 C1 1126'C2
5D45 4159
5D45 4160 A3 D3                          OUT     VNORML                  ;SET V-BUS TO NORMAL MODE
5D47 4161
5D47 4162 C9                             RET
5D48 4163
5D48 4164
5D48 4165                ;*************************************************************************
5D48 4166                ;
5D48 4167                ;  LOAD_UPC_R
5D48 4168                ;  THIS ROUTINE LOADS THE UPC FROM THE UPC_VALUE AREA,
```

```
5D48 4169                                ;  ALSO UPC_VALUE AREA GETS THE VALUE OF THE UPC
5D48 4170                                ;
5D48 4171                                ;*****************************************************************************
5D48 4172
5D48 4173 A2 D3                          LOAD_UPC_R:     OUT     VSHIFT                  ;SET V-BUS SHIFT MODE
5D4A 4174
5D4A 4175 00'C1'21                               LXI     H,UPC_SHIFT             ;POINTER TO UPC DATA BUFF
5D4D 4176 0068'22                                SHLD    SHIFT_PNT
5D50 4177
5D50 4178 B7                                     ORA     A                       ;CLEAR CARRY
5D51 4179
5D51 4180 10E7'CD                                CALL    SHIFTER_R               ;LEFT JUST. 15 BITS
5D54 4181
5D54 4182 C5 46 00'55'21       ******************        DO      15                      ;SHIFT 15 BITS
          77 0F 3E             *
5D5C 4183                      *
5D5C 4184 84 DB                *                         IN      UPC14
5D5E 4185 1F                   *                         RAR                     ;PUT UPC14 IN CARRY
5D5F 4186 10E7'CD              *                         CALL    SHIFTER_R       ;CARRY = MSB ON RET
5D62 4187                      *
5D62 4188 A8 D3                *                         OUT     CLRUPC          ;SET = 0 UPC SER IN
5D64 4189 1169'D2              * ****************        IFC
5D67 4190 A9 D3                * *                       OUT     SETUPC          ;SET = 1 UPC SER IN
5D69 4191                      * ****************        ENDIF
5D69 4192                      *
5D69 4193 27 D3                *                         OUT     SETUPK          ;SET UPC CLOCK
5D6B 4194 26 D3                *                         OUT     CLRUPK          ;CLEAR UPC CLOCK
5D6D 4195                      *
5D6D 4196 35 00'55'21          ******************        ENDDO
          70 C1 115C'C2
5D76 4197
5D76 4198 A3 D3                                          OUT     VNORML                  ;SET V-BUS TO NORMAL MODE
5D78 4199
5D78 4200 A9 D3                                          OUT     CLRACK                  ;CLEAR CONSOLE ACKNOWLEDGE
5D7A 4201 0F 0344'3A           ******************        IF      SETACK_FLG              ;MUST SET UP CONSOLE ACKNOWLEDE
          1183'D2              *
5D81 4202                      *                                                         ;BEFORE RETURNING
5D81 4203 A8 D3                *                         OUT     SETACK                  ;SET CONSOLE ACKNOWLEDGE IF
5D83 4204                      *                                                         ; FLAG SET
5D83 4205                      ******************        ENDIF
5D83 4206
5D83 4207 C9                                             RET
5D84 4208
5D84 4209
5D84 4210                                ;*****************************************************************************
5D84 4211                                ;
5D84 4212                                ;  COMPARE_R
5D84 4213                                ;  THIS ROUTINE IS USED BY THE STRUCTURED MACRO SET AND MAY ALSO
5D84 4214                                ;  BE CALLED DIRECTLY.
5D84 4215                                ;
5D84 4216                                ;  INPUT CONDITIONS:
5D84 4217                                ;       H + L   = POINTER TO DATA FIELD 1
5D84 4218                                ;       D + E   = POINTER TO DATA FIELD 2
5D84 4219                                ;       C       = COUNT IN BYTES FOR COMPARISON
5D84 4220                                ;
5D84 4221                                ;  OUTPUT CONDITIONS:
5D84 4222                                ;       CONDITIONS CODES ARE SET
5D84 4223                                ;
5D84 4224                                ;*****************************************************************************
5D84 4225
```

```
5D84 4226 1A                COMPARE_R:  LDAX    D               ;FIELD 2 BYTE
5D85 4227 BE                            CMP     M               ;COMPARED TO FIELD 1 BYTE
5D86 4228 C0                            RNZ                     ;RETURN WITH CC NON ZERO IF NOT SAME
5D87 4229 23                            INX     H               ;INCREMENT FIELD 1 POINTER
5D88 4230 13                            INX     D               ;INCREMENT FIELD 2 POINTER
5D89 4231 0D                            DCR     C               ;DECREMENT COUNT
5D8A 4232 1184'C2                       JNZ     COMPARE_R       ;LOOP UNTIL COUNT GONE
5D8D 4233 C9                            RET                     ;DONE - RETURN ZERO CC
5D8E 4234
5D8E 4235
5D8E 4236                   ;*****************************************************************************
5D8E 4237                   ;
5D8E 4238                   ;  MOVER_R
5D8E 4239                   ;
5D8E 4240                   ;  THIS ROUTINE MOVES DATA FROM THE ADDRESS GIVEN IN H+L TO THE ADDRESS
5D8E 4241                   ;  GIVEN IN D+E FOR THE COUNT IN C
5D8E 4242                   ;
5D8E 4243                   ;  INPUT CONDITIONS:
5D8E 4244                   ;       H+L     = SOURCE ADDRESS
5D8E 4245                   ;       D+E     = DEST. ADDRESS
5D8E 4246                   ;       C       = LENGTH
5D8E 4247                   ;
5D8E 4248                   ;  OUTPUT CONDITIONS:
5D8E 4249                   ;       REGISTERS ARE INCREMENTED DURING THE MOVE, SO THEY
5D8E 4250                   ;       ARE POINTING TO THE LOCATION AFTER THE DATA
5D8E 4251                   ;
5D8E 4252                   ;*****************************************************************************
5D8E 4253
5D8E 4254 03 0E             MOVER_3_R:  MVI     C,3                     ;MOVE 3 TO COUNT
5D90 4255 11A4'C3                       JMP     MOVER_R                 ;GO MOVE THREE BYTES
5D93 4256
5D93 4257                   MOV_INPUT_DATA0:
5D93 4258 04'3D'21                      LXI     H,INPUT_NUM             ;SOURCE IN INPUT_NUM
5D96 4259 00'A9'11                      LXI     D,DATA0                 ;DEST IS DATA0-DATA3
5D99 4260 11A2'C3                       JMP     MOVER_4_R               ;MOVE 4 BYTES
5D9C 4261
5D9C 4262                   MOV_DATA0_HEXBUF:
5D9C 4263 00'5B'11                      LXI     D,HEX_BUF               ;DESTINATION IS HEX_BUF
5D9F 4264 00'A9'21          MOV_FROM_DATA0: LXI     H,DATA0             ;SOURCE IS DATA0-3
5DA2 4265
5DA2 4266 04 0E             MOVER_4_R:  MVI     C,4                     ;MOVE 4 TO COUNT
5DA4 4267 7E                MOVER_R:    MOV     A,M                     ;GET SOURCE DATA
5DA5 4268 12                            STAX    D                       ;PUT AT DESTINATION
5DA6 4269 23                            INX     H
5DA7 4270 13                            INX     D                       ;INCREMENT POINTERS
5DA8 4271 0D                            DCR     C                       ;DECREMENT COUNT
5DA9 4272 C8                            RZ                              ;RETURN IF DONE
5DAA 4273 11A4'C3                       JMP     MOVER_R                 ;LOOP OTHERWISE
5DAD 4274
5DAD 4275
5DAD 4276
5DAD 4277                   ;*****************************************************************************
5DAD 4278                   ;
5DAD 4279                   ;  PRINT_FLG
5DAD 4280                   ;  THIS ROUTINE PRINTS 5 CHARS AND A SPACE FROM THE POINTER TEMP_WORD
5DAD 4281                   ;
5DAD 4282                   ;*****************************************************************************
5DAD 4283
5DAD 4284 03C6'2A           PRINT_FLG:  LHLD    TEMP_WORD               ;POINTER TO FLAG TO PRINT
5DB0 4285 05 0E                         MVI     C,5
```

```
5DB2 4286 1243'CD                       CALL    PRINTER                 ;PRINT THE 5 CHAR
5DB5 4287 1260'C3                       JMP     SPACE_R                 ;AND THE SPACE
5DB8 4288
5DB8 4289
5DB8 4290
5DB8 4291
5DB8 4292
5DB8 4293
5DB8 4294
5DB8 4295
5DB8 4296
5DB8 4297
5DB8 4298
5DB8 4299
5DB8 430C
5DB8 4301 04 0E                 PRINT_HEX_4:    MVI     C,4                     ;LOAD BYTE COUNT WITH 4
5DBA 4302 11C4'C3                               JMP     PRINT_HEX_R             ;PRINT FOUR CHARS
5DBD 4303
5DBD 4304 02 0E                 PRINT_HEX_2:    MVI     C,2                     ;BYTE COUNT OF 2
5DBF 4305 11C4'C3                               JMP     PRINT_HEX_R             ;PRINT 2 CHARS
5DC2 4306
5DC2 4307 01 0E                 PRINT_HEX_1_R:  MVI     C,1                     ;PRINT 1 CHAR
5DC4 4308 0038'3A               PRINT_HEX_R:    LDA     APT                     ;SEE IF APT PRESENT
5DC7 4309 B7                                    ORA     A                       ;SET CONDITION CODES
5DC8 4310 C0                                    RNZ                             ;DON'T PRINT IF APT
5DC9 4311 00'5B'21              PRINT_HEX_APT:  LXI     H,HEX_BUF
5DCC 4312 0321'22                               SHLD    PRINT_PNT               ;POINTER TO PRINT DATA
5DCF 4313 79                                    MOV     A,C
5DD0 4314 0320'32                               STA     PRINT_CNT               ;PRINT COUNT
5DD3 4315
5DD3 4316 0321'2A               1$:             LHLD    PRINT_PNT
5DD6 4317 7E                                    MOV     A,M                     ;GET FIRST BYTE
5DD7 4318 1F                                    RAR
5DD8 4319 1F                                    RAR
5DD9 4320 1F                                    RAR
5DDA 4321 1F                                    RAR
5DDB 4322 0F E6                                 ANI     <^XF>                   ;GET FIRST NIBBLE
5DDD 4323 11FC'CD                               CALL    PR_HEX_DIGIT            ;PRINT HEX DIGIT
5DE0 4324
5DE0 4325 0321'2A                               LHLD    PRINT_PNT
5DE3 4326 7E                                    MOV     A,M
5DE4 4327 23                                    INX     H
5DE5 4328 0321'22                               SHLD    PRINT_PNT
5DE8 4329 0F E6                                 ANI     <^XF>                   ;GET SECOND NIBBLE
5DEA 4330 11FC'CD                               CALL    PR_HEX_DIGIT
5DED 4331
5DED 4332 03'20'21                              LXI     H,PRINT_CNT             ;DECR PRINT COUNT
5DF0 4333 35                                    DCR     M
5DF1 4334 11D3'C2                               JNZ     1$                      ;RETURN IF DONE
5DF4 4335
5DF4 4336 02F5'3A                               LDA     NO_SPACE                ;GET FLAG
5DF7 4337 B7                                    ORA     A                       ;SET CONDITION CODES
5DF8 4338 1260'CC                               CZ      SPACE_R                 ;PRINT SPACE IF NO FLAG
5DFB 4339 C9                                    RET
5DFC 4340
5DFC 4341
5DFC 4342
5DFC 4343
5DFC 4344
5DFC 4345
```

```
;*****************************************************************************
;
;  PRINT_HEX_R
;  THIS ROUTINE PRINTS HEX NUMBERS
;
;  INPUT CONDITIONS:
;       C = NUMBER OF BYTES TO BE PRINTED
;       DATA IS IN HEX_BUF
;
;*****************************************************************************

;*****************************************************************************
;
;  PR_HEX_DIGIT
;  THIS ROUTINE PRINTS A HEX DIGIT
```

```
5DFC 4346
5DFC 4347
5DFC 4348
5DFC 4349
5DFC 4350 30 C6
5DFE 4351 3A FE
5E00 4352 1205'FA
5E03 4353 07 C6
5E05 4354 1265'C3
5E08 4355
5E08 4356
5E08 4357
5E08 4358
5E08 4359
5E08 4360
5E08 4361
5E08 4362
5E08 4363
5E08 4364
5E08 4365
5E08 4366
5E08 4367
5E08 4368 AF
5E09 4369 40BB 32
5E0C 4370
5E0C 4371 0057'32
5E0F 4372 0333'3A
5E12 4373 B7
5E13 4374 C0
5E14 4375
5E14 4376 04 DB
5E16 4377 2F
5E17 4378 47
5E18 4379 06 DB
5E1A 4380 2F
5E1B 4381 A0
5E1C 4382 01 E6
5E1E 4383 0038'32
5E21 4384
5E21 4385 1ED7'CD
5E24 4386
5E24 4387 02'F0'21          ******************
          01 0E 03'06'11     *
          1237'CA 00CF'CD    *
5E32 4388 27 06              *
5E34 4389 0F7E'CD            *
5E37 4390                    ******************
5E37 4391
5E37 4392 0208'32 AF
5E3B 4393 03'23'21
5E3E 4394 1254'CD
5E41 4395
5E41 4396 FB
5E42 4397
5E42 4398 C9
5E43 4399
5E43 4400
5E43 4401
5E43 4402
5E43 4403
```

```
;  THE HEX NUMBER IS IN THE AC.
;
;*******************************************************************************

PR_HEX_DIGIT:   ADI     HEX_30                  ;ADD FOR 0-9 TO ASCII
                CPI     HEX_3A                  ;GREATER THAN 9 ?
                JM      1$
                ADI     HEX_7                   ;ADD FOR A-F
1$:             JMP     TYPE_CHAR               ;PRINT ASCII CHAR

;*******************************************************************************
;
;  PRINT_PROMPT
;
;  THIS ROUTINE PRINTS THE PROMPT ON THE CONSOLE PORT
;  THIS ROUTINE ALSO CALCULATES A CHECKSUM AND COMPARES IT TO THE CHECKSUM
;  CALCULATED WHEN THE PROGRAM WAS STARTED. IF THEY ARE NOT THE SAME
;  AN ERROR IS PRINTED
;
;*******************************************************************************

PRINT_PROMPT:   XRA     A                       ;CLEAR FLAGS
                STA     OFLAG                   ;CLEAR CONTROL O FLAG IN BOOT
                                                ; BLOCK
                STA     ERROR_CON               ;CLEAR 8085 ERROR LOOP FLAG
                LDA     REPEAT                  ;GET REPEAT FLAG
                ORA     A                       ;SET CONDITION CODES
                RNZ                             ;DONE IF NOT ZERO

                IN      APTFLG                  ;READ APT PRESENT FLAG
                CMA                             ;INVERT VALUE (HIGH=SET NOW)
                MOV     B,A                     ;STORE IN B REG
                IN      REMOTE                  ;GET REMOTE FLAG
                CMA                             ;INVERT VALUE
                ANA     B                       ;AND TWO FLAGS
                ANI     1                       ;MASK ALL BUT BIT 0
                STA     APT                     ;SAVE APT PRESENT FLAG

                CALL    CHECKSUM                ;CALC CHECKSUM

                IFNEQ   NEWSUM,OLDSUM,1

                MVI     B,<^X27>
                CALL    ERROR_ROUTINE           ;CHECKSUM ERROR IN MICMON
                ENDIF

                CLEAR   CNT_PAR                 ;CLEAR PARITY ERROR COUNT
                LXI     H,PROMPT_A              ;PROMPT STRING
                CALL    PRINT_STRING_APT

                EI                              ;ENABLE INTERUPTS

                RET

;*******************************************************************************
;
;  PRINTER
```

```
5E43 4404
5E43 4405
5E43 4406
5E43 4407
5E43 4408
5E43 4409
5E43 4410
5E43 4411 79                    PRINTER:        MOV     A,C                     ;LD A WITH CHAR COUNT
5E44 4412 4133 CD                               CALL    SLVEC1                  ;CALL SEND LINE ROUTINE IN BOOT
5E47 4413                                                                       ; BLOCK.
5E47 4414 1341'C3                               JMP     CHECK_CNTLC             ;IF ^C TYPED, GO TO PARSER
5E4A 4415
5E4A 4416
5E4A 4417
5E4A 4418
5E4A 4419
5E4A 4420
5E4A 4421
5E4A 4422
5E4A 4423
5E4A 4424
5E4A 4425
5E4A 4426
5E4A 4427 0038'3A               PRINT_STRING_R: LDA     APT                     ;GET APT PRESENT FLAG
5E4D 4428 B7                                    ORA     A                       ;SET CONDITION CODES
5E4E 4429 412E CC                               CZ      SLVECT                  ;CALL SEND LINE ROUTINE IN BOOT
5E51 4430                                                                       ; BLOCK, IF APT NOT PRESENT,
5E51 4431                                                                       ; THAT EXPECTS CHAR COUNT
5E51 4432                                                                       ; AS FIRST CHAR IN STRING
5E51 4433 1341'C3                               JMP     CHECK_CNTLC             ;IF ^C TYPED, GO TO PARSER
5E54 4434
5E54 4435
5E54 4436
5E54 4437
5E54 4438
5E54 4439
5E54 4440
5E54 4441
5E54 4442
5E54 4443
5E54 4444
5E54 4445
5E54 4446
5E54 4447 412E CD               PRINT_STRING_APT:
                                                CALL    SLVECT                  ;CALL SEND LINE ROUTINE IN BOOT
5E57 4448                                                                       ; BLOCK THAT EXPECTS CHAR COUNT
5E57 4449                                                                       ; AS FIRST CHAR IN STRING
5E57 4450 1341'C3                               JMP     CHECK_CNTLC             ;IF ^C TYPED, GO TO PARSER
5E5A 4451
5E5A 4452
5E5A 4453
5E5A 4454
5E5A 4455
5E5A 4456
5E5A 4457
5E5A 4458
5E5A 4459
5E5A 4460
5E5A 4461 02'0B'21              CRLF_R:         LXI     H,CRLF_A                ;POINT TO CR
5E5D 4462 124A'C3                               JMP     PRINT_STRING_R          ;PRINT IT
5E60 4463
```

```
5E60 4464
5E60 4465
5E60 4466
5E60 4467
5E60 4468
5E60 4469
5E60 4470
5E60 4471 20 3E                  ;*****************************************************************************
5E60 4472                        ;
5E60 4473                        ;  SPACE_R
5E60 4474                        ;  THIS ROUTINE PRINTS A SPACE
5E60 4475                        ;
5E60 4476                        ;*****************************************************************************
5E60 4477
5E60 4478 1265'C3        SPACE_R:        MVI     A,SPACE_A               ;PRINT SPACE
5E62 4479                                JMP     TYPE_CHAR
5E65 4480
5E65 4481                        ;*****************************************************************************
5E65 4482                        ;
5E65 4483                        ;  TYPE_CHAR
5E65 4484                        ;  THIS ROUTINE LOADS THE ACCUMULATOR WITH AN ASCII CHARACTER AND CALLS THE
5E65 4485                        ;  SEND CHAR ROUTINE IN THE BOOT BLOCK.
5E65 4486                        ;
5E65 4487                        ;*****************************************************************************
5E65 4488
5E65 4489                        TYPE_CHAR:
5E65 4490 4138 CD        PRINT_R:        CALL    SCVECT                  ;CALL SEND CHAR ROUTINE IN
5E68 4491                                                                ; BOOT BLOCK
5E68 4492
5E68 4493 1341'C3                        JMP     CHECK_CNTLC             ;IF ^C TYPED, GO TO PARSER
5E6B 4494
5E6B 4495                        ;*****************************************************************************
5E6B 4496                        ;
5E6B 4497                        ;  INPUT_LINE
5E6B 4498                        ;
5E6B 4499                        ;  THIS ROUTINE INPUTS A LINE OF DATA FROM EITHER THE CONSOLE UART OR THE
5E6B 4500                        ;  APT-RD UART AND PUTS THEM INTO BUFFERS.
5E6B 4501                        ;
5E6B 4502                        ;               IF APT IS PRESENT THE APT-RD PORT WILL RECEIVE
5E6B 4503                        ;       A PROMPT AND WILL BE MONITORED.
5E6B 4504                        ;
5E6B 4505                        ;               IF APT IS NOT PRESENT ONLY THE TERMINAL PORT
5E6B 4506 0231'32 AF             ;       WILL RECEIVE A PROMPT AND BE MONITORED.  THE STANDARD
5E6F 4507                        ;       CONSOLE AND MICRO MONITOR WILL EXPECT THE TU58 TO BE PRESENT
5E6F 4508 0F 0333'3A             ;       IF ANY COMMAND REQUIRING IT IS USED.  IN THE CASE OF DUMP
          127C'D2                ;       MODE ISSUING COMMANDS REQUIRING THE TU58 ARE ILLEGAL.
5E76 4509 4129 CD                ;
5E79 4510 128E'C3                ;*****************************************************************************
5E7C 4511
5E7C 4512
5E7C 4513 50 0E 01'64'21 INPUT_LINE:     CLEAR   ERROR
          0D 23 77 AF
          1282'C2
5E88 4514 01'63'21               IF      REPEAT                          ;IF REPEAT COMMAND, DO THIS
5E88 4515 4123 CD                CALL    GCVECT                          ;CHECK FOR OPERATOR INPUT
5E8B 4516                        JMP     1$                              ;SKIP GET LINE IF NOT
5E8E 4517 0F 0039'3A             ENDIF
          129F'D2
5E95 4518 0039'32 AF             ZERO    TTBUF,80                        ;ZERO BUFFER AREA
5E99 4519 1208'CD
                                 LXI     H,TTBUF_CNT                     ;INIT TTY BUFFER POINTER
                                 CALL    GLVECT                          ;CALL GET LINE ROUTINE IN BOOT
                                                                         ; BLOCK
                         1$:     IF      CNTLC_TYPED
                                 CLEAR   CNTLC_TYPED                     ;CLEAR FLAG
                                 CALL    PRINT_PROMPT                    ;PROMPT FOR NEW INPUT
```

```
ZZ-ENKAA-2.1    4C00    4
5E9C 4520 126B'C3               *
5E9F 4521                       ******************
5E9F 4522
5E9F 4523 01'64'21
5EA2 4524 03E0'22
5EA5 4525 03A0'32 3C AF
5EAA 4526
5EAA 4527
5EAA 4528 C5 46 00'55'21        ******************
          77 50 3E              *
5EB2 4529                       *
5EB2 4530 01CD'3A               *
5EB5 4531 03E0'2A               *
5EB8 4532 BE                    *
5EB9 4533                       *
5EB9 4534 12EB'C2               * ****************
5EBC 4535 0F 03A0'3A            * * ***************
          12DF'D2               * * *
5EC3 4536 03E0'2A               * * *
5EC6 4537 7D                    * * *
5EC7 4538 01'64'01              * * *
5ECA 4539 91                    * * *
5ECB 4540 4F                    * * *
5ECC 4541 50 3E                 * * *
5ECE 4542 91                    * * *
5ECF 4543 3D                    * * *
5ED0 4544 4F                    * * *
5ED1 4545                       * * *
5ED1 4546 03E0'2A               * * *
5ED4 4547 EB                    * * *
5ED5 4548 03E0'2A               * * *
5ED8 4549 23                    * * *
5ED9 4550 11A4'CD               * * *
5EDC 4551                       * * *
5EDC 4552 12E3'C3               * * ***************
5EDF 4553 03'E0'21              * * *
5EE2 4554 34                    * * *
5EE3 4555                       * * *
5EE3 4556                       * * ***************
5EE3 4557                       * *
5EE3 4558 03A0'32 3C AF         * *
5EE8 4559                       * *
5EE8 4560 12F3'C3               * ****************
5EEB 4561 03A0'32 AF            * *
5EEF 4562 03'E0'21              * *
5EF2 4563 34                    * *
5EF3 4564                       * ****************
5EF3 4565                       *
5EF3 4566 020C'3A               *
5EF6 4567 03E0'2A               *
5EF9 4568 BE                    *
5EFA 4569 130B'C2               * ****************
5EFD 4570 0F 03A0'3A            * * ***************
          130B'D2               * * *
5F04 4571 AF                    * * *
5F05 4572 77                    * * *
5F06 4573 020C'3A               * * *
5F09 4574 2B                    * * *
5F0A 4575 77                    * * *
5F0B 4576                       * * ***************
```

```
JMP     INPUT_LINE
ENDIF

LXI     H,TTBUF                 ;POINT TO BEGINNING OF TTBUF
SHLD    TTBUF_PNT
SET     SPACE_FOUND             ;FLAG INDICATING SPACE FOUND
                                ;SET TO REMOVE LEADING SPACES

DO      80                      ;FOR EACH LOCATION IN TTBUF

LDA     BLANK_A                 ;POINT TO ASCII SPACE CHARACTER
LHLD    TTBUF_PNT               ;POINT TO TTBUF CHARACTER
CMP     M                       ;SEE IF CHARACTER IS A SPACE

IFZ                             ;IF IT IS A SPACE
IF      SPACE_FOUND             ;AND IF IT FOLLOWS A SPACE

LHLD    TTBUF_PNT               ;CURRENT LOCATION IN TTBUF
MOV     A,L
LXI     B,TTBUF                 ;BEGINNING OF TTBUF
SUB     C                       ;SUBTRACT TO GET COUNT
MOV     C,A
MVI     A,80                    ;SUBTRACT FROM TOTAL CHARACTERS
SUB     C
DCR     A                       ;C GETS COUNT OF CHARACTERS
MOV     C,A                     ; TO BE MOVED IN TTBUF

LHLD    TTBUF_PNT               ;SHIFT ALL CHARACTERS FOLLOWING
XCHG                            ; SPACE DOWN ONE PLACE IN TTBUF
LHLD    TTBUF_PNT
INX     H
CALL    MOVER_R

ELSE
LXI     H,TTBUF_PNT             ;DO NOT SHIFT TTBUF IS THIS
INR     M                       ; WAS THE FIRST SPACE
                                ;POINT TO NEXT CHARACTER
ENDIF

SET     SPACE_FOUND             ;INDICATE SPACE WAS FOUND

ELSE
CLEAR   SPACE_FOUND             ;INDICATE SPACE WAS NOT FOUND
LXI     H,TTBUF_PNT             ;POINT TO NEXT CHARACTER
INR     M
ENDIF

LDA     CRLF_A+1                ;POINT TO <CR> CHARACTER
LHLD    TTBUF_PNT               ;POINT TO CURRENT CHARACTER
CMP     M                       ;SEE IF WE ARE AT END OF LINE
IFZ                             ;IF END OF LINE
IF      SPACE_FOUND             ;AND IF A SPACE IS PRECEDING

XRA     A                       ;ZERO OUT OLD END OF LINE
MOV     M,A
LDA     CRLF_A+1                ;REMOVE THE SPACE
DCX     H
MOV     M,A
ENDIF
```

```
5F0B 4577                    * ****************          ENDIF
5F0B 4578                    *
5F0B 4579 35 00'55'21        ******************          ENDDO                          ;REPEAT FOR ALL CHARACTERS
          70 C1 12B2'C2
5F14 4580
5F14 4581 01'64'21                                       LXI     H,TTBUF                ;INIT TTY BUFFER POINTER
5F17 4582 03E0'22                                        SHLD    TTBUF_PNT
5F1A 4583
5F1A 4584 C9                                             RET
5F1B 4585
5F1B 4586                    ;*************************************************************************************
5F1B 4587                    ;
5F1B 4588                    ;  INPUT_CHAR_IF
5F1B 4589                    ;  THIS ROUTINE INPUTS A CHAR FROM THE APT-RD PORT
5F1B 4590                    ;
5F1B 4591                    ;  OUTPUT CONDTIONS
5F1B 4592                    ;       ACCUMULATOR  = CHAR THAT WAS INPUT
5F1B 4593                    ;
5F1B 4594                    ;*************************************************************************************
5F1B 4595
5F1B 4596 4145 CD            INPUT_CHAR_IF:  CALL    GSVECT                         ;CALL GET SILO ROUTINE IN
5F1E 4597                                                                           ; BOOT BLOCK.  IF CHAR FOUND,
5F1E 4598                                                                           ; CARRY IS SET AND CHAR IS IN
5F1E 4599                                                                           ; ACCUMULATOR
5F1E 4600
5F1E 4601 131B'D2                                        JNC     INPUT_CHAR_IF          ;LOOP UNTIL CHAR RECEIVED
5F21 4602
5F21 4603 C9                                             RET
5F22 4604                    ;*************************************************************************************
5F22 4605                    ;
5F22 4606                    ;  CNTLC_VEC
5F22 4607                    ;  THIS ROUTINE IS ENTERED AFTER CALLING THE BOOT BLOCK GET CHAR OR GET LINE
5F22 4608                    ;  ROUTINE IF A ^C WAS TYPED.  IT WILL SET A FLAG, RESTORE THE STACK POINTER,
5F22 4609                    ;  AND RETURN TO THE CALLING ROUTINE.
5F22 4610                    ;
5F22 4611                    ;*************************************************************************************
5F22 4612
5F22 4613
5F22 4614 0039'32 3C AF      CNTLC_VEC:      SET     CNTLC_TYPED                    ;SET FLAG INDICATING ^C
5F27 4615 4086 2A                            LHLD    SPBUFF                         ;HL GETS OLD SP
5F2A 4616 F9                                 SPHL                                   ;RESTORE SP
5F2B 4617 0333'32 AF                         CLEAR   REPEAT                         ;CLEAR REPEAT FLAG
5F2F 4618 37                                 STC                                    ;SET CARRY FOR INPUT_CHAR_IF
5F30 4619 03 3E                              MVI     A,<^X3>                        ;PUT HEX FOR ^C IN A
5F32 4620 C9                                 RET                                    ;RETURN WILL NOW WORK
5F33 4621
5F33 4622                    ;*************************************************************************************
5F33 4623                    ;
5F33 4624                    ;  CNTLP_VEC
5F33 4625                    ;  THIS ROUTINE IS ENTERED AFTER CALLING THE BOOT BLOCK GET CHAR OR GET LINE
5F33 4626                    ;  ROUTINE IF A ^P WAS TYPED.  IT WILL FIX THE STACK POINTER AND CALL SET_FOR_
5F33 4627                    ;  CONT TO REENTER THE PARSER FOR THE NEXT COMMAND.
5F33 4628                    ;
5F33 4629                    ;*************************************************************************************
5F33 4630
5F33 4631 4086 2A            CNTLP_VEC:      LHLD    SPBUFF                         ;HL GETS OLD SP
5F36 4632 F9                                 SPHL                                   ;RESTORE SP
5F37 4633 AF                                 XRA     A                              ;CLEAR FLAGS
5F38 4634 003A'32                            STA     CONTINUE                       ;CLEAR CONTINUE FLAG IN CASE
5F3B 4635                                                                           ; 8085 TEST IS EXECUTING
```

```
5F3B 4636 0333'32                                   STA     REPEAT                  ;CLEAR REPEAT FLAG
5F3E 4637 1349'C3                                   JMP     SET_FOR_CONT_R          ;SET UP FOR CO COMMAND AND GO
5F41 4638                                                                           ; TO PARSER
5F41 4639                                   ;*******************************************************************************
5F41 4640                                   ;
5F41 4641                                   ;  CHECK_CNTLC
5F41 4642                                   ;
5F41 4643                                   ;  THIS ROUTINE CHECKS THE CNTLC_TYPED FLAG TO SEE IF A CNTLC HAS BEEN
5F41 4644                                   ;  DETECTED.  IF SO, IT CALLS THE SET_FOR_CONT_R ROUTINE.
5F41 4645                                   ;
5F41 4646                                   ;*******************************************************************************
5F41 4647
5F41 4648 0039'3A                           CHECK_CNTLC:    LDA     CNTLC_TYPED             ;GET FLAG
5F44 4649 B7                                                ORA     A                       ;SET CONDITION CODES
5F45 4650 1349'C4                                           CNZ     SET_FOR_CONT_R          ;CALL SET FOR CONT IF ^C SET
5F48 4651 C9                                                RET                             ;ELSE RETURN
5F49 4652
5F49 4653                                   ;*******************************************************************************
5F49 4654                                   ;
5F49 4655                                   ;  SET_FOR_CONT_R:
5F49 4656                                   ;
5F49 4657                                   ;  THIS ROUTINE SETS UP FOR A CONTINUE COMMAND TO BE EXECUTED IF IT IS
5F49 4658                                   ;  ASKED FOR
5F49 4659                                   ;
5F49 4660                                   ;*******************************************************************************
5F49 4661
5F49 4662 AF                                SET_FOR_CONT_R: XRA     A                       ;CLEAR FLAGS
5F4A 4663 0039'32                                           STA     CNTLC_TYPED             ;CLEAR CNTLC_TYPED
5F4D 4664 02F4'32                                           STA     NO_SAVE_UPC_CSR         ;CLEAR NO_SAVE_UPC_CSR
5F50 4665
5F50 4666 023B'3A                                           LDA     EXECUTE                 ;SAVE EXECUTION FLAGS
5F53 4667 0204'32                                           STA     C_EXECUTE
5F56 4668 0335'3A                                           LDA     RUNNING
5F59 4669 0205'32                                           STA     C_RUNNING
5F5C 4670
5F5C 4671 0F 0335'3A                *******************     IF      RUNNING
          136E'D2                   *
5F63 4672 1399'CD                   *                       CALL    STOP_CPU                ;HALT CPU
5F66 4673 02C8'32 3C AF             *                       SET     MICRO_STEP              ;SET FLAG TO PRINT UPC-1
5F6B 4674 0B2C'CD                   *                       CALL    EXAM_UPC                ;PRINT OUT HALTED ADDRESS
5F6E 4675                           *******************     ENDIF
5F6E 4676
5F6E 4677 02C8'32 AF                                        CLEAR   MICRO_STEP              ;CLEAR MICRO STEP MODE
5F72 4678 00C4'2A                                           LHLD    SAVED_UPC               ;SAVE CSR AND UPC FOR CONT
5F75 4679 00C8'22                                           SHLD    CONT_SAVE_UPC
5F78 4680
5F78 4681 00'BB'21                                          LXI     H,SAVED_CSR
5F7B 4682 00'BE'11                                          LXI     D,CONT_SAVE_CSR
5F7E 4683 118E'CD                                           CALL    MOVER_3_R
5F81 4684
5F81 4685 023B'32 AF                                        CLEAR   EXECUTE                 ;CLEAR EXECUTION FLAG TO ENTER
5F85 4686                                                                               ; PARSER
5F85 4687
5F85 4688 0F 0209'3A                *******************     IFN     CONSOLE_TEST            ;IF 8085 TEST IS NOT RUNNING
          1394'DA                   *
5F8C 4689 003A'32 3C AF             *                       SET     CONTINUE                ;CONTINUE ALWAYS LEGAL FOR WCS
5F91 4690                           *                                                       ; TESTS IF THIS ROUTINE CALLED
5F91 4691 1396'C3                   *******************     ELSE
5F94 4692 3C D3                     *                       OUT     CLRLIT                  ;IF 8085 TEST, TURN OFF RUN
5F96 4693                           *                                                       ; LIGHT
```

```
5F96 4694                       ******************          ENDIF
5F96 4695
5F96 4696 2022'C3                                           JMP     PARSER
5F99 4697
5F99 4698                                       ;*******************************************************************************
5F99 4699                                       ;
5F99 4700                                       ;  STOP_CPU
5F99 4701                                       ;  THIS ROUTINE STOPS THE CPU. IT SAVES THE UPC AND CSR SO THAT
5F99 4702                                       ;  EXECUTION MAY RESUME AT THE NEXT INSTRUCTION
5F99 4703                                       ;
5F99 4704                                       ;*******************************************************************************
5F99 4705
5F99 4706 0335'32 AF                            STOP_CPU:       CLEAR   RUNNING
5F9D 4707 3C D3                                                 OUT     CLRLIT                  ;CLEAR RUNLITE
5F9F 4708
5F9F 4709 20 D3                                                 OUT     CLRCLK                  ;STOP CPU CLOCK
5FA1 4710 A7 D3                                                 OUT     DISMEM                  ;DISABLE MEM REF
5FA3 4711 A1 D3                                                 OUT     DISPAR                  ;DISABLE PARITY
5FA5 4712
5FA5 4713 0F 02F4'3A            ******************              IFN     NO_SAVE_UPC_CSR         ;DON'T SAVE UPC AND CSR
          13C7'DA               *                                                               ; IF FLAG SET
5FAC 4714                       *
5FAC 4715 1148'CD               *                             CALL    LOAD_UPC_R              ;GET UPC AND SAVE UPC
5FAF 4716 00C2'2A               *                             LHLD    UPC_VALUE
5FB2 4717 00C4'22               *                             SHLD    SAVED_UPC
5FB5 4718 1148'CD               *                             CALL    LOAD_UPC_R              ;RETURN VALUE
5FB8 4719                       *
5FB8 4720 1116'CD               *                             CALL    LOAD_CSR_R              ;GET CSR AND SAVE CSR
5FBB 4721 00'B8'21              *                             LXI     H,CSR_VALUE
5FBE 4722 00'BB'11              *                             LXI     D,SAVED_CSR
5FC1 4723 118E'CD               *                             CALL    MOVER_3_R
5FC4 4724 1116'CD               *                             CALL    LOAD_CSR_R              ;RETURN VALUE
5FC7 4725                       *
5FC7 4726                       ******************          ENDIF
5FC7 4727
5FC7 4728 02F4'32 AF                                        CLEAR   NO_SAVE_UPC_CSR         ;CLEAR FLAG
5FCB 4729 1D43'C3                                           JMP     CLEAR_CPU_ATT_R
5FCE 4730
5FCE 4731
5FCE 4732                                       ;*******************************************************************************
5FCE 4733                                       ;
5FCE 4734                                       ;  START_CPU
5FCE 4735                                       ;  THIS ROUTINE STARTS THE CPU AT THE ADDRESS GIVEN BY THE RAM
5FCE 4736                                       ;  LOCATION STARTING_ADD
5FCE 4737                                       ;  IF MICRO STEP IS SET THE CPU IS SET UP BUT NOT STARTED
5FCE 4738                                       ;
5FCE 4739                                       ;*******************************************************************************
5FCE 4740
5FCE 4741 03A1'2A                               START_CPU:      LHLD    STARTING_UPC
5FD1 4742 00C2'22                                               SHLD    UPC_VALUE               ;MOVE STARTING ADDRESS
5FD4 4743
5FD4 4744 1148'CD                                               CALL    LOAD_UPC_R
5FD7 4745 10F9'CD                                               CALL    NOP_CSR_R
5FDA 4746
5FDA 4747 13F2'C3                                               JMP     START_CPU_COM           ;GO TO ROUTINE COMMON TO
5FDD 4748                                                                                       ; START OR RESTART
5FDD 4749
5FDD 4750
5FDD 4751                                       ;*******************************************************************************
5FDD 4752                                       ;
```

```
5FDD 4753
5FDD 4754                       ;  THIS SUBROUTINE IS USED TO RESTART THE CPU AND IT RESTORES THE
5FDD 4755                       ;  CSR AND UPC.  IF MICRO STEP IS SET THE CPU IS SET UP BUT NOT STARTED
5FDD 4756                       ;
5FDD 4757                       ;*************************************************************************
5FDD 4758 00C4'2A               RESTART_CPU:    LHLD    SAVED_UPC               ;RESTORE UPC
5FE0 4759 00C2'22                               SHLD    UPC_VALUE
5FE3 4760
5FE3 4761 1148'CD                               CALL    LOAD_UPC_R
5FE6 4762 00'BB'21                              LXI     H,SAVED_CSR             ;LOAD CSR
5FE9 4763 00'B8'11                              LXI     D,CSR_VALUE
5FEC 4764 118E'CD                               CALL    MOVER_3_R
5FEF 4765 1116'CD                               CALL    LOAD_CSR_R
5FF2 4766
5FF2 4767                       START_CPU_COM:
5FF2 4768 0F 030C'3A            ******************      IF      PAR_ON
          13FB'D2               *
5FF9 4769 A0 D3                 *                       OUT     ENBPAR          ;ENABLE PARITY
5FFB 4770                       ******************      ENDIF
5FFB 4771
5FFB 4772 0F 029F'3A            ******************      IF      MEM_REQ
          1404'D2               *
6002 4773 A6 D3                 *                       OUT     ENBMEM          ;ENABLE MEM REF
6004 4774                       ******************      ENDIF
6004 4775
6004 4776 0F 02C8'3A            ******************      IFN     MICRO_STEP
          140D'DA               *
600B 4777 21 D3                 *                       OUT     SETCLK
600D 4778                       ******************      ENDIF
600D 4779
600D 4780 0335'32 3C AF                                 SET     RUNNING
6012 4781 3D D3                                         OUT     SETLIT          ;SET RUN LITE
6014 4782
6014 4783 C9                                            RET
6015 4784
6015 4785
6015 4786                       ;*************************************************************************
6015 4787                       ;
6015 4788                       ;  WRITE_LS_R
6015 4789                       ;  THIS RUTINE WRITES A 32 BIT LS VALUE
6015 4790                       ;  INPUT CONDITIONS:
6015 4791                       ;       AC = LS ADDRESS THAT IS TO BE WRITTEN
6015 4792                       ;       DATA0 THRU DATA3 CONTAIN THE 32 BITS OF DATA TO BE WRITTEN
6015 4793                       ;
6015 4794                       ;*************************************************************************
6015 4795
6015 4796 0071'32               WRITE_LS_R:     STA     XD_ADDRS                ;SAVE ADDRESS TO BE WRITTEN
6018 4797
6018 4798 04'35'21                              LXI     H,S_WRITE_LS            ;RESET MOVE INSTRUCTION
601B 4799 04'32'11                              LXI     D,X_WRITE_LS+1
601E 4800 118E'CD                               CALL    MOVER_3_R
6021 4801
6021 4802 04'32'21                              LXI     H,X_WRITE_LS+1          ;MODIFY MOVE INSTRUCTION
6024 4803 1450'CD                               CALL    MAKE_XD_R
6027 4804
6027 4805 149E'CD                               CALL    WRITE_DATA_32_R         ;WRITE THE DATA TO 2901
602A 4806
602A 4807 04'31'21                              LXI     H,X_WRITE_LS            ;PERFORM THE WRITE
602D 4808 15C2'CD                               CALL    PERFORM_CSR_R
6030 4809
```

```
6030 4810 4129 C3                       JMP     GCVECT                  ;SEE IF INPUT FOR SILO
6033 4811
6033 4812
6033 4813
6033 4814
6033 4815
6033 4816
6033 4817
6033 4818
6033 4819
6033 4820
6033 4821
6033 4822
6033 4823 07 3E         READ_LS_7:      MVI     A,LS_7                  ;LS ADDRESS 7
6035 4824
6035 4825 0071'32       READ_LS_R:      STA     XD_ADDRS                ;SAVE ADDRESS TO BE READ
6038 4826
6038 4827 04'23'21                      LXI     H,S_READ_LS             ;RESET MOVE INSTRUCTION
603B 4828 04'20'11                      LXI     D,X_READ_LS+1
603E 4829 118E'CD                       CALL    MOVER_3_R
6041 4830
6041 4831 04'20'21                      LXI     H,X_READ_LS+1           ;AND MODIFY MOVE TO DO IT
6044 4832 1450'CD                       CALL    MAKE_XD_R
6047 4833
6047 4834 04'1F'21                      LXI     H,X_READ_LS             ;PERFORM THE READ
604A 4835 15C2'CD                       CALL    PERFORM_CSR_R
604D 4836 1481'C3                       JMP     READ_DATA_32_R
6050 4837
6050 4838
6050 4839
6050 4840
6050 4841
6050 4842
6050 4843
6050 4844
6050 4845
6050 4846
6050 4847
6050 4848
6050 4849
6050 4850 0071'3A       MAKE_XD_R:      LDA     XD_ADDRS                ;PUT BIT 7 OF XD_ADDRS
6053 4851 17                            RAL                             ;INTO BIT 16 OF INSTRUCTION
6054 4852 00 3E                         MVI     A,0
6056 4853 17                            RAL
6057 4854 B6                            ORA     M
6058 4855 77                            MOV     M,A
6059 4856 23                            INX     H
605A 4857
605A 4858 0071'3A                       LDA     XD_ADDRS                ;PUT BITS 0 - 6 INTO BITS
605D 4859                                                               ;15 - 9 OF INSTRUCTION
605D 4860 B7                            ORA     A                       ;CLEAR CARRY
605E 4861 17                            RAL
605F 4862 B6                            ORA     M
6060 4863 77                            MOV     M,A
6061 4864
6061 4865 C9                            RET
6062 4866
6062 4867
6062 4868
6062 4869
```

```
;*****************************************************************************
;
;  READ_LS_R
;  THIS ROUTINE READS A 32 BIT LS VALUE
;  INPUT CONDITIONS:
;       AC = LS ADDRESS THAT IS TO BE READ
;  OUTPUT CONDITIONS:
;       DATA0 THRU DATA3 CONTAIN THE 32 BITS READ
;
;*****************************************************************************

;*****************************************************************************
;
;  MAKE_XD_R
;  THIS ROUTINE MAKES AN INSTRUCTION HAVE THE PROPER XD ADDRESS FIELD
;
;  INPUT CONDITIONS:
;       XD_ADDRS = DATA TO BE INSERTED INTO THE XD ADDRESS FIELD (BITS 9-16)
;       H + L    = POINTS TO THE 3 BYTE INSTRUCTION TO MODIFY
;
;*****************************************************************************

;*****************************************************************************
;
```

```
6062 4870
6062 4871
6062 4872
6062 4873
6062 4874
6062 4875
6062 4876
6062 4877
6062 4878
6062 4879 23
6063 4880
6063 4881 01CC'3A
6066 4882 1F
6067 4883 1F
6068 4884 00 3E
606A 4885 17
606B 4886 B6
606C 4887 77
606D 4888 23
606E 4889
606E 4890 01CC'3A
6071 4891 1F
6072 4892 00 3E
6074 4893 1F
6075 4894 B6
6076 4895 77
6077 4896
6077 4897 C9
6078 4898
6078 4899
6078 4900
6078 4901
6078 4902
6078 4903
6078 4904
6078 4905
6078 4906
6078 4907
6078 4908
6078 4909
6078 4910
6078 4911 23
6079 4912
6079 4913 B7
607A 4914
607A 4915 01B4'3A
607D 4916 17
607E 4917 B6
607F 4918 77
6080 4919
6080 4920 C9
6081 4921
6081 4922
6081 4923
6081 4924
6081 4925
6081 4926
6081 4927
6081 4928
6081 4929
```

```
;*****************************************************************************
;
;  MAKE_B
;  THIS ROUTINE MAKES AN INSTRUCTION HAVE THE PROPER B ADDRESS FIELD
;
;  INPUT CONDITIONS:
;       B_ADDRS   = DATA TO BE INSERTED INTO THE B ADDRESS FIELD (BITS 7-8)
;       H + L     = POINTS TO THE 3 BYTE INSTRUCTION TO MODIFY
;
;*****************************************************************************

MAKE_B:         INX     H                       ;MOVE TO SECOND BYTE

                LDA     B_ADDRS                 ;PUT BIT 2 OF B_ADDRS INTO
                RAR                             ;BIT 8 OF THE INSTRUCTION
                RAR
                MVI     A,0
                RAL
                ORA     M
                MOV     M,A
                INX     H

                LDA     B_ADDRS                 ;PUT BIT 1 OF B_ADDRS INTO
                RAR                             ;BIT 7 OF THE INSTRUCTION
                MVI     A,0
                RAR
                ORA     M
                MOV     M,A

                RET

;*****************************************************************************
;
;  MAKE_A
;  THIS ROUTINE MAKES AN INSTRUCTION HAVE THE PROPER A ADDRESS FIELD
;
;  INPUT CONDITIONS:
;       A_ADDRS   = DATA TO BE INSERTED INTO THE A ADDRESS FIELD (BITS 9-10)
;       H + L     = POINTS TO THE 3 BYTE INSTRUCTION TO MODIFY
;
;*****************************************************************************

MAKE_A:         INX     H                       ;MOVE TO SECOND BYTE

                ORA     A                       ;CLEAR THE CARRY FOR ROTATES

                LDA     A_ADDRS                 ;PUT BIT 2 OF B_ADDRS INTO
                RAL                             ;MOVE TO BITS 9 - 10
                ORA     M
                MOV     M,A                     ;OR IT INTO INSTRUCTION

                RET

;*****************************************************************************
;
;  READ_DATA_32_R
;  THIS ROUTINE IS USED TO READ 32 BITS OF DATA FROM THE
;  CPU 8 BITS AT A TIME VIA THE 2901
;
;*****************************************************************************
```

```
6081 4930
6081 4931 00'AC'11                  READ_DATA_32_R: LXI     D,DATA3                 ;DE POINTS TO DATA AREA
6084 4932 1496'CD                                   CALL    1$                      ;READ BYTE AND STORE
6087 4933
6087 4934 00'AB'11                                  LXI     D,DATA2                 ;DE POINTS TO DATA AREA
608A 4935 1496'CD                                   CALL    1$                      ;READ NEXT BYTE AND STORE
608D 4936
608D 4937 00'AA'11                                  LXI     D,DATA1                 ;DE POINTS TO DATA AREA
6090 4938 1496'CD                                   CALL    1$                      ;READ NEXT BYTE AND STORE
6093 4939
6093 4940 00'A9'11                                  LXI     D,DATA0                 ;DE POINTS TO DATA AREA
6096 4941                                                                           ; READ NEXT BYTE AND STORE
6096 4942
6096 4943 EC D3                     1$:             OUT     YBUSRD
6098 4944 80 DB                                     IN      READ                    ;READ MOST SIG. BYTE
609A 4945 12                                        STAX    D                       ;STORE IN DATA BUFFER
609B 4946 151B'C3                                   JMP     SHIFT_R_2901_R          ;READ NEXT BYTE
609E 4947
609E 4948
609E 4949                           ;*****************************************************************************
609E 4950                           ;
609E 4951                           ;  WRITE_DATA_32_R
609E 4952                           ;  THIS ROUTINE IS USED TO WRITE 32 BITS OF DATA TO THE CPU
609E 4953                           ;  DATA WRITES TO THE CPU ARE DONE IN TWO DIFFERENT MODES
609E 4954                           ;  SLOW MODE: AND FAST MODE:  SLOW MODE USES NO SUPPORT MICRO CODE
609E 4955                           ;  IN THE WCS MEMORY, AND THE FAST MODE DOES. THE MODE IS INITIALLY
609E 4956                           ;  SLOW MODE UNTIL A WCS FILE IMAGE IS LOADED.  THEN IT IS SWITCHED TO
609E 4957                           ;  FAST MODE UNLESS THERE IS AN ERROR IN THE FAST MODE VERIFY.
609E 4958                           ;
609E 4959                           ;*****************************************************************************
609E 4960
609E 4961                           WRITE_DATA_32_R:
609E 4962 0F 0240'3A                ******************  IFN     FAST_WRITE
          14D6'DA                   *
60A5 4963                           *                                               ;SLOW WRITE
60A5 4964 00'A1'21                  *                   LXI     H,X_CLR_WR0         ;CLEAR WORKING REG AND COMPLIM.
60A8 4965 15C2'CD                   *                   CALL    PERFORM_CSR_R       ;TO MAKE ALL 1'S THEN ROTATE
60AB 4966                           *                                               ;LEFT TO MAKE A 1 AND ASH LEFT
60AB 4967                           *                                               ;TO MAKE A ZERO
60AB 4968                           *
60AB 4969 C5 46 00'55'21            * ****************  DO      32
          77 20 3E                  * *
60B3 4970                           * *
60B3 4971 00'AC'21                  * *                 LXI     H,DATA_SHIFT+4      ;SHIFT DATA INTO CPU
60B6 4972 04 0E                     * *                 MVI     C,4
60B8 4973 10F0'CD                   * *                 CALL    SHIFTER1_R
60BB 4974                           * *
60BB 4975 14C7'D2                   * * **************  IFC
60BE 4976 00'79'21                  * * *               LXI     H,X_ROT_L           ;ADDR OF CSR INSTS TO EXECUTE
60C1 4977 15C2'CD                   * * *               CALL    PERFORM_CSR_R       ;MAKE A 1 BY END AROUND CARRY
60C4 4978                           * * *
60C4 4979 14CA'C3                   * * **************  ELSE
60C7 4980 1515'CD                   * * *               CALL    SHIFT_L_2901_R      ;MAKE A 0 BY SHIFT IN ZERO
60CA 4981                           * * **************  ENDIF
60CA 4982                           * *
60CA 4983 35 00'55'21               * ****************  ENDDO
          70 C1 14B3'C2             *
60D3 4984                           *
60D3 4985 1514'C3                   ******************  ELSE
60D6 4986                           *
```

```
ZZ-ENKAA-2.1    4C00    4
60D6 4987 06 00 21          *                   LXI     H,WCS_DAT_32_ADD        ;FAST WRITE
60D9 4988 7C                *                   MOV     A,H
60DA 4989 65                *                   MOV     H,L
60DB 4990 6F                *                   MOV     L,A
60DC 4991 00C2'22           *                   SHLD    UPC_VALUE
60DF 4992 1148'CD           *                   CALL    LOAD_UPC_R
60E2 4993                   *
60E2 4994 10F9'CD           *                   CALL    NOP_CSR_R               ;SET UP SUBROUTINE
60E5 4995                   *
60E5 4996 1570'CD           *                   CALL    MICRO_STEP_CPU_R        ;EXEC INIT FOR SUB.
60E8 4997 1570'CD           *                   CALL    MICRO_STEP_CPU_R
60EB 4998                   *
60EB 4999 C5 46 00'55'21    * ****************  DO      32
          77 20 3E          * *
60F3 5000                   * *
60F3 5001 00'AC'21          * *                 LXI     H,DATA_SHIFT+4          ;POINT TO END OF DATA AREA
60F6 5002 04 0E             * *                 MVI     C,4                     ;BYTES TO SHIFT IN REG C
60F8 5003 10F0'CD           * *                 CALL    SHIFTER1_R
60FB 5004                   * *
60FB 5005 AB D3             * *                 OUT     CLRATN                  ;SET A ZERO TO BE READ
60FD 5006                   * *
60FD 5007 1502'D2           * * ************    IFC
6100 5008 AA D3             * * *               OUT     SETATN                  ;SET A ONE TO BE READ
6102 5009                   * * ************    ENDIF
6102 5010                   * *
6102 5011 1570'CD           * *                 CALL    MICRO_STEP_CPU_R        ;CYCLE THROUGH LOOP IN SUB.
6105 5012 1570'CD           * *                 CALL    MICRO_STEP_CPU_R
6108 5013 1570'CD           * *                 CALL    MICRO_STEP_CPU_R
610B 5014                   * *
610B 5015 35 00'55'21       * ****************  ENDDO
          70 C1 14F3'C2     *
6114 5016                   *
6114 5017                   ******************  ENDIF
6114 5018
6114 5019 C9                                    RET
6115 5020
6115 5021
6115 5022                   ;*******************************************************************************
6115 5023                   ;
6115 5024                   ;  SHIFT_L_2901_R
6115 5025                   ;  THIS ROUTINE SHIFTS THE DATA IN THE 2901 LEFT 1 BIT THIS INCLUDES
6115 5026                   ;  SHIFTING A ZERO INTO THE LSB
6115 5027                   ;
6115 5028                   ;*******************************************************************************
6115 5029
6115 5030                   SHIFT_L_2901_R:
6115 5031 00'75'21                              LXI     H,X_SHIFT_L             ;ADDR OF CSR INSTS TO EXECUTE
6118 5032 15C2'C3                               JMP     PERFORM_CSR_R
611B 5033
611B 5034                   ;*******************************************************************************
611B 5035                   ;
611B 5036                   ;  SHIFT_R_2901_R
611B 5037                   ;  THIS ROUTINE SHIFTS THE DATA IN THE 2901 RIGHT 8 BITS AND LEAVES
611B 5038                   ;  THE LEAST SIG. BYTE ON THE Y-BUS TO BE READ
611B 5039                   ;
611B 5040                   ;*******************************************************************************
611B 5041
611B 5042                   SHIFT_R_2901_R:
611B 5043 00'85'21                              LXI     H,X_SHIFT_R             ;ADDR OF CSR INSTS TO EXECUTE
611E 5044 15C2'C3                               JMP     PERFORM_CSR_R
```

```
6121 5045
6121 5046                       ;*****************************************************************************
6121 5047                       ;
6121 5048                       ;  MICRO_STEPER
6121 5049                       ;  THIS ROUTINE SINGLE STEPS THE CPU. IF THE MACHINE IS IN SPACE BAR
6121 5050                       ;  MODE ANY CHAR OTHER THAN SPACE WILL CLEAR EXECUTE MODE AND STOP
6121 5051                       ;  MICRO STEPING.  IF THE MACHINE IS IN COUNT MODE IT WILL DECREMENT
6121 5052                       ;  THE COUNT AND STOP MICRO STEPING AT 0 COUNT
6121 5053                       ;
6121 5054                       ;*****************************************************************************
6121 5055
6121 5056 01 FE 02C8'3A        ********** MICRO_STEPER:,  IFEQI   MICRO_STEP,1          ;SPACE BAR MODE
     1552'C2                   *
6129 5057                      *
6129 5058 413D CD              *           1$:             CALL    RSVECT               ;CALL READ SILO ROUTINE IN
612C 5059                      *                                                        ; BOOT BLOCK.  IF CHAR FOUND,
612C 5060                      *                                                        ; CARRY IS SET AND CHAR IS IN
612C 5061                      *                                                        ; ACCUMULATOR
612C 5062                      *
612C 5063 1529'D2              *                           JNC     1$                   ;LOOP UNTIL CHAR RECEIVED
612F 5064                      *
612F 5065 7F E6                *                           ANI     HEX_7F               ;STRIP ASCII PARITY BIT
6131 5066 20 FE                *                           CPI     SPACE_A              ;SPACE?
6133 5067 1546'CA              *                           JZ      2$                   ;JUMP TO SINGLE STEP IF SPACE
6136 5068                      *
6136 5069 1399'CD              *                           CALL    STOP_CPU             ;STOP CPU, SAVE CSR AND UPC
6139 5070 02C8'32 AF           *                           CLEAR   MICRO_STEP           ;CLEAR FLAG
613D 5071 0F47'CD              *                           CALL    SAVE_GR              ;SAVE CPU WORKING REGS
6140 5072 1349'CD              *                           CALL    SET_FOR_CONT_R       ;NO SPACE, END MICRO STEPPING
6143 5073 13DD'C3              *                           JMP     RESTART_CPU          ;CONTINUE COMES HERE
6146 5074                      *
6146 5075 4145 CD              *           2$:             CALL    GSVECT               ;CALL GET SILO TO POP OUT SPACE
6149 5076 1570'CD              *                           CALL    MICRO_STEP_CPU_R     ;STEP MACHINE AND PRINT ADDRESS
614C 5077 0B2C'CD              *                           CALL    EXAM_UPC
614F 5078                      *
614F 5079 156F'C3              **********                  ELSE
6152 5080                      *
6152 5081 1570'CD              *                           CALL    MICRO_STEP_CPU_R     ;COUNT MODE
6155 5082 03'A3'21             *                           LXI     H,STEP_CNT           ;DECREMENT STEP COUNT
6158 5083 158C'CD              *                           CALL    DECR_WORD_R
615B 5084 C0                   *                           RNZ                          ;RETURN  UNTIL COUNT EXHAUSTED
615C 5085                      *
615C 5086 1399'CD              *                           CALL    STOP_CPU             ;STOP CPU, SAVE CSR AND UPC
615F 5087 0B2C'CD              *                           CALL    EXAM_UPC             ;PRINT ENDING UPC
6162 5088 02C8'32 AF           *                           CLEAR   MICRO_STEP           ;CLEAR FLAG
6166 5089 0F47'CD              *                           CALL    SAVE_GR              ;SAVE CPU WORKING REGS
6169 5090 1349'CD              *                           CALL    SET_FOR_CONT_R       ;NO SPACE, END MICRO STEPPING
616C 5091 13DD'C3              *                           JMP     RESTART_CPU          ;CONTINUE COMES HERE
616F 5092                      *
616F 5093                      **********                  ENDIF
616F 5094
616F 5095 C9                                               RET
6170 5096
6170 5097
6170 5098                       ;*****************************************************************************
6170 5099                       ;
6170 5100                       ;  MICRO_STEP_CPU_R
6170 5101                       ;  THIS ROUTINE CAUSES THE MACHINE TO DO ONE MICRO INSTRUCTION
6170 5102                       ;
6170 5103                       ;*****************************************************************************
```

```
6170 5104
6170 5105
6170 5106 23 D3          MICRO_STEP_CPU_R:
6172 5107 22 D3                    OUT     SETSS
                                   OUT     CLRSS                ;SINGLE STEP CPU
6174 5108
6174 5109 C9                       RET
6175 5110
6175 5111
6175 5112          ;*****************************************************************************
6175 5113          ;
6175 5114          ;  INC_TEMPW_5
6175 5115          ;  THIS ROUTINE INCREMENTS THE VALUE OF TEMP_WORD BY 5
6175 5116          ;
6175 5117          ;*****************************************************************************
6175 5118
6175 5119 00 06          INC_TEMPW_5:      MVI     B,0
6177 5120 05 0E                            MVI     C,5
6179 5121 03C6'2A                          LHLD    TEMP_WORD
617C 5122 09                               DAD     B                    ;ADD 5 TO TEMP_WORD
617D 5123 03C6'22                          SHLD    TEMP_WORD            ;TO MOVE TO NEXT FLAG NAME
6180 5124
6180 5125 C9                               RET
6181 5126
6181 5127          ;*****************************************************************************
6181 5128          ;
6181 5129          ;  INC_WORD_R
6181 5130          ;  THIS ROUTINE INCREMENTS A WORD OF DATA
6181 5131          ;
6181 5132          ;  INPUT CONDITIONS:
6181 5133          ;       H+L = POINTER TO THE WORD TO INCREMENT
6181 5134          ;  OUTPUT CONDITIONS:
6181 5135          ;       CONDITION CODES
6181 5136          ;
6181 5137          ;*****************************************************************************
6181 5138
6181 5139
6181 5140 23             INC_WORD_R:       INX     H                    ;POINT TO LOW ORDER PART
6182 5141
6182 5142 7E                               MOV     A,M
6183 5143 01 C6                            ADI     1
6185 5144 77                               MOV     M,A                  ;INC VALUE IN LOW ORD. PART
6186 5145 2B                               DCX     H                    ;MOVE TO HIGH ORDER PART
6187 5146
6187 5147 7E                               MOV     A,M
6188 5148 00 CE                            ACI     0                    ;ADD ZERO + CARRY
618A 5149 77                               MOV     M,A
618B 5150
618B 5151 C9                               RET
618C 5152
618C 5153
618C 5154          ;*****************************************************************************
618C 5155          ;
618C 5156          ;  DECR_WORD_R
618C 5157          ;  THIS ROUTINE DECREMENTS A WORD OF DATA
618C 5158          ;
618C 5159          ;  INPUT CONDITIONS:
618C 5160          ;       H+L = POINTER TO THE WORD TO DECREMENT
618C 5161          ;  OUTPUT CONDITIONS:
618C 5162          ;       CONDITION CODES
618C 5163          ;
```

```
618C 5164
618C 5165
618C 5166 23                        DECR_WORD_R:    INX     H                       ;MOVE TO SECOND BYTE
618D 5167
618D 5168 7E                                        MOV     A,M
618E 5169 01 D6                                     SUI     1                       ;DEC VALUE IN LOW ORD. PART
6190 5170 77                                        MOV     M,A
6191 5171 2B                                        DCX     H                       ;MOVE TO HIGH ORD. PART
6192 5172
6192 5173 7E                                        MOV     A,M
6193 5174 00 DE                                     SBI     0                       ;SUB CARRY
6195 5175 77                                        MOV     M,A
6196 5176
6196 5177 00 FE                                     CPI     0
6198 5178 C0                                        RNZ                             ;CHECK FOR ZERO RESULT
6199 5179 23                                        INX     H
619A 5180 7E                                        MOV     A,M
619B 5181 00 FE                                     CPI     0
619D 5182 C9                                        RET                             ;SEND BACK COND. CODE
619E 5183
619E 5184
619E 5185
619E 5186
619E 5187
619E 5188
619E 5189
619E 5190
619E 5191
619E 5192
619E 5193
619E 5194
619E 5195
619E 5196
619E 5197
619E 5198 005A'3A                   SET_LS_STAT:    LDA     FLAGS                   ;GET CURRENT FLAGS
61A1 5199 B0                                        ORA     B                       ;OR IN THIS BIT TO SET FLAG
61A2 5200 005A'32                                   STA     FLAGS                   ;STORE NEW FLAGS
61A5 5201 15AF'C3                                   JMP     WRITE_LS_STAT           ;WRITE FLAGS IN LS
61A8 5202
61A8 5203                           CLEAR_LS_STAT:
61A8 5204 005A'3A                                   LDA     FLAGS                   ;GET CURRENT FLAGS
61AB 5205 A0                                        ANA     B                       ;AND IN THIS BIT TO CLEAR FLAG
61AC 5206 005A'32                                   STA     FLAGS                   ;STORE NEW FLAGS
61AF 5207
61AF 5208 0F 0408'3A        **********************  WRITE_LS_STAT:  IF      WCS_LOADED
          15C1'D2           *
61B6 5209                   *
61B6 5210 005A'3A           *                               LDA     FLAGS           ;GET STATUS FLAGS
61B9 5211 00AC'32           *                               STA     DATA3
61BC 5212                   *
61BC 5213 48 3E             *                               MVI     A,FLAG_LOC      ;WRITE TO LS ERROR CONTROL LOC.
61BE 5214 1415'CD           *                               CALL    WRITE_LS_R
61C1 5215                   *
61C1 5216                   **********************          ENDIF
61C1 5217
61C1 5218 C9                                                RET
61C2 5219
61C2 5220
61C2 5221
61C2 5222
```

```
61C2 5223
61C2 5224
61C2 5225
61C2 5226
61C2 5227
61C2 5228
61C2 5229
61C2 5230
61C2 5231
61C2 5232
61C2 5233 7E                                ;  PERFORM_CSR_R
61C3 5234 0212'32                           ;  THIS ROOTINE WILL PUT INSTRUCTIONS IN TO THE CSR AND EXECUTE THEM
61C6 5235                                   ;
61C6 5236 23                                ;  INPUT CONDTIONS:
61C7 5237 03C6'22                           ;       H+L = POINTER TO INSTRUCTION TABLE
61CA 5238                                   ;                 TABLE = CNT,INST,INST,.....
61CA 5239 03C6'2A                           ;
61CD 5240 00'B8'11                          ;*****************************************************************************
61D0 5241 118E'CD
61D3 5242 03C6'22                           PERFORM_CSR_R:
61D6 5243                                           MOV     A,M                     ;GET COUNT
61D6 5244 1116'CD                                   STA     CSR_CNT                 ;AND SAVE
61D9 5245
61D9 5246 23 D3                                     INX     H
61DB 5247 22 D3                                     SHLD    TEMP_WORD
61DD 5248
61DD 5249 02'12'21                  1$:             LHLD    TEMP_WORD
61E0 5250 35                                        LXI     D,CSR_VALUE             ;MOVE 3 BYTES OF INST TO CSR_VALUE
61E1 5251 15CA'C2                                   CALL    MOVER_3_R
61E4 5252                                           SHLD    TEMP_WORD
61E4 5253 00'9E'21
61E7 5254 00'B8'11                                  CALL    LOAD_CSR_R              ;PUT INST IN CSR
61EA 5255 118E'CD
61ED 5256                                           OUT     SETSS
61ED 5257 1116'C3                                   OUT     CLRSS                   ;SINGLE STEP CPU
61F0 5258
61F0 5259                                           LXI     H,CSR_CNT
61F0 5260                                           DCR     M                       ;DECREMENT THE COUNT
61F0 5261                                           JNZ     1$                      ;LOOP UNITL ZERO
61F0 5262
61F0 5263                                           LXI     H,X_MOV_WRWR            ;LEAVE MOV INST IN CSR TO
61F0 5264                                           LXI     D,CSR_VALUE             ;ENABLE Y-BUS
61F0 5265                                           CALL    MOVER_3_R
61F0 5266
61F0 5267                                           JMP     LOAD_CSR_R              ;LOAD MOV
61F0 5268
61F0 5269
61F0 5270                           ;*****************************************************************************
61F0 5271                           ;
61F0 5272 0213'32 AF                ;  ATTENTION
61F4 5273 1BE6'CD                   ;  THIS ATTENTION ROUTINE IS USED WHEN THE CPU IS EXECUTING TESTS(RUNNING)
61F7 5274                           ;  IT IS BASICLY AN IDLE LOOP FOR THE 8085.  DURING THE IDLE LOOP
61F7 5275 0F 0335'3A      **********;  THE 8085 CHECKS FOR SPECIAL OCCURANCES SUCH AS ^C,^P,OR CPATTN
          164D'D2         *         ;  IF CPATTN SIGNAL COMES HIGH IT MEANS THAT THE CPU MICRO CODED
61FE 5276                 *         ;  DIAGNOSTIC IS REQUESTING SOMETHING.  MICRO STEPPING IS DONE ONCE
61FE 5277 02C8'3A         *         ;  FOR EACH ITTERATION OF THE IDLE LOOP
6201 5278 B7              *         ;
6202 5279 1521'C4         *         ;*****************************************************************************
6205 5280                 *
6205 5281 83 DB           *         ATTENTION:        CLEAR   DATA_XFER_FLG
                                                      CALL    RESET_TIMOUT            ;INIT TIMOUTS AND XFER INFO
                                    ;
                                                      WHILE   RUNNING
                                    ;
                                                      LDA     MICRO_STEP              ;GET MICRO_STEP FLAG
                                                      ORA     A                       ;SET CONDITION CODES
                                                      CNZ     MICRO_STEPER            ;CALL IF MICRO STEP
                                    ;
                                                      IN      CPATTN
```

```
6207 5282 1F                    *
6208 5283 1644'D2               * ****************
620B 5284                       * *
620B 5285 01 3E                 * *
620D 5286 0011'32               * *
6210 5287                       * *
6210 5288 1BE6'CD               * *
6213 5289 1399'CD               * *
6216 5290                       * *
6216 5291 00C4'2A               * *
6219 5292 00C6'22               * *
621C 5293                       * *
621C 5294 0F47'CD               * *
621F 5295                       * *
621F 5296 1341'CD               * *
6222 5297                       * *
6222 5298 40 3E                 * *
6224 5299 1435'CD               * *
6227 5300 164E'CD               * *
622A 5301                       * *
622A 5302 0F 0056'3A            * * **************
          1644'DA               * * *
6231 5303                       * * *
6231 5304 0F 03DC'3A            * * * ************
          1644'DA               * * * *
6238 5305 0F41'CD               * * * *
623B 5306 00C6'2A               * * * *
623E 5307 03A1'22               * * * *
6241 5308 13CE'CD               * * * *
6244 5309                       * * * ************
6244 5310                       * * *
6244 5311                       * * **************
6244 5312                       * *
6244 5313                       * ****************
6244 5314                       *
6244 5315 4129 CD               *
6247 5316 1BBB'CD               *
624A 5317                       *
624A 5318 15F7'C3               ******************
624D 5319
624D 5320 C9
624E 5321
624E 5322
624E 5323
624E 5324
624E 5325
624E 5326
624E 5327
624E 5328
624E 5329
624E 5330
624E 5331 00'AF'11
6251 5332 119F'CD
6254 5333
6254 5334
6254 5335 20 E6 00B2'3A         ******************
          1664'CA               *
625C 5336 030B'32 3C AF         *
6261 5337 171D'CD               *
6264 5338                       ******************
```

```
RAR
IFC                                     ;IF CPU_ATTENTION

MVI     A,1
STA     APT_MESSAGE_CODE+1              ;SET APT START FLAG

CALL    RESET_TIMOUT                    ;CPU HEARD, SO RESET TIMOUT
CALL    STOP_CPU

LHLD    SAVED_UPC                       ;SAVE UPC WHEN GOING TO
SHLD    UPC_SUB                         ;SUBROUTINE

CALL    SAVE_WR

CALL    CHECK_CNTLC                     ;IF CONTROL C TYPED STOP

MVI     A,STATUS
CALL    READ_LS_R                       ;READ LS COMMAND + STATUS
CALL    DO_COMMANDS

IFN     EOS_FLG                         ;IF EOS DONT RESTART

IFN     TEST_ZERO                       ;IF TEST 0 DONT RESTART

CALL    RESTORE_WR
LHLD    UPC_SUB                         ;RETURN FROM SUBROUTINE
SHLD    STARTING_UPC                    ;START CPU AGAIN
CALL    START_CPU
ENDIF

ENDIF

ENDIF

CALL    GCVECT                          ;CHECK FOR OPERATOR REQUEST (^P)
CALL    TIMOUT                          ;CHECK FOR TIMOUT

ENDWHILE

RET

;*****************************************************************************
;
;  DO_COMMANDS
;  THIS ROUTINE DECODES THE LS COMMAND AND STATUS WORD AND CALLS ROUTINES
;  TO PERFORM THE FUNCTIONS WHERE NECESSARY.
;
;*****************************************************************************

DO_COMMANDS:    LXI     D,COMMAND0              ;DESTINATION
                CALL    MOV_FROM_DATA0          ;MOVE 4 BYTES OF DATA FROM
                                                ; DATA0

                IFMBI   COMMAND3,WRONG_LAB      ;BIT 5 = CRD PACK NOT IN RL02

                SET     PACK_ERROR              ;SET FLAG TO INDICATE RL02 ERROR
                CALL    WCS_ERROR               ;CALL ERROR ROUTINE
                ENDIF
```

```
6264 5339
6264 5340 40 E6 00B2'3A          ********************
          1672'CA                *
626C 5341 03'E2'21               *
626F 5342 19A7'CD                *
6272 5343                        ********************
6272 5344
6272 5345 80 E6 00B2'3A          ********************
          1687'CA                *
627A 5346 03'29'21               *
627D 5347 19A7'CD                *
6280 5348 00AC'3A                *
6283 5349 B7                     *
6284 5350 1926'C4                *
6287 5351                        ********************
6287 5352
6287 5353 01 E6 00B1'3A          ********************
          1692'CA                *
628F 5354 171D'CD                *
6292 5355                        ********************
6292 5356
6292 5357 02 E6 00B1'3A          ********************
          16A2'CA                *
629A 5358 0298'32 3C AF          *
629F 5359 194C'CD                *
62A2 5360                        *
62A2 5361                        ********************
62A2 5362
62A2 5363 08 E6 00B1'3A          ********************
          16AD'CA                *
62AA 5364 193A'CD                *
62AD 5365                        ********************
62AD 5366
62AD 5367 10 E6 00B1'3A          ********************
          16BB'CA                *
62B5 5368 02'42'21               *
62B8 5369 19A7'CD                *
62BB 5370                        ********************
62BB 5371
62BB 5372 20 E6 00B1'3A          ********************
          16C6'CA                *
62C3 5373 1C78'CD                *
62C6 5374                        ********************
62C6 5375
62C6 5376 40 E6 00B1'3A          ********************
          16D3'CA                *
62CE 5377 0056'32 3C AF          *
62D3 5378                        ********************
62D3 5379
62D3 5380 80 E6 00B1'3A          ********************
          16DE'CA                *
62DB 5381 1B05'CD                *
62DE 5382                        ********************
62DE 5383
62DE 5384 01 E6 00B0'3A          ********************
          16E9'CA                *
62E6 5385 19BD'CD                *
62E9 5386                        ********************
62E9 5387
62E9 5388 02 E6 00AF'3A          ********************
```

```
IFMBI   COMMAND3,UBE_PRES       ;BIT 6 = PRINT UBE PRESENT
LXI     H,UBE_A
CALL    PRINT_MOD_PRES          ;PRINT PRESENT OR NOT PRESENT
ENDIF

IFMBI   COMMAND3,R80_PRES       ;BIT 7 = PRINT R80 PRESENT
LXI     H,R80_A
CALL    PRINT_MOD_PRES          ;PRINT PRESENT OR NOT PRESENT
LDA     DATA3                   ;SEE IF PRESENT FLAG SET
ORA     A                       ;SET CONDITION CODES
CNZ     PRINT_DRIVE             ;PRINT DRIVE NUMBER IF PRESENT
ENDIF

IFMBI   COMMAND2,WCSERR         ;BIT 8 = ERROR FOUND BY WCS
CALL    WCS_ERROR
ENDIF

IFMBI   COMMAND2,SETPAR_ERR     ;BIT 9 = SET PARITY ERROR
SET     LS_SETPARERR            ;SET FLAG FOR USE LATER
CALL    SET_PAR_INT             ;SET UP PARITY ERROR IN SPECIFIED
                                ; WCS LOCATION
ENDIF

IFMBI   COMMAND2,MM_SIZE        ;BIT 11 = PRINT MM SIZE
CALL    PRINT_MM_SIZE
ENDIF

IFMBI   COMMAND2,FPA_PRES       ;BIT 12 = PRINT FPA PRESENT
LXI     H,FPA_A
CALL    PRINT_MOD_PRES          ;PRINT PRESENT OR NOT PRESENT
ENDIF

IFMBI   COMMAND2,XFER           ;BIT 13 = DATA TRANSFER REQ
CALL    DATA_XFER
ENDIF

IFMBI   COMMAND2,EOS            ;BIT 14 = END OF SECTION
SET     EOS_FLG
ENDIF

IFMBI   COMMAND2,BEG_TST        ;BIT 15 = BEGINING OF TEST
CALL    BEGIN_OF_TEST
ENDIF

IFMBI   COMMAND1,SIGNAL         ;BIT 16 = SET UP OF SIGNAL AS
CALL    SET_SIGNALS
ENDIF

IFMBI   COMMAND0,LOOPING        ;SPECIAL CPU LOOPING CONTROL
```

```
                                                CALL    LOOPER
                                                ENDIF

                                                IFMBI   COMMANDO,SINGLE_ERR     ;BIT 26 = PRINT SINGLE BIT ERR
                                                CALL    PRINT_SINGLE
                                                ENDIF

                                                IFMBI   COMMANDO,CLEARPAR_ERR   ;BIT 27 = CLEAR PARITY ERROR
                                                CLEAR   LS_SETPARERR            ;CLEAR LS SET PARITY ERROR FLG
                                                CALL    CLEAR_PAR_INT           ;RESTORE GOOD PARITY AT ADDRESS
                                                                                ; SPECIFIED IN LS 7
                                                ENDIF

                                                IFMBI   COMMANDO,IDC_PRES       ;BIT 31 = PRINT IDC PRESENT
                                                LXI     H,IDC_A
                                                CALL    PRINT_MOD_PRES          ;PRINT PRESENT OR NOT PRESENT
                                                ENDIF

                                                RET


;*****************************************************************************************
;
;  WCS_ERROR
;  THIS ROUTINE IS EXECUTED WHEN THE COMMAND AND STATUS WORD INDICATES
;  THAT THERE HAS BEEN AN ERROR.  IF BELL IS SET THE BELL IS RUNG
;  IT ALSO DECODES THE MODULES NAME FROM THE BITS AS FOLLOWS
;
;       BIT 0 - WCS M8395
;           1 - CPU M8390
;           2 - MCT M8391
;           3 - FPA M8389
;           4 - MEMORY ARRAY CARD
;           5 - IDC M8388
;
;  THIS ROUTINE IS ALSO USED BY THE 8085 BASED TESTS.  IN THAT CASE, THE
;  CONSOLE_TEST FLAG IS SET AND SOME DIFFERENT BRANCHES ARE TAKEN.
;
;*****************************************************************************************

WCS_ERROR:      IFMBI   FLAGS,BELL_S_DEF
                MVI     A,7                     ;PRINT BELL IF BELL ON
                CALL    TYPE_CHAR               ; ERROR IS SET
                ENDIF

                MVI     A,2                     ;INDICATE ERROR
                STA     APT_MESSAGE_CODE+1      ;STORE IN MAILBOX

                LDA     CONSOLE_TEST            ;GET CONSOLE TEST FLAG
                ORA     A                       ;SET CONDITION CODES
                CZ      LOOP_ERROR              ;CHECK FOR LOOP ON ERROR IF
                                                ; NOT 8085 BASED TEST
                IFNMBI  FLAGS,NER_S_DEF
```

```
633E 5443                       *
633E 5444 0F 0209'3A            * ****************      IFN     CONSOLE_TEST            ;GET VALUES FROM LS IF NOT
          17AB'DA               * *
6345 5445                       * *                                                     ; 8085 BASED CONSOLE TEST
6345 5446 41 3E                 * *                     MVI     A,WCS_ERR_NUM           ;GET ERROR NUMBER FROM LS
6347 5447 1435'CD               * *                     CALL    READ_CS_R
634A 5448 00AC'3A               * *                     LDA     DATA3
634D 5449 003F'32               * *                     STA     CON_ERR_NUM             ;STORE IN MEM
6350 5450                       * *
6350 5451 0F 030B'3A            * * ***************     IF      PACK_ERROR              ;IF PACK ERROR FLAG SET
          175D'D2               * * *
6357 5452 18E4'CD               * * *                   CALL    PACK_LABEL              ;CALL ROUTINE TO READ EXPECTED
635A 5453                       * * *                                                   ; AND RECEIVED LABELS FROM LS
635A 5454 1773'C3               * * ***************     ELSE
635D 5455 42 3E                 * * *                   MVI     A,WCS_EXP_DAT           ;GET EXPECTED VALUE FROM LS
635F 5456 1435'CD               * * *                   CALL    READ_CS_R
6362 5457 00'40'11              * * *                   LXI     D,CON_EXP_DAT           ;DESTINATION
6365 5458 119F'CD               * * *                   CALL    MOV_FROM_DATA0          ;MOVE 4 BYTES OF DATA FROM
6368 5459                       * * *                                                   ; DATA0
6368 5460                       * * *
6368 5461 43 3E                 * * *                   MVI     A,WCS_REC_DAT           ;GET RECEIVED VALUE FROM LS
636A 5462 1435'CD               * * *                   CALL    READ_CS_R
636D 5463 00'4F'11              * * *                   LXI     D,CON_REC_DAT           ;DESTINATION
6370 5464 119F'CD               * * *                   CALL    MOV_FROM_DATA0          ;MOVE 4 BYTES OF DATA FROM
6373 5465                       * * *                                                   ; DATA0
6373 5466                       * * ***************     ENDIF
6373 5467                       * *
6373 5468 44 3E                 * *                     MVI     A,WCS_ADDR_DAT          ;GET OTHER DATA FROM LS
6375 5469 1435'CD               * *                     CALL    READ_CS_R
6378 5470 00'3B'11              * *                     LXI     D,CON_ADD_DAT           ;DESTINATION
637B 5471 119F'CD               * *                     CALL    MOV_FROM_DATA0          ;MOVE 4 BYTES OF DATA FROM
637E 5472                       * *                                                     ; DATA0
637E 5473                       * *
637E 5474 45 3E                 * *                     MVI     A,WCS_MSK_DAT           ;GET ERROR MASK FROM LS
6380 5475 1435'CD               * *                     CALL    READ_CS_R
6383 5476 00'4B'11              * *                     LXI     D,CON_MSK_DAT           ;DESTINATION
6386 5477 119F'CD               * *                     CALL    MOV_FROM_DATA0          ;MOVE 4 BYIES OF DATA FROM
6389 5478                       * *                                                     ; DATA0
6389 5479                       * *
6389 5480 005F'32 AF            * *                     CLEAR   NA_EXP_REC
638D 5481 80 E6 00B0'3A         * * ***************     IFMBI   COMMAND1,NA             ;SEE IF EXP AND REC ARE N/A
          179A'CA               * * *
6395 5482 005F'32 3C AF         * * *                   SET     NA_EXP_REC              ;SET FLAG IF SO
639A 5483                       * * ***************     ENDIF
639A 5484                       * *
639A 5485 0060'32 3C AF         * *                     SET     NA_OTHER
639F 5486 01 E6 00AF'3A         * * ***************     IFMBI   COMMAND0,ADDR           ;SEE IF OTHER DATA IS N/A
          17AB'CA               * * *
63A7 5487 0060'32 AF            * * *                   CLEAR   NA_OTHER                ;CLEAR FLAG IF NOT N/A
63AB 5488                       * * ***************     ENDIF
63AB 5489                       * *
63AB 5490                       * ****************      ENDIF
63AB 5491                       *
63AB 5492 0F 0038'3A            * ****************      IFN     APT                     ;NO ERROR PRINTOUT IF APT
          18B1'DA               * *
63B2 5493 0F 030B'3A            * * ***************     IF      PACK_ERROR
          17C5'D2               * * *
63B9 5494 02'47'21              * * *                   LXI     H,HEADER_A              ;PRINT HEADER FOR WRONG LABEL
63BC 5495 124A'CD               * * *                   CALL    PRINT_STRING_R
63BF 5496 00DE'CD               * * *                   CALL    CRLF
```

```
63C2 5497 17D1'C3                 * * **************  ELSE
63C5 5498 02'47'21                * * *
63C8 5499 124A'CD                 * * *               LXI     H,HEADER_A              ;PRINT HEADER LINE FOR ERROR
63CB 5500 02'65'21                * * *               CALL    PRINT_STRING_R
63CE 5501 124A'CD                 * * *               LXI     H,HEADER_B
63D1 5502                         * * *               CALL    PRINT_STRING_R
63D1 5503                         * * **************  ENDIF
63D1 5504 04'DF'21                * *
63D4 5505 06 0E                   * *                 LXI     H,SECTION_NAME          ;PRINT SECTION NAME LOADED
63D6 5506 1243'CD                 * *                 MVI     C,SEC_LEN
63D9 5507                         * *                 CALL    PRINTER
63D9 5508 1260'CD                 * *
63DC 5509                         * *                 CALL    SPACE_R
63DC 5510 03D9'3A                 * *
63DF 5511 005B'32                 * *                 LDA     TEST_NUM                ;PRINT TEST NUMBER
63E2 5512 11C2'CD                 * *                 STA     HEX_BUF
63E5 5513 1260'CD                 * *                 CALL    PRINT_HEX_1_R
63E8 5514                         * *                 CALL    SPACE_R
63E8 5515 003F'3A                 * *
63EB 5516 005B'32                 * *                 LDA     CON_ERR_NUM             ;ERROR NUMBER LOADED PREVIOUSLY
63EE 5517 11C2'CD                 * *                 STA     HEX_BUF
63F1 5518 1260'CD                 * *                 CALL    PRINT_HEX_1_R           ;PRINT ERROR NUMBER
63F4 5519                         * *                 CALL    SPACE_R
63F4 5520 0F 005F'3A              * * **************  IF      NA_EXP_REC              ;IF EXPECTED AND RECEIVED DATA
          180A'D2                 * * *
63FB 5521                         * * *                                           ; ARE NOT APPLICABLE
63FB 5522 02'E6'21                * * *               LXI     H,NA_A                  ;POINT TO ASCII N/A
63FE 5523 124A'CD                 * * *               CALL    PRINT_STRING_R          ;PRINT UNDER EXPECTED DATA
6401 5524 02'E6'21                * * *               LXI     H,NA_A                  ;POINT TO ASCII N/A
6404 5525 124A'CD                 * * *               CALL    PRINT_STRING_R          ;PRINT UNDER RECEIVED DATA
6407 5526                         * * *
6407 5527 1838'C3                 * * **************  ELSE
640A 5528                         * * *
640A 5529 0F 030B'3A              * * * ************  IF      PACK_ERROR
          1820'D2                 * * * *
6411 5530                         * * * *
6411 5531 04'41'21                * * * *             LXI     H,EXP_LABEL             ;POINT TO EXPECTED LABEL
6414 5532 124A'CD                 * * * *             CALL    PRINT_STRING_R
6417 5533                         * * * *
6417 5534 04'4E'21                * * * *             LXI     H,REC_LABEL             ;POINT TO RECEIVED LABEL
641A 5535 124A'CD                 * * * *             CALL    PRINT_STRING_R
641D 5536                         * * * *
641D 5537 1838'C3                 * * * ************  ELSE
6420 5538                         * * * *
6420 5539 00'40'21                * * * *             LXI     H,CON_EXP_DAT           ;POINT TO EXPECTED DATA
6423 5540 00'5B'11                * * * *             LXI     D,HEX_BUF
6426 5541 11A2'CD                 * * * *             CALL    MOVER_4_R
6429 5542 11B8'CD                 * * * *             CALL    PRINT_HEX_4             ;PRINT EXPECTED DATA
642C 5543                         * * * *
642C 5544 00'4F'21                * * * *             LXI     H,CON_REC_DAT           ;POINT TO RECEIVED DATA
642F 5545 00'5B'11                * * * *             LXI     D,HEX_BUF
6432 5546 11A2'CD                 * * * *             CALL    MOVER_4_R
6435 5547 11B8'CD                 * * * *             CALL    PRINT_HEX_4             ;PRINT RECEIVED DATA
6438 5548                         * * * *
6438 5549                         * * * ************  ENDIF
6438 5550                         * * **************  ENDIF
6438 5551                         * *
6438 5552 0F 030B'3A              * * **************  IFN     PACK_ERROR              ;DO NOT PRINT OTHER,MASK,
          18AE'DA                 * * *
643F 5553                         * * *                                           ; AND MODULE IF PACK_ERROR
```

```
643F 5554 0F 0060'3A                * * * ************  IF      NA_OTHER                ;IF OTHER DATA IS NOT
          184F'D2                   * * * *
6446 5555                           * * * *                                          ; APPLICABLE
6446 5556 02'E6'21                  * * * *             LXI     H,NA_A               ;POINT TO ASCII N/A
6449 5557 124A'CD                   * * * *             CALL    PRINT_STRING_R       ;PRINT UNDER OTHER DATA
644C 5558                           * * * *
644C 5559 185B'C3                   * * * ************  ELSE
644F 5560 00'3B'21                  * * * *             LXI     H,CON_ADD_DAT        ;POINT TO OTHER DATA
6452 5561 00'5B'11                  * * * *             LXI     D,HEX_BUF
6455 5562 11A2'CD                   * * * *             CALL    MOVER_4_R
6458 5563 11B8'CD                   * * * *             CALL    PRINT_HEX_4          ;PRINT OTHER DATA
645B 5564                           * * * ************  ENDIF
645B 5565                           * * *
645B 5566 00'4B'21                  * * *               LXI     H,CON_MSK_DAT        ;POINT TO ERROR MASK
645E 5567 00'5B'11                  * * *               LXI     D,HEX_BUF
6461 5568 11A2'CD                   * * *               CALL    MOVER_4_R
6464 5569 11B8'CD                   * * *               CALL    PRINT_HEX_4          ;PRINT ERROR MASK
6467 5570                           * * *
6467 5571 0F 0209'3A                * * * ************  IFN     CONSOLE_TEST
          18A6'DA                   * * * *
646E 5572 46 3E                     * * * *             MVI     A,WCS_MOD_DAT        ;PRINT MODULE NUMBER
6470 5573 1435'CD                   * * * *             CALL    READ_LS_R            ;READ FROM LS
6473 5574 00AC'3A                   * * * *             LDA     DATA3
6476 5575 03C5'32                   * * * *             STA     TEMP_BYTE
6479 5576 01'3B'21                  * * * *             LXI     H,PRT_MOD_A
647C 5577 03C6'22                   * * * *             SHLD    TEMP_WORD
647F 5578                           * * * *
647F 5579 C5 46 00'55'21            * * * * **********  DO      8                    ;EIGHT POSS. MODULE NAMES
          77 08 3E                  * * * * *
6487 5580                           * * * * *
6487 5581 03C5'3A                   * * * * *           LDA     TEMP_BYTE
648A 5582 0F                        * * * * *           RRC
648B 5583 03C5'32                   * * * * *           STA     TEMP_BYTE
648E 5584                           * * * * *
648E 5585 1897'D2                   * * * * * ********  IFC
6491 5586 11AD'CD                   * * * * * *         CALL    PRINT_FLG
6494 5587 1260'CD                   * * * * * *         CALL    SPACE_R
6497 5588                           * * * * * ********  ENDIF
6497 5589                           * * * * *
6497 5590 1575'CD                   * * * * *           CALL    INC_TEMPW_5          ;INCREMENT TO NEXT FLAG
649A 5591                           * * * * *
649A 5592 35 00'55'21               * * * * **********  ENDDO
          70 C1 1887'C2             * * * *
64A3 5593                           * * * *
64A3 5594 18AE'C3                   * * * ************  ELSE                       ;IF CONSOLE TEST
64A6 5595 00'46'21                  * * * *             LXI     H,CON_MOD_DAT
64A9 5596 05 0E                     * * * *             MVI     C,5
64AB 5597 1243'CD                   * * * *             CALL    PRINTER              ;PRINT MODULE #
64AE 5598                           * * * ************  ENDIF
64AE 5599                           * * **************  ENDIF
64AE 5600                           * *
64AE 5601 125A'CD                   * *                 CALL    CRLF_R
64B1 5602                           * *
64B1 5603                           * ****************  ENDIF
64B1 5604                           ******************  ENDIF
64B1 5605
64B1 5606 0012'32 AF                                    CLEAR   APT_ERROR_NUMBER     ;CLEAR UPPER BYTE OF MAILBOX
64B5 5607 003F'3A                                       LDA     CON_ERR_NUM          ;ERROR NUMBER LOADED PREVIOUSLY
64B8 5608 0013'32                                       STA     APT_ERROR_NUMBER+1   ;PUT IN APT MAIL BOX
64BB 5609
```

```
64BB 5610 005F'32 AF                                    CLEAR   NA_EXP_REC              ;CLR NA FLAG FOR CONSOLE TESTS
64BF 5611 0060'32 AF                                    CLEAR   NA_OTHER                ;CLR NA FLAG FOR CONSOLE TESTS
64C3 5612 030B'32 AF                                    CLEAR   PACK_ERROR              ;CLR WRONG RL02 PACK FLAG
64C7 5613
64C7 5614 08 E6 005A'3A         ******************      IFMBI   FLAGS,HALT_S_DEF
          18D2'CA               *
64CF 5615 1349'CD               *                       CALL    SET_FOR_CONT_R          ;HALT ON ERROR SET,STOP
64D2 5616                       ******************      ENDIF
64D2 5617
64D2 5618 0F 0345'3A            ******************      IF      SHORTEN
          18E3'D2               *
64D9 5619 03B3'32 AF            *                       CLEAR   TABLE_MODE              ;SHORTEN SET.. USE THIS TEST
64DD 5620 03D9'3A               *                       LDA     TEST_NUM                ; NUMBER
64E0 5621 03DB'32               *                       STA     TEST_NUM2
64E3 5622                       ******************      ENDIF
64E3 5623
64E3 5624 C9                                            RET
64E4 5625
64E4 5626
64E4 5627
64E4 5628
64E4 5629
64E4 5630
64E4 5631
64E4 5632
64E4 5633
64E4 5634
64E4 5635 8A 3E                                 PACK_LABEL:     MVI     A,EXPECT_LAB            ;GET FIRST 4 BYTES OF EXPECTED
64E6 5636 1435'CD                                               CALL    READ_LS_R               ; LABEL FROM LS
64E9 5637 04'42'11                                              LXI     D,EXP_LAB_DAT           ;DESTINATION
64EC 5638 119F'CD                                               CALL    MOV_FROM_DATA0          ;MOVE 4 BYTES OF DATA FROM
64EF 5639                                                                                       ; DATA0
64EF 5640
64EF 5641 8B 3E                                                 MVI     A,EXPECT_LAB+1          ;GET NEXT 4 BYTES OF EXPECTED
64F1 5642 1435'CD                                               CALL    READ_LS_R               ; LABEL FROM LS
64F4 5643 04'46'11                                              LXI     D,EXP_LAB_DAT+4         ;DESTINATION
64F7 5644 119F'CD                                               CALL    MOV_FROM_DATA0          ;MOVE 4 BYTES OF DATA FROM
64FA 5645                                                                                       ; DATA0
64FA 5646
64FA 5647 8C 3E                                                 MVI     A,EXPECT_LAB+2          ;GET LAST 4 BYTES OF EXPECTED
64FC 5648 1435'CD                                               CALL    READ_LS_R               ; LABEL FROM LS
64FF 5649 04'4A'11                                              LXI     D,EXP_LAB_DAT+8         ;DESTINATION
6502 5650 119F'CD                                               CALL    MOV_FROM_DATA0          ;MOVE 4 BYTES OF DATA FROM
6505 5651                                                                                       ; DATA0
6505 5652
6505 5653 8E 3E                                                 MVI     A,RECEIVE_LAB           ;GET FIRST 4 BYTES OF RECEIVED
6507 5654 1435'CD                                               CALL    READ_LS_R               ; LABEL FROM LS
650A 5655 04'4F'11                                              LXI     D,REC_LAB_DAT           ;DESTINATION
650D 5656 119F'CD                                               CALL    MOV_FROM_DATA0          ;MOVE 4 BYTES OF DATA FROM
6510 5657                                                                                       ; DATA0
6510 5658
6510 5659 8F 3E                                                 MVI     A,RECEIVE_LAB+1         ;GET NEXT 4 BYTES OF RECEIVED
6512 5660 1435'CD                                               CALL    READ_LS_R               ; LABEL FROM LS
6515 5661 04'53'11                                              LXI     D,REC_LAB_DAT+4         ;DESTINATION
6518 5662 119F'CD                                               CALL    MOV_FROM_DATA0          ;MOVE 4 BYTES OF DATA FROM
651B 5663                                                                                       ; DATA0
651B 5664
651B 5665 90 3E                                                 MVI     A,RECEIVE_LAB+2         ;GET LAST 4 BYTES OF RECEIVED
651D 5666 1435'CD                                               CALL    READ_LS_R               ; LABEL FROM LS
6520 5667 04'57'11                                              LXI     D,REC_LAB_DAT+8         ;DESTINATION
```

;***
;
; PACK_LABEL
; THIS ROUTINE READS THE EXPECTED AND RECEIVED RL02 PACK LABELS FROM LS.
; THIS ROUTINE IS ONLY USED IN ASSOCIATION WITH THE IDC PART II DIAGNOSTIC.
;
;***

```
6523 5668 119F'C3                       JMP     MOV_FROM_DATA0          ;MOVE 4 BYTES OF DATA FROM
6526 5669                                                               ; DATA0
6526 5670
6526 5671
6526 5672
6526 5673
6526 5674
6526 5675
6526 5676
6526 5677
6526 5678
6526 5679 06 3E                 PRINT_DRIVE:    MVI     A,LS_6
6528 5680 1435'CD                               CALL    READ_LS_R               ;READ SIZE WORD
652B 5681
652B 5682 02'19'21                              LXI     H,DRIVE_A               ;PRINT "DRIVE "
652E 5683 124A'CD                               CALL    PRINT_STRING_R
6531 5684
6531 5685 00AC'3A                               LDA     DATA3                   ;GET DRIVE NUMBER
6534 5686 005B'32                               STA     HEX_BUF
6537 5687 11C2'C3                               JMP     PRINT_HEX_1_R
653A 5688
653A 5689
653A 5690
653A 5691
653A 5692
653A 5693
653A 5694
653A 5695
653A 5696
653A 5697
653A 5698 1433'CD               PRINT_MM_SIZE:
                                                CALL    READ_LS_7               ;READ SIZE WORD
653D 5699
653D 5700 02'A0'21                              LXI     H,MEM_SIZE_A            ;PRINT THE SIZE OF MEMORY
6540 5701                                                                       ; IN 256K BLOCKS AS CONFIGURED
6540 5702 124A'CD                               CALL    PRINT_STRING_R          ; BY MICRO DIAGNOSTIC
6543 5703
6543 5704 00AC'3A                               LDA     DATA3
6546 5705 005B'32                               STA     HEX_BUF
6549 5706 11C2'C3                               JMP     PRINT_HEX_1_R
654C 5707
654C 5708
654C 5709
654C 5710
654C 5711
654C 5712
654C 5713
654C 5714
654C 5715
654C 5716 1433'CD               SET_PAR_INT:    CALL    READ_LS_7               ;READ ADDRESS TO SET BAD PARITY
654F 5717
654F 5718 04'3D'11                              LXI     D,INPUT_NUM             ;DESTINATION
6552 5719 119F'CD                               CALL    MOV_FROM_DATA0          ;MOVE 4 BYTES OF DATA FROM
6555 5720                                                                       ; DATA0
6555 5721
6555 5722 006E'32 3C AF                         SET     WCS_BAD_PARITY          ;CHANGE PARITY TO BAD
655A 5723 0F52'C3                               JMP     CHANGE_WCS              ;WRITE WCS WITH BAD PARITY
655D 5724
655D 5725
655D 5726
655D 5727
```

```
655D 5728
655D 5729
655D 5730
655D 5731
655D 5732
655D 5733
655D 5734
655D 5735
655D 5736 1433'CD
6560 5737
6560 5738 04'3D'11
6563 5739 119F'CD
6566 5740
6566 5741
6566 5742 0231'32 AF
656A 5743 0AB5'C3
656D 5744
656D 5745
656D 5746
656D 5747
656D 5748
656D 5749
656D 5750
656D 5751
656D 5752
656D 5753
656D 5754
656D 5755
656D 5756
656D 5757
656D 5758
656D 5759 1433'CD
6570 5760
6570 5761 03'4D'21
6573 5762 124A'CD
6576 5763
6576 5764 00A9'3A
6579 5765 005B'32
657C 5766
657C 5767 11C2'CD
657F 5768
657F 5769 02'2D'21
6582 5770 124A'CD
6585 5771
6585 5772 00'AA'21
6588 5773 00'5B'11
658B 5774 118E'CD
658E 5775
658E 5776 03 0E
6590 5777 11C4'CD
6593 5778
6593 5779 0F 3E
6595 5780 1435'CD
6598 5781
6598 5782 01'C2'21
659B 5783 124A'CD
659E 5784
659E 5785 00AC'3A
65A1 5786 005B'32
65A4 5787 11C2'C3
```

```
;  CLEAR_PAR_INT
;  THIS ROUTINE CLEARS A PARITY ERROR IN WCS ON REQUEST FROM THE CPU DIAG.
;  THIS ALSO ENABLES HALT ON PARITY ERROR
;  THE ADDRESS TO CLEAR THE PARITY ERROR ON IS IN LS 7
;
;*****************************************************************************

CLEAR_PAR_INT:
                CALL    READ_LS_7               ;READ ADDRESS TO SET GOOD PARITY

                LXI     D,INPUT_NUM             ;DESTINATION
                CALL    MOV_FROM_DATA0          ;MOVE 4 BYTES OF DATA FROM
                                                ; DATA0

                CLEAR   ERROR
                JMP     INT_CLEAR_PAR           ;SET THE PARITY GOOD


;*****************************************************************************
;
;  PRINT_SINGLE
;  THIS ROUTINE PRINTS THE NUMBER OF SINGLE BIT ERRORS IN A 256 K BLOCK
;  THIS INFORMATION IS SUPPLIED BY THE CPU DIAGNOSTIC IN LS 7.  THE ARRAY
;  BOARD NUMBER (FOR 64K RAM BOARDS ONLY) IS CONTAINED IN LS T15 (LOCATION
;  F)
;
;  LS 7   BITS 0  - 23  =  # SINGLE BIT ERRORS
;         BITS 24 - 31  =  256 KB BLOCK NUMBER
;
;*****************************************************************************

PRINT_SINGLE:   CALL    READ_LS_7               ;READ DATA FOR PRINTOUT

                LXI     H,SINGLE_A              ;PRINT BODY OF STRING
                CALL    PRINT_STRING_R

                LDA     DATA0                   ;MOVE 256 KB BLOCK NUMBER
                STA     HEX_BUF

                CALL    PRINT_HEX_1_R           ;PRINT BLOCK #

                LXI     H,EQUALS_A
                CALL    PRINT_STRING_R          ;PRINT " = "

                LXI     H,DATA1                 ;MOVE NUMBER ERRORS TO BUF
                LXI     D,HEX_BUF
                CALL    MOVER_3_R

                MVI     C,3
                CALL    PRINT_HEX_R             ;PRINT # SINGLE BIT ERRORS

                MVI     A,LS_F                  ;POINT TO STORAGE OF ARRAY
                CALL    READ_LS_R               ;GET ARRAY NUMBER FOR PRINTOUT

                LXI     H,ARRAY_NUM_A
                CALL    PRINT_STRING_R          ;PRINT "  ARRAY# "

                LDA     DATA3                   ;GET ARRAY NUMBER
                STA     HEX_BUF
                JMP     PRINT_HEX_1_R           ;PRINT ARRAY #
```

```
65A7 5788
65A7 5789
65A7 5790                              ;*****************************************************************************
65A7 5791                              ;
65A7 5792                              ;  PRINT_MOD_PRES
65A7 5793                              ;  THIS ROUTINE PRINTS WHETHER THE MICRO DIAGNOSITIC FINDS THE FPA, UBE, IDC
65A7 5794                              ;  OR R80
65A7 5795                              ;
65A7 5796                              ;*****************************************************************************
65A7 5797
65A7 5798                              PRINT_MOD_PRES:
65A7 5799 124A'CD                                      CALL    PRINT_STRING_R          ;PRINT MODULE NAME
65AA 5800 1433'CD                                      CALL    READ_LS_7               ;LS 7 CONTAINS 1 FOR MOD PRES
65AD 5801                                                                              ; AND 0 FOR NOT PRESENT
65AD 5802 02'CF'21                                     LXI     H,MOD_NOT_PRES_A        ;POINT TO MODULE NOT PRESENT
65B0 5803
65B0 5804 0F 00AC'3A            ******************     IF      DATA3                   ;IF MODULE IS PRESENT
          19BA'D2               *
65B7 5805 02'DC'21              *                      LXI     H,MOD_PRES_A            ;POINT TO MODULE PRESENT
65BA 5806                       ******************     ENDIF
65BA 5807
65BA 5808 124A'C3                                      JMP     PRINT_STRING_R
65BD 5809
65BD 5810                              ;*****************************************************************************
65BD 5811                              ;
65BD 5812                              ;  SET_SIGNALS
65BD 5813                              ;  THIS ROUTINE SETS UP THE SIGNALS AS THEY ARE DESCRIBED IN THE LS
65BD 5814                              ;  COMMAND AND STATUS WORD BITS
65BD 5815                              ;
65BD 5816                              ;*****************************************************************************
65BD 5817
65BD 5818 AF D3                        SET_SIGNALS:    OUT     CLRHLT                  ;CLEAR CONSOLE HALT
65BF 5819
65BF 5820 02 E6 00B0'3A         ******************     IFMBI   COMMAND1,CONHLT         ;BIT 17
          19C9'CA               *
65C7 5821 AE D3                 *                      OUT     SETHLT                  ;SET CONSOLE HALT
65C9 5822                       ******************     ENDIF
65C9 5823
65C9 5824 AD D3                                        OUT     CLRPFI                  ;CLEAR POWER FAIL INIT
65CB 5825
65CB 5826 04 E6 00B0'3A         ******************     IFMBI   COMMAND1,PWRFAIL        ;BIT 18
          19D5'CA               *
65D3 5827 AC D3                 *                      OUT     SETPFI                  ;SET POWER FAIL INIT
65D5 5828                       ******************     ENDIF
65D5 5829
65D5 5830 A5 D3                                        OUT     CLRTIM                  ;CLEAR INTERVAL TIMER INIT
65D7 5831
65D7 5832 08 E6 00B0'3A         ******************     IFMBI   COMMAND1,INVTIM         ;BIT 19
          19E1'CA               *
65DF 5833 A4 D3                 *                      OUT     SETTIM                  ;SET INTERVAL TIMER INIT
65E1 5834                       ******************     ENDIF
65E1 5835
65E1 5836 AB D3                                        OUT     CLRATN                  ;CLEAR CONSOLE ATTENTION
65E3 5837
65E3 5838 10 E6 00B0'3A         ******************     IFMBI   COMMAND1,CONATT         ;BIT 20
          19ED'CA               *
65EB 5839 AA D3                 *                      OUT     SETATN                  ;SET CONSOLE ATTENTION
65ED 5840                       ******************     ENDIF
65ED 5841
65ED 5842 0344'32 AF                                   CLEAR   SETACK_FLG              ;CLEAR FLAG TO CLEAR CONSOLE
```

```
ZZ-ENKAA-2.1    4C00    4                                                    Fiche 1  Frame J10          Sequence 126
65F1 5843                                                                                ; ACKNOWLEDGE BEFORE RETURNING
65F1 5844                                                                                ; TO WCS
65F1 5845
65F1 5846 20 E6 00B0'3A          ******************                IFMBI   COMMAND1,CONACK          ;BIT 21
          19FE'CA                *
65F9 5847 0344'32 3C AF          *                                 SET     SETACK_FLG               ;SET FLAG TO SET CONSOLE
65FE 5848                        *                                                                  ; ACKNOWLEDGE BEFORE RETURNING
65FE 5849                        *                                                                  ; TO WCS
65FE 5850                        ******************                ENDIF
65FE 5851
65FE 5852 029F'32 3C AF                                            SET     MEM_REQ                  ;ALLOW MEMORY REQEST WHEN
6603 5853
6603 5854 40 E6 00B0'3A          ******************                IFMBI   COMMAND1,EMEMREF         ;BIT 22
          1A0F'CA                *
660B 5855 029F'32 AF             *                                 CLEAR   MEM_REQ                  ;DONT ALLOW MEM REQ WHEN
660F 5856                        ******************                ENDIF
660F 5857
660F 5858 34 D3                                                    OUT     INITL                    ;CLEAR UNIBUS INIT
6611 5859
6611 5860 10 E6 00AF'3A          ******************                IFMBI   COMMAND0,UBS_INIT        ;BIT 28
          1A1B'CA                *
6619 5861 35 D3                  *                                 OUT     INITH                    ;SET UNIBUS INIT
661B 5862                        ******************                ENDIF
661B 5863
661B 5864 31 D3                                                    OUT     CLRDCL                   ;CLEAR UNIBUS DC LO
661D 5865
661D 5866 20 E6 00AF'3A          ******************                IFMBI   COMMAND0,UBS_DC_LO       ;BIT 29
          1A2F'CA                *
6625 5867 10F9'CD                *                                 CALL    NOP_CSR_R                ;LOAD CSR WITH A NOP
6628 5868 30 D3                  *                                 OUT     SETDCL                   ;SET UNIBUS DC LO
662A 5869 1570'CD                *                                 CALL    MICRO_STEP_CPU_R         ;CLOCK THE SIGNAL INTO MCT
662D 5870 31 D3                  *                                 OUT     CLRDCL                   ;CLEAR SIGNAL
662F 5871                        ******************                ENDIF
662F 5872
662F 5873 2E D3                                                    OUT     CLRBSY                   ;CLEAR UNIBUS BUSY
6631 5874
6631 5875 40 E6 00AF'3A          ******************                IFMBI   COMMAND0,UBS_BUSY        ;BIT 30
          1A3B'CA                *
6639 5876 2F D3                  *                                 OUT     SETBSY                   ;SET UNIBUS BUSY
663B 5877                        ******************                ENDIF
663B 5878
663B 5879 C9                                                       RET
663C 5880
663C 5881
663C 5882                                          ;*****************************************************************************
663C 5883                                          ;
663C 5884                                          ;  LOOPER
663C 5885                                          ;  THIS ROUTINE READS A COUNT FROM THE CPU AND CAUSES THE CPU TO LOOP
663C 5886                                          ;  THAT NUMBER OF TIMES BY SETTING THE ADDRESS TO ITS LOOP INSTRUCTION
663C 5887                                          ;  WHEN CALLED THE FIRST TIME THIS ROUTINE READS THE COUNT. ON ALL OTHER
663C 5888                                          ;  CALLS UNTIL THE COUNT = 0 THE COUNT IS DECREMENTED.  WHEN THE COUNT = 0
663C 5889                                          ;  THE ADDRESS IS SET FOR NO LOOPING AND THE NEXT CALL IS SET TO READ A
663C 5890                                          ;  NEW COUNT
663C 5891                                          ;
663C 5892                                          ;*****************************************************************************
663C 5893
663C 5894 0F 0283'3A             ****************** LOOPER:          IFN     HAVE_CNT
          1A53'DA                *
6643 5895                        *
6643 5896 47 3E                  *                                 MVI     A,LOOP_CNT_ADD           ;READ LOOP COUNT (ONCE ONLY)
```

```
6645 5897 1435'CD                *                  CALL    READ_LS_R
6648 5898 00AC'3A                *                  LDA     DATA3                   ;GET COUNT FROM DATA WORD
664B 5899 0293'32                *                  STA     LOOP_COUNTER            ;SAVE IT
664E 5900                        *
664E 5901 0283'32 3C AF          *                  SET     HAVE_CNT
6653 5902                        *
6653 5903                        ******************* ENDIF
6653 5904
6653 5905 02'93'21                                  LXI     H,LOOP_COUNTER
6656 5906 35                                        DCR     M                       ;DECREMENT LOOP COUNT
6657 5907 C0                                        RNZ                             ;EXIT IF COUNT NOT EXPIRED
6658 5908
6658 5909 00'C6'21                                  LXI     H,UPC_SUB               ;MOVE UPC ONE EXTRA WORD
665B 5910 1581'CD                                   CALL    INC_WORD_R              ;AND CAUSE IT TO START THERE
665E 5911
665E 5912 0283'32 AF                                CLEAR   HAVE_CNT                ;SET TO READ A NEW COUNT
6662 5913
6662 5914 C9                                        RET
6663 5915
6663 5916
6663 5917                                ;*******************************************************************************
6663 5918                                ;
6663 5919                                ;  LOOP_ERROR
6663 5920                                ;  THIS ROUTINE HANDLES LOOPING ON ERROR FOR THOSE TESTS IN WCS
6663 5921                                ;  THAT CAN NOT HANDLE IT.  LOOPING IS DONE BY STARTING THE CPU AT
6663 5922                                ;  THE NEXT ADDRESS.  NO LOOPING IS DONE BY MOVING THE UPC ADDRESS ONE
6663 5923                                ;  EXTRA WORD AND STARTING THERE
6663 5924                                ;
6663 5925                                ;*******************************************************************************
6663 5926
6663 5927 04 E6 00B1'3A          ****************** LOOP_ERROR:  IFMBI   COMMAND2,CON_LOOP       ;DO IF CPU HAS SIGNALED CONSOLE
          1A79'CA                *
666B 5928                        *                                                  ; TO CONTROL ERROR LOOPING
666B 5929                        *
666B 5930 01 E6 005A'3A          * ****************                IFNMBI  FLAGS,LOOP_S_DEF      ;IF LOOP ON ERROR IS NOT SET
          1A79'C2                * *
6673 5931 00'C6'21               * *                              LXI     H,UPC_SUB             ;GET UPC
6676 5932 1581'CD                * *                              CALL    INC_WORD_R            ;AND INCREMENT ONE EXTRA WORD
6679 5933                        * ****************               ENDIF
6679 5934                        *
6679 5935                        ******************               ENDIF
6679 5936
6679 5937 C9                                                      RET
667A 5938
667A 5939                                ;*******************************************************************************
667A 5940                                ;
667A 5941                                ;  SUB_ATTENTION
667A 5942                                ;  THIS ROUTINE IS USED AS AN IDLE ROUTINE WHEN EXECUTING A SUBROUTINE IN
667A 5943                                ;  WCS.  THERE IS NO READ OF THE COMMAND AND STATUS WORD AFTER AN ATTENTION
667A 5944                                ;  SIGNAL IS FOUND BECAUSE IT IS KNOWN THAT THIS MEANS END OF SUBROUTINE.
667A 5945                                ;  SEE ATTENTION ROUTINE FOR MORE INFO
667A 5946                                ;
667A 5947                                ;*******************************************************************************
667A 5948
667A 5949                                SUB_ATTENTION:
667A 5950 1BE6'CD                                   CALL    RESET_TIMOUT            ;RESET TIMOUT COUNTER
667D 5951
667D 5952 0F 0335'3A             ******************* WHILE   RUNNING
          1AB1'D2                *
6684 5953                        *
```

```
6684 5954 02C8'3A                *
6687 5955 B7                     *
6688 5956 1521'C4                *
668B 5957                        *
668B 5958 83 DB                  *
668D 5959 1F                     *
668E 5960                        *
668E 5961 1AA8'D2                * ****************
6691 5962 82 DB                  * *
6693 5963 01B5'32                * *
6696 5964 1BE6'CD                * *
6699 5965 1399'CD                * *
669C 5966                        * *
669C 5967 22 06                  * *
669E 5968 01B5'3A                * *
66A1 5969 1F                     * *
66A2 5970 0F79'DA                * *
66A5 5971                        * *
66A5 5972                        * *
66A5 5973                        * *
66A5 5974 1341'CD                * *
66A8 5975                        * *
66A8 5976                        * ****************
66A8 5977                        *
66A8 5978 4129 CD                *
66AB 5979 1BBB'CD                *
66AE 5980                        *
66AE 5981 1A7D'C3                ******************
66B1 5982
66B1 5983 C9                                                    RET
66B2 5984
66B2 5985
66B2 5986                                       ;*****************************************************************************
66B2 5987                                       ;
66B2 5988                                       ;  EOP
66B2 5989                                       ;  THIS ROUTINE IS EXECUTED AT END OF PASS
66B2 5990                                       ;
66B2 5991                                       ;*****************************************************************************
66B2 5992
66B2 5993 00'18'21                              EOP:            LXI     H,APT_PASS_CNT          ;GET PASS CNT IN MAILBOX
66B5 5994 1581'CD                                               CALL    INC_WORD_R              ;INCREMENT IT
66B8 5995
66B8 5996 02 E6 005A'3A          ******************             IFNMBI  FLAGS,NER_S_DEF         ;IF NER DONT PRINT EOP
          1ACF'C2                *
66C0 5997 02'27'21               *                              LXI     H,EOP_A                 ;PRINT "EOP"
66C3 5998 124A'CD                *                              CALL    PRINT_STRING_R
66C6 5999                        *
66C6 6000 0018'2A                *                              LHLD    APT_PASS_CNT
66C9 6001 005B'22                *                              SHLD    HEX_BUF
66CC 6002 11BD'CD                *                              CALL    PRINT_HEX_2             ;PRINT PASS COUNT
66CF 6003                        ******************             ENDIF
66CF 6004
66CF 6005 04'74'21                                              LXI     H,SEC_TABLE
66D2 6006 03B4'22                                               SHLD    TABLE_PNT               ;RESET TABLE POINTER
66D5 6007
66D5 6008 0056'32 AF                                            CLEAR   EOS_FLG                 ;RESET END OF SECTION
66D9 6009
66D9 6010 03'14'21               ******************             IFNEQ   PASS_CNT,F_WORD,2       ;-1 PASS CNT =INFINITE,SKIP OUT
          02 0E 00'58'11         *
          1B04'CA 00CF'CD        *
```

```
66E7 6011                       *
66E7 6012 03'14'21              *                       LXI     H,PASS_CNT
66EA 6013 158C'CD               *                       CALL    DECR_WORD_R             ;DECREMENT PASS COUNT
66ED 6014                       *
66ED 6015 03'14'21              * ****************      IFEQ    PASS_CNT,ZERO_WORD,2    ;IF PASS CNT = 0 CAUSE
          02 0E 04'1B'11        * *
          1B04'C2 00CF'CD       * *
66FB 6016                       * *                                                     ;RETURN TO PARSER
66FB 6017 0241'32 3C AF         * *                     SET     FILE_LOADED             ;CAUSE EXIT OF MAIN LOAD LOOP
6700 6018 023B'32 AF            * *                     CLEAR   EXECUTE
6704 6019                       * ****************      ENDIF
6704 6020                       *
6704 6021                       ******************      ENDIF
6704 6022
6704 6023 C9                                            RET
6705 6024
6705 6025                                       ;*******************************************************************************
6705 6026                                       ;
6705 6027                                       ;  BEGIN_OF_TEST
6705 6028                                       ;  THIS ROUTINE HANDLES BEGINING OF TEST SENT FROM THE CPU
6705 6029                                       ;  IF ALSO VERIFIES THE TEST SEQUENCE OF WCS TESTING
6705 6030                                       ;
6705 6031                                       ;*******************************************************************************
6705 6032
6705 6033                                       BEGIN_OF_TEST:
6705 6034 03'D9'21                                      LXI     H,TEST_NUM
6708 6035 34                                            INR     M                       ;INCREMENT TEST NUM
6709 6036
6709 6037 03'D9'21              ******************      IFGT    TEST_NUM,TEST_NUM2,1
          01 0E 03'DB'11        *
          1B1F'F2 00CF'CD       *
6717 6038 0056'32 3C AF         *                       SET     EOS_FLG                 ;FORCE END OF SECTION
671C 6039                       *
671C 6040 1B2C'C3               ******************      ELSE
671F 6041 03D9'3A               *                       LDA     TEST_NUM
6722 6042 0017'32               *                       STA     APT_TEST_NUMBER+1       ;STORE NEW TEST # IN MAILBOX
6725 6043 03DF'3A               *                       LDA     TRACE                   ;GET TRACE FLAG
6728 6044 B7                    *                       ORA     A                       ;SET CONDITION CODES
6729 6045 1B35'C4               *                       CNZ     PRINT_TRACE             ;PRINT SECTION AND TEST #
672C 6046                       *                                                       ; IF TRACE SET
672C 6047                       ******************      ENDIF
672C 6048
672C 6049 1B55'CD                                       CALL    VERIFY_SEQ
672F 6050 029F'32 3C AF                                 SET     MEM_REQ
6734 6051 C9                                            RET
6735 6052
6735 6053                                       ;*******************************************************************************
6735 6054                                       ;
6735 6055                                       ;  PRINT_TRACE
6735 6056                                       ;
6735 6057                                       ;  THIS ROUTINE PRINT THE CURRENT SECTION AND TEST NUMBER WHEN TRACE IS SET.
6735 6058                                       ;
6735 6059                                       ;*******************************************************************************
6735 6060
6735 6061 125A'CD                               PRINT_TRACE:    CALL    CRLF_R                  ;PRINT <CR><LF>
6738 6062 04'DF'21                                              LXI     H,SECTION_NAME          ;PRINT SECTION NAME LOADED
6738 6063 06 0E                                                 MVI     C,SEC_LEN
673D 6064 1243'CD                                               CALL    PRINTER
6740 6065 1260'CD                                               CALL    SPACE_R
6743 6066 1260'CD                                               CALL    SPACE_R
```

```
6746 6067 03'D3'21                                  LXI     H,TEST_A
6749 6068 124A'CD                                   CALL    PRINT_STRING_R          ;PRINT "TEST "
674C 6069 03D9'3A                                   LDA     TEST_NUM
674F 6070 005B'32                                   STA     HEX_BUF
6752 6071 11C2'C3                                   JMP     PRINT_HEX_1_R           ;PRINT TEST #
6755 6072
6755 6073                                   ;*******************************************************************************
6755 6074                                   ;
6755 6075                                   ;  VERIFY_SEQ
6755 6076                                   ;  THIS ROUTINE VERIFIES THE TESTS RUN IN SEQUENCE BY READING THE UPC
6755 6077                                   ;  AND COMPARING IT TO A KNOW VALUE
6755 6078                                   ;
6755 6079                                   ;*******************************************************************************
6755 6080
6755 6081 00C6'2A                           VERIFY_SEQ: LHLD    UPC_SUB
6758 6082 03CA'22                                   SHLD    TEMP_WORD2              ;SAVE GET UPC VALUE
675B 6083
675B 6084 C5 46 00'55'21        **********          DO      4
          77 04 3E              *
6763 6085                       *
6763 6086 03'CA'21              *                   LXI     H,TEMP_WORD2            ;MASK OFF WCS HIGH PAGE
6766 6087 7E                    *                   MOV     A,M
6767 6088 3F E6                 *                   ANI     HEX_3F
6769 6089 77                    *                   MOV     M,A
676A 6090 158C'CD               *                   CALL    DECR_WORD_R             ;INC VALUE BY 4
676D 6091                       *
676D 6092 35 00'55'21           **********          ENDDO
          70 C1 1B63'C2
6776 6093
6776 6094 006C'32 AF                                CLEAR   WCS_ADDRESS
677A 6095 03D9'3A                                   LDA     TEST_NUM                ;GET KNOW VALUE FOR THIS TEST
677D 6096 006D'32                                   STA     WCS_ADDRESS+1           ;FROM WCS
6780 6097 109D'CD                                   CALL    READ_WCS_R
6783 6098
6783 6099 C5 46 00'55'21        **********          DO      4
          77 04 3E              *
678B 6100 00'B6'21              *                   LXI     H,WCS_VALUE+2           ;POINT TO WCS DATA
678E 6101 03 0E                 *                   MVI     C,3
6790 6102 10F0'CD               *                   CALL    SHIFTER1_R              ;SHIFT 17-4(OF JUMP) TO 21-8
6793 6103 35 00'55'21           **********          ENDDO
          70 C1 1B8B'C2
679C 6104
679C 6105 00B4'3A                                   LDA     WCS_VALUE
679F 6106 3F E6                                     ANI     HEX_3F                  ;AND OUT TO ONLY ADDRESS
67A1 6107 00B4'32                                   STA     WCS_VALUE
67A4 6108
67A4 6109 03'CA'21              **********          IFNEQ   TEMP_WORD2,WCS_VALUE,2
          02 0E 00'B4'11        *
          1BBA'CA 00CF'CD       *
67B2 6110 23 06                 *                   MVI     B,<^X23>                ;SEQUENCE ERROR IN WCS TEST
67B4 6111 0F7E'CD               *                   CALL    ERROR_ROUTINE
67B7 6112 1349'CD               *                   CALL    SET_FOR_CONT_R          ;ALLOW CONTINUE
67BA 6113                       **********          ENDIF
67BA 6114
67BA 6115 C9                                        RET
67BB 6116
67BB 6117                                   ;*******************************************************************************
67BB 6118                                   ;
67BB 6119                                   ;  TIMOUT
67BB 6120                                   ;  THIS ROUTINE IS USED TO TELL IF THE CPU IS NOT RESPONDING
```

```
67BB 6121                               ;  IN A REASONABLE TIME PERIOD.  IF THE COUNTER OVERFLOWS A TIMOUT
67BB 6122                               ;  HAS OCCURED.
67BB 6123                               ;
67BB 6124                               ;*****************************************************************************
67BB 6125
67BB 6126                               TIMOUT:
67BB 6127 01 E6 005A'3A      **********                 IFNMBI  FLAGS,LOOP_S_DEF
          1BE2'C2            *
67C3 6128                    *
67C3 6129 03'DD'21           *                          LXI     H,TIMOUT_CNT          ;COUNTER FOR TIMING
67C6 6130 1581'CD            *                          CALL    INC_WORD_R            ;INCREMENT COUNTER
67C9 6131                    *
67C9 6132 03'DD'21           * **********               IFEQ    TIMOUT_CNT,ZERO_WORD,2
          02 0E 04'1B'11     * *
          1BDF'C2 00CF'CD    * *
67D7 6133 24 06              * *                        MVI     B,<^X24>              ;OVERFLOW, PRINT ERROR
67D9 6134 0F7E'CD            * *                        CALL    ERROR_ROUTINE
67DC 6135 1349'CD            * *                        CALL    SET_FOR_CONT_R        ;GO TO COMMAND LEVEL
67DF 6136                    * **********               ENDIF
67DF 6137                    *
67DF 6138 1BE5'C3            **********                 ELSE
67E2 6139 1341'CD            *                          CALL    CHECK_CNTLC           ;IF ERROR LOOPING, CHECK
67E5 6140                    *                                                        ; FOR ^C FROM OPERATOR
67E5 6141                    **********                 ENDIF
67E5 6142
67E5 6143 C9                                            RET
67E6 6144
67E6 6145                               ;*****************************************************************************
67E6 6146                               ;
67E6 6147                               ;  RESET_TIMOUT
67E6 6148                               ;  THIS ROUTINE RESETS THE TIMEOUT COUNTERS TO ZERO FOR A FULL
67E6 6149                               ;  TIMOUT PERIOD
67E6 6150                               ;
67E6 6151                               ;*****************************************************************************
67E6 6152
67E6 6153 03'DD'21                      RESET_TIMOUT:   LXI     H,TIMOUT_CNT
67E9 6154 AF                                            XRA     A
67EA 6155 77                                            MOV     M,A
67EB 6156 23                                            INX     H
67EC 6157 77                                            MOV     M,A
67ED 6158 C9                                            RET
67EE 6159
67EE 6160
67EE 6161                               ;*****************************************************************************
67EE 6162                               ;
67EE 6163                               ;  PARITY_ERR
67EE 6164                               ;  THIS ROUTINE GETS CONTROL ON A PARITY ERROR INTERRUPT
67EE 6165                               ;  THE INTERRUPT COULD ALSO BE KNOWN PARITY ERRORS SUCH AS A SOMM
67EE 6166                               ;
67EE 6167                               ;*****************************************************************************
67EE 6168
67EE 6169 F5                            PARITY_ERR:     PUSH    $PSW                  ;SAVE FLAGS AND ACCUMULATOR
67EF 6170 C5                                            PUSH    B                     ;SAVE BC PAIR
67F0 6171 D5                                            PUSH    D                     ;SAVE DE PAIR
67F1 6172 E5                                            PUSH    H                     ;SAVE HL PAIR
67F2 6173 1399'CD                                       CALL    STOP_CPU              ;STOP THE CLOCK
67F5 6174 0208'3A                                       LDA     CNT_PAR               ;GET COUNT OF PARITY ERRORS
67F8 6175 3C                                            INR     A                     ;INC COUNT
67F9 6176 0208'32                                       STA     CNT_PAR               ;UPDATE COUNT
67FC 6177
```

```
67FC 6178 02 FE                                    CPI     2                       ;SEE IF PARITY COUNT=2 (SECOND
67FE 6179                                                                          ; PARITY ERROR OCCURED)
67FE 6180 1C05'CA                                  JZ      1$                      ;DON'T TRY SAVE_WR IF SO
6801 6181
6801 6182 FB                                       EI                              ;REENABLE INTERRUPTS
6802 6183 0F47'CD                                  CALL    SAVE_WR
6805 6184 00C4'2A                            1$:   LHLD    SAVED_UPC
6808 6185 005B'22                                  SHLD    HEX_BUF                 ;SAVE UPC
680B 6186 00'5B'21                                 LXI     H,HEX_BUF               ;POINT TO STORED UPC
680E 6187 158C'CD                                  CALL    DECR_WORD_R             ;BACK UP UPC..PARITY ERROR
6811 6188                                                                          ; HAS ALREADY REACHED CSR
6811 6189
6811 6190 0F 039F'3A          ******************   IF      SOMM_FLG
          1C3C'D2             *
6818 6191                     *
6818 6192 00'5B'21            * ****************   IFEQ    HEX_BUF,SOMM_ADDRESS,2
          02 0E 03'9D'11      * *
          1C36'C2 00CF'CD     * *
6826 6193 0B45'CD             * *                  CALL    PRINT_UPC               ;PRINT VALUE OF UPC
6829 6194 03'97'21            * *                  LXI     H,SOMM_A                ;PRINT THAT THIS IS A SOMM
682C 6195 124A'CD             * *                  CALL    PRINT_STRING_R
682F 6196 FB                  * *                  EI                              ;REENABLE INTERRUPTS
6830 6197 1349'CD             * *                  CALL    SET_FOR_CONT_R          ;ISSUE PROMPT, ALLOW CONTINUE
6833 6198                     * *
6833 6199 1C39'C3             * ****************   ELSE
6836 6200                     * *
6836 6201 1C5C'CD             * *                  CALL    CHECK_LS_SETPAR         ;CHECK TO SEE IF PARITY ERROR
6839 6202                     * *                                                  ; WAS INTENTIONAL
6839 6203                     * ****************   ENDIF
6839 6204                     *
6839 6205 1C3F'C3             ******************   ELSE
683C 6206                     *
683C 6207 1C5C'CD             *                    CALL    CHECK_LS_SETPAR         ;CHECK TO SEE IF PARITY ERROR
683F 6208                     *                                                    ; WAS INTENTIONAL
683F 6209                     ******************   ENDIF
683F 6210
683F 6211 0F 0299'3A          ******************   IF      LS_PARERR_ACT
          1C4F'D2             *
6846 6212 00'72'21            *                    LXI     H,NOP_INST              ;SET UP TO PUT NOP IN CSR
6849 6213 00'BB'11            *                    LXI     D,SAVED_CSR
684C 6214 118E'CD             *                    CALL    MOVER_3_R
684F 6215                     ******************   ENDIF
684F 6216
684F 6217 0299'32 AF                               CLEAR   LS_PARERR_ACT           ;CLEAR FLG IN CASE IT IS SET
6853 6218 13DD'CD                                  CALL    RESTART_CPU
6856 6219
6856 6220 E1                                       POP     H                       ;RESTORE HL PAIR
6857 6221 D1                                       POP     D                       ;RESTORE DE PAIR
6858 6222 C1                                       POP     B                       ;RESTORE BC PAIR
6859 6223 F1                                       POP     $PSW                    ;RESTORE FLAGS AND ACCUMULATOR
685A 6224
685A 6225 FB                                       EI                              ;REENABLE INTERRUPTS
685B 6226 C9                                       RET
685C 6227
685C 6228                                  ;
685C 6229                                  ; THIS ROUTINE CHECKS TO SEE IF PARITY ERROR WAS CAUSED BY INTENTIONALLY
685C 6230                                  ; PUTTING A PARITY ERROR IN WCS VIA A COMMAND FROM THE CPU MICRO-CODE
685C 6231                                  ;
685C 6232
685C 6233                                  CHECK_LS_SETPAR:
```

```
685C 6234 0F 0298'3A        ******************          IF      LS_SETPARERR
          1C6B'D2           *
6863 6235 0299'32 3C AF     *                           SET     LS_PARERR_ACT           ;SET LS PARITY ERROR ACTIVE FLG
6868 6236                   *
6868 6237 1C77'C3           ******************          ELSE
686B 6238                   *
686B 6239 0B45'CD           *                           CALL    PRINT_UPC               ;PRINT VALUE OF UPC
686E 6240 26 06             *                           MVI     B,<^X26>                ;ERROR - WCS PARITY ERROR
6870 6241 0F7E'CD           *                           CALL    ERROR_ROUTINE
6873 6242 FB                *                           EI                              ;REENABLE INTERRUPTS
6874 6243 1349'CD           *                           CALL    SET_FOR_CONT_R          ;ALLOW CONTINUE FROM HERE
6877 6244                   *
6877 6245                   ******************          ENDIF
6877 6246 C9                                            RET                             ;BACK TO PAR ERR ROUTINE
6878 6247
6878 6248                   ;*************************************************************************************
6878 6249                   ;
6878 6250                   ;  DATA_XFER
6878 6251                   ;  THIS ROUTINE TRANSFERS DATA FROM THE WCS MEMORY TO LS OR MAIN MEMORY
6878 6252                   ;  A POINTER IS USED AND INCREMENTED AFTER EACH XFER.  THIS POINTER POINTS
6878 6253                   ;  TO ANOTHER POINTER THAT POINTS TO A DATA CONTROL BLOCK.  THE DATA
6878 6254                   ;  CONTROL BLOCK CONTAINS THE BYTE COUNT AND  DESTINATION ADDRESS OF THE
6878 6255                   ;  DATA.  THE POINTER IN WCS ALSO CONTAINS ONE BIT THAT TELLS IF THE
6878 6256                   ;  TRANSFER IS FOR LS OR MAIN MEMORY.
6878 6257                   ;
6878 6258                   ;*************************************************************************************
6878 6259
6878 6260 0213'32 3C AF     DATA_XFER:      SET     DATA_XFER_FLG           ;SET XFER FLAG
687D 6261
687D 6262 1433'CD                           CALL    READ_LS_7               ;READ THE POINTER
6880 6263
6880 6264 00'AB'21                          LXI     H,DATA2
6883 6265 7E                                MOV     A,M
6884 6266 3F E6                             ANI     HEX_3F
6886 6267 006C'32                           STA     WCS_ADDRESS
6889 6268 23                                INX     H
688A 6269 7E                                MOV     A,M
688B 6270 006D'32                           STA     WCS_ADDRESS+1           ;SAVE POINTER FOR READ
688E 6271
688E 6272 109D'CD                           CALL    READ_WCS_R              ;READ FIRST DATA WORD(COUNT)
6891 6273
6891 6274 00B5'2A                           LHLD    WCS_VALUE+1
6894 6275 0413'22                           SHLD    XFER_CNT                ;NUMBER OF BYTES TO XFER
6897 6276
6897 6277 0418'32 AF                        CLEAR   XFER_MM                 ;SET TRANSFER TO LS TO START
689B 6278
689B 6279 01 E6 00B4'3A     ******************          IFMBI   WCS_VALUE,MM_OR_LS
          1CA8'CA           *
68A3 6280 0418'32 3C AF     *                           SET     XFER_MM         ;TRANSFER IS TO MM, SET FLAG
68A8 6281                   ******************          ENDIF
68A8 6282
68A8 6283 00'6C'21                          LXI     H,WCS_ADDRESS
68AB 6284 1581'CD                           CALL    INC_WORD_R
68AE 6285 109D'CD                           CALL    READ_WCS_R              ;READ DEST ADDRESS
68B1 6286
68B1 6287 00'B4'21                          LXI     H,WCS_VALUE
68B4 6288 04'15'11                          LXI     D,XFER_DEST
68B7 6289 118E'CD                           CALL    MOVER_3_R               ;SAVE DEST ADDRESS
68BA 6290
68BA 6291 00'B4'21                          LXI     H,WCS_VALUE             ;SAVE DEST ADDRESS IN DATA
```

```
68BD 6292 00'AA'11                                LXI     D,DATA1                 ;AREA FOR WRITE TO LS ON
68C0 6293                                                                         ;MM XFER
68C0 6294 118E'CD                                 CALL    MOVER_3_R
68C3 6295 AF                                      XRA     A
68C4 6296 00A9'32                                 STA     DATA0                   ;ZERO OUT TOP BYTE
68C7 6297
68C7 6298 0F 0418'3A              ******************
          1CD3'D2                 *                 IF      XFER_MM
68CE 6299                         *
68CE 6300 05 3E                   *                 MVI     A,LS_5                  ;LS ADDRESS TO WRITE
68D0 6301 1415'CD                 *                 CALL    WRITE_LS_R              ;WITH MM ADDRESS
68D3 6302                         *
68D3 6303                         ******************  ENDIF
68D3 6304
68D3 6305 006C'2A                                 LHLD    WCS_ADDRESS
68D6 6306 0419'22                                 SHLD    XFER_PNT
68D9 6307
68D9 6308 02E5'32 3C AF           ******************  WHILER  MORE_DATA               ;LOOP CONTROL BYTE
          0F 02E5'3A              *
          1D0A'D2                 *
68E5 6309 1D0B'CD                 *                 CALL    FOUR_BYTES              ;READ LONG WORD OF DATA (WCS)
68E8 6310                         *
68E8 6311 0F 0418'3A              * ****************  IF      XFER_MM
          1CFD'D2                 * *
68EF 6312 06 3E                   * *               MVI     A,LS_6                  ;LS DATA TO WRITE
68F1 6313 1415'CD                 * *               CALL    WRITE_LS_R              ;WITH DATA FOR MM
68F4 6314                         * *
68F4 6315 00 54 21                * *               LXI     H,WCS_DE_MM
68F7 6316 0A49'CD                 * *               CALL    EXEC_SUB                ;DO THE SUBROUTINE
68FA 6317                         * *
68FA 6318 1D07'C3                 * ****************  ELSE
68FD 6319                         * *
68FD 6320 0417'3A                 * *               LDA     XFER_DEST+2             ;ADDRESS TO WRITE IN LS
6900 6321 1415'CD                 * *               CALL    WRITE_LS_R
6903 6322                         * *
6903 6323 04'17'21                * *               LXI     H,XFER_DEST+2           ;INC DEST POINTER FOR LS
6906 6324 34                      * *               INR     M                       ;WRITE(MM IS AUTOMATIC)
6907 6325                         * *
6907 6326                         * ****************  ENDIF
6907 6327                         *
6907 6328 1CDE'C3                 ******************  ENDWHILE
690A 6329
690A 6330 C9                                      RET
690B 6331
690B 6332
690B 6333
690B 6334
690B 6335
690B 6336
690B 6337
690B 6338
690B 6339
690B 6340
690B 6341
690B 6342
690B 6343
690B 6344 1D30'CD                 FOUR_BYTES:     CALL    READ_WCS_DATA           ;GET 2 BYTES OF DATA
690E 6345 00AB'22                                 SHLD    DATA2
6911 6346
6911 6347 1D30'CD                                 CALL    READ_WCS_DATA           ;GET 2 BYTES OF DATA
```

```
;*****************************************************************************
;
;  FOUR_BYTES
;  THIS ROUTINE READS FOUR BYTES OF DATA FROM WCS USING XFER_PNT
;  THE DATA IS MOVED TO THE DATA0,1,2,3 AREAS.  IF THE COUNT XFER_CNT
;  RUNS OUT THE LOOP CONTROL FLAG MORE_DATA IS CLEARED
;
;*****************************************************************************
```

```
6914 6348 00A9'22                                       SHLD    DATA0
6917 6349
6917 6350 04'13'21                                      LXI     H,XFER_CNT
691A 6351 158C'CD                                       CALL    DECR_WORD_R
691D 6352
691D 6353 04'13'21              ******************      IFEQ    XFER_CNT,ZERO_WORD,2
          02 0E 04'1B'11        *
          1D2F'C2 00CF'CD       *
692B 6354                       *
692B 6355 02E5'32 AF            *                       CLEAR   MORE_DATA
692F 6356                       *
692F 6357                       ******************      ENDIF
692F 6358
692F 6359 C9                                            RET
6930 6360
6930 6361
6930 6362                                       READ_WCS_DATA:
6930 6363 04'19'21                                      LXI     H,XFER_PNT              ;INC POINTER AND READ DATA
6933 6364 1581'CD                                       CALL    INC_WORD_R              ;TO TRANSFER TO WCS
6936 6365 0419'2A                                       LHLD    XFER_PNT
6939 6366 006C'22                                       SHLD    WCS_ADDRESS             ;HIGH VALUE
693C 6367 109D'CD                                       CALL    READ_WCS_R
693F 6368
693F 6369 00B5'2A                                       LHLD    WCS_VALUE+1
6942 6370 C9                                            RET
6943 6371
6943 6372
6943 6373                       ;*****************************************************************************
6943 6374                       ;
6943 6375                       ;  CLEAR_CPU_ATT_R
6943 6376                       ;  THIS ROUTINE EXECUTES AN INSTRUCTION IN THE CSR THAT CLEARS
6943 6377                       ;  BOTH CPU ATTENTION AND CPU ACKNOWLEDGE
6943 6378                       ;
6943 6379                       ;*****************************************************************************
6943 6380
6943 6381                       CLEAR_CPU_ATT_R:
6943 6382 1148'CD                               CALL    LOAD_UPC_R              ;SAVE UPC IN ATT_UPC_SAVE
6946 6383 00'C2'21                              LXI     H,UPC_VALUE
6949 6384 00'CC'11                              LXI     D,ATT_UPC_SAVE
694C 6385 118E'CD                               CALL    MOVER_3_R
694F 6386
694F 6387 04'26'21                              LXI     H,X_CLR_CPU_ATT         ;ADDRESS OF INSTRUCTION
6952 6388 15C2'CD                               CALL    PERFORM_CSR_R
6955 6389
6955 6390 00'CC'21                              LXI     H,ATT_UPC_SAVE
6958 6391 00'C2'11                              LXI     D,UPC_VALUE
695B 6392 118E'CD                               CALL    MOVER_3_R
695E 6393 1148'C3                               JMP     LOAD_UPC_R
6961 6394
6961 6395
6961 6396                       ;*****************************************************************************
6961 6397                       ;
6961 6398                       ;  LOAD_SECTION
6961 6399                       ;  THIS ROUTINE LOADS A SECTION INTO WCS OR INTO CONSOLE RAM
6961 6400                       ;  IF THE LOAD IS TO WCS A FLAG IS SET AND THE DATA TRANSFER BRANCH
6961 6401                       ;  AT ADDRESS 0 OF THE WCS IS TAKEN.
6961 6402                       ;
6961 6403                       ;*****************************************************************************
6961 6404
6961 6405                       MOV_LOAD_SECTION:
```

```
6961 6406 03'CC'21                                  LXI   H,TEMP_FLAG              ;POINTER TO TEMP SECTION BUFFER
6964 6407 04'DE'11                                  LXI   D,SECTION_FLAG           ;POINTER TO SECTION BUFFER
6967 6408 07 0E                                     MVI   C,SEC_LEN+1              ;COUNT
6969 6409 11A4'CD                                   CALL  MOVER_R                  ;MOVE ENTRIES TO LOCAL BUFFER
696C 6410
696C 6411 04'DF'21                    LOAD_SECTION: LXI   H,SECTION_NAME
696F 6412 41 07 11                                  LXI   D,SEC_PARMB_NAM
6972 6413 06 0E                                     MVI   C,SEC_LEN                ;MOVE SECTION NAME TO TU58
6974 6414 11A4'CD                                   CALL  MOVER_R                  ;PARAMETER BLOCK
6977 6415
6977 6416 02'3C'21                                  LXI   H,EXT_A                  ;MOVE EXT TO PARM BLOCK
697A 6417 41 0D 11                                  LXI   D,SEC_PARMB_EXT
697D 6418 11A2'CD                                   CALL  MOVER_4_R
6980 6419
6980 6420 AF                                        XRA   A                        ;CLEAR FLAGS
6981 6421 4113 32                                   STA   SEC_PARMB_ADD+2          ;CLEAR 2 MOST SIGNIFICANT BYTES
6984 6422 4114 32                                   STA   SEC_PARMB_ADD+3          ; OF SEC_PARMB_ADD (UNUSED)
6987 6423
6987 6424 0240'32                                   STA   FAST_WRITE               ;CLEAR FAST_WRITE
698A 6425 0408'32                                   STA   WCS_LOADED               ;CLEAR WCS_LOADED
698D 6426 0209'32                                   STA   CONSOLE_TEST             ;CLEAR CONSOLE_TEST
6990 6427
6990 6428 80 E6 04DE'3A                             IFMBI SECTION_FLAG,CON_RAM
          1DA3'CA
6998 6429
6998 6430 06 3E                                     MVI   A,6                      ;DRIVER HAS 6 = RAM LOAD
699A 6431 4115 32                                   STA   SEC_PARMB_DST            ;STORE IN DESTINATION CODE
699D 6432 70 00 21                                  LXI   H,RAM_LOAD_ADD
69A0 6433
69A0 6434 1DAB'C3                                   ELSE
69A3 6435
69A3 6436 07 3E                                     MVI   A,7                      ;DRIVER HAS 7 = WCS LOAD
69A5 6437 4115 32                                   STA   SEC_PARMB_DST            ;STORE IN DESTINATION CODE
69A8 6438 00 00 21                                  LXI   H,WCS_LOAD_ADD
69AB 6439
69AB 6440                                           ENDIF
69AB 6441
69AB 6442 4111 22                                   SHLD  SEC_PARMB_ADD            ;STORE LOAD ADDR IN PARM BLOCK
69AE 6443
69AE 6444 4150 3A                                   LDA   SEC_PARMB_DEF            ;GET DEFAULT TU58 UNIT (THIS
69B1 6445                                                                          ; UNIT)
69B1 6446 4106 32                                   STA   SEC_PARMB_UNIT           ;PARM BLOCK ADDRESS FOR UNIT
69B4 6447
69B4 6448 4117 CD                                   CALL  TU58_DRIVER              ;GET THE FILE
69B7 6449 A7 D3                                     OUT   DISMEM                   ;DISABLE MEM REF (TU58 DRIVER
69B9 6450                                                                          ; ENABLED IT AFTER LOAD)
69B9 6451
69B9 6452 00 FE 4116 3A                             IFNEQI SEC_PARMB_ERR,0         ;CHECK FOR TU58 ERROR
          1DD0'CA
69C1 6453
69C1 6454 AF                                        XRA   A                        ;CLEAR FLAGS
69C2 6455 0240'32                                   STA   FAST_WRITE               ;CLEAR FAST_WRITE
69C5 6456 0408'32                                   STA   WCS_LOADED               ;CLEAR WCS_LOADED
69C8 6457 023B'32                                   STA   EXECUTE                  ;CLEAR EXECUTE TO ALLOW PARSER
69CB 6458                                                                          ; TO PROMPT
69CB 6459 25 06                                     MVI   B,<^X25>
69CD 6460 0F79'C3                                   JMP   ERROR_ROUTINE_NO_RET     ;TU58 ERROR.  DON'T RETURN
69D0 6461                                           ENDIF
69D0 6462
69D0 6463 0241'32 3C AF                             SET   FILE_LOADED              ;SECTION WAS LOADED OK
```

```
                      ******************
                      *
                      *
                      *
                      *
                      *
                      ******************

                      ******************
                      *
                      *
                      *
                      *
                      *
                      ******************

                      ******************
                      *
                      *
                      *
                      *
                      *
                      ******************
```

```
69D5 6464
69D5 6465 80 E6 04DE'3A       ******************          IFMBI   SECTION_FLAG,CON_RAM
          1DE5'CA             *
69DD 6466 0209'32 3C AF       *                           SET     CONSOLE_TEST
69E2 6467                     *
69E2 6468 1DF0'C3             ******************          ELSE
69E5 6469 0408'32 3C AF       *                           SET     WCS_LOADED
69EA 6470 1E8A'CD             *                           CALL    EXEC_TEST0
69ED 6471 10F9'CD             *                           CALL    NOP_CSR_R
69F0 6472                     *
69F0 6473                     ******************          ENDIF
69F0 6474
69F0 6475 C9                                              RET
69F1 6476
69F1 6477
69F1 6478
69F1 6479
69F1 6480
69F1 6481
69F1 6482
69F1 6483
69F1 6484
69F1 6485
69F1 6486
69F1 6487 4150 3A                                         DIR_COMMAND:    LDA     SEC_PARMB_DEF           ;GET DEFAULT TU58 UNIT (THIS
69F4 6488                                                                                                 ; UNIT)
69F4 6489 4106 32                                                         STA     SEC_PARMB_UNIT          ;PARM BLOCK ADDRESS FOR UNIT
69F7 6490 411A C3                                                         JMP     DIR_VECTOR              ;PRINT DIRECTORY
69FA 6491
69FA 6492
69FA 6493
69FA 6494
69FA 6495
69FA 6496
69FA 6497
69FA 6498
69FA 6499
69FA 6500
69FA 6501
69FA 6502 0240'32 3C AF                                   VERIFY_WRITER:  SET     FAST_WRITE              ;ERROR WILL CLEAR FAST WRITE
69FF 6503 06 00 21                                                        LXI     H,WCS_DAT_32_ADD
6A02 6504 7C                                                              MOV     A,H
6A03 6505 65                                                              MOV     H,L
6A04 6506 6F                                                              MOV     L,A
6A05 6507 00C2'22                                                         SHLD    UPC_VALUE
6A08 6508 1148'CD                                                         CALL    LOAD_UPC_R
6A0B 6509
6A0B 6510 10F9'CD                                                         CALL    NOP_CSR_R
6A0E 6511 04'E5'21                                                        LXI     H,WRITE_TAB
6A11 6512 006F'22                                                         SHLD    WRITE_TAB_PNT
6A14 6513 AB D3                                                           OUT     CLRATN
6A16 6514
6A16 6515 C5 46 00'55'21      ******************          DO      5
          77 05 3E            *
6A1E 6516                     *
6A1E 6517 1570'CD             *                           CALL    MICRO_STEP_CPU_R
6A21 6518 1E57'CD             *                           CALL    CHECK_UPC
6A24 6519                     *
6A24 6520 0240'3A             *                           LDA     FAST_WRITE              ;GET FAST WRITE FLAG
6A27 6521 B7                  *                           ORA     A                       ;SET CONDITION CODES
```

```
6A28 6522 1E55'CA                *                      JZ      LEAVE_DO                ;LEAVE DO LOOP IF FAST_WRITE
6A2B 6523                        *                                                      ; CLEAR
6A2B 6524 35 00'55'21            ******************     ENDDO
          70 C1 1E1E'C2
6A34 6525
6A34 6526 AA D3                                         OUT     SETATN
6A36 6527
6A36 6528 C5 46 00'55'21         ******************     DO      3
          77 03 3E               *
6A3E 6529                        *
6A3E 6530 1570'CD                *                      CALL    MICRO_STEP_CPU_R
6A41 6531 1E57'CD                *                      CALL    CHECK_UPC
6A44 6532                        *
6A44 6533 0240'3A                *                      LDA     FAST_WRITE              ;GET FAST WRITE FLAG
6A47 6534 B7                     *                      ORA     A                       ;SET CONDITION CODES
6A48 6535 1E55'CA                *                      JZ      LEAVE_DO                ;LEAVE DO LOOP IF FAST_WRITE
6A4B 6536                        *                                                      ; CLEAR
6A4B 6537 35 00'55'21            ******************     ENDDO
          70 C1 1E3E'C2
6A54 6538
6A54 6539 C9                                            RET
6A55 6540
6A55 6541 C1                                 LEAVE_DO:  POP     B                       ;POP STACK SINCE DO-LOOP DID
6A56 6542                                                                              ; A PUSH
6A56 6543 C9                                            RET                            ;AND RETURN
6A57 6544
6A57 6545                                    ;*****************************************************************************
6A57 6546                                    ;
6A57 6547                                    ;  CHECK_UPC:
6A57 6548                                    ;  THIS ROUTINE COMPARE THE UPC VALUE FROM THE TABLE AND THE VALUE IN
6A57 6549                                    ;  THE UPC AT THE TIME IT IS CALLED.
6A57 6550                                    ;
6A57 6551                                    ;  INPUTCONDITION:
6A57 6552                                    ;       WRITE_TAB_PNT = POINTS TO THE TABLE VALUE
6A57 6553                                    ;
6A57 6554                                    ;  OUTPUT CONDITION:
6A57 6555                                    ;       WRITE_TAB_PNT IS INCREMENTED TO THE NEXT TABLE VALUE
6A57 6556                                    ;
6A57 6557                                    ;*****************************************************************************
6A57 6558
6A57 6559 1148'CD                            CHECK_UPC: CALL    LOAD_UPC_R
6A5A 6560 006F'2A                                      LHLD    WRITE_TAB_PNT
6A5D 6561 7E                                           MOV     A,M
6A5E 6562 00CB'32                                      STA     UPC_COMPARE+1
6A61 6563 23                                           INX     H
6A62 6564 7E                                           MOV     A,M
6A63 6565 00CA'32                                      STA     UPC_COMPARE
6A66 6566 23                                           INX     H
6A67 6567 006F'22                                      SHLD    WRITE_TAB_PNT
6A6A 6568
6A6A 6569 00'CA'21               ******************    IFNEQ   UPC_COMPARE,UPC_VALUE,2
          02 0E 00'C2'11         *
          1E87'CA 00CF'CD        *
6A78 6570 29 06                  *                     MVI     B,<^X29>
6A7A 6571 0F7E'CD                *                     CALL    ERROR_ROUTINE
6A7D 6572 03'7D'21               *                     LXI     H,SLOW_WRITE_A          ;DOING SLOW WRITE
6A80 6573 124A'CD                *                     CALL    PRINT_STRING_R          ;PRINT MESSAGE
6A83 6574 0240'32 AF             *                     CLEAR   FAST_WRITE
6A87 6575                        *
6A87 6576                        ******************    ENDIF
```

```
6A87 6577
6A87 6578 1148'C3                                   JMP     LOAD_UPC_R
6A8A 6579
6A8A 6580
6A8A 6581
6A8A 6582
6A8A 6583
6A8A 6584
6A8A 6585
6A8A 6586
6A8A 6587
6A8A 6588
6A8A 6589 2A 06                                     MVI     B,<^X2A>                ;ERROR CODE IF WCS NOT LOADED
6A8C 6590 0F5E'CD                                   CALL    CHECK_WCS_LOADED        ;CHECK FOR WCS LOADED SET
6A8F 6591                                                                           ; RETURN IF OK
6A8F 6592
6A8F 6593 0F 02F2'3A                                IFN     NOINIT                  ;ELSE CHECK NO INIT FLAG
          1EC8'DA
6A96 6594
6A96 6595 1DFA'CD                                   CALL    VERIFY_WRITER           ;CHECK DATA XFER ROUTINE
6A99 6596
6A99 6597 AF                                        XRA     A
6A9A 6598 03A1'32                                   STA     STARTING_UPC            ;CLEAR STARTING_UPC
6A9D 6599 03A2'32 AF                                CLEAR   STARTING_UPC+1          ;SET TO EXECUTE DATA XFER
6AA1 6600
6AA1 6601 0213'32 3C AF                             WHILER  DATA_XFER_FLG           ;FLAG SET IF DATA XFER EVENT
          0F 0213'3A
          1EC8'D2                                                                   ;LOOP STOPS IF NO EVENT
6AAD 6602
6AAD 6603 13CE'CD                                   CALL    START_CPU
6AB0 6604 03DC'32 3C AF                             SET     TEST_ZERO               ;SET FOR NO RESTART
6AB5 6605 15F0'CD                                   CALL    ATTENTION               ;WAIT FOR CPU ATTENTIONS
6AB8 6606 03DC'32 AF                                CLEAR   TEST_ZERO
6ABC 6607 0F41'CD                                   CALL    RESTORE_WR
6ABF 6608
6ABF 6609 00C6'2A                                   LHLD    UPC_SUB                 ;SET FOR CONTINUE AT NEXT ADDR.
6AC2 6610 03A1'22                                   SHLD    STARTING_UPC
6AC5 6611
6AC5 6612 1EA6'C3                                   ENDWHILE
6AC8 6613
6AC8 6614                                           ENDIF
6AC8 6615
6AC8 6616 C9                                        RET
6AC9 6617
6AC9 6618
6AC9 6619
6AC9 6620
6AC9 6621
6AC9 6622
6AC9 6623
6AC9 6624
6AC9 6625
6AC9 6626 03'A5'21                  SYNTAX_ERROR:   LXI     H,SYNTAX_A
6ACC 6627 124A'CD                                   CALL    PRINT_STRING_R
6ACF 6628
6ACF 6629 AF                                        XRA     A                       ;CLEAR FLAGS
6AD0 6630 02F6'32                                   STA     NOT_FOUND               ;CLEAR NOT_FOUND
6AD3 6631 023B'32                                   STA     EXECUTE                 ;CLEAR EXECUTE SO PARSER WILL
6AD6 6632                                                                           ; PROMPT FOR COMMAND
6AD6 6633 C9                                        RET
```

```
6AD7 6634
6AD7 6635
6AD7 6636
6AD7 6637
6AD7 6638
6AD7 6639
6AD7 6640
6AD7 6641
6AD7 6642
6AD7 6643
6AD7 6644 06'D6'21
6ADA 6645 23'08'11
6ADD 6646 02F0'32 AF
6AE1 6647 02F0'3A
6AE4 6648 86
6AE5 6649 23
6AE6 6650 02F0'32
6AE9 6651 7C
6AEA 6652 BA
6AEB 6653 1EE1'C2
6AEE 6654 7D
6AEF 6655 BB
6AF0 6656 1EE1'C2
6AF3 6657 C9
6AF4 6658
6AF4 6659
6AF4 6660
6AF4 6661
6AF4 6662
6AF4 6663
6AF4 6664
6AF4 6665
6AF4 6666
6AF4 6667
6AF4 6668 030F'32 3C AF
6AF9 6669 C9
6AFA 6670
6AFA 6671
6AFA 6672
6AFA 6673
6AFA 6674
6AFA 6675
6AFA 6676
6AFA 6677
6AFA 6678
6AFA 6679
6AFA 6680
6AFA 6681 0336'32
6AFD 6682 00 3E
6AFF 6683 17
6B00 6684 0339'32
6B03 6685 033D'22
6B06 6686 EB
6B07 6687 033B'22
6B0A 6688 60
6B0B 6689 69
6B0C 6690 0337'22
6B0F 6691 00 00 21
6B12 6692 39
6B13 6693 033F'22
```

```
;*****************************************************************************
;
;  CHECKSUM
;  THIS ROUTINE CHECKSUMS THE PROGRAM AREA  (NOT THE VARIABLE AREA)
;  THE CHECKSUM IS LEFT IN NEWSUM
;
;*****************************************************************************

CHECKSUM:       LXI     H,BEGIN_CHECKSUM
                LXI     D,END_CHECKSUM
                CLEAR   NEWSUM
1$:             LDA     NEWSUM
                ADD     M
                INX     H
                STA     NEWSUM
                MOV     A,H
                CMP     D
                JNZ     1$
                MOV     A,L
                CMP     E
                JNZ     1$
                RET


;*****************************************************************************
;
;  SET_PARDONE
;  THIS ROUTINE SETS THE FLAG PARDONE SO THAT RE-ENTRANT PARSING
;  ROUTINES(SUCH AS DIAGNOSE) KNOW THEY HAVE HAVE END OF LINE.
;
;*****************************************************************************

SET_PARDONE:    SET     PARDONE
                RET


;*****************************************************************************
;
;  BREAK
;  THIS ROUTINE IS EXECUTED TO SAVE THE VALUES OF THE REGISTERS
;  AND ALLOW EXAMINES OF THE MEMORY.  THE ROUTINE RETURNS TO THE PARSER
;  ON EXIT.   THIS IS FOR MAINT. ONLY
;
;*****************************************************************************

BREAK:          STA     SAVE_A                  ;SAVE ACCUMULATOR
                MVI     A,0
                RAL
                STA     SAVE_C                  ;SAVE CARRY
                SHLD    SAVE_HL                 ;SAVE H+L
                XCHG
                SHLD    SAVE_DE                 ;SAVE D+E
                MOV     H,B
                MOV     L,C
                SHLD    SAVE_BC                 ;SAVE B+C
                LXI     H,0
                DAD     $SP
                SHLD    SAVE_SP                 ;SAVE STACK POINTER
```

```
6B16 6694
6B16 6695 01'DD'21                      LXI     H,BREAK_A
6B19 6696 124A'CD                       CALL    PRINT_STRING_R
6B1C 6697
6B1C 6698 0336'3A                       LDA     SAVE_A                  ;PRINT VALUE OF A
6B1F 6699 005B'32                       STA     HEX_BUF
6B22 6700 11C2'CD                       CALL    PRINT_HEX_1_R
6B25 6701
6B25 6702 0339'3A                       LDA     SAVE_C                  ;PRINT VALUE OF CARRY
6B28 6703 005B'32                       STA     HEX_BUF
6B2B 6704 11C2'CD                       CALL    PRINT_HEX_1_R
6B2E 6705
6B2E 6706 0337'2A                       LHLD    SAVE_BC                 ;PRINT VALUE OF BC PAIR
6B31 6707 005B'22                       SHLD    HEX_BUF
6B34 6708 11BD'CD                       CALL    PRINT_HEX_2
6B37 6709
6B37 6710 033B'2A                       LHLD    SAVE_DE                 ;PRINT VALUE OF DE PAIR
6B3A 6711 005B'22                       SHLD    HEX_BUF
6B3D 6712 11BD'CD                       CALL    PRINT_HEX_2
6B40 6713
6B40 6714 033D'2A                       LHLD    SAVE_HL                 ;PRINT VALUE OF HL PAIR
6B43 6715 005B'22                       SHLD    HEX_BUF
6B46 6716 11BD'CD                       CALL    PRINT_HEX_2
6B49 6717
6B49 6718 033F'2A                       LHLD    SAVE_SP                 ;PRINT VALUE OF SP PAIR
6B4C 6719 005B'22                       SHLD    HEX_BUF
6B4F 6720 11BD'CD                       CALL    PRINT_HEX_2
6B52 6721
6B52 6722 2022'C3                       JMP     PARSER
6B55 6723
6B55 6724
6B55 6725               ;*****************************************************************************
6B55 6726               ;
6B55 6727               ;  POWER_REC
6B55 6728               ;  THIS ROUTINE IS BRANCHED TO ON A POWER RECOVERY
6B55 6729               ;  IT WILL CALL THE ROM CODE THAT DOES POWER UP SEQUENCING AND WILL
6B55 6730               ;  DO AN INITIALIZE IF THERE IS WCS CODE LOADED
6B55 6731               ;
6B55 6732               ;*****************************************************************************
6B55 6733
6B55 6734 001B CD       POWER_REC:      CALL    POWER_SEQ               ;CALL ROM POWER SEQ ROUTINE
6B58 6735
6B58 6736                                                               ;THE FOLLOWING DELAY LOOP
6B58 6737                                                               ; IS TO ALLOW AC LOW TO
6B58 6738                                                               ; SETTLE BEFORE RE-ENABLING
6B58 6739                                                               ; INTERRUPTS
6B58 6740 00 00 21                      LXI     H,0                     ;INIT COUNTER IN H&L TO 0
6B5B 6741 2B            1$:             DCX     H                       ;DECREMENT COUNTER
6B5C 6742 7C                            MOV     A,H                     ;GET HIGH BYTE RESULT
6B5D 6743 B5                            ORA     L                       ;'OR' WITH LOW BYTE
6B5E 6744 1F5B'C2                       JNZ     1$                      ;REPEAT IF NOT 0
6B61 6745
6B61 6746 10 3E                         MVI     A,HEX_10                ;A GETS A 10(H)
6B63 6747 30                            SIM                             ;RESET 7.5
6B64 6748
6B64 6749 0401'32 3C AF                 SET     WARM_RESTART            ;INDICATE POWER FAIL RESTART
6B69 6750 1F6D'C3                       JMP     START_UP
6B6C 6751
6B6C 6752               ;*****************************************************************************
6B6C 6753               ;
```

```
6B6C 6754
6B6C 6755                ;  TIMER_VEC
6B6C 6756                ;
6B6C 6757                ;       THIS ROUTINE WILL HANDLE A INTERVAL TIMER INTERRUPT IF THE INTERRUPT
6B6C 6758                ;       IS NOT MASKED OUT.  AT PRESENT, THIS ROUTINE WILL NOT BE ENTERED AND
6B6C 6759                ;       DOES NOTHING.  THIS ROUTINE WILL BE WRITTEN AT A LATER TIME IF NECES-
6B6C 6760                ;       SARY.
6B6C 6761                ;
6B6C 6762                ;*******************************************************************************
6B6C 6763 C9             TIMER_VEC:      RET                             ;DO A RETURN
6B6D 6765                ;*******************************************************************************
6B6D 6766                ;
6B6D 6767                ;  MAIN LINE ROUTINES
6B6D 6768                ;
6B6D 6769                ;*******************************************************************************
6B6D 6770
6B6D 6771                ;*******************************************************************************
6B6D 6772                ;
6B6D 6773                ; START_UP
6B6D 6774                ; THINGS THAT ARE ONLY DONE AT THE BEGINING
6B6D 6775                ;
6B6D 6776                ;*******************************************************************************
6B6D 6777
6B6D 6778
6B6D 6779 40 80 31       START_UP:       LXI     $SP,PRI_STACK
6B70 6780
6B70 6781 F3                             DI                              ;DISABLE INTERUPTS
6B71 6782
6B71 6783 1399'CD                        CALL    STOP_CPU                ;STOP CPU IN CASE IT IS RUNNING
6B74 6784 AF                             XRA     A
6B75 6785 02C8'32                        STA     MICRO_STEP              ;CLEAR MICRO STEP MODE
6B78 6786 3C                             INR     A
6B79 6787 029F'32                        STA     MEM_REQ                 ;ENABLE MEM REQ FOR LATER
6B7C 6788
6B7C 6789 1ED7'CD                        CALL    CHECKSUM
6B7F 6790 02F0'3A                        LDA     NEWSUM                  ;SAVE CHECKSUM VALUE
6B82 6791 0306'32                        STA     OLDSUM
6B85 6792
6B85 6793 03'F5'21                       LXI     H,VERSION               ;PRINT VERSION NUMBER
6B88 6794 124A'CD                        CALL    PRINT_STRING_R
6B8B 6795
6B8B 6796 AF                             XRA     A
6B8C 6797 003A'32                        STA     CONTINUE                ;INIT CONTINUE FLAG
6B8F 6798 0231'32                        STA     ERROR                   ;INIT ERROR FLAG
6B92 6799
6B92 6800 023B'32                        STA     EXECUTE                 ;CLEAR EXECUTE
6B95 6801 006E'32                        STA     WCS_BAD_PARITY          ;SET FOR GOOD PARITY
6B98 6802 3C                             INR     A
6B99 6803 0064'32                        STA     PARITY_ON               ;INIT PARITY CALC
6B9C 6804
6B9C 6805 00'7D'21                       LXI     H,X_CLR_PAGE            ;CLR WCS HIGH PAGE
6B9F 6806 15C2'CD                        CALL    PERFORM_CSR_R
6BA2 6807
6BA2 6808 04 0E 00'A9'21                 ZERO    DATA0,4                 ;RESET OUT OF COMPAT. MODE
          0D 23 77 AF
          1FA8'C2
6BAE 6809 FF 3E                          MVI     A,HEX_FF
6BB0 6810 1415'CD                        CALL    WRITE_LS_R
6BB3 6811
6BB3 6812 030C'32 3C AF                  SET     PAR_ON                  ;ENABLE WCS PARITY FOR LATER
```

```
6BB8 6813
6BB8 6814 20                                        RIM                         ;SET INTERRUPT MASK
6BB9 6815 08 F6                                     ORI     HEX_8
6BBB 6816 01 F6                                     ORI     HEX_1               ;DISABLE 5.5 INTERRUPT
6BBD 6817 0D E6                                     ANI     HEX_0D
6BBF 6818 30                                        SIM
6BC0 6819
6BC0 6820 FB                                        EI                          ;TURN ON INTERRUPTS
6BC1 6821
6BC1 6822 0F 0401'3A            ******************  IF      WARM_RESTART        ;IF POWER FAIL RESTART
          1FD5'D2               *
6BC8 6823 15AF'CD               *                   CALL    WRITE_LS_STAT       ;WRITE CURRENT FLAGS IN LS
6BCB 6824                       *                                               ; ERROR CONTROL WORD
6BCB 6825 0408'3A               *                   LDA     WCS_LOADED          ;GET WCS LOADED FLAG
6BCE 6826 B7                    *                   ORA     A                   ;SET CONDITION CODES
6BCF 6827 1E8A'C4               *                   CNZ     EXEC_TEST0          ;DO WCS INIT IF WCS IS LOADED
6BD2 6828                       *
6BD2 6829 1FD8'C3               ******************  ELSE
6BD5 6830                       *
6BD5 6831 0A9A'CD               *                   CALL    CLEAR_DEFAULT       ;SET FLAGS TO DEFAULT
6BD8 6832                       ******************  ENDIF
6BD8 6833
6BD8 6834 2022'C3                                   JMP     PARSER              ;GO
6BDB 6835
6BDB 6836
6BDB 6837
6BDB 6838
6BDB 6839
6BDB 6840
6BDB 6841
6BDB 6842
6BDB 6843
6BDB 6844
6BDB 6845
6BDB 6846
6BDB 6847
6BDB 6848
6BDB 6849
6BDB 6850
6BDB 6851 03'BB'11                                  PAR_TEMP_BUF:   LXI     D,TEMP_BUF          ;POINTER TO CHAR BUFFER
6BDE 6852
6BDE 6853 0312'22                                   PAR:            SHLD    PARSE_TABLE
6BE1 6854 EB                                                        XCHG
6BE2 6855 0310'22                                                   SHLD    PARSE_PNT           ;SAVE BUFFER TO COMPARE WITH
6BE5 6856 02F6'32 AF                                                CLEAR   NOT_FOUND           ;RESET NOT_FOUND FLAG
6BE9 6857
6BE9 6858 0310'2A                                   2$:             LHLD    PARSE_PNT
6BEC 6859 EB                                                        XCHG                        ;ONE POINTER IN D+E
6BED 6860 0312'2A                                                   LHLD    PARSE_TABLE         ;OTHER IN H+L
6BF0 6861 4E                                                        MOV     C,M                 ;GET COUNTER
6BF1 6862 79                                                        MOV     A,C
6BF2 6863 FF FE                                                     CPI     HEX_FF              ;END OF TABLE??
6BF4 6864 201C'CA                                                   JZ      1$
6BF7 6865 23                                                        INX     H
6BF8 6866 1184'CD                                                   CALL    COMPARE_R           ;COMPARE TABLE AND BUFFER
6BFB 6867
6BFB 6868 200F'C2               ******************  IFZ
6BFE 6869                       *
6BFE 6870 030D'22               *                   SHLD    PAR_VECT
6C01 6871 1063'CD               *                   CALL    SKIP_TO_SPEC        ;SKIP ALL NONPARSED CHARS
```

```
;*****************************************************************************
;
;  PAR
;  THIS ROUTINE PARSES COMMANDS AND DISPATCHES TO THE ADDRESS GIVEN
;
;  INPUT CONDITONS
;       H+L = PARSE NAME TABLE
;             EACH ENTRY = LEN,ASCII,JMP ADDRESS
;             LAST TABLE ENTRY HAS LENTH FF
;       D+E = BUFFER TO COMPARE ASCII WITH
;
;*****************************************************************************
```

```
6C04 6872 030D'2A                *                  LHLD    PAR_VECT                ;PAR_VECT CONTAINS ADDRESS
6C07 6873 5E                     *                  MOV     E,M                     ; THAT POINTS TO APPROPRIATE
6C08 6874 23                     *                  INX     H                       ; ROUTINE
6C09 6875 56                     *                  MOV     D,M
6C0A 6876 EB                     *                  XCHG
6C0B 6877                        *
6C0B 6878 E9                     *                  PCHL                            ;BRANCH TO ROUTINES
6C0C 6879                        *
6C0C 6880 2019'C3                ******************ELSE
6C0F 6881                        *
6C0F 6882 79                     *                  MOV     A,C                     ;USE LEFTOVER COUNT FROM
6C10 6883 02 C6                  *                  ADI     2                       ;COMPARE TO SKIP OVER DATA
6C12 6884 4F                     *                  MOV     C,A
6C13 6885 00 06                  *                  MVI     B,0
6C15 6886 09                     *                  DAD     B                       ;SPIP TO NEXT COMMAND
6C16 6887 0312'22                *                  SHLD    PARSE_TABLE
6C19 6888                        *
6C19 6889                        ******************ENDIF
6C19 6890
6C19 6891 1FE9'C3                                   JMP     2$
6C1C 6892
6C1C 6893 02F6'32 3C AF                     1$:     SET     NOT_FOUND               ;NOT FOUND
6C21 6894 C9                                        RET
6C22 6895
6C22 6896
6C22 6897
6C22 6898
6C22 6899
6C22 6900
6C22 6901
6C22 6902
6C22 6903
6C22 6904 AF                                PARSER: XRA     A
6C23 6905 0345'32                                   STA     SHORTEN                 ;CLEAR SHORTEN
6C26 6906 02C8'32                                   STA     MICRO_STEP              ;CLEAR MICRO_STEP
6C29 6907
6C29 6908 023B'32 AF             ******************WHILENR EXECUTE
          0F 023B'3A             *
          206F'DA                *
6C34 6909                        *
6C34 6910 0F 0334'3A             * ****************IFN     RESTART
          2044'DA                * *
6C3B 6911                        * *
6C3B 6912 1208'CD                * *                CALL    PRINT_PROMPT
6C3E 6913 126B'CD                * *                CALL    INPUT_LINE
6C41 6914                        * *
6C41 6915 204B'C3                * ****************ELSE
6C44 6916                        * *
6C44 6917 40 80 31               * *                LXI     $SP,PRI_STACK           ;SET UP STACK POINTER
6C47 6918 0334'32 AF             * *                CLEAR   RESTART
6C4B 6919                        * *
6C4B 6920                        * ****************ENDIF
6C4B 6921                        *
6C4B 6922 04'F5'21               *                  LXI     H,MAIN_TAB              ;POINTER TO PARSE TABLE
6C4E 6923 01'64'11               *                  LXI     D,TTBUF                 ;POINTER TO CHAR BUFFER
6C51 6924 1FDE'CD                *                  CALL    PAR
6C54 6925                        *
6C54 6926 0F 020A'3A             * ****************IF      CONT_DONE               ;CONTINUE PARSED.. BACKUP
          2065'D2                * *
6C5B 6927                        * *                                             ; INTO STACK ONE LEVEL
```

```
;*****************************************************************************
;
;  PARSER
;  THIS ROUTINE PARSES THE COMMAND LINE GIVEN AFTER THE PROMPT AND DISPATCHES
;  TO THE PROPER SUBROUTINE TO HANDLE EACH KEYWORD
;
;*****************************************************************************
```

```
6C5B 6928 020A'32 AF        * *                 CLEAR   CONT_DONE
6C5F 6929 01 3E             * *                 MVI     A,1
6C61 6930 0011'32           * *                 STA     APT_MESSAGE_CODE+1      ;SET APT START FLAG
6C64 6931 C9                * *                 RET
6C65 6932                   * ****************  ENDIF
6C65 6933                   *
6C65 6934 02F6'3A           *                   LDA     NOT_FOUND               ;GET NOT FOUND FLAG
6C68 6935 B7                *                   ORA     A                       ;SET CONDITION CODES
6C69 6936 1EC9'C4           *                   CNZ     SYNTAX_ERROR            ;REPORT SYNTAX ERROR IF SET
6C6C 6937                   *
6C6C 6938 202D'C3           ******************  ENDWHILE
6C6F 6939
6C6F 6940
6C6F 6941
6C6F 6942
6C6F 6943
6C6F 6944
6C6F 6945
6C6F 6946
6C6F 6947
6C6F 6948
6C6F 6949
6C6F 6950
6C6F 6951
6C6F 6952
6C6F 6953
6C6F 6954
6C6F 6955
6C6F 6956
6C6F 6957
6C6F 6958
6C6F 6959
6C6F 6960
6C6F 6961
6C6F 6962 0F 023B'3A        ******************  WHILE   EXECUTE
          2131'D2           *
6C76 6963                   *
6C76 6964 00 FE 03B3'3A     * ****************  IFNEQI  TABLE_MODE,0
          20DA'CA           * *
6C7E 6965                   * *
6C7E 6966 2134'CD           * *                 CALL    SET_ALL_TESTS           ;SET UP TEST NUM FOR ALL
6C81 6967                   * *                                                 ; TESTS AND CLEAR EOS FLG
6C81 6968 0241'32 AF        * * **************  UNTILR  FILE_LOADED
          0F 0241'3A        * * *
          20D7'DA           * * *
6C8C 6969                   * * *
6C8C 6970 03B4'2A           * * *               LHLD    TABLE_PNT               ;POINTER TO SECTION TABLE
6C8F 6971 03'CC'11          * * *               LXI     D,TEMP_FLAG             ;POINTER TO TEMP DATA BUFFER
6C92 6972 07 0E             * * *               MVI     C,SEC_LEN+1             ;COUNT
6C94 6973 11A4'CD           * * *               CALL    MOVER_R                 ;MOVE ENTRIES TO LOCAL BUFFER
6C97 6974 03B4'22           * * *               SHLD    TABLE_PNT               ;UPDATE SECTION TABLE POINTER
6C9A 6975                   * * *
6C9A 6976 03CC'3A           * * *               LDA     TEMP_FLAG
6C9D 6977 3F E6             * * *               ANI     HEX_3F
6C9F 6978 0343'32           * * *               STA     SECTION_B_NUM           ;SAVE BOARD THAT OWNS THIS SEC
6CA2 6979                   * * *
6CA2 6980 FF FE 03CC'3A     * * * ************  IFEQI   TEMP_FLAG,HEX_FF        ;END OF TABLE??
          20B0'C2           * * * *
6CAA 6981                   * * * *
6CAA 6982 1AB2'CD           * * * *             CALL    EOP
```

```
;*****************************************************************************
;
;  THIS ROUTINE PARSES THE SECTION TABLE
;
;  THERE IS A FLAG WITH EACH SECTION NAME (SECTION_FLAG)
;
;  SECTION_FLAG:          FF = END OF TABLE
;                BIT 7     0 = NAME IS SECTION NAME FOR WCS
;                            1 = NAME IS SECTION NAME FOR CONSOLE RAM
;                BIT 6     1 = OPTIONAL TEST
;             BITS 0 - 5   XX = CPU CLUSTER BOARD NUMBER
;
;  THERE IS A FLAG STATING HOW THE TABLE IS TO BE PARSED (TABLE_MODE)
;
;  TABLE_MODE    0 = TABLE NOT IN USE
;                1 = BOARD MODE
;                2 = ALL SECTION MODE
;
;*****************************************************************************
```

```
6CAD 6983                    * * * *
6CAD 6984 20D4'C3            * * * ************
6CB0 6985                    * * * *
6CB0 6986                    * * * *
6CB0 6987 40 E6 03CC'3A      * * * * **********
          20D4'C2            * * * * *
6CB8 6988                    * * * * *
6CB8 6989 01 FE 03B3'3A      * * * * * ********
          20D1'C2            * * * * * *
6CC0 6990                    * * * * * *
6CC0 6991 47 0343'3A         * * * * * * ******
          A0 01DA'3A         * * * * * * *
          20CE'CA            * * * * * * *
6CCB 6992 1D61'CD            * * * * * * *
6CCE 6993                    * * * * * * *
6CCE 6994                    * * * * * * ******
6CCE 6995                    * * * * * *
6CCE 6996 20D4'C3            * * * * * ********
6CD1 6997 1D61'CD            * * * * * *
6CD4 6998                    * * * * * *
6CD4 6999                    * * * * * ********
6CD4 7000                    * * * * *
6CD4 7001                    * * * * **********
6CD4 7002                    * * * *
6CD4 7003                    * * * ************
6CD4 7004                    * * *
6CD4 7005 2085'C3            * * **************
6CD7 7006                    * *
6CD7 7007 20E1'C3            * ****************
6CDA 7008                    * *
6CDA 7009 0056'3A            * *
6CDD 7010 B7                 * *
6CDE 7011 1AB2'C4            * *
6CE1 7012                    * *
6CE1 7013                    * ****************
6CE1 7014                    *
6CE1 7015 0F 023B'3A         * ****************
          212E'D2            * *
6CE8 7016                    * *
6CE8 7017 0019'3A            * *
6CEB 7018 B7                 * *
6CEC 7019 20F6'C2            * *
6CEF 7020 0018'3A            * *
6CF2 7021 B7                 * *
6CF3 7022 2143'CC            * *
6CF6 7023                    * *
6CF6 7024 0F 0209'3A         * * ************** 1$:
          2118'D2            * * *
6CFD 7025 0240'32 AF         * * *
6D01 7026 04 0E 00'A9'21     * * *
          0D 23 77 AF        * * *
          2107'C2            * * *
6D0D 7027 FF 3E              * * *
6D0F 7028 1415'CD            * * *
6D12 7029 7000 CD            * * *
6D15 7030                    * * *
6D15 7031 212E'C3            * * ***************
6D18 7032                    * * *
6D18 7033 03DA'3A            * * *
6D1B 7034 67                 * * *

ELSE

IFNMBI  TEMP_FLAG,OPTIONAL_TST  ;OPTIONAL TEST??

IFEQI   TABLE_MODE,1            ;BOARD MODE??

IFMB    BOARD_NUM,SECTION_B_NUM ;SEC BELONG TO THIS BOARD?

CALL    MOV_LOAD_SECTION        ;MOVE NAME TO SECTION_NAME AND
                                ; LOAD SECTION
ENDIF

ELSE
CALL    MOV_LOAD_SECTION        ;IF DI MODE, MOVE NAME TO
                                ; SECTION_NAME AND LOAD SECTION
ENDIF

ENDIF

ENDIF

ENDUNTIL

ELSE

LDA     EOS_FLG                 ;GET EOS FLAG
ORA     A                       ;SET CONDITION CODES
CNZ     EOP                     ;CALL EOP IF SET

ENDIF

IF      EXECUTE

LDA     APT_PASS_CNT+1          ;GET PASS CNT
ORA     A                       ;SET CONDITION CODES
JNZ     1$                      ;JUMP OUT IF LSB NOT 0
LDA     APT_PASS_CNT            ;GET MSB OF PASS CNT
ORA     A                       ;SET CONDITION CODES
CZ      PRINT_VERSION           ;PRINT THE VERSION NUMBER IF
                                ; FIRST PASS
IF      CONSOLE_TEST

CLEAR   FAST_WRITE
ZERO    DATA0,4                 ;RESET OUT OF COMPAT. MODE

MVI     A,HEX_FF
CALL    WRITE_LS_R
CALL    EXEC_CONTESTS

ELSE

LDA     TEST_NUM1               ;USE TEST NUMBER AS STARTING
MOV     H,A                     ;ADDRESS IN WCS
```

```
6D1C 7035 AF                    * * *                   XRA     A
6D1D 7036 6F                    * * *                   MOV     L,A
6D1E 7037 03A1'22               * * *                   SHLD    STARTING_UPC
6D21 7038                       * * *
6D21 7039 03DA'3A               * * *                   LDA     TEST_NUM1               ;SET UP FOR INC ON BEG OF TEST
6D24 7040 3D                    * * *                   DCR     A
6D25 7041 03D9'32               * * *                   STA     TEST_NUM
6D28 7042                       * * *
6D28 7043 13CE'CD               * * *                   CALL    START_CPU
6D2B 7044 15F0'CD               * * *                   CALL    ATTENTION
6D2E 7045                       * * *
6D2E 7046                       * * **************      ENDIF
6D2E 7047                       * *
6D2E 7048                       * ****************      ENDIF
6D2E 7049                       *
6D2E 7050 206F'C3               ******************      ENDWHILE
6D31 7051
6D31 7052 2022'C3                                       JMP     PARSER
6D34 7053
6D34 7054
6D34 7055                       ;*************************************************************************
6D34 7056                       ;
6D34 7057                       ;  SET_ALL_TESTS:
6D34 7058                       ;
6D34 7059                       ;  THIS ROUTINE SETS STARTING TEST NUM TO 1, ENDING TEST NUM TO 50(H) AND
6D34 7060                       ;  CLEARS THE EOS FLAG.
6D34 7061                       ;
6D34 7062                       ;*************************************************************************
6D34 7063                       SET_ALL_TESTS:
6D34 7064 01 3E                                         MVI     A,1
6D36 7065 03DA'32                                       STA     TEST_NUM1               ;SET TO START AT TEST 1(D)
6D39 7066
6D39 7067 50 3E                                         MVI     A,HEX_50
6D3B 7068 03DB'32                                       STA     TEST_NUM2               ;SET UP ENDING TEST
6D3E 7069
6D3E 7070 0056'32 AF                                    CLEAR   EOS_FLG
6D42 7071
6D42 7072 C9                                            RET                             ;DONE
6D43 7073
6D43 7074                       ;*************************************************************************
6D43 7075                       ;
6D43 7076                       ;  PRINT_VERSION:
6D43 7077                       ;
6D43 7078                       ;       THIS ROUTINE PRINTS THE VERSION NUMBER OF THE MICRO-DIAGNOSTIC BEING
6D43 7079                       ;       RUN AS WELL AS THE FILE NAME.  THE VERSION NUMBER IS FOUND IN THE
6D43 7080                       ;       FILE LOADED.  FOR CONSOLE FILES, THE VERSION IS AT ADDRESS DEFINED
6D43 7081                       ;       AS CON_VER_ADD; FOR WCS BASED FILES, THE VERSION IS AT THE ADDRESS
6D43 7082                       ;       DEFINED AS WCS_VER_ADD.  IN BOTH CASES THE VERSION IS 2 BYTES LONG.
6D43 7083                       ;       THE FIRST LOCATION IS THE MAJOR REV NUMBER, THE FOLLOWING LOCATION
6D43 7084                       ;       IS THE MINOR REV NUMBER.
6D43 7085                       ;
6D43 7086                       ;*************************************************************************
6D43 7087
6D43 7088                       PRINT_VERSION:
6D43 7089 0038'3A                                       LDA     APT                     ;SEE IF APT PRESENT
6D46 7090 B7                                            ORA     A                       ;SET CONDITION CODES
6D47 7091 C0                                            RNZ                             ;SKIP PRINTING IF APT
6D48 7092
6D48 7093 125A'CD                                       CALL    CRLF_R                  ;ELSE PRINT <CR><LF>
6D4B 7094 125A'CD                                       CALL    CRLF_R                  ;PRINT <CR><LF>
```

```
6D4E 7095
6D4E 7096 7005 3A                                   LDA     CON_NAME_ADD            ;GET NEXT TO LAST LETTER OF
6D51 7097                                                                           ; FILE NAME FROM FILE
6D51 7098 0224'32                                   STA     ENK_END                 ;STORE IN MEMORY
6D54 7099 7006 3A                                   LDA     CON_NAME_ADD+1          ;GET LAST LETTER OF
6D57 7100                                                                           ; FILE NAME FROM FILE
6D57 7101 0225'32                                   STA     ENK_END+1               ;STORE IN MEMORY
6D5A 7102 7004 3A                                   LDA     CON_VER_ADD+1           ;GET CONSOLE MAJOR VERSION #
6D5D 7103 005B'32                                   STA     HEX_BUF                 ;STORE VALUE IN HEX BUF
6D60 7104
6D60 7105 0F 0209'3A                ******************  IFN     CONSOLE_TEST            ;FOR WCS TESTS, DO FOLLOWING
          217F'DA                   *
6D67 7106 06 07 21                  *                   LXI     H,WCS_FILE_NAME         ;ADDRESS OF LAST TWO LETTERS
6D6A 7107                           *                                                   ; OF FILE NAME
6D6A 7108 21C0'CD                   *                   CALL    READ_WCS_SUB            ;SWITCH H&L AND READ WCS
6D6D 7109 0224'32                   *                   STA     ENK_END                 ;STORE VALUE IN MEMORY
6D70 7110 00B6'3A                   *                   LDA     WCS_VALUE+2             ;GET FIRST BYTE
6D73 7111 0225'32                   *                   STA     ENK_END+1               ;STORE VALUE IN MEMORY
6D76 7112                           *
6D76 7113 06 06 21                  *                   LXI     H,WCS_VER_ADD           ;ADDRESS OF VERSION NUMBER
6D79 7114 21C0'CD                   *                   CALL    READ_WCS_SUB            ;SWITCH H&L AND READ WCS
6D7C 7115 005B'32                   *                   STA     HEX_BUF                 ;STORE VALUE IN HEX BUF
6D7F 7116                           ******************  ENDIF
6D7F 7117
6D7F 7118 02'21'21                                  LXI     H,ENK_A                 ;POINT TO ASCII FILE NAME
6D82 7119 04'DF'11                                  LXI     D,SECTION_NAME          ;POINT TO DESTINATION
6D85 7120 06 0E                                     MVI     C,6                     ;6 CHARS TO TRANSFER
6D87 7121 11A4'CD                                   CALL    MOVER_R                 ;MOVE THE BLOCK
6D8A 7122
6D8A 7123 04'DF'21                                  LXI     H,SECTION_NAME          ;GET NAME OF SECTION
6D8D 7124 06 0E                                     MVI     C,SEC_LEN               ;GET LENGTH OF NAME
6D8F 7125 1243'CD                                   CALL    PRINTER                 ;PRINT IT
6D92 7126 03'F2'21                                  LXI     H,VER_A                 ;POINT TO "-V"
6D95 7127 124A'CD                                   CALL    PRINT_STRING_R          ;PRINT IT
6D98 7128
6D98 7129 02F5'32 3C AF                             SET     NO_SPACE                ;NO SPACE AFTER HEX DIGITS
6D9D 7130 11C2'CD                                   CALL    PRINT_HEX_1_R           ;PRINT MAJOR VERSION #
6DA0 7131 02F5'32 AF                                CLEAR   NO_SPACE                ;CLEAR FLAG
6DA4 7132 2E 3E                                     MVI     A,DOT_A                 ;GET ASCII "."
6DA6 7133 1265'CD                                   CALL    TYPE_CHAR               ;PRINT IT
6DA9 7134
6DA9 7135 7003 3A                                   LDA     CON_VER_ADD             ;GET CONSOLE MINOR VERSION #
6DAC 7136 005B'32                                   STA     HEX_BUF                 ;STORE VALUE IN HEX BUF
6DAF 7137
6DAF 7138 0F 0209'3A                ******************  IFN     CONSOLE_TEST            ;IF WCS TEST
          21BC'DA                   *
6DB6 7139 00B6'3A                   *                   LDA     WCS_VALUE+2             ;GET LOW BYTE
6DB9 7140 005B'32                   *                   STA     HEX_BUF                 ;STORE VALUE IN HEX BUF
6DBC 7141                           ******************  ENDIF
6DBC 7142
6DBC 7143 11C2'CD                                   CALL    PRINT_HEX_1_R           ;PRINT MINOR VERSION #
6DBF 7144 C9                                        RET                             ;ALL DONE
6DC0 7145
6DC0 7146 7C                                READ_WCS_SUB:   MOV     A,H             ;SWITCH H&L
6DC1 7147 65                                                MOV     H,L
6DC2 7148 6F                                                MOV     L,A
6DC3 7149 006C'22                                   SHLD    WCS_ADDRESS             ;STORE ADDRESS TO READ
6DC6 7150 109D'CD                                   CALL    READ_WCS_R              ;READ FILE NAME
6DC9 7151 00B5'3A                                   LDA     WCS_VALUE+1             ;GET SECOND BYTE
6DCC 7152 C9                                        RET                             ;DONE
```

```
6DCD 7153
6DCD 7154                                   ;
6DCD 7155                                   ;CONSOLE TEST SUBROUTINES
6DCD 7156                                   ;
6DCD 7157
6DCD 7158                                   ;*****************************************************************************
6DCD 7159                                   ;
6DCD 7160                                   ;  CON_BEGIN_TEST_R  THIS ROUTINE IS USED FOR BEGINNING OF TEST FOR CONSOLE
6DCD 7161                                   ;                    BASED CPU TESTS ONLY.
6DCD 7162                                   ;
6DCD 7163                                   ;                      INPUT CONDITIONS:
6DCD 7164                                   ;                          A=TEST NUMBER
6DCD 7165                                   ;
6DCD 7166                                   ;*****************************************************************************
6DCD 7167
6DCD 7168                                   CON_BEGIN_TEST_R:
6DCD 7169 03D9'32                                       STA     TEST_NUM                ;SAVE TEST NUMBER
6DD0 7170 003A'32 3C AF                                 SET     CONTINUE                ;SET UP TO CONTINUE IF TEST
6DD5 7171                                                                               ; STOPPED BY OTHER THAN ^P
6DD5 7172 4129 CD                                       CALL    GCVECT                  ;CHECK FOR OPERATOR REQUEST
6DD8 7173
6DD8 7174 1341'CD                                       CALL    CHECK_CNTLC             ;IF ^C, GO BACK TO PARSER AND
6DDB 7175                                                                               ; ALLOW CONTINUE
6DDB 7176
6DDB 7177 01 3E                                         MVI     A,1
6DDD 7178 0011'32                                       STA     APT_MESSAGE_CODE+1      ;SET APT START FLAG
6DE0 7179
6DE0 7180 03'DA'21              ******************      IFGT    TEST_NUM1,TEST_NUM,1
          01 0E 03'D9'11        *
          21F6'F2 00CF'CD       *
6DEE 7181 006A'32 3C AF         *                       SET     SKIP_TEST
6DF3 7182                       *
6DF3 7183 221E'C3               ******************      ELSE
6DF6 7184                       *
6DF6 7185 006A'32 AF            *                       CLEAR   SKIP_TEST
6DFA 7186                       *
6DFA 7187 03'D9'21              * ****************      IFGT    TEST_NUM,TEST_NUM2,1
          01 0E 03'DB'11        * *
          2211'F2 00CF'CD       * *
6E08 7188 0056'32 3C AF         * *                     SET     EOS_FLG                 ;FORCE END OF SECTION
6E0D 7189 C9                    * *                     RET
6E0E 7190                       * *
6E0E 7191 221E'C3               * ****************      ELSE
6E11 7192 03D9'3A               * *                     LDA     TEST_NUM                ;GET TEST NUMBER
6E14 7193 0017'32               * *                     STA     APT_TEST_NUMBER+1       ;SAVE IN APT MAILBOX
6E17 7194 03DF'3A               * *                     LDA     TRACE                   ;GET TRACE FLAG
6E1A 7195 B7                    * *                     ORA     A                       ;SET CONDITION CODES
6E1B 7196 1B35'C4               * *                     CNZ     PRINT_TRACE             ;PRINT SECTION AND TEST #
6E1E 7197                       * *                                                     ; IF SET
6E1E 7198                       * ****************      ENDIF
6E1E 7199                       *
6E1E 7200                       ******************      ENDIF
6E1E 7201
6E1E 7202 C9                                            RET
6E1F 7203
6E1F 7204
6E1F 7205
6E1F 7206                                   ;*****************************************************************************
6E1F 7207                                   ;
6E1F 7208                                   ;  CON_ERROR_R  THIS ROUTINE IS EXECUTED TO PRINT AN ERROR FOUND BY A
```

```
6E1F 7209
6E1F 7210
6E1F 7211
6E1F 7212
6E1F 7213
6E1F 7214 3C D3                                 CON_ERROR_R:    OUT     CLRLIT          ;TURN RUN LITE OFF
6E21 7215 223C'CD                                               CALL    MSK_ERROR_R     ;MASK THE EXPECTED AND RECEIVED
6E24 7216                                                                               ; DATA TO ELIMINATE ANY BITS
6E24 7217                                                                               ; WHICH ARE NOT OF INTEREST.
6E24 7218 171D'CD                                               CALL    WCS_ERROR       ;CALL MICMON ROUTINE TO PRINT ERROR
6E27 7219                                                                               ; REPORT
6E27 7220 005A'3A                                               LDA     FLAGS           ;DO NOT SET ERROR CON FLAG IF LOOP
6E2A 7221 1F                                                    RAR                     ; ON ERROR IS NOT SET
6E2B 7222 2239'D2                                               JNC     1$
6E2E 7223
6E2E 7224 0057'32 3C AF                                         SET     ERROR_CON       ;SET FLAG SAYING ERROR HAPPENED AND
6E33 7225                                                                               ; LOOP ON ERROR WAS SET
6E33 7226 4129 CD                                               CALL    GCVECT          ;CHECK FOR OPERATOR INPUT
6E36 7227 1341'CD                                               CALL    CHECK_CNTLC     ;CHECK FOR CONTROL C INPUT
6E39 7228 3D D3                                 1$:             OUT     SETLIT          ;TURN RUN LITE BACK ON
6E3B 7229 C9                                                    RET
6E3C 7230
6E3C 7231
6E3C 7232
6E3C 7233
6E3C 7234
6E3C 7235
6E3C 7236
6E3C 7237
6E3C 7238
6E3C 7239
6E3C 7240
6E3C 7241 00'40'21                              MSK_ERROR_R:    LXI     H,CON_EXP_DAT   ;MOVE MEMORY ADDRESS CON_EXP_DAT
6E3F 7242                                                                               ; INTO THE H,L PAIR.
6E3F 7243 0044'22                                               SHLD    CON_EXP_ADD     ;STORE THE ADDRESS OF THE EXPECTED
6E42 7244                                                                               ; DATA IN CON_EXP_DAT.
6E42 7245 00'4B'11                                              LXI     D,CON_MSK_DAT   ;MOVE MEMORY ADDRESS CON_MSK_DAT
6E45 7246                                                                               ; INTO THE D,E PAIR.
6E45 7247
6E45 7248 C5 46 00'55'21        ****************                DO      4               ;LOOP 4 TIMES TO MASK 4 BYTES OF
          77 04 3E              *
6E4D 7249                       *                                                       ; EXPECTED DATA.
6E4D 7250 0044'2A               *                               LHLD    CON_EXP_ADD     ;MOVE THE MEMORY ADDRESS OF THE BYTE
6E50 7251                       *                                                       ; OF EXPECTED DATA INTO THE H,L PAIR.
6E50 7252 2283'CD               *                               CALL    MSK_BYTE        ;CALL ROUTINE TO MASK 1 BYTE
6E53 7253 0044'22               *                               SHLD    CON_EXP_ADD     ;STORE THE MEMORY ADDRESS OF THE
6E56 7254                       *                                                       ; EXPECTED DATA INTO CON_EXP_ADD.
6E56 7255 35 00'55'21           ****************                ENDDO
          70 C1 224D'C2
6E5F 7256
6E5F 7257 00'4F'21                                              LXI     H,CON_REC_DAT   ;MOVE MEMORY ADDRESS CON_REC_DAT
6E62 7258                                                                               ; INTO THE H,L PAIR.
6E62 7259 0053'22                                               SHLD    CON_REC_ADD     ;STORE THE ADDRESS OF THE RECEIVED
6E65 7260                                                                               ; DATA IN CON_REC_ADD.
6E65 7261 00'4B'11                                              LXI     D,CON_MSK_DAT   ;MOV MEMORY ADDRESS CON_MSK_DAT
6E68 7262                                                                               ; INTO THE D,E PAIR.
6E68 7263
6E68 7264 C5 46 00'55'21        ****************                DO      4               ;LOOP 4 TIMES TO MASK 4 BYTES OF
          77 04 3E              *
6E70 7265                       *                                                       ; RECEIVED DATA.
```

```
6E70 7266 0053'2A                *                       LHLD    CON_REC_ADD     ;LOAD THE ADDRESS OF THE RECEIVED
6E73 7267                        *                                               ; DATA INTO THE H,L PAIR.
6E73 7268 2283'CD                *                       CALL    MSK_BYTE        ;CALL ROUTINE TO MASK ONE BYTE
6E76 7269 0053'22                *                       SHLD    CON_REC_ADD     ;STORE THE ADDRESS OF THE RECEIVED.
6E79 7270                        *                                               ; DATA IN CON_REC_ADD.
6E79 7271 35 00'55'21            ******************      ENDDO
          70 C1 2270'C2
6E82 7272
6E82 7273 C9                                             RET
6E83 7274
6E83 7275 46                                 MSK_BYTE:   MOV     B,M             ;MOVE A BYTE OF EXPECTED DATA INTO
6E84 7276                                                                        ; REGISTER B.
6E84 7277 1A                                             LDAX    D               ;MOVE A BYTE OF MASK DATA INTO
6E85 7278                                                                        ; THE ACCUMULATOR.
6E85 7279 2F                                             CMA                     ;CHANGE THE MASK POLARITY BY
6E86 7280                                                                        ;COMPLEMENTING THE ACCUMULATOR.
6E86 7281 A0                                             ANA     B               ;"AND" THE COMPLEMENTED MASK WITH
6E87 7282                                                                        ; THE REGISTER B CONTENTS.
6E87 7283 77                                             MOV     M,A             ;MOVE THE MASKED EXPECTED DATA BACK
6E88 7284                                                                        ; INTO THE EXPECTED DATA LOCATION.
6E88 7285 23                                             INX     H               ;INCREMENT THE D,E AND H,L REGISTER
6E89 7286 13                                             INX     D               ; PAIRS TO POINT AT THE NEXT BYTES
6E8A 7287                                                                        ; OF EXPECTED AND RECEIVED DATA.
6E8A 7288 C9                                             RET                     ;DONE
6E8B 7289
6E8B 7290
6E8B 7291                        ;*******************************************************************************
6E8B 7292                        ;
6E8B 7293                        ;  WCS_READ_R  THIS ROUTINE READS DATA FROM WCS AT THE LOCATION GIVEN BY
6E8B 7294                        ;              WCS_ADDRESS. THE DATA IS PUT INTO THE CSR AND SHIFTED OUT
6E8B 7295                        ;              INTO WCS_VALUE FOR CHECKING.  AS THE CSR IS READ, IT IS
6E8B 7296                        ;              LOADED WITH A NOP INSTRUCTION.
6E8B 7297                        ;
6E8B 7298                        ;              THIS ROUTINE IS USED BY THE CONSOLE TESTS AND IS TAYLORED
6E8B 7299                        ;              FOR FAST EXECUTION RATHER THAN VERSITILITY.
6E8B 7300                        ;
6E8B 7301                        ;*******************************************************************************
6E8B 7302
6E8B 7303
6E8B 7304 22D7'CD                            WCS_READ_R: CALL    LD_UPC_R        ;LOAD UPC WITH WCS ADDRESS
6E8E 7305 23 D3                                          OUT     SETSS           ;SINGLE STEP MACHINE TO LOAD
6E90 7306 22 D3                                          OUT     CLRSS           ; WCS CONTENTS INTO THE CSR.
6E92 7307
6E92 7308 A2 D3                                          OUT     VSHIFT
6E94 7309
6E94 7310 00'72'21                                       LXI     H,NOP_INST      ;POINT TO NOP INSTRUCTION
6E97 7311 00'B4'11                                       LXI     D,WCS_VALUE     ;POINT TO WCS VALUE AREA
6E9A 7312 118E'CD                                        CALL    MOVER_3_R       ;LOAD WCS_VALUE WITH NOP INST
6E9D 7313
6E9D 7314 18 06                                          MVI     B,24            ;LOOP COUNT IN REGISTER B
6E9F 7315 00'B6'21                           1$:         LXI     H,WCS_VALUE+2   ;POINT TO END OF WCS_VALUE AREA
6EA2 7316 03 0E                                          MVI     C,3             ;BYTE COUNT
6EA4 7317
6EA4 7318 87 DB                                          IN      CSR23           ;GET CSR BIT 23
6EA6 7319 1F                                             RAR                     ;ROTATE CSR 23 INTO CARRY
6EA7 7320 7E                                 2$:         MOV     A,M             ;MOV WCS_VALUE INTO ACCUMULATOR
6EA8 7321 17                                             RAL                     ;CARRY GOES IN BIT 0
6EA9 7322 77                                             MOV     M,A             ;MOV ACCUMULATOR BACK TO MEM
6EAA 7323 2B                                             DCX     H               ;POINT TO NEXT BYTE
6EAB 7324 0D                                             DCR     C               ;DEC BYTE COUNT
```

```
6EAC 7325 22A7'C2                       JNZ     2$              ;REPEAT UNTIL 3 BYTES DONE
6EAF 7326
6EAF 7327 38 D3                         OUT     CLRCSR          ;CLEAR CSR INPUT
6EB1 7328 22B6'D2                       JNC     6$              ;DON'T SET INPUT IF CARRY CLEAR
6EB4 7329 39 D3                         OUT     SETCSR          ;ELSE SET CSR INPUT
6EB6 7330
6EB6 7331 25 D3                 6$:     OUT     SETCSK          ;CLOCK CSR TO SHIFT IT AND ALSO LOAD
6EB8 7332                                                       ; NOP INSTRUCTION
6EB8 7333 24 D3                         OUT     CLRCSK          ;CLEAR UPC CLOCK
6EBA 7334
6EBA 7335 05                            DCR     B               ;DECREMENT SHIFT COUNT
6EBB 7336 229F'C2                       JNZ     1$              ;REPEAT UNTIL 24 BITS SHIFTED
6EBE 7337
6EBE 7338 A3 D3                         OUT     VNORML          ;RETURN TO NORMAL MODE.
6EC0 7339
6EC0 7340 C9                            RET
6EC1 7341
6EC1 7342
6EC1 7343               ;*******************************************************************************
6EC1 7344               ;
6EC1 7345               ;  WCS_WRITE_R  THIS ROUTINE WRITES DATA FROM WCS_VALUE TO WCS AT THE
6EC1 7346               ;               LOCATION GIVEN BY WCS_ADDRESS.
6EC1 7347               ;
6EC1 7348               ;*******************************************************************************
6EC1 7349
6EC1 7350 22D7'CD       WCS_WRITE_R:    CALL    LD_UPC_R        ;LOAD UPC WITH WCS ADDRESS
6EC4 7351
6EC4 7352 00'B6'21                      LXI     H,WCS_VALUE+2   ;POINT AT THE LOW BYTE.
6EC7 7353
6EC7 7354 7E                            MOV     A,M             ;MOVE THE LOW BYTE TO WCSB0.
6EC8 7355 08 D3                         OUT     WCSB0
6ECA 7356 2B                            DCX     H
6ECB 7357
6ECB 7358 7E                            MOV     A,M             ;MOVE THE NEXT BYTE TO WCSB1.
6ECC 7359 09 D3                         OUT     WCSB1
6ECE 7360 2B                            DCX     H
6ECF 7361
6ECF 7362 7E                            MOV     A,M             ;MOVE THE HIGH BYTE TO WCSB2.
6ED0 7363 0A D3                         OUT     WCSB2
6ED2 7364
6ED2 7365 37 D3                         OUT     SETWCK          ;WRITE TO WCS.
6ED4 7366 36 D3                         OUT     CLRWCK
6ED6 7367
6ED6 7368 C9                            RET
6ED7 7369
6ED7 7370               ;*******************************************************************************
6ED7 7371               ;
6ED7 7372               ;  LD_UPC_R     THIS ROUTINE LOADS THE UPC WITH WCS_ADDRESS
6ED7 7373               ;
6ED7 7374               ;*******************************************************************************
6ED7 7375
6ED7 7376 006C'2A       LD_UPC_R:       LHLD    WCS_ADDRESS     ;UPC DATA
6EDA 7377 00C2'22                       SHLD    UPC_VALUE       ;STORE DATA IN UPC_VALUE
6EDD 7378
6EDD 7379 00'C3'21                      LXI     H,UPC_VALUE+1   ;POINT TO END OF UPC_VALUE AREA
6EE0 7380 02 0E                         MVI     C,2             ;BYTE COUNT
6EE2 7381 10F0'CD                       CALL    SHIFTER1_R      ;LEFT JUST. 15 BITS WITH COUNT
6EE5 7382                                                       ; IN REGISTER C
6EE5 7383
6EE5 7384 A2 D3                         OUT     VSHIFT          ;SET V-BUS SHIFT MODE
```

```
ZZ-ENKAA-2.1    4C00    4
6EE7 7385 0F 06
6EE9 7386
6EE9 7387 00'C3'21
6EEC 7388 02 0E
6EEE 7389
6EEE 7390 7E
6EEF 7391 17
6EF0 7392 77
6EF1 7393 2B
6EF2 7394 0D
6EF3 7395 22EE'C2
6EF6 7396
6EF6 7397 A8 D3
6EF8 7398 22FD'D2
6EFB 7399 A9 D3
6EFD 7400
6EFD 7401 27 D3
6EFF 7402 26 D3
6F01 7403
6F01 7404 05
6F02 7405 22E9'C2
6F05 7406
6F05 7407 A3 D3
6F07 7408
6F07 7409 C9
6F08 7410
6F08 7411
6F08 7412
6F08 7413
6F08 7414
6F08 7415
6F08 7416
6F08 7417
6F08 7418
```

```
                MVI     B,15            ;SHIFT COUNT=SHIFT 15 BITS
3$:             LXI     H,UPC_VALUE+1   ;POINTER TO END OF UPC DATA
                MVI     C,2             ;LOAD BYTE COUNT WITH 2
1$:             MOV     A,M             ;MOVE BYTE 1 LEFT
                RAL                     ;CARRY = ROTATE IN BIT NEXT BTYE
                MOV     M,A
                DCX     H               ;MOVE POINTER TO NEXT BYTE
                DCR     C               ;DEC BYTE COUNT
                JNZ     1$              ;REPEAT UNTIL 2 BYTES SHIFTED

                OUT     CLRUPC          ;SET UPC SER IN=0 IN CASE CARRY CLR
                JNC     2$              ;SKIP NEXT INSRUCTION IF CARRY CLR
                OUT     SETUPC          ;ELSE SET UPC SER IN=1

2$:             OUT     SETUPK          ;SET UPC CLOCK
                OUT     CLRUPK          ;CLEAR UPC CLOCK

                DCR     B               ;DECREMENT SHIFT COUNT
                JNZ     3$              ;REPEAT UNTIL 15 BITS SHIFTED

                OUT     VNORML          ;SET V-BUS TO NORMAL MODE

                RET

END_CHECKSUM:

;*****************************************************************************
;
;       NOTE:  THE LAST ADDRESS OF THIS PROGRAM MUST NOT EXCEED 6FFF.  THE
;       8085 BASED MICRODIAGNOSTIC OVERLAYS LOAD STARTING AT 7000 HEX!!!!
;
;*****************************************************************************
```

Symbol table

```
$PSW                            =  00000006
$SP                             =  00000006
A                               =  00000007
ACK_SAVE                           000001B5  R      02
ADDR                            =  00000001
ALLOWP                          =  00000003
APT                                00000038  RG     02
APTFLG                          =  00000004
APT_CON_TEST                       000006E4  R      02
APT_C_DEF                       =  000000DF
APT_DI_COMMAND                     000001B6  R      02
APT_ERROR_NUMBER                   00000012  R      02
APT_HARD_FLG                       0000001D  R      02
APT_MAIL_BOX                       00000010  R      02
APT_MEM_SIZE                       0000001C  R      02
APT_MESSAGE_CODE                   00000010  R      02
APT_PARAM                       =  00000049
APT_PASS_CNT                       00000018  RG     02
APT_RAM_ED_ADD                     00000009  R      02
APT_SOFT_FLG                       0000001E  R      02
APT_START                          000006EC  R      02
APT_SUBTEST                        00000014  R      02
APT_S_DEF                       =  00000020
APT_TEST_NUMBER                    00000016  R      02
APT_WCS_TEST                       000006D6  R      02
ARRAY_NUM_A                        000001C2  R      02
ASCII_TO_BIN                       00001045  R      02
ATTENTION                          000015F0  R      02
ATT_UPC_SAVE                       000000CC  R      02
AUTO                            =  00000005
A_ADDRS                            000001B4  R      02
B                               =  00000000
BEGIN_CHECKSUM                     000006D6  R      02
BEGIN_OF_TEST                      00001B05  R      02
BEG_TST                         =  00000080
BELL_C_DEF                      =  000000FB
BELL_S_DEF                      =  00000004
BLANK_A                            000001CD  R      02
BOARD_BUF                          000001D4  R      02
BOARD_NAME                         000001CE  R      02
BOARD_NUM                          000001DA  R      02
BOARD_PNT                          000001DB  R      02
BOARD_TABLE                        0000045B  R      02
BOOTEN                          =  00000007
BOOTF                           =  00000001
BREAK                              00001EFA  R      02
BREAK_A                            000001DD  R      02
BREAK_ADDRESS                      000001FE  R      02
BREAK_BRANCH                       00000200  R      02
BREAK_FLG                          00000203  R      02
BYTSUM                          =  00004082
B_ADDRS                            000001CC  R      02
C                               =  00000001
CALC_ADD                           00000206  R      02
CC_ADDR                         =  000000FE    G
CHANGE_WCS                         00000F52  R      02
CHECKSUM                           00001ED7  R      02
CHECK_CNTLC                        00001341  R      02
CHECK_ERR_IO                       00000F6E  R      02
CHECK_ERR_FLG                      00000F70  R      02
CHECK_LS_SETPAR                    00001C5C  R      02
CHECK_UPC                          00001E57  R      02
CHECK_WCS_LOADED                   00000F5E  R      02
CLEARPAR_ERR                    =  00000008
CLEAR_BELL                         00000A79  R      02
CLEAR_BREAK                        00000A28  R      02
CLEAR_COMMAND                      00000A5A  R      02
CLEAR_CPU_ATT                      0000012F  RG     02
CLEAR_CPU_ATT_R                    00001D43  R      02
CLEAR_DEFAULT                      00000A9A  R      02
CLEAR_FIFO                         00000A43  R      02
CLEAR_HALT                         00000A61  R      02
CLEAR_LOOP                         00000A6B  R      02
CLEAR_LOOP_COM                     00000A66  R      02
CLEAR_LS_STAT                      000015A8  R      02
CLEAR_NER                          00000A74  R      02
CLEAR_PAR                          00000AAF  R      02
CLEAR_PAR_INT                      0000195D  R      02
CLEAR_SER                          00000A7E  R      02
CLEAR_SOMM                         00000A88  R      02
CLEAR_SP                           00000A83  R      02
CLEAR_TAB                          00000582  R      02
CLEAR_TRACE                        00000A95  R      02
CLRACR                          =  000000A9
CLRACL                          =  00000033
CLRATN                          =  000000AB
CLRBSY                          =  0000002E
CLRCLK                          =  00000020
CLRCSK                          =  00000024
CLRCSR                          =  00000038
CLRDCL                          =  00000031
CLRHLT                          =  000000AF
CLRLIT                          =  0000003C
CLRMCI                          =  00000029
CLRMCK                          =  0000002B
CLRPFI                          =  000000AD
CLRSS                           =  00000022
CLRTIM                          =  000000A5
CLRTMR                          =  0000002C
CLRUPC                          =  000000A8
CLRUPK                          =  00000026
CLRWCK                          =  00000036
CNT                             =  0000089C
CNTA                            =  00000FA7
CNTLC_TYPED                        00000039  RG     02
CNTLC_VEC                          00001322  R      02
CNTLP_VEC                          00001333  R      02
CNT_PAR                            00000208  R      02
COMMAND0                           000000AF  R      02
COMMAND1                           000000B0  R      02
COMMAND2                           000000B1  R      02
COMMAND3                           000000B2  R      02
COMPARE                            000000CF  RG     02
COMPARE_R                          00001184  R      02
```

```
CONACK                          =  00000020
CONATT                          =  00000010
CONHLT                          =  00000002
CONSOLE_TEST                       00000209  R      02
CONTINUE                           0000003A  RG     02
CONT_COMMAND                       00000946  R      02
CONT_DONE                          0000020A  R      02
CONT_SAVE_CSR                      000000BE  R      02
CONT_SAVE_UPC                      000000C8  R      02
CON_ADD_DAT                        00000038  RG     02
CON_BEGIN_TEST                     00000123  RG     02
CON_BEGIN_TEST_R                   000021CD  R      02
CON_ERROR                          00000120  RG     02
CON_ERROR_R                        0000221F  R      02
CON_ERR_NUM                        0000003F  RG     02
CON_EXP_ADD                        00000044  RG     02
CON_EXP_DAT                        00000040  RG     02
CON_LOOP                        =  00000004
CON_MOD_DAT                        00000046  RG     02
CON_MSK_DAT                        0000004B  RG     02
CON_NAME_ADD                    =  00007005
CON_RAM                         =  00000080
CON_REC_ADD                        00000053  RG     02
CON_REC_DAT                        0000004F  RG     02
CON_VER_ADD                     =  00007003
CPATTN                          =  00000083
CPUACK                          =  00000082
CR                              =  0000000D  G
CRLF                               000000DE  RG     02
CRLF_A                             0000020B  R      02
CRLF_R                             0000125A  R      02
CSR07                           =  00000085
CSR15                           =  00000086
CSR23                           =  00000087
CSR_A                              0000020D  R      02
CSR_CNT                            00000212  R      02
CSR_SHIFT                          000000B7  RG     02
CSR_VALUE                          000000B8  RG     02
CTLDIS                          =  000040FE
C_EXECUTE                          00000204  R      02
C_RUNNING                          00000205  R      02
D                               =  00000002
DATA0                              000000A9  RG     02
DATA1                              000000AA  RG     02
DATA2                              000000AB  RG     02
DATA3                              000000AC  RG     02
DATA_SHIFT                         000000A8  RG     02
DATA_XFER                          00001C78  R      02
DATA_XFER_FLG                      00000213  R      02
DBUF_A                             00000214  R      02
DCLO                            =  00000002
DECR_WORD                          00000135  RG     02
DECR_WORD_R                        0000158C  R      02
DEP_COMMAND                        00000C57  R      02
DEP_CSR                            00000C76  R      02
DEP_DBUF                           00000D25  R      02
DEP_ICSR                           00000D19  R      02
DEP_IDAR                           00000D1F  R      02
DEP_LS                             00000CEC  R      02
DEP_MCT                            00000D01  R      02
DEP_MM                             00000D07  R      02
DEP_OS                             00000CDE  R      02
DEP_RAM                            00000C5D  R      02
DEP_SUB                            00000D30  R      02
DEP_SUB_NOADD                      00000D2B  R      02
DEP_TAB                            0000065F  R      02
DEP_TB                             00000D0D  R      02
DEP_UBS                            00000D13  R      02
DEP_UPC                            00000C89  R      02
DEP_WCS                            00000CC2  R      02
DEP_WRK                            00000C95  R      02
DIAG_BOARD                         0000079D  R      02
DIAG_COMMAND                       0000070C  R      02
DIAG_CONTIN                        0000089C  R      02
DIAG_PASS                          000008A2  R      02
DIAG_SECTION                       000007EB  R      02
DIAG_SHORTEN                       000008C7  R      02
DIAG_TAB                           0000055F  R      02
DIAG_TEST                          00000858  R      02
DIR_COMMAND                        00001DF1  R      02
DIR_VECTOR                      =  0000411A
DISMEM                          =  000000A7
DISPAR                          =  000000A1
DOLOOP                             00000055  RG     02
DOT_A                           =  0000002E
DO_COMMANDS                        0000164E  R      02
DRIVE_A                            00000219  R      02
E                               =  00000003
EMEMREF                         =  00000040
ENBMEM                          =  000000A6
ENBPAR                          =  000000A0
END_CHECKSUM                       00002308  R      02
ENK_A                              00000221  R      02
ENK_END                            00000224  R      02
EOP                                00001AB2  R      02
EOP_A                              00000227  R      02
EOS                             =  00000040
EOS_FLG                            00000056  RG     02
EQUALS_A                           0000022D  R      02
ERROR                              00000231  R      02
ERROR_A                            00000232  R      02
ERROR_CON                          00000057  RG     02
ERROR_ROUTINE                      00000F7E  R      02
ERROR_ROUTINE_NO_RET               00000F79  R      02
EXAM_COMMAND                       00000AE1  R      02
EXAM_CSR                           00000B19  R      02
EXAM_DBUF                          00000BF2  R      02
EXAM_DEP_SUB                       00000C36  R      02
EXAM_ICSR                          00000BE0  R      02
EXAM_IDAR                          00000BE9  R      02
EXAM_LS                            00000B7C  R      02
EXAM_MCT                           00000B92  R      02
EXAM_MM                            00000BAD  R      02
EXAM_OS                            00000BB6  R      02
```

```
EXAM_PATT                       00000BFB R      02
EXAM_POSIT                      00000C04 R      02
EXAM_RAM                        00000AEF R      02
EXAM_SUB                        00000C12 R      02
EXAM_SUB_NOADD                  00000C0D R      02
EXAM_TAB                        00000606 R      02
EXAM_TB                         00000BA4 R      02
EXAM_TEMP                       00000239 R      02
EXAM_UBS                        00000B9B R      02
EXAM_UPC                        00000B2C R      02
EXAM_WCS                        00000BCA R      02
EXAM_WRK                        00000B4E R      02
EXECUTE                         0000023B R      02
EXEC_CONTESTS                 = 00007000
EXEC_SUB                        00000A49 R      02
EXEC_TESTO                      00001E8A R      02
EXPECT_LAB                    = 0000008A
EXP_LABEL                       00000441 R      02
EXP_LAB_DAT                     00000442 R      02
EXT_A                           0000023C R      02
FAST_WRITE                      00000240 R      02
FILE_LOADED                     00000241 R      02
FLAGS                           0000005A RG     02
FLAG_LOC                      = 00000048
FOUR_BYTES                      00001D0B R      02
FPA_A                           00000242 R      02
FPA_PRES                      = 00000010
F_WORD                          00000058 RG     02
GCVECT                        = 00004129  G
GLVECT                        = 00004123
GSVECT                        = 00004145
H                             = 00000004
HALT_C_DEF                    = 000000F7
HALT_S_DEF                    = 00000008
HAVE_CNT                        00000283 R      02
HEAD                          = 00000000
HEADER_A                        00000247 R      02
HEADER_B                        00000265 R      02
HEX_0D                        = 0000000D
HEX_1                         = 00000001
HEX_10                        = 00000010
HEX_1F                        = 0000001F
HEX_20                        = 00000020
HEX_21                        = 00000021
HEX_2D                        = 0000002D
HEX_3                         = 00000003  G
HEX_30                        = 00000030
HEX_3A                        = 0000003A
HEX_3F                        = 0000003F
HEX_4                         = 00000004
HEX_40                        = 00000040
HEX_41                        = 00000041
HEX_50                        = 00000050
HEX_7                         = 00000007
HEX_7F                        = 0000007F
HEX_8                         = 00000008
HEX_80                        = 00000080
HEX_BUF                         0000005B RG     02
HEX_FF                        = 000000FF
HLTFLG                        = 00000002
ICSR_A                          00000284 R      02
IDAR_A                          00000289 R      02
IDL_A                           0000028E R      02
IDC_PRES                      = 00000080
INC_TEMPW_5                     00001575 R      02
INC_WORD                        00000108 RG     02
INC_WORD_R                      00001581 R      02
INITH                         = 00000035
INITL                         = 00000034
INPUT_CHAR_IF                   0000131B R      02
INPUT_LINE                      0000126B R      02
INPUT_NUM                       0000043D R      02
INPUT_NUM_CHECK                 00000F6B R      02
INPUT_NUM_IF                    00000FCF R      02
INT_CLEAR_PAR                   00000AB5 R      02
INVTIM                        = 00000008
IN_BIN_STRING                   00000DC0 R      02
ITCSR                         = 0000001D
ITDB                          = 0000001C
L                             = 00000005
LD_UPC                          00000117 RG     02
LD_UPC_R                        000022D7 R      02
LEAVE_DO                        00001E55 R      02
LF                            = 0000000A  G
LITERAL                       = 00000001
LOADER                          00000939 R      02
LOAD_CSR                        000000F0 RG     02
LOAD_CSR_R                      00001116 R      02
LOAD_SECTION                    00001D6C R      02
LOAD_UPC                        000000F3 RG     02
LOAD_UPC_R                      00001148 R      02
LOOPER                          00001A3C R      02
LOOPING                       = 00000002
LOOP_CNT_ADD                  = 00000047
LOOP_COMMAND                    00000AC2 R      02
LOOP_COM_C_DEF                = 000000BF
LOOP_COM_S_DEF                = 00000040
LOOP_COUNTER                    00000293 R      02
LOOP_C_DEF                    = 000000BE
LOOP_ERROR                      00001A63 R      02
LOOP_S_DEF                    = 00000001
LS_5                          = 00000005
LS_6                          = 00000006
LS_7                          = 00000007
LS_A                            00000294 R      02
LS_F                          = 0000000F
LS_PARERR_ACT                   00000299 R      02
LS_SETPARERR                    00000298 R      02
LVC                           = 00000000
M                             = 00000006
M1STK1                        = 00000FA6
M1STK2                        = 00000FA4
M1STK3                        = 00000FA7
MACSTK1                       = 0000089C
```

```
MACSTK2                         = 0000089A
MACSTK3                         = 00000894
MACSTK4                         = 0000088B
MACSTK5                         = 0000088C
MACSTK6                         = 0000088F
MACSTK7                         = 0000088E
MAIN_TAB                          000004F5 R      02
MAKE_A                            00001478 R      02
MAKE_B                            00001462 R      02
MAKE_XD                           0000010B RG     02
MAKE_XD_R                         00001450 R      02
MCPU_A                            00000140 RG     02
MCT_A                             0000029A R      02
MEM_REQ                           0000029F R      02
MEM_SIZE_A                        000002A0 R      02
MFPA_A                            0000014A R      02
MICFLAG                           0000002E R      02
MICRO_STEP                        000002C8 R      02
MICRO_STEPER                      00001521 R      02
MICRO_STEP_CPU                    00000138 RG     02
MICRO_STEP_CPU_R                  00001570 R      02
MIDC_A                            00000154 R      02
MINUS                             000002C9 R      02
MISC2                           = 00000001
MMCT_A                            00000145 R      02
MMEM_A                            0000014F R      02
MM_A                              000002CA R      02
MM_OR_LS                        = 00000001
MM_SIZE                         = 00000008
MOD_NOT_PRES_A                    000002CF R      02
MOD_PRES_A                        000002DC R      02
MOD_PRES_FLG                    = 00000001
MORE_DATA                         000002E5 R      02
MOVER                             000000D2 RG     02
MOVER_3                           0000011A RG     02
MOVER_3_R                         0000118E R      02
MOVER_4                           00000111 RG     02
MOVER_4_R                         000011A2 R      02
MOVER_R                           000011A4 R      02
MOV_DATAO_HEXBUF                  0000119C R      02
MOV_FROM_DATAO                    0000119F R      02
MOV_INPUT_DATAO                   00001193 R      02
MOV_LOAD_SECTION                  00001D61 R      02
MR80_A                            0000015E R      02
MRL02_A                           00000159 R      02
MSK_BYTE                          00002283 R      02
MSK_ERROR                         0000011D RG     02
MSK_ERROR_R                       0000223C R      02
MWCS_A                            0000013B RG     02
NA                              = 00000080
NA_A                              000002E6 R      02
NA_EXP_REC                        0000005F RG     02
NA_OTHER                          00000060 RG     02
NEG_CNT                         = 00000080
NER_C_DEF                       = 000000BD
NER_S_DEF                       = 00000002
NEWSUM                            000002F0 R      02
NOADD_FLG                         000002F1 R      02
NOCHK                           = 000040FF
NOINIT                            000002F2 R      02
NOP_CSR                           000000E4 RG     02
NOP_CSR_R                         000010F9 R      02
NOP_INST                          00000072 RG     02
NOT_FOUND                         000002F6 R      02
NO_RETURN                         000002F3 R      02
NO_SAVE_UPC_CSR                   000002F4 R      02
NO_SPACE                          000002F5 R      02
NUM_SHIFT                         0000043C R      02
OFLAG                           = 000040BB
OKV15                           = 00000007
OLDSUM                            00000306 R      02
OLD_INSTRUCTION                   000002F7 R      02
OLD_TEMP_BUF                      000002FA R      02
OLD_TTBUF                         00000304 R      02
OPTIONAL_TST                    = 00000040
OS_A                              00000307 R      02
OS_ADD                          = 0000007C    G
OUT_BIN_STRING                    00000E75 R      02
OVERLAY_RAM                       00000003 RG     02
OVERLAY_WCS                       00000006 RG     02
PACK_ERROR                        0000030B R      02
PACK_LABEL                        000018E4 R      02
PAR                               00001FDE R      02
PARDONE                           0000030F R      02
PARITY_CALC                       00000EFF R      02
PARITY_ERR                        00001BEE R      02
PARITY_JUMP                       00000061 RG     02
PARITY_ON                         00000064 RG     02
PARSER                            00002022 R      02
PARSE_PNT                         00000310 R      02
PARSE_TABLE                       00000312 R      02
PAR_ON                            0000030C R      02
PAR_TEMP_BUF                      00001FDB R      02
PAR_VECT                          0000030D R      02
PASS_CNT                          00000314 R      02
PATT_A                            00000316 R      02
PERFORM_CSR                       000000ED RG     02
PERFORM_CSR_R                     000015C2 R      02
POSIT_A                           0000031B R      02
POWER_BRANCH                      00000065 RG     02
POWER_REC                         00001F55 R      02
POWER_SEQ                       = 0000001B
PRINT                             000000D5 RG     02
PRINTER                           00001243 R      02
PRINT_CNT                         00000320 R      02
PRINT_CSR_VAL                     00000FC1 R      02
PRINT_DRIVE                       00001926 R      02
PRINT_FLG                         000011AD R      02
PRINT_HEX                         000000D8 RG     02
PRINT_HEX_1                       00000114 RG     02
PRINT_HEX_1_R                     000011C2 R      02
PRINT_HEX_2                       000011BD R      02
PRINT_HEX_4                       000011B8 R      02
PRINT_HEX_APT                     000011C9 R      02
```

```
PRINT_HEX_R                     000011C4 R      02
PRINT_MM_SIZE                   0000193A R      02
PRINT_MOD_PRES                  000019A7 R      02
PRINT_PNT                       00000321 R      02
PRINT_PROMPT                    00001208 R      02
PRINT_R                         00001265 R      02
PRINT_SINGLE                    0000196D R      02
PRINT_STRING                    0000010E RG     02
PRINT_STRING_APT                00001254 R      02
PRINT_STRING_R                  0000124A R      02
PRINT_TRACE                     00001B35 R      02
PRINT_UPC                       00000B45 R      02
PRINT_VERSION                   00002143 R      02
PRI_STACK                   =   00004080
PROMPT_A                        00000323 R      02
PRT_FLAGS_A                     000006AE R      02
PRT_MOD_A                       0000013B R      02
PR_HEX_DIGIT                    000011FC R      02
PWRFAIL                     =   00000004
R80_A                           00000329 R      02
R80_PRES                    =   00000080
RAM_A                           0000032E R      02
RAM_LOAD_ADD                =   00007000
READ                        =   00000080
READJ1                      =   00000003
READJ2                      =   00000004
READ_DATA_32                    000000FC RG     02
READ_DATA_32_R                  00001481 R      02
READ_LS                         00000102 RG     02
READ_LS_7                       00001433 R      02
READ_LS_R                       00001435 R      02
READ_WCS                        000000E7 RG     02
READ_WCS_DATA                   00001D30 R      02
READ_WCS_R                      0000109D R      02
READ_WCS_SUB                    000021C0 R      02
RECEIVE_TAB                 =   0000008E
REC_LABEL                       0000044E R      02
REC_LAB_DAT                     0000044F R      02
REMOTE                      =   00000006
REPEAT                          00000333 R      02
REPEAT_COMMAND                  00000AD6 R      02
RESET_TIMOUT                    00001BE6 R      02
RESTART                         00000334 R      02
RESTART_CPU                     000013DD R      02
RESTORE_WR                      00000F41 R      02
RETURN                          0000055E R      02
RET_VECTOR                  =   0000414D
ROM_X_ADDRS                 =   0000000B
RSVECT                      =   0000413D
RUNNING                         00000335 R      02
SAVED_CSR                       000000BB R      02
SAVED_UPC                       000000C4 R      02
SAVE_A                          00000336 R      02
SAVE_BC                         00000337 R      02
SAVE_C                          00000339 R      02
SAVE_DE                         0000033B R      02
SAVE_HL                         0000033D R      02
SAVE_SP                         0000033F R      02
SAVE_STACK                      00000341 R      02
SAVE_WR                         00000F47 R      02
SCVECT                      =   00004138
SECTION_B_NUM                   00000343 R      02
SELTION_FLAG                    000004DE R      02
SECTION_NAME                    000004DF R      02
SEC_LEN                     =   00000006
SEC_PARMB_ADD               =   00004111
SEC_PARMB_DEF               =   00004150
SEC_PARMB_DST               =   00004115
SEC_PARMB_ERR               =   00004116
SEC_PARMB_EXT               =   0000410D
SEC_PARMB_NAM               =   00004107
SEC_PARMB_UNIT              =   00004106
SEC_TABLE                       00000474 R      02
SEL_FIFO_A                      00000A37 R      02
SEL_FIFO_B                      00000A3D R      02
SER_C_DEF                   =   000000EF
SER_S_DEF                   =   00000010
SETACR                      =   000000A8
SETACK_FLG                      00000344 R      02
SETACL                      =   00000032
SETATN                      =   000000AA
SETBSY                      =   0000002F
SETCLK                      =   00000021
SETCSK                      =   00000025
SETCSR                      =   00000039
SETDCL                      =   00000030
SETHLT                      =   000000AE
SETLIT                      =   0000003D
SETMCI                      =   00000028
SETMCK                      =   0000002A
SETPAR_ERR                  =   00000002
SETPFI                      =   000000AC
SETSS                       =   00000023
SETTIM                      =   000000A4
SETTMR                      =   0000002D
SETUPC                      =   000000A9
SETUPK                      =   00000027
SETWCK                      =   00000037
SET_ALL_TESTS                   00002134 R      02
SET_BELL                        00000996 R      02
SET_BREAK                       00000A00 R      02
SET_COMMAND                     0000097D R      02
SET_FOR_CONT                    0000012C RG     02
SET_FOR_CONT_R                  00001349 R      02
SET_HALT                        00000984 R      02
SET_LOOP                        0000098C R      02
SET_LS_STAT                     0000159E R      02
SET_NER                         00000991 R      02
SET_PAR                         000009E9 R      02
SET_PARDONE                     00001EF4 R      02
SET_PAR_INT                     0000194C R      02
SET_SER                         0000099E R      02
SET_SIGNALS                     000019BD R      02
SET_SOMM                        000009A8 R      02
```

```
SET_SP                          000009A3 R      02
SET_STEP                        000009C6 R      02
SET_TAB                         000005BF R      02
SET_TRACE                       000009E3 R      02
SHIFTER                         000000E1 RG     02
SHIFTER1                        00000132 RG     02
SHIFTER1_R                      000010F0 R      02
SHIFTER_R                       000010E7 R      02
SHIFT_L_2901                    000000F6 RG     02
SHIFT_L_2901_R                  00001515 R      02
SHIFT_PNT                       00000068 RG     02
SHIFT_R_2901                    000000F9 RG     02
SHIFT_R_2901_R                  0000151B R      02
SHORTEN                         00000345 R      02
SHOWFLAG_A                      00000346 R      02
SHOW_COMMAND                    000008D5 R      02
SIGNAL                        = 00000001
SINGLE_A                        0000034D R      02
SINGLE_ERR                    = 00000004
SKIP_CNT                        0000037B R      02
SKIP_TEST                       0000006A RG     02
SKIP_TO_SPEC                    00001063 R      02
SLASH_U                         0000037C R      02
SLOW_WRITE_A                    0000037D R      02
SLVEC1                        = 00004133
SLVECT                        = 0000412E
SOMM_A                          00000397 R      02
SOMM_ADDRESS                    0000039D R      02
SOMM_FLG                        0000039F R      02
SPACE                           000000DB RG     02
SPACE_A                       = 00000020
SPACE_FOUND                     000003A0 R      02
SPACE_R                         00001260 R      02
SPBUFF                        = 00004086
SPECIAL_C_DEF                 = 0000007F
SPECIAL_S_DEF                 = 00000080
START                           00000D4C R      02
STARTING_UPC                    000003A1 R      02
START_CPU                       000013CE R      02
START_CPU_COM                   000013F2 R      02
START_UP                        00001F6D R      02
STATUS                        = 00000040
STEP_CNT                        000003A3 R      02
STNDRD_CON                    = 00000000
STNDRD_CON_COM                  00000705 R      02
STOP_CPU                        00001399 R      02
SUB_ATTENTION                   00001A7A R      02
SYM                           = 0000089C
SYMA                          = 00000FA6
SYNTAX_A                        000003A5 R      02
SYNTAX_ERROR                    00001EC9 R      02
S_ED_WRK                        0000042E R      02
S_READ_LS                       00000423 R      02
S_WRITE_LS                      00000435 R      02
TABLE_MODE                      000003B3 R      02
TABLE_PNT                       000003B4 R      02
TB_A                            000003B6 R      02
TEMP_BUF                        000003BB R      02
TEMP_BYTE                       000003C5 R      02
TEMP_CSR                        00000439 R      02
TEMP_FLAG                       000003CC R      02
TEMP_SECTION                    000003CD R      02
TEMP_SHIFT                      00000438 R      02
TEMP_WORD                       000003C6 R      02
TEMP_WORD1                      000003C8 R      02
TEMP_WORD2                      000003CA R      02
TEST_A                          000003D3 R      02
TEST_NUM                        000003D9 R      02
TEST_NUM1                       000003DA R      02
TEST_NUM2                       000003DB R      02
TEST_ZERO                       000003DC R      02
TIMER_VEC                       00001F6C R      02
TIMOUT                          00001BBB R      02
TIMOUT_CNT                      000003DD R      02
TRACE                           000003DF R      02
TTBUF                           00000164 R      02
TTBUF_CNT                       00000163 R      02
TTBUF_PNT                       000003E0 R      02
TTCR                          = 0000004B
TTDB                          = 00000048
TTMODE                        = 0000004A
TTSR                          = 00000049
TU58_DRIVER                   = 00004117
TUCR                          = 00000047
TUDB                          = 00000044
TUMODE                        = 00000046
TUSR                          = 00000045
TYPE_CHAR                       00001265 R      02
UBE_A                           000003E2 R      02
UBE_PRES                      = 00000040
UBS_A                           000003E7 R      02
UBS_BUSY                      = 00000040
UBS_DC_LO                     = 00000020
UBS_INIT                      = 00000010
UPC14                         = 00000084
UPC14A                        = 00000000
UPC_A                           000003EC R      02
UPC_COMPARE                     000000CA R      02
UPC_SHIFT                       000000C1 R      02
UPC_SUB                         000000C6 R      02
UPC_VALUE                       000000C2 RG     02
VERIFY_SEQ                      00001B55 R      02
VERIFY_WRITER                   00001DFA R      02
VERSION                         000003F5 R      02
VER_A                           000003F2 R      02
VNORML                        = 000000A3
VSHIFT                        = 000000A2
WARM_RESTART                    00000401 R      02
WCSB0                         = 00000008
WCSB1                         = 00000009
WCSB2                         = 0000000A
WCSERR                        = 00000001
WCS_A                           00000402 R      02
WCS_ADDRESS                     0000006C RG     02
```

```
WCS_ADDR_DAT                    = 00000044
WCS_BAD_PARITY                    0000006E RG     02
WCS_CALC                          00000407 R      02
WCS_CL_FIFO                     = 00000062
WCS_DAT_32_ADD                  = 00000600
WCS_DE_DBUF                     = 00000061
WCS_DE_ICSR                     = 0000005F
WCS_DE_IDAR                     = 00000060
WCS_DE_MCT                      = 00000050
WCS_DE_MM                       = 00000054
WCS_DE_TB                       = 00000052
WCS_DE_UBS                      = 00000058
WCS_ERROR                         0000171D R      02
WCS_ERR_NUM                     = 00000041
WCS_EXP_DAT                     = 00000042
WCS_EX_DBUF                     = 0000005C
WCS_EX_ICSR                     = 0000005A
WCS_EX_IDAR                     = 0000005B
WCS_EX_MCT                      = 00000051
WCS_EX_MM                       = 00000055
WCS_EX_PATT                     = 0000005D
WCS_EX_POSIT                    = 0000005E
WCS_EX_TB                       = 00000053
WCS_EX_UBS                      = 00000059
WCS_FILE_NAME                   = 00000607
WCS_LOADED                        00000408 R      02
WCS_LOAD_ADD                    = 00000000
WCS_MOD_DAT                     = 00000046
WCS_MSK_DAT                     = 00000045
WCS_PNT                           00000409 R      02
WCS_READ                          00000126 RG     02
WCS_READ_R                        0000228B R      02
WCS_REC_DAT                     = 00000043
WCS_REST_WR                     = 00000057
WCS_SAVE_WR                     = 00000056
WCS_SEL_FIFOA                   = 00000063
WCS_SEL_FIFOB                   = 00000064
WCS_SHIFT                         0000006B RG     02
WCS_VALUE                         000000B4 RG     02
WCS_VAL_SHIFT                     000000B3 RG     02
WCS_VER_ADD                     = 00000606
WCS_WRITE                         00000129 RG     02
WCS_WRITE_R                       000022C1 R      02
WRITE                           = 000000CC
WRITE_DATA_32                     000000FF RG     02
WRITE_DATA_32_R                   0000149E R      02
WRITE_LS                          00000105 RG     02
WRITE_LS_R                        00001415 R      02
WRITE_LS_STAT                     000015AF R      02
WRITE_TAB                         000004E5 R      02
WRITE_TAB_PNT                     0000006F RG     02
WRITE_WCS                         000000EA RG     02
WRITE_WCS_R                       000010B9 R      02
WRK_A                             0000040A R      02
WRONG_LAB                       = 00000020
XC                                00000D6E R      02
XD_ADDRS                          00000071 RG     02
XFER                            = 00000020
XFER_CNT                          00000413 R      02
XFER_DEST                         00000415 R      02
XFER_MM                           00000418 R      02
XFER_PNT                          00000419 R      02
XU                                00000D66 R      02
X_CLR_CPU_ATT                     00000426 R      02
X_CLR_PAGE                        0000007D RG     02
X_CLR_WRO                         000000A1 RG     02
X_COMMAND                         00000D72 R      02
X_COM_ADDRS                       0000040F R      02
X_COM_WRO                         000000A5 RG     02
X_ED_WRK                          0000042A R      02
X_EX_C                            00000EB9 R      02
X_EX_U                            00000EA7 R      02
X_MOV_WRWR                        0000009E RG     02
X_READ_LS                         0000041F R      02
X_ROT_C                           00000079 RG     02
X_SET_PAGE                        00000081 RG     02
X_SHIFT_L                         00000075 RG     02
X_SHIFT_R                         00000085 RG     02
X_WRITE_LS                        00000431 R      02
Y                               = 00000055
YBUSRD                          = 000000EC
ZERO_WORD                         0000041B R      02
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
. BLANK .	00000000 (0.)	01 (1.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
MICMON	00002308 (8968.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	22	00:00:00.05	00:00:00.29
Command processing	34	00:00:00.22	00:00:00.83
Pass 1	2702	00:01:06.83	00:03:43.44
Symbol table sort	2	00:00:01.49	00:00:05.51
Pass 2	2880	00:00:26.39	00:01:47.31
Symbol table output	52	00:00:00.56	00:00:02.57
Psect synopsis output	3	00:00:00.02	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	5699	00:01:35.57	00:05:39.97

The working set limit was 1000 pages.
713745 bytes (1395 pages) of virtual memory were used to buffer the intermediate code.
There were 50 pages of symbol table space allocated to hold 766 non-local and 250 local symbols.
9373 source lines were read in Pass 1, producing 76 object records in Pass 2.
148 pages of virtual memory were used to define 147 macros.

Macro library statistics

Macro library name	Macros defined
SYS$SYSROOT:[SYSLIB]STARLET.MLB;1	0

0 GETS were required to define 0 macros.

There were no errors, warnings or information messages.

/SHOW=(BIN)/LIST=ENKAA.LIS/OBJECT=ENKAA.OBJ 8085MAC.MAR+MACDEF.MAC+ENKAA.MAC

Fiche	Frame	Sequence		
1	B1	1	Documentation	
1	K2	23	Map	
1	N2	26	4C00 4	
1	L12	154	Symbol table	14-JUN-1983
1	F13	161	Psect synopsis	14-JUN-1983
1	G13	162	Directory	14-JUN-1983