



The main body of the document is a microfiche card containing a grid of frames. Each frame contains a page of text, likely technical specifications or code for the VAX 11/730 micro-monitor. The text is too small to be legible in this image. The frames are arranged in approximately 15 rows and 10 columns.

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

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

The software described in this document is furnished under a license and may be used or copied only in accordance with the terms of such license.

No responsibility is assumed for the use or reliability of software on equipment that is not supplied by Digital or its affiliated companies.

Copyright (c) 1982 by Digital Equipment Corporation. All Rights Reserved.

The following are trademarks of Digital Equipment Corporation.

DEC	DECsystem-10	DECSYSTEM-20
DECUS	MASSBUS	PDP
UNIBUS	VAX	VMS

digital

Table of Contents

1.0	ABSTRACT . . . . .	3
2.0	HARDWARE REQUIREMENTS . . . . .	3
3.0	SOFTWARE REQUIREMENTS . . . . .	3
4.0	PREREQUISITES . . . . .	3
5.0	OPERATING INSTRUCTIONS . . . . .	4
5.1	Options . . . . .	18
5.2	Event Flags . . . . .	18
5.3	APT Setup . . . . .	18
6.0	PROGRAM FUNCTIONAL DESCRIPTION . . . . .	19
6.1	Program Overview . . . . .	19
6.2	Program Size . . . . .	20
6.3	Program Run Times . . . . .	20
6.4	Run-time Dynamics . . . . .	20
6.5	Fault Detection . . . . .	20
6.6	Performance During Hardware Failures . . . . .	21
6.7	Program Applications . . . . .	21
6.8	Test Descriptions . . . . .	21
7.0	MAINTENANCE HISTORY . . . . .	22

## 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 other 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 successfully, 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 its internal tables.

#### SECTION keyword

This function is incompatible 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 incompatible with the following keywords:

SECTION TEST

NOTE: If an error occurs and the SHORTEN keyword was used this will cause MICMON to exit from its 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 incompatible 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 intermittent 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 independent 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 syn. 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 reenables 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 instruction 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 R/1.

MIC>SE BF

#### Clear Fifo

This command executes code in WCS that clears the FIFO address reg currently selected. It should be preceded 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[IT]*#
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 controller'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.





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 occurred. 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 DF (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



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 = TUS8 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 transferred 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 beginning 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 4C00(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 L. 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	.MAIN.	00004C00	00006F07	00002308 (	8968.)	BYTE 0 NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC
		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	00004C4B-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_WRO	00004CA1-R		
DATA_SHIFT	00004CAB-R	PERFORM_CSR	00004CED-R	X_COM_WRO	00004CA5-R		
DECR_WORD	00004D35-R	POWER_BRANCH	00004C65-R	X_MOV_WRRR	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				

ZZ-ENKAA-2.1 Map  
DRBO:[GREEN.CRD]ENKAA.EXE;3

L 2  
14-JUN-1983 10:33

Fiche 1 Frame L2  
VAX-11 Linker V03-03

Sequence 24 Page 2

<u>Symbol</u>	<u>Value</u>	<u>Symbol</u>	<u>Value</u>	<u>Symbol</u>	<u>Value</u>	<u>Symbol</u>	<u>Value</u>
---------------	--------------	---------------	--------------	---------------	--------------	---------------	--------------

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: 0. 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=ZX4C00/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 DJ B0 WCS command.









```

ZZ-FNKA-2.1 4C00 4
4C03 300 000000AB
4C03 301
4C03 302 000000AC
4C03 303 000000AD
4C03 304
4C03 305 000000AE
4C03 306 000000AF
4C03 307
4C03 308 000000CC
4C03 309
4C03 310 000000EC
4C03 311
4C03 312
4C03 313
4C03 314
4C03 315
4C03 316
4C03 317
4C03 318
4C03 319
4C03 320
4C03 321 00000001
4C03 322 00000049
4C03 323 00000080
4C03 324 00004082
4C03 325 000000FE
4C03 326 00000008
4C03 327 00000010
4C03 328 00000020
4C03 329 00000002
4C03 330 00000004
4C03 331 00000080
4C03 332 00007003
4C03 333
4C03 334 00007005
4C03 335
4C03 336 0000000D
4C03 337 000040FE
4C03 338 0000411A
4C03 339 0000002E
4C03 340 00000040
4C03 341 00000040
4C03 342 00007000
4C03 343 0000008A
4C03 344
4C03 345 00000048
4C03 346 00000010
4C03 347 00004129
4C03 348 00004123
4C03 349 00004145
4C03 350 00000001
4C03 351 00000003
4C03 352 00000004
4C03 353 00000007
4C03 354 00000008
4C03 355 00000010
4C03 356 0000000D
4C03 357 0000001F
4C03 358 00000020
4C03 359 00000021

```

F 3

Fiche 1 Frame F3 Sequence 31

```

CLRATN      = <^XAB>      ;(WO) CLEAR CONSOLE ATTENTION
SETPFI      = <^XAC>      ;(WO) SET POWER FAIL INTERRUPT
CLRPF1      = <^XAD>      ;(WO) CLEAR POWER FAIL INTERRUPT
SETHLT      = <^XAE>      ;(WO) SET THE CONSOLE HALT BIT
CLRHLT      = <^XAF>      ;(WO) CLEAR THE CONSOLE HALT BIT
WRITE       = <^XCC>      ;(WO) THE CONSOLE WRITE REGISTER
YBUSRD      = <^XEC>      ;(RD) Y-BUS READ

```

```

:*****
: THE FOLLOWING EQUATES ARE USED AS PROGRAM CONSTANTS
: THIS IS DONE WHERE EVER POSSIBLE TO CONTROL THE USAGE OF THESE
: CONSTANTS -- SO THEY ARE DEFINED ONLY IN ONE PLACE --
:*****

```

```

ADDR         = <^X1>       ;BIT 24 COMMAND + STATUS WORD
APT_PARAM    = <^X49>      ;LS LOCATION FOR APT PARAMETERS
BEG_TST      = <^X80>      ;BIT 15 COMMAND + STATUS WORD
BYTSUM       = <^X4082>    ;ADDRESS OF CHECKSUM IN BOOT BLOCK
CC_ADDR      = <^XFE>      ;LS ADDR OF ALU CC REG
CLEARPAR_ERR = <^X8>       ;BIT 27 COMMAND + STATUS WORD
CONATT       = <^X10>      ;BIT 20 COMMAND + STATUS WORD
CONACK       = <^X20>      ;BIT 21 COMMAND + STATUS WORD
CONHLT       = <^X2>       ;BIT 17 COMMAND + STATUS WORD
CON_LOOP     = <^X4>       ;BIT 10 COMMAND + STATUS WORD
CON_RAM      = <^X80>      ;BIT 7 ON = SECT FOR CONSOLE RAM
CON_VER_ADD  = <^X7003>    ;ADDRESS IN 8085 BASED TEST WHERE
                           ; VERSION NUMBER IS FOUND (2 BYTES)
CON_NAME_ADD = <^X7005>    ;ADDRESS IN 8085 BASED TEST WHERE
                           ; FILE NAME IS FOUND (LAST 2 BYTES)
CR           = <^X0D>      ;HEX VALUE OF CR
CTLDIS      = <^X40FE>    ;BOOT BLOCK DISABLE CNTL CHAR FLAG
DIR_VECTOR   = <^X411A>    ;ADDRESS OF DIR COMMAND IN BOOT BLOCK
DOT_A        = <^X2E>      ;ASCII FOR DOT
EMEMREF      = <^X40>      ;BIT 22 COMMAND + STATUS WORD
EOS          = <^X40>      ;BIT 14 COMMAND + STATUS WORD
EXEC_CONTESTS = <^X7000>  ;BEGINNING OF TESTS = RAM LOAD ADDRESS
EXPECT_LAB   = <^X8A>     ;LS LOCATION OF FIRST BYTE OF EXPECTED
                           ; RLO2 PACK LABEL
FLAG_LOC     = <^X48>     ;LS LOCATION OF FLAG INFO TO WRITE
FPA PRES    = <^X10>      ;BIT 12 COMMAND + STATUS WORD
GCVECT      = <^X4129>    ;ADDR OF GET CHAR ROUTINE IN BOOT BLOCK
GLVECT      = <^X4123>    ;ADDR OF GET LINE ROUTINE IN BOOT BLOCK
GSVECT      = <^X4145>    ;ADDR OF GET SILO ROUTINE IN BOOT BLOCK
HEX_1       = <^X1>       ;HEX VALUE OF 1
HEX_3       = <^X3>       ;HEX VALUE OF 3
HEX_4       = <^X4>       ;HEX VALUE OF 4
HEX_7       = <^X7>       ;HEX VALUE OF 7
HEX_8       = <^X8>       ;HEX VALUE OF 8
HEX_10      = <^X10>     ;HEX VALUE OF 10
HEX_0D      = <^X0D>     ;HEX VALUE OF D
HEX_1F      = <^X1F>     ;HEX VALUE OF 1F
HEX_20      = <^X20>     ;HEX VALUE OF 20
HEX_21      = <^X21>     ;HEX VALUE OF 21

```



```

ZZ-ENKAA-2.1 4C00 4
4C03 420 00000060
4C03 421 00000061
4C03 422 00000050
4C03 423 00000054
4C03 424 00000052
4C03 425 00000058
4C03 426 00000041
4C03 427 0000005C
4C03 428 0000005A
4C03 429 0000005B
4C03 430 00000051
4C03 431 00000055
4C03 432 0000005D
4C03 433 0000005E
4C03 434 00000053
4C03 435 00000059
4C03 436 00000042
4C03 437 00000000
4C03 438 00000046
4C03 439 00000045
4C03 440 00000043
4C03 441 00000057
4C03 442 00000056
4C03 443 00000063
4C03 444
4C03 445 00000064
4C03 446
4C03 447 00000066
4C03 448 00000067
4C03 449 00000020
4C03 450 00000020
4C03 451
4C03 452
4C03 453
4C03 454
4C03 455
4C03 456
4C03 457
4C03 458
4C03 459
4C03 460 00000001
4C03 461 00000040
4C03 462 00000080
4C03 463
4C03 464 00000002
4C03 465 00000004
4C03 466 00000008
4C03 467 00000010
4C03 468 00000020
4C03 469
4C03 470
4C03 471
4C03 472 000000BE
4C03 473
4C03 474 000000BF
4C03 475 0000007F
4C03 476
4C03 477 000000BD
4C03 478
4C03 479 000000FB

```

H 3

Fiche 1 Frame M3 Sequence 33

```

;ADDRESS OF DEPOSIT IDC DAR SUBROUTINE
;ADDRESS OF DEPOSIT IDC DBUF SUBROUTINE
;ADDRESS OF DEPOSIT MCT SUBROUTINE
;ADDRESS OF DEPOSIT MM SUBROUTINE
;ADDRESS OF DEPOSIT TB SUBROUTINE
;ADDRESS OF DEPOSIT UBS SUBROUTINE
;ADDRESS OF ERROR NUMBER DATA WORD
;ADDRESS OF EXAMINE IDC DATA BUF
;ADDRESS OF EXAMINE IDC CSR
;ADDRESS OF EXAMINE IDC DAR
;ADDRESS OF EXAMINE MCT SUBROUTINE
;ADDRESS OF EXAMINE MM SUBROUTINE
;ADDRESS OF EXAMINE IDC ECC PATTERN
;ADDRESS OF EXAMINE IDC ECC POSITION
;ADDRESS OF EXAMINE TB SUBROUTINE
;ADDRESS OF EXAMINE UBS SUBROUTINE
;ADDRESS OF EXPECTED DATA WORD
;ADDRESS WCS OVERLAYS LOAD AT.
;ADDRESS OF MODULE NUMBER DATA WORD
;ADDRESS OF ERROR MASK DATA WORD
;ADDRESS OF RECEIVED DATA WORD
;RESTORE 2901 WORKING REGS SUBR ADDR
;SAVE 2901 WORKING REGS SUBR ADDR
;ADDRESS OF SELECT IDC FIFO A
;SUBROUTINE
;ADDRESS OF SELECT IDC FIFO B
;SUBROUTINE
;ADDRESS OF WCS BASED VERSION NUMBER
;ADDRESS OF WCS BASED FILE NAME
;BIT 5 COMMAND + STATUS WORD
;BIT 13 COMMAND + STATUS WORD

```

```

WCS_DE_IDAR = <^X61>
WCS_DE_DBUF = <^X61>
WCS_DE_MCT = <^X50>
WCS_DE_MM = <^X54>
WCS_DE_TB = <^X52>
WCS_DE_UBS = <^X58>
WCS_ERR_NUM = <^X41>
WCS_EX_DBUF = <^X5C>
WCS_EX_ICSR = <^X5A>
WCS_EX_IDAR = <^X5B>
WCS_EX_MCT = <^X51>
WCS_EX_MM = <^X55>
WCS_EX_PATT = <^X5D>
WCS_EX_POSIT = <^X5E>
WCS_EX_TB = <^X53>
WCS_EX_UBS = <^X59>
WCS_EXP_DAT = <^X42>
WCS_LOAD_ADD = <^X0>
WCS_MOD_DAT = <^X46>
WCS_MSK_DAT = <^X45>
WCS_REC_DAT = <^X43>
WCS_REST_WR = <^X57>
WCS_SAVE_WR = <^X56>
WCS_SEL_FIFOA = <^X63>
WCS_SEL_FIFOB = <^X64>
WCS_VER_ADD = <^X606>
WCS_FILE_NAME = <^X607>
WRONG_LAB = <^X20>
XFER = <^X20>

```

```

:*****
: THE FOLLOWING SET OF EQUATES DEFINE THE BITS USED IN
: THE COMMAND AND STATUS WORD
:*****

```

```

;FLAG EQUATES
LOOP_S_DEF = <^X1> ;MASK USED TO SET LOOP ON ERROR
LOOP_COM_S_DEF = <^X40> ;MASK USED TO SET LOOP COMAND BIT
SPECIAL_S_DEF = <^X80> ;MASK USED TO SET SPECIAL BIT (USED
; TO LOOP ON SIZING TESTS)
NER_S_DEF = <^X2> ;MASK USED TO SET NO ERROR REPORTS
BELL_S_DEF = <^X4> ;MASK USED TO SET JELL ON ERROR
HALT_S_DEF = <^X8> ;MASK USED TO SET HALT ON ERROR
SER_S_DEF = <^X10> ;MASK USED TO SET SINGLE ERR. REP.
APT_S_DEF = <^X20> ;MASK USED TO SET APT PRESENT
; INDICATOR (NOT SETABLE BY SET
; COMMAND)
LOOP_C_DEF = <^XBE> ;MASK USED TO CLEAR LOOP ON ERROR
; AND LOOP COMMAND BITS
LOOP_COM_C_DEF = <^XBF> ;MASK USED TO CLEAR LOOP COMAND BIT
SPECIAL_C_DEF = <^X7F> ;MASK USED TO CLEAR SPECIAL BIT (USED
; TO LOOP ON SIZING TESTS)
NER_C_DEF = <^XBD> ;MASK USED TO CLEAR NO ERROR REPORT
; AND LOOP COMMAND BITS
BELL_C_DEF = <^XFB> ;MASK USED TO CLEAR BELL ON ERROR

```

```

ZZ-ENKAA-2.1 4C00 4
4C03 480 00000F7
4C03 481 00000EF
4C03 482 00000DF
4C03 483
4C03 484
4C03 485
4C03 486
4C03 487
4C03 488
4C03 489
4C03 490
4C03 491
4C03 492
4C03 493 00004106
4C03 494 00004107
4C03 495 0000410D
4C03 496 00004111
4C03 497 00004115
4C03 498 00004116
4C03 499 00004150
4C03 500
4C03 501
4C03 502
4C03 503
4C03 504
4C03 505
4C03 506
4C03 507
4C03 508
4C03 509
4C03 510
4C03 511
4C03 512
4C03 513
4C03 514
4C03 515
4C03 516
4C03 517
4C03 518
4C03 519
4C03 520
4C03 521
4C03 522
4C03 523
4C03 524
4C03 525
4C03 526
4C03 527
4C03 528
4C03 529
4C03 530
4C03 531
4C03 532
4C03 533
4C03 534
4C03 535
4C03 536
4C03 537
4C03 538
4C03 539

```

1 3

Fiche 1 Frame 13 Sequence 34

```

HALT_C_DEF = <^XF7> :MASK USED TO CLEAR HALT ON ERROR
SER_C_DEF = <^XEF> :MASK USED TO CLEAR SINGLE ERR. REP.
APT_C_DEF = <^XDF> :MASK USED TO CLEAR APT PRESENT
: INDICATOR (NOT CLEARABLE BY CLEAR
: COMMAND)

```

PARAMETER BLOCK FOR TUSB DRIVER

```

:*****
: TUSB DRIVER PARAMETER BLOCK
SEC_PARB_UNIT = <^X4106> :UNIT NUMBER
SEC_PARB_NAM = <^X4107> :FILENAME
SEC_PARB_EXT = <^X410D> :DOT PLUS FILE EXTENTION
SEC_PARB_ADD = <^X4111> :LOAD ADDRESS
SEC_PARB_DST = <^X4115> :DESTINATION 6=RAM 7=WCS
SEC_PARB_ERR = <^X4116> :ERROR RETURN BYTE
SEC_PARB_DEF = <^X4150> :DEFAULT DRIVE (WHERE MICMON WAS LOADED
: FROM)

```

EXTERNAL REFERENCES  
ALL EXTERNAL REFERENCES MUST BE LISTED HERE

```

.GLOBAL COMPARE,EOS_FLG,FLAGS,HEX_BJF
.GLOBAL MOVER,PRINT,PRINT_HEX,SPACE
.GLOBAL MWCS_A,MCPU_A,CR,[F,HEX_3,CC_ADDR,CRLF
.GLOBAL SHIFT_PNT,F_WORD,SHIFTER,DOLOOP
.GLOBAL NOP_CSR,READ_WCS,WCS_ADDRESS,WCS_VALUE
.GLOBAL WRITE_WCS,X_SET_PAGE,PERFORM_CSR
.GLOBAL X_CLR_PAGE,WCS_VAL_SHIFT
.GLOBAL CSR_VALUE,LOAD_CSR,UPC_VALUE,LOAD_UPC
.GLOBAL PARITY_JUMP,PARITY_ON,X_MOV_WRRW
.GLOBAL SHIFT_C_2901,SHIFT_R_2901,DATA0
.GLOBAL READ_DATA_32,WRITE_DATA_32,X_CLR_WRO
.GLOBAL DATAT,DATA2,DATA3,DATA_SHIFT
.GLOBAL READ_LS,WRITE_LS,INC_WORD,OS_ADD
.GLOBAL NOP_INST,CSR_SHIFT,X_COM_WRO
.GLOBAL MAKE_XD,XD_ADDRS,X_ROT_L,X_SHIFT_L
.GLOBAL WCS_BAD_PARITY,POWER_BRANCH,WCS_SHIFT

.GLOBAL PRINT_STRING,MOVER_4,PRINT_HEX_1,LD_UPC
.GLOBAL CON_ADD_DAT,CON_ERR_NUM,CON_EXP_DAT
.GLOBAL CON_EXP_ADD,CON_MOD_DAT,CON_MSK_DAT
.GLOBAL CON_REC_DAT,CON_REC_ADD,ERROR_CON,MOVER_3
.GLOBAL NA_EXP_REC,NA_OTHER,SKIP_TEST,MSK_ERROR
.GLOBAL CON_ERROR,CON_BEGIN_TEST,GCVECT,WCS_READ
.GLOBAL WCS_WRITE,CONTINUE,CNTLC_TYPED,SET_FOR_CONI
.GLOBAL CLEAR_CPU_ATT,SHIFTER1,X_SHIFT_R,DECR_WORD
.GLOBAL MICRO_STEP_CPU,WRITE_TAB_PNT
.GLOBAL APT,APT_PASS_CNT

```

APT INFORMATION.... THE ADDRESS OF THESE VARIABLES MUST NOT CHANGE

```

ZZ-ENKAA-2.1 4C00 4
4C03 540
4C03 541
4C03 542
4C03 543
4C03 544 06E4'C3
4C06 545
4C06 546 06D6'C3
4C09 547
4C09 548 7000
4C0B 549 00000010
4C10 550
4C10 551 0000
4C12 552 0000
4C14 553 0000
4C16 554 0000
4C18 555 0000
4C1A 556 0000001C
4C1C 557 00
4C1D 558 00
4C1E 559 00
4C1F 560 00000022
4C22 561
4C22 562
4C22 563
4C22 564
4C22 565
4C22 566
4C22 567
4C22 568
4C22 569
4C22 570
4C22 571
4C22 572 1BEE'C3
4C25 573 1F55'C3
4C28 574 1322'C3
4C2B 575 1333'C3
4C2E 576 01
4C2F 577 1F6C'C3
4C32 578 00000038
4C38 579
4C38 580
4C38 581
4C38 582
4C38 583
4C38 584
4C38 585
4C38 586
4C38 587
4C38 588 00
4C39 589 00
4C3A 590 00
4C3B 591 0000003F
4C3F 592 01
4C40 593 00000044
4C44 594 00000046
4C46 595 0000004B
4C4B 596 0000004F
4C4F 597 00000053
4C53 598 00000055
4C55 599 00

```

: IN OFFSET FROM THE BEGINNING OF THIS PROGRAM.

```

:*****
OVERLAY_RAM:: JMP APT_CON_TEST ;STARTING ADDRESS FOR RAM TESTS
: STARTED BY APT
OVERLAY_WCS:: JMP APT_WCS_TEST ;STARTING ADDRESS FOR WCS TESTS
: STARTED BY APT
APT_RAM_LD_ADD: .WORD EXEC_CONTESTS ;POINTER TO RAM OVERLAY LOAD ADDRESS
               .BLKB 5
APT_MAIL_BOX:
APT_MESSAGE_CODE: .WORD 0 ;FLAG 1=TEST START 2=ERROR
APT_ERROR_NUMBER: .WORD 0 ;DIAGNOSTIC ERROR NUMBER
APT_SUBTEST: .WORD 0 ;SUBTEST NUMBER FILLER
APT_TEST_NUMBER: .WORD 0 ;FAILING TEST NUMBER
APT_PASS_CNT: .WORD 0 ;# PASSES EXECUTED
               .BLKB 2 ;RESERVED FILLER
APT_MEM_SIZE: .BYTE 0 ;APT MEMORY SIZE
APT_HARD_FLG: .BYTE 0 ;APT HARDWARE INFORMATION
APT_SOFT_FLG: .BYTE 0 ;APT SOFTWARE INFORMATION
               .BLKB 3 ;RESERVED FILLER

```

```

:*****
: RETURN VECTORS - THE FOLLOWING FOUR INSTRUCTIONS ARE USED AS THE RETURN
: POINT FOR SOME BOOT BLOCK ROUTINES. THE NEXT BYTE IS A FLAG READ BY THE
: BOOT BLOCK TO DETERMINE IF MICROM IS LOADED. THE NEXT TWO INSTRUCTIONS
: ARE ALSO USED AS VECTORS FROM THE BOOT BLOCK. THEY MUST NOT CHANGE IN
: OFFSET FROM THE BEGINNING OF THIS PROGRAM.
:*****

```

```

MICFLAG: JMP PARITY_ERR ;RETURN FROM PARITY ERROR
          JMP POWER_REC ;RETURN FROM POWER FAIL RECOVER
          JMP CNTLC_VEC ;RETURN FROM CNTL C TYPED
          JMP CNTLP_VEC ;RETURN FROM CNTL P TYPED
          .BYTE 1 ;FLAG SO BOOT BLOCK KNOWS MIC RESIDENT
          JMP TIMER_VEC ;RETURN FROM INTERVAL TIMER INTERRUPT
          .BLKB 6 ;RESERVED FILLER FOR NEW RETURN VECTORS

```

```

:*****
: THE FOLLOWING DECLARATIONS ARE FOR GLOBAL VARIABLES.
: DO NOT DISPLACE FROM THEIR CURRENT LOCATIONS.
: ADD NEW VARIABLES AFTER ALL GLOBAL DECLARATIONS.
:*****

```

```

APT: .BYTE 0 ;FLAG 1=APT PRESENT
CNTLC_TYPED: .BYTE 0 ;FLAG 1=CONTROL C WAS TYPED
CONTINUE: .BYTE 0 ;FLAG 1=CONTINUE MODE ENABLED
CON_ADD_DAT: .BLKB 4 ;OTHER DATA
CON_ERR_NUM: .BYTE 1 ;CONSOLE BASED ERROR NUMBER
CON_EXP_DAT: .BLKB 4 ;EXPECTED DATA
CON_EXP_ADD: .BLKB 2 ;EXP DAT ADDR USED BY MSK_ERR
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

```

```

ZZ-ENKAA-2.1      4C00      4
4C56 600 00
4C57 601 00
4C58 602 FFFF
4C5A 603 00
4C5B 604 0000005F
4C5F 605 00
4C60 606 00
4C61 607 1BEE'C3
4C64 608 01
4C65 609 1F55'C3
4C68 610 0C70
4C6A 611 00
4C6B 612 02
4C6C 613 0000
4C6E 614 00
4C6F 615 0000
4C71 616 00
4C72 617
4C72 618
4C72 619
4C72 620 DB
4C73 621 00
4C74 622 15
4C75 623 01
4C76 624 23
4C77 625 D0
4C78 626 15
4C79 627
4C79 628 01
4C7A 629 A3
4C7B 630 C0
4C7C 631 15
4C7D 632
4C7D 633 01
4C7E 634 90
4C7F 635 A0
4C80 636 15
4C81 637
4C81 638 01
4C82 639 10
4C83 640 B0
4C84 641 15
4C85 642
4C85 643
4C85 644
4C85 645
4C85 646
4C85 647
4C85 648
4C85 649 08
4C86 650 23
4C87 651 40
4C88 652 15
4C89 653
4C89 654 23
4C8A 655 40
4C8B 656 15
4C8C 657
4C8C 658 23
4C8D 659 40

```

```

K 3
EOS_FLG: .BYTE 0
ERROR_CON: .BYTE 0
F_WORD: .WORD <^XFFFF>
FLAGS: .BYTE 0
HEX_BUF: .BLKB 4
NA_EXP_REC: .BYTE 0
NA_OTHER: .BYTE 0
PARITY_JUM: JMP PARITY_ERR
PARITY_ON: .BYTE 1
POWER_BRANCH: JMP POWER_REC
SHIFT_PNT: .WORD 0
SKIP_TEST: .BYTE 0
WCS_SHIFT: .BYTE 2
WCS_ADDRESS: .WORD 0
WCS_BAD_PARITY: .BYTE 0
WRITE_TAB_PNT: .WORD 0
XD_ADDRS: .BYTE 0

;GLOBAL CSR EXECUTABLE CODE
NOP_INST: .BYTE <^XDB>
           .BYTE <^X00>
           .BYTE <^X15>
X_SHIFT_L: .BYTE 1
           .BYTE <^X23>
           .BYTE <^XD0>
           .BYTE <^X15>
X_ROT_L: .BYTE 1
          .BYTE <^XA3>
          .BYTE <^XC0>
          .BYTE <^X15>
X_CLR_PAGE: .BYTE 1
            .BYTE <^X90>
            .BYTE <^XA0>
            .BYTE <^X15>
X_SET_PAGE: .BYTE 1
            .BYTE <^X10>
            .BYTE <^XB0>
            .BYTE <^X15>

; This subroutine shifts the contents of WR[0] right 8 bits and leaves the
; result on the Y-bus. The monitor can clock the console read register to
; read the result. To be executed 3 times by the monitor for a 32 bit WR
; read. ROR will have to have been checked by the console.
X_SHIFT_R: .BYTE 8
           .BYTE <^X23>
           .BYTE <^X40>
           .BYTE <^X15>
           .BYTE <^X23>
           .BYTE <^X40>
           .BYTE <^X15>
           .BYTE <^X23>
           .BYTE <^X40>

```

Fiche 1 Frame K3 Sequence 36

```

;FLAG 1=END OF SECTION
;FLAG 1=ERROR EXISTS
;HALT, LOOP, NER, BELL, SER
;HEX PRINT OUT BUFFER
;FLAG 1=N/A FOR EXP AND REC DA
;FLAG 1=N/A FOR OTHER DATA
;FLAG 1=CALCULATE PARITY
;SHIFT ROUTINE DATA POINTER
;FLAG 1=SKIP TEST # IS TO LOW
;TWO BYTES OF SHIFT DATA FOLLOW
;WCS ADDRESS TO READ OR WRITE
;FLAG 1=WRITE BAD PARITY
;POINTER TO UPC ADDRESS TABLE
;XD ADDRS FIELD OF CPU INST

```

```

;NOP INST FOR CPU CSR
;CLR WR[1]
;ROL WR[0]
;SHIFT LEFT 1 PLACE
;MISC [CLR.HIGH.PAGE]
;MISC [SET.HIGH.PAGE]

```

```

;ROR
;ROR
;ROR

```

```

ZZ-ENKAA-2.1      4C00  4
4C8E 660 15
4C8F 661
4C8F 662 23
4C90 663 40
4C91 664 15
4C92 665
4C92 666 23
4C93 667 40
4C94 668 15
4C95 669
4C95 670 23
4C96 671 40
4C97 672 15
4C98 673
4C98 674 23
4C99 675 40
4C9A 676 15
4C9B 677
4C9B 678 23
4C9C 679 40
4C9D 680 15
4C9E 681
4C9E 682 A0
4C9F 683 00
4CA0 684 15
4CA1 685
4CA1 686
4CA1 687
4CA1 688 02
4CA2 689 2F
4CA3 690 80
4CA4 691 15
4CA5 692
4CA5 693 A0
4CA6 694 C0
4CA7 695 15
4CA8 696
4CA8 697
4CA8 698
4CA8 699
4CA8 700
4CA8 701
4CA8 702
4CA8 703 04
4CA9 704 00
4CAA 705 00
4CAB 706 00
4CAC 707 00
4CAD 708 000000AF
4CAF 709
4CAF 710 00
4CB0 711 00
4CB1 712 00
4CB2 713 00
4CB3 714
4CB3 715 03
4CB4 716 000000B7
4CB7 717
4CB7 718
4CB7 719

```

```

L 3
Fiche 1 Frame L3 Sequence 37
.BYTE <^X15>
.BYTE <^X23> ;ROR
.BYTE <^X40>
.BYTE <^X15>
.BYTE <^XA0> ;MOV WR[0] TO WR[0]
.BYTE <^X00> ;MOVE DATA TO Y-BUS
.BYTE <^X15> ;TO BE READ
X_MOV_WRWR:
.BYTE <^XA0>
.BYTE <^X00>
.BYTE <^X15>
X_CLR_WRO:
.BYTE 2
.BYTE <^X2F> ;CLR WR[0]
.BYTE <^X80>
.BYTE <^X15>
X_COM_WRO:
.BYTE <^XA0> ;COM WR[0]
.BYTE <^XC0>
.BYTE <^X15>
:.....
: DATA AREA FOR READS AND WRITES TO CPU (32 BITS)
:.....
DATA_SHIFT: .BYTE 4 ;FOUR BYTE OF SHIFT DATA
DATA0: .BYTE 0 ;MSB OF 32 BIT DATA AREA
DATA1: .BYTE 0
DATA2: .BYTE 0
DATA3: .BYTE 0 ;LSB OF 32 BIT DATA AREA
.BLK 2 ;EXTENDED DATA FOR 3 BYTE MOVES
COMMAND0: .BYTE 0 ;MSB OF 32 BIT COMMAND+STATUS
COMMAND1: .BYTE 0 ;WORD
COMMAND2: .BYTE 0
COMMAND3: .BYTE 0 ;LSB OF 32 BIT COMMAND+STATUS
WCS_VAL_SHIFT: .BYTE 3 ;SHIFT 3 BYTES OF DATA
WCS_VALOE: .BLK 3 ;WCS WORD TO BE WRITTEN
:.....

```

```

ZZ-ENKAA-2.1 4C00 4
4CB7 720
4CB7 721
4CB7 722
4CB7 723
4CB7 724
4CB7 725 03
4CBB 726 000000BB
4CBB 727 000000BE
4CBE 728 000000C1
4CC1 729
4CC1 730
4CC1 731
4CC1 732
4CC1 733
4CC1 734
4CC1 735
4CC1 736 02
4CC2 737 000000C4
4CC4 738 000000C6
4CC6 739 000000C8
4CCB 740 000000CA
4CCA 741 000000CC
4CCC 742 000000CF
4CCF 743
4CCF 744
4CCF 745 1184'C3
4CD2 746 11A4'C3
4CD5 747 1265'C3
4CDB 748 11C4'C3
4CDB 749 1260'C3
4CDE 750 125A'C3
4CE1 751 10E7'C3
4CE4 752 10F9'C3
4CE7 753 109D'C3
4CEA 754 10B9'C3
4CED 755 15C2'C3
4CF0 756 1116'C3
4CF3 757 1148'C3
4CF6 758 1515'C3
4CF9 759 151B'C3
4CFC 760 1481'C3
4CFF 761 149E'C3
4D02 762 1435'C3
4D05 763 1415'C3
4D08 764 1581'C3
4D0B 765 1450'C3
4D0E 766 124A'C3
4D11 767 11A2'C3
4D14 768 11C2'C3
4D17 769 22D7'C3
4D1A 770 118E'C3
4D1D 771 223C'C3
4D20 772 221F'C3
4D23 773 21CD'C3
4D26 774 228B'C3
4D29 775 22C1'C3
4D2C 776 1349'C3
4D2F 777 1D43'C3
4D32 778 10F0'C3
4D35 779 158C'C3

```

```

: CSR DATA BUFFER
:.....
CSR_SHIFT: .BYTE 3 ;3 BYTES OF DATA
CSR_VALUE: .BLKB 3
SAVED_CSR: .BLKB 3
CONT_SAVE_CSH: .BLKB 3
:.....
: UPC DATA BUFFER
:.....
UPC_SHIFT: .BYTE 2 ;2 BYTES OF DATA
UPC_VALUE: .BLKB 2
SAVED_UPC: .BLKB 2
UPC_SOB: .BLKB 2
CONT_SAVE_UPC: .BLKB 2
UPC_COMPARE: .BLKB 2
ATT_UPC_SAVE: .BLKB 3
: JUMPS TO GLOBAL SUBROUTINES
COMPARE: JMP COMPARE_R
MOVER: JMP MOVER_R
PRINT: JMP PRINT_R
PRINT_HEX: JMP PRINT_HEX_R
SPACE: JMP SPACE_R
CRLF: JMP CRLF_R
SHIFTER: JMP SHIFTER_R
NOP_CSR: JMP NOP_CSR_R
READ_WCS: JMP READ_WCS_R
WRITE_WCS: JMP WRITE_WCS_R
PERFORM_CSR: JMP PERFORM_CSR_R
LOAD_CSR: JMP LOAD_CSR_R
LOAD_UPC: JMP LOAD_UPC_R
SHIFT_L_2901: JMP SHIFT_L_2901_R
SHIFT_R_2901: JMP SHIFT_R_2901_R
READ_DATA_32: JMP READ_DATA_32_R
WRITE_DATA_32: JMP WRITE_DATA_32_R
READ_CS: JMP READ_CS_R
WRITE_LS: JMP WRITE_LS_R
INC_WORD: JMP INC_WORD_R
MAKE_XD: JMP MAKE_XD_R
PRINT_STRING: JMP PRINT_STRING_R
MOVER_4: JMP MOVER_4_R
PRINT_HEX_1: JMP PRINT_HEX_1_R
LD_UPC: JMP LD_UPC_R
MOVER_3: JMP MOVER_3_R
MSK_ERROR: JMP MSK_ERROR_R
CON_ERROR: JMP CON_ERROR_R
CON_BEGIN_TEST: JMP CON_BEGIN_TEST_R
W'S_READ: JMP WCS_READ_R
WCS_WRITE: JMP WCS_WRITE_R
SET_FOR_CONT: JMP SET_FOR_CONT_R
CLEAR_CPU_ATT: JMP CLEAR_CPU_ATT_R
SHIFTER1: JMP SHIFTER1_R
DECR_WORD: JMP DECR_WORD_R

```

```

ZZ-ENKAA-2.1 4C00 4
4D38 780 1570'C3
4D38 781
4D38 782
4D38 783
4D38 784
4D38 785
4D38 786
4D38 787
4D38 788
4D38 789 34 39 33 38 4D
4D40 790 30 39 33 38 4D
4D45 791 31 39 33 38 4D
4D4A 792 39 38 33 38 4D
4D4F 793 38 32 37 38 4D
4D54 794 38 38 33 38 4D
4D59 795 20 32 30 4C 52
4D5E 796 20 20 30 38 52
4D63 797
4D63 798
4D63 799
4D63 800
4D63 801
4D63 802
4D63 803
4D63 804 00
4D64 805
4D64 806 000001B4
4D84 807
4D84 808
4D84 809
4D84 810
4D84 811
4D84 812
4D84 813
4D84 814
4D84 815
4D84 816
4D84 817
4D84 818 00
4D85 819 00
4D86 820 45 54 20 49 44
      0D 4F 43 20 31 20
4DC1 821 00
4DC2 822 09
4DC3 823 52 52 41 20 20
      23 20 59 41
4DCC 824 00
4DCD 825 20
4DCE 826 000001D4
4DD4 827 000001DA
4DDA 828 00
4DDB 829 0000
4DDD 830 20
4DDE 831 41 45 52 42 0D
      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

```

N 3

Fiche 1 Frame N3  
MICRO\_STEP\_CPU: JMP MICRO\_STEP\_CPU\_R

Sequence 39

```

.....
MODULE LITERALS
.....

```

```

PRT MOD_A:
MWCS_A: .ASCII 'M8394' :WCS
MCPA_A: .ASCII 'M8390' :CPU
MMCT_A: .ASCII 'M8391' :MCT
MFPA_A: .ASCII 'M8389' :FPA
MMEM_A: .ASCII 'M8728' :MEMORY ARRAY CARD
MIDC_A: .ASCII 'M8388' :IDC
MRLO2_A: .ASCII 'RLO2' :RLO2
MR80_A: .ASCII 'R80' :R80

```

```

.....
INPUT BUFFER AREA
.....

```

```

TTBUF_CNT: .BYTE 0 :CHAR CNT OF INPUT LINE PUT
           : HERE BY GET LINE ROUTINE
TTBUF: .BLKB 80 :TTY BUFFER

```

```

.....
ALL PRECEDING DECLARATIONS ARE FOR GLOBAL VARIABLES.
DO NOT DISPLACE ANY OF THESE FROM THEIR CURRENT LOCATIONS.
ALL NEW VARIABLES MUST BE ADDED BELOW THIS POINT.
.....

```

```

:VARIABLES
A_ADDR: .BYTE 0 :A ADDRS FIELD OF CPU INST
ACK_SAVE: .BYTE 0 :SAVE CPU ACK FOR SUBROUTINES
APT_DI_COMMAND: .ASCII 'DI TE 1 CO'<CR> :ASCII COMMAND STRING TO START

ARRAY_NUM_A: .BYTE 0 :WCS TESTS UNDER APT
             .BYTE 9 :CHAR COUNT
             .ASCII ' ARRAY #'

B_ADDR: .BYTE 0 :B ADDRS FIELD OF CPU INST
BLANK_A: .ASCII ' ' :ONE ASCII SPACE
BOARD_NAME: .BLKB 6 :BOARD NAME AREA
BOARD_BUF: .BLKB 6 :BOARD NAME AREA
BOARD_NUM: .BYTE 0 :NUMBER OF BOARD INPUT
BOARD_PNT: .WORD 0 :POINTER TO BOARD TABLE
BREAK_A: .BYTE 32 :CHAR COUNT
         .ASCII <CR>'BREAK'<CR>' A C BC DE HL SP'<CR>

```







```

ZZ-ENKAA-2.1      4C00      4
4FDC  974 00
4FDD  975 0000
4FDF  976 00
4FE0  977 0000
4FE2  978 04
4FE3  979 45 42 55 0D
4FE7  980 20 20 53 42 55
4FEC  981 05
4FED  982 20 43 50 55 0D
4FF2  983 02
4FF3  984 56 2D
4FF5  985 0B
4FF6  986 20 52 45 56 0D
      0D 30 31 2E 32 30
5001  987 00
5002  988 04
5003  989 20 53 43 57
5007  990 00
5008  991 00
5009  992 00
500A  993 04
500B  994 20 4B 52 57
500F  995 00000413
5013  996 0000
5015  997 00000418
5018  998 00
5019  999 0000
501B 1000 0000041F
501F 1001
501F 1002
501F 1003
501F 1004
501F 1005
501F 1006
501F 1007 01
5020 1008 36
5021 1009 00
5022 1010 15
5023 1011
5023 1012 36
5024 1013 00
5025 1014 15
5026 1015
5026 1016
5026 1017 01
5027 1018 90
5028 1019 C0
5029 1020 15
502A 1021
502A 1022
502A 1023 01
502B 1024 A0
502C 1025 00
502D 1026 15
502E 1027
502E 1028 A0
502F 1029 00
5030 1030 15
5031 1031
5031 1032

```

E 4

Fiche 1 Frame E4 Sequence 43

```

TEST_ZERO:      .BYTE      0      ;FLAG 1=EXECUTING TEST 0
TIMOUT_CNT:    .WORD      0      ;TIME OUT COUNTER
TRACE:         .BYTE      0      ;FLAG 1=TRACE MODE
TTBUF_PNT:    .WORD      0      ;POINTER TO TTY INPUT BUFFER
UBE_A:         .BYTE      4      ;CHAR COUNT
               .ASCII    <CR>'UBE''
UBS_A:         .ASCII    'UBS
UPC_A:         .BYTE      5      ;CHAR COUNT
               .ASCII    <CR>'UPC ''
VER_A:         .BYTE      2      ;CHAR COUNT
               .ASCII    '-V''
VERSION:       .BYTE     11      ;CHAR COUNT
               .ASCII    <CR>'VER 02.10''<CR>

WARM_RESTART: .BYTE      0      ;FLAG 1=POWER FAIL RESTART
WCS_A:         .BYTE      4      ;CHAR COUNT
               .ASCII    'WCS ''
WCS_CALC:      .BYTE      0      ;FLAG 1=WCS PARITY CALCULATION
WCS_LOADED:    .BYTE      0      ;FLAG 1=WCS OVERLAY LOADED
WCS_PNT:       .BYTE      0      ;COUNT FOR WCS 3 BYTE LOADS
WRK_A:         .BYTE      4      ;CHAR COUNT
               .ASCII    'WRK ''

X_COM_ADDRS:   .BLKB      4
XFER_CNT:      .WORD      0      ;LONG WORD XFER COUNTER
XFER_DEST:     .BLKB      3      ;ADDR DATA XFER TO
XFER_MM:       .BYTE      0      ;FLAG FOR MM OR LS XFER
XFER_PNT:      .WORD      0
ZERO_WORD:     .BLKB      4

;CSR EXECUTABLE CODE
;NOTE: THE INSTRUCTIONS THAT ARE PAIRED WITH S XXXX ARE DYNAMICLY MODIFIED
;      DURING EXECUTION. THUS THE ADDRESS 0 WAS ASSEMBLED INTO THEM. THIS
;      ADDRESS MY BE MODIFIED

X_READ_LS:     .BYTE      1      ;MOV LS[0] TO WR[0]
               .BYTE    <^X36>
               .BYTE    <^X00>
               .BYTE    <^X15>

S_READ_LS:     .BYTE      1      ;MOV LS[0] TO WR[0]
               .BYTE    <^X36>
               .BYTE    <^X00>
               .BYTE    <^X15>

X_CLR_CPU_ATT: .BYTE      1      ;INSTRUCTION COUNT
               .BYTE    <^X90> ;MISC [CLR.CP.ATTN.AND.ACK]
               .BYTE    <^XC0>
               .BYTE    <^X15>

X_ED_WRK:      .BYTE      1      ;INSTRUCTION COUNT
               .BYTE    <^XA0> ;MOV WR[0] TO WR[0]
               .BYTE    <^X00>
               .BYTE    <^X15>

S_ED_WRK:      .BYTE      1      ;MOV WR[0] TO WR[0]
               .BYTE    <^XA0>
               .BYTE    <^X00>
               .BYTE    <^X15>

```

```

ZZ-ENKAA-2.1      4C00      4
5031 1033 01
5032 1034 BE
5033 1035 00
5034 1036 15
5035 1037
5035 1038 BE
5036 1039 00
5037 1040 15
5038 1041
5038 1042
5038 1043
5038 1044
5038 1045
5038 1046
5038 1047
5038 1048
5038 1049
5038 1050 03
5039 1051 0000043C
503C 1052
503C 1053
503C 1054
503C 1055
503C 1056
503C 1057
503C 1058
503C 1059 04
503D 1060 00000441
5041 1061
5041 1062
5041 1063
5041 1064
5041 1065
5041 1066
5041 1067
5041 1068 0C
5042 1069 0000044E
504E 1070
504E 1071 0C
504F 1072 0000045B
505B 1073
505B 1074
505B 1075
505B 1076
505B 1077
505B 1078
505B 1079
505B 1080
505B 1081 01
505C 1082 53 43 57
505F 1083 02
5060 1084 50 41 44
5063 1085 02
5064 1086 55 50 43
5067 1087 04
5068 1088 54 43 40
506B 1089 08
506C 1090 41 50 46
506F 1091 10
5070 1092 43 44 49

```

```

F 4
Fiche 1 Frame F4 Sequence 44
X_WRITE_LS: .BYTE 1 ;INSTRUCTION COUNT
               .BYTE <^XBE> ;MOV WR[0] TO LS[0]
               .BYTE <^X00>
               .BYTE <^X15>
S_WRITE_LS: .BYTE <^XBE> ;MOV WR[0] TO LS[0]
               .BYTE <^X00>
               .BYTE <^X15>
:*****
: TEMP CSR AREA FOR SHIFTING DATA
:*****
TEMP_SHIFT: .BYTE 3 ;3 BYTES OF DATA IN CSR
TEMP_CSR: .BLKB 3 ;TEMP CSR STORAGE
:*****
: INPUT HEX NUMBER AREA
:*****
NUM_SHIFT: .BYTE 4 ;4 BYTES OF HEX DIGITS
INPT_NUM: .BLKB 4
:*****
: EXPECTED AND RECEIVED RLO2 LABEL AREA
:*****
EXP_LABEL: .BYTE 12 ;LABEL IS 12 BYTES LONG
EXP_LAB_DAT: .BLKB 12 ;STORAGE FOR EACH BYTE
REC_LABEL: .BYTE 12 ;CHAR COUNT
REC_LAB_DAT: .BLKB 12 ;STORAGE FOR EACH BYTE
:*****
: BOARD NAME TABLE
:*****
BOARD_TABLE . .BYTE 1 ;BOARD NUMBER
               .ASCII 'WCS'
               .BYTE 2 ;BOARD NUMBER
               .ASCII 'DAP'
               .BYTE 2 ;BOARD NUMBER
               .ASCII 'CPU'
               .BYTE 4 ;BOARD NUMBER
               .ASCII 'MCT'
               .BYTE 8 ;BOARD NUMBER
               .ASCII 'FPA'
               .BYTE <^X10> ;BOARD NUMBER
               .ASCII 'IDC'

```



```

ZZ-ENKAA-2.1 4C00 4
SOC1 1153
SOC1 1154 08
SOC2 1155 45 43 4B 4E 45
SOC7 1156 00
SOC8 1157
SOC8 1158 10
SOC9 1159 46 43 4B 4E 45
SOCE 1160 00
SOCF 1161
SOCF 1162 50
SOD0 1163 47 43 4B 4E 45
SOD5 1164 00
SOD6 1165
SOD6 1166 10
SOD7 1167 48 43 4B 4E 45
SODC 1168 00
SODD 1169
SODD 1170 FF
SOE 1171
SOE 1172 00
SODF 1173 000004E5
SOE5 1174
SOE5 1175
SOE5 1176
SOE5 1177
SOE5 1178
SOE5 1179
SOE5 1180
SOE5 1181
SOE5 1182
SOE5 1183
SOE5 1184 0601
SOE7 1185 0602
SOE9 1186 0603
SOEB 1187 0604
SOED 1188 0606
SOEF 1189 0603
SOF1 1190 0605
SOF3 1191 0606
SOF5 1192
SOF5 1193
SOF5 1194
SOF5 1195
SOF5 1196
SOF5 1197
SOF5 1198 03
SOF6 1199 55 2F 58
SOF9 1200 0D66'
SOFB 1201
SOFB 1202 03
SOFc 1203 43 2F '8
SOFf 1204 0D6E'
S101 1205
S101 1206 03
S102 1207 52 49 44
S105 1208 1Df1'
S107 1209
S107 1210 02
S108 1211 49 44
S10A 1212 070C'

```

H 4

Fiche 1 Frame M4

Sequence 46

```

.BYTE <^X08> ;WCS BASED FPA TEST
.ASCII 'ENKCE'
.BYTE 0

.BYTE <^X10> ;WCS BASED IDC TEST
.ASCII 'ENKCF'
.BYTE 0

.BYTE <^X50> ;WCS BASED IDC TEST
.ASCII 'ENKCG' ; (OPTIONAL)
.BYTE 0

.BYTE <^X10> ;WCS BASED IDC TEST
.ASCII 'ENKCH' ; (OPTIONAL)
.BYTE 0

.BYTE <^XFF> ;END OF TABLE

SECTION_FLAG: .BYTE 0 ;SECTION TYPE AND BOARD NUM
SECTION_NAME: .BLKB SEC_LEN ;SAVE BYTES FOR NAME

```

\*\*\*\*\*

UPC VERIFY ADDRESS TABLE

THIS TABLE OF VALUES IS COMPARED TO ACTUAL UPC VALUES TO VERIFY  
THE WRITE OF DATA TO THE CPU IS WORKING CORRECTLY

\*\*\*\*\*

```

WRITE_TAB: .WORD <^X601>
           .WORD <^X602>
           .WORD <^X603>
           .WORD <^X604>
           .WORD <^X606>
           .WORD <^X603>
           .WORD <^X605>
           .WORD <^X606>

```

\*\*\*\*\*

CHAR STRINGS AND JUMP ADDRESSES FOR PARSING AT COMMAND LEVEL

\*\*\*\*\*

```

MAIN_TAB: .BYTE 3
           .ASCII 'X/U' ;X COMMAND
           .WORD XU

           .BYTE 3
           .ASCII 'X/C' ;X COMMAND
           .WORD XC

           .BYTE 3
           .ASCII 'DIR' ;DIR MUST COME B4 DI..'
           .WORD DIR_COMMAND ;DIRECTORY COMMAND

           .BYTE 2
           .ASCII 'DI' ;DI MUST COME B4 D..'
           .WORD DIAG_COMMAND ;DIAGNOSE COMMAND

```

```

ZZ-ENKAA-2.1 4000 4
S10C 1213
S10C 1214 02
S10D 1215 48 53
S10F 1216 08D5'
S111 1217
S111 1218 02
S112 1219 4C 43
S114 1220 0A5A'
S116 1221
S116 1222 01
S117 1223 43
S118 1224 0946'
S11A 1225
S11A 1226 02
S11B 1227 45 52
S11D 1228 0705'
S11F 1229
S11F 1230 01
S120 1231 52
S121 1232 0AD6'
S123 1233
S123 1234 02
S124 1235 45 53
S126 1236 097D'
S128 1237
S128 1238 02
S129 1239 4F 4C
S12B 1240 0AC2'
S12D 1241
S12D 1242 03
S12E 1243 45 2F 54
S131 1244 0705'
S133 1245
S133 1246 02
S134 1247 58 45
S136 1248 0AE1'
S138 1249
S138 1250 01
S139 1251 45
S13A 1252 0AE1'
S13C 1253
S13C 1254 02
S13D 1255 45 44
S13F 1256 0C57'
S141 1257
S141 1258 01
S142 1259 44
S143 1260 0C57'
S145 1261
S145 1262 03
S146 1263 55 2F 53
S149 1264 0D4C'
S14B 1265
S14B 1266 02
S14C 1267 44 4C
S14E 1268 0939'
S150 1269
S150 1270 01
S151 1271 49
S152 1272 1E8A'

```

```

1 4
Fiche 1 Frame 14 Sequence 47

.BYTE 2
.ASCII 'SH'
.WORD SHOW_COMMAND ;SHOW COMMAND

.BYTE 2
.ASCII 'CL'
.WORD CLEAR_COMMAND ;CLEAR COMMAND

.BYTE 1
.ASCII 'C'
.WORD CONT_COMMAND ;CONTINUE COMMAND

.BYTE 2
.ASCII 'RE'
.WORD STNDRD_CON_COM ;RE MUST COME BEFORE R!!!!
;RETURN TO STANDARD CONSOLE

.BYTE 1
.ASCII 'R'
.WORD REPEAT_COMMAND ;REPEAT COMMAND

.BYTE 2
.ASCII 'SE'
.WORD SET_COMMAND ;SET COMMAND

.BYTE 2
.ASCII 'LO'
.WORD LOOP_COMMAND ;LOOP COMMAND

.BYTE 3
.ASCII 'T/E'
.WORD STNDRD_CON_COM ;RETURN TO STANDARD CONSOLE

.BYTE 2
.ASCII 'EX'
.WORD EXAM_COMMAND ;EX MUST COME B4 E!!!!
;EXAMINE COMMAND

.BYTE 1
.ASCII 'E'
.WORD EXAM_COMMAND ;EXAMINE COMPATABLE WITH STANDARD

.BYTE 2
.ASCII 'DE'
.WORD DEP_COMMAND ;DE MUST COM B4 D!!!!
;DEPOSIT COMMAND

.BYTE 1
.ASCII 'D'
.WORD DEP_COMMAND ;DEPOSIT COMPATABLE WITH STANDARD

.BYTE 3
.ASCII 'S/U'
.WORD START ;START AT 80B5 ADDRESS

.BYTE 2
.ASCII 'LD'
.WORD LOADER ;LOAD SECTION (NO EXECUTE)

.BYTE 1
.ASCII 'I'
.WORD EXEC_TESTO ;INIT (EXEC WCS TEST 0)

```

```

ZZ-ENKAA-2.1 4C00 4
5154 1273
5154 1274 02
5155 1275 42 41
5157 1276 1F6D'
5159 1277
5159 1278 01
515A 1279 0D
515B 1280 055E'
515D 1281
515D 1282 FF
515E 1283
515E 1284 C9
515F 1285
515F 1286
515F 1287
515F 1288
515F 1289
515F 1290
515F 1291
515F 1292 02
5160 1293 4F 42
5162 1294 079D'
5164 1295
5164 1296 02
5165 1297 45 53
5167 1298 07EB'
5169 1299
5169 1300 02
516A 1301 45 54
516C 1302 0858'
516E 1303
516E 1304 02
516F 1305 4F 43
5171 1306 089C'
5173 1307
5173 1308 02
5174 1309 41 50
5176 1310 08A2'
5178 1311
5178 1312 02
5179 1313 48 53
517B 1314 08C7'
517D 1315
517D 1316 01
517E 1317 00
517F 1318 1EF4'
5181 1319
5181 1320 FF
5182 1321
5182 1322
5182 1323
5182 1324
5182 1325
5182 1326
5182 1327
5182 1328 02
5183 1329 41 48
5185 1330 0A61'
5187 1331
5187 1332 02

```

J 4

Fiche 1 Frame J4

Sequence 48

```

.BYTE 2
.ASCII 'AB' ;ABORT (FUTURE EXPANSION)
.WORD START_UP

.BYTE 1
.ASCII <CR> ;END ON LINE HAS <CR>
.WORD RETURN

.BYTE <^XFF> ;END OF TABLE

RETURN: RET ;RETURN INSTRUCTION FOR <CR> CASE

```

```

.....
: CHAR STRINGS AND JUMP ADDRESSES FOR PARSING THE DIAGNOSE COMMAND
:
.....

```

```

DIAG_TAB: .BYTE 2
.ASCII 'BO' ;DIAGNOSE BOARD
.WORD DIAG_BOARD

.BYTE 2
.ASCII 'SE' ;DIAGNOSE SECTION
.WORD DIAG_SECTION

.BYTE 2
.ASCII 'TE' ;DIAGNOSE TEST
.WORD DIAG_TEST

.BYTE 2
.ASCII 'CO' ;DIAGNOSE CONTINUE
.WORD DIAG_CONTIN

.BYTE 2
.ASCII 'PA' ;DIAGNOSE PASS
.WORD DIAG_PASS

.BYTE 2
.ASCII 'SH' ;DIAGNOSE SHORTEN
.WORD DIAG_SHORTEN

.BYTE 1
.BYTE 0 ;END OF LINE HAS 0 AFTER <CR>
.WORD SET_PARDONE

.BYTE <^XFF> ;END OF TABLE

```

```

.....
: CHAR STRINGS AND JUMP ADDRESSES FOR THE CLEAR COMMAND
:
.....

```

```

CLEAR_TAB: .BYTE 2
.ASCII 'HA' ;CLEAR HALT ON ERROR
.WORD CLEAR_HALT

.BYTE 2

```

```

ZZ-ENKAA-2.1 4C00 4
5188 1333 4F 4C
518A 1334 0A6B'
518C 1335
518C 1336 02
518D 1337 45 4E
518F 1338 0A74'
5191 1339
5191 1340 02
5192 1341 45 42
5194 1342 0A79'
5196 1343
5196 1344 02
5197 1345 45 53
5199 1346 0A7E'
519B 1347
519B 1348 02
519C 1349 50 53
519E 1350 0A83'
51A0 1351
51A0 1352
51A0 1353 02
51A1 1354 52 54
51A3 1355 0A95'
51A5 1356
51A5 1357 02
51A6 1358 4F 53
51A8 1359 0A88'
51AA 1360
51AA 1361 02
51AB 1362 45 44
51AD 1363 0A9A'
51AF 1364
51AF 1365 02
51B0 1366 41 50
51B2 1367 0AAF'
51B4 1368
51B4 1369 02
51B5 1370 49 46
51B7 1371 0A43'
51B9 1372
51B9 1373 02
51BA 1374 52 42
51BC 1375 0A28'
51BE 1376
51BE 1377 FF
51BF 1378
51BF 1379
51BF 1380
51BF 1381
51BF 1382
51BF 1383
51BF 1384
51BF 1385 02
51C0 1386 41 48
51C2 1387 0984'
51C4 1388
51C4 1389 02
51C5 1390 4F 4C
51C7 1391 098C'
51C9 1392

```

```

K 4
Fiche 1 Frame K4 Sequence 49
.ASCII 'LO'
.WORD CLEAR_LOOP ;CLEAR LOOP ON ERROR
.BYTE 2
.ASCII 'NE'
.WORD CLEAR_NER ;CLEAR NO ERROR REPORTS
.BYTE 2
.ASCII 'BE'
.WORD CLEAR_BELL ;CLEAR BELL ON ERROR
.BYTE 2
.ASCII 'SE'
.WORD CLEAR_SER ;CLEAR SINGLE BIT ERROR REPORTS
.BYTE 2
.ASCII 'SP'
.WORD CLEAR_SP ;CLEAR SPECIAL BIT (FOR LOOPING ON
; SIZING TESTS)
.BYTE 2
.ASCII 'TR'
.WORD CLEAR_TRACE ;CLEAR TRACE MODE
.BYTE 2
.ASCII 'SO'
.WORD CLEAR_SOMM ;CLEAR STOP ON MICRO MATCH
.BYTE 2
.ASCII 'DE'
.WORD CLEAR_DEFAULT ;CLEAR FLAGS TO DEFAULT VALUES
.BYTE 2
.ASCII 'PA'
.WORD CLEAR_PAR ;CLEAR PARITY COMMAND
.BYTE 2
.ASCII 'FI'
.WORD CLEAR_FIFO ;CLEAR IDC FIFO ADDRESS
.BYTE 2
.ASCII 'BR'
.WORD CLEAR_BREAK ;CLEAR IDC FIFO ADDRESS
.BYTE <^XFF> ;END OF TABLE

```

```

.....
:
: CHAR STRINGS AND JUMP ADDRESSES FOR THE SET COMMAND
:
:
:.....

```

```

SET_TAB: .BYTE 2
.ASCII 'HA'
.WORD SET_HALT ;SET HALT ON ERROR
.BYTE 2
.ASCII 'LO'
.WORD SET_LOOP ;SET LOOP ON ERROR

```

```

ZZ-ENKAA-2.1 4C00 4
51C9 1393 02
51CA 1394 45 4E
51CC 1395 0991'
51CE 1396
51CF 1397 02
51D1 1398 45 42
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'

```

L 4

Fiche 1 Frame L4

Sequence 50

```

.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"
.WORD SET_BREAK ;SET BREAK POINT

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

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

```

```

ZZ-ENKAA-2.1 4000 4
520B 1453
520B 1454 02
520C 1455 50 55
520E 1456 0B2C'
5210 1457
5210 1458 02
5211 1459 42 55
5213 1460 0B9B'
5215 1461
5215 1462 01
5216 1463 55
5217 1464 0AEF'
5219 1465
5219 1466 02
521A 1467 53 43
521C 1468 0B19'
521E 1469
521E 1470 01
521F 1471 43
5220 1472 0BCA'
5222 1473
5222 1474 02
5223 1475 43 4D
5225 1476 0B92'
5227 1477
5227 1478 02
5228 1479 52 57
522A 1480 0B4E'
522C 1481
522C 1482 02
522D 1483 53 4C
522F 1484 0B7C'
5231 1485
5231 1486 02
5232 1487 42 54
5234 1488 0BA4'
5236 1489
5236 1490 02
5237 1491 4D 4D
5239 1492 0BA0'
523B 1493
523B 1494 02
523C 1495 53 4F
523E 1496 0BB6'
5240 1497
5240 1498 02
5241 1499 43 57
5243 1500 0BCA'
5245 1501
5245 1502 02
5246 1503 43 49
5248 1504 0BE0'
524A 1505
524A 1506 02
524B 1507 44 49
524D 1508 0BE9'
524F 1509
524F 1510 02
5250 1511 42 44
5252 1512 0BF2'

```

M 4

Fiche 1 Frame M4

Sequence 51

```

.BYTE 2
.ASCII 'UP' ;UP MUST COME BEFORE U!!!!
.WORD EXAM_UPC ;EXAMINE UPC REGISTER

.BYTE 2
.ASCII 'UB' ;UB MUST COME BEFORE U!!!!
.WORD EXAM_UBS

.BYTE 1
.ASCII 'U' ;E/U FOR COMPATABILITY WITH STANDARD
.WORD EXAM_RAM

.BYTE 2
.ASCII 'CS' ;CS MUST COME BEFORE C
.WORD EXAM_CSR ;EXAMINE CSR REGISTER

.BYTE 1
.ASCII 'C' ;E/C FOR COMPATABILITY WITH STANDARD
.WORD EXAM_WCS

.BYTE 2
.ASCII 'MC' ;EXAMINE MEMORY CONTROL + STATUS
.WORD EXAM_MCT

.BYTE 2
.ASCII 'WR' ;EXAMINE 2901 WORKING REGS
.WORD EXAM_WRK

.BYTE 2
.ASCII 'LS' ;EXAMINE LS LOCATIONS
.WORD EXAM_LS

.BYTE 2
.ASCII 'TB' ;EXAMINE TRANSLATION BUFFER
.WORD EXAM_TB

.BYTE 2
.ASCII 'MM' ;EXAMINE MAIN MEMORY
.WORD EXAM_MM

.BYTE 2
.ASCII 'OS' ;EXAMINE OS REGISTER
.WORD EXAM_OS

.BYTE 2
.ASCII 'WC' ;EXAMINE WCS MEMORY
.WORD EXAM_WCS

.BYTE 2
.ASCII 'IC' ;EXAMINE IDC CSR
.WORD EXAM_ICSR

.BYTE 2
.ASCII 'ID' ;EXAMINE IDC DISK ADD REG
.WORD EXAM_IDAR

.BYTE 2
.ASCII 'DB' ;EXAMINE IDC DATA BUFFER
.WORD EXAM_DBUF

```

```

ZZ-ENKAA-2.1 4C00 4
5254 1513
5254 1514 02
5255 1515 41 50
5257 1516 0BFB'
5259 1517
5259 1518 02
525A 1519 4F 50
525C 1520 0C04'
525E 1521
525E 1522 FF
525F 1523
525F 1524
525F 1525
525F 1526
525F 1527
525F 1528
525F 1529
525F 1530
525F 1531 02
5260 1532 41 52
5262 1533 0C5D'
5264 1534
5264 1535 02
5265 1536 50 55
5267 1537 0C89'
5269 1538
5269 1539 02
526A 1540 42 55
526C 1541 0D13'
526E 1542
526E 1543 01
526F 1544 55
5270 1545 0C5D'
5272 1546
5272 1547 02
5273 1548 53 43
5275 1549 0C76'
5277 1550
5277 1551 01
5278 1552 43
5279 1553 0CC2'
527B 1554
527B 1555 02
527C 1556 43 4D
527E 1557 0D01'
5280 1558
5280 1559 02
5281 1560 52 57
5283 1561 0C95'
5285 1562
5285 1563 02
5286 1564 53 4C
5288 1565 0CEC'
528A 1566
528A 1567 02
528B 1568 42 54
528D 1569 0D0D'
528F 1570
528F 1571 02
5290 1572 4D 4D

```

N 4

Fiche 1 Frame N4

Sequence 52

```

.BYTE 2
.ASCII 'PA'
.WORD EXAM_PATT ;EXAMINE IDC ECC PATTERN

.BYTE 2
.ASCII 'PO'
.WORD EXAM_POSIT ;EXAMINE IDC ECC POSITION

.BYTE <^XFF> ;END OF TABLE

```

```

.....
: CHAR STRINGS AND JUMP ADDRESSES FOR THE DEPOSIT COMMAND
:.....

```

```

DEP_TAB:
.BYTE 2
.ASCII 'RA'
.WORD DEP_RAM ;DEPOSIT 8085 RAM

.BYTE 2
.ASCII 'UP'
.WORD DEP_UPC ;DEPOSIT UPC REGISTER

.BYTE 2
.ASCII 'UB'
.WORD DEP_UBS ;UB MUST COME BEFORE U!!!!

.BYTE 1
.ASCII 'U'
.WORD DEP_RAM ;D/U FOR COMPATABILITY WITH STANDARD

.BYTE 2
.ASCII 'CS'
.WORD DEP_CSR ;DEPOSIT CSR REGISTER

.BYTE 1
.ASCII 'C'
.WORD DEP_WCS ;D/C FOR COMPATABILITY WITH STANDARD

.BYTE 2
.ASCII 'MC'
.WORD DEP_MCT ;DEPOSIT MEMORY CONTROL + STATUS

.BYTE 2
.ASCII 'WR'
.WORD DEP_WRK ;DEPOSIT 2901 WORKING REGS

.BYTE 2
.ASCII 'LS'
.WORD DEP_LS ;DEPOSIT LS LOCATIONS

.BYTE 2
.ASCII 'TB'
.WORD DEP_TB ;DEPOSIT TRANSLATION BUFFER

.BYTE 2
.ASCII 'TM'

```









```

Z7-ENKAA-2.1 4000 4
53D3 1809 *****
53D3 1810 *****
53D3 1811 01'CE'21
          03 OE 01'D5'11
          07E8'C2 00CF'CD
53E1 1812 *****
53E1 1813 01D4'3A
53E4 1814 01DA'32
53E7 1815 C9
53E8 1816 *****
53E8 1817 *****
53E8 1818 *****
53E8 1819 07BA'C3
53EB 1820 *****
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
53EF 1834 *****
53EF 1835 06 OE 03'CD'21
          0D 23 77 AF
          07F5'C2
53FB 1836 *****
53FB 1837 03E0'2A
53FE 1838 03C6'22
5401 1839 1063'CD
5404 1840 03C6'2A
5407 1841 03'CD'11
540A 1842 037B'3A
540D 1843 4F
540E 1844 11A4'CD
5411 1845 *****
5411 1846 04'74'21
5414 1847 03B4'22
5417 1848 *****
5417 1849 03B4'2A
541A 1850 04'DE'11
541D 1851 07 OE
541F 1852 11A4'CD
5422 1853 03B4'22
5425 1854 *****
5425 1855 FF FE 04DE'3A
          0843'C2
542D 1856 *****
542D 1857 04DE'32 AF
5431 1858 03'CD'21
5434 1859 04'DF'11
5437 1860 06 OE
5439 1861 11A4'CD
543C 1862 *****
543C 1863 1D6C'CD

```

```

F 5
Fiche 1 Frame F5 Sequence 57
ENDIF
IFEQ BOARD_NAME,BOARD_BUF+1,3 ;COMPARE BOARD TO NAME GIVEN
LDA BOARD_BUF ;EQUAL SAVE BOARD #
STA BOARD_NUM
RET ;DONE
ENDIF
JMP 1$ ;LOOP UNTIL ERROR OR BOARD
; FOUND
*****
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.
*****
DIAG_SECTION: CLEAR TABLE_MODE ;SET FOR NO TABLE USAGE
ZERO TEMP_SECTION,SEC_LEN ;START WITH CLEAR FIELD
LHLD TTBUF_PNT
SHLD TEMP_WORD ;SAVE POINTER TO SECTION
CALL SKIP_TO_SPEC ;SKIP OVER IT TO GET COUNT
LHLD TEMP_WORD ;MOVE NAME TO SAVE AREA
LXI D,TEMP_SECTION
LDA SKIP_CRT
MOV C,A ;SECT NAME LENGTH
CALL MOVER_R
LXI H,SEC_TABLE
SHLD TABLE_PNT ;RESET TABLE POINTER
1$: LHLD TABLE_PNT ;MOVE ENTRY IN SEC TABLE TO
LXI D,SECTION_FLAG ;BUFFER AREA
MVI C,SEC_LEN+1 ;COUNT
CALL MOVER_R
SHLD TABLE_PNT ;UPDATE SECTION TABLE POINTER
IFEQI SECTION_FLAG,HEX_FF ;END OF TABLE
CLEAR SECTION_FLAG ;MAKE WCS DEFAULT AREA
LXI H,TEMP_SECTION ;NAME IS NOT FOUND IN TABLE
LXI D,SECTION_NAME ;SO LOAD IT INTO WCS IF IT
MVI C,SEC_LEN ;IS FOUND ON THE TAPE
CALL MOVER_R
CALL LOAD_SECTION

```

```

22-ENKAA-2.1 4000 4
543F 1864 C9
5440 1865
5440 1866 0855'C3
5443 1867
5443 1868 03'CD'21
06 OE 04'DF'11
0855'C2 00CF'CD
5451 1869
5451 1870 1D6C'CD
5454 1871 C9
5455 1872
5455 1873
5455 1874
5455 1875
5455 1876
5455 1877 0817'C3
5458 1878
5458 1879
5458 1880
5458 1881
5458 1882
5458 1883
5458 1884
5458 1885
5458 1886
5458 1887
5458 1888 OF 0209'3A
086F'DA
545F 1889
545F 1890 OF 0408'3A
086F'DA
5466 1891
5466 1892 023B'32 AF
546A 1893 28 06
546C 1894 OF79'C3
546F 1895
546F 1896
546F 1897
546F 1898
546F 1899 03B3'32 AF
5473 1900
5473 1901 OFCF'CD
5476 1902
5476 1903 02 06
5478 1904 0231'3A
547B 1905 B7
547C 1906 OF79'C2
547F 1907
547F 1908 0440'3A
5482 1909 03DA'32
5485 1910
5485 1911 OFCF'CD
5488 1912
5488 1913 OF 0231'3A
0895'D2
548F 1914 03DA'3A
5492 1915
5492 1916 0898'C3
5495 1917
5495 1918 0440'3A

```

```

G 5
Fiche 1 Frame G5 Sequence 58
RET ;EXIT FROM SEARCH LOOP
ELSE
IFEQ TEMP_SECTION,SECTION_NAME,SEC_LEN
CALL LOAD_SECTION
RET ;EXIT FROM SEARCH LOOP
ENDIF
ENDIF
JMP 1$ ;LOOP UNTIL SECTION FOUND
*****
DIAG_TEST
THIS SUBROUTINE INPUTS THE TEST PARAMETERS OF THE DIAGNOSE KEYWORD
ONE TEST NUMBER MEANS EXECUTE THAT TEST ONLY
TWO TEST NUMBERS MEANS EXECUTE THE TESTS BETWEEN THAT RANGE INCLUSIVE
*****
DIAG_TEST: IFN CONSOLE_TEST ;IF CONSOLE TEST, SKIP THIS
;
IFN WCS_LOADED ;ELSE SEE IF WCS LOADED
; IF NOT, REPORT ERROR
CLEAR EXECUTE ;CLEAR EXECUTE FLAG TO PROMPT
MVI B,<^X28> ;CODE FOR NO TEST CODE ERROR
JMP ERROR_ROUTINE_NO_RET ;REPORT ERROR
ENDIF
ENDIF
CLEAR TABLE_MODE ;SET TO NO TABLE MODE
CALL INPUT_NUM_IF ;LOOK FOR FIRST TEST NUMBER
MVI B,<^X02> ;NO FIRST NUMBER - ERROR
LDA ERROR ;GET ERROR FLAG
ORA A ;SET CONDITION CODES
JNZ ERROR_ROUTINE_NO_RET ;REPORT IF ERROR. DON'T RETURN
LDA INPUT_NUM+3 ;FIRST NUMBER IS FIRST TEST
STA TEST_NUM1 ;SAVE IT
CALL INPUT_NUM_IF ;GET SECOND NUMBER IF THERE
IF ERROR
LDA TEST_NUM1 ;NO SECOND NUM - USE FIRST TEST
ELSE
LDA INPUT_NUM+3 ;SAVE ENDING TEST NUMBER

```

```

ZZ-ENKAA-2.1 4C00 4
5498 1919 *****
5498 1920
5498 1921 03DB'32
549B 1922
549B 1923 C9
549C 1924
549C 1925
549C 1926
549C 1927
549C 1928
549C 1929
549C 1930
549C 1931
549C 1932
549C 1933
549C 1934
549C 1935
549C 1936
549C 1937
549C 1938
549C 1939 50 3E
549E 1940 03DB'32
54A1 1941
54A1 1942 C9
54A2 1943
54A2 1944
54A2 1945
54A2 1946
54A2 1947
54A2 1948
54A2 1949
54A2 1950
54A2 1951
54A2 1952
54A2 1953
54A2 1954 OFCF'CD
54A5 1955
54A5 1956 03 06
54A7 1957 0231'3A
54AA 1958 B7
54AB 1959 OF79'C2
54AE 1960
54AE 1961 OF 02C9'3A
          08C0'D2
54B5 1962
54B5 1963 FF 3E
54B7 1964 0314'32
54BA 1965 0315'32
54BD 1966
54BD 1967 08C6'C3
54C0 1968
54C0 1969 043F'2A
54C3 1970 0314'22
54C6 1971
54C6 1972 *****
54C6 1973
54C6 1974 C9
54C7 1975
54C7 1976
54C7 1977

```

H 5

Fiche 1 Frame M5

Sequence 59

```

ENDIF
STA TEST_NUM2
RET

```

```

*****
DIAG_CONTIN
THIS SUBROUTINE HANDLES THE DIAGNOSE CONTINUE KEYWORD
A CONTINUE WILL CAUSE THE TESTING TO GO TO THE END OF THE SECTION

```

```

NOTE: - - TO BE EFFECTIVE THIS KEYWORD MUST BE GIVEN AFTER THE TEST
KEYWORD AND THE TEST KEYWORD MUST HAVE ONLY 1 PARAMETER
IF THE TEST PARAMETER IS GIVEN AFTER THE CONTINUE KEYWORD
IT WILL BE CANCELED OUT.
*****

```

```

DIAG_CONTIN: MVI A,HEX_50 ;MOVE LAST POSSIBLE TEST
             STA TEST_NUM2 ;TO ENDING TEST NUMBER
             RET

```

```

*****
DIAG_PASS
THIS SUBROUTINE INPUTS THE DIAGNOSE PASS COUNT.
IF NOT SPECIFIED THE PASS COUNT WILL DEFAULT TO ONE.
IF THE PASS COUNT IS SPECIFIED AS -1 THIS INDICATES A CONTINUOUS LOOP
*****

```

```

DIAG_PASS: CALL INPUT_NUM_IF ;GET THE PASS COUNT
           MVI B,<^X03> ;PASS COUNT ERROR
           LDA ERROR ;GET ERROR FLAG
           ORA A ;SET CONDITION CODES
           JNZ ERROR_ROUTINE_NO_RET ;REPORT IF ERROR. DON'T RETURN
           IF MINUS
           MVI A,HEX_FF ;SET TO INFINITE PASSES
           STA PASS_CNT
           STA PASS_CNT+1
           ELSE
           LHL INPUT_NUM+2
           SHLD PASS_CNT ;MOVE 2 BYTES PASS COUNT
           ENDIF
           RET

```

```

ZZ-ENKAA-2.1 4000 4
54C7 1978
54C7 1979
54C7 1980
54C7 1981
54C7 1982
54C7 1983
54C7 1984
54C7 1985
54C7 1986
54C7 1987
54C7 1988 FF 3E
54C9 1989 0314'32
54CC 1990 0315'32
54CF 1991
54CF 1992 0345'32 3C AF
54D4 1993
54D4 1994 C9
54D5 1995
54D5 1996
54D5 1997
54D5 1998
54D5 1999
54D5 2000
54D5 2001
54D5 2002
54D5 2003
54D5 2004
54D5 2005
54D5 2006 03'46'21
54D8 2007 124A'CD
54DB 2008
54DB 2009 01 3E
54DD 2010
54DD 2011 03C5'32
54E0 2012
54E0 2013 06'AE'21
54E3 2014 03C6'22
54E6 2015
54E6 2016 C5 46 00'55'21
77 05 3E
54EE 2017
54EE 2018 47 03C5'3A
A0 005A'3A
08FC'CA
54F9 2019
54F9 2020 11AD'CD
54FC 2021
54FC 2022
54FC 2023
54FC 2024 1575'CD
54FF 2025 03'C5'21
5502 2026 7E
5503 2027 17
5504 2028 77
5505 2029
5505 2030 35 00'55'21
70 C1 08EE'C2
550E 2031
550E 2032 03DF'3A
5511 2033 B7

```

```

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
*****

```

```

DIAG_SHORTEN: MVI A,HEX_FF ;SET TO INFINITE PASSES
STA PASS_CNT
STA PASS_CNT+1
SET SHORTEN ;SET SHORTEN MODE FLAG
RET

```

```

*****
SHOW_COMMAND
THIS SUBROUTINE HANDLES THE COMMAND SHOW
IT PRINTS THE STATE OF THE FLAGS HALT,LOOP,NER,BELL,SOMM AND THE
SOMM ADDRESS IF THE SOMM FLAG IS SET
*****

```

```

SHOW_COMMAND: LXI H,SHOWFLAG_A
CALL PRINT_STRING_R
MVI A,1 ;SET UP ROTATING ONE TO MASK
;FLAGS WITH
STA TEMP_BYTE
LXI H,PRT_FLAGS_A ;SET UP POINTER TO ASCII
SHLD TEMP_WORD ;NAMES OF FLAGS
DO 5 ;5 FLAGS TO PRINT IN BYTE
*
*
IFMB FLAGS,TEMP_BYTE ;IS FLAG SET FOR THIS BIT
*
*
CALL PRINT_FLG ;PRINT FLAG NAME IF SET
ENDIF
CALL INC_TEMPW_5 ;INC TEMP WORD TO NEXT FLAG
LXI H,TEMP_BYTE ;MOVE MASK ONE BIT LEFT
MOV A,M
RAL
MOV M,A
ENDDO
LDA TRACE ;GET TRACE FLAG
ORA A ;SET CONDITION CODES

```

```

ZZ-ENKAA-2.1 4C00 4
5512 2034 11AD'C4
5515 2035
5515 2036 1575'CD
5518 2037
5518 2038 0F 039F'3A
092B'D2
551F 2039
551F 2040 11AD'CD
5522 2041
5522 2042 039D'2A
5525 2043 005B'22
5528 2044 118D'CD
552B 2045
552B 2046
552B 2047
552B 2048 1575'CD
552E 2049
552E 2050 005A'3A
5531 2051 47
5532 2052 80 3E
5534 2053 A0
5535 2054 11AD'C4
5538 2055
5538 2056 C9
5539 2057
5539 2058
5539 2059
5539 2060
5539 2061
5539 2062
5539 2063
5539 2064
5539 2065
5539 2066 02F2'32 3C AF
553E 2067 07EB'CD
5541 2068 02F2'32 AF
5545 2069 C9
5546 2070
5546 2071
5546 2072
5546 2073
5546 2074
5546 2075
5546 2076
5546 2077
5546 2078
5546 2079
5546 2080 04 06
5548 2081
5548 2082 003A'3A
554B 2083 B7
554C 2084 0F7E'CA
554F 2085
554F 2086
554F 2087 0F 0209'3A
0959'DA
0F41'CD
5556 2088
5559 2089
5559 2090
5559 2091 3D D3

```

```

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

```

```

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

```

```

J 5
Fiche 1 Frame J5 Sequence 61
CNZ PRINT_FLG ;PRINT OUT TRACE FLAG IF SET
CALL INC_TEMPW_5
IF SOMM_FLG ;PRINT OUT SOMM FLAG IF SET
CALL PRINT_FLG
LHLD SOMM_ADDRESS ;PRINT SOMM ADDRESS WITH FLAG
SHLD HEX_BUF
CALL PRINT_HEX_2
ENDIF
CALL INC_TEMPW_5 ;POINT TO NEXT FLAG NAME
LDA FLAGS ;GET FLAGS INDICATOR
MOV B,A ;PUT BYTE IN B
MVI A,HEX_80 ;PUT AN 80 IN A
ANA B ;AND 80 WITH FLAGS INDICATOR
CNZ PRINT_FLG ;PRINT FLAG NAME IF SET
RET

```

```

*****
:
: LOADER
: THIS ROUTINE LOADS A IMAGE INTO WCS OR THE CONSOLE BUT DOES
: NOT START IT AND DOES NOT INITIALIZE IT
:
*****

```

```

LOADER: SET NOINIT
CALL DIAG_SECTION
CLEAR NOINIT
RET

```

```

*****
:
: CONT COMMAND
: THE CONTINUE COMMAND ALLOWS CONTINUATION OF TEST EXECUTION AFTER
: TESTING HAS BEEN HALTED
:
*****

```

```

CONT_COMMAND: MVI B,<^X04> ;ERROR CODE IF CONTINUE CLEAR
LDA CONTINUE ;GET CONTINUE FLAG
ORA A ;SET CONDITION CODES
JZ ERROR_ROUTINE ;NO CONTINUE AT THIS POINT IF
; CONTINUE FLAG IS CLEAR
; ELSE CONTINUE ON
IFN CONSOLE_TEST ;DO WR RESTORE IF 80B5
CALL RESTORE_WR ; TESTS ARE NOT EXECUTING
ENDIF
OUT SETLIT ;TURN RUN LITE BACK ON

```

```

22-ENKAA-2.1 4000 4
555B 2092
555B 2093 0204'3A
555E 2094 023B'32
5561 2095
5561 2096 00'BE'21
5564 2097 00'BB'11
5567 2098 118E'CD
556A 2099
556A 2100 00C8'2A
556D 2101 00C4'22
5570 2102
5570 2103 0205'3A
5573 2104 B7
5574 2105 13DD'C4
5577 2106
5577 2107 020A'32 3C AF
557C 2108
557C 2109 C9
557D 2110
557D 2111
557D 2112
557D 2113
557D 2114
557D 2115
557D 2116
557D 2117
557D 2118
557D 2119 05'BF'21
5580 2120 1FDB'CD
5583 2121
5583 2122 C9
5584 2123
5584 2124
5584 2125
5584 2126
5584 2127
5584 2128
5584 2129
5584 2130
5584 2131
5584 2132
5584 2133
5584 2134
5584 2135 0A66'CD
5587 2136 08 06
5589 2137 159E'C3
558C 2138
558C 2139
558C 2140
558C 2141
558C 2142
558C 2143
558C 2144
558C 2145
558C 2146
558C 2147
558C 2148
558C 2149 01 06
558E 2150 159E'C3
5591 2151

```

K 5

Fiche 1 Frame K5

Sequence 62

```

LDA C_EXECUTE ;RESTORE FLAGS
STA EXECUTE

```

```

LXI H,CONT_SAVE_CSR ;RESTORE CSR
LXI D,SAVED_CSR
CALL MOVER_3_R

```

```

LHLD CONT_SAVE_UPC ;RESTORE UPC
SHLD SAVED_UPC

```

```

LDA C_RUNNING ;GET OLD RUNNING FLAG
ORA A ;SET CONDITION CODES
CNZ RESTART_CPU ;RESTART IF SET

```

```

SET CONT_DONE

```

```

RET

```

```

:*****
:
: SET_COMMAND
: THE SET COMMAND IS USED TO SET THE STATE OF FLAGS USED TO CONTROL
: THE EXECUTION OF DIAGNOSTICS
:
:*****

```

```

SET_COMMAND: LXI H,SET_TAB ;POINTER TO PARSE TABLE
CALL PAR_TEMP_BUF ;PARSE SET COMMANDS
RET

```

```

:*****
:
: THIS SUBROUTINE HANDLES THE SET HALT COMMAND
: SET HALT IS FOR THE HALT ON ERROR FUNCTION
: THIS SET COMMAND SETS A LOCAL FLAG AND WRITES THE FLAG
: STATUS WORD IN LS
:
:*****

```

```

SET_HALT: CALL CLEAR_LOOP_COM ;CLEAR LOOP COMMAND BIT
MVI B,HALT_S_DEF ;BIT TO SET IN B
JMP SET_LS_STAT

```

```

:*****
:
: SET_LOOP
: THIS SUBROUTINE HANDLES THE SET LOOP COMMAND
: SET LOOP IS FOR LOOP ON ERROR FUNCTION
: THIS SET COMMAND SETS A LOCAL FLAG AND WRITES THE FLAG
: STATUS WORD IN LS
:
:*****

```

```

SET_LOOP: MVI B,LOOP_S_DEF ;BIT TO SET IN B
JMP SET_LS_STAT

```

ZZ-ENKAA-2.1 4000 4  
 5591 2152  
 5591 2153  
 5591 2154  
 5591 2155  
 5591 2156  
 5591 2157  
 5591 2158  
 5591 2159  
 5591 2160  
 5591 2161  
 5591 2162  
 5591 2163  
 5591 2164 02 06  
 5593 2165 159E'C3  
 5596 2166  
 5596 2167  
 5596 2168  
 5596 2169  
 5596 2170  
 5596 2171  
 5596 2172  
 5596 2173  
 5596 2174  
 5596 2175  
 5596 2176  
 5596 2177  
 5596 2178  
 5596 2179 0A66'CD  
 5599 2180 04 06  
 559B 2181 159E'C3  
 559E 2182  
 559E 2183  
 559E 2184  
 559E 2185  
 559E 2186  
 559E 2187  
 559E 2188  
 559E 2189  
 559E 2190  
 559E 2191  
 559E 2192  
 559E 2193  
 559E 2194 10 06  
 55A0 2195 159E'C3  
 55A3 2196  
 55A3 2197  
 55A3 2198  
 55A3 2199  
 55A3 2200  
 55A3 2201  
 55A3 2202  
 55A3 2203  
 55A3 2204  
 55A3 2205  
 55A3 2206  
 55A3 2207  
 55A3 2208  
 55A3 2209 80 06  
 55A5 2210 159E'C3  
 55A8 2211

```

*****
:
: SET_NER
: THIS SUBROUTINE HANDLES THE SET_NER COMMAND
: SET_NER IS FOR 'NO ERROR REPORT' FUNCTION
: THIS SET COMMAND SETS A LOCAL FLAG AND WRITES THE FLAG
: STATUS WORD IN LS
:
*****
  
```

```

SET_NER:      MVI      B,NER_S_DEF          ;BIT TO SET IN B
              JMP      SET_LS_STAT
  
```

```

*****
:
: SET_BELL
: THIS SUBROUTINE HANDLES THE SET_BELL COMMAND
: SET_BELL RINGS THE TERMINAL BELL ON ERROR
: THIS SET COMMAND SETS A LOCAL FLAG AND WRITES THE FLAG
: STATUS WORD IN LS
:
*****
  
```

```

SET_BELL:     CALL     CLEAR_LOOP_COM      ;CLEAR LOOP COMMAND BIT
              MVI      B,BELL_S_DEF       ;BIT TO SET IN B
              JMP      SET_LS_STAT
  
```

```

*****
:
: SET_SER
: THIS SUBROUTINE HANDLES THE 'SET SINGLE BIT ERROR REPORTS' COMMAND.
: SET_SER CAUSES MEMORY SINGLE BIT ERRORS TO BE REPORTED AS ERRORS
: THIS SET COMMAND SETS A LOCAL FLAG AND WRITES THE FLAG
: STATUS WORD IN LS
:
*****
  
```

```

SET_SER:      MVI      B,SER_S_DEF        ;BIT TO SET IN B
              JMP      SET_LS_STAT        ;WRITE IN LS ERROR CONTROL
  
```

```

*****
:
: SET_SP
: THIS SUBROUTINE HANDLES THE 'SET SPECIAL BIT' COMMAND. SET_SP CAUSES
: SIZING TESTS TO LOOP REGARDLESS OF THERE RESULTS. IT IS USED TO LOOP
: ON A SIZING TEST THAT DOESN'T FIND A DEVICE THAT IS PRESENT OR VICE
: VERSA. TESTS THAT USE IT WILL EXPLAIN IT'S OPERATION IN THE TEST
: DOCUMENTATION.
:
*****
  
```

```

SET_SP:      MVI      B,SPECIAL_S_DEF     ;BIT TO SET IN B
              JMP      SET_LS_STAT        ;WRITE IN LS ERROR CONTROL
  
```

```

ZZ-ENKAA-2.1 4000 4
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
55AB 2224 B7
55AC 2225 0A88'C4
55AF 2226
55AF 2227
55AF 2228
55AF 2229 0F6B'CD
55B2 2230
55B2 2231
55B2 2232
55B2 2233 AF
55B3 2234 3C
55B4 2235 039F'32
55B7 2236 030C'32
55BA 2237 006E'32
55BD 2238
55BD 2239 043F'2A
55C0 2240 039D'22
55C3 2241 0F52'C3
55C6 2242
55C6 2243
55C6 2244
55C6 2245
55C6 2246
55C6 2247
55C6 2248
55C6 2249
55C6 2250
55C6 2251
55C6 2252
55C6 2253
55C6 2254
55C6 2255
55C6 2256
55C6 2257 0FCF'CD
55C9 2258
55C9 2259 0F 0231'3A
09D5'D2
01 3E
55D0 2260
55D2 2261
55D2 2262 09DD'C3
55D5 2263 043F'2A
55D8 2264 03A3'22
55D8 2265 03 3E
55DD 2266
55DD 2267
55DD 2268 02C8'32
55E0 2269 0946'C3
55E3 2270

```

```

*****
SET SOMM
THIS ROUTINE SETS THE STOP ON MICRO MATCH FLAG AND WRITES BAD PARITY
AT THE GIVEN ADDRESS. WHEN A PARITY ERROR OCCURS THIS ADDRESS IS
CHECKED TO SEE IF IT WAS PUT THERE INTENTIONALLY. IF A SET SOMM ADDRESS
IS GIVEN WHEN THERE IS ONE ALREADY IN EFFECT THE PARTIY IS CORRECTED
AT THE OLD SOMM ADDRESS BEFORE THE NEW ONE IS WRITTEN
*****

```

```

SET_SOMM:      LDA      SOMM_FLG      ;GET SOMM FLAG
                ORA      A              ;SET CONDITION CODES
                CNZ      CLEAR_SOMM    ;IF SOMM SET WHEN SET IS DONE
                ; THE OLD SOMM ADDRESS IS
                ; CLEARED FIRST

                CALL     INPUT_NUM_CHECK ;CHECK FOR INPUT,REPORT ERR
                ; IF NO INPUT, ELSE LOAD
                ; INPUT_NUM WITH SOMM ADDRESS

                XRA      A              ;CLEAR A
                INR      A              ;INCR A
                STA      SOMM_FLG      ;SET SOMM FLAG
                STA      PAR_ON        ;SET PAR_ON FLAG
                STA      WCS_BAD_PARITY ;SET WCS_BAD_PARITY FLAG

                LHL      INPUT_NUM+2    ;LOAD INPUT_NUM+2
                SHLD     SOMM_ADDRESS  ;SAVE THE SOMM ADDRESS
                JMP      CHANGE_WCS    ;CHANGE PARITY TO BAD

```

```

*****
SET_STEP
THIS ROUTINE SETS MICRO STEPPING MODE
MICRO_STEP 0 = NO MICRO STEP
            1 = SPACE BAR MODE
            3 = COUNT MODE

MICRO STEPPING WILL BE DONE AS PART OF THE IDLE-ATTENTION
LOOPS USED IN THE PROGRAM
*****

```

```

SET_STEP:      CALL     INPUT_NUM_IF

                IF      ERROR          ;ERROR SET IF NO NUMBER
                MVI     A,1            ;THEN SET TO SPACE BAR MODE

                ELSE
                LHL      INPUT_NUM+2    ;LOAD INPUT_NUM+2
                SHLD     STEP_CNT      ;SAVE NUMBER OF COUNTS
                MVI     A,3            ;SET TO COUNT MODE
                ENDIF

                STA      MICRO_STEP    ;SAVE MODE IN MICRO_STEP
                JMP      CONT_COMMAND

```

27-ENKAA-2.1 4C00 4  
 55E3 2271  
 55E3 2272  
 55E3 2273  
 55E3 2274  
 55E3 2275  
 55E3 2276  
 55E3 2277  
 55E3 2278  
 55E3 2279  
 55E3 2280  
 55E3 2281 03DF'32 3C AF  
 55E8 2282  
 55E8 2283 C9  
 55E9 2284  
 55E9 2285  
 55E9 2286  
 55E9 2287  
 55E9 2288  
 55E9 2289  
 55E9 2290  
 55E9 2291  
 55E9 2292  
 55E9 2293  
 55E9 2294  
 55E9 2295 0FCF'CD  
 55EC 2296  
 55EC 2297 0F 0231'3A  
 09FB'DA  
 55F3 2298 006E'32 3C AF  
 55FB 2300 0F52'CD  
 55FB 2301  
 55FB 2302 030C'32 AF  
 55FF 2303  
 55FF 2304 C9  
 5600 2305  
 5600 2306  
 5600 2307  
 5600 2308  
 5600 2309  
 5600 2310  
 5600 2311  
 5600 2312  
 5600 2313  
 5600 2314  
 5600 2315  
 5600 2316  
 5600 2317  
 5600 2318  
 5600 2319  
 5600 2320  
 5600 2321 0F6B'CD  
 5603 2322  
 5603 2323  
 5603 2324  
 5603 2325 0203'3A  
 5606 2326 B7  
 5607 2327 0A28'C4  
 560A 2328  
 560A 2329 0203'32 3C AF

```

*****
:
: SET TRACE
: THIS SUBROUTINE SETS THE TRACE FLAG
: WHEN THE TRACE FLAG IS SET THE TEST NUMBERS WILL BE PRINTED AS THEY
: BEGIN
:
*****

```

```

SET_TRACE:   SET     TRACE                ;SET TO TRACE MODE
             RET

```

```

*****
:
: SET PAR
: THIS ROUTINE SETS A PARITY ERROR IN WCS AT THE GIVEN ADDRESS
: IF NO ADDRESS IS GIVEN THEN ONLY A DISABLE HALT ON PARITY ERRORS
: WILL BE DONE
:
*****

```

```

SET_PAR:     CALL    INPUT_NUM_IF        ;CHECK FOR PARITY ERROR ADDR
             IFN    ERROR                ;IF NUMBER INPUT
             SET    WCS_BAD_PARITY      ;SET FLAG
             CALL   CHANGE_WCS         ;CHANGE PARITY TO BAD
             ENDF
             CLEAR  PAR_ON              ;SET FOR NO HALT ON PAR ERR
             RET

```

```

*****
:
: SET BREAK
: THIS SUBROUTINE SETS A BRANCH IN RAM AT THE ADDRESS GIVEN
: THE BRANCH GOES TO A ROUTINE THAT WILL PRINT OUT INFO AND THEN
: RETURN TO THE PARSER SO EXAMINES CAN BE MADE. THE LOCATION SELECTED
: AND ITS CONTENTS ARE SAVED AND RESTORE IF A BREAK IS SET AT ANOTHER
: ADDRESS, OR A CL BR IS ISSUED.
:
: CAUTION: THIS USES 3 BYTES AT THE ADDRESS GIVEN, UNTIL CLEARED OR
: CHANGED TO ANOTHER LOCATION.
:
*****

```

```

SET_BREAK:   CALL    INPUT_NUM_CHECK    ;CHECK FOR INPUT,REPORT ERR
             ; IF NO INPUT, ELSE LOAD
             ; INPUT_NUM WITH BREAK ADDRESS
             LDA    BREAK_FLG          ;SEE IF BREAK WAS ISSUED BEFORE
             ORA    A                  ;SET CONDITION CODES
             CNZ    CLEAR_BREAK        ;CLEAR OLD BREAK ADDRESS
             SET    BREAK_FLG          ;SET FLAG

```



```

ZZ-ENKAA-2.1 4C00 4
5643 2390
5643 2391
5643 2392 00 62 21
5646 2393 0A49'C3
5649 2394
5649 2395
5649 2396
5649 2397
5649 2398
5649 2399
5649 2400
5649 2401
5649 2402
5649 2403
5649 2404
5649 2405
5649 2406
5649 2407
5649 2408 11 06
564B 2409 0F5E'CD
564E 2410
564E 2411
564E 2412 7C
564F 2413 65
5650 2414 6F
5651 2415 03A1'22
5654 2416
5654 2417 13CE'CD
5657 2418 1A7A'C3
565A 2419
565A 2420
565A 2421
565A 2422
565A 2423
565A 2424
565A 2425
565A 2426
565A 2427
565A 2428
565A 2429 05'82'21
565D 2430 1FDB'CD
5660 2431
5660 2432 C9
5661 2433
5661 2434
5661 2435
5661 2436
5661 2437
5661 2438
5661 2439
5661 2440
5661 2441
5661 2442
5661 2443
5661 2444
5661 2445
5661 2446 F7 06
5663 2447 15AB'C3
5666 2448
5666 2449

```

```

:*****
CLEAR_FIFO: LXI H,WCS_CL_FIFO ;ADDRESS OF WCS SUBROUTINE
             JMP EXEC_SUB ;EXECUTE SUBROUTINE

```

```

:*****
EXEC_SUB
: THIS ROUTINE EXECUTES A SUBROUTINE IN WCS WHICH PERFORMS A SPECIFIC
: FUNCTION.
: INPUT CONDITIONS
: H+L = ADDRESS OF WCS SUBROUTINE
:*****

```

```

EXEC_SUB: MVI B,<^X11> ;ERROR CODE IF WCS NOT LOADED
          CALL CHECK_WCS_LOADED ;CHECK FOR WCS LOADED SET
          ; RETURN IF OK
          MOV A,H ;GET SUBROUTINE ADDR
          MOV H,L
          MOV L,A
          SHLD STARTING_UPC
          CALL START_CPU ;START THE SUBROUTINE
          JMP SUB_ATTENTION ;WAIT FOR EXIT

```

```

:*****
CLEAR_COMMAND
: THE CLEAR COMMAND IS USED TO CLEAR THE FLAGS USED TO CONTROL
: THE EXECUTION OF DIAGNOSTICS
:*****

```

```

CLEAR_COMMAND: LXI H,CLEAR_TAB ;POINTER TO PARSE TABLE
              CALL PAR_TEMP_BUF ;PARSE THE COMMAND
              RET

```

```

:*****
CLEAR_HALT
: THIS SUBROUTINE HANDLES THE CLEAR HALT COMMAND
: CLEAR HALT IS TO DISABLE HALT ON ERROR
: THIS CLEAR COMMAND CLEARS A LOCAL FLAG AND WRITES THE FLAG
: STATUS WORD IN LS
:*****

```

```

CLEAR_HALT: MVI B,HALT_C_DEF ;BIT TO CLEAR IN B
           JMP CLEAR_CS_STAT

```

5666 2450  
5666 2451  
5666 2452  
5666 2453  
5666 2454  
5666 2455  
5666 2456  
5666 2457  
5666 2458  
5666 2459  
5666 2460  
5668 2461  
5668 2462  
5668 2463  
5668 2464  
5668 2465  
5668 2466  
5668 2467  
5668 2468  
5668 2469  
5668 2470  
5668 2471  
5668 2472  
5668 2473  
5668 2474  
5668 2475  
5668 2476  
566F 2477  
5671 2478  
5671 2479  
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  
5676 2496  
5676 2497  
5679 2498  
5679 2499  
5679 2500  
5679 2501  
5679 2502  
5679 2503  
5679 2504  
5679 2505  
5679 2506  
5679 2507  
5679 2508  
5679 2509

BF 06  
15A8'C3

0057'32 AF  
BE 06

15A8'C3

BD 06

15A8'C3

```

*****
CLEAR_LOOP_COM:
THIS SUBROUTINE HANDLES THE CLEAR LOOP 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_COM: MVI    B,LOOP_COM_C_DEF        ;BIT TO CLEAR LOOP COMMAND
                 JMP    CLEAR_LS_STAT

*****

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_LOOP:    CLEAR  ERROR_CON                ;CLEAR 8085 ERROR LOOP FLAG
               MVI    B,LOOP_C_DEF            ;BITS TO CLEAR IN B (LOOP
               JMP    CLEAR_LS_STAT          ; AND LOOP COMMAND BITS)

*****

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_NER:    MVI    B,NER_C_DEF              ;BITS TO CLEAR IN B (NER
               JMP    CLEAR_LS_STAT          ; AND LOOP COMMAND BITS)

*****

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
*****

```

```

ZZ-ENKAA-2.1 4C00 4
5679 2510
5679 2511
5679 2512
5679 2513 FB 06
567B 2514 15A8'C3
567E 2515
567E 2516
567E 2517
567E 2518
567E 2519
567E 2520
567E 2521
567E 2522
567E 2523
567E 2524
567E 2525
567E 2526
567E 2527
567E 2528 EF 06
5680 2529 15A8'C3
5683 2530
5683 2531
5683 2532
5683 2533
5683 2534
5683 2535
5683 2536
5683 2537
5683 2538
5683 2539
5683 2540 7F 06
5685 2541 15A8'C3
5688 2542
5688 2543
5688 2544
5688 2545
5688 2546
5688 2547
5688 2548
5688 2549
5688 2550
5688 2551
5688 2552 039F'32 AF
568C 2553
568C 2554 039D'2A
568F 2555 043F'22
5692 2556
5692 2557 0F52'C3
5695 2558
5695 2559
5695 2560
5695 2561
5695 2562
5695 2563
5695 2564
5695 2565
5695 2566
5695 2567
5695 2568
5695 2569 03DF'32 AF

```

```

:*****
CLEAR_BELL: MVI B,BELL_C_DEF ;LOAD B REG WITH BIT TO CLR
            JMP CLEAR_CS_STAT

```

```

:*****
: CLEAR SER
: THIS SUBROUTINE HANDLES THE CLEAR SINGLE BIT ERROR REPORTS COMMAND
: CLEAR SER CAUSES MEMORY SINGLE BIT ERRORS NOT TO BE REPORTED AS ERRORS
: THIS CLEAR COMMAND CLEARS A LOCAL FLAG AND WRITES THE FLAG
: STATUS WORD IN LS
:*****

```

```

CLEAR_SER: MVI B,SER_C_DEF ;LOAD B REG WITH BIT TO CLR
           JMP CLEAR_LS_STAT

```

```

:*****
: CLEAR SP
: THIS SUBROUTINE CLEARS THE SPECIAL BIT. SEE SET SP FOR AN EXPLANATION
: OF THE SPECIAL BIT.
:*****

```

```

CLEAR_SP: MVI B,SPECIAL_C_DEF ;BIT TO CLEAR IN B
          JMP CLEAR_LS_STAT ;WRITE IN LS ERROR CONTROL

```

```

:*****
: CLEAR SOMM
: THIS ROUTINE CLEARS THE STOP ON MICRO MATCH FLAG AND WRITES GOOD
: PARITY AT THE MATCH ADDRESS.
:*****

```

```

CLEAR_SOMM: CLEAR SOMM_FLG
           LHLD SOMM_ADDRESS ;GET SOMM ADDRESS TO CLEAR
           SHLD INPUT_NUM+2
           JMP CHANGE_WCS ;SET PARITY GOOD ON ADDRESS

```

```

:*****
: CLEAR TRACE
: THIS SUBROUTINE CLEARS THE TRACE FLAG
: WHEN THE TRACE FLAG IS CLEARED THE TEST NUMBERS WILL NOT BE PRINTED
: AS THEY BEGIN
:*****

```

```

CLEAR_TRACE: CLEAR TRACE ;CLEAR TRACE FLAG

```

```

ZZ-ENKAA-2.1 4C00 4
5699 2570
5699 2571 C9
569A 2572
569A 2573
569A 2574
569A 2575
569A 2576
569A 2577
569A 2578
569A 2579
569A 2580 0A79'CD
569D 2581 0A74'CD
56A0 2582 0984'CD
56A3 2583 0A6B'CD
56A6 2584 0A95'CD
56A9 2585 0A7E'CD
56AC 2586 0A83'C3
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
56B2 2599
56B2 2600 0AB5'C3
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
56B8 2615 B7
56B9 2616 0F52'CC
56BC 2617
56BC 2618
56BC 2619 030C'32 3C AF
56C1 2620
56C1 2621 C9
56C2 2622
56C2 2623
56C2 2624
56C2 2625
56C2 2626
56C2 2627
56C2 2628
56C2 2629

```

RET

```

*****
: CLEAR DEFAULT
: THIS ROUTINE SETS THE FLAGS TO THEIR DEFAULT VALUES
*****

```

```

CLEAR_DEFAULT: CALL CLEAR_BELL
                CALL CLEAR_NER
                CALL SET_HALT
                CALL CLEAR_LOOP
                CALL CLEAR_TRACE
                CALL CLEAR_SER
                JMP CLEAR_SP

```

```

*****
: 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
*****

```

```

CLEAR_PAR: CALL INPUT_NUM_IF ;GET CLEAR ADDRESS
           JMP INT_CLEAR_PAR ;CALL INTERNAL SUBROUTINE

```

```

*****
: 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
*****

```

```

INT_CLEAR_PAR: LDA ERROR ;GET ERROR FLAG
               ORA A ;SET CONDITION CODES
               CZ CHANGE_WCS ;CHANGE PARITY TO GOOD IF
                           ; ADDRESS GIVEN
               SET PAR_ON ;SET FOR HALT ON PAR ERR
               RET

```

```

*****
: 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

```

```

ZZ-ENKAA-2.1 4C00 4
56C2 2630
56C2 2631
56C2 2632
56C2 2633
56C2 2634 0A79'CD
56C5 2635 0A61'CD
56C8 2636 0991'CD
56CB 2637 098C'CD
56CE 2638 40 06
56D0 2639 159E'CD
56D3 2640
56D3 2641 0946'C3
56D6 2642
56D6 2643
56D6 2644
56D6 2645
56D6 2646
56D6 2647
56D6 2648
56D6 2649
56D6 2650 0333'32 3C AF
56DB 2651 04'F5'21
56DE 2652 1FDB'C3
56E1 2653
56E1 2654
56E1 2655
56E1 2656
56E1 2657
56E1 2658
56E1 2659
56E1 2660
56E1 2661 06'06'21
56E4 2662 1FDB'CD
56E7 2663
56E7 2664 0333'3A
56EA 2665 B7
56EB 2666 125A'C4
56EE 2667 C9
56EF 2668
56EF 2669
56EF 2670
56EF 2671
56EF 2672
56EF 2673
56EF 2674
56EF 2675
56EF 2676
56EF 2677
56EF 2678
56EF 2679 0F6B'CD
56F2 2680
56F2 2681
56F2 2682
56F2 2683 043F'2A
56F5 2684 005B'22
56F8 2685
56F8 2686 03'2E'21
56FB 2687 124A'CD
56FE 2688 118D'CD
5701 2689

```

```

: THIS COMMAND ALSO WRITES THE LS STATUS WORD
:*****
LOOP_COMMAND: CALL CLEAR_BELL
               CALL CLEAR_HALT
               CALL SET_NER
               CALL SET_LOOP
               MVI B,LOOP_COM_S_DEF ;BIT TO SET FOR LOOP COMMAND
               CALL SET_LS_STAT
               JMP CONT_COMMAND

:*****
REPEAT_COMMAND
:*****
REPEAT_COMMAND: SET REPEAT ;SET REPEAT FLAG
                LXI H,MAIN_TAB ;POINTER TO MAIN TABLE
                JMP PAR_TEMP_BUF

:*****
EXAM_COMMAND
:*****
EXAM_COMMAND: LXI H,EXAM_TAB ;POINTER TO PARSE TABLE
              CALL PAR_TEMP_BUF
              LDA REPEAT ;GET REPEAT FLAG
              ORA A ;SET CONDITION CODES
              CNZ CRLF_R ;DO <CR><LF> IF REPEAT
              RET

:*****
EXAM_RAM
: THIS ROUTINE EXAMINES THE RAM OR ROM OF THE 8085 CONSOLE AT THE
: ADDRESS GIVEN
:*****
EXAM_RAM: CALL INPUT_NUM_CHECK ;CHECK FOR INPUT,REPORT ERR
          ; IF NO INPUT, ELSE LOAD
          ; INPUT_NUM WITH RAM ADDR
          LHL INPUT_NUM+2 ;ELSE GET ADDRESS
          SHLD HEX_BUF ;MOVE ADDR TO PRINT AREA
          LXI H,RAM_A
          CALL PRINT_STPING_R
          CALL PRINT_HEX_2 ;PRINT ADDRESS

```

```

ZZ-ENKAA-2.1 4C00 4
5701 2690 043F'3A
5704 2691 67
5705 2692 0440'3A
5708 2693 6F
5709 2694
5709 2695 00'5E'11
570C 2696 04 0E
570E 2697
570E 2698 7E
570F 2699 12
5710 2700 23
5711 2701 1B
5712 2702 0D
5713 2703 0B0E'C2
5716 2704
5716 2705 11B8'C3
5719 2706
5719 2707
5719 2708
5719 2709
5719 2710
5719 2711
5719 2712
5719 2713
5719 2714
5719 2715 1116'CD
571C 2716
571C 2717 02'0D'21
571F 2718 124A'CD
5722 2719
5722 2720 0FC1'CD
5725 2721
5725 2722 0064'32 AF
5729 2723
5729 2724 1116'C3
572C 2725
572C 2726
572C 2727
572C 2728
572C 2729
572C 2730
572C 2731
572C 2732
572C 2733
572C 2734 1148'CD
572F 2735 00C2'2A
5732 2736 005B'22
5735 2737
5735 2738 0F 02C8'3A
                    *****
                    *
                    *
                    *
                    *
                    *
                    *****
573C 2739 00'5B'21
573F 2740 158C'CD
5742 2741
5742 2742
5742 2743
5742 2744
5742 2745 1148'CD
5745 2746
5745 2747 03'EC'21
5748 2748 124A'CD

```

```

H 6
Fiche 1 Frame H6 Sequence 72
LDA INPUT_NUM+2 ;PUT ADDRESS IN H+L TO READ RAM
MOV H,A
LDA INPUT_NUM+3
MOV L,A
LXI D,HEX_BUF+3 ;MOVE 4 BYTES TO BE PRINTED
MVI C,4 ;COUNTER IN C
1$: MOV A,M ;GET BYTE
STAX D ;STORE AT HEX_BUF
INX H ;POINT TO NEXT BYTE
DCX D ;BACKUP HEX_BUF
DCR C ;DECREMENT COUNTER
JNZ 1$ ;REPEAT UNTIL 4 BYTES LOADED
JMP PRINT_HEX_4 ;PRINT 4 BYTES AT RAM ADDRESS

```

```

*****
EXAM_CSR
THIS ROUTINE EXAMINES THE CSR
*****

```

```

EXAM_CSR: CALL LOAD_CSR_R ;GET THE CSR
LXI H,CSR_A ;PRINT 'CSR '
CALL PRINT_STRING_R
CALL PRINT_CSR_VAL
CLEAR PARITY_ON ;DONT CALC PARITY ON REWRITE
JMP LOAD_CSR_R ;RETURN THE CSR

```

```

*****
EXAM_UPC
THIS ROUTINE EXAMINES THE UPC
*****

```

```

EXAM_UPC: CALL LOAD_UPC_R ;GET THE UPC
LHLD UPC_VALUE
SHLD HEX_BUF ;SAVE THE VALUE
IF MICRO_STEP ;SEE IF MICRO_STEP MODE
LXI H,HEX_BUF ;POINT TO STORED UPC
CALL DECR_WORD_R ;BACK UP UPC TO REFLECT
; ACTUAL INSTRUCTION IN
; PRINTOUT
ENDIF
CALL LOAD_UPC_R ;RETURN THE UPC
PRINT_UPC: LXI H,UPC_A ;PRINT 'UPC '
CALL PRINT_STRING_R

```

```

ZZ-ENKAA-2.1 4000 4
574B 2749 11BD' C3
574E 2750
574E 2751
574E 2752
574E 2753
574E 2754
574E 2755
574E 2756
574E 2757
574E 2758
574E 2759 0F6B' CD
5751 2760
5751 2761
5751 2762
5751 2763 04'2E'21
5754 2764 04'2B'11
5757 2765 118E' CD
575A 2766
575A 2767 0440'3A
575D 2768 01B4'32
5760 2769 04'2B'21
5763 2770 1478' CD
5766 2771
5766 2772 04'2A'21
5769 2773 15C2' CD
576C 2774 1481' CD
576F 2775
576F 2776 119C' CD
5772 2777
5772 2778 04'0A'21
5775 2779 124A' CD
5778 2780 11B8' CD
577B 2781
577B 2782 C9
577C 2783
577C 2784
577C 2785
577C 2786
577C 2787
577C 2788
577C 2789
577C 2790
577C 2791
577C 2792
577C 2793 0F6B' CD
577F 2794
577F 2795
577F 2796
577F 2797 0440'3A
5782 2798 1435' CD
5785 2799
5785 2800 119C' CD
5788 2801
5788 2802 02'94'21
578B 2803 124A' CD
578E 2804 11B8' CD
5791 2805
5791 2806 C9
5792 2807
5792 2808

```

```

*****
:
: EXAM_WRK
: THIS ROUTINE EXAMINES 2901 WORKING REGS
:
*****

```

```

EXAM_WRK:      CALL      INPUT_NUM_CHECK      ;CHECK FOR INPUT,REPORT ERR
:              ; IF NO INPUT, ELSE LOAD
:              ; INPUT_NUM WITH WR NUMBER

                LXI      H,S_ED_WRK          ;RESET THE INSTRUCTION TO
                LXI      D,X_ED_WRK+1        ;NO MODIFIED BITS
                CALL     MOVER_3_R

                LDA      INPUT_NUM+3        ;MAKE MOVE WR[X] TO WR [0]
                STA      A_ADDRS
                LXI      H,X_ED_WRK+1
                CALL     MAKE_A

                LXI      H,X_ED_WRK
                CALL     PERFORM_CSR_R      ;MOV REG TO READ REG
                CALL     READ_DATA_32_R     ;READ 32 BITS OF DATA

                CALL     MOV_DATA0_HEXBUF   ;MOVE DATA TO HEX PRINT AREA

                LXI      H,WRK_A
                CALL     PRINT_STRING_R
                CALL     PRINT_HEX_4        ;PRINT THE DATA

                RET

```

```

*****
:
: EXAM_LS
: THIS ROUTINE EXAMINES LS LOCATIONS
:
*****

```

```

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

                LDA      INPUT_NUM+3        ;ELSE GET ADDRESS
                CALL     READ_CS_R          ;READ THE LS LOCATION

                CALL     MOV_DATA0_HEXBUF   ;MOVE DATA TO HEX PRINT AREA

                LXI      H,LS_A
                CALL     PRINT_STRING_R      ;PRINT STRING
                CALL     PRINT_HEX_4        ;PRINT THE DATA

                RET

```

```

ZZ-ENKAA-2.1 4000 4
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
579B 2819
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
57A4 2832
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
57AD 2844
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
57B6 2856
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
57BB 2868 00AC'3A

```

```

*****
:
: EXAM_MCT
: THIS ROUTINE EXAMINES MEMORY CONTROL + STATUS WORDS
:
*****

```

```

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

```

```

*****
:
: EXAM_UBS
: THIS ROUTINE EXAMINES UNIBUS TRANSLATION BUFFER
:
*****

```

```

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

```

```

*****
:
: EXAM_TB
: THIS ROUTINE EXAMINES TRANSLATION BUFFER WORDS
:
*****

```

```

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

```

```

*****
:
: EXAM_MM
: THIS ROUTINE EXAMINES MAIN MEMORY
:
*****

```

```

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

```

```

*****
:
: EXAM_OS
: THIS ROUTINE EXAMINES OS CPU REG
:
*****

```

```

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

```

```

ZZ-ENKAA-2.1 4000 4
57BE 2869 005B'32
57C1 2870
57C1 2871 03'07'21
57C4 2872 124A'CD
57C7 2873 11C2'C3
57CA 2874
57CA 2875
57CA 2876
57CA 2877
57CA 2878
57CA 2879
57CA 2880
57CA 2881
57CA 2882
57CA 2883 0F6B'CD
57CD 2884
57CD 2885
57CD 2886
57CD 2887 043F'2A
57D0 2888 006C'22
57D3 2889 109D'CD
57D6 2890
57D6 2891 04'02'21
57D9 2892 124A'CD
57DC 2893
57DC 2894 0FC1'CD
57DF 2895
57DF 2896 C9
57E0 2897
57E0 2898
57E0 2899
57E0 2900
57E0 2901
57E0 2902
57E0 2903
57E0 2904
57E0 2905
57E0 2906
57E0 2907 00 5A 21
57E3 2908 02'84'11
57E6 2909 0COD'C3
57E9 2910
57E9 2911
57E9 2912
57E9 2913
57E9 2914
57E9 2915
57E9 2916
57E9 2917
57E9 2918
57E9 2919
57E9 2920 00 5B 21
57EC 2921 02'89'11
57EF 2922 0COD'C3
57F2 2923
57F2 2924
57F2 2925
57F2 2926
57F2 2927
57F2 2928

```

```

K 6
Fiche 1 Frame K6 Sequence 75
STA HEX_BUF
LXI H,OS_A
CALL PRINT_STRING_R
JMP PRINT_HEX_1_R ;PRINT THE DATA

```

```

:*****
: EXAM_WCS
: THIS ROUTINE EXAMINES WCS LOCATIONS
:*****

```

```

EXAM_WCS: CALL INPUT_NUM_CHECK ;CHECK FOR INPUT,REPORT ERR
; IF NO INPUT, ELSE LOAD
; INPUT_NUM WITH WCS ADDR
LHLD INPUT_NUM+2 ;ELSE GET ADDRESS
SHLD WCS_ADDRESS
CALL READ_WCS_R ;READ THE DATA
LXI H,WCS_A
CALL PRINT_STRING_R
CALL PRINT_CSR_VAL ;PRINT THE DATA
RET

```

```

:*****
: EXAM_ICSR
: THIS ROUTINE EXAMINES THE IDC CSR
:*****

```

```

EXAM_ICSR: LXI H,WCS_EX_ICSR
LXI D,ICSR_A
JMP EXAM_SOB_NOADD

```

```

:*****
: EXAM_IDAR
: THIS ROUTINE EXAMINES THE IDC DISK ADDRESS REGISTER
:*****

```

```

EXAM_IDAR: LXI H,WCS_EX_IDAR
LXI D,IDAR_A
JMP EXAM_SOB_NOADD

```

```

:*****
: EXAM_DBUF

```

```

ZZ-ENKAA-2.1 4C00 4
57F2 2929
57F2 2930
57F2 2931
57F2 2932
57F2 2933 00 5C 21
57F5 2934 02'14'11
57F8 2935 0C0D'C3
57FB 2936
57FB 2937
57FB 2938
57FB 2939
57FB 2940
57FB 2941
57FB 2942
57FB 2943
57FB 2944
57FB 2945
57FB 2946 00 5D 21
57FE 2947 03'16'11
5801 2948 0C0D'C3
5804 2949
5804 2950
5804 2951
5804 2952
5804 2953
5804 2954
5804 2955
5804 2956
5804 2957
5804 2958
5804 2959 00 5E 21
5807 2960 03'1B'11
580A 2961 0C0D'C3
580D 2962
580D 2963
580D 2964
580D 2965
580D 2966
580D 2967
580D 2968
580D 2969
580D 2970
580D 2971
580D 2972
580D 2973
580D 2974
580D 2975
580D 2976
580D 2977
580D 2978
580D 2979 02F1'32 3C AF
5812 2980
5812 2981 11 06
5814 2982 0F5E'CD
5817 2983
5817 2984
5817 2985 0C36'CD
581A 2986
581A 2987 0F6E'CD
581D 2988 0239'2A

```

```

: THIS ROUTINE EXAMINES THE IDC DATA BUFFER
:
:*****

```

```

EXAM_DBUF:  LXI  H,WCS_EX_DBUF
            LXI  D,DBUF_A
            JMP  EXAM_S0B_NOADD

```

```

:*****
: EXAM_PATT
: THIS ROUTINE EXAMINES THE IDC ECC PATTERN REGISTER
:*****

```

```

EXAM_PATT:  LXI  H,WCS_EX_PATT
            LXI  D,PATT_A
            JMP  EXAM_S0B_NOADD

```

```

:*****
: EXAM_POSIT
: THIS ROUTINE EXAMINES THE IDC ECC POSITION REGISTER
:*****

```

```

EXAM_POSIT: LXI  H,WCS_EX_POSIT
            LXI  D,POSIT_A
            JMP  EXAM_S0B_NOADD

```

```

:*****
: EXAM_SUB
: THIS ROUTINE IS CALLED BY EXAMINE ROUTINES THAT WANT TO CALL A
: SUBROUTINE IN THE WCS WITH PARAMETER OF ADDRESS IN LS 5. THE
: EXAM_SUB_NOADD ENTRY IS FOR EXAMINES THAT DO NOT USE AN ADDRESS

```

```

: INPUT CONDITIONS
: H+L = ADDRESS OF WCS SUBROUTINE
: D+E = ADDRESS OF STRING PRINTOUT (IMPLIED 4 CHAR LENGTH)
: INPUT_NUM = ADDRESS TO EXAMINE (IF APPLICABLE)
:*****

```

```

EXAM_SUB_NOADD: SET  NOADD_FLG          ;SET FLAG, NO ADDRESS REQUIRED

```

```

EXAM_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
               LHLD EXAM_TEMP

```

```

ZZ-ENKAA-2.1 4C00 4
5820 2989 05 0E
5822 2990 1243'CD
5825 2991 13CE'CD
5828 2992 1A7A'CD
582B 2993 06 3E
582D 2994 1435'CD
5830 2995 119C'CD
5833 2996 11B8'C3
5836 2997
5836 2998
5836 2999
5836 3000
5836 3001
5836 3002
5836 3003
5836 3004
5836 3005
5836 3006
5836 3007
5836 3008 7C
5837 3009 65
5838 3010 6F
5839 3011 03A1'22
583C 3012
583C 3013 EB
583D 3014 0239'22
5840 3015
5840 3016 0F 02F1'3A
                    0C52'DA
                    *
5847 3017 0FCF'CD
                    *
584A 3018
                    *
584A 3019
                    *
584A 3020 1193'CD
                    *
584D 3021
                    *
584D 3022 05 3E
                    *
584F 3023 1415'CD
                    *
5852 3024
                    *
5852 3025
                    *
5852 3026 02F1'32 AF
                    *
5856 3027 C9
5857 3028
5857 3029
5857 3030
5857 3031
5857 3032
5857 3033
5857 3034
5857 3035
5857 3036 06'5F'21
585A 3037 1FDB'C3
585D 3038
585D 3039
585D 3040
585D 3041
585D 3042
585D 3043
585D 3044
585D 3045
585D 3046
585D 3047

```

```

MVI C,5
CALL PRINTER
CALL START_CPU ;START THE SUBROUTINE
CALL SUB_ATTENTION ;WAIT FOR EXIT
MVI A,LS_6 ;READ RETURNED DATA FROM LS 6
CALL READ_LS_R
CALL MOV_DATA0_HEXBUF ;MOVE DATA TO HEX PRINT AREA
JMP PRINT_HEX_4 ;PRINT THE DATA

```

```

*****
:
: EXAM_DEP_SUB
:
: THIS ROUTINE SETS UP THE ADDRESS FOR EXAMINE AND DEPOSIT ROUTINES THAT
: USE WCS SUBROUTINES. IT ALSO CHECKS FOR ERRORS IN THE COMMAND.
:
*****

```

```

EXAM_DEP_SUB: MOV A,H ;GET STARTING ADDRESS
              MOV H,L
              MOV L,A
              SHLD STARTING_UPC
              XCHG
              SHLD EXAM_TEMP ;GET LABEL IF EXAMINE
              IFN NOADD_FLG ;SEE IF NEED ADDRESS
              CALL INPUT_NUM_IF ;GET ADDRESS INPUT, SET
                                ; ERROR FLG IF NO INPUT
              CALL MOV_INPUT_DATA0 ;MOVE 4 BYTES OF INPUT NUM
                                ; TO DATA0 THROUGH DATA3
              MVI A,LS_5
              CALL WRITE_LS_R ;PUT ADDRESS IN LS 5
              ENDF
              CLEAR NOADD_FLG ;CLEAR FLAG IF SET
              RET ;DONE

```

```

*****
:
: DEP_COMMAND
:
*****

```

```

DEP_COMMAND: LXI H,DEP_TAB ;POINTER TO PARSE TABLE
              JMP PAR_TEMP_BUF

```

```

*****
:
: DEP_RAM
: THIS ROUTINE DEPOSITS THE RAM OF THE 8085 CONSOLE WITH THE DATA AT THE
: ADDRESS GIVEN
:
*****

```

```

ZZ-ENKAA-2.1 4C00 4
585D 3048 0F6B'CD
5860 3049
5860 3050
5860 3051
5860 3052 043F'3A
5863 3053 67
5864 3054 0440'3A
5867 3055 6F
5868 3056 03C6'22
586B 3057
586B 3058 0F6B'CD
586E 3059
586E 3060
586E 3061
586E 3062 03C6'2A
5871 3063 0440'3A
5874 3064 77
5875 3065
5875 3066 C9
5876 3067
5876 3068
5876 3069
5876 3070
5876 3071
5876 3072
5876 3073
5876 3074
5876 3075
5876 3076
5876 3077 0F6B'CD
5879 3078
5879 3079
5879 3080
5879 3081 04'3E'21
587C 3082 00'B8'11
587F 3083 118E'CD
5882 3084
5882 3085 0064'32 AF
5886 3086
5886 3087 1116'C3
5889 3088
5889 3089
5889 3090
5889 3091
5889 3092
5889 3093
5889 3094
5889 3095
5889 3096
5889 3097 0F6B'CD
588C 3098
588C 3099
588C 3100
588C 3101 043F'2A
588F 3102 00C2'22
5892 3103
5892 3104 1148'C3
5895 3105
5895 3106
5895 3107

```

N 6

fiche 1 Frame N6 Sequence 78

```

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

LDA INPUT_NUM+2 ;NUMBER IS DEP ADDRESS
MOV H,A
LDA INPUT_NUM+3
MOV L,A
SHLD TEMP_WORD

CALL INPUT_NUM_CHECK ;CHECK FOR INPUT,REPORT ERR
; IF NO INPUT, ELSE LOAD
; INPUT_NUM WITH RAM DATA

LHLD TEMP_WORD ;POINT TO TEMP WORD
LDA INPUT_NUM+3 ;GET DATA
MOV M,A ;DEP DATA

RET

```

```

:*****
:
: DEP_CSR
: THIS ROUTINE DEPOSITS THE CSR.
:*****

```

```

DEP_CSR: CALL INPUT_NUM_CHECK ;CHECK FOR INPUT,REPORT ERR
; IF NO INPUT, ELSE LOAD
; INPUT_NUM WITH CSR DATA

LXI H,INPUT_NUM+1 ;MOVE TO CSR LOAD AREA
LXI D,CSR_VALUE
CALL MOVER_3_R

CLEAR PARITY_ON ;DONT WRITE PARITY BIT
;ON DEPOSITS
JMP LOAD_CSR_R ;PUT THE VALUE IN CSR

```

```

:*****
:
: DEP_UPC
: THIS ROUTINE DEPOSITS THE UPC.
:*****

```

```

DEP_UPC: CALL INPUT_NUM_CHECK ;CHECK FOR INPUT,REPORT ERR
; IF NO INPUT, ELSE LOAD
; INPUT_NUM WITH UPC DATA

LHLD INPUT_NUM+2 ;POINT TO DATA
SHLD UPC_VALUE ;MOVE VALUE TO CSR LOAD AREA

JMP LOAD_UPC_R ;PUT VALUE IN CSR

```

```

:*****
:

```

```

22-ENKAA-2.1 4C00 4
5895 3108
5895 3109
5895 3110
5895 3111
5895 3112
5895 3113 0F6B'CD
5898 3114
5898 3115
5898 3116
5898 3117 0440'3A
5898 3118 03C5'32
589E 3119
589E 3120 0F6B'CD
58A1 3121
58A1 3122
58A1 3123
58A1 3124 04'2E'21
58A4 3125 04'2B'11
58A7 3126 118E'CD
58AA 3127
58AA 3128 1193'CD
58AD 3129
58AD 3130
58AD 3131 03C5'3A
58B0 3132 01CC'32
58B3 3133 04'2B'21
58B6 3134 1462'CD
58B9 3135
58B9 3136 149E'CD
58BC 3137 04'2A'21
58BF 3138 15C2'C3
58C2 3139
58C2 3140
58C2 3141
58C2 3142
58C2 3143
58C2 3144
58C2 3145
58C2 3146
58C2 3147
58C2 3148 0F6B'CD
58C5 3149
58C5 3150
58C5 3151
58C5 3152 043F'2A
58C8 3153
58C8 3154 006C'22
58CB 3155
58CB 3156 0F6B'CD
58CE 3157
58CE 3158
58CE 3159
58CE 3160 04'3E'21
58D1 3161 00'B4'11
58D4 3162 118E'CD
58D7 3163
58D7 3164 0064'32 AF
58DB 3165 10B9'C3
58DE 3166
58DE 3167

```

```

: DEP WRK
: THIS ROUTINE DEPOSITS 2901 WORKING REGS
:*****
DEP_WRK:      CALL      INPUT_NUM_CHECK      ;CHECK FOR INPUT,REPORT ERR
:              ; IF NO INPUT, ELSE LOAD
:              ; INPUT_NUM WITH WR NUMBER

              LDA      INPUT_NUM+3          ;GET WRK REG NUMBER
              STA      TEMP_BYTE            ;SAVE WORKING REG TO DEP

              CALL     INPUT_NUM_CHECK      ;CHECK FOR INPUT,REPORT ERR
:              ; IF NO INPUT, ELSE LOAD
:              ; INPUT_NUM WITH WR DATA

              LXI     H,S_ED_WRK           ;RESET THE INSTRUCTION TO
              LXI     D,X_ED_WRK+1        ;NO MODIFIED BITS
              CALL    MOVER_3_R

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

              LDA     TEMP_BYTE
              STA     B_ADDR5              ;MOV WORKING REG 0 TO REG X
              LXI     H,X_ED_WRK+1
              CALL    MAKE_B               ;MODIFY INST TO BE MOVE TO X0

              CALL    WRITE_DATA_32_R     ;WRITE THE DATA TO 2901
              LXI     H,X_ED_WRK
              JMP     PERFORM_CSR_R       ;MOV REG TO READ REG

:*****
: DEP WCS
: THIS ROUTINE DEPOSITS WCS LOCATIONS
:*****
DEP_WCS:      CALL     INPUT_NUM_CHECK     ;CHECK FOR INPUT,REPORT ERR
:              ; IF NO INPUT, ELSE LOAD
:              ; INPUT_NUM WITH WCS ADDRESS

              LHLD   INPUT_NUM+2          ;MOVE FIRST NUM TO WCS
:              ; ADDRESS

              SHLD   WCS_ADDRESS

              CALL   INPUT_NUM_CHECK     ;CHECK FOR INPUT,REPORT ERR
:              ; IF NO INPUT, ELSE LOAD
:              ; INPUT_NUM WITH WCS DATA

              LXI   H,INPUT_NUM+1        ;MOVE SECOND NUMBER TO
              LXI   D,WCS_VALUE          ;WCS_VALUE
              CALL  MOVER_3_R

              CLEAR  PARITY_ON           ;DON'T CALC PARITY ON WRITE
              JMP   WRITE_WCS_R         ;WRITE SECOND NUM TO WCS

:*****

```

58DE 3168  
 58DE 3169  
 58DE 3170  
 58DE 3171  
 58DE 3172  
 58DE 3173  
 58DE 3174 0F6B'CD  
 58E1 3175  
 58E1 3176  
 58E1 3177  
 58E1 3178 0440'3A  
 58E4 3179 00AC'32  
 58E7 3180  
 58E7 3181 7C 3E  
 58E9 3182 1415'C3  
 58EC 3183  
 58EC 3184  
 58EC 3185  
 58EC 3186  
 58EC 3187  
 58EC 3188  
 58EC 3189  
 58EC 3190  
 58EC 3191  
 58EC 3192 0F6B'CD  
 58EF 3193  
 58EF 3194  
 58EF 3195  
 58EF 3196 0440'3A  
 58F2 3197 03C5'32  
 58F5 3198  
 58F5 3199 0F6B'CD  
 58F8 3200  
 58F8 3201  
 58F8 3202  
 58F8 3203 1193'CD  
 58FB 3204  
 58FB 3205  
 58FB 3206 03C5'3A  
 58FE 3207 1415'C3  
 5901 3208  
 5901 3209  
 5901 3210  
 5901 3211  
 5901 3212  
 5901 3213  
 5901 3214  
 5901 3215  
 5901 3216  
 5901 3217 00 50 21  
 5904 3218 0D30'C3  
 5907 3219  
 5907 3220  
 5907 3221  
 5907 3222  
 5907 3223  
 5907 3224  
 5907 3225  
 5907 3226  
 5907 3227

```

: DEP OS
: THIS ROUTINE DEPOSITS THE GIVEN DATA INTO OS
:
:*****
    
```

```

DEP_OS:      CALL      INPUT_NUM_CHECK      ;CHECK FOR INPUT,REPORT ERR
:             ; IF NO INPUT, ELSE LOAD
:             ; INPUT_NUM WITH OS DATA

             LDA      INPUT_NUM+3          ;GET 8 BITS AND MOVE TO
             STA      DATA3              ;WRITE AREA

             MVI      A,OS_ADD            ;OS ADDRESS IN LS ADDRESS SPACE
             JMP      WRITE_LS_R
    
```

```

:*****
: DEP LS
: THIS ROUTINE DEPOSITS DATA INTO LS AT THE GIVEN ADDRESS
:
:*****
    
```

```

DEP_LS:      CALL      INPUT_NUM_CHECK      ;CHECK FOR INPUT,REPORT ERR
:             ; IF NO INPUT, ELSE LOAD
:             ; INPUT_NUM WITH LS ADDRESS

             LDA      INPUT_NUM+3          ;GET LS ADDRESS AND SAVE
             STA      TEMP_BYTE

             CALL     INPUT_NUM_CHECK      ;CHECK FOR INPUT,REPORT ERR
:             ; IF NO INPUT, ELSE LOAD
:             ; INPUT_NUM WITH LS DATA

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

             LDA      TEMP_BYTE           ;GET ADDRESS AGAIN
             JMP      WRITE_LS_R         ;AND PERFORM WRITE
    
```

```

:*****
: DEP MCT
: THIS ROUTINE DEPOSITS MEMORY CONTROL + STATUS
:
:*****
    
```

```

DEP_MCT:     LXI      H,WCS_DE_MCT
             JMP      DEP_SOB
    
```

```

:*****
: DEP MM
: THIS ROUTINE DEPOSITS MAIN MEMORY
:
:*****
    
```

```

ZZ-ENKAA-2.1 4C00 4
5907 3228 00 54 21
590A 3229 0D30'C3
590D 3230
590D 3231
590D 3232
590D 3233
590D 3234
590D 3235
590D 3236
590D 3237
590D 3238
590D 3239 00 52 21
5910 3240 0D30'C3
5913 3241
5913 3242
5913 3243
5913 3244
5913 3245
5913 3246
5913 3247
5913 3248
5913 3249
5913 3250 00 58 21
5916 3251 0D30'C3
5919 3252
5919 3253
5919 3254
5919 3255
5919 3256
5919 3257
5919 3258
5919 3259
5919 3260
5919 3261 00 5F 21
591C 3262 0D2B'C3
591F 3263
591F 3264
591F 3265
591F 3266
591F 3267
591F 3268
591F 3269
591F 3270
591F 3271
591F 3272 00 60 21
5922 3273 0D2B'C3
5925 3274
5925 3275
5925 3276
5925 3277
5925 3278
5925 3279
5925 3280
5925 3281
5925 3282
5925 3283 00 61 21
5928 3284 0D2B'C3
5928 3285
5928 3286
5928 3287

```

```

DEP_MM:      LXI      H,WCS_DE_MM
              JMP      DEP_SOB

```

```

:*****
:
: DEP_TB
: THIS ROUTINE DEPOSITS TRANSLATION BUFFER LOCATIONS
:*****

```

```

DEP_TB:      LXI      H,WCS_DE_TB
              JMP      DEP_SOB

```

```

:*****
:
: DEP_UBS
: THIS ROUTINE DEPOSITS UNIBUS TRANSLATION BUFFER LOCATIONS
:*****

```

```

DEP_UBS:     LXI      H,WCS_DE_UBS
              JMP      DEP_SOB

```

```

:*****
:
: DEP_ICSR
: THIS ROUTINE DEPOSITS THE IDC CSR
:*****

```

```

DEP_ICSR:    LXI      H,WCS_DE_ICSR
              JMP      DEP_SOB_NOADD

```

```

:*****
:
: DEP_IDAR
: THIS ROUTINE DEPOSITS THE IDC DISK ADDRESS REGISTER
:*****

```

```

DEP_IDAR:    LXI      H,WCS_DE_IDAR
              JMP      DEP_SOB_NOADD

```

```

:*****
:
: DEP_DBUF
: THIS ROUTINE DEPOSITS THEN IDC DATA BUFFER
:*****

```

```

DEP_DBUF:    LXI      H,WCS_DE_DBUF
              JMP      DEP_SOB_NOADD

```

```

ZZ-ENKAA-2.1 4C00 4
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
5938 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 IS ;JUMP IF NOT S/U 0
LDA INPUT_NUM+3 ;GET LOW BYTE OF ADDRESS
ORA A ;SET CONDITION CODES

```

```

ZZ-ENKAA-2.1 4C00 4
595E 3348 0705'CA
5961 3349
5961 3350
5961 3351 0440'3A
5964 3352 6F
5965 3353 E9
5966 3354
5966 3355
5966 3356
5966 3357
5966 3358
5966 3359
5966 3360
5966 3361
5966 3362
5966 3363
5966 3364
5966 3365
5966 3366
5966 3367
5966 3368 037C'32 3C AF
596B 3369 0D72'C3
596E 3370
596E 3371 037C'32 AF
5972 3372
5972 3373 02F3'32 3C AF
5977 3374 1063'CD
597A 3375
597A 3376 0FCF'CD
597D 3377
597D 3378 31 06
597F 3379
597F 3380 0F70'CD
5982 3381
5982 3382 04'3D'21
5985 3383 04'0F'11
5988 3384 11A2'CD
5988 3385
5988 3386 0FCF'CD
598E 3387
598E 3388 32 06
5990 3389
5990 3390 0F70'CD
5993 3391
5993 3392 40FE 32 3C AF
5998 3393
5998 3394 131B'CD
5998 3395
5998 3396
5998 3397 30 06
599D 3398 4082 3A
59A0 3399 B7
59A1 3400 0F7E'C2
59A4 3401
59A4 3402
59A4 3403 80 E6 043D'3A
                    *****
                    *
                    *
                    *
                    *****
59AC 3404 0E75'CD
59AF 3405
59AF 3406 0DB5'C3

```

F 7

Fiche 1 Frame F7 Sequence 83

```

JZ STNDRD_CON_COM ;GO TO ROUTINE THAT RESETS
; INTERVAL TIMER AND GOES TO
; ADDRESS 0 IF S/U 0
LDA INPUT_NUM+3
MOV L,A
PCHL ;MOVE NUMBER TO PC(BRANCH)

```

18:

```

*****
X COMMAND
THIS COMMAND WILL INPUT OR OUTPUT BINARY DATA FROM CONSOLE RAM OR
THE WCS. THE SPECIFICATION FOR THIS COMMAND IS IN THE MID RANGE
CONSOLE SPEC...
NOTE: FOR PURPOSES OF THE MICRO MONITOR IF THE ADDRESS 4100 IS GIVEN
AND A POSITIVE NUMBER (MEANING DEPOSIT TO CONSOLE RAM) THE PROGRAM
WILL BRANCH TO THE ROM TO HAVE THE COMMAND EXECUTED
*****

```

```

XU: SET SLASH_U ;SLASH U COMMAND TYPED
JMP X_COMMAND

XC: CLEAR SLASH_U ;SLASH P COMMAND TYPED

X_COMMAND: SET NO_RETURN ;DON'T RETURN IF ERROR
CALL SKIP_TO_SPEC ;SKIP TO ADDRESS FIELD

CALL INPUT_NUM_IF ;GET ADDRESS

MVI B,<^X31> ;ERROR CODE IF ERROR SET
; (NO X COMMAND ADDRESS)
CALL CHECK_ERR_FLG ;SEE IF ERROR, RETURN IF NOT

LXI H,INPUT_NUM
LXI D,X_COM_ADDR
CALL MOVER_4_R ;SAVE ADDRESS FIELD

CALL INPUT_NUM_IF ;GET COUNT

MVI B,<^X32> ;ERROR CODE IF ERROR SET
; (NO X COMMAND COUNT)
CALL CHECK_ERR_FLG ;SEE IF ERROR, RETURN IF NOT

SET CTLDIS ;DISABLE CONTROL CHARACTER
; CHECK
CALL INPUT_CHAR_IF ;GET THE CHECKSUM (WAITS IN
; ROUTINE TIL CHAR RECEIVED)

MVI B,<^X30> ;ERROR CODE IF CHECKSUM ERR
LDA BYTSUM ;GET CHECKSUM OF LINE
ORA A ;SET CONDITION CODES
JNZ ERROR_ROUTINE ;REPORT CHECKSUM ERROR IF
; BYTSUM IS NOT 0

IFMBI INPUT_NUM,NEG_CNT ;ELSE SEE IF WRITE OR READ

CALL OUT_BIN_STRING ;COUNT IS NEG ..OUTPUT DATA

ELSE

```

```

ZZ-ENKAA-2.1 4C00 4
59B2 3407
59B2 3408 ODC0'CD
59B5 3409
59B5 3410
59B5 3411 AF
59B6 3412 40FE 32
59B9 3413 40FF 32
59BC 3414 02F3'32
59BF 3415 C9
59C0 3416
59C0 3417
59C0 3418
59C0 3419
59C0 3420
59C0 3421
59C0 3422
59C0 3423
59C0 3424
59C0 3425
59C0 3426
59C0 3427 41 FE 0411'3A
ODD7'C2
59C8 3428
59C8 3429 00 FE 0412'3A
ODD7'C2
59D0 3430 037C'3A
59D3 3431 B7
59D4 3432 000B C2
59D7 3433
59D7 3434
59D7 3435
59D7 3436
59D7 3437
59D7 3438 0411'2A
59DA 3439 00C2'22
59DD 3440 1148'CD
59E0 3441
59E0 3442 10F9'CD
59E3 3443
59E3 3444 0409'32 3C AF
59E8 3445
59E8 3446 1208'CD
59E8 3447 4082 32 AF
59EF 3448
59EF 3449 0F 037C'3A
OE16'D2
59F6 3450
59F6 3451 4145 CD
59F9 3452 0DF6'D2
59FC 3453
59FC 3454 04'11'21
59FF 3455 56
5A00 3456 23
5A01 3457 5E
5A02 3458 EB
5A03 3459
5A03 3460 77
5A04 3461
5A04 3462 04'11'21
5A07 3463 1581'CD

```

```

CALL IN_BIN_STRING ;COUNT IS POS ..INPUT DATA
ENDIF
XRA A ;CLEAR FLAGS
STA CTLDIS ;REENABLE CNTL CHAR CHECK
STA NOCHK ;REENABLE CR ON OUTPUT TO TERM
STA NO_RETURN ;INIT NO RETURN ON ERROR FLAG
RET

```

```

*****
IN BIN STRING
THIS ROUTINE IS USED TO INPUT A BINARY STRING FOR THE X COMMAND
THE ADDRESS MUST BE IN X_COM_ADDR5
THE COUNT MUST BE IN INPOT_NOM
*****

```

```

***** IN_BIN_STRING: IFEQI X_COM_ADDR5+2,HEX_41 ;SEE IF DESTINATION ADDRESS IS
; 4100(H)
IFEQI X_COM_ADDR5+3,0
LDA SLASH_U ;SEE IF DESTINATION IS 8085 RAM
ORA A ;SET CONDITION CODES
JNZ ROM_X_ADDR5 ;GO TO ROM CODE IF DEST IS 8085
; RAM SINCE LOAD WILL OVERLAY
; MICMON
ENDIF
ENDIF
LHLD X_COM_ADDR5+2 ;GET ADDRESS TO WRITE
SHLD UPC_VALUE
CALL LOAD_UPC_R ;LOAD STARTING UPC, IN CASE
; THIS IS A WCS LOAD
; KEEP MACHINE IDLE.
CALL NOP_CSR_R
SET WCS_PNT ;SET TO FIRST BYTE OF 3
;BYTES OF WCS WORDS
CALL PRINT_PROMPT ;ELSE PRINT PROMPT FOR DATA
CLEAR BYTSUM ;START NEW CHECKSUM
IF SLASH_U ;DO DEPOSIT TO RAM IF X/U
CALL GSVECT ;GET NEXT CHAR
JNC 1$ ;LOOP UNTIL CHAR RECEIVED
LXI H,X_COM_ADDR5+2 ;POINT TO RAM ADDRESS
MOV D,M ;STORE HIGH BYTE IN D
INX H ;POINT TO LOW BYTE
MOV E,M ;STORE LOW BYTE IN E
XCHG ;LOAD HL WITH DE
MOV M,A ;WRITE CHAR INTO RAM
LXI H,X_COM_ADDR5+2 ;INCREMENT ADDRESS
CALL INC_WORD_R

```

1\$:

```

ZZ-ENKAA-2.1 4C00 4
SA0A 3464
SA0A 3465 04'3F'21
SA0D 3466 158C'CD
SA10 3467 0DF6'C2
SA13 3468
SA13 3469 0E65'C3
SA16 3470
SA16 3471 03 FE 0409'3A
      OE35'C2
SA1E 3472 4145 CD
SA21 3473 0E1E'D2
SA24 3474 0A D3
SA26 3475
SA26 3476 37 D3
SA28 3477 36 D3
SA2A 3478
SA2A 3479 27 D3
SA2C 3480 26 D3
SA2E 3481
SA2E 3482 0409'32 AF
SA32 3483 0E58'C3
SA35 3484
SA35 3485
SA35 3486 01 FE 0409'3A
      OE48'C2
SA3D 3487 4145 CD
SA40 3488 0E3D'D2
SA43 3489 08 D3
SA45 3490 0E58'C3
SA48 3491
SA48 3492
SA48 3493 02 FE 0409'3A
      OE58'C2
SA50 3494 4145 CD
SA53 3495 0E50'D2
SA56 3496 09 D3
SA58 3497
SA58 3498
SA58 3499 04'09'21
SA5B 3500 34
SA5C 3501
SA5C 3502 04'3F'21
SA5F 3503 158C'CD
SA62 3504 0E16'C2
SA65 3505
SA65 3506
SA65 3507
SA65 3508 4145 CD
SA68 3509 0E65'D2
SA68 3510
SA68 3511 30 06
SA6D 3512 4082 3A
SA70 3513 B7
SA71 3514 0F7E'C2
SA74 3515
SA74 3516
SA74 3517 C9
SA75 3518
SA75 3519
SA75 3520

```

H 7

Fiche 1 Frame H7

Sequence 85

```

LXI H, INPUT_NUM+2
CALL DECR_WORD_R      ;DECREMENT COUNT
JNZ 1$                ;LOOP UNTIL COUNT=0

ELSE                  ;DO DEPOSIT TO WCS IF X/C

IFEQI WCS_PNT,3
CALL GSVECT           ;GET NEXT CHAR
JNC 7$                ;LOOP UNTIL CHAR RECEIVED
OUT WCSB2             ;WRITE THIRD BYTE IN REG

OUT SETWCK            ;WRITE WCS
OUT CLRWCK

OUT SETUPK           ;INCREMENT UPC TO NEXT ADDRESS
OUT CLRUPK

CLEAR WCS_PNT        ;RESET TO FIRST BYTE
JMP 3$                ;SKIP OTHER CHECKS
ENDIF

IFEQI WCS_PNT,1
CALL GSVECT           ;GET NEXT CHAR
JNC 6$                ;LOOP UNTIL CHAR RECEIVED
OUT WCSB0             ;WRITE FIRST BYTE IN REG
JMP 3$                ;SKIP OTHER CHECKS
ENDIF

IFEQI WCS_PNT,2
CALL GSVECT           ;GET NEXT CHAR
JNC 5$                ;LOOP UNTIL CHAR RECEIVED
OUT WCSB1             ;WRITE SECOND BYTE IN REG

LXI H,WCS_PNT        ;INC POINTER TO NEXT BYTE
INR M

LXI H, INPUT_NUM+2
CALL DECR_WORD_R      ;DECREMENT COUNT
JNZ 2$                ;LOOP UNTIL COUNT=0

ENDIF

CALL GSVECT           ;GET NEXT CHAR (CHECKSUM)
JNC 4$                ;LOOP UNTIL CHAR RECEIVED

MVI B,<^X30>          ;ERROR CODE IF CHECKSUM ERR
LDA BYTSUM            ;GET CHECKSUM OF DATA
ORA A                 ;SET CONDITION CODES
JNZ ERROR_ROUTINE    ;REPORT CHECKSUM ERROR IF
                     ; BYTSUM IS NOT 0

RET

```

;.....

```

SA75 3521
SA75 3522
SA75 3523
SA75 3524
SA75 3525
SA75 3526
SA75 3527
SA75 3528
SA75 3529 1208'CD
SA78 3530
SA78 3531 AF
SA79 3532 4082 32
SA7C 3533 40FE 32
SA7F 3534
SA7F 3535 3C
SA80 3536 40FF 32
SA83 3537 0409'32
SA86 3538
SA86 3539
SA86 3540 0F 037C'3A
          0E93'D2
SA8D 3541 0EA7'CD
SA90 3542
SA90 3543 0E96'C3
SA93 3544
SA93 3545 0EB9'CD
SA96 3546
SA96 3547
SA96 3548 04'3F'21
SA99 3549 158C'CD
SA9C 3550 0E86'C2
SA9F 3551
SA9F 3552 4082 3A
SAA2 3553 2F
SAA3 3554 3C
SAA4 3555
SAA4 3556 1265'C3
SAA7 3557
SAA7 3558
SAA7 3559
SAA7 3560
SAA7 3561
SAA7 3562
SAA7 3563
SAA7 3564
SAA7 3565
SAA7 3566
SAA7 3567 0411'3A
SAAA 3568 67
SAA8 3569 0412'3A
SAAE 3570 6F
SAAF 3571 7E
SAB0 3572 1265'CD
SAB3 3573
SAB3 3574 04'11'21
SAB6 3575 1581'C3
SAB9 3576
SAB9 3577
SAB9 3578
SAB9 3579

```

```

:
: OUT BIN STRING
: THIS ROUTINE OUTPUTS A BINARY STRING FOR THE X COMMAND
: THE ADDRESS MUST BE IN X_COM_ADDR8
: THE COUNT MUST BE IN INPUT_NUM
:
:*****
OUT_BIN_STRING: CALL PRINT_PROMPT ;PRINT PROMPT
                  XRA A ;CLERA FLAGS
                  STA BYTSUM ;CLEAR OUTPUT CHECKSUM
                  STA CTLDIS ;CLEAR CONTROL CHAR DISABLE
                  INR A ;SET FLAGS
                  STA NOCHK ;IGNORE <CR>
                  STA WCS_PNT ;SET TO FIRST BYTE OF 3
                  ;BYTES OF WCS WORDS
                  IF SLASH_U
                  CALL X_EX_U ;GET OUTPUT FROM RAM
                  ELSE
                  CALL X_EX_C ;GET OUTPUT FROM WCS
                  ENDIF
                  LXI H,INPUT_NUM+2
                  CALL DECR_WORD_R ;DECREMENT NEG COUNT UNTIL 0
                  JNZ 1$ ;LOOP UNTIL COUNT=0
                  LDA BYTSUM ;CALCULATE CHECKSUM TO ADD
                  CMA ;UP TO ZERO WITH BINARY
                  INR A ;STRING SENT
                  JMP TYPE_CHAR ;SEND CHECKSUM CHAR
:
:*****
X_EX_U
: THIS ROUTINE EXAMINES THE CONSOLE RAM FOR THE X COMMAND
: THE RECIVED BYTE IS SENT TO APT AND THE ADDRESS IS INCREMENTED
:
:*****
X_EX_U: LDA X_COM_ADDR8+2 ;PUT ADDRESS IN H+L
        MOV H,A
        LDA X_COM_ADDR8+3
        MOV L,A
        MOV A,M ;GET CHAR FROM MEMORY
        CALL TYPE_CHAR ;OUTPUT TO APT
        LXI H,X_COM_ADDR8+2 ;INCREMENT ADDRESS
        JMP INC_WORD_R
:
:*****

```

```

ZZ-ENKAA-2.1 4C00 4
5AB9 3580
5AB9 3581
5AB9 3582
5AB9 3583
5AB9 3584
5AB9 3585
5AB9 3586
5AB9 3587
5AB9 3588 01 FE 0409'3A
          0ED4'C2
5AC1 3589
5AC1 3590 0411'2A
5AC4 3591 006C'22
5AC7 3592
5AC7 3593 0064'32 Af
5ACB 3594 109D'CD
5ACE 3595
5ACE 3596 00B4'3A
5AD1 3597 1265'CD
5AD4 3598
5AD4 3599
5AD4 3600 02 FE 0409'3A
          0EE2'C2
5ADC 3601
5ADC 3602 00B5'3A
5ADF 3603 1265'CD
5AE2 3604
5AE2 3605
5AE2 3606 03 FE 0409'3A
          0EFA'C2
5AEA 3607
5AEA 3608 00B6'3A
5AED 3609 1265'CD
5AFO 3610
5AFO 3611 0409'32 Af
5AF4 3612
5AF4 3613 04'11'21
5AF7 3614 1581'CD
5AFA 3615
5AFA 3616
5AFA 3617
5AFA 3618 04'09'21
5AFD 3619 34
5AFE 3620
5AFE 3621 C9
5AFF 3623
5AFF 3624
5AFF 3625
5AFF 3626
5AFF 3627
5AFF 3628
5AFF 3629
5AFF 3630
5AFF 3631
5AFF 3632
5AFF 3633
5AFF 3634
5AFF 3635
5AFF 3636
5AFF 3637

```

```

: X_EX_C
: THIS ROUTINE EXAMINES THE WCS FOR THE X COMMAND
: THE ROUTINE READ THE DATA FROM THE WCS 24 BITS AT A TIME AND GIVES
: ONE BYTE IN PRINT_CHAR EACH TIME IT IS CALLED. THE ADDRESS IS
: INCREMENTED AFTER THE 24 BITS OF DATA (3 BYTES) IS PASSED
:*****

```

```

***** X_EX_C:      IFEQI      WCS_PNT,1
*
*
*          LHL      X_COM_ADDR+2      ;GET ADDRESS TO READ
*          SHLD     WCS_ADDRESS
*
*          CLEAR   PARITY_ON          ;DON'T CALC PARITY ON READ
*          CALL    READ_WCS_R
*
*          LDA     WCS_VALUE          ;GET FIRST BYTE OF WORD
*          CALL    TYPE_CHAR          ;SEND TO APT
*          ENDIF
*****
*          IFEQI   WCS_PNT,2
*
*          LDA     WCS_VALUE+1        ;GET SECOND BYTE OF WORD
*          CALL    TYPE_CHAR          ;SEND TO APT
*          ENDIF
*****
*          IFEQI   WCS_PNT,3
*
*          LDA     WCS_VALUE+2        ;GET THIRD BYTE OF WORD
*          CALL    TYPE_CHAR          ;SEND TO APT
*          CLEAR   WCS_PNT           ;RESET TO FIRST BYTE
*          LXI     H,X_COM_ADDR+2    ;INCREMENT ADDRESS
*          CALL    INC_WORD_R
*          ENDIF
*****
*          LXI     H,WCS_PNT         ;INC POINTER TO NEXT BYTE
*          INR     M
*
*          RET

```

```

:*****
: MISC SUBROUTINES
:*****

```

```

: PARITY CALC
: THIS ROUTINE CALCULATES PARITY FOR THE WCS WORDS
: AND PLACES IT IN BIT 23 OF WCS WORD IF THE PARITY SELECTOR IS SET
: TO BAD PARITY AS USED IN STOP ON MICRO MATCH THE PARITY WILL BE
: CHANGED TO BAD PARITY IN BIT 23 OF CSR_VALUE

```

```

: INPUT CONDITIONS:

```



22-ENKAA-2.1 4C00 4  
 5B41 3695  
 5B41 3696  
 5B41 3697  
 5B41 3698  
 5B41 3699  
 5B41 3700  
 5B41 3701  
 5B41 3702  
 5B41 3703  
 5B41 3704 00 57 21  
 5B44 3705 0A49'C3  
 5B47 3706  
 5B47 3707  
 5B47 3708  
 5B47 3709  
 5B47 3710  
 5B47 3711  
 5B47 3712  
 5B47 3713  
 5B47 3714  
 5B47 3715  
 5B47 3716 02F4'32 3C AF  
 5B4C 3717  
 5B4C 3718 00 56 21  
 5B4F 3719 0A49'C3  
 5B52 3720  
 5B52 3721  
 5B52 3722  
 5B52 3723  
 5B52 3724  
 5B52 3725  
 5B52 3726  
 5B52 3727  
 5B52 3728  
 5B52 3729  
 5B52 3730  
 5B52 3731  
 5B52 3732  
 5B52 3733  
 5B52 3734  
 5B52 3735  
 5B52 3736  
 5B52 3737 043F'2A  
 5B55 3738 006C'22  
 5B58 3739 109D'CD  
 5B58 3740  
 5B58 3741 10B9'C3  
 5B5E 3742  
 5B5E 3743  
 5B5E 3744  
 5B5E 3745  
 5B5E 3746  
 5B5E 3747  
 5B5E 3748  
 5B5E 3749  
 5B5E 3750  
 5B5E 3751  
 5B5E 3752  
 5B5E 3753  
 5B5E 3754

```

.....
: RESTORE_WR
: THIS ROUTINE CALLS A WCS SUBROUTINE THAT RESTORES THE WORKING REGS
: OF THE 2901 FROM A SAVED AREA
:.....

```

```

RESTORE_WR:  LXI  H,WCS_REST_WR      ;ADDRESS OF SUBROUTINE
              JMP  EXEC_SUB        ;DO THE SUBROUTINE

```

```

.....
: SAVE_WR
: THIS ROUTINE CALLS A WCS SUBROUTINE THAT SAVES THE CONTENTS OF THE
: 2901 IN A SPECIAL SAVE AREA SO THEY MAY BE RESTORED LATER.
:.....

```

```

SAVE_WR:     SET  NO_SAVE_UPC_CSR    ;SET FLAG SO SAVED CSR AND
              ;UPC ARE NOT DESTROYED
              LXI  H,WCS_SAVE_WR    ;ADDRESS OF SUBROUTINE
              JMP  EXEC_SUB        ;DO THE SUBROUTINE

```

```

.....
: CHANGE_WCS
: THIS ROUTINE READS A WCS WORD, THEN CALCULATES PARITY ON IT AND
: WRITES IT BACK. THIS IS USED FOR SETTING AND CLEARING SOMM
: ADDRESSES AND SETING BAD PARITY
:
: INPUT CONDITIONS:
:   WCS BAD PARITY MUST BE SET OR CLEARED
:   BECAUSE IT IS USED IN A CALLED ROUTINE
:
:   INPUT_NUM MUST HAVE THE ADDRESS TO BE CHANGED
:.....

```

```

CHANGE_WCS:  LHLD  INPUT_NUM+2
              SHLD  WCS_ADDRESS      ;PUT ADDRESS IN UPC TO READ
              CALL  READ_WCS_R       ;FROM WCS
              JMP  WRITE_WCS_R      ;WRITE WORD BACK WITH
              ;PARITY

```

```

.....
: CHECK_WCS_LOADED
:
: THIS ROUTINE CHECKS THE WCS LOADED FLAG AND REPORTS AN ERROR IF SET. IT
: WILL POP THE STACK IF ERROR IS SET, SO THAT THE RETURN IN THE ERROR ROUTINE
: WILL SKIP THE REST OF THE ROUTINE THAT HAD THE ERROR.
:.....

```

```

CHECK_WCS_LOADED:

```

```

ZZ-ENKAA-2.1 4C00 4
5B5E 3755 0408'3A
5B61 3756 B7
5B62 3757 C0
5B63 3758
5B63 3759 02F1'32 AF
5B67 3760 F1
5B68 3761
5B68 3762
5B68 3763 0F7E'C3
5B68 3764
5B68 3765
5B68 3766
5B68 3767
5B68 3768
5B68 3769
5B68 3770
5B68 3771
5B68 3772
5B68 3773
5B68 3774
5B68 3775
5B68 3776 0FCF'CD
5B6E 3777
5B6E 3778 10 06
5B70 3779
5B70 3780
5B70 3781 0231'3A
5B73 3782 B7
5B74 3783 C8
5B75 3784
5B75 3785 F1
5B76 3786
5B76 3787
5B76 3788 0F7E'C3
5B79 3789
5B79 3790
5B79 3791
5B79 3792
5B79 3793
5B79 3794
5B79 3795
5B79 3796
5B79 3797
5B79 3798
5B79 3799
5B79 3800
5B79 3801 02F3'32 3C AF
5B7E 3802
5B7E 3803 78
5B7F 3804 005B'32
5B82 3805 0013'32
5B85 3806 01 3E
5B87 3807 0012'32
5B8A 3808
5B8A 3809
5B8A 3810 AF
5B8B 3811 0231'32
5B8E 3812 02F1'32
5B91 3813 40FE 32
5B94 3814

```

M 7

Fiche 1 Frame M7 Sequence 90

```

LDA WCS_LOADED ;GET WCS LOADED FLAG
ORA A ;SET CONDITION CODES
RNZ ;RETURN IF WCS LOADED

CLEAR NOADD_FLG ;ELSE CLEAR FLAG IF SET
POP $PSW ;POP THE STACK TO REMOVE
; RETURN ADDRESS TO CALLING
; ROUTINE
JMP ERROR_ROUTINE ;GO REPORT ERROR BELOW

:*****
: CHECK_ERR_FLG
: THIS ROUTINE CHECKS THE ERROR FLAG AND REPORTS AN ERROR IF SET. IT WILL
: POP THE STACK IF ERROR IS SET, SO THAT THE RETURN IN THE ERROR ROUTINE
: WILL SKIP THE REST OF THE ROUTINE THAT HAD THE ERROR.
:*****

INPUT_NUM_CHECK:
CALL INPUT_NUM_IF ;GET INPUT DATA

CHECK_ERR_10: MVI B,<^X10> ;ERROR CODE IF ERROR SET
; (NO INPUT ADDRESS OR DATA)

CHECK_ERR_FLG: LDA ERROR ;GET ERROR FLAG
ORA A ;SET CONDITION CODES
RZ ;RETURN IF NO ERROR
POP $PSW ;POP THE STACK TO REMOVE
; RETURN ADDRESS TO CALLING
; ROUTINE
JMP ERROR_ROUTINE ;GO REPORT ERROR BELOW

:*****
: ERROR ROUTINE
: THIS ROUTINE WILL REPORT AN ERROR AND ITS ERROR NUMBER
:
: INPUT CONDITIONS:
: B = ERROR CODE
:*****

ERROR_ROUTINE_NO_RET:
SET NO_RETURN ;SET FLAG, JUMP PARSER AFTER
; ERROR REPORTED

ERROR_ROUTINE: MOV A,B ;ERROR CODE FROM B TO AC
STA HEX_BUF ;SAVE ERROR CODE
STA APT_ERROR_NUMBER+1 ;SAVE IN APT MAILBOX
MVI A,1 ;1 IN ACCUM
STA APT_ERROR_NUMBER ;1 IN HIGH BYTE TO INDICATE
; MONITOR ERROR

XRA A ;CLEAR FLAGS
STA ERROR ;RESET ERROR CONDITON
STA NOADD_FLG ;MAKE SURE EXAM FLAG IS CLEAR
STA CTLDIS ;ENABLE CONTROL CHAR CHECK
; IF DISABLED

```

```

ZZ-ENKAA-2.1 4C00 4
5B94 3815 02 3E
5B96 3816 0011'32
5B99 3817 02'0B'21
5B9C 3818 1254'CD
5B9F 3819 3F 3E
5BA1 3820 1265'CD
5BA4 3821
5BA4 3822 01 0E
5BA6 3823 11C9'CD
5BA9 3824
5BA9 3825 02'32'21
5BAC 3826 1254'CD
5BAF 3827
5BAF 3828 02F3'3A
5BB2 3829 B7
5BB3 3830 C8
5BB4 3831
5BB4 3832 AF
5BB5 3833 02F3'32
5BB8 3834 003A'32
5BBB 3835 40 80 31
5BBE 3836 2022'C3
5BC1 3837
5BC1 3838
5BC1 3839
5BC1 3840
5BC1 3841
5BC1 3842
5BC1 3843
5BC1 3844
5BC1 3845 00'BB'21
5BC4 3846 00'5B'11
5BC7 3847 118E'CD
5BCA 3848
5BCA 3849 03 0E
5BCC 3850 11C4'C3
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
OF E9'C2
5BDB 3870
5BDB 3871 02C9'32 3C AF
5BE0 3872 1063'CD
5BE3 3873 1063'CD

```

```

N 7
Fiche 1 Frame N7 Sequence 91
MVI A,2 ;INDICATE ERROR
STA APT MESSAGE_CODE+1 ;STORE IN MAILBOX
LXI H,CRLF A ;POINT TO CR
CALL PRINT_STRING_APT ;PRINT CRLF
MVI A,HEX_3F ;QUESTION MARK
CALL TYPE_CHAR

MVI C,1 ;PRINT 1 BYTE (2 DIGITS)
CALL PRINT_HEX_APT ;PRINT ERROR CODE IN HEX BUF

LXI H,ERROR A ;PRINT " ERROR"
CALL PRINT_STRING_APT

LDA NO_RETURN ;SEE IF CAN'T RETURN
ORA A ;SET CONDITION CODES
RZ ;RETURN IF FLAG CLEAR

XRA A
STA NO_RETURN ;ELSE CLEAR FLAG
STA CONTINUE ;NO CONTINUE ALLOWED
LXI $SP,PRI_STACK ;RESET STACK POINTER
JMP PARSER ;GO TO PARSER

```

```

*****
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

```



```

ZZ-ENKAA-2.1 4C00 4
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

```

```

ZZ-ENKAA-2.1 4C00 4
5C8E 3991
5C8E 3992 109A'C3
5C91 3993 0A 0E
5C93 3994 03'BB'11
5C96 3995 11A4'CD
5C99 3996 C9
5C9A 3997
5C9A 3998
5C9A 3999
5C9A 4000 1078'C3
5C9D 4001
5C9D 4002
5C9D 4003
5C9D 4004
5C9D 4005
5C9D 4006
5C9D 4007
5C9D 4008
5C9D 4009
5C9D 4010
5C9D 4011
5C9D 4012
5C9D 4013
5C9D 4014
5C9D 4015
5C9D 4016
5C9D 4017
5C9D 4018 006C'2A
5CA0 4019 00C2'22
5CA3 4020 1148'CD
5CA6 4021
5CA6 4022 10F9'CD
5CA9 4023
5CA9 4024 23 D3
5CAB 4025 22 D3
5CAD 4026
5CAD 4027 1116'CD
5CB0 4028
5CB0 4029 00'B8'21
5CB3 4030 00'B4'11
5CB6 4031 118E'C3
5CB9 4032
5CB9 4033
5CB9 4034
5CB9 4035
5CB9 4036
5CB9 4037
5CB9 4038
5CB9 4039
5CB9 4040
5CB9 4041
5CB9 4042
5CB9 4043
5CB9 4044
5CB9 4045
5CB9 4046
5CB9 4047 006C'2A
5CBC 4048 00C2'22
5CBF 4049 1148'CD
5CC2 4050

```

```

*
*****
2$:
*
*
*
*
*
*****

```

D 8

```

ELSE ;IF SPECIAL CHARACTER
MVI C,10
LXI D,TEMP_BUF
CALL MOVER_R ;PUT 10 CHARS INTO TEMP_BUF
RET ;GET OUT OF LOOP
ENDIF
JMP 1$ ;LOOP UNTIL SPECIAL CHAR

```

```

*****
READ_WCS_R
THIS ROUTINE READS 3 BYTES OF DATA FROM THE WCS TO CSR VALUE USING
THE ADDRESS IN WCS_ADDRESS. ALSO A NOP IS PLACED IN THE CSR TO PREVENT
THE MACHINE FROM DOING NASTY THINGS

INPUT CONDITIONS:
WCS_ADDRESS HAS WCS ADDRESS TO READ

OUTPUT CONDITIONS:
CSR_VALUE AND WCS_VALUE CONTAIN THE VALUE OF THE WORD READ
*****

```

```

READ_WCS_R: LHLD WCS_ADDRESS ;GET ADDRESS TO READ
            SHLD UPC_VALUE
            CALL LOAD_UPC_R

            CALL NOP_CSR_R

            OUT SETSS ;SING STEP MACHINE TO GET
            OUT CLRSS ;WCS VALUE INTO CSR

            CALL LOAD_CSR_R ;READ IT OUT

            LXI H,CSR_VALUE
            LXI D,WCS_VALUE
            JMP MOVER_3_R

```

```

*****
WRITE_WCS_R
THIS ROUTINE WRITES 3 BYTES OF DATA TO THE WCS FROM WCS_VALUE
BEFORE THE WRITE OCCURES PARITY IS CALCULATED ON THAT 3-BYTE WCS WORD
AND PLACED IN BIT 23.

INPUT CONDITIONS:
WCS_ADDRESS HAS WCS ADDRESS TO READ
WCS_BAD_PARITY = 0 FOR GOOD PARITY, 1 FOR BAD PARITY
PARITY_ON = 0 FOR NO PARITY CALC, 1 TO CALCULATE PARITY
*****

```

```

WRITE_WCS_R: LHLD WCS_ADDRESS ;GET ADDRESS TO WRITE
            SHLD UPC_VALUE
            CALL LOAD_UPC_R

```

```

ZZ-ENKAA-2.1 4C00 4
SCC2 4051 10F9'CD
SCC5 4052
SCC5 4053 0407'32 3C AF
SCCA 4054 00'B4'21
SCCD 4055 0EFF'CD
SCD0 4056 0407'32 AF
SCD4 4057
SCD4 4058 00'B6'21
SCD7 4059 7E
SCD8 4060
SCD8 4061 08 D3
SCDA 4062 2B
SCDB 4063
SCDB 4064 7E
SCDC 4065 09 D3
SCDE 4066 2B
SCDF 4067
SCDF 4068 7E
SCE0 4069 0A D3
SCE2 4070
SCE2 4071 37 D3
SCE4 4072 36 D3
SCE6 4073 C9
SCE7 4074
SCE7 4075
SCE7 4076
SCE7 4077
SCE7 4078
SCE7 4079
SCE7 4080
SCE7 4081
SCE7 4082
SCE7 4083
SCE7 4084
SCE7 4085
SCE7 4086
SCE7 4087
SCE7 4088
SCE7 4089
SCE7 4090
SCE7 4091 F5
SCE8 4092 0068'2A
SCEB 4093 4E
SCEC 4094 00 06
SCEE 4095 09
SCEF 4096 F1
SCF0 4097
SCF0 4098 7E
SCF1 4099 17
SCF2 4100 77
SCF3 4101 0D
SCF4 4102 C8
SCF5 4103 2B
SCF6 4104
SCF6 4105 10F0'C3
SCF9 4106
SCF9 4107
SCF9 4108
SCF9 4109
SCF9 4110

```

```

E 8
Fiche 1 Frame E8 Sequence 95
CALL NOP_CSR_R ;KEEP MACHINE IDLE.
SET WCS_CALC ;PARITY CALC ON WCS
LXI H,WCS_VALUE ;CALC PARITY ON WCS WORD
CALL PARITY_CALC
CLEAR WCS_CALC ;CLEAR FLAG
LXI H,WCS_VALUE+2
MOV A,M ;GET LOW BYTE
OUT WCSB0 ;CLOCK INTO BYTE 0 OF LATCH
DCX H ;INC POINTER TO NEXT BYTE
MOV A,M ;GET MIDDLE BYTE
OUT WCSB1 ;CLOCK INTO BYTE 1 OF LATCH
DCX H ;INC POINTER TO NEXT BYTE
MOV A,M ;GET HIGH BYTE
OUT WCSB2 ;CLOCK INTO BYTE 2 OF LATCH
OUT SETWCK ;CLOCK 24 BITS INTO WCS
OUT CLRWCK ;CLEAR WCS CLOCK
RET

```

```

*****
: 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
*****

```

```

SHIFTER_R: PUSH $PSW ;SAVE CARRY
LHLD SHIFT_PNT ;POINTER TO SHIFT DATA
MOV C,M ;GET COUNT IN BYTES
MVI B,0 ;MAKE COUNT IN REG PAIR
DAD B ;ADD COUNT , GOTO LAST BYTE
POP $PSW

SHIFTER1_R: MOV A,M ;MOVE BYTE 1 LEFT
RAL ;CARRY = ROTATE IN BIT NEXT BYTE
MOV M,A
DCR C ;DEC BYTE COUNT
RZ ;RETURN IF ZERO
DCX H ;MOVE POINTER TO NEXT BYTE
JMP SHIFTER1_R ;REPEAT UNTIL BYTE COUNT=0

```

```

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

```

```

ZZ-ENKAA-2.1 4C00 4
SCF9 4111
SCF9 4112
SCF9 4113
SCF9 4114 00'72'21
SCFC 4115 00'88'11
SCFF 4116 118E'CD
SD02 4117 0064'3A
SD05 4118 03C5'32
SD08 4119 0064'32 AF
SD0C 4120
SD0C 4121 1116'CD
SD0F 4122
SD0F 4123 03C5'3A
SD12 4124 0064'32
SD15 4125
SD15 4126 C9
SD16 4127
SD16 4128
SD16 4129
SD16 4130
SD16 4131
SD16 4132
SD16 4133
SD16 4134
SD16 4135
SD16 4136
SD16 4137 A2 D3
SD18 4138
SD18 4139 00'88'21
SD1B 4140 0EFF'CD
SD1E 4141
SD1E 4142 C5 46 00'55'21
          77 18 3E
SD26 4143
SD26 4144 87 DB
SD28 4145 1F
SD29 4146 00'BA'21
SD2C 4147 03 0E
SD2E 4148 10F0'CD
SD31 4149
SD31 4150
SD31 4151 38 D3
SD33 4152 1138'D2
SD36 4153 39 D3
SD38 4154
SD38 4155 25 D3
SD3A 4156 24 D3
SD3C 4157
SD3C 4158 35 00'55'21
          70 C1 1126'C2
SD45 4159
SD45 4160 A3 D3
SD47 4161
SD47 4162 C9
SD48 4163
SD48 4164
SD48 4165
SD48 4166
SD48 4167
SD48 4168
    
```

```

:.....
NOP_CSR_R:  LXI    H,NOP_INST
            LXI    D,CSR_VALUE
            CALL   MOVER_3_R      ;MOVE NOP TO CSR SAVE AREA
            LDA    PARITY_ON     ;SAVE PARITY ON FLAG
            STA    TEMP_BYTE     ;STORE IN TEMP BYTE
            CLEAR  PARITY_ON     ;CALC NO PARITY ON NOP
            CALL   LOAD_CSR_R    ;LOAD IT
            LDA    TEMP_BYTE     ;GET OLD PARITY_ON FLAG
            STA    PARITY_ON     ;RESTORE FLAG
            RET
    
```

```

:.....
LOAD_CSR_R
THIS ROUTINE LOADS THE CSR FROM THE CSR VALUE AREA,
ALSO CSR_VALUE AREA GETS THE VALUE OF THE CSR
:.....
    
```

```

LOAD_CSR_R:  OUT    VSHIFT      ;SET V-BUS SHIFT MODE
            LXI    H,CSR_VALUE  ;CALC PARITY ON NEW CSR VALUE
            CALL   PARITY_CALC
            DO     24           ;SHIFT 24 BITS
            IN     CSR23       ;GET OLD CSR DATA
            RAR          ;PUT CSR 23 IN CARRY
            LXI    H,CSR_VALUE+2 ;POINTER TO LAST CSR DATA BYTE
            MVI    C,3        ;BYTES TO SHIFT
            CALL   SHIFTER1_R  ;CARRY = MSB OF NEW CSR DATA ON
            ; RETURN
            OUT    CLRCSR     ;CLEAR CSR INPUT
            JNC    1$         ;DON'T SET INPUT IF CARRY CLEAR
            OUT    SETCSR     ;ELSE SET CSR INPUT
            OUT    SETCSK     ;SET UPC CLOCK
            OUT    CLRCSK     ;CLEAR UPC CLOCK
            ENDDO
            OUT    VNORMAL    ;SET V-BUS TO NORMAL MODE
            RET
    
```

```

:.....
LOAD_UPC_R
THIS ROUTINE LOADS THE UPC FROM THE UPC_VALUE AREA,
    
```

```

22-ENKAA-2.1 4C00 4
5D48 4169
5D48 4170
5D48 4171
5D48 4172
5D48 4173 A2 D3
5D4A 4174
5D4A 4175 00'C1'21
5D4D 4176 0068'22
5D50 4177
5D50 4178 B7
5D51 4179
5D51 4180 10E7'CD
5D54 4181
5D54 4182 C5 46 00'55'21
77 OF 3E
5D5C 4183
5D5C 4184 84 DB
5D5E 4185 1F
5D5F 4186 10E7'CD
5D62 4187
5D62 4188 AB D3
5D64 4189 1169'D2
5D67 4190 A9 D3
5D69 4191
5D69 4192
5D69 4193 27 D3
5D6B 4194 26 D3
5D6D 4195
5D6D 4196 35 00'55'21
70 C1 115C'C2
5D76 4197
5D76 4198 A3 D3
5D78 4199
5D78 4200 A9 D3
5D7A 4201 OF 0344'3A
1183'D2
5D81 4202
5D81 4203 AB D3
5D83 4204
5D83 4205
5D83 4206
5D83 4207 C9
5D84 4208
5D84 4209
5D84 4210
5D84 4211
5D84 4212
5D84 4213
5D84 4214
5D84 4215
5D84 4216
5D84 4217
5D84 4218
5D84 4219
5D84 4220
5D84 4221
5D84 4222
5D84 4223
5D84 4224
5D84 4225
    
```

```

: ALSO UPC_VALUE AREA GETS THE VALUE OF THE UPC
:*****
LOAD_UPC_R:  OUT  VSHIFT          ;SET V-BUS SHIFT MODE
              LXI  H,UPC_SHIFT    ;POINTER TO UPC DATA BUFF
              SHLD SHIFT_PNT
              ORA  A              ;CLEAR CARRY
              CALL SHIFTER_R      ;LEFT JUST. 15 BITS
              DO   15             ;SHIFT 15 BITS
              IN   UPC14
              RAR                    ;PUT UPC14 IN CARRY
              CALL SHIFTER_R      ;CARRY = MSB ON RET
              OUT  CLRUPC          ;SET = 0 UPC SER IN
              IFC                    ;SET = 1 UPC SER IN
              OUT  SETUPC
              ENDIF
              OUT  SETUPK          ;SET UPC CLOCK
              OUT  CLRUPK          ;CLEAR UPC CLOCK
              ENDDO
              OUT  VNORMAL        ;SET V-BUS TO NORMAL MODE
              OUT  CLRACK         ;CLEAR CONSOLE ACKNOWLEDGE
              IF   SETACK_FLG     ;MUST SET UP CONSOLE ACKNOWLEDE
              OUT  SETACK         ;BEFORE RETURNING
              ;SET CONSOLE ACKNOWLEDGE IF
              ; FLAG SET
              ENDIF
              RET
    
```

```

:*****
: COMPARE R
: THIS ROUTINE IS USED BY THE STRUCTURED MACRO SET AND MAY ALSO
: BE CALLED DIRECTLY.
: INPUT CONDITIONS:
:   H + L = POINTER TO DATA FIELD 1
:   D + E = POINTER TO DATA FIELD 2
:   C     = COUNT IN BYTES FOR COMPARISON
: OUTPUT CONDITIONS:
:   CONDITIONS CODES ARE SET
:*****
    
```

```

ZZ-ENKAA-2.1      4C00      4
5D84 4226 1A
5D85 4227 BE
5D86 4228 C0
5D87 4229 23
5D88 4230 13
5D89 4231 0D
5D8A 4232 1184'C2
5D8D 4233 C9
5D8E 4234
5D8E 4235
5D8E 4236
5D8E 4237
5D8E 4238
5D8E 4239
5D8E 4240
5D8E 4241
5D8E 4242
5D8E 4243
5D8E 4244
5D8E 4245
5D8E 4246
5D8E 4247
5D8E 4248
5D8E 4249
5D8E 4250
5D8E 4251
5D8E 4252
5D8E 4253
5D8E 4254 03 0E
5D90 4255 11A4'C3
5D93 4256
5D93 4257
5D93 4258 04'3D'21
5D96 4259 00'A9'11
5D99 4260 11A2'C3
5D9C 4261
5D9C 4262
5D9C 4263 00'58'11
5D9F 4264 00'A9'21
5DA2 4265
5DA2 4266 04 0E
5DA4 4267 7E
5DA5 4268 12
5DA6 4269 23
5DA7 4270 13
5DA8 4271 0D
5DA9 4272 C8
5DAA 4273 11A4'C3
5DAD 4274
5DAD 4275
5DAD 4276
5DAD 4277
5DAD 4278
5DAD 4279
5DAD 4280
5DAD 4281
5DAD 4282
5DAD 4283
5DAD 4284 03C6'2A
5DB0 4285 05 0E

```

```

H 8
Fiche 1 Frame H8 Sequence 98
COMPARE_R: LDAX D ;FIELD 2 BYTE
            CMP M ;COMPARED TO FIELD 1 BYTE
            RNZ ;RETURN WITH CC NON ZERO IF NOT SAME
            INX H ;INCREMENT FIELD 1 POINTER
            INX D ;INCREMENT FIELD 2 POINTER
            DCR C ;DECREMENT COUNT
            JNZ COMPARE_R ;LOOP UNTIL COUNT GONE
            RET ;DONE - RETURN ZERO CC

```

```

*****
MOVER_R
THIS ROUTINE MOVES DATA FROM THE ADDRESS GIVEN IN H+L TO THE ADDRESS
GIVEN IN D+E FOR THE COUNT IN C
INPUT CONDITIONS:
H+L = SOURCE ADDRESS
D+E = DEST. ADDRESS
C = LENGTH
OUTPUT CONDITIONS:
REGISTERS ARE INCREMENTED DURING THE MOVE, SO THEY
ARE POINTING TO THE LOCATION AFTER THE DATA
*****

```

```

MOVER_3_R: MVI C,3 ;MOVE 3 TO COUNT
           JMP MOVER_R ;GO MOVE THREE BYTES
MOV_INPUT_DATA0: LXI H,INPUT_NUM ;SOURCE IN INPUT_NUM
                 LXI D,DATA0 ;DEST IS DATA0-DATA3
                 JMP MOVER_4_R ;MOVE 4 BYTES
MOV_DATA0_HEXBUF: LXI D,HEX_BUF ;DESTINATION IS HEX_BUF
MOV_FROM_DATA0: LXI H,DATA0 ;SOURCE IS DATA0-3
MOVER_4_R: MVI C,4 ;MOVE 4 TO COUNT
MOVER_R: MOV A,M ;GET SOURCE DATA
         STAX D ;PUT AT DESTINATION
         INX H ;INCREMENT POINTERS
         INX D ;INCREMENT POINTERS
         DCR C ;DECREMENT COUNT
         RZ ;RETURN IF DONE
         JMP MOVER_R ;LOOP OTHERWISE

```

```

*****
PRINT_FLG
THIS ROUTINE PRINTS 5 CHARS AND A SPACE FROM THE POINTER TEMP_WORD
*****
PRINT_FLG: LHL TEMP_WORD ;POINTER TO FLAG TO PRINT
           MVI C,5

```

```

ZZ-ENKAA-2.1 4C00 4
5DB2 4286 1243'CD
5DB5 4287 1260'C3
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 4300
5DB8 4301 04 OE
5DBA 4302 11C4'C3
5DBD 4303
5DBD 4304 02 OE
5DBF 4305 11C4'C3
5DC2 4306
5DC2 4307 01 OE
5DC4 4308 0038'3A
5DC7 4309 B7
5DC8 4310 C0
5DC9 4311 00'5B'21
5DCC 4312 0321'22
5DCF 4313 79
5DD0 4314 0320'32
5DD3 4315
5DD3 4316 0321'2A
5DD6 4317 7E
5DD7 4318 1F
5DD8 4319 1F
5DD9 4320 1F
5DDA 4321 1F
5DDB 4322 0F E6
5DDD 4323 11FC'CD
5DE0 4324
5DE0 4325 0321'2A
5DE3 4326 7E
5DE4 4327 23
5DE5 4328 0321'22
5DE8 4329 0F E6
5DEA 4330 11FC'CD
5DED 4331
5DED 4332 03'20'21
5DF0 4333 35
5DF1 4334 11D3'C2
5DF4 4335
5DF4 4336 02F5'3A
5DF7 4337 B7
5DF8 4338 1260'CC
5DFB 4339 C9
5DFC 4340
5DFC 4341
5DFC 4342
5DFC 4343
5DFC 4344
5DFC 4345

```

I 8

Fiche 1 Frame 18 Sequence 99

```

CALL PRINTER ;PRINT THE 5 CHAR
JMP SPACE_R ;AND THE SPACE

```

```

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

```

```

PRINT_HEX_4: MVI C,4 ;LOAD BYTE COUNT WITH 4
             JMP PRINT_HEX_R ;PRINT FOUR CHARS

PRINT_HEX_2: MVI C,2 ;BYTE COUNT OF 2
             JMP PRINT_HEX_R ;PRINT 2 CHARS

PRINT_HEX_1_R: MVI C,1 ;PRINT 1 CHAR
PRINT_HEX_R: LDA APT ;SEE IF APT PRESENT
             ORA A ;SET CONDITION CODES
             RNZ ;DON'T PRINT IF APT

PRINT_HEX_APT: LXI H,HEX_BUF
               SHLD PRINT_PNT ;POINTER TO PRINT DATA
               MOV A,C
               STA PRINT_CNT ;PRINT COUNT

1$: LHL PRINT_PNT
   MOV A,M ;GET FIRST BYTE
   RAR
   RAR
   RAR
   RAR
   ANI <^XF> ;GET FIRST NIBBLE
   CALL PR_HEX_DIGIT ;PRINT HEX DIGIT

   LHL PRINT_PNT
   MOV A,M
   INX H
   SHLD PRINT_PNT
   ANI <^XF> ;GET SECOND NIBBLE
   CALL PR_HEX_DIGIT

   LXI H,PRINT_CNT ;DECR PRINT COUNT
   DCR M
   JNZ 1$ ;RETURN IF DONE

   LDA NO_SPACE ;GET FLAG
   ORA A ;SET CONDITION CODES
   CZ SPACE_R ;PRINT SPACE IF NO FLAG
   RET

```

```

*****
: PR_HEX_DIGIT
: THIS ROUTINE PRINTS A HEX DIGIT

```

```

ZZ-ENKAA-2.1 4C00 4
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$
1$: ADI HEX_7 :ADD FOR A-F
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 0 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 INTERRUPTS

RET

```

```

:*****
: PRINTER

```

ZZ-ENKAA-2.1 4C00 4  
 SE43 4404  
 SE43 4405  
 SE43 4406  
 SE43 4407  
 SE43 4408  
 SE43 4409  
 SE43 4410  
 SE43 4411 79  
 SE44 4412 4133 CD  
 SE47 4413  
 SE47 4414 1341'C3  
 SE4A 4415  
 SE4A 4416  
 SE4A 4417  
 SE4A 4418  
 SE4A 4419  
 SE4A 4420  
 SE4A 4421  
 SE4A 4422  
 SE4A 4423  
 SE4A 4424  
 SE4A 4425  
 SE4A 4426  
 SE4A 4427 0038'3A  
 SE4D 4428 B7  
 SE4E 4429 412E CC  
 SE51 4430  
 SE51 4431  
 SE51 4432  
 SE51 4433 1341'C3  
 SE54 4434  
 SE54 4435  
 SE54 4436  
 SE54 4437  
 SE54 4438  
 SE54 4439  
 SE54 4440  
 SE54 4441  
 SE54 4442  
 SE54 4443  
 SE54 4444  
 SE54 4445  
 SE54 4446  
 SE54 4447 412E CD  
 SE57 4448  
 SE57 4449  
 SE57 4450 1341'C3  
 SE5A 4451  
 SE5A 4452  
 SE5A 4453  
 SE5A 4454  
 SE5A 4455  
 SE5A 4456  
 SE5A 4457  
 SE5A 4458  
 SE5A 4459  
 SE5A 4460  
 SE5A 4461 02'0B'21  
 SE5D 4462 124A'C3  
 SE60 4463

```

: THIS ROUTINE PRINTS A STRING OF CHARS
: INPUT CONDITIONS
:   H+L = POINTER TO THE STRING
:   C   = NUMBER OF CHARS IN THE STRING
:*****
PRINTER:      MOV     A,C           ;LD A WITH CHAR COUNT
              CALL    SLVECT      ;CALL SEND LINE ROUTINE IN BOOT
              ;BLOCK.
              JMP     CHECK_CNTLC  ;IF ^C TYPED, GO TO PARSER
:*****

: PRINT_STRING_R
: THIS ROUTINE PRINTS A STRING OF CHARS WITH THE CHAR COUNT AS THE FIRST BYTE
: OF THE STRING TO THE TERMINAL IF APT IS NOT PRESENT.
: INPUT CONDITIONS
:   H+L = POINTER TO THE STRING BEGINNING WITH CHAR COUNT
:*****
PRINT_STRING_R: LDA     APT           ;GET APT PRESENT FLAG
                ORA     A           ;SET CONDITION CODES
                CZ     SLVECT      ;CALL SEND LINE ROUTINE IN BOOT
                ;BLOCK, IF APT NOT PRESENT,
                ; THAT EXPECTS CHAR COUNT
                ; AS FIRST CHAR IN STRING
                JMP     CHECK_CNTLC ;IF ^C TYPED, GO TO PARSER
:*****

: PRINT_STRING_APT
: THIS ROUTINE PRINTS A STRING OF CHARS WITH THE CHAR COUNT AS THE FIRST BYTE
: OF THE STRING TO THE TERMINAL OR TO APT IF PRESENT.
: INPUT CONDITIONS
:   H+L = POINTER TO THE STRING BEGINNING WITH CHAR COUNT
:*****
PRINT_STRING_APT: CALL    SLVECT      ;CALL SEND LINE ROUTINE IN BOOT
                ;BLOCK THAT EXPECTS CHAR COUNT
                ; AS FIRST CHAR IN STRING
                JMP     CHECK_CNTLC ;IF ^C TYPED, GO TO PARSER
:*****

: CRLF_R
: THIS ROUTINE PRINTS A RETURN AND LINE FEED (THE BOOT BLOCK PRINT LINE
: ROUTINE SUPPLIES THE LINE FEED).
:*****
CRLF_R:      LXI     H,CRLF_A      ;POINT TO CR
              JMP     PRINT_STRING_R ;PRINT IT
    
```





```

ZZ-ENKAA-2.1 4C00 4
SF0B 4577
SF0B 4578
SF0B 4579 35 00'55'21
70 C1 12B2'C2
SF14 4580
SF14 4581 01'64'21
SF17 4582 03E0'22
SF1A 4583
SF1A 4584 C9
SF1B 4585
SF1B 4586
SF1B 4587
SF1B 4588
SF1B 4589
SF1B 4590
SF1B 4591
SF1B 4592
SF1B 4593
SF1B 4594
SF1B 4595
SF1B 4596 4145 CD
SF1E 4597
SF1E 4598
SF1E 4599
SF1E 4600
SF1E 4601 131B'D2
SF21 4602
SF21 4603 C9
SF22 4604
SF22 4605
SF22 4606
SF22 4607
SF22 4608
SF22 4609
SF22 4610
SF22 4611
SF22 4612
SF22 4613
SF22 4614 0039'32 3C AF
SF27 4615 4086 2A
SF2A 4616 F9
SF2B 4617 0333'32 AF
SF2F 4618 37
SF30 4619 03 3E
SF32 4620 C9
SF33 4621
SF33 4622
SF33 4623
SF33 4624
SF33 4625
SF33 4626
SF33 4627
SF33 4628
SF33 4629
SF33 4630
SF33 4631 4086 2A
SF36 4632 F9
SF37 4633 AF
SF38 4634 003A'32
SF38 4635

```

```

*****
*****

```

```

N 8
Fiche 1 Frame N8 Sequence 104
ENDIF
ENDDO ;REPEAT FOR ALL CHARACTERS
LXI M,TTBUF ;INIT TTY BUFFER POINTER
SHLD TIBUF_PNT
RET

```

```

.....
: INPUT_CHAR_IF
: THIS ROUTINE INPUTS A CHAR FROM THE APT-RD PORT
:
: OUTPUT CONDITIONS
: ACCUMULATOR = CHAR THAT WAS INPUT
:
:.....

```

```

INPUT_CHAR_IF: CALL GSVECT ;CALL GET SILO ROUTINE IN
; BOOT BLOCK. IF CHAR FOUND,
; CARRY IS SET AND CHAR IS IN
; ACCUMULATOR
JNC INPUT_CHAR_IF ;LOOP UNTIL CHAR RECEIVED
RET

```

```

.....
: CNTLC_VEC
: THIS ROUTINE IS ENTERED AFTER CALLING THE BOOT BLOCK GET CHAR OR GET LINE
: ROUTINE IF A ^C WAS TYPED. IT WILL SET A FLAG, RESTORE THE STACK POINTER,
: AND RETURN TO THE CALLING ROUTINE.
:
:.....

```

```

CNTLC_VEC: SET CNTLC_TYPED ;SET FLAG INDICATING ^C
LHLD SPBUFF ;HL GETS OLD SP
SPLH ;RESTORE SP
CLEAR REPEAT ;CLEAR REPEAT FLAG
STC ;SET CARRY FOR INPUT_CHAR_IF
MVI A,<^X3> ;PUT HEX FOR ^C IN A
RET ;RETURN WILL NOW WORK

```

```

.....
: CNTLP_VEC
: THIS ROUTINE IS ENTERED AFTER CALLING THE BOOT BLOCK GET CHAR OR GET LINE
: ROUTINE IF A ^P WAS TYPED. IT WILL FIX THE STACK POINTER AND CALL SET_FOR_
: CONT TO REENTER THE PARSER FOR THE NEXT COMMAND.
:
:.....

```

```

CNTLP_VEC: LHLD SPBUFF ;HL GETS OLD SP
SPLH ;RESTORE SP
XRA A ;CLEAR FLAGS
STA CONTINUE ;CLEAR CONTINUE FLAG IN CASE
; BOBS TEST IS EXECUTING

```

ZZ-ENKAA-2.1 4C00 4  
 SF3B 4636 0333'32  
 SF3E 4637 1349'C3  
 SF41 4638  
 SF41 4639  
 SF41 4640  
 SF41 4641  
 SF41 4642  
 SF41 4643  
 SF41 4644  
 SF41 4645  
 SF41 4646  
 SF41 4647  
 SF41 4648 0039'3A  
 SF44 4649 B7  
 SF45 4650 1349'C4  
 SF48 4651 C9  
 SF49 4652  
 SF49 4653  
 SF49 4654  
 SF49 4655  
 SF49 4656  
 SF49 4657  
 SF49 4658  
 SF49 4659  
 SF49 4660  
 SF49 4661  
 SF49 4662 AF  
 SF4A 4663 0039'32  
 SF4D 4664 02F4'32  
 SF50 4665  
 SF50 4666 023B'3A  
 SF53 4667 0204'32  
 SF56 4668 0335'3A  
 SF59 4669 0205'32  
 SF5C 4670  
 SF5C 4671 0F 0335'3A  
 136E'D2  
 SF63 4672 1399'CD  
 SF66 4673 02C8'32 3C AF  
 SF6B 4674 0B2C'CD  
 SF6E 4675  
 SF6E 4676  
 SF6E 4677 02C8'32 AF  
 SF72 4678 00C4'2A  
 SF75 4679 00C8'22  
 SF78 4680  
 SF78 4681 00'BB'21  
 SF7B 4682 00'BE'11  
 SF7E 4683 118E'CD  
 SF81 4684  
 SF81 4685 023B'32 AF  
 SF85 4686  
 SF85 4687  
 SF85 4688 0F 0209'3A  
 1394'DA  
 SF8C 4689 003A'32 3C AF  
 SF91 4690  
 SF91 4691 1396'C3  
 SF94 4692 3C D3  
 SF96 4693

```

STA REPEAT ;CLEAR REPEAT FLAG
JMP SET_FOR_CONT_R ;SET UP FOR CO COMMAND AND GO
; TO PARSER
*****
CHECK_CNTLC
THIS ROUTINE CHECKS THE CNTLC TYPED FLAG TO SEE IF A CNTLC HAS BEEN
DETECTED. IF SO, IT CALLS THE SET_FOR_CONT_R ROUTINE.
*****
CHECK_CNTLC: LDA CNTLC_TYPED ;GET FLAG
ORA A ;SET CONDITION CODES
CNZ SET_FOR_CONT_R ;CALL SET FOR CONT IF ^C SET
RET ;ELSE RETURN
*****
SET_FOR_CONT_R:
THIS ROUTINE SETS UP FOR A CONTINUE COMMAND TO BE EXECUTED IF IT IS
ASKED FOR
*****
SET_FOR_CONT_R: XRA A ;CLEAR FLAGS
STA CNTLC_TYPED ;CLEAR CNTLC_TYPED
STA NO_SAVE_UPC_CSR ;CLEAR NO_SAVE_UPC_CSR
LDA EXECUTE ;SAVE EXECUTION FLAGS
STA C_EXECUTE
LDA RUNNING
STA C_RUNNING
IF RUNNING
CALL STOP_CPU ;HALT CPU
SET MICRO_STEP ;SET FLAG TO PRINT UPC-1
CALL EXAM_UPC ;PRINT OUT HALTED ADDRESS
ENDIF
CLEAR MICRO_STEP ;CLEAR MICRO STEP MODE
LHLD SAVED_UPC ;SAVE CSR AND UPC FOR CONT
SHLD CONT_SAVE_UPC
LXI H, SAVED_CSR
LXI D, CONT_SAVE_CSR
CALL MOVER_3_R
CLEAR EXECUTE ;CLEAR EXECUTION FLAG TO ENTER
; PARSER
IFN CONSOLE_TEST ;IF 8085 TEST IS NOT RUNNING
SET CONTINUE ;CONTINUE ALWAYS LEGAL FOR WCS
; TESTS IF THIS ROUTINE CALLED
ELSE
OUT CLRLIT ;IF 8085 TEST, TURN OFF RUN
; LIGHT
  
```



```

ZZ-ENKAA-2.1 4C00 4
5FDD 4753
5FDD 4754
5FDD 4755
5FDD 4756
5FDD 4757
5FDD 4758 00C4'2A
5FE0 4759 00C2'22
5FE3 4760
5FE3 4761 1148'CD
5FE6 4762 00'BB'21
5FE9 4763 00'BB'11
5FEC 4764 118E'CD
5FEF 4765 1116'CD
5FF2 4766
5FF2 4767
5FF2 4768 0F 030C'3A
13FB'D2
5FF9 4769 A0 D3
5FFB 4770
5FFB 4771
5FFB 4772 0F 029F'3A
1404'D2
6002 4773 A6 D3
6004 4774
6004 4775
6004 4776 0F 02C8'3A
140D'DA
6008 4777 21 D3
600D 4778
600D 4779
600D 4780 0335'32 3C AF
6012 4781 3D D3
6014 4782
6014 4783 C9
6015 4784
6015 4785
6015 4786
6015 4787
6015 4788
6015 4789
6015 4790
6015 4791
6015 4792
6015 4793
6015 4794
6015 4795
6015 4796 0071'32
6018 4797
6018 4798 04'35'21
6018 4799 04'32'11
601E 4800 118E'CD
6021 4801
6021 4802 04'32'21
6024 4803 1450'CD
6027 4804
6027 4805 149E'CD
602A 4806
602A 4807 04'31'21
602D 4808 15C2'CD
6030 4809

```

```

: THIS SUBROUTINE IS USED TO RESTART THE CPU AND IT RESTORES THE
: CSR AND UPC. IF MICRO STEP IS SET THE CPU IS SET UP BUT NOT STARTED
:
:*****

```

```

RESTART_CPU:  LHL D   SAVED_UPC           ;RESTORE UPC
              SHLD  UPC_VALUE

```

```

              CALL  LOAD_UPC_R
              LXI  H,SAVED_CSR           ;LOAD CSR
              LXI  D,CSR_VALUE
              CALL  MOVER_3_R
              CALL  LOAD_CSR_R

```

```

START_CPU_COM:

```

```

              IF    PAR_ON
              OUT   ENBPTR               ;ENABLE PARITY
              ENDF

```

```

              IF    MEM_REQ
              OUT   ENBMEM               ;ENABLE MEM REF
              ENDF

```

```

              IFN  MICRO_STEP
              OUT   SETCLK
              ENDF

```

```

              SET  RUNNING
              OUT  SETLIT                 ;SET RUN LITE
              RET

```

```

:*****
: WRITE_LS_R
: THIS ROUTINE WRITES A 32 BIT LS VALUE
: INPUT CONDITIONS:
:   AC = LS ADDRESS THAT IS TO BE WRITTEN
:   DATA0 THRU DATA3 CONTAIN THE 32 BITS OF DATA TO BE WRITTEN
:*****

```

```

WRITE_LS_R:  STA    XD_ADDR              ;SAVE ADDRESS TO BE WRITTEN

```

```

              LXI  H,S_WRITE_LS         ;RESET MOVE INSTRUCTION
              LXI  D,X_WRITE_LS+1
              CALL MOVER_3_R

```

```

              LXI  H,X_WRITE_LS+1       ;MODIFY MOVE INSTRUCTION
              CALL MAKE_XD_R

```

```

              CALL WRITE_DATA_32_R      ;WRITE THE DATA TO 2901

```

```

              LXI  H,X_WRITE_LS         ;PERFORM THE WRITE
              CALL PERFORM_CSR_R

```

```

ZZ-ENKAA-2.1 4000 4
6030 4810 4129 C3
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
6035 4824
6035 4825 0071'32
6038 4826
6038 4827 04'23'21
6038 4828 04'20'11
603E 4829 118E'CD
6041 4830
6041 4831 04'20'21
6044 4832 1450'CD
6047 4833
6047 4834 04'1F'21
604A 4835 15C2'CD
604D 4836 1481'C3
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
6053 4851 17
6054 4852 00 3E
6056 4853 17
6057 4854 86
6058 4855 77
6059 4856 23
605A 4857
605A 4858 0071'3A
605D 4859
605D 4860 87
605E 4861 17
605F 4862 86
6060 4863 77
6061 4864
6061 4865 C9
6062 4866
6062 4867
6062 4868
6062 4869

```

E 9

JMP GCVECT

Fiche 1 Frame E9

Sequence 108

;SEE IF INPUT FOR SILO

```

*****
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
*****

```

```

READ_LS_7:      MVI      A,LS_7              ;LS ADDRESS 7
READ_LS_R:      STA      XD_ADDRS          ;SAVE ADDRESS TO BE READ
                LXI      H,S_READ_LS      ;RESET MOVE INSTRUCTION
                LXI      D,X_READ_LS+1
                CALL     MOVER_3_R
                LXI      H,X_READ_LS+1    ;AND MODIFY MOVE TO DO IT
                CALL     MAKE_XD_R
                LXI      H,X_READ_LS      ;PERFORM THE READ
                CALL     PERFORM_CSR_R
                JMP      READ_DATA_32_R

```

```

*****
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
*****

```

```

MAKE_XD_R:      LDA      XD_ADDRS          ;PUT BIT 7 OF XD_ADDRS
                RAL                      ;INTO BIT 16 OF INSTRUCTION
                MVI      A,0
                RAL
                ORA      M
                MOV      M,A
                INX      H
                LDA      XD_ADDRS          ;PUT BITS 0 - 6 INTO BITS
                ORA      A                ;15 - 9 OF INSTRUCTION
                RAL                      ;CLEAR CARRY
                ORA      M
                MOV      M,A
                RET

```

```

ZZ-ENKAA-2.1  4C00  4
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_ADDR = 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_ADDR      ;PUT BIT 2 OF B_ADDR INTO
              RAR      R              ;BIT 8 OF THE INSTRUCTION
              RAR      R
              MVI      A,0
              RAL      R
              ORA      M
              MOV      M,A
              INX      H
              LDA      B_ADDR      ;PUT BIT 1 OF B_ADDR INTO
              RAR      R              ;BIT 7 OF THE INSTRUCTION
              MVI      A,0
              RAR      R
              ORA      M
              MOV      M,A
              RET

```

```

MAKE_A
THIS ROUTINE MAKES AN INSTRUCTION HAVE THE PROPER A ADDRESS FIELD
INPUT CONDITIONS:
A_ADDR = 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_ADDR      ;PUT BIT 2 OF B_ADDR INTO
              RAL      R              ;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  
 6084 4932 1496'CD  
 6087 4933  
 6087 4934 00'AB'11  
 608A 4935 1496'CD  
 608D 4936  
 608D 4937 00'AA'11  
 6090 4938 1496'CD  
 6093 4939  
 6093 4940 00'A9'11  
 6096 4941  
 6096 4942  
 6096 4943 EC D3  
 6098 4944 80 DB  
 609A 4945 12  
 609B 4946 151B'C3  
 609E 4947  
 609E 4948  
 609E 4949  
 609E 4950  
 609E 4951  
 609E 4952  
 609E 4953  
 609E 4954  
 609E 4955  
 609E 4956  
 609E 4957  
 609E 4958  
 609E 4959  
 609E 4960  
 609E 4961  
 609E 4962 OF 0240'3A  
 14D6'DA  
 60A5 4963  
 60A5 4964 00'A1'21  
 60AB 4965 15C2'CD  
 60AB 4966  
 60AB 4967  
 60AB 4968  
 60AB 4969 C5 46 00'55'21  
 77 20 3E  
 60B3 4970  
 60B3 4971 00'AC'21  
 60B6 4972 04 OE  
 60B8 4973 10F0'CD  
 60B8 4974  
 60B8 4975 14C7'D2  
 60BE 4976 00'79'21  
 60C1 4977 15C2'CD  
 60C4 4978  
 60C4 4979 14CA'C3  
 60C7 4980 1515'CD  
 60CA 4981  
 60CA 4982  
 60CA 4983 35 00'55'21  
 70 C1 14B3'C2  
 60D3 4984  
 60D3 4985 1514'C3  
 60D6 4986

```

READ_DATA_32_R: LXI    D,DATA3      ;DE POINTS TO DATA AREA
                  CALL   1$         ;READ BYTE AND STORE

                  LXI    D,DATA2    ;DE POINTS TO DATA AREA
                  CALL   1$         ;READ NEXT BYTE AND STORE

                  LXI    D,DATA1    ;DE POINTS TO DATA AREA
                  CALL   1$         ;READ NEXT BYTE AND STORE

                  LXI    D,DATA0    ;DE POINTS TO DATA AREA
                  ; READ NEXT BYTE AND STORE

1$:              OUT    YBUSRD
                  IN     READ       ;READ MOST SIG. BYTE
                  STAX   D          ;STORE IN DATA BUFFER
                  JMP    SHIFT_R_2901_R ;READ NEXT BYTE
  
```

```

*****
WRITE DATA 32 R
THIS ROUTINE IS USED TO WRITE 32 BITS OF DATA TO THE CPU
DATA WRITES TO THE CPU ARE DONE IN TWO DIFFERENT MODES
SLOW MODE: AND FAST MODE: SLOW MODE USES NO SUPPORT MICRO CODE
IN THE WCS MEMORY, AND THE FAST MODE DOES. THE MODE IS INITIALLY
SLOW MODE UNTIL A WCS FILE IMAGE IS LOADED. THEN IT IS SWITCHED TO
FAST MODE UNLESS THERE IS AN ERROR IN THE FAST MODE VERIFY.
*****
  
```

```

WRITE_DATA_32_R: IFN    FAST_WRITE

                  ;SLOW WRITE
                  LXI    H,X CLR_WRO ;CLEAR WORKING REG AND COMPLIM.
                  CALL   PERFORM_CSR_R ;TO MAKE ALL 1'S THEN ROTATE
                  ;LEFT TO MAKE A 1 AND ASH LEFT
                  ;TO MAKE A ZERO

                  DO     32

                  LXI    H,DATA_SHIFT+4 ;SHIFT DATA INTO CPU
                  MVI    C,4
                  CALL   SHIFTER1_R

                  IFC
                  LXI    H,X ROT_L   ;ADDR OF CSR INSTS TO EXECUTE
                  CALL   PERFORM_CSR_R ;MAKE A 1 BY END AROUND CARRY

                  ELSE
                  CALL   SHIFT_L_2901_R ;MAKE A 0 BY SHIFT IN ZERO
                  ENDIF

                  ENDDO

                  ELSE
  
```



ZZ-ENKAA-2.1 4C00 4

6121 5045  
 6121 5046  
 6121 5047  
 6121 5048  
 6121 5049  
 6121 5050  
 6121 5051  
 6121 5052  
 6121 5053  
 6121 5054  
 6121 5055  
 6121 5056 01 FE 02C8'3A  
 1552'C2

6129 5057  
 6129 5058 413D CD  
 612C 5059  
 612C 5060  
 612C 5061  
 612C 5062  
 612C 5063 1529'D2  
 612F 5064  
 612F 5065 7F E6  
 6131 5066 20 FE  
 6133 5067 1546'CA  
 6136 5068  
 6136 5069 1399'CD  
 6139 5070 02C8'32 AF  
 613D 5071 0F47'CD  
 6140 5072 1349'CD  
 6143 5073 13DD'C3  
 6146 5074  
 6146 5075 4145 CD  
 6149 5076 1570'CD  
 614C 5077 0B2C'CD  
 614F 5078  
 614F 5079 156F'C3  
 6152 5080  
 6152 5081 1570'CD  
 6155 5082 03'A3'21  
 6158 5083 158C'CD  
 6158 5084 C0  
 615C 5085  
 615C 5086 1399'CD  
 615F 5087 0B2C'CD  
 6162 5088 02C8'32 AF  
 6166 5089 0F47'CD  
 6169 5090 1349'CD  
 616C 5091 13DD'C3  
 616F 5092

616F 5093  
 616F 5094  
 616F 5095 C9  
 6170 5096  
 6170 5097  
 6170 5098  
 6170 5099  
 6170 5100  
 6170 5101  
 6170 5102  
 6170 5103

```

*****
: MICRO STEPER
: THIS ROUTINE SINGLE STEPS THE CPU. IF THE MACHINE IS IN SPACE BAR
: MODE ANY CHAR OTHER THAN SPACE WILL CLEAR EXECUTE MODE AND STOP
: MICRO STEPPING. IF THE MACHINE IS IN COUNT MODE IT WILL DECREMENT
: THE COUNT AND STOP MICRO STEPPING AT 0 COUNT
*****

```

```

***** MICRO_STEPPER:  IFEQI  MICRO_STEP,1  ;SPACE BAR MODE
*
*
1$:  CALL  RSVECT  ;CALL READ SILO ROUTINE IN
*      ; BOOT BLOCK. IF CHAR FOUND,
*      ; CARRY IS SET AND CHAR IS IN
*      ; ACCUMULATOR
*
*      JNC  1$  ;LOOP UNTIL CHAR RECEIVED
*
*      ANI  HEX 7F  ;STRIP ASCII PARITY BIT
*      CPI  SPACE_A  ;SPACE?
*      JZ   2$  ;JUMP TO SINGLE STEP IF SPACE
*
*      CALL STOP_CPU  ;STOP CPU, SAVE CSR AND UPC
*      CLEAR MICRO_STEP  ;CLEAR FLAG
*      CALL SAVE_WR  ;SAVE CPU WORKING REGS
*      CALL SET_FOR_CONT_R  ;NO SPACE, END MICRO STEPPING
*      JMP  RESTART_CPU  ;CONTINUE COMES HERE
*
2$:  CALL  GSVECT  ;CALL GET SILO TO POP OUT SPACE
*      CALL  MICRO_STEP_CPU_R  ;STEP MACHINE AND PRINT ADDRESS
*      CALL  EXAM_UPC
*
*      ELSE
*
*      CALL  MICRO_STEP_CPU_R  ;COUNT MODE
*      LXI  H,STEP_CNT  ;DECREMENT STEP COUNT
*      CALL  DECR_WORD_R
*      RNZ  ;RETURN UNTIL COUNT EXHAUSTED
*
*      CALL  STOP_CPU  ;STOP CPU, SAVE CSR AND UPC
*      CALL  EXAM_UPC  ;PRINT ENDING UPC
*      CLEAR MICRO_STEP  ;CLEAR FLAG
*      CALL  SAVE_WR  ;SAVE CPU WORKING REGS
*      CALL  SET_FOR_CONT_R  ;NO SPACE, END MICRO STEPPING
*      JMP  RESTART_CPU  ;CONTINUE COMES HERE
*
***** ENDIF
RET

```

```

*****
: MICRO_STEP_CPU_R
: THIS ROUTINE CAUSES THE MACHINE TO DO ONE MICRO INSTRUCTION
*****

```

```

ZZ-ENKAA-2.1  4C00  4
6170 5104
6170 5105
6170 5106 23 D3
6172 5107 22 D3
6174 5108
6174 5109 C9
6175 5110
6175 5111
6175 5112
6175 5113
6175 5114
6175 5115
6175 5116
6175 5117
6175 5118
6175 5119 00 06
6177 5120 05 0E
6179 5121 03C6'2A
617C 5122 09
617D 5123 03C6'22
6180 5124
6180 5125 C9
6181 5126
6181 5127
6181 5128
6181 5129
6181 5130
6181 5131
6181 5132
6181 5133
6181 5134
6181 5135
6181 5136
6181 5137
6181 5138
6181 5139
6181 5140 23
6182 5141
6182 5142 7E
6183 5143 01 C6
6185 5144 77
6186 5145 2B
6187 5146
6187 5147 7E
6188 5148 00 CE
618A 5149 77
6188 5150
6188 5151 C9
618C 5152
618C 5153
618C 5154
618C 5155
618C 5156
618C 5157
618C 5158
618C 5159
618C 5160
618C 5161
618C 5162
618C 5163

```

```

MICRO_STEP_CPU_R:
    OUT    SETSS
    OUT    CLRSS                ;SINGLE STEP CPU
    RET

```

```

.....
: INC_TEMPW_5
: THIS ROUTINE INCREMENTS THE VALUE OF TEMP_WORD BY 5
:.....

```

```

INC_TEMPW_5:    MVI    B,0
                MVI    C,5
                LHL    TEMP_WORD
                DAD    B                ;ADD 5 TO TEMP_WORD
                SHLD   TEMP_WORD       ;TO MOVE TO NEXT FLAG NAME
                RET

```

```

.....
: INC_WORD_R
: THIS ROUTINE INCREMENTS A WORD OF DATA
:
: INPUT CONDITIONS:
:   H+L = POINTER TO THE WORD TO INCREMENT
: OUTPUT CONDITIONS:
:   CONDITION CODES
:.....

```

```

INC_WORD_R:    INX    H                ;POINT TO LOW ORDER PART
                MOV    A,M
                ADI    1
                MOV    M,A             ;INC VALUE IN LOW ORD. PART
                DCX    H                ;MOVE TO HIGH ORDER PART
                MOV    A,M
                ACI    0                ;ADD ZERO + CARRY
                MOV    M,A
                RET

```

```

.....
: DECR_WORD_R
: THIS ROUTINE DECREMENTS A WORD OF DATA
:
: INPUT CONDITIONS:
:   H+L = POINTER TO THE WORD TO DECREMENT
: OUTPUT CONDITIONS:
:   CONDITION CODES
:.....

```



```

ZZ-ENKAA-2.1 4C00 4
61C2 5223
61C2 5224
61C2 5225
61C2 5226
61C2 5227
61C2 5228
61C2 5229
61C2 5230
61C2 5231
61C2 5232
61C2 5233 7E
61C3 5234 0212'32
61C6 5235
61C6 5236 23
61C7 5237 03C6'22
61CA 5238
61CA 5239 03C6'2A
61CD 5240 00'BB'11
61D0 5241 118E'CD
61D3 5242 03C6'22
61D6 5243
61D6 5244 1116'CD
61D9 5245
61D9 5246 23 D3
61DB 5247 22 D3
61DD 5248
61DD 5249 02'12'21
61E0 5250 35
61E1 5251 15CA'C2
61E4 5252
61E4 5253 00'9E'21
61E7 5254 00'BB'11
61EA 5255 118E'CD
61ED 5256
61ED 5257 1116'C3
61F0 5258
61F0 5259
61F0 5260
61F0 5261
61F0 5262
61F0 5263
61F0 5264
61F0 5265
61F0 5266
61F0 5267
61F0 5268
61F0 5269
61F0 5270
61F0 5271
61F0 5272 0213'32 AF
61F4 5273 18E6'CD
61F7 5274
61F7 5275 OF 0335'3A
164D'D2
61FE 5276
61FE 5277 02C8'3A
6201 5278 B7
6202 5279 1521'C4
6205 5280
6205 5281 83 DB

```

```

: PERFORM_CSR_R
: THIS ROUTINE WILL PUT INSTRUCTIONS IN TO THE CSR AND EXECUTE THEM
:
: INPUT CONDITIONS:
:   H+L = POINTER TO INSTRUCTION TABLE
:   TABLE = CNT,INST,INST,.....
:
:*****

```

```

PERFORM_CSR_R:
MOV     A,M           ;GET COUNT
STA     CSR_CNT      ;AND SAVE

INX     H
SHLD    TEMP_WORD

1$:
LHLD    TEMP_WORD
LXI     D,CSR_VALUE  ;MOVE 3 BYTES OF INST TO CSR_VALUE
CALL    MOVER_3_R
SHLD    TEMP_WORD

CALL    LOAD_CSR_R   ;PUT INST IN CSR

OUT     SETSS
OUT     CLRSS        ;SINGLE STEP CPU

LXI     H,CSR_CNT
DCR     M             ;DECREMENT THE COUNT
JNZ     1$           ;LOOP UNTIL ZERO

LXI     H,X MOV WRWR ;LEAVE MOV INST IN CSR TO
LXI     D,CSR_VALUE  ;ENABLE Y-BUS
CALL    MOVER_3_R

JMP     LOAD_CSR_R   ;LOAD MOV

```

```

:*****
: ATTENTION
: THIS ATTENTION ROUTINE IS USED WHEN THE CPU IS EXECUTING TESTS(RUNNING)
: IT IS BASICLY AN IDLE LOOP FOR THE 8085. DURING THE IDLE LOOP
: THE 8085 CHECKS FOR SPECIAL OCCURANCES SUCH AS ^C,^P,OR CPATN
: IF CPATN SIGNAL COMES HIGH IT MEANS THAT THE CPU MICRO CODED
: DIAGNOSTIC IS REQUESTING SOMETHING. MICRO STEPPING IS DONE ONCE
: FOR EACH ITERATION OF THE IDLE LOOP
:*****

```

```

ATTENTION:
CLEAR  DATA_XFER_FLG
CALL   RESET_TIMEOUT      ;INIT TIMOUTS AND XFER INFO

WHILE  RUNNING

*
*
LDA    MICRO_STEP         ;GET MICRO STEP FLAG
ORA    A                  ;SET CONDITION CODES
CNZ    MICRO_STEPPER      ;CALL IF MICRO STEP

IN     CPATN

```



```

ZZ-ENKAA-2.1 4C00 4
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 19B0'CD *
62E9 5386 *****
62E9 5387
62E9 5388 02 E6 00AF'3A *****

```

N 9

Fiche 1 Frame N9

Sequence 117

```

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

```









```

ZZ-ENKAA-2.1 4C00 4
64BB 5610 005F'32 AF
64BF 5611 0060'32 AF
64C3 5612 030B'32 AF
64C7 5613
64C7 5614 08 E6 005A'3A *****
          18D2'CA *
64CF 5615 1349'CD *
64D2 5616 *****
64D2 5617
64D2 5618 0F 0345'3A *****
          18E3'D2 *
64D9 5619 0383'32 AF *
64DD 5620 03D9'3A *
64E0 5621 03DB'32 *
64E3 5622 *****
64E3 5623
64E3 5624 C9
64E4 5625
64E4 5626
64E4 5627
64E4 5628
64E4 5629
64E4 5630
64E4 5631
64E4 5632
64E4 5633
64E4 5634
64E4 5635 8A 3E
64E6 5636 1435'CD
64E9 5637 04'42'11
64EC 5638 119F'CD
64EF 5639
64EF 5640
64EF 5641 8B 3E
64F1 5642 1435'CD
64F4 5643 04'46'11
64F7 5644 119F'CD
64FA 5645
64FA 5646
64FA 5647 8C 3E
64FC 5648 1435'CD
64FF 5649 04'4A'11
6502 5650 119F'CD
6505 5651
6505 5652
6505 5653 8E 3E
6507 5654 1435'CD
650A 5655 04'4F'11
650D 5656 119F'CD
6510 5657
6510 5658
6510 5659 8F 3E
6512 5660 1435'CD
6515 5661 04'53'11
6518 5662 119F'CD
651B 5663
651B 5664
651B 5665 90 3E
651D 5666 1435'CD
6520 5667 04'57'11

```

F 10

Fiche 1 Frame f10 Sequence 122

```

CLEAR NA_EXP_REC ;CLR NA FLAG FOR CONSOLE TESTS
CLEAR NA_OTHER ;CLR NA FLAG FOR CONSOLE TESTS
CLEAR PACK_ERROR ;CLR WRONG RLO2 PACK FLAG

IFMBI FLAGS,HALT_S_DEF

CALL SET_FOR_CONT_R ;HALT ON ERROR SET,STOP
ENDIF

IF SHORTEN

CLEAR TABLE_MODE ;SHORTEN SET.. USE THIS TEST
LDA TEST_NUM ;NUMBER
STA TEST_NUM2
ENDIF

RET

```

```

:*****
:
: PACK_LABEL
: THIS ROUTINE READS THE EXPECTED AND RECEIVED RLO2 PACK LABELS FROM LS.
: THIS ROUTINE IS ONLY USED IN ASSOCIATION WITH THE IDC PART II DIAGNOSTIC.
:*****

```

```

PACK_LABEL: MVI A,EXPECT_LAB ;GET FIRST 4 BYTES OF EXPECTED
            CALL READ_LS_R ; LABEL FROM LS
            LXI D,EXP_LAB_DAT ;DESTINATION
            CALL MOV_FROM_DATA0 ;MOVE 4 BYTES OF DATA FROM
            ; DATA0

            MVI A,EXPECT_LAB+1 ;GET NEXT 4 BYTES OF EXPECTED
            CALL READ_LS_R ; LABEL FROM LS
            LXI D,EXP_LAB_DAT+4 ;DESTINATION
            CALL MOV_FROM_DATA0 ;MOVE 4 BYTES OF DATA FROM
            ; DATA0

            MVI A,EXPECT_LAB+2 ;GET LAST 4 BYTES OF EXPECTED
            CALL READ_LS_R ; LABEL FROM LS
            LXI D,EXP_LAB_DAT+8 ;DESTINATION
            CALL MOV_FROM_DATA0 ;MOVE 4 BYTES OF DATA FROM
            ; DATA0

            MVI A,RECEIVE_LAB ;GET FIRST 4 BYTES OF RECEIVED
            CALL READ_LS_R ; LABEL FROM LS
            LXI D,REC_LAB_DAT ;DESTINATION
            CALL MOV_FROM_DATA0 ;MOVE 4 BYTES OF DATA FROM
            ; DATA0

            MVI A,RECEIVE_LAB+1 ;GET NEXT 4 BYTES OF RECEIVED
            CALL READ_LS_R ; LABEL FROM LS
            LXI D,REC_LAB_DAT+4 ;DESTINATION
            CALL MOV_FROM_DATA0 ;MOVE 4 BYTES OF DATA FROM
            ; DATA0

            MVI A,RECEIVE_LAB+2 ;GET LAST 4 BYTES OF RECEIVED
            CALL READ_LS_R ; LABEL FROM LS
            LXI D,REC_LAB_DAT+8 ;DESTINATION

```

```

ZZ-ENKAA-2.1 4C00 4
6523 5668 119F'C3
6526 5669
6526 5670
6526 5671
6526 5672
6526 5673
6526 5674
6526 5675
6526 5676
6526 5677
6526 5678
6526 5679 06 3E
6528 5680 1435'CD
6528 5681
6528 5682 02'19'21
652E 5683 124A'CD
6531 5684
6531 5685 00AC'3A
6534 5686 005B'32
6537 5687 11C2'C3
653A 5688
653A 5689
653A 5690
653A 5691
653A 5692
653A 5693
653A 5694
653A 5695
653A 5696
653A 5697
653A 5698 1433'CD
653D 5699
653D 5700 02'A0'21
6540 5701
6540 5702 124A'CD
6543 5703
6543 5704 00AC'3A
6546 5705 005B'32
6549 5706 11C2'C3
654C 5707
654C 5708
654C 5709
654C 5710
654C 5711
654C 5712
654C 5713
654C 5714
654C 5715
654C 5716 1433'CD
654F 5717
654F 5718 04'3D'11
6552 5719 119F'CD
6555 5720
6555 5721
6555 5722 006E'32 3C AF
655A 5723 0F52'C3
655D 5724
655D 5725
655D 5726
655D 5727

```

G 10

Fiche 1 Frame G10

Sequence 123

```

JMP MOV_FROM_DATA0 ;MOVE 4 BYTES OF DATA FROM
; DATA0

```

```

*****
: PRINT DRIVE
: THIS ROUTINE PRINTS THE DRIVE NUMBER THAT AN R80 WAS FOUND ON FOR THE
: IDC DIAGNOSTIC.
*****

```

```

PRINT_DRIVE: MVI A,LS_6
CALL READ_LS_R ;READ SIZE WORD

LXI H,DRIVE_A ;PRINT 'DRIVE '
CALL PRINT_STRING_R

LDA DATA3 ;GET DRIVE NUMBER
STA HEX_BUF
JMP PRINT_HEX_1_R

```

```

*****
: PRINT MM SIZE
: THIS ROUTINE PRINTS THE SIZE OF MM AS CONFIGURED BY THE MICRO DIAGNOSTIC
*****

```

```

PRINT_MM_SIZE: CALL READ_LS_7 ;READ SIZE WORD

LXI H,MEM_SIZE_A ;PRINT THE SIZE OF MEMORY
; IN 256K BLOCKS AS CONFIGURED
CALL PRINT_STRING_R ; BY MICRO DIAGNOSTIC

LDA DATA3
STA HEX_BUF
JMP PRINT_HEX_1_R

```

```

*****
: SET PAR INT
: THIS ROUTINE SETS A PARITY ERROR IN WCS ON REQUEST FROM THE CPU DIAG.
: THE ADDRESS TO SET THE PARITY ERROR ON IS IN LS 7
*****

```

```

SET_PAR_INT: CALL READ_LS_7 ;READ ADDRESS TO SET BAD PARIT

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

SET WCS_BAD_PARITY ;CHANGE PARITY TO BAD
JMP CHARGE_QCS ;WRITE WCS WITH BAD PARITY

```

```

27-ENKAA-2.1 4C00 4
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

```

H 10

Fiche 1 Frame H10

Sequence 124

```

: 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 115 (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 #

```

```

ZZ-ENKAA-2.1  4COG  4
65A7 5788
65A7 5789
65A7 5790
65A7 5791
65A7 5792
65A7 5793
65A7 5794
65A7 5795
65A7 5796
65A7 5797
65A7 5798
65A7 5799 124A'CD
65AA 5800 1433'CD
65AD 5801
65AD 5802 02'CF'21
65B0 5803
65B0 5804 0F 00AC'3A
        19BA'D2
65B7 5805 02'DC'21
65BA 5806
65BA 5807
65BA 5808 124A'C3
65BD 5809
65BD 5810
65BD 5811
65BD 5812
65BD 5813
65BD 5814
65BD 5815
65BD 5816
65BD 5817
65BD 5818 AF D3
65BF 5819
65BF 5820 02 E6 00B0'3A
        19C9'CA
65C7 5821 AE D3
65C9 5822
65C9 5823
65C9 5824 AD D3
65CB 5825
65CB 5826 04 E6 00B0'3A
        19D5'CA
65D3 5827 AC D3
65D5 5828
65D5 5829
65D5 5830 A5 D3
65D7 5831
65D7 5832 08 E6 00B0'3A
        19E1'CA
65DF 5833 A4 D3
65E1 5834
65E1 5835
65E1 5836 AB D3
65E3 5837
65E3 5838 10 E6 00B0'3A
        19ED'CA
65EB 5839 AA D3
65ED 5840
65ED 5841
65ED 5842 0344'32 AF

```

```

*****
: PRINT MOD PRES
: THIS ROUTINE PRINTS WHETHER THE MICRO DIAGNOSTIC FINDS THE FPA, UBE, IDC
: OR R80
*****

```

```

PRINT_MOD_PRES:
CALL PRINT_STRING_R      ;PRINT MODULE NAME
CALL READ_CS_7          ;LS 7 CONTAINS 1 FOR MOD PRES
                        ; AND 0 FOR NOT PRESENT
LXI H,MOD_NOT_PRES_A   ;POINT TO MODULE NOT PRESENT
IF DATA3              ;IF MODULE IS PRESENT
LXI H,MOD_PRES_A       ;POINT TO MODULE PRESENT
ENDIF
JMP PRINT_STRING_R

```

```

*****
: SET SIGNALS
: THIS ROUTINE SETS UP THE SIGNALS AS THEY ARE DESCRIBED IN THE LS
: COMMAND AND STATUS WORD BITS
*****

```

```

SET_SIGNALS:
OUT CLRHLT              ;CLEAR CONSOLE HALT
IFMBI COMMAND1,CONHLT  ;BIT 17
OUT SETHLT              ;SET CONSOLE HALT
ENDIF
OUT CLRPF1              ;CLEAR POWER FAIL INIT
IFMBI COMMAND1,PWRFAIL ;BIT 18
OUT SETPF1              ;SET POWER FAIL INIT
ENDIF
OUT CLRTIM              ;CLEAR INTERVAL TIMER INIT
IFMBI COMMAND1,INVTIM  ;BIT 19
OUT SETTIM              ;SET INTERVAL TIMER INIT
ENDIF
OUT CLRATN              ;CLEAR CONSOLE ATTENTION
IFMBI COMMAND1,CONATT  ;BIT 20
OUT SETATN              ;SET CONSOLE ATTENTION
ENDIF
CLEAR SETACK_FLG       ;CLEAR FLAG TO CLEAR CONSOLE

```

ZZ-ENKAA-2.1 4C00 4

J 10

Fiche 1 Frame J10

Sequence 126

```

65F1 5843
65F1 5844
65F1 5845
65F1 5846 20 E6 00B0'3A *****
          19FE'CA *
65F9 5847 0344'32 3C AF *
65FE 5848 *
65FE 5849 *
65FE 5850 *****
65FE 5851
65FE 5852 029F'32 3C AF
6603 5853
6603 5854 40 E6 00B0'3A *****
          1A0F'CA *
660B 5855 029F'32 AF *
660F 5856 *****
660F 5857
660F 5858 34 D3
6611 5859
6611 5860 10 E6 00AF'3A *****
          1A1B'CA *
6619 5861 35 D3 *
661B 5862 *****
661B 5863
661B 5864 31 D3
661D 5865
661D 5866 20 E6 00AF'3A *****
          1A2F'CA *
6625 5867 10F9'CD *
6628 5868 30 D3 *
662A 5869 1570'CD *
662D 5870 31 D3 *
662F 5871 *****
662F 5872
662F 5873 2E D3
6631 5874
6631 5875 40 E6 00AF'3A *****
          1A3B'CA *
6639 5876 2F D3 *
663B 5877 *****
663B 5878
663B 5879 C9
663C 5880
663C 5881
663C 5882
663C 5883
663C 5884
663C 5885
663C 5886
663C 5887
663C 5888
663C 5889
663C 5890
663C 5891
663C 5892
663C 5893
663C 5894 0F 0283'3A *****
          1A53'DA *
6643 5895 *
6643 5896 47 3E *

```

```

; ACKNOWLEDGE BEFORE RETURNING
; TO WCS
IFMBI COMMAND1,CONACK ;BIT 21
SET SETACK_FLG ;SET FLAG TO SET CONSOLE
; ACKNOWLEDGE BEFORE RETURNING
; TO WCS
ENDIF
SET MEM_REQ ;ALLOW MEMORY REQUEST WHEN
IFMBI COMMAND1,EMEMREF ;BIT 22
CLEAR MEM_REQ ;DONT ALLOW MEM REQ WHEN
ENDIF
OUT INITL ;CLEAR UNIBUS INIT
IFMBI COMMAND0,UBS_INIT ;BIT 28
OUT INITH ;SET UNIBUS INIT
ENDIF
OUT CLRDCI ;CLEAR UNIBUS DC LO
IFMBI COMMAND0,UBS_DC_LO ;BIT 29
CALL NOP_CSR_R ;LOAD CSR WITH A NOP
OUT SETDCI ;SET UNIBUS DC LO
CALL MICRO_STEP_CPU_R ;CLOCK THE SIGNAL INTO MCT
OUT CLRDCI ;CLEAR SIGNAL
ENDIF
OUT CLRBSY ;CLEAR UNIBUS BUSY
IFMBI COMMAND0,UBS_BUSY ;BIT 30
OUT SETBSY ;SET UNIBUS BUSY
ENDIF
RET

```

```

*****
:
: LOOPER
: THIS ROUTINE READS A COUNT FROM THE CPU AND CAUSES THE CPU TO LOOP
: THAT NUMBER OF TIMES BY SETTING THE ADDRESS TO ITS LOOP INSTRUCTION
: WHEN CALLED THE FIRST TIME THIS ROUTINE READS THE COUNT. ON ALL OTHER
: CALLS UNTIL THE COUNT = 0 THE COUNT IS DECREMENTED. WHEN THE COUNT = 0
: THE ADDRESS IS SET FOR NO LOOPING AND THE NEXT CALL IS SET TO READ A
: NEW COUNT
*****

```

```

***** LOOPER: IFN HAVE_CNT
*
*
* MVI A,LOOP_CNT_ADD ;READ LOOP COUNT (ONCE ONLY)

```



```

ZZ-ENKAA-2.1 4C00 4
6684 5954 02C8'3A
6687 5955 B7
6688 5956 1521'C4
668B 5957
668B 5958 83 DB
668D 5959 1F
668E 5960
668E 5961 1AAB'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
66B2 5984
66B2 5985
66B2 5986
66B2 5987
66B2 5988
66B2 5989
66B2 5990
66B2 5991
66B2 5992
66B2 5993 00'18'21
66B5 5994 1581'CD
66B8 5995
66B8 5996 02 E6 005A'3A
1ACF'C2
66C0 5997 02'27'21
66C3 5998 124A'CD
66C6 5999
66C6 6000 0018'2A
66C9 6001 005B'22
66CC 6002 118D'CD
66CF 6003
66CF 6004
66CF 6005 04'74'21
66D2 6006 03B4'22
66D5 6007
66D5 6008 0056'32 AF
66D9 6009
66D9 6010 03'14'21
02 OE 00'58'11
1804'CA 00CF'CD

```

L 10

Fiche 1 Frame L10 Sequence 128

```

LDA MICRO_STEP ;GET MICRO STEP FLAG
ORA A ;SET CONDITION CODES
CNZ MICRO_STEPPER ;MICRO STEP CPU IF SET

IN CPATN
RAR

IFC ;IF CPU_ATTENTION
IN CPUACK
STA ACK_SAVE
CALL RESET_TIMEOUT
CALL STOP_CPU ;STOP SUBROUTINE
MVI B,<^X22> ;ERROR CODE IF CPU ACK SET
LDA ACK_SAVE
RAR
JC ERROR_ROUTINE_NO_RET ;GO TO ERROR ROUTINE IF CPU
; ACK (CARRY) IS SET. DON'T
; RETURN
CALL CHECK_CNTLIC ;ELSE IF ^C TYPED, GO TO PARSER
ENDIF
CALL GCVECT ;CHECK FOR OPERATOR REQUEST
CALL TIMEOUT
ENDWHILE
RET

```

```

:*****
:
: EOP
: THIS ROUTINE IS EXECUTED AT END OF PASS
:*****

```

```

EOP: LXI H,APT_PASS_CNT ;GET PASS CNT IN MAILBOX
CALL INC_WORD_R ;INCREMENT IT
IFNMBI FLAGS,NER_S_DEF ;IF NER DONT PRINT EOP
LXI H,EOP_A ;PRINT 'EOP'
CALL PRINT_STRING_R
LHLD APT_PASS_CNT
SHLD HEX_BUF
CALL PRINT_HEX_2 ;PRINT PASS COUNT
ENDIF
LXI H,SEC_TABLE
SHLD TABLE_PNT ;RESET TABLE POINTER
CLEAR EOS_FLG ;RESET END OF SECTION
IFNEQ PASS_CNT,F_WORD,2 ;-1 PASS CNT -INFINITE,SKIP OUT

```

```

66E7 6011 *
66E7 6012 03'14'21 *
66EA 6013 158C'CD *
66ED 6014 *
66ED 6015 03'14'21 * *****
          02 OE 04'1B'11 *
          1B04'C2 00CF'CD *

```

```

LXI H,PASS_CNT
CALL DECR_WORD_R ;DECREMENT PASS COUNT
IFEQ PASS_CNT,ZERO_WORD,2 ;IF PASS CNT = 0 CAUSE
;RETURN TO PARSER
SET FILE_LOADED ;CAUSE EXIT OF MAIN LOAD LOOP
CLEAR EXECUTE
ENDIF
ENDIF
RET

```

```

66FB 6016 *
66FB 6017 0241'32 3C AF *
6700 6018 023B'32 AF *
6704 6019 * *****
6704 6020 *
6704 6021 * *****
6704 6022 *
6704 6023 C9
6705 6024
6705 6025
6705 6026
6705 6027
6705 6028
6705 6029
6705 6030
6705 6031
6705 6032
6705 6033

```

```

:*****
: BEGIN OF TEST
: THIS ROUTINE HANDLES BEGINING OF TEST SENT FROM THE CPU
: IF ALSO VERIFIES THE TEST SEQUENCE OF WCS TESTING
:*****

```

BEGIN\_OF\_TEST:

```

LXI H,TEST_NUM
INR M ;INCREMENT TEST NUM
IFGT TEST_NUM,TEST_NUM2,1
SET EOS_FLG ;FORCE END OF SECTION
ELSE
LDA TEST_NUM
STA APT_TEST_NUMBER+1 ;STORE NEW TEST # IN MAILBOX
LDA TRACE ;GET TRACE FLAG
ORA A ;SET CONDITION CODES
CNZ PRINT_TRACE ;PRINT SECTION AND TEST #
; IF TRACE SET
ENDIF
CALL VERIFY_SEQ
SET MEM_REQ
RET

```

```

6705 6034 03'D9'21 *****
6708 6035 34 *
6709 6036 *
6709 6037 03'D9'21 *****
          01 OE 03'DB'11 *
          1B1F'F2 00CF'CD *
6717 6038 0056'32 3C AF *
671C 6039 *
671C 6040 1B2C'C3 *****
671F 6041 03D9'3A *
6722 6042 0017'32 *
6725 6043 03DF'3A *
6728 6044 B7 *
6729 6045 1B35'C4 *
672C 6046 *
672C 6047 *****
672C 6048
672C 6049 1B55'CD
672F 6050 029F'32 3C AF
6734 6051 C9
6735 6052
6735 6053
6735 6054
6735 6055
6735 6056
6735 6057
6735 6058
6735 6059
6735 6060
6735 6061 125A'CD
6738 6062 04'DF'21
6738 6063 06 OE
673D 6064 1243'CD
6740 6065 1260'CD
6743 6066 1260'CD

```

```

:*****
: PRINT_TRACE
: THIS ROUTINE PRINT THE CURRENT SECTION AND TEST NUMBER WHEN TRACE IS SET.
:*****

```

PRINT\_TRACE:

```

CALL CRLF_R ;PRINT <CR><LF>
LXI H,SECTION_NAME ;PRINT SECTION NAME LOADED
MVI C,SEC_LEN
CALL PRINTER
CALL SPACE_R
CALL SPACE_R

```

```

ZZ-ENKAA-2.1      4C00      4
6746 6067 03'D3'21
6749 6068 124A'CD
674C 6069 03D9'3A
674F 6070 005B'32
6752 6071 11C2'C3
6755 6072
6755 6073
6755 6074
6755 6075
6755 6076
6755 6077
6755 6078
6755 6079
6755 6080
6755 6081 00C6'2A
6758 6082 03CA'22
675B 6083
675B 6084 C5 46 00'55'21
              77 04 3E
6763 6085
6763 6086 03'CA'21
6766 6087 7E
6767 6088 3F E6
6769 6089 77
676A 6090 158C'CD
676D 6091
676D 6092 35 00'55'21
              70 C1 1B63'C2
6776 6093
6776 6094 006C'32 AF
677A 6095 03D9'3A
677D 6096 006D'32
6780 6097 109D'CD
6783 6098
6783 6099 C5 46 00'55'21
              77 04 3E
678B 6100 00'B6'21
678E 6101 03 OE
6790 6102 10F0'CD
6793 6103 35 00'55'21
              70 C1 1B8B'C2
679C 6104
679C 6105 00B4'3A
679F 6106 3F E6
67A1 6107 00B4'32
67A4 6108
67A4 6109 03'CA'21
              02 OE 00'B4'11
              1BBA'CA 00CF'CD
67B2 6110 23 06
67B4 6111 0F7E'CD
67B7 6112 1349'CD
67BA 6113
67BA 6114
67BA 6115 C9
67BB 6116
67BB 6117
67BB 6118
67BB 6119
67BB 6120

```

N 10

Fiche 1 Frame N10

Sequence 130

```

LXI H,TEST_A
CALL PRINT_STRING_R ;PRINT 'TEST '
LDA TEST_NUM
STA HEX_BUF
JMP PRINT_HEX_1_R ;PRINT TEST #

```

```

*****
: VERIFY_SEQ
: THIS ROUTINE VERIFIES THE TESTS RUN IN SEQUENCE BY READING THE UPC
: AND COMPARING IT TO A KNOW VALUE
*****

```

```

VERIFY_SEQ:  LHL  UPC_SUB
              SHLD TEMP_WORD2 ;SAVE GET UPC VALUE
              DO 4
              *
              *
              LXI H,TEMP_WORD2 ;MASK OFF WCS HIGH PAGE
              MOV A,M
              ANI HEX_3F
              MOV M,A
              CALL DECR_WORD_R ;INC VALUE BY 4
              ENDDO
              CLEAR WCS_ADDRESS
              LDA TEST_NUM ;GET KNOW VALUE FOR THIS TEST
              STA WCS_ADDRESS+1 ;FROM WCS
              CALL READ_WCS_R
              DO 4
              *
              *
              LXI H,WCS_VALUE+2 ;POINT TO WCS DATA
              MVI C,3
              CALL SHIFTER1_R ;SHIFT 17-4(OFF JUMP) TO 21-8
              ENDDO
              LDA WCS_VALUE
              ANI HEX_3F ;AND OUT TO ONLY ADDRESS
              STA WCS_VALUE
              IFNEQ TEMP_WORD2,WCS_VALUE,2
              *
              *
              MVI B,<^X23> ;SEQUENCE ERROR IN WCS TEST
              CALL ERROR_ROUTINE
              CALL SET_FOR_CONT_R ;ALLOW CONTINUE
              ENDDIF
              RET

```

```

*****
: TIMEOUT
: THIS ROUTINE IS USED TO TELL IF THE CPU IS NOT RESPONDING
*****

```

```

27-ENKAA-2.1 4000 4
67BB 6121
67BB 6122
67BB 6123
67BB 6124
67BB 6125
67BB 6126
67BB 6127 01 E6 005A'3A
1BE2'C2
67C3 6128
67C3 6129 03'DD'21
67C6 6130 1581'CD
67C9 6131
67C9 6132 03'DD'21
02 OE 04'1B'11
1BDF'C2 00CF'CD
67D7 6133 24 06
67D9 6134 0F7E'CD
67DC 6135 1349'CD
67DF 6136
67DF 6137
67DF 6138 1BE5'C3
67E2 6139 1341'CD
67E5 6140
67E5 6141
67E5 6142
67E5 6143 C9
67E6 6144
67E6 6145
67E6 6146
67E6 6147
67E6 6148
67E6 6149
67E6 6150
67E6 6151
67E6 6152
67E6 6153 03'DD'21
67E9 6154 AF
67EA 6155 77
67EB 6156 23
67EC 6157 77
67ED 6158 C9
67EE 6159
67EE 6160
67EE 6161
67EE 6162
67EE 6163
67EE 6164
67EE 6165
67EE 6166
67EE 6167
67EE 6168
67EE 6169 F5
67EF 6170 C5
67F0 6171 D5
67F1 6172 E5
67F2 6173 1399'CD
67F5 6174 0208'3A
67F8 6175 3C
67F9 6176 0208'32
67FC 6177

```

```

: IN A REASONABLE TIME PERIOD. IF THE COUNTER OVERFLOWS A TIMEOUT
: HAS OCCURED.
:*****

```

TIMEOUT:

```

IFNMBI  FLAGS,LOOP_S_DEF
*
*
LXI  H,TIMOUT_CNT      ;COUNTER FOR TIMING
CALL  INC_WORD_R      ;INCREMENT COUNTER
*
*
IFEQ  TIMEOUT_CNT,ZERO_WORD,2
*
*
MVI  B,<^X24>         ;OVERFLOW, PRINT ERROR
CALL  ERROR_ROUTINE
CALL  SET_FOR_CONT_R  ;GO TO COMMAND LEVEL
ENDIF
*
*
ELSE
CALL  CHECK_CNTL_C   ;IF ERROR LOOPING, CHECK
                        ; FOR ^C FROM OPERATOR
*
*
ENDIF
*
*
RET

```

```

:*****
:
: RESET TIMEOUT
: THIS ROUTINE RESETS THE TIMEOUT COUNTERS TO ZERO FOR A FULL
: TIMEOUT PERIOD
:*****

```

```

RESET_TIMEOUT:  LXI  H,TIMOUT_CNT
                XRA  A
                MOV  M,A
                INX  H
                MOV  M,A
                RET

```

```

:*****
:
: PARITY ERR
: THIS ROUTINE GETS CONTROL ON A PARITY ERROR INTERRUPT
: THE INTERRUPT COULD ALSO BE KNOWN PARITY ERRORS SUCH AS A SOMM
:*****

```

```

PARITY_ERR:    PUSH  SPSW      ;SAVE FLAGS AND ACCUMULATOR
                PUSH  B        ;SAVE BC PAIR
                PUSH  D        ;SAVE DE PAIR
                PUSH  H        ;SAVE HL PAIR
                CALL  STOP_CPU ;STOP THE CLOCK
                LDA   CNT_PAR  ;GET COUNT OF PARITY ERRORS
                INR   A        ;INC COUNT
                STA   CNT_PAR  ;UPDATE COUNT

```

```

ZZ-ENKAA-2.1 4C00 4
67FC 6178 02 FE
67FE 6179
67FE 6180 1C05'CA
6801 6181
6801 6182 FB
6802 6183 0F47'CD
6805 6184 00C4'2A
6808 6185 005B'22
680B 6186 00'5B'21
680E 6187 158C'CD
6811 6188
6811 6189
6811 6190 0F 039F'3A
1C3C'D2
*****
6818 6191
6818 6192 00'5B'21
02 0E 03'9D'11
1C36'C2 00CF'CD
*****
6826 6193 0B45'CD
6829 6194 03'97'21
682C 6195 124A'CD
682F 6196 FB
6830 6197 1349'CD
6833 6198
6833 6199 1C39'C3
*****
6836 6200
6836 6201 1C5C'CD
*****
6839 6202
6839 6203
*****
6839 6204
6839 6205 1C3F'C3
*****
683C 6206
683C 6207 1C5C'CD
*****
683F 6208
683F 6209
*****
683F 6210
683F 6211 0F 0299'3A
1C4F'D2
*****
6846 6212 00'72'21
6849 6213 00'BB'11
684C 6214 118E'CD
*****
684F 6215
684F 6216
684F 6217 0299'32 AF
6853 6218 13DD'CD
6856 6219
6856 6220 E1
6857 6221 D1
6858 6222 C1
6859 6223 F1
685A 6224
685A 6225 FB
685B 6226 C9
685C 6227
685C 6228
685C 6229
685C 6230
685C 6231
685C 6232
685C 6233

```

1\$:

```

C 11
Fiche 1 Frame C11 Sequence 132
CPI 2 ;SEE IF PARITY COUNT=2 (SECOND
; PARITY ERROR OCCURED)
JZ 1$ ;DON'T TRY SAVE_WR IF SO
EI ;REENABLE INTERRUPTS
CALL SAVE_WR
LHLD SAVED_UPC
SHLD HEX_BUF ;SAVE UPC
LXI H,HEX_BUF ;POINT TO STORED UPC
CALL DECR_WORD_R ;BACK UP UPC..PARITY ERROR
; HAS ALREADY REACHED CSR
IF SOMM_FLG
IFEQ HEX_BUF,SOMM_ADDRESS,2
CALL PRINT_UPC ;PRINT VALUE OF UPC
LXI H,SOMM_A ;PRINT THAT THIS IS A SOMM
CALL PRINT_STRING_R
EI ;REENABLE INTERRUPTS
CALL SET_FOR_CONT_R ;ISSUE PROMPT, ALLOW CONTINUE
ELSE
CALL CHECK_LS_SETPAR ;CHECK TO SEE IF PARITY ERROR
; WAS INTENTIONAL
ENDIF
ELSE
CALL CHECK_LS_SETPAR ;CHECK TO SEE IF PARITY ERROR
; WAS INTENTIONAL
ENDIF
IF LS_PARERR_ACT
LXI H,NOP_INST ;SET UP TO PUT NOP IN CSR
LXI D,SAVED_CSR
CALL MOVER_3_R
ENDIF
CLEAR LS_PARERR_ACT ;CLEAR FLG IN CASE IT IS SET
CALL RESTART_CPU
POP H ;RESTORE HL PAIR
POP D ;RESTORE DE PAIR
POP B ;RESTORE BC PAIR
POP $PSW ;RESTORE FLAGS AND ACCUMULATOR
EI ;REENABLE INTERRUPTS
RET

```

```

: THIS ROUTINE CHECKS TO SEE IF PARITY ERROR WAS CAUSED BY INTENTIONALLY
: PUTTING A PARITY ERROR IN WCS VIA A COMMAND FROM THE CPU MICRO-CODE
:

```

CHECK\_LS\_SETPAR:







```

ZZ-ENKAA-2.1 4C00 4
6961 6406 03'CC'21
6964 6407 04'DE'11
6967 6408 07 OE
6969 6409 11A4'CD
696C 6410
696C 6411 04'DF'21
696F 6412 41 07 11
6972 6413 06 OE
6974 6414 11A4'CD
6977 6415
6977 6416 02'3C'21
697A 6417 41 OD 11
697D 6418 11A2'CD
6980 6419
6980 6420 AF
6981 6421 4113 32
6984 6422 4114 32
6987 6423
6987 6424 0240'32
698A 6425 0408'32
698D 6426 0209'32
6990 6427
6990 6428 80 E6 04DE'3A
1DA3'CA
6998 6429
6998 6430 06 3E
699A 6431 4115 32
699D 6432 70 00 21
69A0 6433
69A0 6434 1DAB'C3
69A3 6435
69A3 6436 07 3E
69A5 6437 4115 32
69AB 6438 00 00 21
69AB 6439
69AB 6440
69AB 6441
69AB 6442 4111 22
69AE 6443
69AE 6444 4150 3A
69B1 6445
69B1 6446 4106 32
69B4 6447
69B4 6448 4117 CD
69B7 6449 A7 D3
69B9 6450
69B9 6451
69B9 6452 00 FE 4116 3A
1DD0'CA
69C1 6453
69C1 6454 AF
69C2 6455 0240'32
69C5 6456 0408'32
69C8 6457 0238'32
69CB 6458
69CB 6459 25 06
69CD 6460 0F79'C3
69D0 6461
69D0 6462
69D0 6463 0241'32 3C AF

```

```

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

```

LOAD\_SECTION:

```

G 11
Fiche 1 Frame G11 Sequence 136
LXI H,TEMP_FLAG ;POINTER TO TEMP SECTION BUFFER
LXI D,SECTION_FLAG ;POINTER TO SECTION BUFFER
MVI C,SEC_LEN+1 ;COUNT
CALL MOVER_R ;MOVE ENTRIES TO LOCAL BUFFER

LXI H,SECTION_NAME
LXI D,SEC_PARMB_NAM
MVI C,SEC_LEN ;MOVE SECTION NAME TO TUSB
CALL MOVER_R ;PARAMETER BLOCK

LXI H,EXT_A ;MOVE EXT TO PARM BLOCK
LXI D,SEC_PARMB_EXT
CALL MOVER_4_R

XRA A ;CLEAR FLAGS
STA SEC_PARMB_ADD+2 ;CLEAR 2 MOST SIGNIFICANT BYTES
STA SEC_PARMB_ADD+3 ; OF SEC_PARMB_ADD (UNUSED)

STA FAST_WRITE ;CLEAR FAST_WRITE
STA WCS_COADED ;CLEAR WCS_COADED
STA CONSOLE_TEST ;CLEAR CONSOLE_TEST

IFMBI SECTION_FLAG,CON_RAM

MVI A,6 ;DRIVER HAS 6 = RAM LOAD
STA SEC_PARMB_DST ;STORE IN DESTINATION CODE
LXI H,RAM_LOAD_ADD

ELSE

MVI A,7 ;DRIVER HAS 7 = WCS LOAD
STA SEC_PARMB_DST ;STORE IN DESTINATION CODE
LXI H,WCS_LOAD_ADD

ENDIF

SHLD SEC_PARMB_ADD ;STORE LOAD ADDR IN PARM BLOCK
LDA SEC_PARMB_DEF ;GET DEFAULT TUSB UNIT (THIS
; UNIT)
STA SEC_PARMB_UNIT ;PARM BLOCK ADDRESS FOR UNIT

CALL TUSB_DRIVER ;GET THE FILE
OUT DISMEM ;DISABLE MEM REF (TUSB DRIVER
; ENABLED IT AFTER LOAD)

IFNEQI SEC_PARMB_ERR,0 ;CHECK FOR TUSB ERROR

XRA A ;CLEAR FLAGS
STA FAST_WRITE ;CLEAR FAST_WRITE
STA WCS_COADED ;CLEAR WCS_COADED
STA EXECUTE ;CLEAR EXECUTE TO ALLOW PARSER
; TO PROMPT

MVI B,<^X25>
JMP ERROR_ROUTINE_NO_RET ;TUSB ERROR. DON'T RETURN

ENDIF

SET FILE_LOADED ;SECTION WAS LOADED OK

```

```

ZZ-ENKAA-2.1 4C00 4
69D5 6464
69D5 6465 80 E6 04DE'3A *****
      1DE5'CA *
69DD 6466 0209'32 3C AF *
69E2 6467 *
69E2 6468 1DF0'C3 *****
69E3 6469 0408'32 3C AF *
69EA 6470 1E8A'CD *
69ED 6471 10F9'CD *
69F0 6472 *
69F0 6473 *****
69F0 6474 *
69F0 6475 C9 *
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 *
69F4 6488 *
69F4 6489 4106 32 *
69F7 6490 411A C3 *
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 *
69FF 6503 06 00 21 *
6A02 6504 7C *
6A03 6505 65 *
6A04 6506 6F *
6A05 6507 00C2'22 *
6A08 6508 1148'CD *
6A08 6509 *
6A08 6510 10F9'CD *
6A0E 6511 04'E5'21 *
6A11 6512 006F'22 *
6A14 6513 AB D3 *
6A16 6514 *
6A16 6515 C5 46 00'55'21 *****
      77 05 3E *
6A1E 6516 *
6A1E 6517 1570'CD *
6A21 6518 1E57'CD *
6A24 6519 *
6A24 6520 0240'3A *
6A27 6521 B7 *

```

H 11

Fiche 1 Frame H11

Sequence 137

```

IFMBI SECTION_FLAG,CON_RAM
SET CONSOLE_TEST
ELSE
SET WCS_LOADED
CALL EXEC_TESTO
CALL NOP_CSR_R
ENDIF
RET

```

```

*****
DIR COMMAND
THIS ROUTINE LOADS THE DEFAULT TUSB UNIT (UNIT THAT THIS TAPE IS IN)
INTO THE PARAMB BLOCK AND EXECUTES THE BOOT BLOCK RESIDENT DIR
COMMAND.
*****

```

```

DIR_COMMAND: LDA SEC_PARB_DEF ;GET DEFAULT TUSB UNIT (THIS
              STA SEC_PARB_UNIT ; UNIT)
              JMP DIR_VECTOR ;PARAM BLOCK ADDRESS FOR UNIT
              ;PRINT DIRECTORY

```

```

*****
VERIFY WRITER
THIS ROUTINE SINGLE STEPS THROUGH THE WRITE DATA 32 MICRO CODE
AND COMPARES IT TO AN EXPECTED VALUE TABLE
*****

```

```

VERIFY_WRITER: SET FAST_WRITE ;ERROR WILL CLEAR FAST WRITE
                LXI H,WCS_DAT_32_ADD
                MOV A,H
                MOV H,L
                MOV L,A
                SHLD UPC_VALUE
                CALL LOAD_UPC_R
                CALL NOP_CSR_R
                LXI H,WRITE_TAB
                SHLD WRITE_TAB_PNT
                OUT CLRATN
                DO 5
                CALL MICRO_STEP_CPU_R
                CALL CHECK_UPC
                LDA FAST_WRITE ;GET FAST WRITE FLAG
                ORA A ;SET CONDITION CODES

```



6A87 6577  
6A87 6578 1148'C3  
6A8A 6579  
6A8A 6580  
6A8A 6581  
6A8A 6582  
6A8A 6583  
6A8A 6584  
6A8A 6585  
6A8A 6586  
6A8A 6587  
6A8A 6588  
6A8A 6589 2A 06  
6A8C 6590 0F5E'CD  
6A8F 6591  
6A8F 6592  
6A8F 6593 0F 02F2'3A  
1EC8'DA  
6A96 6594  
6A96 6595 1DFA'CD  
6A99 6596  
6A99 6597 AF  
6A9A 6598 03A1'32  
6A9D 6599 03A2'32 AF  
6AA1 6600  
6AA1 6601 0213'32 3C AF  
OF 0213'3A  
1EC8'D2  
6AAD 6602  
6AAD 6603 13CE'CD  
6AB0 6604 03DC'32 3C AF  
6AB5 6605 15F0'CD  
6AB8 6606 03DC'32 AF  
6ABC 6607 0F41'CD  
6ABF 6608  
6ABF 6609 00C6'2A  
6AC2 6610 03A1'22  
6AC5 6611  
6AC5 6612 1EA6'C3  
6AC8 6613  
6AC8 6614  
6AC8 6615  
6AC8 6616 C9  
6AC9 6617  
6AC9 6618  
6AC9 6619  
6AC9 6620  
6AC9 6621  
6AC9 6622  
6AC9 6623  
6AC9 6624  
6AC9 6625  
6AC9 6626 03'A5'21  
6ACC 6627 124A'CD  
6ACF 6628  
6ACF 6629 AF  
6AD0 6630 02F6'32  
6AD3 6631 023B'32  
6AD6 6632  
6AD6 6633 C9

JMP LOAD\_UPC\_R

```

*****
EXEC_TESTO
THIS ROUTINE EXECUTES TEST ZERO OF A WCS IMAGE ONLY IF FILE LOADED
IS SET. TEST ZERO IS CONTINUED EACH TIME A DATA REQUEST IS RETURNED
WHEN A DATA REQUEST IS NOT RETURNED THEN EXECUTION OF TEST 0 STOPS
*****

```

```

EXEC_TESTO:  MVI    B,<^X2A>          ;ERROR CODE IF WCS NOT LOADED
              CALL   CHECK_WCS_LOADED ;CHECK FOR WCS LOADED SET
              ;      RETURN IF OK

              IFN    NOINIT          ;ELSE CHECK NO INIT FLAG

              CALL   VERIFY_WRITER    ;CHECK DATA XFER ROUTINE

              XRA    A                ;CLEAR STARTING_UPC
              STA    STARTING_UPC     ;SET TO EXECUTE DATA XFER
              CLEAR  STARTING_UPC+1

              WHILER DATA_XFER_FLG   ;FLAG SET IF DATA XFER EVENT
              ;      ;LOOP STOPS IF NO EVENT

              CALL   START_CPU        ;SET FOR NO RESTART
              SET    TEST_ZERO        ;WAIT FOR CPU ATTENTIONS
              CALL   ATTENTION
              CLEAR  TEST_ZERO
              CALL   RESTORE_WR

              LHLD   UPC_SUB           ;SET FOR CONTINUE AT NEXT ADDR.
              SHLD  STARTING_UPC

              ENDWHILE

              ENDIF

              RET

```

```

*****
SYNTAX_ERROR
THIS ROUTINE PRINTS "SYNTAX ERROR"
*****

```

```

SYNTAX_ERROR: LXI    H,SYNTAX_A
              CALL   PRINT_STRING_R

              XRA    A                ;CLEAR FLAGS
              STA    NOT_FOUND        ;CLEAR NOT FOUND
              STA    EXECUTE          ;CLEAR EXECUTE SO PARSER WILL
              ;      ;PROMPT FOR COMMAND

              RET

```

```

ZZ-ENKAA-2.1 4C00 4
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
6800 6684 0339'32
6803 6685 033D'22
6806 6686 EB
6807 6687 033B'22
680A 6688 60
680B 6689 69
680C 6690 0337'22
680F 6691 00 00 21
6812 6692 39
6813 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  
 6B19 6696 124A'CD  
 6B1C 6697  
 6B1C 6698 0336'3A  
 6B1F 6699 005B'32  
 6B22 6700 11C2'CD  
 6B25 6701  
 6B25 6702 0339'3A  
 6B28 6703 005B'32  
 6B2B 6704 11C2'CD  
 6B2E 6705  
 6B2E 6706 0357'2A  
 6B31 6707 005B'22  
 6B34 6708 11BD'CD  
 6B37 6709  
 6B37 6710 0338'2A  
 6B3A 6711 005B'22  
 6B3D 6712 11BD'CD  
 6B40 6713  
 6B40 6714 033D'2A  
 6B43 6715 005B'22  
 6B46 6716 11BD'CD  
 6B49 6717  
 6B49 6718 033F'2A  
 6B4C 6719 005B'22  
 6B4F 6720 11BD'CD  
 6B52 6721  
 6B52 6722 2022'C3  
 6B55 6723  
 6B55 6724  
 6B55 6725  
 6B55 6726  
 6B55 6727  
 6B55 6728  
 6B55 6729  
 6B55 6730  
 6B55 6731  
 6B55 6732  
 6B55 6733  
 6B55 6734 001B CD  
 6B58 6735  
 6B58 6736  
 6B58 6737  
 6B58 6738  
 6B58 6739  
 6B58 6740 00 00 21  
 6B58 6741 2B  
 6B5C 6742 7C  
 6B5D 6743 B5  
 6B5E 6744 1F5B'C2  
 6B61 6745  
 6B61 6746 10 3E  
 6B63 6747 30  
 6B64 6748  
 6B64 6749 0401'32 3C AF  
 6B69 6750 1F6D'C3  
 6B6C 6751  
 6B6C 6752  
 6B6C 6753

```

LXI H,BREAK_A
CALL PRINT_STRING_R

LDA SAVE_A ;PRINT VALUE OF A
STA HEX_BUF
CALL PRINT_HEX_1_R

LDA SAVE_C ;PRINT VALUE OF CARRY
STA HEX_BUF
CALL PRINT_HEX_1_R

LHLD SAVE_BC ;PRINT VALUE OF BC PAIR
SHLD HEX_BUF
CALL PRINT_HEX_2

LHLD SAVE_DE ;PRINT VALUE OF DE PAIR
SHLD HEX_BUF
CALL PRINT_HEX_2

LHLD SAVE_HL ;PRINT VALUE OF HL PAIR
SHLD HEX_BUF
CALL PRINT_HEX_2

LHLD SAVE_SP ;PRINT VALUE OF SP PAIR
SHLD HEX_BUF
CALL PRINT_HEX_2

JMP PARSER
    
```

```

:*****
:
: POWER_REC
: THIS ROUTINE IS BRANCHED TO ON A POWER RECOVERY
: IT WILL CALL THE ROM CODE THAT DOES POWER UP SEQUENCING AND WILL
: DO AN INITIALIZE IF THERE IS WCS CODE LOADED
:*****
POWER_REC: CALL POWER_SEQ ;CALL ROM POWER SEQ ROUTINE

: THE FOLLOWING DELAY LOOP
: IS TO ALLOW AC LOW TO
: SETTLE BEFORE RE-ENABLING
: INTERRUPTS
: INIT COUNTER IN H&L TO 0
1$: DCX H ;DECREMENT COUNTER
MOV A,H ;GET HIGH BYTE RESULT
ORA L ;'OR' WITH LOW BYTE
JNZ 1$ ;REPEAT IF NOT 0

MVI A,HEX_10 ;A GETS A 10(H)
SIM ;RESET 7.5

SET WARM_RESTART ;INDICATE POWER FAIL RESTART
JMP START_UP
:*****
:
    
```

```

ZZ-ENKAA-2.1  4C00  4
6B6C 6754
6B6C 6755
6B6C 6756
6B6C 6757
6B6C 6758
6B6C 6759
6B6C 6760
6B6C 6761
6B6C 6762
6B6C 6763 C9
6B6D 6765
6B6D 6766
6B6D 6767
6B6D 6768
6B6D 6769
6B6D 6770
6B6D 6771
6B6D 6772
6B6D 6773
6B6D 6774
6B6D 6775
6B6D 6776
6B6D 6777
6B6D 6778
6B6D 6779 40 80 31
6B70 6780
6B70 6781 F3
6B71 6782
6B71 6783 1399'CD
6B74 6784 AF
6B75 6785 02C8'32
6B78 6786 3C
6B79 6787 029F'32
6B7C 6788
6B7C 6789 1ED7'CD
6B7F 6790 02F0'3A
6B82 6791 0306'32
6B85 6792
6B85 6793 03'F5'21
6B88 6794 124A'CD
6B88 6795
6B88 6796 AF
6B8C 6797 003A'32
6B8F 6798 0231'32
6B92 6799
6B92 6800 023B'32
6B95 6801 006E'32
6B98 6802 3C
6B99 6803 0064'32
6B9C 6804
6B9C 6805 00'7D'21
6B9F 6806 15C2'CD
6BA2 6807
6BA2 6808 04 0E 00'A9'21
        OD 23 77 AF
        1FAB'C2
6BAE 6809 FF 3E
6BB0 6810 1415'CD
6BB3 6811
6BB3 6812 030C'32 3C AF

```

```

: TIMER_VEC
:
: THIS ROUTINE WILL HANDLE A INTERVAL TIMER INTERRUPT IF THE INTERRUPT
: IS NOT MASKED OUT. AT PRESENT, THIS ROUTINE WILL NOT BE ENTERED AND
: DOES NOTHING. THIS ROUTINE WILL BE WRITTEN AT A LATER TIME IF NECES-
: SARY.
:.....
TIMER_VEC:      RET                      ;DO A RETURN
:.....
: MAIN LINE ROUTINES
:.....
: START UP
: THINGS THAT ARE ONLY DONE AT THE BEGINING
:.....
START_UP:      LXI      $SP,PRI_STACK
:
:                   DI                      ;DISABLE INTERUPTS
:
:                   CALL     STOP_CPU       ;STOP CPU IN CASE IT IS RUNNING
:                   XRA      A
:                   STA      MICRO_STEP    ;CLEAR MICRO STEP MODE
:                   INR      A
:                   STA      MEM_REQ       ;ENABLE MEM REQ FOR LATER
:
:                   CALL     CHECKSUM
:                   LDA      NEWSUM
:                   STA      OLDSUM        ;SAVE CHECKSUM VALUE
:
:                   LXI      H,VERSION     ;PRINT VERSION NUMBER
:                   CALL     PRINT_STRING_R
:
:                   XRA      A
:                   STA      CONTINUE      ;INIT CONTINUE FLAG
:                   STA      ERROR        ;INIT ERROR FLAG
:
:                   STA      EXECUTE      ;CLEAR EXECUTE
:                   STA      WCS_BAD_PARITY ;SET FOR GOOD PARITY
:                   INR      A
:                   STA      PARITY_ON    ;INIT PARITY CALC
:
:                   LXI      H,X CLR_PAGE  ;CLR WCS HIGH PAGE
:                   CALL     PERFORM_CSR_R
:
:                   ZERO     DATA0,4     ;RESET OUT OF COMPAT. MODE
:
:                   MVI      A,HEX_FF
:                   CALL     WRITE_LS_R
:
:                   SET      PAR_ON       ;ENABLE WCS PARITY FOR LATER

```













```

ZZ-ENKAA-2.1  4C00  4
6DCD 7153
6DCD 7154
6DCD 7155
6DCD 7156
6DCD 7157
6DCD 7158
6DCD 7159
6DCD 7160
6DCD 7161
6DCD 7162
6DCD 7163
6DCD 7164
6DCD 7165
6DCD 7166
6DCD 7167
6DCD 7168
6DCD 7169 03D9'32
6DD0 7170 003A'32 3C AF
6DD5 7171
6DD5 7172 4129 CD
6DD8 7173
6DD8 7174 1341'CD
6DDB 7175
6DDB 7176
6DDB 7177 01 3E
6DDD 7178 0011'32
6DE0 7179
6DE0 7180 03'DA'21
        01 0E 03'D9'11
        21F6'F2 00CF'CD
6DEE 7181 006A'32 3C AF
6DF3 7182
6DF3 7183 221E'C3
6DF6 7184
6DF6 7185 006A'32 AF
6DFA 7186
6DFA 7187 03'D9'21
        01 0E 03'DB'11
        2211'F2 00CF'CD
6E08 7188 0056'32 3C AF
6E0D 7189 C9
6E0E 7190
6E0E 7191 221E'C3
6E11 7192 03D9'3A
6E14 7193 0017'32
6E17 7194 03DF'3A
6E1A 7195 B7
6E1B 7196 1B35'C4
6E1E 7197
6E1E 7198
6E1E 7199
6E1E 7200
6E1E 7201
6E1E 7202 C9
6E1F 7203
6E1F 7204
6E1F 7205
6E1F 7206
6E1F 7207
6E1F 7208

```

```

: CONSOLE TEST SUBROUTINES
:

```

```

: *****
: CON_BEG!N TEST_R THIS ROUTINE IS USED FOR BEGINNING OF TEST FOR CONSOLE
: BASED CPU TESTS ONLY.
:
: INPUT CONDITIONS:
: A=TEST NUMBER
: *****

```

```

CON_BEGIN_TEST_R:
      STA TEST_NUM           ;SAVE TEST NUMBER
      SET CONTINUE          ;SET UP TO CONTINUE IF TEST
                           ; STOPPED BY OTHER THAN ^P
                           ;CHECK FOR OPERATOR REQUEST
      CALL GCVECT
      CALL CHECK_CNTL      ;IF ^C, GO BACK TO PARSER AND
                           ; ALLOW CONTINUE
      MVI A,1
      STA APT_MESSAGE_CODE+1 ;SET APT START FLAG
      IFGT TEST_NUM1,TEST_NUM,1
      SET SKIP_TEST
      ELSE
      CLEAR SKIP_TEST
      IFGT TEST_NUM,TEST_NUM2,1
      SET EOS_FLG           ;FORCE END OF SECTION
      RET
      ELSE
      LDA TEST_NUM           ;GET TEST NUMBER
      STA APT_TEST_NUMBER+1 ;SAVE IN APT MAILBOX
      LDA TRACE              ;GET TRACE FLAG
      ORA A                  ;SET CONDITION CODES
      CNZ PRINT_TRACE        ;PRINT SECTION AND TEST #
                           ; IF SET
      ENDIF
      ENDIF
      RET

```

```

: *****
: CON_ERROR_R THIS ROUTINE IS EXECUTED TO PRINT AN ERROR FOUND BY A

```

CONSOLE BASED CPU TEST.

```

ZZ-ENKAA-2.1 4C00 4
6E1F 7209
6E1F 7210
6E1F 7211
6E1F 7212
6E1F 7213
6E1F 7214 3C D3
6E21 7215 223C'CD
6E24 7216
6E24 7217
6E24 7218 171D'CD
6E27 7219
6E27 7220 005A'3A
6E2A 7221 1F
6E2B 7222 2239'D2
6E2E 7223
6E2E 7224 0057'32 3C AF
6E33 7225
6E33 7226 4129 CD
6E36 7227 1341'CD
6E39 7228 3D D3
6E3B 7229 C9
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
6E3F 7242
6E3F 7243 0044'22
6E42 7244
6E42 7245 00'4B'11
6E45 7246
6E45 7247
6E45 7248 C5 46 00'55'21
77 04 3E
6E4D 7249
6E4D 7250 0044'2A
6E50 7251
6E50 7252 2283'CD
6E53 7253 0044'22
6E56 7254
6E56 7255 35 00'55'21
70 C1 224D'C2
6E5F 7256
6E5F 7257 00'4F'21
6E62 7258
6E62 7259 0053'22
6E65 7260
6E65 7261 00'4B'11
6E68 7262
6E68 7263
6E68 7264 C5 46 00'55'21
77 04 3E
6E70 7265

```

```

CON_ERROR_R: OUT CLRLIT ;TURN RUN LITE OFF
CALL MSK_ERROR_R ;MASK THE EXPECTED AND RECEIVED
; DATA TO ELIMINATE ANY BITS
; WHICH ARE NOT OF INTEREST.
CALL WCS_ERROR ;CALL MICMON ROUTINE TO PRINT ERROR
; REPORT
LDA FLAGS ;DO NOT SET ERROR CON FLAG IF LOOP
RAR ; ON ERROR IS NOT SET
JNC 1$
SET EPROR_CON ;SET FLAG SAYING ERROR HAPPENED AND
; LOOP ON ERROR WAS SET
CALL GCVECT ;CHECK FOR OPERATOR INPUT
CALL CHECK_CNTL C ;CHECK FOR CONTROL C INPUT
1$: OUT SETLIT ;TURN RUN LITE BACK ON
RET

```

```

MSK_ERROR_R THIS ROUTINE MASKS DATA FROM BOTH THE CON EXP_DAT AND
CON REC DAT AREAS BY USING THE ERROR MASK DATA. A ONE
IN THE ERROR MASK INDICATES THAT THAT PARTICULAR BIT IS
NOT OF INTEREST AND SHOULD BE SET TO ZERO.

```

```

MSK_ERROR_R: LXI H,CON_EXP_DAT ;MOVE MEMORY ADDRESS CON_EXP_DAT
; INTO THE H,L PAIR.
SHLD CON_EXP_ADD ;STORE THE ADDRESS OF THE EXPECTED
; DATA IN CON_EXP DAT.
LXI D,CON_MSK_DAT ;MOVE MEMORY ADDRESS CON_MSK_DAT
; INTO THE D,E PAIR.
DO 4 ;LOOP 4 TIMES TO MASK 4 BYTES OF
; EXPECTED DATA.
LHLD CON_EXP_ADD ;MOVE THE MEMORY ADDRESS OF THE BYTE
; OF EXPECTED DATA INTO THE H,L PAIR.
CALL MSK_BYTE ;CALL ROUTINE TO MASK 1 BYTE
SHLD CON_EXP_ADD ;STORE THE MEMORY ADDRESS OF THE
; EXPECTED DATA INTO CON_EXP_ADD.
ENDDO
LXI H,CON_REC_DAT ;MOVE MEMORY ADDRESS CON_REC_DAT
; INTO THE H,L PAIR.
SHLD CON_REC_ADD ;STORE THE ADDRESS OF THE RECEIVED
; DATA IN CON_REC ADD.
LXI D,CON_MSK_DAT ;MOV MEMORY ADDRESS CON_MSK_DAT
; INTO THE D,E PAIR.
DO 4 ;LOOP 4 TIMES TO MASK 4 BYTES OF
; RECEIVED DATA.

```

```

22-ENKAA-2.1 4C00 4
6E70 7266 0053'2A
6E73 7267
6E73 7268 2283'CD
6E76 7269 0053'22
6E79 7270
6E79 7271 35 00'55'21
70 C1 2270'C2

6E82 7272
6E82 7273 C9
6E83 7274
6E83 7275 46
6E84 7276
6E84 7277 1A
6E85 7278
6E85 7279 2F
6E86 7280
6E86 7281 A0
6E87 7282
6E87 7283 77
6E88 7284
6E88 7285 23
6E89 7286 13
6E8A 7287
6E8A 7288 C9
6E8B 7289
6E8B 7290
6E8B 7291
6E8B 7292
6E8B 7293
6E8B 7294
6E8B 7295
6E8B 7296
6E8B 7297
6E8B 7298
6E8B 7299
6E8B 7300
6E8B 7301
6E8B 7302
6E8B 7303
6E8B 7304 22D7'CD
6E8E 7305 23 D3
6E90 7306 22 D3
6E92 7307
6E92 7308 A2 D3
6E94 7309
6E94 7310 00'72'21
6E97 7311 00'B4'11
6E9A 7312 118E'CD
6E9D 7313
6E9D 7314 18 06
6E9F 7315 00'B6'21
6EA2 7316 03 0E
6EA4 7317
6EA4 7318 87 DB
6EA6 7319 1F
6EA7 7320 7E
6EA8 7321 17
6EA9 7322 77
6EAA 7323 2B
6EAB 7324 0D

```

```

*
*
*
*
*
*****

```

I 12

Fiche 1 Frame I12 Sequence 151

```

LHLD CON_REC_ADD ;LOAD THE ADDRESS OF THE RECEIVED
; DATA INTO THE H,L PAIR.
CALL MSK_BYTE ;CALL ROUTINE TO MASK ONE BYTE
SHLD CON_REC_ADD ;STORE THE ADDRESS OF THE RECEIVED.
; DATA IN CON_REC_ADD.

ENDDO

RET

MSK_BYTE: MOV B,M ;MOVE A BYTE OF EXPECTED DATA INTO
; REGISTER B.
LDAX D ;MOVE A BYTE OF MASK DATA INTO
; THE ACCUMULATOR.
CMA ;CHANGE THE MASK POLARITY BY
; COMPLEMENTING THE ACCUMULATOR.
ANA B ;"AND" THE COMPLEMENTED MASK WITH
; THE REGISTER B CONTENTS.
MOV M,A ;MOVE THE MASKED EXPECTED DATA BACK
; INTO THE EXPECTED DATA LOCATION.
INX H ;INCREMENT THE D,E AND H,L REGISTER
INX D ;PAIRS TO POINT AT THE NEXT BYTES
; OF EXPECTED AND RECEIVED DATA.
; DONE

```

```

*****
WCS_READ_R THIS ROUTINE READS DATA FROM WCS AT THE LOCATION GIVEN BY
WCS ADDRESS. THE DATA IS PUT INTO THE CSR AND SHIFTED OUT
INTO WCS_VALUE FOR CHECKING. AS THE CSR IS READ, IT IS
LOADED WITH A NOP INSTRUCTION.

THIS ROUTINE IS USED BY THE CONSOLE TESTS AND IS TAYLORED
FOR FAST EXECUTION RATHER THAN VERSITILITY.
*****

```

```

WCS_READ_R: CALL LD_UPC_R ;LOAD UPC WITH WCS ADDRESS
OUT SETSS ;SINGLE STEP MACHINE TO LOAD
OUT CLRSS ; WCS CONTENTS INTO THE CSR.

OUT VSHIFT

LXI H,NOP_INST ;POINT TO NOP INSTRUCTION
LXI D,WCS_VALUE ;POINT TO WCS VALUE AREA
CALL MOVER_3_R ;LOAD WCS_VALUE WITH NOP INST

MVI B,24 ;LOOP COUNT IN REGISTER B
1$: LXI H,WCS_VALUE+2 ;POINT TO END OF WCS_VALUE AREA
MVI C,3 ;BYTE COUNT

IN CSR23 ;GET CSR BIT 23
RAR ;ROTATE CSR 23 INTO CARRY
MOV A,M ;MOV WCS_VALUE INTO ACCUMULATOR
RAL ;CARRY GOES IN BIT 0
MOV M,A ;MOV ACCUMULATOR BACK TO MEM
DCX H ;POINT TO NEXT BYTE
DCR C ;DEC BYTE COUNT

```

```

77-ENKAA-2.1 4C00 4
6EAC 7325 22A7'C2
6EAF 7326
6EAF 7327 38 D3
6EB1 7328 22B6'D2
6EB4 7329 39 D3
6EB6 7330
6EB6 7331 25 D3
6EB8 7332
6EB8 7333 24 D3
6EBA 7334
6EBA 7335 05
6EBB 7336 229F'C2
6EBE 7337
6EBE 7338 A3 D3
6ECO 7339
6ECO 7340 C9
6EC1 7341
6EC1 7342
6EC1 7343
6EC1 7344
6EC1 7345
6EC1 7346
6EC1 7347
6EC1 7348
6EC1 7349
6EC1 7350 22D7'CD
6EC4 7351
6EC4 7352 00'B6'21
6EC7 7353
6EC7 7354 7E
6EC8 7355 08 D3
6ECA 7356 2B
6ECB 7357
6ECB 7358 7E
6ECC 7359 09 D3
6ECE 7360 2B
6ECF 7361
6ECF 7362 7E
6ED0 7363 0A D3
6ED2 7364
6ED2 7365 37 D3
6ED4 7366 36 D3
6ED6 7367
6ED6 7368 C9
6ED7 7369
6ED7 7370
6ED7 7371
6ED7 7372
6ED7 7373
6ED7 7374
6ED7 7375
6ED7 7376 006C'2A
6EDA 7377 00C2'22
6EDD 7378
6EDD 7379 00'C3'21
6EE0 7380 02 0E
6EE2 7381 10F0'CD
6EE5 7382
6EE5 7383
6EE5 7384 A2 D3

```

J 12

Fiche 1 Frame J12 Sequence 152

```

JNZ 2$ ;REPEAT UNTIL 3 BYTES DONE
OUT CLRCSR ;CLEAR CSR INPUT
JNC 6$ ;DON'T SET INPUT IF CARRY CLEAR
OUT SETCSR ;ELSE SET CSR INPUT
6$: OUT SETCSK ;CLOCK CSR TO SHIFT IT AND ALSO LOAD
; NOP INSTRUCTION
OUT CLRCSK ;CLEAR UPC CLOCK
DCR B ;DECREMENT SHIFT COUNT
JNZ 1$ ;REPEAT UNTIL 24 BITS SHIFTED
OUT Vnorml ;RETURN TO NORMAL MODE.
RET

```

```

:*****
: WCS_WRITE_R THIS ROUTINE WRITES DATA FROM WCS_VALUE TO WCS AT THE
: LOCATION GIVEN BY WCS_ADDRESS.
:*****

```

```

WCS_WRITE_R: CALL LD_UPC_R ;LOAD UPC WITH WCS ADDRESS
LXI H,WCS_VALUE+2 ;POINT AT THE LOW BYTE.
MOV A,M ;MOVE THE LOW BYTE TO WCSB0.
OUT WCSB0
DCX H
MOV A,M ;MOVE THE NEXT BYTE TO WCSB1.
OUT WCSB1
DCX H
MOV A,M ;MOVE THE HIGH BYTE TO WCSB2.
OUT WCSB2
OUT SETWCK ;WRITE TO WCS.
OUT CLRWCK
RET

```

```

:*****
: LD_UPC_R THIS ROUTINE LOADS THE UPC WITH WCS_ADDRESS
:*****

```

```

LD_UPC_R: LHLD WCS_ADDRESS ;UPC DATA
SHLD UPC_VALUE ;STORE DATA IN UPC_VALUE
LXI H,UPC_VALUE+1 ;POINT TO END OF UPC_VALUE AREA
MVI C,2 ;BYTE COUNT
CALL SHIFTER1_R ;LEFT JUST. 15 BITS WITH COUNT
; IN REGISTER C
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
6EFO 7392 77
6EF1 7393 2B
6EF2 7394 0D
6EF3 7395 22EE'C2
6EF6 7396
6EF6 7397 A8 D3
6EF8 7398 22FD'D2
6EF8 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

```

K 12

Fiche 1 Frame K12 Sequence 153

```

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 BYTE
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 INSTRUCTION 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 VNORMAL ;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!!!
:*****

```

SPSW	=	00000006		
SSP	=	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_DT_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_CD_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_ADDR		000001B4	R	02
B	=	00000000		
BEGIN_CHECKSUM		000006D6	R	02
BEGIN_OF_TEST		00001B05	R	02
BEG_TST	=	00000080		
BELC_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
BOOTER	=	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_ADDR		000001CC	R	02
C	=	00000001		
CALC_ADD		00000206	R	02
CC_ADDR	=	000000FE	R	02
CHANGE_WCS		00000F52	R	02
CHECKSUM		00001ED7	R	02

CHECK_CNTLC		00001341	R	02
CHECK_ERR_10		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_BECL		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
CLRACK	=	000000A9		
CLRACL	=	00000033		
CLRATN	=	000000AB		
CLRBSY	=	0000002E		
CLRCLK	=	00000020		
CLRCCK	=	00000024		
CLRCR	=	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		0000003B	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_NOM		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
CSRO7	=	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		
ENBPARG	=	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

ZZ-ENKAA-2.1 Symbol table  
.MAIN.  
Symbol table

N 12  
14-JUN-1983  
Fiche 1 Frame N12  
Sequence 156  
14-JUN-1983 10:27:37 VAX-11 Macro V03-00  
14-JUN-1983 10:27:02 DRBO:[GREEN.CRD]ENKAA.MAC;57  
Page 229  
(1)

EXAM_PATT	00000BF8	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	000008A4	R	02
EXAM_TEMP	00000239	R	02
EXAM_UBS	00000898	R	02
EXAM_UPC	00000B2C	R	02
EXAM_WCS	00000BCA	R	02
EXAM_WRK	00000B4E	R	02
EXECOTE	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
GLVECT	= 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
IDC_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	RR	02
INPUT_NUM	0000043D	RR	02
INPUT_NUM_CHECK	00000F6B	RR	02
INPUT_NUM_IF	00000FCF	RR	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  
 MFP\_A 0000014A R 02  
 MICFLAG 0000002E R 02  
 MICRO\_STEP 000002C8 R 02  
 MICRO\_STEPPER 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\_DATA0\_HEXBUF 0000119C R 02  
 MOV\_FROM\_DATA0 0000119F R 02  
 MOV\_INPUT\_DATA0 00001193 R 02  
 MOV\_LOAD\_SECTION 00001D61 R 02  
 MRB0\_A 0000015E R 02  
 MRLO2\_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 = 0000008D  
 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 000018EE 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
RBO_A            00000329 R    02
RBO_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_LAB      = 0000008E
REC_LABEL        0000044E R    02
REC_LAB_DAT      0000044F R    02
REMOTE           = 00000006
REPEAT           00000333 R    02
REPEAT_COMMAND   00000AD6 R    02
RESET_TIMEOUT    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          00000338 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
SECTION_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
SETACK           = 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_BELC         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      0000198D R    02
SET_SOMM         000009AB 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	00001518	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
SLVECT1	= 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_CPO	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	0000038B	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	000018BB	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	= 00000048		
TTDB	= 00000048		
TIMODE	= 0000004A		
TTSR	= 00000049		
TUSB_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_INTT	= 00000010		
UPCT4	= 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

ZZ-ENKAA-2.1 Symbol table  
 .MAIN.  
 Symbol table

E 13  
 14-JUN-1983

Fiche 1 Frame E13

Sequence 160

14-JUN-1983 10:27:37 VAX-11 Macro V03-00 Page 233  
 14-JUN-1983 10:27:02 DRB0:[GREEN.CRD]ENKAA.MAC;57 (1)

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		0000006E	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		00000066	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_WRRWR		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

ZZ-ENKAA-2.1 Psect synopsis  
 .MAIN.  
 Psect synopsis

F 13  
 14-JUN-1983  
 Fiche 1 Frame F13  
 14-JUN-1983 10:27:37 VAX-11 Macro V03-00  
 14-JUN-1983 10:27:02 DRBO:[GREEN.CRD]ENKAA.MAC;57  
 Sequence 161  
 Page 234  
 (1)

-----  
 ! Psect synopsis !  
 -----

PSECT name	Allocation	PSECT %n.	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
1	F13	161	Psect synopsis
1	G13	162	Directory

14-JUN-1983  
14-JUN-1983  
14-JUN-1983