

The main body of the page is a grid of 14 columns and 14 rows of small, illegible text fragments. Each cell in the grid contains a small, rectangular block of text, which appears to be a snippet of code or a small diagram. The text is too small and faded to be read, but the overall layout suggests a structured data table or a series of small diagrams arranged in a grid.



IDENTIFICATION

Product code: ZZ-ECKAA-8.7
 Product name: ECKAA807 VAX 11/750 MICRO DIAGNOSTIC MONITOR
 Product date: APRIL 1986
 Maintainer: BASE SYSTEMS DIAGNOSTIC ENGINEERING

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) 1980,1986 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

d	i	g	i	t	a	l
---	---	---	---	---	---	---

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	MONITOR COMMANDS	5
5.1.1	Diagnose	5
5.1.2	Examine	7
5.1.3	Show Flags	8
5.1.4	Set Flag	9
5.1.5	Clear Flag	9
5.1.6	Set Stop-on-Micromatch	10
5.1.7	Clear Stop-on-Micromatch	10
5.1.8	Set Control File	11
5.1.9	Clear Control File	12
5.1.10	Set Step Instruction	12
5.1.11	Set Step Cycle	13
5.1.12	Set Step Tick	13
5.1.13	Show Visibility Bus	14
5.1.14	Continue	14
5.1.15	Loop	15
5.1.16	Return	15
5.2	OPTIONS	15
5.3	EVENT FLAGS	16
6.0	PROGRAM FUNCTIONAL DESCRIPTION	16
6.1	Program Overview	16
6.2	Program Size	16
6.3	Program Run Times	17
6.4	Run-time Dynamics	17
6.5	Fault Detection	17
6.6	Performance During Hardware Failures	18
6.7	Program Applications	18
6.8	Test Descriptions	18
7.0	MAINTENANCE HISTORY	18

1.0 ABSTRACT

This program is the heart of the VAX-11/750 micro diagnostic package. It provides the mechanism for loading, controlling, and monitoring each micro diagnostic. This program resides in 8085 - RAM on the RDM module (L0006).

The micro monitor is responsible for the parsing and execution of commands typed in at the terminal. It controls the loading of each individual micro diagnostic overlay from the TU58 console drive into RAM memory on the RDM module.

2.0 HARDWARE REQUIREMENTS

This program requires the RDM module (L0006) installed in the CPU backplane, a local terminal, and a working TU58 drive. It is, however, necessary to have the CPU present to make use of the monitor for its purpose in executing micro diagnostics.

There is no restriction to the monitor when adding hardware to the system.

3.0 SOFTWARE REQUIREMENTS

This micro monitor runs as a standalone program residing in the RDM 8085 - RAM memory. It does not require or use the resident micro code. Any required micro code will be loaded with each separate micro diagnostic test.

4.0 PREREQUISITES

The only prerequisites are that the RDM selftest, that is ran when the machine is powered up, executes with no errors, and that the TU58 console drive is working.

5.0 OPERATING INSTRUCTIONS

The following section describes the commands that can be performed under the micro monitor. Several commands control the microdiagnostic monitor. The monitor may be entered either directly or indirectly. To enter directly from the RDM console command mode (RDM> prompt), type TE/C. To enter indirectly, type TE and then await for the micro diagnostic to complete. To enter directly during test execution, type CTRL/C. The microdiagnostic program enters the monitor automatically if it finds either a TU58 drive error or a hardware error. When the monitor is waiting for a command, it prompts with MIC>.

Any numeric values (addresses and data) must be in hex. Any values enclosed in brackets ([]) is optional. Anytime that an exclamation point is used to describe a command it means exclusive OR.

The list below contains legal command op codes:

Diagnose Loop Set Clear Show Return Continue Examine

This list below contains legal command keyword arguments:

TEST	VA(virtual address)
VBUS(visibility bus)	MA(memory address)
LOOP	PC(program counter)
STEP	PB(PC backup)
PASS	RT(R Temps)
NER(no error report)	MT(M Temps)
CYCLE(one microcycle)	PS(processor status longword)
BELL	MD(memory data)
TICK(one-half microcycle)	WD(write data)
FLAG	SR(status and control registers)
HALT	SF(status flags)
SOMM(stop-on micromatch)	INSTRUCTION
CONTINUE	CF(control file)
IB(inhibit burst)	QA(quality assurance)
TR(trace flag)	QV(quick verify)
DM(diagnostic module)	SB(single bit errors)

5.1 MONITOR COMMANDS

5.1.1 Diagnose -

The Diagnose command is used to initiate micro diagnostic execution. It initializes the program control flags and starts program execution. The execution of these diagnostics can be controlled by the following arguments:

```
TE[ST] CO[NTINUE]          Q[UICK]V[ERIFY]  
PA[SS] S[INGLE]B[IT ERRORS] D[IAGNOSTIC]M[ODULE]
```

The Diagnose command initializes the flag setting as follows:

Flag	Initial Setting
LOOP	clear
NER	clear
BELL	clear
HALT	set
IB	clear
QA	no change
TR	no change

If Diagnose is type without any arguments, the monitor executes all tests on the installed tape once. Control then returns to the monitor (MIC> prompt).

The format of the Diagnose command is as follows:

```
DI[TE:<test_number> [[<test_number> ]![[CO]] ]!SB !QV  
PA:<pass_count> !DM]
```

The following are some examples of the Diagnose command:

```
MIC> DI
```

Execute all tests on the installed tape once.

Diagnose Test

The TEST argument may be used with the Diagnose command in three ways.

1. Diagnose TEST followed by a single test number executes the specified test indefinitely. Type CTRL/C to escape from the loop and return to the monitor.

2. Diagnose TEST followed by a test number and Continue executes the specified test and all following tests on the

installed tape cartridge.

3. Diagnose TEST followed by two test numbers executes tests between and including the two numbers specified. Control then returns to the monitor. If the first test number typed is out of range, the program types an error message and returns control to the monitor.

MIC> DI TE:2

Test 2 loops indefinitely. Use CTRL/C to stop execution and return to the MIC> prompt.

MIC> DI TE:2 CO

Start with test 2 and run remaining tests on the installed tape.

MIC> DI TE:3 6

Execute tests 3,4,5,6 then return to the MIC> prompt.

Diagnose Pass

The PASS argument may be used with the Diagnose command to set the pass count to a specified number. The default pass count is 1. A pass count of 0 causes the program to run indefinitely. At the end of the specified number of passes, control returns to the monitor (MIC> prompt).

MIC> DI PA:2

Execute all tests on the installed tape for two passes then return to the monitor.

Diagnose Quick Verify

The QV (quick verify) argument may be used with the Diagnose command to run a predetermined subset of tests on the microdiagnostic tape. It was designed for use by DIGITAL manufacturing groups.

MIC> DI QV

Execute all tests on the installed tape that are selected for Quick Verify only.

Diagnose Single Bit errors

The SB (single bit errors) argument enables the reporting of single bit errors from the microdiagnostic. If this argument is specified the microdiagnostic will report the number of single bit errors found when running the memory tests. If this argument is not specified, then there will be no report of single bit errors.

MIC> DI SB

Execute all tests on the installed tape and when running the memory tests, report the number of single bit errors found in the array.

Diagnose Diagnostic Module

The DM (diagnostic module) argument may be used with the diagnose command to run those tests that check out the diagnostic module only. This command should be used only if the VAX-11/750 is working properly.

MIC> DI DM Execute the RDM micro diagnostic from the tape installed in the TUS8 drive.

5.1.2 Examine -

The Examine command is used to examine various registers and flags within the 750 CPU. The format of the command is:

EX <name_list>[:<reg_number>]

The name list and reg_number are as follows:

name list	reg number	
VA		virtual address register
MA		memory address register
PC		program counter
PB		PC backup register
RT	0 --> 2F	RTemp registers
MT	0 --> 0F	MTemp registers
PS		processor status longword
MD		memory data register
WD		write data register
SR	0 --> 0E	status & control registers
SF		status flags

ST step counter

MIC> EX VA

Examine the contents of the Virtual Address register.

MIC> EX RT:C

Examine the contents of RTemp number C.

5.1.3 Show Flags -

This command displays the current states of program control flags. The format of the command is as follows:

SH FL

The current program control flags are listed below:

HALT Halt - This flag calls the monitor when error is detected and error message has been typed.

LOOP Loop on error - This flag is useful only in the event of a program detected error. Flag may be set with the Loop command, or manually by typing SE FL LO. When this flag is set and program detects an error, the test loops on minimum amount of code necessary to recreate the error. type CTRL/C to escape from the loop and return to the monitor.

NER No error report - If flag is set, program does not report errors.

BELL Bell on error - If flag is set, program rings terminal bell on first occurrence of an error and on every fifth occurrence.

QA Quality assurance - If flag is set, program responds to each test as if it had detected a failure.

IB Inhibit burst flag - If this flag and the loop flag are set, the program does not try to loop on DCS microcode only. Rather, it loops between ERRLOOP and IFERROR pseudo instructions.

TR Trace flag - If flag is set, monitor types test names as well as test numbers.

Example:

```
MIC> SH FL
```

```
FLAGS SET: LO,NE,BE,  
FLAGS CLEAR: HA,IB,QA,TR
```

5.1.4 Set Flag -

This command sets (enables) any of the program control flags. The format of the command is as follows:

```
SE FL <flag_name_list>
```

For a description of the flag_name_list, see the section on SHOW FLAG command.

```
MIC> SE FL NE
```

Sets the No Error report flag.

```
MIC> SE FL LO BE
```

Sets the Loop and Bell flags.

5.1.5 Clear Flag -

This command clears (disables) any of the program control flags. The format of the command is as follows:

```
CL FL <flag_name_list>
```

For a description of the flag_name_list, see the section on SHOW FLAG command.

```
MIC> CL FL HA
```

Clears the Halt flag.

5.1.6 Set Stop-on-Micromatch -

This command stops execution of code in DCS at the specified address. DCS addresses range from 0 to 3F. Add 1800 to the desired DCS address. Type the Continue command after the MIC> prompt to proceed from your current halt, after typing SE SO <cs_address> to get to that address. The format of the command is as follows:

```
SE SO:<cs_address>
```

Example:

```
MIC> SE SO:1803  
MIC> CO  
MICRO BREAK MATCH, CS ADDR = 1803, DCS ADDR = 03  
MIC>
```

This command causes the machine to stop at micro address 1803 (DCS address 03). After the command stops execution, the MIC> prompt is issued and the monitor will await your command.

5.1.7 Clear Stop-on-Micromatch -

This command clears the stop-on-micromatch function. Add 1800 to the desired DCS address to create a scope sync pulse at that address. If <cs_address> is specified, a scop sync pulse is generated on slot 6, pin C81 when the current address matches <cs_address>. The pulse occurs with the M clock that marks the beginning of the specified microcycle. The format of the command is as follows:

```
CL SO[:<cs_address>]
```

Example:

```
MIC> CL SO:1803
```

This command clears the stop-on-micromatch function and loads 1803 into the RDM match register for a scope sync.

5.1.8 Set Control File -

This command sets the specified <bit_number> in the control file of the specified <dc_s_address>. The format for the command is as follows:

```
SE CF:<dc_s_address> <bit_number>
```

The control file bit functions are defined as follows:

- | | |
|-----------------|--|
| Bit <0> VSTB | This bit strobes visibility bus at rising edge of the next M clock |
| Bit <1> DCSACL | This bit clears DCS address register. |
| Bit <2> STPCLK | This bit stops the VAX-11/750 CPU clock. |
| Bit <3> ENTRACE | This bit latches the trace register at the rising edge of the next M clock. When this bit is not set, the trace register monitors control store address lines continuously. |
| Bit <4> HRWBUS | This bit reads W-BUS by strobing the WH register at the rising edge of the next M clock. |
| Bit <5> WBUSDR | This bit enables the WD register onto W-BUS. |
| Bit <6> STBCMI | This bit reads CMI data by strobing MH register with CMI data at rising edge of the next M clock. |
| Bit <7> SATRIG | This bit inhibits clocking signature analyzer on the next M clock. This signal connects to a post on the module for use by signature analyzer (backplane pin C73 on slot 6). |

Example:

```
MIC> SE CF:3 4
```

Set control file bit 4 (HRWBUS) in the DCS address of 03.

5.1.9 Clear Control File -

This command clears the specified bits in the control file of the <dc_s_address>. The format of the command is as follows:

```
CL CF:<dc_s_address> <bit_number>
```

For a description of <bit_number>, see the section on SET CONTROL FILE.

Example:

```
MIC> CL CF:3 4
```

Clear control file bit 4 (HRWBUS) in the DCS address of 03.

5.1.10 Set Step Instruction -

This command steps through the pseudo-instructions (8085 code) in the current test. If <test_pc> is specified, the step function does not start until the instruction at the <test_pc> address is ready to execute. If <test_pc> is not specified, the step function begins at the next pseudo instruction of the current test, following a Loop or Continue command.

In either case, when the program stops at the next pseudo instruction, press the space bar to continue stepping through the pseudo instructions. Type a carriage return to escape from the step mode back to the monitor. The format of the command is as follows:

```
SE ST IN[:<test_pc>]
```

Example:

```
MIC> SE ST IN  
MIC> DI  
TPC = 001F
```

This example causes the step mode to be enabled and after typing DI to start program execution, the micro diagnostic stops in the first pseudo instruction. The typing of a space bar will cause the diagnostic to step one pseudo instruction.

5.1.11 Set Step Cycle -

This command steps through DCS microinstructions one CPU machine cycle at a time (M clock). After entering in this command, the Continue, Loop, or Diagnose command must be typed to begin stepping. After the program stops at the first DCS address in the current test, press the space bar to step through the test. Type a carriage return to escape and return control to the monitor. The format of the command is as follows:

```
SE ST CY
```

Example:

```
MIC> SE ST CY  
MIC> DI  
DCS ADDR = 00
```

This example sets the cycle step mode and then starts execution of the diagnostic. When the first DCS microword is fetched for execution (DCS address 00), the monitor will stop execution, print the address on the terminal, and await for a space bar or a carriage return.

5.1.12 Set Step Tick -

This command steps through DCS micro instructions, one B clock at a time. After entering this command, the Continue, Loop, or Diagnose command must be typed to begin stepping. After the program stops at the first DCS address in the current test, press the space bar to step through the test. Type a carriage return to escape and return control to the monitor. The format of this command is as follows:

```
SE ST TI
```

Example:

```
MIC> SE ST TI  
MIC> DI  
DCS ADDR = 00
```

5.1.13 Show Visibility Bus -

This command strobes the visibility bus (VBUS) and prints out the current signal states. This command is most useful after executing the Initialize pseudo instruction, and when stepping through microcode in DCS. Show Visibility Bus displays the 40 bits of the visibility bus, with the low order bit on the right. The format of this command is as follows:

SH VB

Example:

```
MIC> SH VB  
VBUS = 1110100010111010000100001011111010010111  
MIC>
```

5.1.14 Continue -

This command continues testing after the program stops, following error detection or CTRL/C. This command does not modify any flags. The format of this command is as follows:

CO

Example:

```
MIC> DI  
ECKAA-V5.00 MIC(L0003)-V00.05  
01,  
?ERROR:0030 TEST:01 SUBTEST:01  
  
DATA:      AAAAAAAAA  
           AAABAAAA  
           00000001
```

```
MIC> CO  
02,03,04,05,06,07,08,09,0A,0B,0C,  
END OF PASS 01
```

MIC>

This example shows that after an error in the micro diagnostic, the test can be resumed by typing the Continue command at the MIC> prompt after the error message has completed.

5.1.15 Loop -

This command puts the program in an error loop after it detect and reports an error. The command clears the HALT flag, sets the NER and LOOP flags, and performs a Continue command function. The format of the command is as follows:

LO

Example:

MIC> DI

ECKAA-V5.00 MIC(L0003)-V00.05

01.

?ERROR:0030 TEST:01 SUBTEST:01

DATA: AAAAA
 AAA8AAAA
 00000001

MIC> LO

In this example, when the Loop command is typed after the error, the diagnostic will loop on test 01 - subtest 01. To stop the loop type CTRL/C.

5.1.16 Return -

This command returns the microdiagnostic monitor to the RDM console command mode (RDM> prompt). The format of the command is as follows:

RE

Example:

MIC> RE

RDM>

5.2 OPTIONS

Not applicable

5.3 EVENT FLAGS

Not applicable

6.0 PROGRAM FUNCTIONAL DESCRIPTION

6.1 Program Overview

The monitor is loaded from the TU58 console drive by the RDM software then control is transferred to it. There are two ways that the monitor will take over.

The first is by using the TE command from the RDM prompt. When this is done, the monitor will start the execution of the micro diagnostic, that is in the TU58, for one pass. When the first pass is complete, control is passed to the command parser, which, issues the MIC> prompt.

The second is by using the TE/C command from the RDM prompt. When this is done, the command parser is entered and the MIC> prompt is issued. The command parser will wait for a command to be given.

The command parser takes characters entered on the terminal (as commands) and evaluates them. It has a table of possible commands that can be issued and if the command typed on the terminal is not supported then an error message is typed at the terminal. If the command entered is supported (one that is located in the table) then the parser dispatches to the appropriate routine. When the parser sees the Diagnose command typed on the terminal the parser will call the appropriate sub-routines to read the tu58. After the tu58 is read, control is passed to the Opcode Dispatch routine.

The Opcode Dispatch routine will interpret the test data (pseudo opcodes) and dispatch to the appropriate routine. After each routine has executed control is returned to the dispatcher. The Opcode dispatcher will continue to read test overlays from the tu58 drive until the last test has been reached and executed. At this point control is returned to the parser and the MIC> prompt is issued. Control can be either returned to the RDM or another micro diagnostic can be executed.

6.2 Program Size

The monitor is approximately 11.1 kbytes in length.

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:
test_pc, test_number, subtest_number, expected_data,
received_data, indexes, r_temps, gatearrays, modules.

?ERROR: test_pc TEST: test_number SUBTEST: subtest_number

DATA: expected_data
received_data
indexes
r_temps (0-F)

FAILING GATE ARRAYS: gate_arrays

FAILING MODULES: modules

test_pc Location of failure within test overlay.
test_number Test number that failed.
subtest_number Subtest number with the test that failed.
expected_data Expected data from test code.
received_data Received data from test code.
indexes Loop I, J, and K (dependent on test).
rtemps CPU-RTEMPS 0:F (dependent on test).
gatearrays Failing gate array callout list.
modules Failing module callout list.

6.6 Performance During Hardware Failures

If the RDM hardware starts failing then monitor should not be used until this problem is resolved. If a power failure takes place the monitor will have to be reloaded and restarted.

6.7 Program Applications

This program must be used to run any micro diagnostics on the VAX-11/750. Its application depends on the reason for executing the micro diagnostics. This 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 operational goodness of the 750 cpu on a periodic or as needed basis.

6.8 Test Descriptions

Not Applicable.

7.0 MAINTENANCE HISTORY

- 1-00 Don Monroe 16-JUN-80
Initial Release
- 1-01 Don Monroe 26-JUN-80
Added pass count option to QA Flag. Conditions the QA flag on a specified pass count.
- 1-02 Don Monroe 3-Jul-80
Fixed bug in BURST CLOCK routine related to stepping the clock with the LOOP & ERROR flags set and an INHIBIT on the Burst Clock call.
- 1-03 Don Monroe 3-Jul-80
Fixed bug in typeout of Test PC.
- 1-04 Don Monroe 23-Jul-80
Added a feature to BURST CLOCK that enables the detection of the clock being hung. Added CMI BUS name to module name list.
- 1-05 Don Monroe/Bill Landry 27-AUG-80
Altered monitor to load DCS with NOPS before using any micro-code.
- 1-06 Don Monroe 9-Sep-80
Modified LOADREG routine so that if the register is the RD Control Register, the state of the front panel lights is copied into it.
- 1-07 Don Monroe 10-Sep-80
Added the ERRLOG and DUMPLOG pseudo instructions. Also added

- the CMA (M8728) to the module callout list.
- 1-08 Don Monroe 16-Sep-80
Fixed bugs in DUMP LOG routine.
- 1-09 Don Monroe 16-Sep-80
Fixed bugs in ERROR LOG routine. Added setting and clearing
of error flags in ERROR LOG routine.
- 1-10 Bill Landry 19-Sep-80
Fixed bug in ERROR LOG routine to clear INIT FLAG after first
pass.
- 1-11 Don Monroe 23-Sep-1980
Fixed more bugs in error log routine.
- 2-00 Don Monroe 29-Sep-1980
Second Release
- 2-01 Don Monroe 7-Oct-80
Modified Gate Array list to support bit slice algorithm in
error report routine. Added the bit slice algorithm.
- 2-02 Don Monroe 20-Oct-80
Fixed SHOW VBUS routine to ensure ctrl file is not holding
vbus in load state.
- 2-03 Don Monroe 17-Nov-1980
Fixed bug in CHECK_QV routine.
- 3-00 Don Monroe 17-NOV-1980
Third Release
- 3-01 Don Monroe 29-Dec-1980
Added DI DM command to invoke only the tests for the RDM.
- 3-02 Don Monroe 30-Dec-1980
Fixed bugs.
- 3-03 Don Monroe 30-Dec-1980
Fixed bugs.
- 3-04 Don Monroe 31-Dec-1980
Fixed bugs.
- 4-00 Don Monroe 5-Jan-1980
Added code to enable the clock interrupts if key switch is
in remote position.
- 4-01 Don Monroe 17-Feb-81
Added PRINT_N and PRINT_S pseudo instructions.
Added NOTEQUAL condition to SKIP pseudo instruction.
- 5-00 Don Monroe 2-Mar-1981
Release.
- 5-01 Bill Landry 19-MAY-1981
Added code to fix ERROR_LOG ROUTINE. Routine would take
fail exit if entry was already in log.
- 5-02 Don Monroe 21-May-81
Started work on EXAMINE command.
- 5-03 Don Monroe 3-Jun-81
Fixed bugs in EXAMINE routine.
- 5-04 Don Monroe 8-Jun-81
Fixed bugs in EXAMINE routine.
Added an initialize of DCS on entry to the monitor.
- 5-05 Don Monroe 8-Jun-81
Moved the clock stop code from program init to entry
point into monitor so EXAMINE's would work on TE/C cmd.
- 5-06 Don Monroe 12-Jun-81
Fixed the RDM_WDR micro word.

- 5-07 Don Monroe 12-Jun-81
Fixed mask on examine of SF and SC.
- 6-00 Don Monroe 23-Jun-81
Changed revision for release.
- 6-01 Don Monroe 23-Jul-81
Changed SET STEP CYCLE/TICK routines to clear the
other flag. Modified typeout routine to automatically
insert a Carriage Return Line Feed after 80 characters.
- 6-02 Don Monroe 24-Aug-81
Added new Gate Array names for FPA module.
- 6-03 Don Monroe 2-Sep-81
Masked PSL data on PSL EXAMINE.
- 6-04 Dave Spain 4-Sep-81
Added missing FPA gate arrays.
- 6-05 Don Monroe 8-Sep-81
Fixed bug in FPA bit-sliced name tables.
Added ENABLE_STALL routine and modified BRSTCLOCK routine
to support it.
- 6-06 Don Monroe 22-Sep-81
Fixed bug in PSL Examine. The AND with immediate mask used
the wrong instruction.
- 6-07 Don Monroe 24-Sep-81
Modified Single Bit Error Messages. Changed Load Overlay
Routine to do an Initialize before trying to load cache
and other internals.
- 7-00 Don Monroe 10-Nov-81
Added SBE keyword and flag to DIAGNOSE command to enable
the DUMP_LOG routine to print the state of the error log.
- 7-01 Dave Spain 28-Jan-82
Added L0016 memory controller and MAD gate array, and changed
CMC and CMA names. L0011 controller is now MC0, L0016 is MC1.
M8728 256K array card is now MA0, M8750 1M array card is MA1.
Changed callouts of FCS, FMR and FFA to callout all slices.
- 7-02 Dave Spain 1-Feb-82
Fixed bug in FCS, FMR and FFA callouts introduced in 7-01.
- 7-03 Dave Spain 3-Feb-82
Replaced references to LOWER 256 KB in correctable bits
messages to ARRAY 0.
- 7-04 Dave Spain 19-Feb-82
Removed CMC and CMA module references from error callouts.
Now calls out Memory Controller and Memory Array Cards under
the L and M module numbers only.
- 7-05 Dave Spain 8-Mar-82
Fixed bug in ERROR_LOG routine. Wasn't checking test address
against top of error log correctly.
- 7-06 Dave Shull 10-Mar-82
Changed CRD messages (LOG_MSG_2 & LOG_MSG_3) to reference
the memory array card as ARRAY versus ARRAY 0. This allows
sharing of message for either array 0, array 1, etc.
- 7-07 Dave Shull 22-Mar-82
Changed the DUMP_LOG routine so it would set the ERR_LOG_INIT
flag. This way the combinations of ERRLOG and DUMPL0G pseudo's
can be used more than once within the same micro to check more
than one array.

- 7-08 Dave Shull 06-Apr-82
Changed the TYPE_ERROR and TYPE_GA_LIST routines so they would print the failing Gate Array(s) and Module(s) in column.
- 8-00 Dave Shull 08-Apr-82
Removed the BEGNSA and ENDSA pseudo execution code which included the following:
- Removed symbol SA_ADDRESS:
- Removed BEGIN_SA and END_SA from INSTR_TABLE:
- 8-01 Dave Spain 05-May-82
Changed bit-slice tables for FPA gate arrays to enhance isolation for error callouts.
- 8-02 Dave Shull 11-May-82
Changed the LOAD_INDEX: routine so that it will handle the case where there is an address specified that points to the value to be loaded into the index, with No index specified for that address.
- 8-03 Dave Shull 03-Jun-82
Fixed the COMPARE_VBUS: routine so it would work with an index greater than 80 (hex). Prior to this fix, values greater than 80 (hex) would not get processed correctly.
- 8-04 Dave Shull 14-Jun-82
Changed CMK callout to CML for new gate array.
- 8-05 Dave Shull 03-Nov-82
Changed the CCS module callout from L0005 to L0008. This is for the new PCS module.

! Object Module Synopsis !

Module Name	Ident	Bytes	File	Creation Date	Creator
ECKAA	V08.05	11358	WRKD\$0:(SHULL.COMET.MICMON)ECKAA.OBJ;	17-DEC-1982 16:38	VAX-11 Macro V03-00

! Program Section Synopsis !

Psect Name	Module Name	Base	End	Length	Align	Attributes
. BLANK .	ECKAA	00000000	00002C5D	00002C5E (11358.)	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC
		00000000	00002C5D	00002C5E (11358.)	BYTE 0

! Symbols By Name !

Symbol	Value	Symbol	Value	Symbol	Value	Symbol	Value
ADD_NAME	00000949-R						
ALP_NAME	000006F9-R						
BIT_SLICE_TBL	000006DB-R						
FAC_NAME	00000B84-R						
FCS_NAME	00000B96-R						
FEX_NAME	00000B50 R						
FFH_NAME	00000C2F R						
FFL_NAME	00000BF2-R						
FIC_NAME	00000B6F-R						
FIO_NAME	00000AD7-R						
FMR_NAME	00000BD3-R						
MDL_NAME	000009D5-R						
MDR_NAME	00000860-R						
MEC_NAME	000009A6-R						
SRM_NAME	000007B2-R						
UDP_NAME	00000A32-R						

Key for special characters above:

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

! Image Synopsis !

Virtual memory allocated: 00000000 00002DFF 00002E00 (11776. bytes, 23. pages)
Stack size: 0. pages
Image binary virtual block limits: 1. 23. (23. blocks)
Image name and identification: ECKAA V08.05
Number of files: 1.
Number of modules: 1.
Number of program sections: 2.
Number of global symbols: 16.
Number of image sections: 1.
Image type: SYSTEM.
Map format: DEFAULT in file WRKD\$0:[SHULL.COMET.MICMON]ECKAA.MAP;806
Estimated map length: 10. blocks

! Link Run Statistics !

Performance Indicators	Page Faults	CPU Time	Elapsed Time
-----	-----	-----	-----
Command processing:	30	00:00:00.07	00:00:00.12
Pass 1:	25	00:00:00.21	00:00:00.47
Allocation/Relocation:	4	00:00:00.05	00:00:00.22
Pass 2:	24	00:00:02.19	00:00:03.11
Map data after object module synopsis:	7	00:00:00.06	00:00:00.10
Symbol table output:	1	00:00:00.01	00:00:00.11
Total run values:	91	00:00:02.59	00:00:04.13

Using a working set limited to 300 pages and 29 pages of data storage (excluding image)

Total number object records read (both passes): 176
of which 0 were in libraries and 2 were DEBUG data records containing 41 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 0 global symbol table records was written

/SYST=0/EXE=ECKAA.EXE;805 ECKAA

Table of contents

(2)	419	DECLARATIONS
(3)	852	EQUATED SYMBOLS
(4)	863	"EQUATED SYMBOLS
(5)	866	"LINKAGE FROM RDM SYSTEM
(6)	1434	"OWN STORAGE
(7)	2215	"PROGRAM INITIALIZATION ROUTINE
(8)	2402	"COMMON ROUTINES
(8)	2409	READ TU58
(9)	2481	OPEN FILE ON TU58
(10)	2530	" TYPE TU58 ERROR MESSAGE
(11)	2576	TYPE ASCII ROUTINE
(12)	2640	TYPE NUMERIC (NIBBLE, BYTE, WORD, OR LONG)
(13)	2714	TERMINAL INTERRUPT SERVICE ROUTINE
(14)	2757	WRITE DCS ROUTINE
(15)	2853	SINGLE MICRO INSTRUCTION ROUTINE
(16)	2910	" SINGLE CLOCK TICK ROUTINE
(17)	2961	" ABORT ROUTINE
(18)	3011	SET DCS CONTROL FILE ROUTINE
(19)	3072	CLEAR DCS CONTROL FILE ROUTINE
(20)	3131	" READ OVERLAY ROUTINE
(21)	3429	CHECK QV ROUTINE"
(22)	3491	READ V BUS ROUTINE
(23)	3540	DIRECTORY SEARCH ROUTINE
(24)	3730	TYPE TEST NUMBER ROUTINE
(25)	3867	DESELECT DCS ROUTINE
(26)	3899	SELECT DCS
(27)	3922	START DCS EXECUTION ROUTINE
(28)	3989	CMI DISABLE ROUTINE
(29)	4020	CMI ENABLE ROUTINE
(30)	4051	INIT CPU ROUTINE
(31)	4084	" FETCH CS LOCATION 0
(32)	4108	" TYPE FLAGS ROUTINE
(33)	4140	" FILL DCS ROUTINE
(34)	4164	"COMMAND PARSER
(34)	4165	GET COMMAND LINE
(35)	4286	PARSE COMMAND LINE
(36)	4397	SEARCH COMMAND/KEYWORD LIST ROUTINE
(37)	4481	SKIP SPACES ROUTINE
(38)	4540	PUT D REGISTER IN PARSE TABLE ROUTINE
(39)	4577	PARSE HEX NUMBER ROUTINE
(40)	4708	ACTION ROUTINES
(40)	4709	DIAGNOSE ACTION ROUTINE
(41)	4902	SHOW ACTION ROUTINE
(42)	5021	CONTINUE ACTION ROUTINE
(43)	5064	RETURN ACTION ROUTINE
(44)	5082	LOOP ACTION ROUTINE
(45)	5112	" SET ACTION ROUTINE
(46)	5307	" CLEAR ACTION ROUTINE
(47)	5436	EXAMINE ACTION ROUTINE
(48)	5665	DECODE FLAG NAME ROUTINE
(49)	5728	THE INTERPRETER
(49)	5729	OP CODE DISPATCH ROUTINE
(50)	5863	LOAD DCS ROUTINE
(51)	5934	NEW TEST ROUTINE
(52)	6080	INITIALIZE ROUTINE
(53)	6132	LOOP ROUTINE
(54)	6224	END LOOP ROUTINE

(55)	6349	ERROR LOOP ROUTINE
(56)	6377	PATTERN GENERATE ROUTINE
(57)	6515	LOAD REGISTER ROUTINE
(58)	6589	SKIP ROUTINE
(59)	6692	SUB TEST ROUTINE
(60)	6735	FETCH ROUTINE
(61)	6794	MASK ROUTINE
(62)	6871	COMPARE REGISTER ROUTINE
(63)	6956	COMPARE REGISTER MASKED ROUTINE
(64)	7086	COMPARE V BUS ROUTINE
(65)	7205	IF ERROR ROUTINE
(66)	7342	BURST CLOCK ROUTINE
(68)	7797	END TEST ROUTINE
(69)	7835	END PASS ROUTINE
(70)	7865	LOAD FIELD ROUTINE
(71)	7998	LOAD INDEX ROUTINE
(72)	8083	INCREMENT INDEX ROUTINE
(73)	8115	SAVE INDEX ROUTINE
(74)	8150	ENABLE MICRO TRAP ROUTINE
(75)	8177	ENABLE MICRO STALL ROUTINE
(76)	8204	ERROR LOG ROUTINE
(77)	8368	PRINT STRING ROUTINE
(78)	8395	PRINT NUMERIC ROUTINE
(79)	8430	DUMP ERROR LOG ROUTINE
(80)	8468	INTERPRETER SUBROUTINES
(80)	8469	INITIALIZE LOOP VALUE ROUTINE
(81)	8511	GET INDEX VALUE ROUTINE
(82)	8589	MULTIPLY A TIMES B&C ROUTINE
(83)	8636	SAVE BAD DATA ROUTINE
(84)	8689	SAVE GOOD DATA ROUTINE
(85)	8742	COMPARE GOOD AND BAD ROUTINE
(86)	8812	RING BELL ROUTINE
(87)	8855	TYPE ERROR ROUTINE
(88)	9106	SINGLE PSEUDO INSTRUCTION ROUTINE
(89)	9171	INSERT FIELD ROUTINE
(90)	9361	CALCULATE PARITY ROUTINE
(91)	9462	CONVERT BIT NUMBER TO BIT MASK ROUTINE
(92)	9493	CHECK VBUS ROUTINE
(93)	9551	CHECK QA FLAG
(94)	9603	TYPE GATE ARRAY LIST
(95)	9671	BIT SLICE ALGORITHM
(96)	9813	READ RTEMP REGISTER
(97)	9849	DISPLAY DATA
(98)	9870	READ MSRC FIELD

```
0000 1 ;DEBUG = 0
0000 2 .TITLE ECKAA VAX-11/750 MICRO DIAGNOSTIC MONITOR
0000 3 .IDENT /V08.07/
00000008 0000 4 PRIM_REV = 8 ; If these go GTR 9 then change GENERATE_TITLE
00000007 0000 5 SEC_REV = 7 ; macro (remove leading 0).
0000 6
0000 7 ;
0000 8 ; COPYRIGHT (C) 1980,1982
0000 9 ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
0000 10 ;
0000 11 ; THIS SOFTWARE IS FURNISHED UNDER A LICENSE FOR USE ONLY ON A SINGLE
0000 12 ; COMPUTER SYSTEM AND MAY BE COPIED ONLY WITH THE INCLUSION OF THE
0000 13 ; ABOVE COPYRIGHT NOTICE. THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
0000 14 ; MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
0000 15 ; EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
0000 16 ; TERMS. TITLE TO AND OWNERSHIP OF THE SOFTWARE SHALL AT ALL TIMES
0000 17 ; REMAIN IN DEC.
0000 18 ;
0000 19 ; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 20 ; AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 21 ; CORPORATION.
0000 22 ;
0000 23 ; DEC ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 24 ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
0000 25 ;
0000 26
0000 27 ;++
0000 28 ;
0000 29 ; FACILITY: VAX 11/750 Micro Diagnostic Package
0000 30 ;
0000 31 ; FUNCTIONAL DESCRIPTION:
0000 32 ;
0000 33 ;
0000 34 ; ENVIRONMENT: Stand alone.
0000 35 ;
0000 36 ; AUTHOR: Donald W. Monroe, CREATION DATE:
0000 37 ;
0000 38 ; MODIFIED BY:
0000 39 ; 0-2 Don Monroe 9 Aug 79
0000 40 ; Fixed load of RTEMP data. QA flag was missing from ASCII
0000 41 ; list. Also fixed macro package to create the test files
0000 42 ; correctly for RTEMP, CACHE, and DCS data.
0000 43 ; 0-3 Don Monroe 9 Aug 79
0000 44 ; Added load of CACHE data. Added disable of CMI in INIT routine.
0000 45 ; 0-4 Don Monroe 14 Aug 79
0000 46 ; Fixed the FETCH routine.
0000 47 ; 0-5 Don Monroe
0000 48 ; Changed the INIT routine to not force a CS Address. Deleted
0000 49 ; the call to disable the CMI from the INIT routine.
0000 50 ; Fixed deposit with line feed terminator in debug routine.
0000 51 ; Fixed a bug in the module typeout routine.
0000 52 ; 0-6 Don Monroe 16 Aug 79
0000 53 ; Fixed bugs in ENDLOOP and BURST CLOCK routines. Added typeout
0000 54 ; of TRACE register and PARITY ERROR on unexpected clock stop.
0000 55 ; Added the ENABLE MICRO TRAP pseudo instruction.
```

0000	56	:	0-7	Don Monroe 17 Aug 79
0000	57	:		Fixed bugs with the WRITE\$_DCS macro calls.
0000	58	:	0-8	Don Monroe 21 Aug 79
0000	59	:		Fixed bug in Start_Execution routine that left clear control
0000	60	:		file bit set when clock was ticked.
0000	61	:		Fixed bug in Read_overlay routine when loading RTEMP data into
0000	62	:		the scratch pads. Same bug in Type_Error routine also.
0000	63	:		Fixed a bug to make the BREAK command work.
0000	64	:	0-9	Don Monroe 22 Aug 79
0000	65	:		Fixed bug in WRITE\$_DCS macro.
0000	66	:	0-10	Don Monroe 22 Aug 79
0000	67	:		Fixed a bug in the TYPE_ERROR routine when reading RTEMP's.
0000	68	:		Added test number to TU58 error typeout.
0000	69	:		Changed RTEMP typeout to start at R0 instead of R10(X).
0000	70	:	0-11	Don Monroe 23 Aug 79
0000	71	:		Added typeout of Loop flags on Unexpected Stop Clock.
0000	72	:	0-12	Don Monroe 9 Oct 79
0000	73	:		Made ENABLE MICRO TRAP instruction work. Added an argument
0000	74	:		to BRSTCLK that inhibits a RET to 0.
0000	75	:	0-13	Don Monroe 22 Oct 1979
0000	76	:		Cleared the CYCLE and TICK flags in BURST_CLOCK routine if
0000	77	:		a space is not typed.
0000	78	:		Added a CLOCKX to the micro code that writes the CACHE.
0000	79	:		Fixed bugs in micro words to disable the CMI and write
0000	80	:		the cache. Added a disable cmi to INIT_THE_WORLD.
0000	81	:	0-14	Don Monroe 29 Oct 1979
0000	82	:		Patched test number typeout routine to type all numbers.
0000	83	:		Removed disable CMI call from INIT routine.
0000	84	:		Fixed BAD DATA in Compare V Bus Routine.
0000	85	:		Fixed Burst Clock routine to work correctly when the stop
0000	86	:		clock bit is set in location 3F.
0000	87	:	0-15	Don Monroe 31 Oct 1979
0000	88	:		Fixed set/clear control file routines to restore the correct
0000	89	:		control file (current dcs address - 1).
0000	90	:	0-16	Don Monroe 13 Nov 79
0000	91	:		Added micro instruction to CACHE WRITE sequence to hold data
0000	92	:		for another cycle.
0000	93	:	0-17	Don Monroe 19 Nov 79
0000	94	:		Added routine to disable cmi routine to disable memory mgmt.
0000	95	:		Modified single cycle and single tick routines to leave
0000	96	:		MASTER HALT ENABLE off while ticking the clock.
0000	97	:		Fixed bug in BURST_CLOCK routine for loop on error.
0000	98	:	0-18	Don Monroe 26 Nov 79
0000	99	:		Added PROC INIT to disable CMI micro code. Added call to
0000	100	:		DISABLE CMI routine in the INITIALIZE routine.
0000	101	:	0-19	Don Monroe 26 Nov 79
0000	102	:		Modified startup code for real environment.
0000	103	:	0 20	Don Monroe 26 Nov 79
0000	104	:		Changed COMPARE V BUS routine to put value of VBUS bit in
0000	105	:		bit position 8 (for typeout) instead of 7.
0000	106	:	0-21	Don Monroe 27 Nov 79
0000	107	:		Made QA flag set the ERROR and LOOP ERROR flags
0000	108	:		unconditionally in the IF_ERROR routine.
0000	109	:		Fixed INIT routine to leave DCS deselected when it exits.
0000	110	:	0-22	Don Monroe 29 Nov 79

0000	111 ;		Fixed BURST CLOCK routine to stop ticking the clock if a
0000	112 ;		micro trap occurs during single cycle or single tick.
0000	113 ;		Added code to enable/disable signature analyzer in the
0000	114 ;		BEGIN/END SA routines.
0000	115 ;	0-23	Don Monroe 30 Nov 79
0000	116 ;		Fixed bug in BEGIN SA ROUTINE to save current PC rather
0000	117 ;		than PC+1.
0000	118 ;	0-24	Bill Landry 5 Dec 79
0000	119 ;		Added code to not test for trap condition at dcs address
0000	120 ;		0 when in single cycle mode.
0000	121 ;	0-25	Don Monroe 10 Dec 79
0000	122 ;		Modified for the Second Pass Multi-Wire RDM.
0000	123 ;	0-26	Don Monroe 11 Dec 79
0000	124 ;		Forgot to rearrange the parity bit mask for the scrambled
0000	125 ;		micro word.
0000	126 ;	0-27	Don Monroe 17 Dec 79
0000	127 ;		Fixed bug in startup code to scramble the builtin micro code.
0000	128 ;	0-28	Don Monroe 18 Dec 79
0000	129 ;		Found bug in scramble table. !'s instead of ;'s.
0000	130 ;	0-29	Don Monroe 31 Dec 79
0000	131 ;		Fixed GO command bug in ODT routine to start at the correct
0000	132 ;		place. Changed NOP micro word to not have STOP CLOCK bit set.
0000	133 ;	0-30	Don Monroe 2 Jan 80
0000	134 ;		Fixed bug in startup code that lost the state of the C bit.
0000	135 ;		Also fixed bug in ODT routine that lost the C bit and did
0000	136 ;		not save the PSW correctly.
0000	137 ;	0-31	Don Monroe 3 Jan 80
0000	138 ;		Fixed bug in ODT routine that unconditionally set the C
0000	139 ;		bit on a 'G' command. Changed the BURST CLOCK routine to
0000	140 ;		read the DCS ADDRESS register for the typeout instead of
0000	141 ;		using an internal counter.
0000	142 ;	0-32	Don Monroe 7 Jan 80
0000	143 ;		Changed format of show flags typeout. Added load of VBUS to
0000	144 ;		SHOW VBUS command. Added typeout of remaining VBUS bits to
0000	145 ;		error typeout if CMP was a CMPVBUS. Added a TYPE_UNCOUNTED
0000	146 ;		macro to type an uncounted ascii string. Changed format
0000	147 ;		of SET/CLR CF data from the bit mask to the bit number.
0000	148 ;	0-33	Don Monroe 8 Jan 80
0000	149 ;		Fixed bugs introduced in version 32.
0000	150 ;	0-34	Don Monroe 9 Jan 80
0000	151 ;		Added EXAMINE and DEPOSIT commands for test overlays.
0000	152 ;		Fixed bug in set step cycle.
0000	153 ;		Fixed bug in somm and jazzed-up print out.
0000	154 ;	0-35	Don Monroe 10 Jan 80
0000	155 ;		Fixed bug in CPU_INIT that forgot about SOMM_FLAG.
0000	156 ;	0-36	Don Monroe 10 Jan 80
0000	157 ;		Modified Directory Search routine to support the new
0000	158 ;		directory format.
0000	159 ;	0-37	Don Monroe 23 Jan 80
0000	160 ;		Modified Directory Search routine to support the file version
0000	161 ;		number encoded in the directory. Also modified BURST_CLOCK
0000	162 ;		routine to try and fix control c problem.
0000	163 ;	0-38	Don Monroe 24 Jan 80
0000	164 ;		Save and restore SOFT_CTRL_REG in BURST_CLOCK routine so
0000	165 ;		that a CTRL C typed before the BURST_CLOCK will be remembered.

0000	166	:	0-39	Don Monroe 29 Jan 80
0000	167	:		Added a clock tick to the ENDSA pseudo instruction to shut off
0000	168	:		the signature analyzer.
0000	169	:	0-40	Don Monroe 29 Jan 80
0000	170	:		Forgot to enable the control file bit in EDIT 39.
0000	171	:	0-41	Don Monroe 30 Jan 80
0000	172	:		Reversed the order of loading the DCS ADDRESS register and
0000	173	:		clearing the control file in the START_EXECUTION routine.
0000	174	:		Changed END_SA routine to actually execute a micro word
0000	175	:		instead of just fetching it.
0000	176	:	0-42	Don Monroe 4 Feb 80
0000	177	:		Fixed bug in DISABLE_CMI routine introduced when the DCS
0000	178	:		SCRATCH area was moved.
0000	179	:	0-43	Don Monroe 4 Feb 80
0000	180	:		Added DCS BUS ENABLE bit to maintenance register.
0000	181	:	0-44	Don Monroe 5 Feb 80
0000	182	:		Added micro code to INIT_CPU routine to clear the PSL.
0000	183	:	0-45	Don Monroe 6 Feb 80
0000	184	:		Modified DIRECT_SEARCH to save the total number of tests
0000	185	:		in the file so the END_TEST pseudo op knows when not to
0000	186	:		try and read the next test.
0000	187	:	0-46	Don Monroe 7 Feb 80
0000	188	:		Fixed bug in DIRECTORY_SEARCH routine.
0000	189	:	0-47	Don Monroe 29 Feb 80
0000	190	:		Added new pseudo op called ONERROR. Added additional
0000	191	:		parameters and functionality to LDFIELD pseudo op.
0000	192	:	0-48	Don Monroe 3 Mar 80
0000	193	:		Deleted the ONERROR pseudo op and made the same functionality
0000	194	:		part of the SKIP pseudo op. Refer to the macro library for
0000	195	:		details on the SKIP pseudo op.
0000	196	:	0-49	Don Monroe 3 Mar 80
0000	197	:		Fixed bug in SKIP introduced in version 0-48.
0000	198	:	0-50	Dave Spain 3 Mar 80
0000	199	:		Fixed bug in SKIP left over from version 0-49.
0000	200	:	0-51	Don Monroe 7 Mar 80
0000	201	:		Fixed bug in LOAD FIELD introduced in version 47.
0000	202	:		Changed SUBTEST routine to get the subtest number from
0000	203	:		the test stream instead of incrementing. Fixed bug in
0000	204	:		RETURN TO RDM routine causing the clock not to start.
0000	205	:	0-52	Don Monroe 27 Mar 80
0000	206	:		Added an inhibit scramble parameter to the LDFIELD pseudo
0000	207	:		instruction. Change module typeout to DCxxx numbers.
0000	208	:	0-53	Don Monroe 28 Mar 80
0000	209	:		Fixed bug introduced by version 52.
0000	210	:	0-54	Don Monroe 2-Apr-80
0000	211	:		Changed the LDINDEX pseudo op. See the description in the
0000	212	:		Library for more details.
0000	213	:	0-55	Don Monroe 3-Apr-80
0000	214	:		Fixed bug introduced in version 54.
0000	215	:	0-56	Don Monroe 30-Apr-80
0000	216	:		Changed source operand index on LOADDCS instruction to
0000	217	:		index by the size of the load.
0000	218	:		Added CCS module name and renamed MCT to CMC. Also changed
0000	219	:		return after end of pass to micro-monitor instead of RDM.
0000	220	:	0-57	Dave Spain 5-May-80

0000	221 ;		Added missing CCS module file name string. Also rearranged
0000	222 ;		the file names so that CCS = ECKAE and CMC = ECKAG.
0000	223 ;	0-58	Bill Landry 6-May-80
0000	224 ;		Deleted UBI and CCS from file name strings. Changed file
0000	225 ;		names so that CMC = ECKAE and FPA = ECKAD.
0000	226 ;	0-59	Don Monroe 18-May-80
0000	227 ;		Added trace flag and typeout of test titles.
0000	228 ;	0-60	Don Monroe 23-May-80
0000	229 ;		Fixed bug in Trace Flag.
0000	230 ;	0-61	Don Monroe 3-Jun-80
0000	231 ;		Fixed bug in Type Test Number routine.
0000	232 ;	0-62	Don Monroe 4-Jun-80
0000	233 ;		Fixed more bugs in type test number routine.
0000	234 ;	0-63	Don Monroe 6-JUN-80
0000	235 ;		Fixed more bugs in test number typeout.
0000	236 ;		Added L000x numbers to module name typeouts.
0000	237 ;	0-64	Don Monroe 9-JUN-80
0000	238 ;		Changed RDM tape name to ECKAF.
0000	239 ;	0-65	Don Monroe 11-Jun-80
0000	240 ;		Added DC6xx numbers to INT and CON.
0000	241 ;		Fixed test number typeout so lines contain same number of tests
0000	242 ;	0-66	Don Monroe 11-Jun-80
0000	243 ;		Fixed module typeout code to index the correct table.
0000	244 ;	0-67	Don Monroe 12-JUN-80
0000	245 ;		Fixed bugs in module typeout code.
0000	246 ;	1-00	Don Monroe 16-JUN-80
0000	247 ;		Initial Release
0000	248 ;	1-01	Don Monroe 26-JUN-80
0000	249 ;		Added pass count option to QA Flag. Conditions the QA flag
0000	250 ;		on a specified pass count.
0000	251 ;	1-02	Don Monroe 3-Jul-80
0000	252 ;		Fixed bug in BURST CLOCK routine related to stepping the
0000	253 ;		clock with the LOOP & ERROR flags set and an INHIBIT on the
0000	254 ;		Burst Clock call.
0000	255 ;	1-03	Don Monroe 3-Jul-80
0000	256 ;		Fixed bug in typeout of Test PC.
0000	257 ;	1-04	Don Monroe 23-Jul-80
0000	258 ;		Added a feature to BURST CLOCK that enables the detection
0000	259 ;		of the clock being hung. Added CMI BUS name to module name
0000	260 ;		list.
0000	261 ;	1-05	Don Monroe/Bill Landry 27-AUG-80
0000	262 ;		Altered monitor to load DCS with NOPS before using any micro-
0000	263 ;		code.
0000	264 ;	1-06	Don Monroe 9-Sep-80
0000	265 ;		Modified LOADREG routine so that if the register is the RD
0000	266 ;		Control Register, the state of the front panel lights is
0000	267 ;		copied into it.
0000	268 ;	1-07	Don Monroe 10-Sep-80
0000	269 ;		Added the ERRLOG and DUMPLOG pseudo instructions. Also added
0000	270 ;		the CMA (M8728) to the module callout list.
0000	271 ;	1-08	Don Monroe 16-Sep-80
0000	272 ;		Fixed bugs in DUMP LOG routine.
0000	273 ;	1-09	Don Monroe 16-Sep-80
0000	274 ;		Fixed bugs in ERROR LOG routine. Added setting and clearing
0000	275 ;		of error flags in ERROR LOG routine.

0000	276 ;	1-10	Bill Landry 19-Sep-80
0000	277 ;		Fixed bug in ERROR LOG routine to clear INIT FLAG after first
0000	278 ;		pass.
0000	279 ;	1-11	Don Monroe 23-Sep-1980
0000	280 ;		Fixed more bugs in error log routine.
0000	281 ;	2-00	Don Monroe 29-Sep-1980
0000	282 ;		Second Release
0000	283 ;	2-01	Don Monroe 7-Oct-80
0000	284 ;		Modified Gate Array list to support bit slice algorithm in
0000	285 ;		error report routine. Added the bit slice algorithm.
0000	286 ;	2-02	Don Monroe 20-Oct-80
0000	287 ;		Fixed SHOW VBUS routine to ensure ctrl file is not holding
0000	288 ;		vbus in load state.
0000	289 ;	2-03	Don Monroe 17-Nov-1980
0000	290 ;		Fixed bug in CHECK_QV routine.
0000	291 ;	3-00	Don Monroe 17-NOV-1980
0000	292 ;		Third Release
0000	293 ;	3-01	Don Monroe 29-Dec-1980
0000	294 ;		Added DI DM command to invoke only the tests for the RDM.
0000	295 ;	3-02	Don Monroe 30-Dec-1980
0000	296 ;		Fixed bugs.
0000	297 ;	3-03	Don Monroe 30-Dec-1980
0000	298 ;		Fixed bugs.
0000	299 ;	3-04	Don Monroe 31-Dec-1980
0000	300 ;		Fixed bugs.
0000	301 ;	4-00	Don Monroe 5-Jan-1980
0000	302 ;		Added code to enable the clock interrupts if key switch is
0000	303 ;		in remote position.
0000	304 ;	4-01	Don Monroe 17-Feb-81
0000	305 ;		Added PRINT_N and PRINT_S pseudo instructions.
0000	306 ;		Added NOTEQUAL condition to SKIP pseudo instruction.
0000	307 ;	5-00	Don Monroe 2-Mar-1981
0000	308 ;		Release.
0000	309 ;	5-01	Bill Landry 19-MAY-1981
0000	310 ;		Added code to fix ERROR_LOG ROUTINE. Routine would take
0000	311 ;		fail exit if entry was already in log.
0000	312 ;	5-02	Don Monroe 21-May-81
0000	313 ;		Started work on EXAMINE command.
0000	314 ;	5-03	Don Monroe 3-Jun-81
0000	315 ;		Fixed bugs in EXAMINE routine.
0000	316 ;	5-04	Don Monroe 8-Jun-81
0000	317 ;		Fixed bugs in EXAMINE routine.
0000	318 ;		Added an initialize of DCS on entry to the monitor.
0000	319 ;	5-05	Don Monroe 8-Jun-81
0000	320 ;		Moved the clock stop code from program init to entry
0000	321 ;		point into monitor so EXAMINE's would work on TE/C cmd.
0000	322 ;	5-06	Don Monroe 12-Jun-81
0000	323 ;		Fixed the RDM_WDR micro word.
0000	324 ;	5-07	Don Monroe 12-Jun-81
0000	325 ;		Fixed mask on examine of SF and SC.
0000	326 ;	6-00	Don Monroe 23-Jun-81
0000	327 ;		Changed revision for release.
0000	328 ;	6-01	Don Monroe 23-Jul-81
0000	329 ;		Changed SET STEP CYCLE/TICK routines to clear the
0000	330 ;		other flag. Modified timeout routine to automatically

0000	331 ;		insert a Carriage Return Line Feed after 80 characters.
0000	332 ;	6-02	Don Monroe 24-Aug-81
0000	333 ;		Added new Gate Array names for FPA module.
0000	334 ;	6-03	Don Monroe 2-Sep-81
0000	335 ;		Masked PSL data on PSL EXAMINE.
0000	336 ;	6-04	Dave Spain 4-Sep-81
0000	337 ;		Added missing FPA gate arrays.
0000	338 ;	6-05	Don Monroe 8-Sep-81
0000	339 ;		Fixed bug in FPA bit-sliced name tables.
0000	340 ;		Added ENABLE_STALL routine and modified BRSTCLOCK routine
0000	341 ;		to support it.
0000	342 ;	6-06	Don Monroe 22-Sep-81
0000	343 ;		Fixed bug in PSL Examine. The AND with immediate mask used
0000	344 ;		the wrong instruction.
0000	345 ;	6-07	Don Monroe 24-Sep-81
0000	346 ;		Modified Single Bit Error Messages. Changed Load Overlay
0000	347 ;		Routine to do an Initialize before trying to load cache
0000	348 ;		and other internals.
0000	349 ;	7-00	Don Monroe 10-Nov-81
0000	350 ;		Added SBE keyword and flag to DIAGNOSE command to enable
0000	351 ;		the DUMP_LOG routine to print the state of the error log.
0000	352 ;	7-01	Dave Spain 28-Jan-82
0000	353 ;		Added L0016 memory controller and MAD gate array, and changed
0000	354 ;		CMC and CMA names. L0011 controller is now MC0, L0016 is MC1.
0000	355 ;		M8728 256K array card is now MA0, M8750 1M array card is MA1.
0000	356 ;		Changed callouts of FCS, FMR and FFA to callout all slices.
0000	357 ;	7-02	Dave Spain 1-Feb-82
0000	358 ;		Fixed bug in FCS, FMR and FFA callouts introduced in 7-01.
0000	359 ;	7-03	Dave Spain 3-Feb-82
0000	360 ;		Replaced references to LOWER 256 KB in correctable bits
0000	361 ;		messages to ARRAY 0.
0000	362 ;	7-04	Dave Spain 19-Feb-82
0000	363 ;		Removed CMC and CMA module references from error callouts.
0000	364 ;		Now calls out Memory Controller and Memory Array Cards under
0000	365 ;		the L and M module numbers only.
0000	366 ;	7-05	Dave Spain 8-Mar-82
0000	367 ;		Fixed bug in ERROR_LOG routine. Wasn't checking test address
0000	368 ;		against top of error log correctly.
0000	369 ;	7-06	Dave Shull 10-Mar-82
0000	370 ;		Changed CRD messages (LOG_MSG_2 & LOG_MSG_3) to reference
0000	371 ;		the memory array card as ARRAY versus ARRAY 0. This allows
0000	372 ;		sharing of message for either array 0, array 1, etc.
0000	373 ;	7-07	Dave Shull 22-Mar-82
0000	374 ;		Changed the DUMP_LOG routine so it would set the ERR_LOG_INIT
0000	375 ;		flag. This way the combinations of ERRLOG and DUMPL0G pseudo's
0000	376 ;		can be used more than once within the same micro to check more
0000	377 ;		than one array.
0000	378 ;	7-08	Dave Shull 06-Apr-82
0000	379 ;		Changed the TYPE_ERROR and TYPE_GA_LIST routines so they would
0000	380 ;		print the failing Gate Array(s) and Module(s) in column.
0000	381 ;	8-00	Dave Shull 08-Apr-82
0000	382 ;		Removed the BEGNSA and ENDSA pseudo execution code which
0000	383 ;		included the following:
0000	384 ;		- Removed symbol SA_ADDRESS;
0000	385 ;		- Removed BEGIN_SA and END_SA from INSTR_TABLE;

0000	386	:	8-01	Dave Spain 05-May-82
0000	387	:		Changed bit-slice tables for FPA gate arrays to enhance
0000	388	:		isolation for error callouts.
0000	389	:	8-02	Dave Shull 11-May-82
0000	390	:		Changed the LOAD_INDEX routine so that it will handle the case
0000	391	:		where there is an address specified that points to the value to
0000	392	:		be loaded into the index, with No index specified for that
0000	393	:		address.
0000	394	:	8-03	Dave Shull 03-Jun-82
0000	395	:		Fixed the COMPARE_VBUS routine so it would work with an index
0000	396	:		greater than 80 (hex). Prior to this fix, values greater than
0000	397	:		80 (hex) would not get processed correctly.
0000	398	:	8-04	Dave Shull 14-Jun-82
0000	399	:		Changed CMK callout to CML for new gate array.
0000	400	:	8-05	Dave Shull 03-Nov-82
0000	401	:		Changed the CCS module callout from L0005 to L0008. This is
0000	402	:		for the new PCS module.
0000	403	:	8-06	Joe Zagarella 20-dec-85
0000	404	:		Modified to be compatible with the L0003-YA module. Added PC_WB
0000	405	:		to WCTRL field of microword that does LONLIT_0 in DISABLE_CMI
0000	406	:		subroutine. By loading the PC, the XB does a prefetch and is
0000	407	:		filled with valid data and therefore will not try to do any
0000	408	:		more prefetching. It was the prefetching of the XB that
0000	409	:		indirectly caused a glitch on the PAD lines during a write to
0000	410	:		cache.
0000	411	:	8-07	Joe Zagarella 11-feb-86
0000	412	:		Added module callouts for new memory controller(L0022) and
0000	413	:		new 4mb array card(M7199)
0000	414	:		
0000	415	:--		

```
0000 417 .LIST MC
0000 418 .NLIST CND,ME,MD
0000 419 .SBTTL DECLARATIONS
0000 420
0000 421 ;
0000 422 ; INCLUDE FILES:
0000 423 ;
0000 424 .LIBRARY /WRK$LIBRARY:8085MAC/
0000 425 .LIBRARY /WRK$LIBRARY:COMETMAC/
0000 426 .LIBRARY /WRK$LIBRARY:RDMMAC/
0000 427
0000 428 ;
0000 429 ; MACROS:
0000 430 ;
0000 431
0000 463
0000 489
0000 517
0000 548
0000 579
0000 609
0000 655
0000 706
0000 729
0000 753
0000 766
0000 784
0000 785
0000 817
0000 842
0000 850
```

```
0000 852      .SBTTL  EQUATED SYMBOLS
0000 853      ;
0000 854      ; EQUATED SYMBOLS:
0000 855      ;
0000 856      EQUATE                                ; THIS MACRO IS IN 8085MAC. DEFINES
0000      ;
0000      ; 8085 REGISTER DEFINITIONS
0000      ;
0000      ;
0000      ; REGISTER PAIRS
0000      ;
00000000 0000      B      =      0
00000002 0000      D      =      2
00000004 0000      H      =      4
00000006 0000      $SP    =      6
00000006 0000      $PSW   =      6
0000      ;
0000      ; NON REGISTER PAIRS
0000      ;
00000001 0000      C      =      1
00000003 0000      E      =      3
00000005 0000      L      =      5
00000006 0000      H      =      6
00000007 0000      A      =      7
0000      ;
0000      ; 8085 ASSEMBLY LANGUAGE DEFINITION MACRO'S
0000      ;
0000      .MCALL  MOV,MVI,LXI,STAX,LDAX,STA,LDA,SHLD,LHLD,XCHG
0000      .MCALL  PUSH,POP,XTHL,SPHL,JMP,JC,JNC,JZ,JNZ,JP,JM,JPE,JPO,PCHL
0000      .MCALL  CALL,CC,CNC,CZ,CNZ,CP,CM,CPE,CPO
0000      .MCALL  RET,RC,RNC,RZ,RNZ,RP,RM,RPE,RPO,RST
0000      .MCALL  INR,DCR,INX,DCX,ADD,ADC,ADI,ACI,DAD,SUB,SBB,SUI,SBI
0000      .MCALL  ANA,XRA,ORA,CMP,ADI,XRI,ORI,CPI
0000      .MCALL  RLC,RRC,RAL,RAR,CMA,STC,CMC,NOP,HLT,RIM,SIM
0000      857      ; 8085 REGISTERS. THIS
0000      858      ; MACRO CALL MUST APPEAR BEFORE ANY CODE
```

```
0000 860 ; IN COMETMAC. DEFINES EQUATED SYMBOLS
0000 861 ; AND HEAD WHICH IS USED TO CALCULATE
0000 862 ; ALL ADDRESS OFFSETS.
0000 863
EQUATE_SYMBOLS 1
FFFF7C00 0000 HEAD = -^X8400
0000 ;
0000 ; RDM ADDRESS DEFINITIONS:
0000 ;
00001800 0000 DCS_START = ^X1800 ; START ADDRESS OF DCS
0000 ;
0000 ; RDM RETURN ERROR CODE DEFINITIONS:
0000 ;
0000000A 0000 $CONTROL_C_CODE = ^XA
00000009 0000 $FILE_NOT_FOUND = 9
0000 ;
0000 ; SOFTWARE CONTROL REGISTER BIT DEFINITIONS:
0000 ;
00000001 0000 CONTROL_C_FLAG = 1
00000002 0000 CONTINUE_FLAG = 2
00000004 0000 DIAGNOSE_FLAG = 4
00000008 0000 FETCH_FLAG = 8
00000010 0000 LOST_FLAG = ^X10
00000020 0000 CONT_FLAG = ^X20
00000040 0000 TSTSPAN_FLAG = ^X40
00000080 0000 BURST_STOP_FLAG = ^X80
0000 ;
0000 ; SOFTWARE CONTROL REGISTER 'A' BIT DEFINITIONS:
0000 ;
00000001 0000 SOMM_FLAG = 1
00000002 0000 TICK_FLAG = 2
00000004 0000 INSTR_FLAG = 4
00000008 0000 CYCLE_FLAG = 8
00000010 0000 ERROR_FLAG = ^X10
00000020 0000 LOOP_ERROR_FLAG = ^X20
00000040 0000 EXP_UTRAP_FLAG = ^X40
00000080 0000 VBUS_COMPARE = ^X80
0000 ;
0000 ; SOFTWARE CONTROL REGISTER 'B' BIT DEFINITIONS:
0000 ;
00000001 0000 ERR_LOG_INIT = 1
00000002 0000 QV_FLAG = 2
00000004 0000 RDM_FLAG = 4
00000008 0000 RDM_LAST = 8
00000010 0000 EXP_STALL_FLAG = ^X10
00000020 0000 SBE_FLAG = ^X20
0000 ;
0000 ; PROGRAM CONTROL REGISTER BIT DEFINITIONS:
0000 ;
00000001 0000 HALT_FLAG = 1
00000002 0000 LOOP_FLAG = 2
00000004 0000 NER_FLAG = 4
00000008 0000 BELL_FLAG = 8
00000010 0000 SA_FLAG = ^X10
00000020 0000 IB_FLAG = ^X20
```



```

00007402 0000 DCSCR=DCS_CTRL_REG ;Diagnostic control store control reg
00007403 0000 DCSCF=DCS_CTRL_FILE ;Diagnostic control store control file
00007404 0000 CSAMR=ADDR_MATCH_LO ;Control store address match register
00007406 0000 STATUS=STATUS_REG ;Status register
00007410 0000 MAREG=A_REG_BYTE_0 ;Memory address register
00007414 0000 MDREG=MD_REG_BYTE_0 ;Memory data register
00007418 0000 MHREG=MH_REG_BYTE_0 ;Memory holding register
0000741C 0000 CMICtrl=CMI_CTRL_REG ;CMI control
0000741D 0000 MAIN32=MAINT_REG ;32 Bit path maintenance control
0000741E 0000 RDCTRL=RD_CTRL_REG ;Remote diagnosis control
00007420 0000 FRONT1=FRONT_PNL_1 ;Front panel readback 1
00007421 0000 FRONT2=FRONT_PNL_2 ;Front panel readback 2
00007422 0000 CSTRP=CS_ADD_TRAP_LOW ;Control store address trap register
00007424 0000 CSBUF=CS_ADD_BUFF_LOW ;Control store address buffer
00007426 0000 BADD1=BUFF_ADDR_LOW ;Control store address trace readback
00007427 0000 CLKDCS=DCS_ADDR_REG_RO ;Clock control & DCS address register
0000742C 0000 WDREG=WD_REG_BYTE_0 ;WBUS data register
00007430 0000 WHREG=WH_REG_BYTE_0 ;WBUS holding register
  
```

;; RDM REGISTER BIT DEFINITIONS:

```

00000001 0000 HALT_ON_MATCH = 1
00000002 0000 CLEAR_CTRL_FILE = 2
00000004 0000 TRACE_ENABL = 4
00000008 0000 PAR_CHK_ENABL = 8
00000010 0000 PAR_STOP_ENABL = ^X10
00000020 0000 CLK_CTRL_0 = ^X20 ; ACTIVE LOW
00000040 0000 CLK_CTRL_1 = ^X40 ; ACTIVE LOW
00000080 0000 MICRO_ADDR_INH = ^X80
  
```

;; FOLLOWING IS A FUNCTION TABLE OF THE CLOCK CONTROL BITS

CLK_CTRL 1,0	FUNCTION
11	HALT
01	SINGLE MICRO INSTRUCTION
10	SINGLE TICK
00	RUN

;; DCS CONTROL FILE ALL OF THESE BITS ARE ACTIVE LOW

```

00000001 0000 V_BUS_STROBE = 1
00000002 0000 DCS_ADDR_CLEAR = 2
00000004 0000 STOP_CLOCK = 4
00000008 0000 EN_TRACE_REG = 8
00000010 0000 H_FROM_WBUS = ^X10
00000020 0000 WBUS_FROM_D = ^X20
00000040 0000 STROBE_CMI = ^X40
00000080 0000 SA_CLOCK = ^X80
  
```

;; CS ADDRESS MATCH REGISTER (HIGH)

```

00000040 0000 VBUS_SERIAL_IN = ^X40 ; ACTIVE LOW
00000080 0000 VBUS_CLOCK = ^X80
  
```

```
0000 ; STATUS REGISTER
0000 ;
00000001 0000 VBUS_SERIAL_OUT = 1
00000002 0000 TRAP_OFF = 2
00000008 0000 CMI_STATUS_0 = 8
00000010 0000 CMI_STATUS_1 = ^X10
0000 ;
0000 ; THE CMI STATUS BITS ARE ENCODED AS FOLLOWS:
0000 ;
0000 ; 00 = NO RESPONSE
0000 ; 01 = UNCORRECTABLE ERROR
0000 ; 10 = CORRECTABLE ERROR
0000 ; 11 = NO ERRORS
00000020 0000 CMI_COMPLETE = ^X20
00000080 0000 ;
0000 ; CLOCK_STOPED = ^X80
0000 ;
0000 ; CMI CONTROL REGISTER
0000 ;
00000001 0000 CMI_CTRL_CLEAR = 1
00000008 0000 CMI_GO = 8
00000020 0000 CMI_WRITE = ^X20
00000040 0000 CMI_READ = ^X40
00000080 0000 SA_START_STOP = ^X80
0000 ;
0000 ; 32 BIT MAINTENANCE CONTROL REGISTER
0000 ;
00000001 0000 MAINT_TRAP_OFF = 1
00000002 0000 MAINT_STB_INH = 2
00000004 0000 MAINT_DCS_ENAB = 4
00000008 0000 MAINT_WB_TO_WH = 8
00000010 0000 MAINT_WD_TO_WB = ^X10
00000020 0000 MAINT_CMI_TO_MH = ^X20
00000040 0000 MAINT_MD_TO_CMI = ^X40
00000080 0000 MAINT_A_TO_CMI = ^X80
0000 ;
0000 ; RD CONTROL REGISTER
0000 ;
00000008 0000 MASTER_HALT_EN = 8
00000010 0000 VBUS_LOAD = ^X10
00000020 0000 TRAP_HALT_EN = ^X20
00000040 0000 DC_LOW = ^X40
00000080 0000 AC_LOW = ^X80
0000 ;
0000 ; FRONT PA REGISTER 1
0000 ;
00000001 0000 FP_BOOT_0 = 1
00000002 0000 FP_BOOT_1 = 2
00000004 0000 FP_START_0 = 4
00000008 0000 FP_START_1 = 8
00000010 0000 CPU_RUN = ^X10
00000020 0000 PARITY_ERROR = ^X20
0000 ;
0000 ; FRONT PANEL REGISTER 2
0000 ;
```

```
00000001 0000 REMOTE = 1
00000002 0000 FAULT = 2
00000004 0000 TEST = 4
:
:
:
00000040 0000 PB_INIT = ^X40 : ACTIVE LOW
00000080 0000 FRONT_PNL_LOCK = ^X80
:
: DCS ADDRESS REGISTER READ ONLY
:
00000040 0000 CLK_CTRL_0_RO = ^X40 : ACTIVE HIGH
00000080 0000 CLK_CTRL_1_RO = ^X80 : ACTIVE HIGH
:
: KEYWORD LIST OFFSET DEFINITIONS:
:
00000000 0000 TEST_KEY = 0
00000002 0000 PASS_KEY = 2
00000004 0000 FLAG_KEY = 4
00000006 0000 SOMM_KEY = 6
00000008 0000 HALT_KEY = 8
0000000A 0000 LOOP_KEY = 10
0000000C 0000 MER_KEY = 12
0000000E 0000 BELL_KEY = 14
00000010 0000 SA_KEY = 16
00000012 0000 IB_KEY = 18
00000014 0000 QA_KEY = 20
00000016 0000 TR_KEY = 22
00000018 0000 INSTR_KEY = 24
0000001A 0000 LOST_KEY = 26
0000001C 0000 CONT_KEY = 28
0000001E 0000 STEP_KEY = 30
00000020 0000 CYCLE_KEY = 32
00000022 0000 TICK_KEY = 34
00000024 0000 CF_KEY = 36
00000026 0000 VB_KEY = 38
00000028 0000 QV_KEY = 40
0000002A 0000 RDM_KEY = 42
0000002C 0000 RT_KEY = 44
0000002E 0000 MT_KEY = 46
00000030 0000 PC_KEY = 48
00000032 0000 VA_KEY = 50
00000034 0000 MD_KEY = 52
00000036 0000 PB_KEY = 54
00000038 0000 MA_KEY = 56
0000003A 0000 WD_KEY = 58
0000003C 0000 SR_KEY = 60
0000003E 0000 PS_KEY = 62
00000040 0000 SF_KEY = 64
0000001E 0000 ST_KEY = STEP_KEY
00000042 0000 SB_E_KEY = 66
```

: THE FOLLOWING EQUATES ARE NOT USED BY THE 1

: PSEUDO INSTRUCTION OP CODE DEFINITIONS:

```
00000000 0000 ;
00000002 0000 OP_LOAD_DCS = 0
00000004 0000 OP_BURST_CLOCK = 2
00000006 0000 OP_COMPARE_REG = 4
00000008 0000 OP_COMPARE_REGM = 6
0000000A 0000 OP_BEGIN_LOOP = 8
0000000C 0000 OP_END_LOOP = 10
0000000E 0000 OP_END_TEST = 12
00000010 0000 OP_ERROR_LOOP = 14
00000012 0000 OP_PATTERN_GEN = 16
00000014 0000 OP_IF_ERROR = 18
00000016 0000 OP_INIT = 20
00000018 0000 OP_LOAD_REG = 22
0000001A 0000 OP_MASK = 24
0000001C 0000 OP_NEW_TEST = 26
0000001E 0000 OP_SKIP = 28
00000020 0000 OP_SUB_TEST = 30
00000022 0000 OP_COMPARE_VB = 32
00000024 0000 OP_FETCH = 34
00000026 0000 OP_END_FILE = 36
00000028 0000 OP_LOAD_FIELD = 38
0000002A 0000 OP_LOAD_INDEX = 40
0000002C 0000 OP_INC_INDEX = 42
0000002E 0000 OP_SAVE_INDEX = 44
00000030 0000 OP_ENABL_TRAP = 46
00000032 0000 OP_DUMPLÖG = 48
00000034 0000 OP_ERRLOG = 50
00000036 0000 OP_PRINT_S = 52
00000038 0000 OP_PRINT_N = 54
00000038 0000 OP_ENABL_STALL = 56
```

; GATE ARRAY NAME DEFINITIONS:

```
00000000 0000 ;
00000001 0000 ;
00000002 0000 ;
00000003 0000 ;
00000004 0000 ;
00000005 0000 ;
00000006 0000 ;
00000007 0000 ;
00000008 0000 ;
00000009 0000 ;
0000000A 0000 ;
0000000B 0000 ;
0000000C 0000 ;
0000000D 0000 ;
0000000E 0000 ;
0000000F 0000 ;
00000010 0000 ;
00000011 0000 ;
00000012 0000 ;
00000013 0000 ;
00000014 0000 ;
00000015 0000 ;
ALK = 0
SRK = 1
SPA = 2
CLA = 3
CCC = 4
IRD = 5
TOK = 6
PHB = 7
MSQ = 8
SAC = 9
CAK = 10
ADK = 11
PRK = 12
UTR = 13
CML = 14
ACV = 15
INT = 16
CON = 17
MAP = 18
MAD = 19
FQA = 20
FCC = 21
```

```
0000
0000
0000
0000
00000016 0000
00000017 0000
00000018 0000
00000019 0000
0000001A 0000
0000001B 0000
0000001C 0000
0000001D 0000
0000001E 0000
0000001F 0000
00000020 0000
00000021 0000
00000022 0000
00000023 0000
00000024 0000
00000025 0000
0000
0000
0000
00000000 0000
00000001 0000
00000002 0000
00000003 0000
00000004 0000
00000005 0000
00000006 0000
00000007 0000
00000008 0000
00000009 0000
0000000A 0000
0000000B 0000
0000000C 0000
0000
0000
0000
00000001 0000
00000002 0000
00000004 0000
0000000A 0000
00002C5E 0000
00000400 0000
000007A2 0000
000003E2 0000
0000
00000001 0000
00000000 0000
00000000 0000
00000001 0000
```

```

: THE FOLLOWING GATE ARRAY'S ARE IN A SPECIAL GROUP BECAUSE THEY ARE
: BIT SLICED. THE ERROR ROUTINE TRYS TO FIGURE OUT WHICH SLICE IS BAD
: IF ANY ONE OF THESE GATE ARRAYS IS IN THE CALLOUT LIST.
:
ALP = 22
SRM = 23
MDR = 24
ADD = 25
MEC = 26
MDL = 27
UDP = 28
FIO = 29
FEX = 30
FIC = 31
FAC = 32
FCS = 33
FMR = 34
FFL = 35
FFH = 36
END_SLICE = 37 ; ONE MORE THAN LAST BIT-SLICED ARRAY

: MODULE NAME DEFINITIONS:
:
DPM = 0
MIC = 1
FPA = 2
MCO = 3
RDM = 4
CCS = 5
UBI = 6
CMI = 7
MA0 = 8
MC1 = 9
MA1 = 10
MC2 = 11
MA2 = 12

: MISCELLANEOUS EQUATES:
:
BYTE = 1
WORD = 2
LONG = 4
MICROWORD = 10
MONITOR_SIZE = ^X2C5E
STACK_SIZE = 1024
M$TEST_SIZE = 14336 - STACK_SIZE - MONITOR_SIZE
; MAXIMUM SIZE OF A TEST OVERLAY
M$TEST_SIZE_D = M$TEST_SIZE - 960
; MAXIMUM SIZE OF TEST OVERLAY WITH
; 'DEBUG' ENABLED IN 1

HIGH = 1
LOW = 0
ONERROR = 0
NOERROR = 1
```

ZZ-ECKAA-8.7 V08.07
ECKAA
V08.07

E 4
EQUATED SYMBOLS 11-FEB-1986
VAX-11/750 MICRO DIAGNOSTIC MONITOR
EQUATED SYMBOLS

Fiche 1 Frame E4 Sequence 43
11-FEB-1986 10:07:52 VAX/VMS Macro V04-00 Page 18
11-FEB-1986 10:06:37 [VAX750.MONITOR]ECKAA.MAR;811 (4)

00000002	0000	ONEQUAL	=	2	
00000003	0000	NOTEQUAL	=	3	
00000010	0000	MAX_ARGUMENTS	=	16	; BYTE LENGTH OF LARGEST PSEUDO OP
00000036	0000	DCS_SCRATCH_ADR	=	54	; DCS SCRATCH ADDRESS START
000032FC	0000	ERROR_LOG_ADR	=	^XB6FC	+ HEAD
	0000				; START ADDRESS OF MEMORY ERROR LOG
	0000	864			

```

0000 866      .SBTTL  "LINKAGE FROM RDM SYSTEM
0000 867
0000 1368    ;
0000 1369    ; LINKAGE FROM RDM SYSTEM
0000 1370    ; THE 'COMETMAC' MACRO PACKAGE WILL CAUSE THE STARTING ADDRESS OF THIS
0000 1371    ; FILE TO BE 8400(X).
0000 1372    ;
0000 1373    ;           JMP           START1           ; GETS ASSEMBLED AT 8400(X)
0003 1374    ;           LXI           H,0             ; GET CURRENT SP
0006 1375    ;           DAD           $SP              ;
0007 1376    ;           SHLD          SAVED_SP       ; SAVE FOR ABORT ROUTINE
000A 1377    ;           ABORT          ; THIS GETS ASSEMBLED AT LOCATION 8403(X)
000D 1378    ;                               ; THE RDM CODE WILL COME HERE IF
000D 1379    ;                               ; IT NEEDS TO ABORT THE MONITOR
000D 1380    CMD_ENTRY:
000D 1381    ;           RET
000E 1383
000E 1384    START1: PUSH          $PSW             ; SAVE THE C BIT
000F 1385    ;           LXI           H,0             ; GET THE STACK POINTER
0012 1386    ;           DAD           $SP              ;
0013 1387    ;           INX          H               ; ACCOUNT FOR THE PSW PUSH
0014 1388    ;           INX          H               ;
0015 1389    ;           SHLD          SAVED_SP       ; SAVE IT
0018 1390    ;
0018 1391    ; SCRAMBLE THE MICRO CODE IN THE MONITOR
0018 1392    ;
0018 1393    ;           LXI           H,NOP_MICRO_WORD ; GET START ADDRESS OF MICRO CODE
001B 1394    ;           MVI           A,NO_MICRO_WORDS ; GET NUMBER OF MICRO WORDS
001D 1395    1$:  PUSH          $PSW             ; SAVE THE NUMBER OF MICRO WORDS
001E 1396    ;           PUSH         H             ; COPY H & L INTO D & E
001F 1397    ;           POP          D             ;
0020 1398    ;           MVI           B,0             ; SET THE STARTING BIT NUMBER
0022 1399    ;           MVI           C,79           ; AND THE ENDING BIT NUMBER
0024 1400    ;           PUSH         H             ; SAVE H & L
0025 1401    ;           XRA          A             ; CLEAR INHIBIT PARITY FLAG
0026 1402    ;           PUSH         $PSW          ; CLEAR INHIBIT SCRAMBLE FLAG
0027 1403    ;           CALL         INSERT_FIELD  ; SCRAMBLE THE MICRO WORD
002A 1404    ;           POP          H             ; RESTORE H & L
002B 1405    ;           MVI           E,11          ; ADD 11 TO H & L
002D 1406    ;           MVI           D,0             ; ...
002F 1407    ;           DAD           D             ;
0030 1408    ;           POP          $PSW          ; GET THE NUMBER OF MICRO WORDS
0031 1409    ;           DCR          A             ; DECREMENT
0032 1410    ;           JNZ         1$            ; BRANCH IF MORE TO SCRAMBLE
0035 1411    ;
0035 1412    ; STOP THE CLOCK AND INITIALIZE SOME MORE PARAMETERS
0035 1413    ;
0035 1414    ;           MVI           A,<CLK_CTRL_0!CLK_CTRL_1> ; GET DATA TO STOP THE CLOCK
0037 1415    ;           STA          DCS_CTRL_REG   ; STOP THE CLOCK
003A 1416    ;           STA          DCS_CTRL_RE_CPY ; SAVE A COPY OF THE CONTROL REGISTER
003D 1417    ;           LDA          FRONT_PNL_2   ; GET TEST, FAULT, AND REMOTE BITS
0040 1418    ;           ANI          <TEST!FAULT!REMOTE> ; ...
0042 1419    ;           STA          RD_CTRL_REG   ; CLEAR MASTER HALT ENABLE
0045 1420    ;           ORI          MASTER_HALT_EN ; SET MASTER HALT ENABLE
0047 1421    ;           STA          RD_CTRL_REG   ; ...

```

ZZ-ECKAA-8.7 V08.07
ECKAA
V08.07

G 4

LINKAGE FROM RD11-FEB-1986
VAX-11/750 MICRO DIAGNOSTIC MONITOR
LINKAGE FROM RDM SYSTEM

Fiche 1 Frame G4 Sequence 45
11-FEB-1986 10:07:52 VAX/VMS Macro V04-00 Page 20
11-FEB-1986 10:06:37 [VAX750.MONITOR]ECKAA.MAR;811 (5)

004A	1422	MVI	A,<MAINT_TRAP_OFF!-
004A	1423		MAINT_DCS_ENABL>; GET DATA TO INIT THE MAINT REG
004C	1424	STA	MAINT_REG ; LOAD THE MAINT REG
004F	1425	CALL	FILL_DCS ; FILL DCS WITH NOP'S
0052	1426	:	
0052	1427	;	CHECK ENTRY FLAG
0052	1428	:	
0052	1429	POP	\$PSW ; RESTORE THE C BIT
0053	1430	CC	GET_CMD_LINE ; ENTER HERE IF 'T/C' COMMAND USED
0056	1431	JMP	PROGRAM_INIT ; START EXECUTION
0059	1432		

```
0059 1434 .SBTTL OWN STORAGE
0059 1435 ;
0059 1436 ; OWN STORAGE
0059 1437 ;
0059 1438 ;
0059 1439 ;
0059 1440 ; SOFTWARE CONTROL REGISTERS
0059 1441 ;
0059 1442 ;
00 0059 1443 SOFT_CTRL_REG: .BYTE ; SOFTWARE CONTROL REGISTER
00 005A 1444 SOFT_CTRL_REGA: .BYTE ; SOFTWARE CONTROL REGISTER A
01 005B 1445 SOFT_CTRL_REGB: .BYTE ERR_LOG_INIT ; SOFTWARE CONTROL REGISTER B
00 005C 1446 DCS_CTRL_RE_CPY: .BYTE ; USED TO SIMULATE BIT SETS AND BIT
005D 1447 ; CLEARS TO THE DCS CONTROL REGISTER
005D 1448 ;
005D 1449 ;
005D 1450 ; PROGRAM CONTROL REGISTER
005D 1451 ;
005D 1452 ;
01 005D 1453 PROG_CTRL_REG: .BYTE HALT_FLAG ; HALT FLAG IS INITIALLY SET
005E 1454 ;
005E 1455 ;
005E 1456 ; PROGRAM_INIT PARAMETERS:
005E 1457 ;
005E 1458 ;
0000 005E 1459 SAVED_SP: .WORD ; STACK POINTER ON CALL FROM RDM
00 0060 1460 TEST_NUMBER: .BYTE ; CONTAINS THE CURRENT TEST NUMBER
00 0061 1461 SUB_TEST_NUMB: .BYTE ; CONTAINS THE CURRENT SUB TEST NUMBER
00 0062 1462 PASS_COUNT: .BYTE ; CONTAINS THE CURRENT PASS COUNT
00 0063 1463 FILE_NAM_INDEX: .BYTE ; CONTAINS AN INDEX FOR FILE NAME LIST
FFFF 0064 1464 STARTING_RECORD: .WORD -1 ; CONTAINS THE STARTING RECORD NUMBER
0066 1465 ; OF THE TEST FILE LAST OPENED
00 0066 1466 TOTAL_NMB_TESTS: .BYTE ; NUMBER OF TESTS IN THE FILE
0067 1467 ;
0067 1468 PROGRAM_TITLE: GENERATE_TITLE \PRIM_REV,\SEC_REV
45 54 20 44 49 4C 41 56 4E 49 3F 00' 0076 1469 DIRECT_MSG: .ASCII /?INVALID TEST NUMBER/ ; SPECIAL TEST IS NOT IN FILE
52 45 42 4D 55 4E 20 54 53 0082
14 0076
008B 1470 MODULE_NAM_TBL: ADDRESS <DPM_NAME> ; TABLE OF POINTERS TO MODULE NAME
008D 1471 ADDRESS <MIC_NAME> ; ASCII STRINGS
008F 1472 ADDRESS <FPA_NAME> ; ...
0091 1473 ADDRESS <MCO_NAME> ; ...
0093 1474 ADDRESS <RDM_NAME> ; ...
0095 1475 ADDRESS <CCS_NAME> ; ...
0097 1476 ADDRESS <UBI_NAME> ; ...
0099 1477 ADDRESS <CHI_NAME> ; ...
009B 1478 ADDRESS <MA0_NAME> ; ...
009D 1479 ADDRESS <MC1_NAME> ; ...
009F 1480 ADDRESS <MA1_NAME> ; ...
00A1 1481 ADDRESS <MC2_NAME> ; ...
00A3 1482 ADDRESS <MA2_NAME> ; ...
00A5 1483 ;
00A5 1484 MODULE_LST:
29 32 30 30 30 4C 28 4D 50 44 00' 00A5 1485 DPM_NAME: .ASCII /DPM(L0002)/ ; ASCII MODULE NAMES
0A 00A5
```

```
29 33 30 30 30 4C 28 43 49 4D 00' 00B0 1486 MIC_NAME: .ASCIC /MIC(L0003)/ ; ...
0A 00B0
29 31 30 30 30 4C 28 41 50 46 00' 00BB 1487 FPA_NAME: .ASCIC /FPA(L0001)/ ; ...
0A 00BB
20 29 31 31 30 30 4C 28 20 20 00' 00C6 1488 MCO_NAME: .ASCIC / (L0011) / ; ...
0A 00C6
29 36 30 30 30 4C 28 4D 44 52 00' 00D1 1489 RDM_NAME: .ASCIC /RDM(L0006)/ ; ...
0A 00D1
29 38 30 30 30 4C 28 53 43 43 00' 00DC 1490 CCS_NAME: .ASCIC /CCS(L0008)/ ; ...
0A 00DC
29 34 30 30 30 4C 28 49 42 55 00' 00E7 1491 UBI_NAME: .ASCIC /UBI(L0004)/ ; ...
0A 00E7
20 20 53 55 42 20 20 49 4D 43 00' 00F2 1492 CMI_NAME: .ASCIC /CMI BUS / ; ...
0A 00F2
20 29 38 32 37 38 4D 28 20 20 00' 00FD 1493 MA0_NAME: .ASCIC / (M8728) / ; ...
0A 00FD
20 29 36 31 30 30 4C 28 20 20 00' 0108 1494 MC1_NAME: .ASCIC / (L0016) / ; ...
0A 0108
20 29 30 35 37 38 4D 28 20 20 00' 0113 1495 MA1_NAME: .ASCIC / (M8750) / ; ...
0A 0113
20 29 32 32 30 30 4C 28 20 20 00' 011E 1496 MC2_NAME: .ASCIC / (L0022) / ; ...
0A 011E
20 29 39 39 31 37 4D 28 20 20 00' 0129 1497 MA2_NAME: .ASCIC / (M7199) / ; ...
0A 0129
0134 1498
0134 1499 ;
0134 1500 ; TUSB READ PARAMETERS
0134 1501 ;
0134 1502
0000 0134 1503 TUSB_BUFF_ADDR: .WORD ; TUSB BUFFER ADDRESS
00 0136 1504 TUSB_REC_COUNT: .BYTE ; TUSB RECORD COUNT
0000 0137 1505 TUSB_REC_NUMB: .WORD ; TUSB RECORD NUMBER
0139 1506
0139 1507 ;
0139 1508 ; TUSB OPEN PARAMETERS
0139 1509 ;
0139 1510
0000 0139 1511 FILE_NAM_PTR: .WORD ; CONTAINS ADDRESS OF FILE NAME
0000 013B 1512 CURRENT_REC_NUMB: .WORD ; CONTAINS THE CURRENT TUSB RECORD #
0000 013D 1513 TOTAL_BLOCKS: .WORD ; CONTAINS FILE LENGTH IN BLOCKS
013F 1514
013F 1515 ;
013F 1516 ; TUSB ERROR PARAMETERS
013F 1517 ;
013F 1518
00 013F 1519 TUSB_ERR_CODE: .BYTE ; TUSB ERROR CODE
52 4F 52 52 45 20 38 35 55 54 3F 00' 0140 1520 TUSB_ERROR: .ASCIC /?TUSB ERROR: / ; TUSB ERROR MESSAGE
20 3A 014C
0D 0140
014E 1521
014E 1522 ;
014E 1523 ; TUSB FILE NAME STRINGS
014E 1524 ;
0000014E 014E 1525 FILE_NAME_HEAD = .
014E 1526 FILE_FOR_DPM: RAD50 <ECK>
```

```
0150 1527 RAD50 <AB >
0152 1528 RAD50 <EXE>
0154 1529 FILE_FOR_MIC: RAD50 <ECK>
0156 1530 RAD50 <AC >
0158 1531 RAD50 <EXE>
015A 1532 FILE_FOR_FPA: RAD50 <ECK>
015C 1533 RAD50 <AD >
015E 1534 RAD50 <EXE>
0160 1535 FILE_FOR_CMC: RAD50 <ECK>
0162 1536 RAD50 <AE >
0164 1537 RAD50 <EXE>
0166 1538 FILE_FOR_RDM: RAD50 <ECK>
0168 1539 RAD50 <AF >
016A 1540 RAD50 <EXE>
0000016C 016C 1541 RDM_TAIL = . ; THIS EQUATE MUST ALWAYS IMMEDIATELY
016C 1542 ; FOLLOW THE RDM FILE NAME.
00000004 016C 1543 RDM_INDEX = <<RDM_TAIL - FILE_NAME_HEAD> / 6> - 1
016C 1544
0000016C 016C 1545 FILE_NAME_TAIL = .
00000005 016C 1546 NUMBER_OF_FILES = <FILE_NAME_TAIL-FILE_NAME_HEAD>/6
016C 1547
016C 1548 ;
016C 1549 ; TERMINAL OUTPUT PARAMETERS
016C 1550 ;
016C 1551
0000 016C 1552 MESSAGE_POINTER: .WORD ; CONTAINS POINTER TO ASCII STRING
00 016E 1553 MESSAGE_LENGTH: .BYTE ; CONTAINS THE LENGTH OF ASCII MESSAGE
016F 1554
016F 1555 ;
016F 1556 ; TERMINAL INPUT PARAMETERS
016F 1557 ;
016F 1558
0000018D 016F 1559 TERM_INP_BUFF: .BLKB 30 ; INPUT BUFFER FOR TERMINAL READS
018D 1560
018D 1561 ;
018D 1562 ; TERMINAL ERROR PARAMETERS:
018D 1563 ;
018D 1564
00 018D 1565 TERM_ERR_CODE: .BYTE ; CONTAINS THE TERMINAL ERROR CODE
45 20 4C 41 4E 49 4D 52 45 54 3F 00' 018E 1566 TERM_ERROR: .ASCIC /?TERMINAL ERROR: / ; TERMINAL ERROR MESSAGE
20 3A 52 4F 52 52 019A
11 018E
01A0 1567 ;
01A0 1568 ; TYPE_NUMERIC PARAMETERS
01A0 1569 ;
01A0 1570
0000 01A0 1571 TYPEN_DATA_PTR: .WORD ; CONTAINS POINTER TO DATA
00 01A2 1572 TYPEN_DATA_TYPE: .BYTE ; CONTAINS THE DATA TYPE FLAG
000001AC 01A3 1573 TEMP_BUFFER: .BLKB 9 ; BUFFER FOR CONVERT CALL
01AC 1574
01AC 1575 ;
01AC 1576 ; WRITE_DCS PARAMETERS
01AC 1577 ;
01AC 1578
0000 01AC 1579 DCS_WRITE_ADR: .WORD ; CONTAINS DCS ADDRESS TO START AT
```

```

0000 01AE 1580 DCS_WRITE_PTR: .WORD ; CONTAINS ADDRESS OF DATA TO WRITE
      00 01B0 1581 DCS_WRITE_WDCNT: .BYTE ; CONTAINS NO. OF MICRO WORDS TO WRITE
      01B1 1582
      01B1 1583 ;
      01B1 1584 ; SINGLE_MIC_INSTR PARAMETERS:
      01B1 1585 ;
      01B1 1586
      00 01B1 1587 MIC_INSTR_CNT: .BYTE ; CONTAINS THE CYCLE COUNT
      01B2 1588
      01B2 1589 ;
      01B2 1590 ; SINGLE_TICK PARAMETERS:
      01B2 1591 ;
      01B2 1592
      00 01B2 1593 MIC_TICK_CNT: .BYTE ; CONTAINS THE TICK COUNT
      01B3 1594
      01B3 1595 ;
      01B3 1596 ; SET/CLEAR CONTROL FILE PARAMETERS:
      01B3 1597 ;
      01B3 1598
      00 01B3 1599 SETCLR_CF_ADDR: .BYTE ; CONTAINS DCS ADDRESS
      00 01B4 1600 SETCLR_CF_DATA: .BYTE ; CONTAINS THE DATA TO SET AND CLEAR
      01B5 1601
      01B5 1602 ;
      01B5 1603 ; READ OVERLAY PARAMETERS:
      01B5 1604 ;
      01B5 1605
      0000 01B5 1606 INIT_TEST_PC: .WORD ; GETS LOADED WITH BASE ADDRESS OF ST
      0000 01B7 1607 C_INIT_TEST_PC: .WORD ; GETS LOADED WITH TWO'S COMPLIMENT
      01B9 1608 ; OF BASE ADDRESS OF TEST
      00 01B9 1609 TEST_SIZE: .BYTE ; CONTAINS SIZE OF CURRENT TEST IN
      01BA 1610 ; RECORDS
      01BA 1611 ;
      01BA 1612 ; NOTE: ALL MICRO WORDS IN THIS PROGRAM MUST BE CONTIGUOUS IN THE FOLLOWING
      01BA 1613 ; BLOCK SINCE THEY ARE SCRAMBLED WHEN THE MONITOR STARTSUP.
      01BA 1614 ;
  
```

```

000001BA 01BA 1615 MICRO_CODE_HEAD =
F7 08 00 03 64 02 00 04 70 18 00 01BA 1616 NOP_MICRO_WORD: .BYTE 0, ^X18, ^X70, 04, 00, 02, ^X64, 03, 00, ^X08, ^XF7
D7 48 40 03 64 02 00 04 70 18 00 01C5 1617 R_RDM_MIC_WORD: .BYTE 0, ^X18, ^X70, 04, 00, 02, ^X64, 03, ^X40, ^X48, ^XD7
E7 08 00 02 47 02 00 04 70 18 00 01D0 1618 RDM_R_MIC_WORD: .BYTE 0, ^X18, ^X70, 04, 00, 02, ^X47, 02, 00, 08, ^XE7
67 08 00 59 24 03 00 04 70 18 00 01DB 1619 RDM_M_MIC_WORD: .BYTE 0, ^X18, ^X70, 04, 00, 03, ^X24, ^X59, 00, 08, ^X67
67 48 13 59 24 03 00 4C 70 18 00 01E6 1620 RDM_WD_MIC_WORD: .BYTE 0, ^X18, ^X70, ^X4C, 00, 03, ^X24, ^X59, ^X13, ^X48, ^X67
57 48 00 03 64 03 00 68 70 18 00 01F1 1621 RDM_SR_MIC_WORD: .BYTE 0, ^X18, ^X70, ^X68, 00, 03, ^X64, 03, 00, ^X48, ^X57
67 48 00 03 64 03 00 64 70 18 00 01FC 1622 .BYTE 0, ^X18, ^X70, ^X64, 00, 03, ^X64, 03, 00, ^X48, ^X67
67 48 00 03 64 03 00 08 70 18 00 0207 1623 RDM_PS_MIC_WORD: .BYTE 0, ^X18, ^X70, 08, 00, 03, ^X64, 03, 00, ^X48, ^X67
67 C8 00 03 64 03 00 18 70 18 00 0212 1624 RDM_ST_MIC_WORD: .BYTE 0, ^X18, ^X70, ^X18, 00, 03, ^X64, 03, 00, ^XC8, ^X67
67 48 00 03 64 03 00 38 70 18 00 021D 1625 RDM_SF_MIC_WORD: .BYTE 0, ^X18, ^X70, ^X38, 00, 03, ^X64, 03, 00, ^X48, ^X67
  
```

```

0228 1626
0228 1627 ;
0228 1628 ; THE FOLLOWING MICRO WORDS DISABLE THE CMI AND CLEAR THE PSL
0228 1629 ;
0228 1630 ; BUS/PRINIT
0228 1631 ; LONLIT [0E000000]
0228 1632 ; MEMSCAR LONLIT
0228 1633 ; LONLIT [01000000]
0228 1634 ; MEMSCR_LONLIT
  
```

```
0228 1635 ; LONLIT_[0],WCTRL/PC_WB
0228 1636 ; PSL_R[LONLIT]
0228 1637
F7 08 00 03 64 02 00 04 10 18 00 0228 1638 LOAD_MEMSCAR: .BYTE 0,^X18,^X10, 04, 00, 02,^X64, 03, 00, 08,^XF7
F7 B8 00 78 FF FF FF 84 70 18 00 0233 1639 .BYTE 0,^X18,^X70,^X84,^XFF,^XFF,^XFF,^X78, 00,^XB8,^XF7
F7 48 00 00 47 02 D4 68 70 18 00 023E 1640 .BYTE 0,^X18,^X70,^X68,^XD4, 02,^X47, 00, 00,^X48,^XF7
F7 B8 00 7F 7F FF FF 84 70 18 00 0249 1641 LOAD_MEMSCR: .BYTE 0,^X18,^X70,^X84,^XFF,^XFF,^X7F,^X7F, 00,^XB8,^XF7
F7 C8 00 00 47 02 D4 60 70 18 00 0254 1642 .BYTE 0,^X18,^X70,^X60,^XD4, 02,^X47, 00, 00,^XC8,^XF7
F7 78 00 7F FF FF FF C8 70 18 00 025F 1643 .BYTE 0,^X18,^X70,^XC8,^XFF,^XFF,^XFF,^X7F, 00,^X78,^XF7
F7 C8 00 5B F4 02 D4 00 70 18 00 026A 1644 .BYTE 0,^X18,^X70, 00,^XD4, 02,^XF4,^X5B, 00,^XC8,^XF7
0275 1645
0275 1646 ;
0275 1647 ; THE FOLLOWING MICRO WORDS ARE USED TO WRITE THE CACHE
0275 1648 ;
0275 1649 ; LONLIT_[06000000]
0275 1650 ;
0275 1651 ; LONLIT_[0]
0275 1652 ;
0275 1653 ; VA_RDM
0275 1654 ; WB_RDM,WRITE,CLKX
0275 1655 ; WB_RDM,WCTRL/WDR_WB
0275 1656 ; VA_VA+4
0275 1657
F7 F8 00 7C FF FF FF 84 70 18 00 0275 1658 CACHE_ENA_ADR: .BYTE 0,^X18,^X70,^X84,^XFF,^XFF,^XFF,^X7C, 00,^XF8,^XF7
0280 1659 MME_DIS_ADR:
0280 1660 MME_DIS_DATA:
F7 F8 00 7F FF FF FF 84 70 18 00 0280 1661 CACHE_ENA_DATA: .BYTE 0,^X18,^X70,^X84,^XFF,^XFF,^XFF,^X7F, 00,^XF8,^XF7
028B 1662
D7 08 00 03 64 02 00 4A 70 18 00 028B 1663 CACHE_WRITE: .BYTE 0,^X18,^X70,^X4A, 00, 02,^X64, 03, 00, 08,^XD7
D7 88 00 03 64 02 00 5D 80 98 00 0296 1664 .BYTE 0,^X98,^X80,^X5D, 00, 02,^X64, 03, 00,^X88,^XD7
D7 88 00 03 64 02 00 5C 70 18 00 02A1 1665 .BYTE 0,^X18,^X70,^X5C, 00, 02,^X64, 03, 00,^X88,^XD7
F7 88 00 03 64 02 00 44 70 18 00 02AC 1666 .BYTE 0,^X18,^X70,^X44, 00, 02,^X64, 03, 00,^X88,^XF7
000002B7 02B7 1667 MICRO_CODE_TAIL =
00000017 02B7 1668 NO_MICRO_WORDS = <MICRO_CODE_TAIL - MICRO_CODE_HEAD>/11
02B7 1669
02B7 1670 ;
02B7 1671 ; TYPE_TEST_NUMBER PARAMETERS:
02B7 1672 ;
00 02B7 1673 TYPE_TEST_FLAG: .BYTE 0 ; INDICATES IF FIRST TEST NUMBER HAS
02B8 1674 ; BEEN TYPED
00 02B8 1675 TESTS_PER_LINE: .BYTE 0 ; CONTAINS THE NUMBER OF TEST NUMBERS
02B9 1676 ; TYPED ON THE CURRENT LINE
0000 02B9 1677 TITLE_LENGTH: .WORD 0 ; CONTAINS LENGTH OF TEST TITLE
20 54 53 45 54 00 02BB 1678 TEST_MSG_1: .ASCII /TEST /
05 02BB
20 3A 00 02C1 1679 TEST_MSG_2: .ASCII /: /
02 02C1
02C4 1680
02C4 1681 ;
02C4 1682 ; READ_V_BUS PARAMETERS:
02C4 1683 ;
02C4 1684
000002EC 02C4 1685 BIT_SLICE_TEMP: ; TEMPORARY STORAGE FOR BIT SLICE ROUTINE
02C4 1686 V_BUS_BUFFER: .BLKB 40 ; BUFFER FOR THE V BUS BITS
02EC 1687
```

```
02EC 1688 ;  
02EC 1689 ; GET_CMD_LINE PARAMETERS:  
02EC 1690 ;  
02EC 1691 ;  
3E 43 49 4D 00' 02EC 1692 PROMPT: .ASCIC /MIC>/ ; DIAGNOSTIC PROMPT  
04 02EC  
1E 02F1 1693 CMD_LINE_LENGTH: .BYTE 30 ; NUMBER OF CHARACTERS IN COMMAND LINE  
01 02F2 1694 ONE: .BYTE 1 ; NUMBER OF CHARACTERS FOR OTHER READS  
4F 43 20 44 49 4C 41 56 4E 49 3F 00' 02F3 1695 INVALID_COMMAND: .ASCIC /?INVALID COMMAND/ ; COMMAND LINE ERROR MESSAGE  
44 4E 41 4D 4D 02FF  
10 02F3  
0304 1696 CMD_DISPATCH: ADDRESS <DIAGNOSE_ACTION> ; POINTER TO DIAGNOSE ACTION ROUTINE  
0306 1697 ADDRESS <SHOW_ACTION> ; POINTER TO SHOW ACTION ROUTINE  
0308 1698 ADDRESS <CONTINUE_ACTION> ; POINTER TO CONTINUE ACTION ROUTINE  
030A 1699 ADDRESS <RETURN_ACTION> ; POINTER TO RETURN ACTION ROUTINE  
030C 1700 ADDRESS <SET_ACTION> ; POINTER TO SET ACTION ROUTINE  
030E 1701 ADDRESS <CLEAR_ACTION> ; POINTER TO CLEAR ACTION ROUTINE  
0310 1702 ADDRESS <LOOP_ACTION> ; POINTER TO LOOP ACTION ROUTINE  
0312 1703 ADDRESS <EXAMINE_ACTION>  
0314 1704 ADDRESS <DEPOSIT_ACTION>  
0316 1705 ADDRESS <BREAK_ACTION>  
0318 1706  
0318 1707 ;  
0318 1708 ; PARSE_CMD_LINE PARAMETERS:  
0318 1709 ;  
0318 1710  
0318 1711 PARSE_TBL_PTR: ADDRESS <PARSE_TABLE> ; CONTAINS ADDRESS OF PARSE TABLE  
0000032E 031A 1712 PARSE_TABLE: .BLKB 20 ; PARSE TABLE  
49 44 032E 1713 COMMAND_LIST: .ASCII /DI/ ; DIAGNOSE  
48 53 0330 1714 .ASCII /SH/ ; SHOW  
4F 43 0332 1715 .ASCII /CO/ ; CONTINUE  
45 52 0334 1716 .ASCII /RE/ ; RETURN  
45 53 0336 1717 .ASCII /SE/ ; SET  
4C 43 0338 1718 .ASCII /CL/ ; CLEAR  
4F 4C 033A 1719 .ASCII /LO/ ; LOOP  
58 45 033C 1720 .ASCII /EX/ ; EXAMINE  
45 44 033E 1721 .ASCII /DE/ ; DEPOSIT  
52 42 0340 1722 .ASCII /BR/ ;  
FF 0342 1723 .BYTE -1 ; LIST TERMINATOR  
0343 1724  
45 54 0343 1725 KEYWORD_LIST: .ASCII /TE/ ; TEST  
41 50 0345 1726 .ASCII /PA/ ; PASS  
4C 46 0347 1727 .ASCII /FL/ ; FLAG  
4F 53 0349 1728 .ASCII /SO/ ; SOMM  
034B 1729  
41 48 034B 1730 HALT_FLAG_NAME: .ASCII /HA/ ; HALT  
4F 4C 034D 1731 .ASCII /LO/ ; LOOP  
45 4E 034F 1732 .ASCII /NE/ ; NER  
45 42 0351 1733 .ASCII /BE/ ; BELL  
41 53 0353 1734 .ASCII /SA/ ; SINGATURE ANALYZER  
42 49 0355 1735 .ASCII /IB/ ; INHIBIT BURST FLAG  
41 51 0357 1736 .ASCII /QA/ ; QA FLAG  
52 54 0359 1737 .ASCII /TR/ ; TRACE FLAG  
035B 1738  
4E 49 035B 1739 .ASCII /IN/ ; INSTRUCTION
```

```
54 4C 035D 1740 .ASCII /LT/ ; LOOP ON TEST
4F 43 035F 1741 .ASCII /CO/ ; CONTINUE
54 53 0361 1742 .ASCII /ST/ ; STEP
59 43 0363 1743 .ASCII /CY/ ; CYCLE
49 54 0365 1744 .ASCII /TI/ ; TICK
46 43 0367 1745 .ASCII /CF/ ; CONTROL FILE
42 56 0369 1746 .ASCII /VB/ ; VISIBILITY BUS
56 51 036B 1747 .ASCII /QV/ ; QUICK VERIFY
4D 44 036D 1748 .ASCII /DM/ ; RDM TESTS
54 52 036F 1749 .ASCII /RT/ ; R TEMPS
54 4D 0371 1750 .ASCII /MT/ ; M TEMPS
43 50 0373 1751 .ASCII /PC/ ; PC REGISTER
41 56 0375 1752 .ASCII /VA/ ; VA
44 4D 0377 1753 .ASCII /MD/ ; MDR
42 50 0379 1754 .ASCII /PB/ ; PC BACKUP
41 4D 037B 1755 .ASCII /MA/ ; MEMORY ADDRESS REG
44 57 037D 1756 .ASCII /WD/ ; WDR
52 53 037F 1757 .ASCII /SR/ ; STATUS & CONTROL
53 50 0381 1758 .ASCII /PS/ ; PSL
46 53 0383 1759 .ASCII /SF/ ; FLAGS
42 53 0385 1760 .ASCII /SB/ ; ENABLE SINGLE BIT ERROR REPORT
FF 0387 1761 .BYTE -1 ; LIST TERMINATOR
0388 1762
0388 1763 ;
0388 1764 ; PARSE_HEX_NUMB PARAMETERS:
0388 1765 ;
0388 1766
00000000 0388 1767 PARSE_HEX_TEMP: .LONG 0 ; WORKING SPACE TO MAKE 32 BIT NUMBER
038C 1768
038C 1769 ;
038C 1770 ; DIAGNOSE ACTION ROUTINE PARAMETERS:
038C 1771 ;
038C 1772
00 038C 1773 SPEC_TEST_NUMB: .BYTE ; CONTAINS STARTING TEST NO. FOR LOST
00 038D 1774 TEST_SPAN_END: .BYTE ; CONTAINS END TEST NO. FO 'TEST SPAN'
01 038E 1775 USER_PASS_CNT: .BYTE 1 ; CONTAINS THE USER SPECIFIED PASS CNT
038F 1776
038F 1777 ;
038F 1778 ; SET/CLEAR ACTION ROUTINE PARAMETERS:
038F 1779 ;
038F 1780
0000 038F 1781 SOMM_ADDRESS: .WORD ; CONTAINS ADDRESS OF A SOMM
0391 1782
0391 1783 ;
0391 1784 ; SHOW ACTION ROUTINE PARAMETERS:
0391 1785 ;
0391 1786
20 3A 54 45 53 20 53 47 41 4C 46 00' 0391 1787 SHOW_MESSAGE_1: .ASCIC /FLAGS SET: /
0B 0391
52 41 45 4C 43 20 53 47 41 4C 46 00' 039D 1788 SHOW_MESSAGE_2: .ASCIC /FLAGS CLEAR: /
20 3A 03A9
0D 039D
00 03AB 1789 SHOW_TEMP: .BYTE ; TEMP LOCATION FOR THIS ROUTINE
20 3D 53 55 42 56 20 20 20 00' 03AC 1790 VBUS_MSG: .ASCIC / VBUS= /
09 03AC
```

```

03B6 1791
03B6 1792 ;
03B6 1793 ; OP_CODE_DISPATCH PARAMETERS:
03B6 1794 ;
03B6 1795
0000 03B6 1796 TEST_PC: .WORD ; CONTAINS THE TEST STREAM POINTER OF
03B8 1797 ; THE NEXT PSEUDO INSTRUCTION
0000 03B8 1798 CURRENT_PC: .WORD ; CONTAINS THE TEST STREAM POINTER OF
03BA 1799 ; THE CURRENT PSEUDO INSTRUCTION
0000 03BA 1800 SPEC_TEST_PC: .WORD ; CONTAINS THE USER DEFINED TEST PC
0000 03BC 1801 EXAMINE_TEMP: .WORD ; WORKING LOACTION FOR 'EXAMINE'
03BE 1802
000003CE 03BE 1803 ARG_LIST: .BLKB MAX_ARGUMENTS ; RECEIVES THE PSEUDO INSTR ARGUMENTS
03CE 1804 INSTR_TABLE: ADDRESS <LOAD_DCS> ; 0
03D0 1805 ADDRESS <BURST_CLOCK> ; 2
03D2 1806 ADDRESS <COMPARE_REG> ; 4
03D4 1807 ADDRESS <COMPARE_REG_MSK>; 6
03D6 1808 ADDRESS <BEGIN_LOOP> ; 8
03D8 1809 ADDRESS <END_LOOP> ; 10
03DA 1810 ADDRESS <END_TEST> ; 12
03DC 1811 ADDRESS <ERROR_LOOP> ; 14
03DE 1812 ADDRESS <PATTERN_GEN> ; 16
03E0 1813 ADDRESS <IF_ERROR> ; 18
03E2 1814 ADDRESS <INIT_THE_WORLD>; 20
03E4 1815 ADDRESS <LOAD_REG> ; 22
03E6 1816 ADDRESS <MASK> ; 24
03E8 1817 ADDRESS <NEW_TEST> ; 26
03EA 1818 ADDRESS <SKIP> ; 28
03EC 1819 ADDRESS <SUB_TEST> ; 30
03EE 1820 ADDRESS <COMPARE_VBUS> ; 32
03F0 1821 ADDRESS <FETCH_MIC_INSTR>; 34
03F2 1822 ADDRESS <END_PASS> ; 36
03F4 1823 ADDRESS <LOAD_FIELD> ; 42
03F6 1824 ADDRESS <LOAD_INDEX> ; 44
03F8 1825 ADDRESS <INCREMENT_INDEX>; 46
03FA 1826 ADDRESS <SAVE_INDEX> ; 48
03FC 1827 ADDRESS <ENABLE_TRAP> ; 50
03FE 1828 ADDRESS <DUMP_LOG> ; 52
0400 1829 ADDRESS <ERROR_LOG> ; 54
0402 1830 ADDRESS <PRINT_S> ; 56
0404 1831 ADDRESS <PRINT_N> ; 58
0406 1832 ADDRESS <ENABLE_STALL> ; 60
0408 1833 ;
0408 1834 ; BEGIN_LOOP AND END_LOOP PARAMETERS:
0408 1835 ;
0408 1836
0408 1837 INDEX_NAM_TBL: ADDRESS <I_INDEX_TBL>
040A 1838 ADDRESS <J_INDEX_TBL>
040C 1839 ADDRESS <K_INDEX_TBL>
0000 040E 1840 I_INDEX_TBL: .WORD ; CONTAINS CURRENT VALUE OF INDEX
0000 0410 1841 .WORD ; CONTAINS END VALUE OF INDEX
00 0412 1842 .BYTE ; CONTAINS THE INCREMENT VALUE
0000 0413 1843 .WORD ; CONTAINS THE TEST PC OF START OF LOOP
0000041C 0415 1844 J_INDEX_TBL: .BLKB 7
00000423 041C 1845 K_INDEX_TBL: .BLKB 7

```

```
0423 1846
0423 1847 ;
0423 1848 ; NEW_TEST ROUTINE PARAMETERS:
0423 1849 ;
0423 1850
0000 0423 1851 LOOP_ADDRESS: .WORD ; CONTAINS THE TEST PC FOR LOOP ON TEST
0000 0425 1852 ERROR_LOOP_ADDR: .WORD ; CONTAINS THE TEST PC FOR A 'SCOPE' LOOP
0427 1853
0427 1854 ;
0427 1855 ; PATTERN GENERATE ROUTINE PARAMETERS:
0427 1856 ;
0427 1857
AAAAAAA 0427 1858 PATTERN_ADDRESS: .LONG ^XAAAAAAA ; THESE ARE THE 6 TEST PATTERNS
5555555 042B 1859 .LONG ^X5555555 ; USED BY THE PATTERN GENERATE
3333333 042F 1860 .LONG ^X3333333 ; PSEUDO INSTRUCTION
0F0F0F0F 0433 1861 .LONG ^X0F0F0F0F ; ...
00FF00FF 0437 1862 .LONG ^X00FF00FF ; ...
0000FFFF 043B 1863 .LONG ^X0000FFFF ; ...
043F 1864
043F 1865 ;
043F 1866 ; COMPARE PARAMETERS:
043F 1867 ;
043F 1868
00000443 043F 1869 GOOD_DATA: .BLKB 4 ; CONTAINS THE EXPECTED DATA OF A COMPARE -
cont> ROUTINE
00000447 0443 1870 BAD_DATA: .BLKB 4 ; CONTAINS THE RECIEVED DATA OF A COMPARE -
cont> ROUTINE
00 0447 1871 DATA_TYPE: .BYTE ; CONTAINS THE NUMBER OF BYTES
0448 1872 ; IN GOOD AND BAD DATA
00 0448 1873 MODE_TYPE: .BYTE ; CONTAINS THE MODE FLAG OF THE
0449 1874 ; LAST CMPREG OR CMPREGMSK
FFFF 0449 1875 QA_COUNT: .WORD -1 ; CONTAINS INDEX COUNT FOR QA FLAG
0000 044B 1876 QA_INDEX: .WORD ; GETS LOADED WITH CURRENT INDEX VALUE
044D 1877 ; IN 'COMPARE' ROUTINES
0000 044D 1878 VBUS_TBL_ADDR: .WORD ; CONTAINS ADDRESS OF VBUS TABLE IN
044F 1879 ; THE TEST OVERLAY
00 044F 1880 VBUS_TBL_COUNT: .BYTE ; CONTAINS NUMBER OF BITS NOT COMPARED
0450 1881 ; BY THE CMPVBUS PSEUDO INSTRUCTION
0450 1882
0450 1883 ;
0450 1884 ; BURST CLOCK PARAMETERS:
0450 1885 ;
0450 1886
00 0450 1887 STEP_ADDRESS: .BYTE ; CONTAINS THE DCS ADDRESS THAT IS
0451 1888 ; TYPED FOR SINGLE TICK AND CYCLE
0000 0451 1889 BURST_TEMP: .WORD ; TEMPORARY STORAGE FOR THIS ROUTINE
0000 0453 1890 CLK_MSG_ADDR: .WORD ; GETS LOADED WITH A MESSAGE ADDRESS
00 0455 1891 M_CLK_ADDR: .BYTE ; USED TO CHECK IF M CLOCK STILL RUNNING
52 44 44 41 20 53 43 44 20 20 20 00 0456 1892 DCS_ADDR_MSG: .ASCIC / DCS ADDR= /
20 3D 0462
0D 0456
4B 41 45 52 42 20 4F 52 43 49 4D 00 0464 1893 SOMM_MSG: .ASCIC /MICRO BREAK MATCH, CS ADDR= /
41 20 53 43 20 2C 48 43 54 41 4D 20 0470
20 3D 52 44 44 047C
1C 0464
20 44 45 54 43 45 50 58 45 4E 55 00 0481 1894 CLK_STOP_MSG1: .ASCIC /UNEXPECTED CLOCK STOP. DCS ADDR= /
20 2E 50 4F 54 53 20 4B 43 4F 4C 43 048D
```

```
20 3D 52 44 44 41 20 53 43 44 0499
21 0481
4D 55 4E 20 54 53 45 54 20 20 20 00 04A3 1895 CLK_STOP_MSG2: .ASCIC / TEST NUMBER= /
20 3D 52 45 42 04AF
10 04A3
20 3D 43 50 54 00 04B4 1896 CLK_STOP_MSG3: .ASCIC /TPC= /
05 04B4
44 44 41 20 50 41 52 54 20 20 20 00 04BA 1897 CLK_STOP_MSG4: .ASCIC / TRAP ADDRESS= /
20 3D 53 53 45 52 04C6
11 04BA
4E 55 48 20 4B 43 4F 4C 43 20 4D 00 04CC 1898 CLK_STOP_MSG5: .ASCIC /M CLOCK HUNG. DCS ADDR= /
3D 52 44 44 41 20 53 43 44 20 2E 47 04D8
20 04E4
18 04CC
04E5 1899
04E5 1900 ;
04E5 1901 ; ERROR LOG AND LOG DUMP PARAMETERS AND MESSAGES
04E5 1902 ;
04E5 1903 ;
000004E8 04E5 1904 LOG_ADDRESS: .BLKB 3 ; TEMPORARY STORAGE FOR ERROR ADDRESS
00 04E8 1905 LOG_SYNDROME: .BYTE ; TEMPORARY STORAGE FOR ERROR SYNDROME
4C 46 52 45 56 4F 20 54 53 45 54 00 04E9 1906 LOG_MSG_1: .ASCIC /TEST OVERFLOWS INTO ERROR LOG/
52 52 45 20 4F 54 4E 49 20 53 57 4F 04F5
47 4F 4C 20 52 4F 0501
1D 04E9
43 20 46 4F 20 52 45 42 4D 55 4E 00 0507 1907 LOG_MSG_2: .ASCIC /NUMBER OF CORRECTABLE BITS IN THIS ARRAY = /
42 20 45 4C 42 41 54 43 45 52 52 4F 0513
20 53 49 48 54 20 4E 49 20 53 54 49 051F
20 3D 20 59 41 52 52 41 052B
2B 0507
41 48 54 20 52 45 54 41 45 52 47 00 0533 1908 LOG_MSG_3: .ASCIC /GREATER THAN 64 CORRECTABLE BITS IN THIS ARRAY, /-
54 43 45 52 52 4F 43 20 34 36 20 4E 053F
4E 49 20 53 54 49 42 20 45 4C 42 41 054B
2C 59 41 52 52 41 20 53 49 48 54 20 0557
45 54 52 4F 42 41 20 54 53 45 54 20 0563
44 056F
3C 0533
0570 1909 /TEST ABORTED/
0570 1910
0570 1911 ;
0570 1912 ; PARITY CALCULATION PARAMETERS:
0570 1913 ;
0570 1914 ;
DD 09 F1 B3 BF F6 99 00 95 00 0570 1915 P0_PARITY_MSK: .BYTE 00,^X95, 00,^X99,^XF6,^XBF,^XB3,^XF1, 09,^XDD
22 F6 0E 4C 00 09 66 F7 6A FF 057A 1916 P1_PARITY_MSK: .BYTE ^XFF,^X6A,^XF7,^X66, 09, 00,^X4C,^X0E,^XF6,^X22
0584 1917
0584 1918 ;
0584 1919 ; END PASS PARAMETERS:
0584 1920 ;
0584 1921 ;
53 53 41 50 20 46 4F 20 44 4E 45 00 0584 1922 ENDPASS_MSG: .ASCIC /END OF PASS / ; END OF PASS MESSAGE
20 0590
0C 0584
0591 1923
0591 1924 ;
```

```
0591 1925 ; RING_BELL PARAMETERS:
0591 1926 ;
0591 1927 ;
00 0591 1928 BELL_COUNT: .BYTE ; CONTAINS THE NUMBER OF TIMES THE
0592 1929 ; RING_BELL ROUTINE HAS BEEN CALLED
07 01 0592 1930 BELL: .BYTE 1,7 ; ASCII MESSAGE FOR A BELL
0594 1931 ;
0594 1932 ;
0594 1933 ; TYPE_ERROR PARAMETERS:
0594 1934 ;
0594 1935 ;
00000596 0594 1936 ERROR_PC: .BLKB 2 ; CONTAINS THE TEST PC OF AN 'IFERROR'
0596 1937 ; INSTRUCTION FOR FAILURE TYPE OUT
0000059A 0596 1938 TYPE_ERR_BUFFER: .BLKB 4 ; BUFFER FOR ERROR DATA TYPE OUT
00 059A 1939 TYPE_ERR_TEMP: .BYTE ; TEMP STORAGE FOR ROUTINE
20 3A 52 4F 52 52 45 3F 00' 059B 1940 ERROR_MSG: .ASCIC /?ERROR: /
08 059B
20 3A 54 53 45 54 20 00' 05A4 1941 TEST_MSG: .ASCIC / TEST: /
07 05A4
20 3A 54 53 45 54 42 55 53 20 00' 05AC 1942 SUB_TEST_MSG: .ASCIC / SUBTEST: /
0A 05AC
20 20 20 3A 41 54 41 44 00' 05B7 1943 DATA_MSG: .ASCIC /DATA: /
08 05B7
54 4F 4E 20 00' 05C0 1944 NOT_MSG: .ASCIC / NOT/
04 05C0
20 20 20 20 20 20 20 20 00' 05C5 1945 TAB: .ASCIC / /
08 05C5
2D 00' 05CE 1946 HYPHEN: .ASCIC /-/
01 05CE
41 54 41 44 20 53 55 42 56 00' 05D0 1947 VBUS_MSG_2: .ASCIC /VBUS DATA/
09 05D0
54 41 47 20 47 4E 49 4C 49 41 46 00' 05DA 1948 FAIL_ARRAY_MSG: .ASCIC /FAILING GATE ARRAYS: /
20 3A 53 59 41 52 52 41 20 45 05E6
15 05DA
20 20 20 20 20 20 20 20 20 20 00' 05F0 1949 FAIL_ARR_INDEX: .ASCIC / /
20 20 20 20 20 20 20 20 20 20 05FC
15 05F0
44 4F 4D 20 47 4E 49 4C 49 41 46 00' 0606 1950 MODULE_MSG: .ASCIC /FAILING MODULES: /
20 3A 53 45 4C 55 0612
11 0606
20 20 20 20 20 20 20 20 20 20 00' 0618 1951 FAIL_MOD_INDEX: .ASCIC / /
20 20 20 20 20 20 0624
11 0618
2C 00' 062A 1952 COMMA_MSG: .ASCIC /,/
01 062A
062C 1953
062C 1954 ;
062C 1955 ; GATE ARRAY NAME LIST
062C 1956 ;
062C 1957 ;
43 44 00' 062C 1958 DC_MSG: .ASCIC /DC/
02 062C
29 4B 4C 41 28 35 31 36 00' 062F 1959 GATE_ARRAY_LST: .ASCIC /615(ALK)/ ; ALK
08 062F
29 4B 52 53 28 34 31 36 00' 0638 1960 .ASCIC /614(SRK)/ ; SRK
```

29	41	50	53	28	36	31	36	00	08	0641	1961	.ASCIC	/616(SPA)/	; SPA
29	41	4C	43	28	32	31	36	00	08	064A	1962	.ASCIC	/612(CLA)/	; CLA
29	43	43	43	28	30	31	36	00	08	0653	1963	.ASCIC	/610(CCC)/	; CCC
29	44	52	49	28	32	32	36	00	08	065C	1964	.ASCIC	/622(IRD)/	; IRD
29	4B	4F	54	28	30	32	36	00	08	0665	1965	.ASCIC	/620(TOK)/	; TOK
29	42	48	50	28	39	32	36	00	08	066E	1966	.ASCIC	/629(PHB)/	; PHB
29	51	53	4D	28	31	32	36	00	08	0677	1967	.ASCIC	/621(MSQ)/	; MSQ
29	43	41	53	28	37	31	36	00	08	0680	1968	.ASCIC	/617(SAC)/	; SAC
29	4B	41	43	28	37	32	36	00	08	0689	1969	.ASCIC	/627(CAK)/	; CAK
29	4B	44	41	28	36	32	36	00	08	0692	1971	.ASCIC	/626(ADK)/	; ADK
29	4B	52	50	28	34	32	36	00	08	069B	1972	.ASCIC	/624(PRK)/	; PRK
29	52	54	55	28	38	32	36	00	08	06A4	1973	.ASCIC	/628(UTR)/	; UTR
29	4C	4D	43	28	31	35	36	00	08	06AD	1974	.ASCIC	/651(CML)/	; CML
29	56	43	41	28	35	32	36	00	08	06B6	1975	.ASCIC	/625(ACV)/	; ACV
29	54	4E	49	28	30	33	36	00	08	06BF	1976	.ASCIC	/630(INT)/	; INT
29	4E	4F	43	28	31	31	36	00	08	06C8	1978	.ASCIC	/611(CON)/	; CON
29	50	41	4D	28	32	33	36	00	08	06D1	1979	.ASCIC	/632(MAP)/	; MAP
29	44	41	4D	28	30	35	36	00	08	06DA	1981	.ASCIC	/650(MAD)/	; MAD
29	41	51	46	28	32	34	36	00	08	06E3	1982	.ASCIC	/642(FQA)/	; FQA
29	43	43	46	28	33	34	36	00	08	06EC	1984	.ASCIC	/643(FCC)/	; FCC

06F5 1985 ;
06F5 1986 ; THE FOLLOWING GATE ARRAYS ARE BIT SLICED AND THEREFORE HAVE UNIQUE
06F5 1987 ; CALLOUT MESSAGES
06F5 1988 ;
06F5 1989 ;
06F5 1990 BIT_SLICE_TBL:: ADDRESS <ALP_NAME>
06F7 1991 ADDRESS <SRM_NAME>
06F9 1992 ADDRESS <MDR_NAME>
06FB 1993 ADDRESS <ADD_NAME>
06FD 1994 ADDRESS <MEC_NAME>

Address	Hex	Label	Value
04	0A4C	UDP_NAME::	.BYTE 4
03030303	0A4D		.LONG ^X03030303
0C0C0C0C	0A51		.LONG ^X0C0C0C0C
30303030	0A55		.LONG ^X30303030
C0C0C0C0	0A59		.LONG ^XC0C0C0C0
0A5D	2089		
0A5D	2090		.ASCIC /618(UDP 1 <00,01,08,09,16,17,24,25>)/ ; UDP
3C 20 31 20 50 44 55 28 38 31 36 00	0A69		
2C 39 30 2C 38 30 2C 31 30 2C 30 30	0A75		
3E 35 32 2C 34 32 2C 37 31 2C 36 31	0A81		
29	0A81		
24	0A5D		
3C 20 32 20 50 44 55 28 38 31 36 00	0A82	2091	.ASCIC /618(UDP 2 <02,03,10,11,18,19,26,27>)/ ; UDP
2C 31 31 2C 30 31 2C 33 30 2C 32 30	0A8E		
3E 37 32 2C 36 32 2C 39 31 2C 38 31	0A9A		
29	0AA6		
24	0A82		
3C 20 33 20 50 44 55 28 38 31 36 00	0AA7	2092	.ASCIC /618(UDP 3 <04,05,12,13,20,21,28,29>)/ ; UDP
2C 33 31 2C 32 31 2C 35 30 2C 34 30	0AB3		
3E 39 32 2C 38 32 2C 31 32 2C 30 32	0ABF		
29	0ACB		
24	0AA7		
3C 20 34 20 50 44 55 28 38 31 36 00	0ACC	2093	.ASCIC /618(UDP 4 <06,07,14,15,22,23,30,31>)/ ; UDP
2C 35 31 2C 34 31 2C 37 30 2C 36 30	0AD8		
3E 31 33 2C 30 33 2C 33 32 2C 32 32	0AE4		
29	0AF0		
24	0ACC		
0AF1	2094		
08	0AF1	FIO_NAME::	.BYTE 8
0000000F	0AF2		.LONG ^X0000000F
000000F0	0AF6		.LONG ^X000000F0
00000F00	0AFA		.LONG ^X00000F00
0000F000	0AFE		.LONG ^X0000F000
000F0000	0B02		.LONG ^X000F0000
00F00000	0B06		.LONG ^X00F00000
0F000000	0B0A		.LONG ^X0F000000
F0000000	0B0E		.LONG ^XF00000000
0B12	2104		
0B12	2105		.ASCIC /636(FIO 0)/ ; FIO
29 30 20 4F 49 46 28 36 33 36 00	0A	0B12	
29 31 20 4F 49 46 28 36 33 36 00	0A	0B1D	2106
29 32 20 4F 49 46 28 36 33 36 00	0A	0B1D	
29 33 20 4F 49 46 28 36 33 36 00	0A	0B28	2107
29 34 20 4F 49 46 28 36 33 36 00	0A	0B28	
29 35 20 4F 49 46 28 36 33 36 00	0A	0B33	2108
29 36 20 4F 49 46 28 36 33 36 00	0A	0B33	
29 37 20 4F 49 46 28 36 33 36 00	0A	0B3E	2109
29 38 20 4F 49 46 28 36 33 36 00	0A	0B3E	
29 39 20 4F 49 46 28 36 33 36 00	0A	0B49	2110
29 40 20 4F 49 46 28 36 33 36 00	0A	0B49	
29 41 20 4F 49 46 28 36 33 36 00	0A	0B54	2111
29 42 20 4F 49 46 28 36 33 36 00	0A	0B54	
29 43 20 4F 49 46 28 36 33 36 00	0A	0B5F	2112
29 44 20 4F 49 46 28 36 33 36 00	0A	0B5F	
0B6A	2113		
02	0B6A	FEX_NAME::	.BYTE 2

	00007000	OB6B	2115	.LONG	^X00007000		
	00000F80	OB6F	2116	.LONG	^X00000F80		
		OB73	2117				
29 31 20 58 45 46 28 31 34 36 00	0A	OB73	2118	.ASCIC	/641(FEX 1)/		; FEX
29 30 20 58 45 46 28 31 34 36 00	0A	OB7E	2119	.ASCIC	/641(FEX 0)/		; FEX
		OB7E					
	01	OB89	2120				
	FFFFFFFF	OB89	2121	FIC_NAME::	.BYTE	1	
		OB8A	2122	.LONG	^XFFFFFFFF		
		OB8E	2123				
20 52 54 52 43 4E 49 28 32 31 36 00	29 41 4C 43	OB8E	2124	.ASCIC	/612(INCRTR CLA)/		; FIC
	0F	OB9A					
		OB8E					
	01	OB9E	2125				
	FFFFFFFF	OB9E	2126	FAC_NAME::	.BYTE	1	
		OB9F	2127	.LONG	^XFFFFFFFF		
		OBA3	2128				
41 4C 43 20 55 4C 41 28 32 31 36 00	29	OBA3	2129	.ASCIC	/612(ALU CLA)/		; FAC
	0C	OBAF					
		OBA3					
	04	OB80	2130				
	11111111	OB80	2131	FCS_NAME::	.BYTE	4	
	22222222	OB81	2132	.LONG	^X11111111		
	44444444	OB85	2133	.LONG	^X22222222		
	88888888	OB89	2134	.LONG	^X44444444		
		OBBD	2135	.LONG	^X88888888		
		OBC1	2136				
29 30 20 53 43 46 28 37 33 36 00	0A	OBC1	2137	.ASCIC	/637(FCS 0)/		; FCS
29 31 20 53 43 46 28 37 33 36 00	0A	OBC1					
29 31 20 53 43 46 28 37 33 36 00	0A	OBC1	2138	.ASCIC	/637(FCS 1)/		; FCS
29 32 20 53 43 46 28 37 33 36 00	0A	OBC1					
29 32 20 53 43 46 28 37 33 36 00	0A	OBC1	2139	.ASCIC	/637(FCS 2)/		; FCS
29 33 20 53 43 46 28 37 33 36 00	0A	OBC1					
29 33 20 53 43 46 28 37 33 36 00	0A	OBC1	2140	.ASCIC	/637(FCS 3)/		; FCS
		OBE2					
	02	OBED	2141				
	0F0F0F0F	OBED	2142	FMR_NAME::	.BYTE	2	
	F0F0F0F0	OBEE	2143	.LONG	^X0F0F0F0F		
		0BF2	2144	.LONG	^XF0F0F0F0		
		0BF6	2145				
29 30 20 52 4D 46 28 39 33 36 00	0A	0BF6	2146	.ASCIC	/639(FMR 0)/		; FMR
29 31 20 52 4D 46 28 39 33 36 00	0A	0BF6					
29 31 20 52 4D 46 28 39 33 36 00	0A	0C01	2147	.ASCIC	/639(FMR 1)/		; FMR
		0C01					
	04	0C0C	2148				
	000000FF	0C0C	2149	FFL_NAME::	.BYTE	4	; FFA LOW HALF
	0000FF00	0C0D	2150	.LONG	^X000000FF		
	00FF0000	0C11	2151	.LONG	^X0000FF00		
	FF000000	0C15	2152	.LONG	^X00FF0000		
		0C19	2153	.LONG	^XFF000000		
		0C1D	2154				
29 30 20 41 46 46 28 38 33 36 00	0A	0C1D	2155	.ASCIC	/638(FFA 0)/		
29 31 20 41 46 46 28 38 33 36 00	0A	0C1D					
		0C28	2156	.ASCIC	/638(FFA 1)/		

```

29 32 20 41 46 46 28 38 33 36 00' 0A 0C28
0A 0C33 2157 .ASCIC /638(FFA 2)/
29 33 20 41 46 46 28 38 33 36 00' 0A 0C33
0A 0C3E 2158 .ASCIC /638(FFA 3)/
0A 0C3E
0C49 2159
04 0C49 2160 FFH_NAME:: .BYTE 4 ; FFA HIGH HALF
000000FF 0C4A 2161 .LONG ^X000000FF
0000FF00 0C4E 2162 .LONG ^X0000FF00
00FF0000 0C52 2163 .LONG ^X00FF0000
FF000000 0C56 2164 .LONG ^XFF0000C0
0C5A 2165
29 34 20 41 46 46 28 38 33 36 00' 0A 0C5A 2166 .ASCIC /638(FFA 4)/
0A 0C5A
29 35 20 41 46 46 28 38 33 36 00' 0A 0C65 2167 .ASCIC /638(FFA 5)/
0A 0C65
29 36 20 41 46 46 28 38 33 36 00' 0A 0C70 2168 .ASCIC /638(FFA 6)/
0A 0C70
29 37 20 41 46 46 28 38 33 36 00' 0A 0C7B 2169 .ASCIC /638(FFA 7)/
0A 0C7B
0C86 2170
0C86 2171 ;
0C86 2172 ; SINGLE PSEUDO INSTRUCTION PARAMETERS:
0C86 2173 ;
0C86 2174
20 3D 43 50 54 20 20 20 20 00' 0A 0C86 2175 TPC_MSG: .ASCIC / TPC= /
09 0C86
0C90 2176
0C90 2177 ;
0C90 2178 ; INSERT FIELD PARAMETERS:
0C90 2179 ;
0C90 2180
00000C9A 0C90 2181 INS_FLD_BUFF: .BLK 10 ; TEMPORARY BUFFER
00 0C9A 2182 CURRENT_BIT: .BYTE ; CONTAINS CURRENT BIT NUMBER
00 0C9B 2183 BIT_COUNT: .BYTE ; CONTAINS NUMBER OF BITS IN FIELD
00 0C9C 2184 SCRAMBLE_FLAG: .BYTE ; INHIBIT SCRAMBLE FLAG
0C9D 2185
0C9D 2186 ;
0C9D 2187 ; READ RTEMP PARAMETERS:
0C9D 2188 ;
00 0C9D 2189 READ_R_TEMP: .BYTE ; TEMP STORAGE
0C9E 2190
0C9E 2191 ;
0C9E 2192 ; THE FOLLOWING 80 BYTES CONTAIN THE BIT NUMBERS OF THE SCRAMBLED DCS.
0C9E 2193 ;
0C9E 2194 ;
0C9E 2195 ;
0C9E 2196 ;
01 00 45 1D 46 1E 1A 44 0C9E 2197 DCS_BIT_MAP: .BYTE 48,26,30,70,29,69,00,01 ; 0-7
23 19 07 06 05 04 03 02 0CA6 2198 .BYTE 02,03,04,05,06,07,25,35 ; 8-15
15 11 16 17 09 0B 3B 14 0CAE 2199 .BYTE 20,59,11,09,23,22,17,21 ; 16-23
0F 39 12 0D 0E 3A 10 49 0CB6 2200 .BYTE 73,16,58,14,13,18,57,15 ; 24-31
29 21 4C 31 4B 2D 36 38 0CBE 2201 .BYTE 56,54,45,75,49,76,33,41 ; 32-39
2B 2F 26 30 4E 4A 48 22 0CC6 2202 .BYTE 34,72,74,78,48,38,47,43 ; 40-47
27 3D 3E 28 37 40 35 3F 0CCE 2203 .BYTE 63,53,64,55,40,62,61,39 ; 48-55

```

ZZ-ECKAA-8.7 V08.07
ECKAA
V08.07

OWN STORAGE M 5
VAX-11/750 MICRO DIAGNOSTIC MONITOR 11-FEB-1986
OWN STORAGE

Fiche 1 Frame M5 Sequence 64
11-FEB-1986 10:07:52 VAX/VMS Macro V04-00 Page 39
11-FEB-1986 10:06:37 [VAX750.MONITOR]ECKAA.MAR;811 (6)

08	1B	1C	1F	0C	0A	3C	18	OCD6	2204	.BYTE	24,60,10,12,31,28,27,08	; 56-63
41	24	25	4F	43	34	2C	2A	OCDE	2205	.BYTE	42,44,52,67,79,37,36,65	; 64-71
13	2E	20	4D	32	42	33	47	OCE6	2206	.BYTE	71,51,66,50,77,32,46,19	; 72-79
								OCEE	2207			
								OCEE	2208	:		
								OCEE	2209	:	MISCELLANEOUS ASCII MESSAGES	
								OCEE	2210	:		
								OCEE	2211			
				0A	0D	02		OCEE	2212	CRLF:	.BYTE 2,13,10	; ASCII FOR CARRIAGE RET, LINE FEED
								OCF1	2213			

```

OCF1 2215      .SBTTL  'PROGRAM INITIALIZATION ROUTINE
OCF1 2216      ;++
OCF1 2217      ;
OCF1 2218      ; FUNCTIONAL DESCRIPTION:
OCF1 2219      ;
OCF1 2220      ;     THIS ROUTINE PERFORMS THE FOLLOWING FUNCTIONS:
OCF1 2221      ;
OCF1 2222      ;     0) STOP THE CLOCK
OCF1 2223      ;     1) INITIALIZE THE TEST PC AND THE STACK POINTER
OCF1 2224      ;     2) OPEN THE APPROPRIATE TEST STREAM FILE
OCF1 2225      ;     3) TYPE THE PROGRAM NAME
OCF1 2226      ;     4) IF THE 'LOSS' OR 'LOST' FLAG IS SET, CALL THE DIRECTORY ROUTINE
OCF1 2227      ;     5) READ A TEST OVERLAY INTO MEMORY
OCF1 2228      ;     6) CALL THE INTERPRETER TO START TESTING
OCF1 2229      ;     7) CHECK IS ALL SPECIFIED PASSES HAVE BEEN EXECUTED. I THEY HAVE,
OCF1 2230      ;     RETURN TO THE RDM SYSTEM VIA 'ABORT', OTHERWISE RESTART EXECUTION.
OCF1 2231      ;
OCF1 2232      ; CALLING SEQUENCE:
OCF1 2233      ;
OCF1 2234      ;     THIS ROUTINE IS ENTERED FROM THE RDM SOFTWARE OR THE COMMAND PARSER
OCF1 2235      ;     IF A 'DIAGNOSE' COMMAND IS EXECUTED.
OCF1 2236      ;
OCF1 2237      ; INPUT PARAMETERS:
OCF1 2238      ;
OCF1 2239      ;     NONE
OCF1 2240      ;
OCF1 2241      ; IMPLICIT INPUTS:
OCF1 2242      ;
OCF1 2243      ;     NONE
OCF1 2244      ;
OCF1 2245      ; OUTPUT PARAMETERS:
OCF1 2246      ;
OCF1 2247      ;     NONE
OCF1 2248      ;
OCF1 2249      ; IMPLICIT OUTPUTS:
OCF1 2250      ;
OCF1 2251      ;     FILE_NAM_PTR - CONTAINS THE ADDRESS OF THE TEST STREAM FILE NAME
OCF1 2252      ;     THAT IS ON THE CURRENTLY LOADED CARTRIDGE
OCF1 2253      ;
OCF1 2254      ; COMPLETION CODES:
OCF1 2255      ;
OCF1 2256      ;     NONE
OCF1 2257      ;
OCF1 2258      ; --
OCF1 2259      ;
OCF1 2260      ; PROGRAM_INIT:
OCF1 2261      ;
OCF1 2262      ;     SETUP INPUT REQUEST
OCF1 2263      ;
OCF1 2264      ;     $TERM_INIT      ; CLEAR OUT THE TERMINAL DRIVER
OCF6 2265      ;     $TERM_READ    TERM_INP_BUFF,-
OCF6 2266      ;     ONE_TERM_INTERRUPT
OD01 2267      ;     LHLD          SAVED_SP      ; GET INITIAL STACK POINTER
OD04 2268      ;     SPHL          ; INIT THE STACK POINTER
OD05 2269      ;     TYPE          CRLF          ; TYPE A CARRIAGE RETURN LINE FEED

```

```

OD0E 2270          TYPE          PROGRAM_TITLE ; TYPE THE TITLE AND VERSION
OD17 2271          XRA            A             ; CLEAR THE 'A' REG
OD18 2272          STA            PASS_COUNT   ; INIT THE PASS COUNT
OD1B 2273          ;
OD1B 2274          ; CHECK IF GOT HERE WITH A 'DI DM' COMMAND
OD1B 2275          ;
OD1B 2276          LDA            SOFT_CTRL_REGB ; GET THE RDM FLAG
OD1E 2277          MOV            B,A          ; SAVE LAST TIME RDM FLAG
OD1F 2278          ANI            RDM_FLAG     ;
OD21 2279          JZ             21$         ; BRANCH IF NOT SET
OD24 2280          MOV            A,B          ; SET RDM_LAST FLAG
OD25 2281          ORI            RDM_LAST    ; ...
OD27 2282          STA            SOFT_CTRL_REGB ;
OD2A 2283          MVI            A,RDM_INDEX ; SET FILE NAME INDEX TO RDM FILE NAME
OD2C 2284          JMP            22$         ; GO CHECK IF ON TAPE
OD2F 2285          ;
OD2F 2286          ; NOT 'DI DM' . CHECK IF LAST EXECUTION WAS A 'DI DM'
OD2F 2287          ;
OD2F 2288          21$: MOV            A,B          ; GET RDM_LAST FLAG
OD30 2289          ANI            RDM_LAST    ; WAS LAST EXECUTION RDM TESTS?
OD32 2290          JZ             23$         ; BRANCH IF NO
OD35 2291          ;
OD35 2292          ; LAST EXECUTION WAS "DI DM" SO SEARCH DIRECTORY FOR DEFAULT FILE
OD35 2293          ;
OD35 2294          MOV            A,B          ; CLEAR RDM_LAST FLAG
OD36 2295          ANI            <^CRDM_LAST> ; ...
OD38 2296          STA            SOFT_CTRL_REGB ;
OD3B 2297          JMP            20$         ; FIND OUT WHICH TAPE THIS IS
OD3E 2298          ;
OD3E 2299          ; LAST EXECUTION NOT 'DI DM' . CHECK IF PREVIOUS EXECUTION
OD3E 2300          ;
OD3E 2301          23$: LHLD         STARTING_RECORD ; INIT THE CURRENT RECORD NUMBER
OD41 2302          SHLD         CURRENT_REC_NMB ;
OD44 2303          XRA            A             ; INIT A REG
OD45 2304          ORA            H             ; SEE IF A FILE HAS BEEN OPENED
OD46 2305          JP             6$          ; BRANCH IF IT HAS
OD49 2306          ;
OD49 2307          ; SEARCH THE DIRECTORY FOR THE DEFAULT FILE
OD49 2308          ;
OD49 2309          20$: XRA            A             ; INIT THE FILE NAME INDEX
OD4A 2310          22$: STA            FILE_NAM_INDEX ; ...
OD4D 2311          ;
OD4D 2312          ; NOW FIGURE OUT WHICH TAPE THIS IS BY TRYING TO OPEN THE TEST STREAM FILES.
OD4D 2313          ;
OD4D 2314          1$: LXI            H,FILE_FOR_DPM ; GET ADDRESS OF DPM FILE NAME
OD50 2315          MVI            D,0         ; SETUP A CONSTANT 6 TO ADD TO
OD52 2316          MVI            E,6         ; H & L REG'S
OD54 2317          LDA            FILE_NAM_INDEX ; GET THE FILE NAME INDEX
OD57 2318          2$: DCR            A             ; DECREMENT
OD58 2319          JM             3$         ; BRANCH IF DONE INDEXING
OD5B 2320          DAD            D             ; INDEX FILE NAME ADDRESS
OD5C 2321          JMP            2$         ; CONTINUE
OD5F 2322          ;
OD5F 2323          ; POINTER TO RAD50 FILE NAME IS IN H & L
OD5F 2324          ;

```

```
OD5F 2325 3$:     SHLD                    FILE_NAM_PTR     ; SAVE FOR 'OPEN FILE' ROUTINE
OD62 2326     ;
OD62 2327     ; TRY AND OPEN THE FILE
OD62 2328     ;
OD62 2329     CALL                    OPEN_TU58        ; TRY AND OPEN THE FILE
OD65 2330     JC                        7$                ; BRANCH IF ERROR
OD68 2331     LHLD                    CURRENT_REC_NMB ; GET THE CURRENT RECORD NUMBER
OD68 2332     SHLD                    STARTING_REC_NMB ; INIT THE STARTING RECORD NUMBER
OD6E 2333     JMP                      6$                ; CONTINUE
OD71 2334     ;
OD71 2335     ; FILE IS NOT ON TAPE. SEE IF WE'VE TRIED ALL THE FILE NAMES
OD71 2336     ;
OD71 2337     ;
OD71 2338 7$:     LDA                    FILE_NAM_INDEX   ; GET THE CURRENT INDEX
OD74 2339     INR                      A                ; INCREMENT
OD75 2340     STA                      FILE_NAM_INDEX   ; SAVE
OD78 2341     CPI                      NUMBER_OF_FILES ; TRIED ALL FILE NAMES?
OD7A 2342     JNZ                      1$                ; BRANCH IF NO (TRY NEXT NAME)
OD7D 2343     CALL                    TYPE_TU58_ERROR ; TYPE AN ERROR MESSAGE
OD80 2344 5$:     CALL                    GET_CMD_LINE     ; GO TO THE COMMAND PARSER
OD83 2345     JMP                      5$                ; DON'T ALLOW CONTINUE
OD86 2346     ;
OD86 2347     ; FOUND A FILE. FIND THE RECORD NUMBER OF THE SPECIFIED TEST
OD86 2348     ;
OD86 2349 6$:     CALL                    DIRECT_SEARCH    ; GET THE RECORD NUMBER OF THE
OD89 2350                                                    ; SPECIFIED TEST
OD89 2351     JNC                      8$                ; BRANCH IF NO ERROR
OD8C 2352     ;
OD8C 2353     ; SEE IF ERROR CODE IS -1
OD8C 2354     ;
OD8C 2355     CPI                      -1                ; IF -1, GO BACK AND SCAN THE
OD8E 2356                                                    ; DIRECTORY AGAIN
OD8E 2357     JZ                        20$              ; ...
OD91 2358     ;
OD91 2359     ; TEST SELECTED IS NOT IN THIS FILE. TYPE AN ERROR MESSAGE
OD91 2360     ;
OD91 2361     TYPE                    DIRECT_MSG       ; TYPE THE ERROR MESSAGE
OD9A 2362     TYPE                    CRLF             ;
ODA3 2363     JMP                      5$                ; CALL THE COMMAND PARSER
ODA6 2364     ;
ODA6 2365     ; LOCATION CURRENT_REC_NMB CONTAINS THE RECORD NUMBER FOR THE START OF
ODA6 2366     ; THE OVERLAY. NOW LOAD THE OVERLAY INTO MEMORY.
ODA6 2367     ;
ODA6 2368 8$:     CALL                    READ_OVERLAY     ; READ THE OVERLAY INTO MEMORY
ODA9 2369     LHLD                    INIT_TEST_PC     ; GET THE INITIAL TEST PC
ODAC 2370     SHLD                    TEST_PC          ; INIT THE TEST PC
ODAF 2371     ;
ODAF 2372     ; FIRST OVERLAY IS LOADED, START THE TEST.
ODAF 2373     ;
ODAF 2374     CALL                    OP_CODE_DISP     ; START TESTING
ODB2 2375     ;
ODB2 2376     ; RETURN HERE FROM END OF PASS ROUTINE
ODB2 2377     ;
ODB2 2378     TYPE                    CRLF             ; TYPE THE END OF PASS MESSAGE
ODBB 2379     TYPE                    ENDPASS_MSG     ; ...
```

ODC4	2380	TYPEB	PASS_COUNT	:	TYPE THE PASS COUNT
ODCF	2381	TYPE	CRLF	:	
ODD8	2382	LDA	PASS_COUNT	:	SEE IF WE HAVE EXECUTED
ODDB	2383	MOV	B,A	:	THE REQUIRED NUMBER OF PASSES
ODDC	2384	LDA	USER_PASS_CNT	:	...
ODDF	2385	CMP	B	:	
ODE0	2386	JZ	9\$:	BRANCH IF NO MORE PASSES
ODE3	2387	10\$: XRA	A	:	INITIALZE THE PARAMETERS FOR
ODE4	2388	STA	TESTS_PER_LINE	:	TYPING THE TEST NUMBERS
ODE7	2389	STA	TYPE_TEST_FLAG	:	
ODEA	2390	STA	TEST_NUMBER	:	INIT THE INITIAL TEST NUMBER
ODED	2391	CALL	OPEN_TU58	:	RE-OPEN THE TEST FILE
ODF0	2392	JNC	6\$:	BRANCH IF NO ERRORS TO START
ODF3	2393			:	ANOTHER PASS
ODF3	2394	:		:	
ODF3	2395	:		:	ERROR WHEN REOPENING THE FILE. TYPE THE MESSAGE AND CALL COMMAND PARSER
ODF3	2396	:		:	
ODF3	2397	CALL	TYPE_TU58_ERROR	:	TYPE AN ERROR MESSAGE
ODF6	2398	9\$: CALL	GET_CMD_LINE	:	CALL THE COMMAND PARSER
ODF9	2399	JMP	9\$:	DON'T ALLOW CONTINUE
ODFC	2400				

```
ODFC 2402      .SBTTL 'COMMON ROUTINES
ODFC 2403      :
ODFC 2404      : THE FOLLOWING ROUTINES ARE COMMON TO BOTH THE COMMAND PARSER AND
ODFC 2405      : THE INTERPRETER.
ODFC 2406      :
ODFC 2407      :
ODFC 2408      :
ODFC 2409      .SBTTL " READ TUS8
ODFC 2410      ;++
ODFC 2411      :
ODFC 2412      : FUNCTIONAL DESCRIPTION:
ODFC 2413      :
ODFC 2414      :     THIS ROUTINE IS USED TO READ RECORDS FROM THE TUS8 SERIAL LINE.
ODFC 2415      :
ODFC 2416      : CALLING SEQUENCE:
ODFC 2417      :
ODFC 2418      :     THIS ROUTINE IS CALLED BY THE FOLLOWING MACRO:
ODFC 2419      :
ODFC 2420      :         READ_RECORD [<BUFFER ADDRESS>],[<NUMBER OF RECORDS>]
ODFC 2421      :
ODFC 2422      :     THIS ROUTINE IS CALLED BY THE FOLLOWING ROUTINES:
ODFC 2423      :
ODFC 2424      :         READ_OVERLAY
ODFC 2425      :         DIRECT_SEARCH
ODFC 2426      :
ODFC 2427      :     THIS ROUTINE CALLS THE FOLLOWING ROUTINES:
ODFC 2428      :
ODFC 2429      :         RDM TUS8 DRIVER
ODFC 2430      :         TYPE_TUS8_ERROR
ODFC 2431      :         GET_CMD_LINE
ODFC 2432      :
ODFC 2433      : INPUT PARAMETERS:
ODFC 2434      :
ODFC 2435      :     REGISTER PAIR H & L - CONTAINS THE ADDRESS OF A BUFFER TO READ THE
ODFC 2436      :     DATA INTO.
ODFC 2437      :     REGISTER A           - CONTAINS THE NUMBER FO RECORDS TO READ.
ODFC 2438      :
ODFC 2439      : IMPLICIT INPUTS:
ODFC 2440      :
ODFC 2441      :     CURRENT_REC_NMB     - CONTAINS THE CURRENT POSITION OF THE TAPE
ODFC 2442      :
ODFC 2443      : OUPUT PARAMETERS:
ODFC 2444      :
ODFC 2445      :     REGISTER PAIR H & L - CONTAINS THE ADDRESS OF THE READ BUFFER.
ODFC 2446      :
ODFC 2447      : IMPLICIT OUTPUTS:
ODFC 2448      :
ODFC 2449      :     DATA MOVED TO THE BUFFER
ODFC 2450      :     CURRENT_REC_NMB     - CONTAINS THE CURRENT POSITION OF THE TAPE
ODFC 2451      :     TUS8_ERR_CODE      - GETS THE ERROR CODE IF AN ERROR OCCURS
ODFC 2452      :
ODFC 2453      : SIDE EFFECTS:
ODFC 2454      :
ODFC 2455      :     IF AN ERROR OCCURS ON THE READ, AN ERROR MESSAGE IS TYPED FOLLOWED
ODFC 2456      :     BY THE ERROR CODE, AND EXECUTION IS TRANSFERED TO THE COMMAND
```

```
0DFC 2457 ;      PARSER.  
0DFC 2458 ;  
0DFC 2459 ;--  
0DFC 2460  
0DFC 2461  
0DFC 2462 READ_TUS8-  
0DFC 2463      SHLD      TUS8_BUFF_ADDR ; SAVE THE BUFFER ADDRESS  
0DFF 2464      STA      TUS8_REC_COUNT ; AND THE RECORD COUNT  
0E02 2465      LHL      CURRENT_REC_NMB ; GET THE CURRENT RECORD NUMBER  
0E05 2466      SHLD     TUS8_REC_NUMB ; SAVE  
0E08 2467      MOV      C,A           ; UPDATE CURRENT RECORD NUMBER  
0E09 2468      MVI      B,0           ; ...  
0E0B 2469      DAD      B             ; ...  
0E0C 2470      SHLD     CURRENT_REC_NMB ; ...  
0E0F 2471      $TUS8_READ TUS8_BUFF_ADDR,TUS8_REC_NUMB,TUS8_REC_COUNT  
0E1C 2472      JNC      1$           ; EXIT IF NO ERROR  
0E1F 2473      STA      TUS8_ERR_CODE ; SAVE THE ERROR CODE  
0E22 2474      CALL     TYPE_TUS8_ERROR ; TYPE THE ERROR MESSAGE AND CODE  
0E25 2475 2$:    CALL     GET_CMD_LINE ; GO TO THE COMMAND PARSER  
0E28 2476      JMP      2$           ; DON'T ALLOW CONTINUE COMMAND  
0E2B 2477 1$:    LHL      TUS8_BUFF_ADDR ; RESTORE THE BUFFER ADDRESS TO H & L  
0E2E 2478      RET  
0E2F 2479
```

```

0E2F 2481          .SBTTL " OPEN FILE ON TUS8
0E2F 2482          ;++
0E2F 2483          ; FUNCTIONAL DESCRIPTION
0E2F 2484          ;
0E2F 2485          ;     THIS ROUTINE TRIES TO OPEN A FILE ON THE TUS8.
0E2F 2486          ;
0E2F 2487          ; CALLING SEQUENCE
0E2F 2488          ;
0E2F 2489          ;     CALL     OPEN_TUS8
0E2F 2490          ;
0E2F 2491          ;     THIS ROUTINE IS CALLED BY THE 'PROGRAM_INIT' ROUTINE
0E2F 2492          ;
0E2F 2493          ;     THIS ROUTINE CALLS THE FOLLOWING ROUTINES:
0E2F 2494          ;
0E2F 2495          ;         TPOpen (IN RDM SYSTEM SOFTWARE)
0E2F 2496          ;         TYPE_TUS8_ERROR (IF DEVICE ERROR OCCURS)
0E2F 2497          ;         GET_CMD_LINE (IF DEVICE ERROR OCCURS)
0E2F 2498          ;
0E2F 2499          ; IMPLICIT INPUTS
0E2F 2500          ;
0E2F 2501          ;     FILE_NAM_PTR - CONTAINS ADDRESS OF RADIX 50 FILE NAME
0E2F 2502          ;
0E2F 2503          ; IMPLICIT OUTPUTS
0E2F 2504          ;
0E2F 2505          ;     CURRENT_REC_NMB - RECEIVES THE STARTING RECORD NUMBER OF THE FILE
0E2F 2506          ;     TUS8_ERR_CODE  - RECEIVES AN ERROR CODE IF OPEN FAILED
0E2F 2507          ;     TOTAL_BLOCKS   - RECEIVES THE LENGTH OF THE FILE IN BLOCKS
0E2F 2508          ;
0E2F 2509          ; COMPLETION CODES:
0E2F 2510          ;
0E2F 2511          ;     C BIT SET   - FILE WAS NOT FOUND.
0E2F 2512          ;     C BIT CLEAR - NO ERROR
0E2F 2513          ;
0E2F 2514          ; --
0E2F 2515          ;
0E2F 2516          ;
0E2F 2517          OPEN_TUS8:
0E2F 2518          $TUS8_OPEN      FILE_NAM_PTR,CURRENT_REC_NMB,TOTAL_BLOCKS
0E3A 2519          JNC              1$                    ; BRANCH IF NO ERROR
0E3D 2520          STA              TUS8_ERR_CODE        ; SAVE THE ERROR CODE
0E40 2521          CPI              $FILE_NOT_FOUND     ; WAS IT A FILE NOT FOUND?
0E42 2522          JZ               3$                    ; BRANCH IF YES
0E45 2523          CALL            TYPE_TUS8_ERROR     ; TYPE AN ERROR MESSAGE
0E48 2524 2$:      CALL            GET_CMD_LINE        ; CALL THE COMMAND PARSER
0E4B 2525          JMP              2$                    ; DON'T ALLOW CONTINUE
0E4E 2526 3$:      STC                                 ; FILE NOT FOUND
0E4F 2527 1$:      RET                                ; RETURN TO CALLER
0E50 2528

```

```
0E50 2530 .SBTTL TYPE TUS8 ERROR MESSAGE
0E50 2531 ;**
0E50 2532 ;
0E50 2533 ; FUNCTIONAL DESCRIPTION:
0E50 2534 ;
0E50 2535 ; THIS ROUTINE IS USED TO TYPE AN ERROR MESSAGE IF AN ERROR OCCURS
0E50 2536 ; ON AN 'OPEN' OR 'READ' REQUEST FROM THE TUS8.
0E50 2537 ;
0E50 2538 ; CALLING SEQUENCE:
0E50 2539 ;
0E50 2540 ; CALL TYPE_TUS8_ERROR
0E50 2541 ;
0E50 2542 ; THIS ROUTINE IS CALLED BY THE FOLLOWING ROUTINES:
0E50 2543 ;
0E50 2544 ; OPEN_TUS8
0E50 2545 ; READ_TUS8
0E50 2546 ;
0E50 2547 ; THIS ROUTINE CALLS THE FOLLOWING ROUTINES:
0E50 2548 ;
0E50 2549 ; TYPE
0E50 2550 ; TYPEB
0E50 2551 ;
0E50 2552 ; IMPLICIT INPUTS:
0E50 2553 ;
0E50 2554 ; TUS8_ERR_CODE - CONTAINS THE ERROR CODE
0E50 2555 ;
0E50 2556 ; IMPLICIT OUTPUTS:
0E50 2557 ;
0E50 2558 ; NONE
0E50 2559 ;
0E50 2560 ; SIDE EFFECTS:
0E50 2561 ;
0E50 2562 ; NONE
0E50 2563 ;
0E50 2564 ;--
0E50 2565
0E50 2566
0E50 2567 TYPE_TUS8_ERROR:
0E50 2568 TYPE CRLF ; TYPE A CARRIAGE RETURN LINE FEED
0E59 2569 TYPE TUS8_ERROR ; TYPE THE ERROR MESSAGE
0E62 2570 TYPEB TUS8_ERR_CODE ; TYPE THE ERROR CODE
0E6D 2571 TYPE TEST_MSG ; TYPE 'TEST:'
0E76 2572 TYPEB TEST_NUMBER ; TYPE THE TEST NUMBER
0E81 2573 RET ; RETURN TO CALLER
0E82 2574
```

```
0E82 2576 .SBTTL TYPE ASCII ROUTINE
0E82 2577 ;**
0E82 2578 ;
0E82 2579 ; FUNCTIONAL DESCRIPTION
0E82 2580 ;
0E82 2581 ; THIS ROUTINE TYPES AN ASCII STRING ON THE CONSOLE TERMINAL.
0E82 2582 ; THE STRING MUST HAVE ITS CHARACTER COUNT IN THE FIRST BYTE.
0E82 2583 ; AN ALTERNATE ENTRY POINT (TYPE_ASCII_U) ALLOWS UNCOUNTED STRINGS
0E82 2584 ; TO BE TYPED.
0E82 2585 ;
0E82 2586 ; CALLING SEQUENCE:
0E82 2587 ;
0E82 2588 ; THE ROUTINE IS CALLED WITH THE FOLLOWING MACRO:
0E82 2589 ;
0E82 2590 ; TYPE MESSAGE_ADDRESS
0E82 2591 ; TYPE_UNCOUNTED CHARACTER_COUNT
0E82 2592 ;
0E82 2593 ; THE ROUTINE IS CALLED BY THE FOLLOWING ROUTINES:
0E82 2594 ;
0E82 2595 ; PROGRAM_INIT
0E82 2596 ; TYPE_TU58_ERROR
0E82 2597 ; TYPE_NUMERIC
0E82 2598 ; TYPE_TEST_NUMB
0E82 2599 ; GET_CMD_LINE
0E82 2600 ; SHOW_ACTION
0E82 2601 ; BURST_CLOCK
0E82 2602 ; RING_BELL
0E82 2603 ; TYPE_ERROR
0E82 2604 ; SINGLE_INSTR
0E82 2605 ;
0E82 2606 ; THE ROUTINE CALLS THE FOLLOWING ROUTINES:
0E82 2607 ;
0E82 2608 ; TMWRIT (IN THE RDM SYSTEM SOFTWARE)
0E82 2609 ;
0E82 2610 ; IMPLICIT INPUTS:
0E82 2611 ;
0E82 2612 ; MESSAGE_POINTER - CONTAINS THE ADDRESS OF THE ASCII STRING.
0E82 2613 ;
0E82 2614 ; IMPLICIT OUTPUTS:
0E82 2615 ;
0E82 2616 ; NONE
0E82 2617 ;
0E82 2618 ; SIDE EFFECTS:
0E82 2619 ;
0E82 2620 ; IF AN ERROR OCCURS ON THE TERMINAL WRITE, EXECUTION OF THE
0E82 2621 ; DIAGNOSTIC IS ABORTED.
0E82 2622 ;
0E82 2623 ;--
0E82 2624 ;
0E82 2625 ;
0E82 2626 TYPE_ASCII:
0E82 2627 LHLD MESSAGE_POINTER ; GET THE ADDRESS OF THE MESSAGE
0E85 2628 MOV A,M ; GET THE STRING LENGTH
0E86 2629 STA MESSAGE_LENGTH ; PUT IN IMPLICIT STORAGE
0E89 2630 INX H ; POINT PAST THE CHAR COUNT
```

ZZ-ECKAA-8.7 V08.07
ECKAA
V08.07

J 6
TYPE ASCII ROUTINE
VAX-11/750 MICRO DIAGNOSTIC MONITOR

Fiche 1 Frame J6 Sequence 74
11-FEB-1986 10:07:52 VAX/VMS Macro V04-00 Page 49
11-FEB-1986 10:06:37 [VAX750.MONITOR]ECKAA.MAR;811 (11)

0E8A 2631 SHLD
0E8D 2632 TYPE_ASCII_U:
0E8D 2633 \$TERM_WRITE
0E8D 2634
0E98 2635 JNC
0E9B 2636 ABORT
0E9E 2637 1\$: RET
0E9F 2638

MESSAGE_POINTER ; ...
MESSAGE_POINTER,-
MESSAGE_LENGTH ; WRITE THE STRING
1\$; EXIT IF NO ERROR
; ABORT, FATAL ERROR
; RETURN TO THE CALLER

0E9F	2695	PUSH	\$PSW	:	SAVE DATA TYPE
0EA0	2696	ANI	<^X7F>	:	DISCARD BIT 7
0EA2	2697	STA	TYPEN_DATA_TYPE	:	SAVE THE DATA TYPE
0EA5	2698	RLC		:	GENERATE ASCII BYTE COUNT
0EA6	2699	STA	TEMP_BUFFER	:	PUT IN CONVERT BUFFER
0EA9	2700	\$CONVERT	TYPEN_DATA_PTR,-	:	CONVERT THE NUMERIC TO AN
0EA9	2701		TEMP_BUFFER+1,-	:	AN ASCII STRING
0EA9	2702		TYPEN_DATA_TYPE	:	...
0EB4	2703	POP	\$PSW	:	GET THE DATA TYPE BACK
0EB5	2704	ANI	<^X80>	:	NIBBLE DATA TYPE?
0EB7	2705	JZ	1\$:	BRANCH IF NO
0EBA	2706	MVI	A,1	:	SET THE ASCII BYTE COUNT
0EBC	2707	STA	TEMP_BUFFER+1	:	TO ONE DIGIT
0EBF	2708	TYPE	TEMP_BUFFER+1	:	TYPE THE SINGLE DIGIT
0EC8	2709	JMP	2\$:	
0ECB	2710	1\$:	TEMP_BUFFER	:	TYPE THE STRING
0ED4	2711	2\$:		:	RETURN TO CALLER
0ED5	2712	RET		:	

```

OED5 2714 .SBTTL . TERMINAL INTERRUPT SERVICE ROUTINE
OED5 2715 ;**
OED5 2716 ;
OED5 2717 ; FUNCTIONAL DESCRIPTION:
OED5 2718 ;
OED5 2719 ; THIS ROUTINE IS INVOKED BY AN INTERRUPT FROM THE TERMINAL.
OED5 2720 ; IF THE CHARACTER TYPED WAS A CONTROL C, THE CTRL C BIT IN THE
OED5 2721 ; 'SWR' IS SET.
OED5 2722 ;
OED5 2723 ; THE ROUTINE INITIATES ANOTHER TERMINAL REQUEST, AND EXITS.
OED5 2724 ;
OED5 2725 ; INPUT PARAMETERS:
OED5 2726 ;
OED5 2727 ; C BIT SET - INDICATES ERROR. ERROR CODE IS IN THE A REGISTER.
OED5 2728 ;
OED5 2729 ; IMPLICIT INPUTS:
OED5 2730 ;
OED5 2731 ; NONE
OED5 2732 ;
OED5 2733 ; IMPLICIT OUTPUTS:
OED5 2734 ;
OED5 2735 ; SOFT_CTRL_REG - CONTROL_C_FLAG SET IF CHARACTER WAS A CONTROL C.
OED5 2736 ;
OED5 2737 ; SIDE EFFECTS:
OED5 2738 ;
OED5 2739 ; IF AN ERROR IS DETECTED, THE ABORT ROUTINE IS CALLED.
OED5 2740 ; IF NO ERROR, ANOTHER INPUT REQUEST IS QUEUED.
OED5 2741 ;
OED5 2742 ;--
OED5 2743
OED5 2744 TERM_INTERRUPT:
OED5 2745 JNC 10$
OED8 2746 CPI $CONTROL_C_CODE ; EXIT IF NO ERROR
OEDA 2747 JZ 2$ ; WAS IT A CONTROL C?
OEDD 2748 ABORT ; BRANCH IF YES
OEE0 2749 2$: LDA SOFT_CTRL_REG ; ABORT THE DIAGNOSTIC
OEE3 2750 ORI CONTROL_C_FLAG ; GET THE SWITCH REGISTER
OEE5 2751 STA SOFT_CTRL_REG ; SET THE CONTROL C BIT
OEE8 2752 10$: $TERM_READ ; QUE ANOTHER INPUT REQUEST
OEE8 2753 ONE,TERM_INTERRUPT ;
OEF3 2754 RET ; RETURN TO CALLING SEQUENCE
OEF4 2755

```

```
0EF4 2757      .SBTTL  WRITE DCS ROUTINE
0EF4 2758      :++
0EF4 2759      :
0EF4 2760      : FUNCTIONAL DESCRIPTION:
0EF4 2761      :
0EF4 2762      :     THIS ROUTINE LOADS THE DIAGNOSTIC CONTROL STORE (DCS). IT IS
0EF4 2763      :     ENTERED WITH THE STARTING ADDRESS OF THE DCS, THE MICRO WORD
0EF4 2764      :     COUNT, AND A POINTER TO THE DATA TO LOAD. IT LOADS THE SPECIFIED
0EF4 2765      :     NUMBER OF MICRO WORDS INTO DCS AT THE STARTING ADDRESS AND
0EF4 2766      :     RETURNS. THE CURRENT VALUE OF THE DCS ADDRESS REGISTER IS SAVED AND
0EF4 2767      :     RESTORED.
0EF4 2768      :
0EF4 2769      : CALLING SEQUENCE:
0EF4 2770      :
0EF4 2771      :     THIS ROUTINE IS INVOKED WITH THE FOLLOWING MACRO:
0EF4 2772      :
0EF4 2773      :         WRITE$_DCS      DCS_ADDRESS,DATA_ADDRESS,WORD_COUNT
0EF4 2774      :
0EF4 2775      :     THE ROUTINE IS CALLED BY THE FOLLOWING ROUTINES:
0EF4 2776      :
0EF4 2777      :         LOAD_DCS
0EF4 2778      :         READ_OVERLAY
0EF4 2779      :         DIABLE_CMI
0EF4 2780      :
0EF4 2781      : IMPLICIT INPUTS:
0EF4 2782      :
0EF4 2783      :     DCS_WRITE_ADR  - CONTAINS ADDRESS TO START LOADING AT
0EF4 2784      :     DCS_WRITE_WDCNT - CONTAINS THE NUMBER OF MICRO WORDS TO LOAD
0EF4 2785      :     DCS_WRITE_PTR  - CONTAINS ADDRESS OF THE DATA TO LOAD
0EF4 2786      :     DCS_CTRL_RE_CPY - CONTAINS THE CURRENT VALUE OF THE DCS CONTROL REGISTER
0EF4 2787      :
0EF4 2788      : IMPLICIT OUTPUTS:
0EF4 2789      :
0EF4 2790      :     NONE
0EF4 2791      :
0EF4 2792      : SIDE EFFECTS:
0EF4 2793      :
0EF4 2794      :     NONE
0EF4 2795      :
0EF4 2796      : --
0EF4 2797      :
0EF4 2798      WRITE_DCS:
0EF4 2799      LDA      DCS_ADDR_REG_RO ; READ AND SAVE THE
0EF7 2800      PUSH     $PSW          ; CONTENTS OF THE DCS ADR REG
0EF8 2801      CALL     DESELECT_DCS  ; DESELECT DCS
0EFB 2802      LDA      DCS_WRITE_ADR  ; GET ADDRESS TO START WRITE
0EFE 2803      STA      DCS_ADDR_REG_WO ; SET THE DCS ADDR REG
0F01 2804      LDA      DCS_WRITE_WDCNT ; GET THE # OF MICRO WRDS TO WRITE
0F04 2805      MOV      B,A            ; SAVE IN REG B
0F05 2806      LHL     DCS_WRITE_PTR  ; GET THE ADDRESS OF THE DATA
0F08 2807      LXI     D,DCS_DATA_REG ; GET ADDRESS OF DCS DATA REG
0F0B 2808      :
0F0B 2809      : NOW START LOADING THE DCS. REGISTERS H & L HAVE ADDRESS OF DATA.
0F0B 2810      : REGISTERS D & E HAVE ADDRESS OF DCS DATA REG.
0F0B 2811      : REGISTER B CONTAINS NO. OF MICRO WORDS TO LOAD.
```



```

OF30 2853      .SBTTL      SINGLE MICRO INSTRUCTION ROUTINE
OF30 2854      ;++
OF30 2855      ;
OF30 2856      ; FUNCTIONAL DESCRIPTION:
OF30 2857      ;
OF30 2858      ;     THIS ROUTINE TICKS THE CPU CLOCK IN 'SINGLE MICRO INSTRUCTION'
OF30 2859      ;     MODE.
OF30 2860      ;
OF30 2861      ; CALLING SEQUENCE:
OF30 2862      ;
OF30 2863      ;     THIS ROUTINE IS CALLED WITH THE FOLLOWING MACRO:
OF30 2864      ;
OF30 2865      ;           SGL_MIC_INSTR      CYCLE_COUNT
OF30 2866      ;
OF30 2867      ;     THE ROUTINE IS CALLED BY THE FOLLOWING ROUTINES:
OF30 2868      ;
OF30 2869      ;           FETCH_MIC_INSTR
OF30 2870      ;           BURST_CLOCK
OF30 2871      ;
OF30 2872      ; IMPLICIT INPUTS:
OF30 2873      ;
OF30 2874      ;     MIC_INSTR_CNT      - CONTAINS THE NUMBER OF MICRO CYCLES TO TICK THE CLK.
OF30 2875      ;     DCS_CTRL_RE_CPY - CONTAINS THE CURRENT VALUE OF THE DCS CONTROL REGISTER
OF30 2876      ;
OF30 2877      ; IMPLICIT OUTPUTS:
OF30 2878      ;
OF30 2879      ;     NONE
OF30 2880      ;
OF30 2881      ;--
OF30 2882      ;
OF30 2883      SINGLE_MIC_INST:
OF30 2884      ;
OF30 2885      ; CLEAR MASTER HALT ENABLE
OF30 2886      ;
OF30 2887      ;     LDA      FRONT_PNL_2      ; GET THE STATE OF THE LIGHTS
OF33 2888      ;     ANI      <TEST!FAULT!REMOTE> ; MASK
OF35 2889      ;     STA      RD_CTRL_REG      ; CLEAR MASTER HALT ENABLE
OF38 2890      ;
OF38 2891      ;     LDA      MIC_INSTR_CNT      ; GET THE COUNT
OF3B 2892      ;     MOV      B,A      ; SAVE IN 'B' REGISTER
OF3C 2893      ;     LXI      H,DCS_CTRL_REG      ; GET ADDRESS OF CONTROL REG
OF3F 2894      ;
OF3F 2895      1$: LDA      DCS_CTRL_RE_CPY ; GET CONTROL REG COPY
OF42 2896      ;     ANI      <^CLK_CTRL_1> ; ACTIVATE CONTROL BIT ONE
OF44 2897      ;     MOV      M,A      ; WRITE THE CONTROL REGISTER
OF45 2898      ;     XRI      CLK_CTRL_1      ; INACTIVATE CONTROL BIT 1
OF47 2899      ;     MOV      M,A      ; WRITE THE CONTROL REGISTER
OF48 2900      ;     DCR      B      ; DECREMENT THE COUNT
OF49 2901      ;     JNZ      1$      ; CONTINUE UNTIL 0
OF4C 2902      ;     LDA      FRONT_PNL_2      ; GET STATE OF THE LIGHTS
OF4F 2903      ;     ANI      <TEST!FAULT!REMOTE> ; ...
OF51 2904      ;     ORI      <MASTER_HALT_EN!- ; SET MASTER HALT ENABLE AND TRAP
OF51 2905      ;     TRAP_HALT_EN> ; HALT ENABLE
OF53 2906      ;     STA      RD_CTRL_REG      ; ...
OF56 2907      ;     RET      ; RETURN TO CALLING SEQUENCE
  
```

ZZ-ECKAA-8.7 V08.07
ECKAA
V09.07

D 7

SINGLE MICRO	11-FEB-1986	Fiche 1	Frame D7	Sequence 81	
VAX-11/750 MICRO DIAGNOSTIC MONITOR	11-FEB-1986 10:07:52	VAX/VMS Macro	V04-00	Page	56
" SINGLE MICRO INSTRUCTION ROUTINE	11-FEB-1986 10:06:37	[VAX750.MONITOR]	ECKAA.MAR;811		(15)

0F57 2908

```
0F57 2910      .SBTTL      SINGLE CLOCK TICK ROUTINE
0F57 2911      ;++
0F57 2912      ;
0F57 2913      ; FUNCTIONAL DESCRIPTION:
0F57 2914      ;
0F57 2915      ;     THIS ROUTINE STEPS THE CPU CLOCK THE SPECIFIED NUMBER OF TIMES.
0F57 2916      ;
0F57 2917      ; CALLING SEQUENCE:
0F57 2918      ;
0F57 2919      ;     THIS ROUTINE IS CALLED BY THE FOLLOWING MACRO:
0F57 2920      ;
0F57 2921      ;           SGL_TICK      TICKS
0F57 2922      ;
0F57 2923      ;     THE ROUTINE IS CALLED BY THE 'BURST_CLOCK' ROUTINE.
0F57 2924      ;
0F57 2925      ; IMPLICIT INPUTS:
0F57 2926      ;
0F57 2927      ;     MIC_TICK_CNT      - CONTAINS THE NUMBER OF TICKS TO STEP THE CLOCK.
0F57 2928      ;     DCS_CTRL_RE_CPY - CONTAINS THE CURRENT VALUE OF THE DCS CONTROL REGISTER
0F57 2929      ;
0F57 2930      ; IMPLICIT OUTPUTS:
0F57 2931      ;
0F57 2932      ;     NONE
0F57 2933      ;
0F57 2934      ;--
0F57 2935
0F57 2936 SINGLE_TICK:
0F57 2937      LDA          FRONT_PNL_2      ; GET STATE OF LIGHTS
0F5A 2938      ANI          <TEST!FAULT!REMOTE> ; ...
0F5C 2939      STA          RD_CTRL_REG      ; CLEAR MASTER HALT ENABLE
0F5F 2940      LDA          MIC_TICK_CNT      ; GET THE COUNT
0F62 2941      MOV          B,A              ; SAVE IN B REGISTER
0F63 2942      LXI          H,DCS_CTRL_REG   ; GET ADDRESS OF CONTROL REG
0F66 2943
0F66 2944 1$:      LDA          DCS_CTRL_RE_CPY ; READ THE CONTROL REGISTER COPY
0F69 2945      ANI          <^CLK_CTRL_0>      ; ACTIVATE CONTROL BIT 0
0F6B 2946      MOV          M,A              ; WRITE THE CONTROL REG
0F6C 2947      ORI          CLK_CTRL_0        ; INACTIVATE CONTROL BIT 0
0F6E 2948      MOV          M,A              ; WRITE THE CONTROL REG
0F6F 2949      MVI          A,<^XFF>          ; SETUP TO DECREMENT THE COUNT
0F71 2950      ADD          B              ; DECREMENT THE COUNT
0F72 2951      MOV          B,A              ; ...
0F73 2952      JNZ          1$              ; CONTINUE UNTIL 0
0F76 2953      LDA          FRONT_PNL_2      ; GET STATE OF LIGHTS
0F79 2954      ANI          <TEST!FAULT!REMOTE> ; ...
0F7B 2955      ORI          <MASTER_HALT_EN!- ; SET MASTER HALT ENABLE AND TRAP
0F7B 2956      TRAP_HALT_EN> ; HALT ENABLE
0F7D 2957      STA          RD_CTRL_REG      ; CLEAR MASTER HALT ENABLE
0F80 2958      RET
0F81 2959
```

```
0F81 2961      .SBTTL      ABORT ROUTINE
0F81 2962      ;++
0F81 2963      ;
0F81 2964      ; FUNCTIONAL DESCRIPTION:
0F81 2965      ;
0F81 2966      ;     THIS ROUTINE INITIALIZES THE CPU AND THE STACK POINTER AND
0F81 2967      ;     RETURNS TO THE RDM MONITOR.
0F81 2968      ;
0F81 2969      ; CALLING SEQUENCE:
0F81 2970      ;
0F81 2971      ;     THIS ROUTINE IS CALLED BY THE FOLLOWING MACRO:
0F81 2972      ;
0F81 2973      ;     ABORT
0F81 2974      ;
0F81 2975      ;     THE ROUTINE IS CALLED BY THE FOLLOWING ROUTINES:
0F81 2976      ;
0F81 2977      ;     PROGRAM_INIT
0F81 2978      ;     TYPE_ASCII
0F81 2979      ;     TERM_INTERRUPT
0F81 2980      ;     RETURN_ACTION
0F81 2981      ;
0F81 2982      ; IMPLICIT INPUTS:
0F81 2983      ;
0F81 2984      ;     NONE
0F81 2985      ;
0F81 2986      ; IMPLICIT OUTPUTS:
0F81 2987      ;
0F81 2988      ;     NONE
0F81 2989      ;
0F81 2990      ; SIDE EFFECTS:
0F81 2991      ;
0F81 2992      ;     THE DIAGNOSTIC IS EXITED.
0F81 2993      ;
0F81 2994      ;--
0F81 2995      ;
0F81 2996      RETURN_TO_RDM:
0F81 2997      CALL      CPU_INIT      ; ASCERT DC LOW
0F84 2998      CALL      ENABLE_CMI   ; ENABLE THE CMI WITH PROC INIT
0F87 2999      CALL      FETCH_0      ; FETCH CS LOCATION 0
0F8A 3000      MVI      A,0          ; TURN ON THE CLOCK
0F8C 3001      STA      MAINT_REG    ; ENABLE THE CLOCK INTERRUPT
0F8F 3002      STA      DCS_CTRL_REG ; TURN ON THE CPU CLOCK
0F92 3003      LDA      FRONT_PNL_2 ; GET STATE OF LIGHTS
0F95 3004      ANI      <REMOTE!FAULT!TEST>; ...
0F97 3005      STA      RD_CTRL_REG  ; CLEAR MASTER HALT ENABLE
0F9A 3006      LHLD     SAVED_SP     ; GET ADDRESS OF RDM RETURN POINT
0F9D 3007      SPHL     ; LOAD THE STACK POINTER
0F9E 3008      RET      ; RETURN TO RDM
0F9F 3009
```

```

0F9F 3011      .SBTTL      SET DCS CONTROL FILE ROUTINE
0F9F 3012      ;++
0F9F 3013      ;
0F9F 3014      ; FUNCTIONAL DESCRIPTION:
0F9F 3015      ;
0F9F 3016      ;     THIS ROUTINE SETS BITS IN THE DCS CONTROL FILE. IT IS ENTERED WITH
0F9F 3017      ;     THE DCS ADDRESS AND A BIT MAP.
0F9F 3018      ;
0F9F 3019      ; CALLING SEQUENCE:
0F9F 3020      ;
0F9F 3021      ;     CALL      SET_DCS_CF
0F9F 3022      ;
0F9F 3023      ;     THE ROUTINE IS CALLED BY THE FOLLOWING ROUTINES:
0F9F 3024      ;
0F9F 3025      ;           SET_ACTION
0F9F 3026      ;           BURST_CLOCK
0F9F 3027      ;
0F9F 3028      ; IMPLICIT INPUTS:
0F9F 3029      ;
0F9F 3030      ;     SETCLR_CF_ADDR - CONTAINS THE DCS ADDRESS TO SET THE BITS AT.
0F9F 3031      ;     SETCLR_CF_DATA - CONTAINS THE BIT MAP OF THE BITS TO SET.
0F9F 3032      ;     DCS_CTRL_RE_COPY- CONTAINS THE CURRENT VALUE OF THE DCS CONTROL REGISTER
0F9F 3033      ;
0F9F 3034      ; IMPLICIT OUTPUTS:
0F9F 3035      ;
0F9F 3036      ;     NONE
0F9F 3037      ;
0F9F 3038      ;--
0F9F 3039
0F9F 3040 SET_DCS_CF:
0F9F 3041      LDA      DCS_ADDR_REG_RO ; GET THE CURRENT DCS ADDRESS
0FA2 3042      DCR      A              ; BACKUP TO CURRENT MICRO INSTR
0FA3 3043      PUSH     $PSW           ; SAVE IT ON THE STACK
0FA4 3044      CALL     DESELECT_DCS ; MAKE SURE DCS IS DESELECTED
0FA7 3045      LDA      SETCLR_CF_ADDR ; GET ADDRESS TO SET BITS AT
0FAA 3046      STA      DCS_ADDR_REG_WO ; LOAD THE DCS ADDRESS REGISTER
0FAD 3047      STA      DCS_ADDR_REG_WO ;
0FB0 3048      LDA      SETCLR_CF_DATA ; GET BITS TO SET
0FB3 3049      CMA      ;
0FB4 3050      MOV      B,A           ; SAVE IN B REG
0FB5 3051      LDA      DCS_CTRL_FILE ; GET THE CURRENT BITS
0FB8 3052      ANA      B              ; INSERT NEW BITS
0FB9 3053      STA      DCS_DATA_REG  ; LOAD THE CONTROL FILE
0FBC 3054      CALL     SELECT_DCS  ; RESTORE THE MATCH REGISTER
0FBF 3055      POP      $PSW         ; GET SAVED DCS ADDRESS REG
0FC0 3056      STA      DCS_ADDR_REG_WO ; RESTORE THE SAVED ADDRESS
0FC3 3057      STA      DCS_ADDR_REG_WO ;
0FC6 3058      PUSH     $PSW         ; SAVE A REG
0FC7 3059      MVI      A,<MAINT_TRAP_OFF!- ; DISABLE LOADING THE CONTROL FILE
0FC7 3060      MAINT_STB_INH!-
0FC7 3061      MAINT_DCS_ENABL>;
0FC9 3062      STA      MAINT_REG      ;
0FCC 3063      POP      $PSW         ; GET SAVED DCS ADDRESS
0FCD 3064      INR      A              ; POINT AT ADDRESS BEING FETCHED
0FCE 3065      STA      DCS_ADDR_REG_WO ; LOAD THE ADDRESS REGISTER
  
```

ZZ-ECKAA-8.7 V08.07
ECKAA
V08.07

H 7
SET DCS CONTR11-FEB-1986
VAX-11/750 MICRO DIAGNOSTIC MONITOR
SET DCS CONTROL FILE ROUTINE

Fiche 1 Frame H7 Sequence 85
11-FEB-1986 10:07:52 VAX/VMS Macro V04-00 Page 60
11-FEB-1986 10:06:37 [VAX750.MONITOR]ECKAA.MAR;811 (18)

OFD1 3066 MVI
OFD1 3067
OFD3 3068 STA
OFD6 3069 RET
OFD7 3070

A,<MAINT_TRAP_OFF!-
MAINT_DCS_ENABL>; DEASCERT STROBE INHIBIT
MAINT_REG ;
; EXIT

```

OFD7 3072      .SBTTL      CLEAR DCS CONTROL FILE ROUTINE
OFD7 3073      ;**
OFD7 3074      ;
OFD7 3075      ; FUNCTIONAL DESCRIPTION:
OFD7 3076      ;
OFD7 3077      ;     THIS ROUTINE CLEARS BITS IN THE DCS CONTROL FILE. IT IS ENTERED
OFD7 3078      ;     WITH THE DCS ADDRESS AND A BIT MAP OF THE BITS TO CLEAR.
OFD7 3079      ;
OFD7 3080      ; CALLING SEQUENCE:
OFD7 3081      ;
OFD7 3082      ;     CALL      CLEAR_DCS_CF
OFD7 3083      ;
OFD7 3084      ;     THE ROUTINE IS CALLED BY THE FOLLOWING ROUTINES:
OFD7 3085      ;
OFD7 3086      ;           CLEAR_ACTION
OFD7 3087      ;           BURST_CLOCK
OFD7 3088      ;
OFD7 3089      ; IMPLICIT INPUTS:
OFD7 3090      ;
OFD7 3091      ;     SETCLR_CF_ADDR - CONTAINS THE DCS ADDRESS TO CLEAR THE BITS AT
OFD7 3092      ;     SETCLR_CF_DATA - CONTAINS THE BIT MAP OF THE BITS TO CLEAR
OFD7 3093      ;
OFD7 3094      ; IMPLICIT OUTPUTS:
OFD7 3095      ;
OFD7 3096      ;     NONE
OFD7 3097      ;
OFD7 3098      ;--
OFD7 3099      ;
OFD7 3100     CLEAR_DCS_CF:
OFD7 3101     LDA      DCS_ADDR_REG_RO ; GET THE CURRENT DCS ADDRESS
OFDA 3102     DCR      A                ; POINT AT CURRENTLY LATCHED ADDRESS
OFDB 3103     PUSH     $PSW             ; SAVE
OFDC 3104     CALL    DESELECT_DCS    ; MAKE SURE DCS IS DESELECTED
OFDF 3105     LDA      SETCLR_CF_ADDR ; GET ADDRESS TO CLEAR THE BITS AT
OFE2 3106     STA      DCS_ADDR_REG_WO ; LOAD THE DCS ADDR REG
OFE5 3107     STA      DCS_ADDR_REG_WO ;
OFE8 3108     LDA      SETCLR_CF_DATA ; GET BIT MAP TO CLEAR
OFEB 3109     MOV      B,A             ; SAVE IN B REG
OFEC 3110     LDA      DCS_CTRL_FILE  ; GET THE CURRENT BITS IN THE CTRL FILE
OFEF 3111     ORA      B                ; CLEAR THE BITS IN THE CF
OFF0 3112     STA      DCS_DATA_REG   ; WRITE THE CONTROL FILE
OFF3 3113     CALL    SELECT_DCS     ; RESTORE THE MATCH REGISTER
OFF6 3114     POP      $PSW           ; GET SAVED DCS ADDRESS
OFF7 3115     STA      DCS_ADDR_REG_WO ; RESTORE THE DCS ADDRESS
OFFA 3116     STA      DCS_ADDR_REG_WO ;
OFFD 3117     PUSH     $PSW           ; SAVE A REG
OFFE 3118     MVI      A,<MAINT_TRAP_OFF!- ; DISABLE LOADING THE
OFFE 3119     MAINT_STB_INH!-
OFFE 3120     MAINT_DCS_ENABL>; CONTROL FILE
1000 3121     STA      MAINT_REG      ; OF CONTROL FILE
1003 3122     POP      $PSW           ; GET SAVED DCS ADDRESS
1004 3123     INR      A                ; POINT AT ADDRESS BEING FETCHED
1005 3124     STA      DCS_ADDR_REG_WO ; LOAD THE ADDRESS REGISTER
1008 3125     MVI      A,<MAINT_TRAP_OFF!-
1008 3126     MAINT_DCS_ENABL>; DISABLE THE STROBE INHIBIT
  
```

ZZ-ECKAA-8.7 V08.07
ECKAA
V08.07

J 7

CLEAR DCS CON11-FEB-1986 Fiche 1 Frame J7 Sequence 87
VAX-11/750 MICRO DIAGNOSTIC MONITOR 11-FEB-1986 10:07:52 VAX/VMS Macro V04-00 Page 62
CLEAR DCS CONTROL FILE ROUTINE 11-FEB-1986 10:06:37 [VAX750.MONITOR]ECKAA.MAR;811 (19)

100A	3127	STA	MAINT_REG	:	EXIT
100D	3128	RET		:	
100E	3129			:	

```
100E 3131      .SBTTL  READ OVERLAY ROUTINE
100E 3132      :++
100E 3133      :
100E 3134      : FUNCTIONAL DESCRIPTION:
100E 3135      :
100E 3136      : THIS ROUTINE READS A TEST OVERLAY FROM THE CURRENTLY OPEN
100E 3137      : FILE ON THE TUS8. THE OVERLAY FORMAT IS AS FOLLOWS. THE OVERLAY
100E 3138      : HAS TWO FORMATS DEPENDING ON THE CONTENTS OF THE FIRST BYTE:
100E 3139      :
100E 3140      : FORMAT 1 - NO DATA TO LOAD INTO CPU OR DCS
100E 3141      : .BYTE  1 (BITS <6:0>)
100E 3142      : .BYTE  NOT USED
100E 3143      : .BYTE  NUMBER OF RECORDS IN THE TEST
100E 3144      : .ASCIC TEST TITLE
100E 3145      : .BLKB  TEST INSTRUCTIONS
100E 3146      :
100E 3147      :
100E 3148      : .BLKB  FILL TO A RECORD BOUNDRY
100E 3149      :
100E 3150      : FORMAT 2 - DATA TO LOAD INTO CPU OR DCS
100E 3151      : .BYTE  0 (BITS <6:0>)
100E 3152      : .BYTE  NUMBER OF RECORDS OF DATA
100E 3153      : .BYTE  NUMBER OF R TEMP LONG WORDS
100E 3154      : .BYTE  R TEMP DATA
100E 3155      :
100E 3156      : .BYTE  NUMBER OF CACHE LONG WORDS
100E 3157      : .BYTE  CACHE DATA
100E 3158      :
100E 3159      : .BYTE  NUMBER OF DCS MICRO WORDS
100E 3160      : .BYTE  DCS DATA
100E 3161      :
100E 3162      :
100E 3163      : .BLKB  FILL TO A RECORD BOUNDRY
100E 3164      : .BYTE  NUMBER OF RECORDS IN THE TEST
100E 3165      : .ASCIC TEST TITLE
100E 3166      : .BYTE  TEST INSTRUCTIONS
100E 3167      :
100E 3168      :
100E 3169      : .BLKB  FILL TO A RECORD BOUNDRY
100E 3170      :
100E 3171      : THE RTEMP, CACHE, AND DCS DATA IS LOADED INTO THE RESPECTIVE
100E 3172      : PLACES, THEN THE THE TEST IS READ INTO THE BUFFER.
100E 3173      :
100E 3174      : IF NO DCS DATA IS PRESENT IN THE OVERLAY, THE DCS IS LOADED WITH NOP
100E 3175      : MICRO INSTRUCTIONS.
100E 3176      :
100E 3177      : CALLING SEQUENCE:
100E 3178      :
100E 3179      : CALL  READ_OVERLAY
100E 3180      :
100E 3181      : THE ROUTINE IS CALLED BY THE FOLLOWING ROUTINES:
100E 3182      :
100E 3183      : PROGRAM_INIT
100E 3184      : END_TEST
100E 3185      :
```

```

100E 3186 ; THE ROUTINE CALLS THE FOLLOWING ROUTINES:
100E 3187 ;
100E 3188 ; READ_TU58
100E 3189 ; WRITE_DCS
100E 3190 ; TYPE_TEST_NUMB
100E 3191 ;
100E 3192 ; IMPLICIT INPUTS:
100E 3193 ;
100E 3194 ; CURRENT_REC_NMB - CONTAINS THE RECORD NUMBER OF THE CURRENTLY
100E 3195 ; OPEN FILE.
100E 3196 ; TEST_BUFFER - IS THE ADDRESS OF THE READ BUFFER
100E 3197 ;
100E 3198 ; IMPLICIT OUTPUTS:
100E 3199 ;
100E 3200 ; CURRENT_REC_NMB - POINTS TO THE CURRENT RECORD OF THE CURRENTLY
100E 3201 ; OPEN FILE.
100E 3202 ; INIT_TEST_PC - GETS LOADED WITH THE PC OF THE BASE OF THE TEST
100E 3203 ;
100E 3204 ;--
100E 3205
100E 3206 READ_OVERLAY:
100E 3207 READ_RECORD TEST_BUFFER,1 ; READ THE FIRST RECORD OF THE
1016 3208 ; OVERLAY INTO 'TEST_BUFFER'
1016 3209 ;
1016 3210 ; BUFFER POINTER IS IN H & L
1016 3211 ;
1016 3212 MOV A,M ; GET THE DATA FLAG OF THE OVERLAY
1017 3213 PUSH $PSW ; SAVE
1018 3214 ANI 1 ; CPU DATA?
101A 3215 JNZ 40$ ; BRANCH IF NO
101D 3216 INX H ; POINT AT RECORD COUNT FOR DATA
101E 3217 MOV A,M ; GET THE RECORD COUNT
101F 3218 DCR A ; SUBTRACT RECORD JUST READ
1020 3219 JZ 1$ ; BRANCH IF ALL RECORDS IN MEMORY
1023 3220 ;
1023 3221 ; READ THE REST OF THE DATA INTO THE BUFFER
1023 3222 ;
1023 3223 MOV B,A ; SAVE A
1024 3224 POP $PSW ; GET TEST SECTION FLAG
1025 3225 PUSH $PSW ; SAVE AGAIN
1026 3226 CALL CHECK_QV ; CHECK IF QV PASS
1029 3227 JC 30$ ; BRANCH IF TEST IS TO BE SKIPPED
102C 3228 MOV A,B ; RESTORE RECORD NUMBER
102D 3229 PUSH H ; SAVE THE BUFFER POINTER
102E 3230 READ_RECORD TEST_BUFFER+128 ; READ THE REST OF THE DATA
1034 3231 POP H
1035 3232 ;
1035 3233 ;
1035 3234 ; FILL THE DCS WITH NOP INSTRUCTIONS.
1035 3235 ;
1035 3236 1$: PUSH H
1036 3237 CALL FILL_DCS ; FILL DCS WITH NOP'S
1039 3238 POP H ; RESTORE THE BUFFER POINTER
103A 3239 ;
103A 3240 ; NOW LOAD THE R TEMP DATA

```

```

103A 3241 ;
103A 3242      INX      H      ; INCREMENT BUFFER POINTER
103B 3243      MOV      A,M    ; GET THE WORD COUNT
103C 3244      CPI      -1     ; IS THERE ANY R TEMP DATA?
103E 3245      JZ       10$    ; BRANCH IF NO
1041 3246
1041 3247      INX      H      ; POINT AT FIRST BYTE OF DATA
1042 3248      PUSH     $PSW   ; SAVE THE COUNT
1043 3249      XRA      A      ; INIT THE REGISTER NUMBER
1044 3250      STA      TYPE_ERR_TEMP ;
1047 3251 5$:   PUSH     H      ; SAVE DATA POINTER
1048 3252      LXI      D,R RDM MIC WORD ; GET ADDRESS OF MICRO WORD
104B 3253      LXI      H,TYPE_ERR_TEMP ; AND ADDRESS OF REGISTER NUMBER
104E 3254      MVI      B,34    ; PUT STARTING BIT NUMBER IN B REG
1050 3255      MVI      C,39    ; AND ENDING BIT NUMBER
1052 3256      XRA      A      ; CLEAR INHIBIT PARITY FLAG
1053 3257      PUSH     $PSW   ; CLEAR INHIBIT SCRAMBLE FLAG
1054 3258      CALL     INSERT_FIELD ; PUT REGISTER NUMBER IN MICRO WORD
1057 3259      WRITE$ _DCS DCS_SCRATCH_ADR,- ;
1057 3260      WRITE$ _DCS R_RDM_MIC_WORD,1 ; LOAD THE DCS
106A 3261      WRITE$ _DCS DCS_SCRATCH_ADR+1,- ;
106A 3262      NOP     _MICRO_WORD,1 ;
107D 3263      POP      H      ; RESTORE H & L
107E 3264 ;
107E 3265 ; PUT THE DATA IN THE D REGISTER
107E 3266 ;
107E 3267      LXI      D,WD_REG_BYTE_0 ; GET ADDRESS OF D REGISTER
1081 3268      MVI      C,4      ; SETUP LOOP FOR 4 BYTES
1083 3269 7$:   MOV      A,M    ; GET BYTE OF DATA
1084 3270      STAX     D      ; PUT IN D REGISTER
1085 3271      INX      H      ; INCREMENT POINTERS
1086 3272      INX      D      ;
1087 3273      DCR      C      ; DONE 4 BYTES?
1088 3274      JNZ     7$     ; BRANCH IF NO
108B 3275 ;
108B 3276 ; EXECUTE THE MICRO INSTRUCTION
108B 3277 ;
108B 3278      EXECUTE DCS_SCRATCH_ADR,1 ; EXECUTE ONE MICRO INSTRUCTION
109E 3279      LDA      TYPE_ERR_TEMP ; INCREMENT THE REGISTER NUMBER
10A1 3280      INR      A      ; ...
10A2 3281      STA      TYPE_ERR_TEMP ;
10A5 3282      POP     $PSW   ; GET THE REGISTER COUNT BACK
10A6 3283      DCR      A      ; LOADED ALL THE REGISTERS?
10A7 3284      PUSH     $PSW   ; SAVE THE REGISTER COUNT
10A8 3285      JNZ     5$     ; BRANCH IF NO
10AB 3286      POP     $PSW   ; CLEANUP THE STACK
10AC 3287      DCX      H      ; BACKUP POINTER FOR CACHE DATA
10AD 3288 ;
10AD 3289 ; NOW LOAD THE CACHE DATA
10AD 3290 ;
10AD 3291 10$:  INX      H      ; INC POINTER
10AE 3292      MOV      A,M    ; GET THE COUNT
10AF 3293      CPI      -1     ; IS THERE ANY CACHE DATA?
10B1 3294      JZ       20$    ; BRANCH IF NO
10B4 3295

```

```

10B4 3296      INX      H      ; POINT AT THE FIRST DATA WORD
10B5 3297      PUSH     $PSW   ; SAVE THE WORD COUNT
10B6 3298      PUSH     H      ; SAVE THE DATA POINTER
10B7 3299      CALL     DISABLE_CMI ; DISABLE THE CMI
10BA 3300      ;
10BA 3301      ; ENABLE THE CACHE
10BA 3302      ; MICRO CODE TO DISABLE THE CMI IS ALREADY IN DCS
10BA 3303      ;
10BA 3304      WRITE$_DCS DCS_SCRATCH_ADR+1,-
10BA 3305      CACHE_ENA_ADR,1 ; LOAD LONLIT TO SELECT REGISTER
10CD 3306      WRITE$_DCS DCS_SCRATCH_ADR+3,-
10CD 3307      CACHE_ENA_DATA,1 ; AND THE DATA TO WRITE
10E0 3308      EXECUTE  DCS_SCRATCH_ADR+1,4 ; EXECUTE THE INSTRUCTIONS
10F3 3309      ;
10F3 3310      ; NOW LOAD THE MICRO CODE TO LOAD THE CACHE
10F3 3311      ;
10F3 3312      WRITE$_DCS DCS_SCRATCH_ADR,-
10F3 3313      CACHE_WRITE,4 ; LOAD THE MICRO CODE
1106 3314      LXI      H,0      ; INIT THE VA TO ZERO
1109 3315      SHLD     WD_REG_BYTE_0 ; ...
110C 3316      SHLD     WD_REG_BYTE_2 ; ...
110F 3317      EXECUTE  DCS_SCRATCH_ADR,1 ; ...
1122 3318      POP      H      ; GET THE ADDRESS OF THE DATA
1123 3319      ;
1123 3320      ; PUT A DATA WORD IN THE D REGISTER
1123 3321      ;
1123 3322      13$: MVI      B,4      ; SET THE LOOP COUNT
1125 3323      LXI      D,WD_REG_BYTE_0 ; GET ADDRESS OF D REGISTER
1128 3324      15$: MOV      A,M      ; GET A BYTE OF DATA
1129 3325      STAX     D      ; PUT IN D REGISTER
112A 3326      INX      H      ; INCREMENT THE POINTERS
112B 3327      INX      D      ;
112C 3328      DCR      B      ; DONE 4 BYTES?
112D 3329      JNZ     15$      ; BRANCH IF NO
1130 3330      ;
1130 3331      ; LOAD THE CACHE
1130 3332      ;
1130 3333      EXECUTE  DCS_SCRATCH_ADR+1,3
1143 3334      ;
1143 3335      ; CHECK IF ALL DATA LOADED
1143 3336      ;
1143 3337      POP      $PSW   ; GET THE WORD COUNT
1144 3338      DCR      A      ; DONE?
1145 3339      PUSH     $PSW   ; SAVE THE WORD COUNT
1146 3340      JNZ     13$      ; BRANCH IF MORE WORDS
1149 3341      POP      $PSW   ; CLEANUP THE STACK
114A 3342      DCX      H      ; BACKUP THE POINTER
114B 3343      ;
114B 3344      ; NOW LOAD THE DCS
114B 3345      ;
114B 3346      ; NOW LOAD THE DCS IF THERE IS DATA TO LOAD
114B 3347      ;
114B 3348      20$: INX      H      ; INC POINTER
114C 3349      MOV      A,M      ; GET THE DCS WORD COUNT
114D 3350      MOV      B,A      ; SAVE IN B REGISTER

```

```

114E 3351      CPI          -1      ; IS THERE ANY DCS DATA?
1150 3352      JZ           30$     ; BRANCH IF NO
1153 3353      INX          H       ; POINT AT FIRST DCS BYTE
1154 3354      WRITE$_DCS   0       ; LOAD DCS WITH THE MICRO WORDS
1163 3355      ;
1163 3356      ; NOW READ THE TEST INTO THE TEST BUFFER
1163 3357      ;
1163 3358      30$:      LXI          H,TEST_BUFFER ; INIT THE INITIAL TEST PC
1166 3359      INX          H       ; ...
1167 3360      SHLD         INIT_TEST_PC ; ...
116A 3361      MOV          A,L     ; AND THE TWO'S COMPLIMENT OF THE
116B 3362      CMA          ; INITIAL TEST PC
116C 3363      MOV          L,A     ; ...
116D 3364      MOV          A,H     ; ...
116E 3365      CMA          ; ...
116F 3366      MOV          H,A     ; ...
1170 3367      INX          H       ; ...
1171 3368      SHLD         C_INIT_TEST_PC ; ...
1174 3369      READ_RECORD  TEST_BUFFER,1 ; READ THE NEXT RECORD
117C 3370      32$:      MOV          A,M     ; GET THE NUMBER OF RECORDS IN THE TEST
117D 3371      STA          TEST_SIZE ; SAVE
1180 3372      DCR          A       ; ACCOUNT FOR ONE JUST READ
1181 3373      MOV          B,A     ; SAVE NUMBER OF RECORDS
1182 3374      POP          $PSW    ; GET TEST SECTION FLAG
1183 3375      PUSH         $PSW    ; SAVE AGAIN
1184 3376      CALL         CHECK_QV ; CHECK IF QV PASS
1187 3377      JNC          33$     ; BRANCH IF TEST IS NOT TO BE SKIPPED
118A 3378      MVI          B,1     ; SETUP FOR NO TYPEOUT
118C 3379      CALL         TYPE_TEST_NUMB ; UPDATE TEST NUMBER
118F 3380      POP          $PSW    ; DISCARD TEST SECTION FLAG
1190 3381      LDA          TEST_NUMBER ; UPDATE TEST NUMBER
1193 3382      INR          A       ; ...
1194 3383      STA          TEST_NUMBER ; ...
1197 3384      MOV          B,A     ; SAVE
1198 3385      LDA          TOTAL_NMB_TESTS ; GET TOTAL NUMBER OF TESTS ON TAPE
119B 3386      CMP          B       ; DONE YET?
119C 3387      JNZ          READ_OVERLAY ; BRANCH IF NO
119F 3388      ;
119F 3389      ; SKIPPED THE LAST TEST, SIMULATE AN END_FILE PSEUDO OP.
119F 3390      ;
119F 3391      LHL         INIT_TEST_PC ; GET INITIAL TEST PC
11A2 3392      SHLD         TEST_PC   ; ...
11A5 3393      MVI          M,OP_END_FILE ; PUT END FILE OPCODE IN MEMORY
11A7 3394      INX          H       ; ...
11A8 3395      MVI          M,0     ; ...
11AA 3396      JMP          36$     ; EXIT
11AD 3397      ;
11AD 3398      ; NOT A QV PASS OR TEST IS PART OF QV SECTION. READ IT IN
11AD 3399      ;
11AD 3400      33$:      MOV          A,B     ; RESTORE NUMBER OF RECORDS TO READ
11AE 3401      ORA          A       ; ANY MORE RECORDS TO READ?
11AF 3402      JZ           31$     ; BRANCH IF NO
11B2 3403      READ_RECORD  TEST_BUFFER+128 ; READ THE REST OF THE TEST INTO MEMORY
11B8 3404      31$:      MVI          B,0     ; SETUP TO TYPE TEST NUMBER
11BA 3405      CALL         TYPE_TEST_NUMB ; TYPE THE TEST NUMBER

```

ZZ-ECKAA-8.7 V08.07
ECKAA
V08.07

C 8

" READ OVERLAY 11-FEB-1986
VAX-11/750 MICRO DIAGNOSTIC MONITOR
READ OVERLAY ROUTINE

Fiche 1 Frame C8 Sequence 93
11-FEB-1986 10:07:52 VAX/VMS Macro V04-00 Page 68
11-FEB-1986 10:06:37 [VAX750.MONITOR]ECKAA.MAR;811 (20)

```
11BD 3406 35$: POP          $PSW          ; RETURN THE TEST SECTION FLAG
11BE 3407 36$: RET          ; EXIT
11BF 3408
11BF 3409 ;
11BF 3410 ; THERE WAS NO DATA FOR THIS TEST. READ THE REST OF THE TEST INTO MEMORY.
11BF 3411 ;
11BF 3412 40$: INX          H              ; POINT AT THE FIRST INSTRUCTION
11C0 3413          INX          H              ; FOR THE TEST
11C1 3414          PUSH        H              ; SAVE H & L
11C2 3415          INX          H              ;
11C3 3416          SHLD        INIT_TEST_PC   ; INIT THE INITIAL TEST PC
11C6 3417          MOV         A,L           ; AND THE TWO'S COMPLIMENT INITIAL
11C7 3418          CMA          ; TEST PC
11C8 3419          MOV         L,A           ; ...
11C9 3420          MOV         A,H           ; ...
11CA 3421          CMA          ; ...
11CB 3422          MOV         H,A           ; ...
11CC 3423          INX          H              ; ...
11CD 3424          SHLD        C_INIT_TEST_PC ; ...
11D0 3425          POP         H              ; POINT BACK AT RECORD COUNT
11D1 3426          JMP         32$          ; READ THE REST OF THE TEST INTO MEMORY
11D4 3427
```

```

11D4 3429      .SBTTL      CHECK QV ROUTINE
11D4 3430      ;++
11D4 3431      ;
11D4 3432      ; FUNCTIONAL DESCRIPTION:
11D4 3433      ;
11D4 3434      ;     THIS ROUTINE IS USED TO CHECK IF A QV PASS HAS BEEN SPECIFIED AND
11D4 3435      ;     IF THE TEST TO BE EXECUTED IS IN THE QV SECTION.
11D4 3436      ;
11D4 3437      ; CALLING SEQUENCE:
11D4 3438      ;
11D4 3439      ;     CALL      CHECK_QV
11D4 3440      ;
11D4 3441      ;     THE ROUTINE IS CALLED BY THE  READ_OVERLAY  ROUTINE.
11D4 3442      ;
11D4 3443      ; EXPLICIT INPUTS:
11D4 3444      ;
11D4 3445      ;     A - CONTAINS THE TEST SECTION FLAG
11D4 3446      ;     B - CONTAINS THE NUMBER OF RECORDS TO SKIP
11D4 3447      ;
11D4 3448      ; EXPLICIT OUTPUTS:
11D4 3449      ;
11D4 3450      ;     B - UNCHANGED
11D4 3451      ;
11D4 3452      ; IMPLICIT OUTPUTS:
11D4 3453      ;
11D4 3454      ;     CURRENT_REC_NMB - UPDATED IF THIS IS A QV PASS AND THE TEST IS
11D4 3455      ;     NOT PART OF THE QV SECTION.
11D4 3456      ;
11D4 3457      ; ROUTINE VALUE:
11D4 3458      ;
11D4 3459      ;     CF - SET IF THE CURRENT_REC_NMB WAS UPDATED.
11D4 3460      ;--
11D4 3461      CHECK_QV:
11D4 3462      PUSH      $PSW          ; SAVE A REG
11D5 3463      LDA      SOFT_CTRL_REGB ; GET QV FLAG
11D8 3464      ANI      QV_FLAG      ;
11DA 3465      JZ       10$          ; BRANCH IF NOT QV PASS
11DD 3466      POP      $PSW
11DE 3467      PUSH      $PSW
11DF 3468      ANI      <^X80>      ; IS THIS TEST IN QV SECTION?
11E1 3469      JNZ      10$          ; BRANCH IF YES
11E4 3470      ;
11E4 3471      ; SKIP THIS TEST
11E4 3472      ;
11E4 3473      PUSH      H
11E5 3474      LHL     CURRENT_REC_NMB ; GET CURRENT TAPE RECORD NUMBER
11E8 3475      PUSH      B
11E9 3476      MOV     C,B
11EA 3477      MVI     B,0          ; SKIP DATA FOR THIS TEST
11EC 3478      DAD     B
11ED 3479      POP     B
11EE 3480      SHLD   CURRENT_REC_NMB ; RESTORE
11F1 3481      POP     H
11F2 3482      POP     $PSW
11F3 3483      STC

```

ZZ-ECKAA-8.7 V08.07
ECKAA
V08.07

E 8
" CHECK QV ROUT11-FEB-1986
VAX-11/750 MICRO DIAGNOSTIC MONITOR
CHECK QV ROUTINE

Fiche 1 Frame E8 Sequence 95
11-FEB-1986 10:07:52 VAX/VMS Macro V04-00 Page 70
11-FEB-1986 10:06:37 [VAX750.MONITOR]ECKAA.MAR;811 (21)

11F4	3484		JMP	20\$; EXIT
11F7	3485	10\$:	POP	\$PSW		
11F8	3486		STC			
11F9	3487		CMC			
11FA	3488	20\$:	RET			
11FB	3489					

```

11FB 3491      .SBTTL      READ V BUS ROUTINE
11FB 3492      ;++
11FB 3493      ;
11FB 3494      ; FUNCTIONAL DESCRIPTION:
11FB 3495      ;
11FB 3496      ;     THIS ROUTINE READS THE CURRENT STATE OF THE VISIBILITY BUS
11FB 3497      ;     INTO A TEMPORARY BUFFER. EACH BYTE OF THE BUFFER CONTAINS ONE BIT
11FB 3498      ;     OF THE V BUS IN BIT POSITION 0.
11FB 3499      ;
11FB 3500      ; CALLING SEQUENCE:
11FB 3501      ;
11FB 3502      ;     CALL      READ_V_BUS
11FB 3503      ;
11FB 3504      ;     THE ROUTINE IS CALLED BY THE FOLLOWING ROUTINES:
11FB 3505      ;
11FB 3506      ;           SHOW_ACTION
11FB 3507      ;           COMPARE_VBUS
11FB 3508      ;
11FB 3509      ; IMPLICIT INPUTS:
11FB 3510      ;
11FB 3511      ;     NONE
11FB 3512      ;
11FB 3513      ; IMPLICIT OUTPUTS:
11FB 3514      ;
11FB 3515      ;     V_BUS_BUFFER - CONTAINS THE CURRENT STATE OF THE V BUS.
11FB 3516      ;
11FB 3517      ;--
11FB 3518      ;
11FB 3519      READ_V_BUS:
11FB 3520          MVI      B,40          ; SET A LOOP COUNT
11FD 3521          LXI      H,V_BUS_BUFFER ; GET ADDRESS OF BUFFER
1200 3522 1$:      LDA      STATUS_REG ; READ THE V BUS
1203 3523          ANI      VBUS_SERIAL_OUT ; DISCARD UPPER BITS
1205 3524          MOV      M,A          ; PUT IN BUFFER
1206 3525          INX      H          ; INCREMENT BUFFER ADDRESS
1207 3526          MVI      A,VBUS_CLOCK ; GET DATA TO CLOCK THE V BUS
1209 3527          STA      ADDR_MATCH_HI ; CLOCK THE BUS
120C 3528          XRA      A          ; ...
120D 3529          STA      ADDR_MATCH_HI ; ...
1210 3530          DCR      B          ; DECREMENT THE LOOP COUNT
1211 3531          JNZ      1$          ; BRANCH IF NOT DONE
1214 3532      ;
1214 3533      ; RESTORE THE MATCH REGISTER TO THE SOMM ADDRESS
1214 3534      ;
1214 3535          LDA      SOMM_ADDRESS+1 ; GET UPPER ORDER SOMM ADDRESS
1217 3536          STA      ADDR_MATCH_HI ; RESTORE THE MATCH REGISTER
121A 3537          RET
121B 3538

```

```
121B 3540      .SBTTL      DIRECTORY SEARCH ROUTINE
121B 3541      ;+
121B 3542      ;
121B 3543      ; FUNCTIONAL DESCRIPTION:
121B 3544      ;
121B 3545      ; THIS ROUTINE IS USED TO FIND THE STARTING RECORD NUMBER OF AN
121B 3546      ; OVERLAY. THE ROUTINE MUST BE CALLED WITH THE STARTING RECORD
121B 3547      ; NUMBER OF THE FILE AND THE SPECIFIED TEST NUMBER IF THE LOST
121B 3548      ; FLAG IS SET. THE ROUTINE RETURNS THE STARTING RECORD NUMBER
121B 3549      ; OF THE SPECIFIED TEST OVERLAY.
121B 3550      ;
121B 3551      ; THE ROUTINE ALSO CHECKS THAT THE FILE ID CODE MATCHES THE
121B 3552      ; CURRENT FILE_NAM_INDEX. IF IT DOES, THE MODULE NAME IS TYPED
121B 3553      ; AND THE VERSION NUMBER OF THE FILE.
121B 3554      ; IF IT DOES NOT MATCH, A FAILURE RETURN IS MADE WITH -1 IN THE
121B 3555      ; A REGISTER.
121B 3556      ;
121B 3557      ; THE DIRECTORY FORMAT IS AS FOLLOWS:
121B 3558      ;
121B 3559      ;           .BYTE   RECORD LENGTH OF DIRECTORY
121B 3560      ;           .BYTE   TOTAL NUMBER OF TESTS IN FILE
121B 3561      ;           .BYTE   NUMBER OF CHARACTERS IN VERSION NUMBER
121B 3562      ;           .BLKB  ASCII VERSION NUMBER
121B 3563      ;           .BYTE   FILE ID CODE
121B 3564      ;           .BYTE   RECORD LENGTH OF TEST 1
121B 3565      ;           .BYTE   RECORD LENGTH OF TEST 2
121B 3566      ;           .
121B 3567      ;           .
121B 3568      ;           .BYTE   RECORD LENGTH OF TEST N
121B 3569      ;           .BYTE   FF
121B 3570      ;           .BYTE   FF
121B 3571      ;           .
121B 3572      ;           .
121B 3573      ;           FILLED TO END OF RECORD
121B 3574      ;
121B 3575      ; CALLING SEQUENCE:
121B 3576      ;
121B 3577      ;           CALL   DIRECT_SEARCH
121B 3578      ;
121B 3579      ;           THE ROUTINE IS CALLED BY THE 'PROGRAM_INIT' ROUTINE
121B 3580      ;
121B 3581      ;           THE ROUTINE CALLS THE FOLLOWING ROUTINES:
121B 3582      ;
121B 3583      ;           READ_TU58
121B 3584      ;
121B 3585      ; IMPLICIT INPUTS:
121B 3586      ;
121B 3587      ;           FILE_NAM_INDEX - CONTAINS THE FILE NAME CODE
121B 3588      ;           CURRENT_REC_NMB - CONTAINS THE STARTING RECORD NUMBER OF THE FILE
121B 3589      ;
121B 3590      ; IMPLICIT OUTPUTS:
121B 3591      ;
121B 3592      ;           CURRENT_REC_NMB - CONTAINS THE STARTING RECORD NUMBER OF THE TEST OVERLAY
121B 3593      ;           TOTAL_NMB_TESTS - CONTAINS THE NUMBER OF TESTS IN THE FILE
121B 3594      ;
```



```
1263 3650          TYPE          CRLF
126C 3651  ;
126C 3652  ; CHECK IF MORE DIRECTORY TO READ
126C 3653  ;
126C 3654          POP           $PSW          ; GET DIRECTORY LENGTH
126D 3655          PUSH          $PSW          ; SAVE
126E 3656          DCR           A             ; ACCOUNT FOR RECORD JUST READ
126F 3657          JZ            1$           ; BRANCH IF ALL DIRECTORY IS HERE
1272 3658  ;
1272 3659  ; READ THE REST OF THE DIRECTORY
1272 3660  ;
1272 3661          READ_RECORD   TEST_BUFFER+128 ; READ THE REST OF THE DIRECTORY
1278 3662  ;
1278 3663  ; CHECK IF THE LOST FLAG IS SET
1278 3664  ;
1278 3665 1$:     LDA           SOFT_CTRL_REG ; GET THE LOST FLAG
127B 3666          ANI           LOST_FLAG    ; IS IT SET?
127D 3667          JNZ          3$           ; BRANCH IF YES
1280 3668  ;
1280 3669  ; SPECIAL TEST IS NOT SPECIFIED SO JUST GET RECORD NUMBER OF FIRST TEST
1280 3670  ;
1280 3671          MVI           D,0         ; SET THE RECORD NUMBER OF THE
1282 3672          MVI           E,0         ; TEST TO ZERO
1284 3673          JMP           5$         ; GO CALCULATE PHYSICAL RECOD NUMBER
1287 3674  ;
1287 3675  ; SEARCH THE DIRECTORY FOR THE RELATIVE RECORD NUMBER OF THE SPECIFIED TEST.
1287 3676  ;
1287 3677 3$:     LDA           SPEC_TEST_NUMB ; GET THE SPECIFIED TEST NUMBER
128A 3678          MOV           B,A         ; SAVE AS A LOOP COUNT
128B 3679          LXI           H,TEST_BUFFER+10 ; GET ADDRESS OF DIRECTORY
128E 3680          LXI           D,0         ; INIT THE RECORD NUMBER
1291 3681 4$:     DCR           B             ; FOUND THE TEST YET?
1292 3682          JZ            8$         ; BRANCH IF YES
1295 3683          PUSH          H           ; SAVE THE REGISTERS
1296 3684          PUSH          D           ;
1297 3685          MOV           A,M         ; GET LOW BYTE OF TEST LENGTH
1298 3686          ORA           A           ; CHECK IF END OF FILE
1299 3687          JM            7$         ; BRANCH IF END OF FILE
129C 3688          MOV           E,A         ; PUT TEST LENGTH IN D & E
129D 3689          MVI           D,0         ;
129F 3690          POP           H           ; GET ACCUMULATING RECORD NUMBER
12A0 3691          DAD           D           ; ADD LENGTH OF THIS TEST
12A1 3692          XCHG          ; PUT IN D & E
12A2 3693          POP           H           ; GET DIRECTORY ADDRESS
12A3 3694          INX           H           ; POINT AT NEXT TEST ENTRY
12A4 3695          JMP           4$         ; CHECK IF DONE
12A7 3696  ;
12A7 3697  ; CHECK IF THIS TEST IS IN THE FILE
12A7 3698  ;
12A7 3699 8$:     MOV           A,M         ; GET THE DIRECTORY ENTRY FOR THIS
12A8 3700          ORA           A           ; TEST
12A9 3701          JP            5$         ; BRANCH IF TEST EXISTS
12AC 3702          JMP           9$         ; TEST DOES NOT EXIST
12AF 3703  ;
12AF 3704  ; FOUND THE END OF FILE BEFORE THE SPECIFIED TEST
```

```
12AF 3705 ;  
12AF 3706 7$: POP H ; CLEANUP THE STACK  
12B0 3707 POP H ; ...  
12B1 3708  
12B1 3709 9$: POP $PSW ; REMOVE SAVED DIRECTORY LENGTH FROM STACK  
12B2 3710 POP H ; REMOVE SAVED RECORD NUMBER FOM STACK  
12B3 3711 XRA A ; SET RETURN CODE  
12B4 3712 STC ; RETURN WITH C BIT SET  
12B5 3713 RET ; ...  
12B6 3714 ;  
12B6 3715 ; CALCULATE THE STARTING RECORD NUMBER OF THE TEST OVERLAY  
12B6 3716 ;  
12B6 3717 5$: POP $PSW ; GET DIRECTORY LENGTH  
12B7 3718 MOV L,A ; ...  
12B8 3719 MVI H,0 ; ...  
12BA 3720 DAD D ; ADD THE RECORD NUMBER OF THE TEST  
12BB 3721 XCHG  
12BC 3722 POP H ; GET THE RECORD NUMBER OF THE FILE  
12BD 3723 DAD D ; ADD THE RELATIVE TEST RECORD NO.  
12BE 3724 SHLD CURRENT_REC_NMB ; SAVE AS CURRENT RECORD NUMBER  
12C1 3725 STC  
12C2 3726 CMC  
12C3 3727 RET  
12C4 3728 ; EXIT WITH C BIT CLEAR
```

```
12C4 3730      .SBTTL      TYPE TEST NUMBER ROUTINE
12C4 3731      ;++
12C4 3732      ;
12C4 3733      ; FUNCTIONAL DESCRIPTION:
12C4 3734      ;
12C4 3735      ;     THIS ROUTINE IS USED TO TYPE THE TEST NUMBER OF THE TERMINAL. IT
12C4 3736      ;     IS CALLED BY THE READ OVERLAY ROUTINE.
12C4 3737      ;
12C4 3738      ;     IF IT'S THE FIRST CALL, THE LOST FLAG IS TESTED AND IF SET, THE
12C4 3739      ;     TEST NUMBER IS OBTAINED FROM THE USER SPECIFIED TEST NUMBER. IN ALL
12C4 3740      ;     OTHER CASES, THE TEST NUMBER IS OBTAINED FROM LOCATION 'TEST_NUMBER'.
12C4 3741      ;
12C4 3742      ;     IF THE TRACE (TR) FLAG IS SET, THE ENTIRE TEST TITLE IS PRINTED.
12C4 3743      ;
12C4 3744      ; CALLING SEQUENCE:
12C4 3745      ;
12C4 3746      ;     CALL      TYPE_TEST_NUMB
12C4 3747      ;
12C4 3748      ;     THE ROUTINE IS CALLED BY THE 'READ_OVERLAY' ROUTINE.
12C4 3749      ;
12C4 3750      ;     THE ROUTINE CALLS THE FOLLOWING ROUTINES:
12C4 3751      ;
12C4 3752      ;             TYPE_ASCII (TYPE)
12C4 3753      ;             TYPE_NUMERIC (TYPEB)
12C4 3754      ;
12C4 3755      ; IMPLICIT INPUTS:
12C4 3756      ;
12C4 3757      ;     TESTS_PER_LINE - CONTAINS THE NUMBER OF TEST NUMBERS TYPED ON THE
12C4 3758      ;     CURRENT LINE.
12C4 3759      ;     TEST_NUMBER   - CONTAINS THE NUMBER (MINUS ONE) OF THE TEST THAT
12C4 3760      ;     WAS JUST LOADED EXCEPT WHEN THE LOST FLAG IS SET.
12C4 3761      ;     SPEC_TEST_NUMB - CONTAINS THE FIRST TEST NUMBER THAT GETS EXECUTED
12C4 3762      ;     IF THE LOST FLAG IS SET.
12C4 3763      ;
12C4 3764      ; EXPLICIT INPUTS:
12C4 3765      ;
12C4 3766      ;     B - NON ZERO MEANS DON'T TYPE TEST NUMBER
12C4 3767      ;
12C4 3768      ; IMPLICIT OUTPUTS:
12C4 3769      ;
12C4 3770      ;     TESTS_PER_LINE - SAME AS ABOVE
12C4 3771      ;
12C4 3772      ;--
12C4 3773      ;
12C4 3774      ; TYPE_TEST_NUMB:
12C4 3775      ;
12C4 3776      ; CHECK IF THIS IS THE INITIAL CALL. IF SO, CHECK THE LOST FLAG. IF IT'S
12C4 3777      ; SET, GET THE TEST NUMBER FROM THE USER SPECIFIED TEST NUMBER.
12C4 3778      ;
12C4 3779      ;     LDA      TYPE_TEST_FLAG ; GET THE FLAG
12C7 3780      ;     ORA      A           ; SET THE CONDITION CODES
12C8 3781      ;     JNZ     10$        ; BRANCH IF NOT INITIAL CALL
12CB 3782      ;
12CB 3783      ; INITIAL CALL. CHECK IF THE LOST FLAG IS SET.
12CB 3784      ;
```

```

12CB 3785 LDA SOFT_CTRL_REG ; GET THE LOST FLAG
12CE 3786 ANI LOST_FLAG ; IS IT SET?
12D0 3787 JZ 1$ ; BRANCH IF NO
12D3 3788 :
12D3 3789 ; LOST FLAG IS SET. GET THE TEST NUMBER FROM THE USER SPECIFIED NUMBER
12D3 3790 :
12D3 3791 LDA SPEC_TEST_NUMB ; GET THE USER SPECIFIED TEST NUMBER
12D6 3792 JMP 20$ ; GO TYPE IT
12D9 3793 :
12D9 3794 ; LOST FLAG IS NOT SET. TYPE TEST NUMBER 1
12D9 3795 :
12D9 3796 1$: MVI A,1 ; SET TEST NUMBER TO 1
12DB 3797 JMP 20$ ; TYPE IT
12DE 3798 10$: LDA TEST_NUMBER ; GET THE LAST TEST NUMBER
12E1 3799 INR A ; SET TO NEW TEST NUMBER
12E2 3800 20$: STA ARG_LIST ; SAVE TEST NUMBER TO TYPE
12E5 3801 MOV A,B ; GET TIMEOUT FLAG
12E6 3802 ORA A ; SET CONDITION CODES
12E7 3803 JZ 21$ ; BRANCH IF GOING TO TYPE
12EA 3804 LDA TESTS_PER_LINE ; SETUP TESTS PER LINE SO
12ED 3805 DCR A ; IT DOESN'T CHANGE
12EE 3806 STA TESTS_PER_LINE ; INIT TESTS PER LINE
12F1 3807 JMP 17$ ; EXIT
12F4 3808 21$: LDA PROG_CTRL_REG ; GET THE TRACE FLAG
12F7 3809 ANI TR_FLAG ;
12F9 3810 JZ 30$ ; BRANCH IF ITS CLEAR
12FC 3811 TYPE TEST_MSG_1 ; TYPE 'TEST '
1305 3812 TYPEB ARG_LIST ; TYPE THE TEST NUMBER
1310 3813 TYPE TEST_MSG_2 ; TYPE ': '
1319 3814 LHLD INIT_TEST_PC ; GET POINTER TO TEST TITLE
131C 3815 SHLD MESSAGE_POINTER ; PASS TO TYPE ASCII ROUTINE
131F 3816 CALL TYPE_ASCII ; TYPE THE TEST TITLE
1322 3817 TYPE CRLF ; ...
132B 3818 :
132B 3819 ; UPDATE THE INITIAL TEST PC AND COMPLIMENT INITIAL TEST PC
132B 3820 :
132B 3821 17$: LHLD INIT_TEST_PC ; GET POINTER TO TEST TITLE
132E 3822 MOV L,M ; GET THE LENGTH OF THE STRING
132F 3823 INR L ; COUNT THE COUNT BYTE
1330 3824 MOV C,L ; SAVE IN C
1331 3825 MVI H,0 ;
1333 3826 SHLD TITLE_LENGTH ; SAVE
1336 3827 XCHG
1337 3828 LHLD INIT_TEST_PC
133A 3829 DAD D ; ADD TO INITIAL TEST PC
133B 3830 SHLD INIT_TEST_PC ;
133E 3831 XRA A ; INIT A REG
133F 3832 SUB C ; GET TWO'S COMPLIMENT OF TITLE
1340 3833 MOV C,A ; LENGTH
1341 3834 MVI B,-1 ;
1343 3835 LHLD C_INIT_TEST_PC ; GET COMPLIMENT INIT TEST PC
1346 3836 DAD B ; ADJUST BY TITLE LENGTH
1347 3837 SHLD C_INIT_TEST_PC ; ...
134A 3838 :
134A 3839 ; INCREMENT THE TESTS PER LINE AND SET THE TYPE_TEST_FLAG

```

```
134A 3840 ;
134A 3841 LDA TESTS_PER_LINE ; GET NUMBER OF TESTS PER LINE
134D 3842 INR A ; INCREMENT IT
134E 3843 STA TESTS_PER_LINE ; SAVE NEW VALUE OF TESTS ON LINE
1351 3844 MVI A,1 ; SET THE TYPE FLAG
1353 3845 STA TYPE_TEST_FLAG ; ...
1356 3846 RET ; EXIT
1357 3847 ;
1357 3848 ; TYPE THE TEST NUMBER THAT IS IN THE ARG_LIST.
1357 3849 ;
1357 3850 30$: LDA TESTS_PER_LINE ; GET NUMBER OF TEST NUMBERS TYPED
135A 3851 CPI 20 ; TYPED 20 TESTS YET?
135C 3852 JNZ 25$ ; BRANCH IF NO
135F 3853 ;
135F 3854 ; TIME TO TYPE A CARRIAGE RETURN LINE FEED
135F 3855 ;
135F 3856 TYPE CRLF ; TYPE CARRIAGE RETURN LINE FEED
1368 3857 XRA A ; RESET THE COUNT
1369 3858 STA TESTS_PER_LINE ; ...
136C 3859 ;
136C 3860 ; TYPE THE TEST NUMBER FOLLOWED BY A COMMA
136C 3861 ;
136C 3862 25$: TYPEB ARG_LIST ; TYPE THE NUMBER
1377 3863 TYPE COMMA_MSG ; TYPE A COMMA
1380 3864 JMP 17$ ; ...
1383 3865
```

```
1383 3867      .SBTTL      DESELECT DCS ROUTINE
1383 3868      ;++
1383 3869      ;
1383 3870      ; FUNCTIONAL DESCRIPTION:
1383 3871      ;
1383 3872      ;     THIS ROUTINE DESELECTS DCS BY SETTING PARITY CHECK AND MICRO ADDRESS
1383 3873      ;     INHIBIT AND LOADING THE MATCH REGISTER WITH 0
1383 3874      ;
1383 3875      ; CALLING_SEQUENCE:
1383 3876      ;
1383 3877      ;     CALL      DESELECT_DCS
1383 3878      ;
1383 3879      ; IMPLICIT INPUTS:
1383 3880      ;
1383 3881      ;     NONE
1383 3882      ;
1383 3883      ; IMPLICIT OUTPUTS:
1383 3884      ;
1383 3885      ;     NONE
1383 3886      ;
1383 3887      ;--
1383 3888
1383 3889 DESELECT_DCS:
1383 3890     LDA      DCS_CTRL_RE_CPY ; GET A COPY OF THE CONTROL REG
1386 3891     ORI      <PAR_CHK_ENABL!- ; SET PARITY CHECK AND MICRO
1386 3892     MICRO_ADDR_INH> ; ADDRESS INHIBIT
1388 3893     STA      DCS_CTRL_REG      ; LOAD THE CONTROL REGISTER
1388 3894     LXI      H,0              ; PUT ADDRESS 0 IN THE MATCH
138E 3895     SHLD   ADDR_MATCH_LO     ; REGISTER
1391 3896     RET
1392 3897
```

```
1392 3899 .SBTTL SELECT DCS
1392 3900 ;++
1392 3901 ;
1392 3902 ; FUNCTIONAL DESCRIPTION:
1392 3903 ;
1392 3904 ; THIS ROUTINE RESTORES THE MATCH REGISTER TO THE SOMM ADDRESS AND
1392 3905 ; CLEARS THE PARITY CHECK AND MICRO ADDRESS INHIBIT BITS IN THE
1392 3906 ; CONTROL REGISTER.
1392 3907 ;
1392 3908 ; CALLING SEQUENCE:
1392 3909 ;
1392 3910 ; CALL SELECT_DCS
1392 3911 ;
1392 3912 ;--
1392 3913
1392 3914 SELECT_DCS:
1392 3915 LDA DCS_CTRL_RE_CPY ; GET A COPY OF THE CONTROL REG
1395 3916 STA DCS_CTRL_REG ; LOAD THE CONTROL REGISTER
1398 3917 LHLD SOMM_ADDRESS ; GET THE SOMM ADDRESS
139B 3918 SHLD ADDR_MATCH_LO ; RESTORE THE MATCH REGISTER
139E 3919 RET
139F 3920
```

```

139F 3922      .SBTTL      START DCS EXECUTION ROUTINE
139F 3923      ;++
139F 3924      ;
139F 3925      ; FUNCTIONAL DESCRIPTION:
139F 3926      ;
139F 3927      ;     THIS ROUTINE LOADS THE CONTROL STORE LATCHES WITH THE DCS MICRO
139F 3928      ;     WORD AT THE SPECIFIED DCS ADDRESS.
139F 3929      ;
139F 3930      ; CALLING SEQUENCE:
139F 3931      ;
139F 3932      ;     CALL      START_EXECUTION
139F 3933      ;
139F 3934      ; IMPLICIT INPUTS:
139F 3935      ;
139F 3936      ;     NONE
139F 3937      ;
139F 3938      ; INPUT PARAMETERS:
139F 3939      ;
139F 3940      ;     A REGISTER - CONTAINS THE DCS ADDRESS OF THE MICRO WORD
139F 3941      ;
139F 3942      ; OUTPUT PARAMETERS:
139F 3943      ;
139F 3944      ;     NONE
139F 3945      ;
139F 3946      ;--
139F 3947
139F 3948      START_EXECUTION:
139F 3949      ;
139F 3950      ; SET THE 'PARITY CHECK', 'MICRO ADDRESS INHIBIT', AND 'CLEAR CONTROL FILE'
139F 3951      ; BITS IN THE DCS CONTROL REGISTER.
139F 3952      ;
139F 3953      STA      DCS_ADDR_REG_WO ; LOAD THE DCS ADDRESS REGISTER
13A2 3954
13A2 3955      LDA      DCS_CTRL_RE_CPY ; GET CONTENTS OF DCS CONTROL REG
13A5 3956      ORI      <PAR_CHK_ENABL!-; SET THE PARITY CHECK AND MICRO
13A5 3957      MICRO_ADDR_INH!-; ADDRESS INHIBIT AND CLEAR THE
13A5 3958      CLEAR_CTRL_FILE>; CONTROL FILE
13A7 3959      STA      DCS_CTRL_REG      ; LOAD THE CONTROL REGISTER
13AA 3960      ANI      <^CCLEAR_CTRL_FILE>; DEASCERT CLEAR CONTROL FILE
13AC 3961      STA      DCS_CTRL_REG      ;
13AF 3962      STA      DCS_CTRL_RE_CPY ; KEEP OTHER BITS ASCERTED WHEN CLOCK
13B2 3963      ; IS TICKED
13B2 3964      ;
13B2 3965      ; TICK THE CLOCK TO LATCH THE SPECIFIED MICRO WORD
13B2 3966      ;
13B2 3967      PUSH     H                ; SAVE H & L
13B3 3968      MVI     L,<DCS_START&255> ; GET THE DCS BASE ADDRESS
13B5 3969      MVI     H,<DCS_START/256> ; ...
13B7 3970      SHLD   ADDR_MATCH_LO    ; LOAD THE ADDRESS MATCH REGISTER
13BA 3971      SGL_MIC_INSTR 1        ; STEP THE CLOCK
13C2 3972      ;
13C2 3973      ; NOW DEASCERT 'PARITY CHECK' AND 'MICRO ADDRESS INHIBIT'
13C2 3974      ;
13C2 3975      LDA      DCS_CTRL_RE_CPY ; GET THE CONTROL REGISTER
13C5 3976      ANI      <^C<PAR_CHK_ENABL!-; CLEAR PARITY CHECK ENABLE

```

ZZ-ECKAA-8.7 V08.07
ECKAA
V08.07

D 9

" START DCS EXE11-FEB-1986 Fiche 1 Frame D9 Sequence 107
VAX-11/750 MICRO DIAGNOSTIC MONITOR 11-FEB-1986 10:07:52 VAX/VMS Macro V04-00 Page 82
START DCS EXECUTION ROUTINE 11-FEB-1986 10:06:37 [VAX750.MONITOR]ECKAA.MAR;811 (27)

13C5	3977		MICRO_ADDR_INH>>; AND MICRO ADDRESS INHIBIT
13C7	3978	STA	DCS_CTRL_REG ; LOAD THE CONTROL REGISTER
13CA	3979	STA	DCS_CTRL_RE_CPY ; SAVE A COPY
13CD	3980	:	
13CD	3981	:	LOAD THE SOMM ADDRESS IN THE MATCH REGISTER
13CD	3982	:	
13CD	3983	LHLD	SOMM_ADDRESS ; GET THE SOMM ADDRESS
13D0	3984	SHLD	ADDR_MATCH_LO ; LOAD THE MATCH REGISTER
13D3	3985	POP	H ; RESTORE H & L
13D4	3986	RET	; EXIT
13D5	3987		

```
13D5 3989      .SBTTL      CMI DISABLE ROUTINE
13D5 3990      ;++
13D5 3991      ;
13D5 3992      ; FUNCTIONAL DESCRIPTION:
13D5 3993      ;
13D5 3994      ;     THIS ROUTINE LOADS AND EXECUTES MICRO CODE TO DISABLE THE CMI AND
13D5 3995      ;     MEMORY MANAGEMENT.
13D5 3996      ;
13D5 3997      ; CALLING SEQUENCE:
13D5 3998      ;
13D5 3999      ;     CALL      DISABLE_CMI
13D5 4000      ;
13D5 4001      ;--
13D5 4002
13D5 4003  DISABLE_CMI:
13D5 4004      PUSH      B          ; SAVE THE REGISTERS
13D6 4005      PUSH      D          ; ...
13D7 4006      PUSH      H          ; ...
13D8 4007      PUSH      $PSW       ; ...
13D9 4008      WRITE$ _DCS        DCS_SCRATCH_ADR,-
13D9 4009      LOAD  MEMSCAR,7 ; LOAD THE MICRO CODE
13EC 4010      WRITE$ _DCS        DCS_SCRATCH_ADR+7,-
13EC 4011      NOP  MICRO_WORD,1 ; ...
13FF 4012      EXECUTE   DCS_SCRATCH_ADR,7 ; EXECUTE THE MICRO CODE
1412 4013      POP      $PSW       ; RESTORE THE REGISTERS
1413 4014      POP      H          ; ...
1414 4015      POP      D          ; ...
1415 4016      POP      B          ; ...
1416 4017      RET
1417 4018
```

```
1417 4020      .SBTTL " CMI ENABLE ROUTINE
1417 4021      ;++
1417 4022      ;
1417 4023      ; FUNCTIONAL DESCRIPTION:
1417 4024      ;
1417 4025      ;     THIS ROUTINE LOADS AND EXECUTES MICRO CODE TO ENABLE THE CMI
1417 4026      ;     BY EXECUTING A PROCESSOR INIT BUS FUNCTION.
1417 4027      ;
1417 4028      ; CALLING SEQUENCE:
1417 4029      ;
1417 4030      ;     CALL     ENABLE_CMI
1417 4031      ;
1417 4032      ;--
1417 4033
1417 4034      ENABLE_CMI:
1417 4035          PUSH     B           ; SAVE THE REGISTERS
1418 4036          PUSH     D           ; ...
1419 4037          PUSH     H           ; ...
141A 4038          PUSH     $PSW      ; ...
141B 4039          WRITE$ _DCS      DCS_SCRATCH_ADR,-
141B 4040          LOAD  MEMSCAR,1 ; LOAD THE MICRO CODE
142E 4041          WRITE$ _DCS      DCS_SCRATCH_ADR+1,-
142E 4042          NOP  MICRO_WORD,1 ; ...
1441 4043          EXECUTE  DCS_SCRATCH_ADR,1 ; EXECUTE THE MICRO CODE
1454 4044          POP     $PSW      ; RESTORE THE REGISTERS
1455 4045          POP     H           ; ...
1456 4046          POP     D           ; ...
1457 4047          POP     B           ; ...
1458 4048          RET              ; EXIT
1459 4049
```

```
1459 4051      .SBTTL      INIT CPU ROUTINE
1459 4052      ;++
1459 4053      ;
1459 4054      ; FUNCTIONAL DESCRIPTION:
1459 4055      ;
1459 4056      ;       THIS ROUTINE SETS THEN CLEARS DC LOW IN THE CPU AND THEN FETCHES
1459 4057      ;       LOCATION 0 IN THE CPU CONTROL STORE.
1459 4058      ;
1459 4059      ; CALLING SEQUENCE:
1459 4060      ;
1459 4061      ;       CALL      CPU_INIT
1459 4062      ;
1459 4063      ;--
1459 4064
1459 4065 CPU_INIT:
1459 4066      MVI          A,<MAINT TRAP OFF!-
1459 4067      MAINT_DCS_ENABL>; GET DATA TO INIT THE MAINTENANCE REG
1459 4068      STA          MAINT_REG      ; INIT THE MAINTENANCE REG
145E 4069      LDA          SOFT_CTRL_REGA ; GET THE SOMM FLAG
1461 4070      ANI          SOMM_FLAG
1463 4071      ORI          <CLK_CTRL_0!- ; GET DATA TO INIT THE DCS CONTROL REG
1463 4072      CLK_CTRL_1>
1465 4073      STA          DCS_CTRL_REG      ; INIT THE CONTROL REG
1468 4074      STA          DCS_CTRL_RE_CPY ; SAVE A COPY
146B 4075      LDA          FRONT_PNL_2 ; GET REMOTE, FAULT, AND TEST BITS
146E 4076      ANI          <REMOTE!FAULT!TEST>;
1470 4077      ORI          DC_LOW      ; SET THE DC LOW BIT
1472 4078      STA          RD_CTRL_REG
1475 4079      XRI          DC_LOW      ; CLEAR DC LOW BIT
1477 4080      STA          RD_CTRL_REG      ; ...
147A 4081      RET
147B 4082
```

```
147B 4084 .SBTTL FETCH CS LOCATION 0
147B 4085 ;**
147B 4086 ;
147B 4087 ; FUNCTIONAL DESCRIPTION:
147B 4088 ;
147B 4089 ; THIS ROUTINE FETCHES LOCATION 0 OF THE CPU CONTROL STORE
147B 4090 ;
147B 4091 ;--
147B 4092
147B 4093 FETCH_0:
147B 4094 CALL DESELECT_DCS ; FORCE CS ADDRESS ZERO
147E 4095 LDA DCS_CTRL_RE_CPY ; SET PARITY CHECK AND UADDR INH
1481 4096 ORI <PAR_CHK_ENABL!- ; ...
1481 4097 MICRO_ADDR_INH> ; ...
1483 4098 STA DCS_CTRL_RE_CPY ; ...
1486 4099 SGL_MIC_INSTR 1 ; TICK THE CLOCK
148E 4100 LDA DCS_CTRL_RE_CPY ; DEASCERT PARITY CHECK AND MICRO
1491 4101 ANI <^C<PAR_CHK_ENABL!- ; ADDRESS INHIBIT
1491 4102 MICRO_ADDR_INH>> ; ...
1493 4103 STA DCS_CTRL_RE_CPY ; ...
1496 4104 STA DCS_CTRL_REG ; ...
1499 4105 RET
149A 4106
```

```
149A 4108 .SBTTL TYPE FLAGS ROUTINE
149A 4109 ;++
149A 4110 ;
149A 4111 ; FUNCTIONAL DESCRIPTION:
149A 4112 ;
149A 4113 ; THIS ROUTINE IS CALLED WITH THE PROGRAM CONTROL REGISTER BIT MASK
149A 4114 ; IN THE A REGISTER. IT CONVERTS THE BITS INTO THE FLAG MNEMONICS
149A 4115 ; AND TYPES THEM.
149A 4116 ;--
149A 4117
149A 4118 TYPE_FLAGS:
149A 4119 ORA A ; ANY FLAGS TO TYPE?
149B 4120 JZ 20$ ; BRANCH IF NO
149E 4121 MVI B,8 ; SET THE LOOP COUNT FOR 7 FLAGS
14A0 4122 LXI H,HALT_FLAG_NAME ; GET ADDRESS OF ASCII STRING
14A3 4123 1$: RAR ; SEE IF BIT IS SET
14A4 4124 JNC 10$ ; BRANCH IF CLEAR
14A7 4125 PUSH B ; SAVE PARAMETERS
14A8 4126 PUSH H ; ...
14A9 4127 PUSH $PSW ; ...
14AA 4128 TYPE_UNCOUNTED 2 ; TYPE THE FLAG NAME
14B5 4129 TYPE COMMA_MSG ; TYPE A COMMA
14BE 4130 POP $PSW ; RESTORE THE PARAMTERS
14BF 4131 POP H ; ...
14C0 4132 POP B ; ...
14C1 4133 10$: INX H ; POINT AT NEXT FLAG NAME
14C2 4134 INX H ; ...
14C3 4135 DCR B ; CHECKED ALL 7 BITS?
14C4 4136 JNZ 1$ ; BRANCH IF NO
14C7 4137 20$: TYPE CRLF ; TERMINATE THE LINE
14D0 4138 RET ; EXIT
```

```
14D1 4140 .SBTTL " FILL DCS ROUTINE
14D1 4141 ;++
14D1 4142 ;
14D1 4143 ; FUNCTIONAL DESCRIPTION:
14D1 4144 ;
14D1 4145 ; THIS ROUTINE IS USED TO FILL DCS WITH 'NOP' MICRO INSTRUCTIONS.
14D1 4146 ;
14D1 4147 ; CALLING SEQUENCE:
14D1 4148 ;
14D1 4149 ; CALL FILL_DCS
14D1 4150 ;
14D1 4151 ;--
14D1 4152 FILL_DCS:
14D1 4153 MVI B,64 ; SET THE LOOP COUNT
14D3 4154 MVI C,0 ; INIT THE DCS ADDRESS TO LOAD AT
14D5 4155 1$: PUSH B ; SAVE THE LOOP COUNT
14D6 4156 MOV A,C ; PUT DCS ADDRESS IN A REG
14D7 4157 WRITE$ _DCS ,NOP_MICRO_WORD,1 ; LOAD A MICRO WORD
14E8 4158 POP B ; GET THE LOOP COUNT
14E9 4159 INR C ; INCREMENT THE DCS ADDRESS
14EA 4160 DCR B ; LOADED 64 WORDS YET?
14EB 4161 JNZ 1$ ; BRANCH IF NO
14EE 4162 RET ; EXIT
```

```

14EF 4164      .SBTTL  'COMMAND PARSE
14EF 4165      .SBTTL  ' GET COMMAND LINE
14EF 4166      ;++
14EF 4167      ;
14CF 4168      ; FUNCTIONAL DESCRIPTION:
14EF 4169      ;
14EF 4170      ; THIS ROUTINE INITIALIZES THE CONTROL C FLAG, THE TERMINAL DRIVER,
14EF 4171      ; TYPES THE DIAGNOSTIC PROMPT, AND CALLS THE RDM TERMINAL DRIVER
14EF 4172      ; TO PERFORM A READ. WHEN THE DRIVER RETURNS, ERRORS ARE CHECKED.
14EF 4173      ; IF A CONTROL C WAS TYPED, THE ROUTINE RESTARTS. IF ANY OTHER ERROR
14EF 4174      ; WAS DETECTED, AN ERROR MESSAGE IS TYPED WITH THE ERROR CODE AND THE
14EF 4175      ; ROUTINE RESTARTS. IF NO ERROR WAS DETECTED, THE PARSE ROUTINE IS
14EF 4176      ; CALLED.
14EF 4177      ;
14EF 4178      ; IF THE PARSE ROUTINE DOES NOT FIND ANY ERRORS IN THE COMMAND LINE,
14EF 4179      ; THE APPROPRIATE COMMAND ACTION ROUTINE IS CALLED.
14EF 4180      ;
14EF 4181      ; IF THE COMMAND WAS A 'CONTINUE' COMMAND, RETURN IS MADE TO THE
14EF 4182      ; CALLING SEQUENCE. IF THE COMMAND WAS A 'DIAGNOSE' COMMAND, RETURN
14EF 4183      ; IS MADE TO THE 'INTERPRETER'. ANY OTHER COMMAND (EXCEPT 'RETURN')
14EF 4184      ; CAUSES THIS ROUTINE TO RESTART.
14EF 4185      ;
14EF 4186      ; CALLING SEQUENCE:
14EF 4187      ;
14EF 4188      ; CALL    GET_CMD_LINE
14EF 4189      ;
14EF 4190      ; IMPLICIT INPUTS:
14EF 4191      ;
14EF 4192      ; SOFT_CTRL_REG - CONTAINS THE SOFTWARE CONTROL FLAGS.
14EF 4193      ;
14EF 4194      ; IMPLICIT OUTPUTS:
14EF 4195      ;
14EF 4196      ; REFER TO THE INDIVIDUAL COMMAND EXECUTION ROUTINES.
14EF 4197      ;
14EF 4198      ;--
14EF 4199      ;
14EF 4200      GET_CMD_LINE:
14EF 4201          LDA      SOFT_CTRL_REG      ; GET THE SWITCH REGISTER
14F2 4202          ANI      <^C<CONTROL_C_FLAG!- ; CLEAR SOME FLAGS
14F2 4203          DIAGNOSE_FLAG!CONTINUE_FLAG>> ; ...
14F4 4204          STA      SOFT_CTRL_REG
14F7 4205          XRA      A                  ; INITIALIZE THE BELL COUNT
14F8 4206          STA      BELL_COUNT        ; FOR THE RING_BELL ROUTINE
14FB 4207          STA      TESTS_PER_LINE
14FE 4208      1$:   $TERM_INIT          ; INIT THE TERMINAL DRIVER
1503 4209          TYPE     CRLF              ; TYPE A CARRIAGE RETURN LINE FEED
150C 4210          TYPE     PROMPT          ; TYPE THE PROMPT
1515 4211          $TERM_READ          ; GET A COMMAND LINE
1515 4212          CMD_LINE_LENGTH
1520 4213          JNC      2$              ; BRANCH IF NO ERROR
1523 4214          CPI      $CONTROL_C_CODE ; WAS IT A CONTROL C?
1525 4215          JZ       1$              ; BRANCH IF YES
1528 4216          STA      TERM_ERR_CODE   ; SAVE THE ERROR CODE
152B 4217          TYPE     CRLF              ; TYPE A CARRIAGE RETURN LINE FEED
1534 4218          TYPE     TERM_ERROR        ; TYPE THE TERMINAL ERROR MSG

```

```

153D 4219          TYPEB          TERM_ERR_CODE      ; TYPE THE ERROR CODE
1548 4220          JMP              1$                ; RESTART THIS ROUTINE
154B 4221
154B 4222          ;
154B 4223          ; GOT A COMMAND LINE. CALL THE PARSER.
154B 4224          ;
154B 4225
154B 4226 2$:     LDA              TERM_INP_BUFF+1    ; GET FIRST CHARACTER
154E 4227          CPI              <^XD>           ; IS IT A CARRIAGE RETURN?
1550 4228          JZ              1$                ; BRANCH IF YES
1553 4229          CALL             COMMAND_PARSE    ; PARSE THE COMMAND LINE
1556 4230          JNC             3$                ; BRANCH IF NO ERROR
1559 4231 5$:     TYPE             INVALID_COMMAND  ; TYPE AN ERROR MESSAGE
1562 4232          JMP              1$                ; RESTART THIS ROUTINE
1565 4233          ;
1565 4234          ; COMMAND PARSE WAS SUCCESSFUL. CALL THE COMMAND ACTION ROUTINE.
1565 4235          ;
1565 4236
1565 4237 3$:     LDA              PARSE_TABLE        ; GET THE COMMAND CODE IN 'A' REG
1568 4238          ANI              <^X7F>          ; THROW AWAY KEYWORD FLAG BIT
156A 4239          MOV              L,A              ; SAVE IN 'L' REG
156B 4240          MVI              H,0             ; CLEAR THE 'H' REG
156D 4241          LXI              B,CMD_DISPATCH  ; GET ADDRESS OF COMMAND DISPATCH
1570 4242          ;
1570 4243          DAD              B                ; GET POINTER TO COMMAND ACTION
1571 4244          ;
1571 4245          XCHG                     ; PUT IN D & E
1572 4246          LDAX             D                ; GET LOW PART OF COMMAND ACTION RTN
1573 4247          MOV              L,A              ; ADDRESS AND PUT IN 'L' REG
1574 4248          INX                     ;
1575 4249          LDAX             D                ; GET HIGH PART AND PUT IN
1576 4250          MOV              H,A              ; H REGISTER
1577 4251          LXI              B,4$            ; GET A RETURN ADDRESS FOR THE
157A 4252          ;
157A 4253          PUSH             B                ; PUT IT ON THE STACK
157B 4254          LXI              D,PARSE_TABLE+1  ; GET ADDRESS OF PARSE TABLE
157E 4255          PCHL                     ; GO TO THE COMMAND ACTION ROUTINE
157F 4256          ;
157F 4257          ; COMMAND ACTION ROUTINES WILL RETURN HERE.
157F 4258          ;
157F 4259 4$:     JC              5$                ; BRANCH IF ERROR ON COMMAND EXEC.
1582 4260          ;
1582 4261          ; SEE WHAT KIND OF COMMAND IT WAS. IF 'CONTINUE', RETURN TO CALLING
1582 4262          ; SEQUENCE. IF 'DIAGNOSE', RETURN TO THE 'PROGRAM INIT'. ELSE RESTART
1582 4263          ; THIS ROUTINE.
1582 4264          ;
1582 4265          LXI              H,SOFT_CTRL_REG    ; GET ADDRESS OF SOFTWARE CTRL REG
1585 4266          MOV              A,M              ; GET CONTROL FLAGS
1586 4267          ANI              CONTINUE_FLAG    ; CONTINUE FLAG SET?
1588 4268          JNZ             7$                ; BRANCH IF YES
158B 4269 6$:     MOV              A,M              ; GET CONTROL FLAGS AGAIN
158C 4270          ANI              DIAGNOSE_FLAG    ; DIAGNOSE FLAG SET?
158E 4271          JZ              1$                ; BRANCH IF NO
1591 4272          LXI              H,PROGRAM_INIT   ; GET ADDRESS OF INIT ROUTINE
1594 4273          XTHL                     ; PUT IT ON THE STACK

```

ZZ-ECKAA-8.7 V08.07
ECKAA
V08.07

M 9
" GET COMMAND L11-FEB-1986
VAX-11/750 MICRO DIAGNOSTIC MONITOR
GET COMMAND LINE

Fiche 1 Frame M9 Sequence 116
11-FEB-1986 10:07:52 VAX/VMS Macro V04-00 Page 91
11-FEB-1986 10:06:37 [VAX750.MONITOR]ECKAA.MAR;811 (34)

1595	4274	7\$:	\$TERM_READ	TERM_INP_BUFF,- ; QUE A TERMINAL READ REQUEST
1595	4275			ONE,TERM_INTERRUPT ;
15A0	4276		RET	; GO TO APPROPRIATE ROUTINE
15A1	4277			
15A1	4278			
15A1	4279	BREAK_ACTION:		
15A1	4280		CALL	CMD_ENTRY
15A4	4281		POP	H ; DICARD RETURN PC
15A5	4282		JMP	GET_CMD_LINE
15A8	4283			
15A8	4284			

```

15A8 4286      .SBTTL      PARSE COMMAND LINE
15A8 4287      ;++
15A8 4288      ;
15A8 4289      ; FUNCTIONAL DESCRIPTION:
15A8 4290      ;
15A8 4291      ;     THIS ROUTINE IS RESPONSIBLE FOR PARSING COMMAND LINES INTO
15A8 4292      ;     NUMERIC CODES THAT ARE PLACED IN THE PARSE TABLE.
15A8 4293      ;
15A8 4294      ; IMPLICIT INPUTS:
15A8 4295      ;
15A8 4296      ;     TERM_INP_BUFF - BUFFER THAT CONTAINS THE ASCII COMMAND LINE
15A8 4297      ;
15A8 4298      ; IMPLICIT OUTPUTS:
15A8 4299      ;
15A8 4300      ;     PARSE_TABLE - CONTAINS THE NUMERIC CODES OF THE PARSED COMMAND LINE
15A8 4301      ;
15A8 4302      ; COMPLETION CODES:
15A8 4303      ;     C BIT SET - THE COMMAND LINE DID NOT PARSE
15A8 4304      ;     C BIT CLEAR - THE PARSE WAS SUCCESSFUL
15A8 4305      ;
15A8 4306      ;--
15A8 4307      ;
15A8 4308      COMMAND_PARSE:
15A8 4309      LXI          H,PARSE_TABLE      ; GET THE ADDRESS OF THE PARSE TABLE
15A8 4310      SHLD         PARSE_TBL_PTR    ; INIT THE PARSE TABLE POINTER
15A8 4311      LXI          H,TERM_INP_BUFF   ; GET THE ADDRESS OF THE INPUT STRING
15A8 4312      INX          H                ; POINT AT FIRST CHARACTER
15A8 4313      LXI          B,COMMAND_LIST   ; GET THE ADDRESS OF THE COMMAND LIST
15A8 4314      ;
15A8 4315      ; NOW SEARCH THE COMMAND LIST FOR A MATCH
15A8 4316      ;
15A8 4317      CALL         SEARCH_LIST      ; SEARCH THE COMMAND LIST
15A8 4318      JC          PARSE_ERR        ; IF NOT IN LIST, EXIT WITH C SET
15A8 4319      MOV          A,D              ; GET THE CODE
15A8 4320      ORI          <^X80>        ; INSERT FLAG BIT
15A8 4321      MOV          D,A              ; PUT BACK IN D REG
15A8 4322      CALL         PUT_D_IN_TBL    ; PUT COMMAND CODE IN PARSE TABLE
15A8 4323      ;
15A8 4324      ; THROW AWAY SPACES
15A8 4325      ;
15A8 4326      CALL         SKIP_SPACES     ; BUMP STRING PTR PAST SPACES
15A8 4327      JNC         1$              ; BRANCH IF CARRIAGE RETURN NOT FOUND
15A8 4328      2$:      MVI          D,<^X55> ; PUT PARSE TABLE TERMINATOR IN THE
15A8 4329      CALL         PUT_D_IN_TBL    ; PARSE TABLE
15A8 4330      JMP          PARSE_DONE     ; EXIT WITH C BIT CLEAR
15A8 4331      ;
15A8 4332      ; NOW SEARCH THE KEYWORD LIST FOR A MATCH
15A8 4333      ;
15A8 4334      1$:      LXI          B,KEYWORD_LIST ; GET ADDRESS OF LIST
15A8 4335      CALL         SEARCH_LIST      ; SEARCH THE KEYWORD LIST
15A8 4336      JC          PARSE_ERR        ; IF NOT IN LIST, EXIT WITH C SET
15A8 4337      MOV          A,D              ; GET THE CODE
15A8 4338      ORI          <^X80>        ; INSERT KEYWORD CODE BIT
15A8 4339      MOV          D,A              ; RETURN TO 'D' REG
15A8 4340      CALL         PUT_D_IN_TBL    ; PUT THE KEYWORD CODE IN PARSE TABLE

```

```

15E0 4341 ;
15E0 4342 ; THROW AWAY SPACES
15E0 4343 ;
15E0 4344      MOV      A,M      ; GET THE NEXT CHARACTER IN STRING
15E1 4345      CPI      <^X3A>   ; IS CHARACTER A ':' ?
15E3 4346      JZ       4$      ; BRANCH IF YES
15E6 4347      CALL     SKIP_SPACES ; BUMP STRING PTR PAST SPACES
15E9 4348      JC       2$      ; EXIT IF CARRIAGE RETURN FOUND
15EC 4349      MOV      A,M      ; GET THE NEXT CHARACTER IN STRING
15ED 4350      CPI      <^X3A>   ; IS CHARACTER A ':' ?
15EF 4351      JNZ     1$      ; IF NOT, SEARCH FOR A KEYWORD
15F2 4352 ;
15F2 4353 ; FOUND THE START OF A COLON LIST. CHECK IF A NUMBER FOLLOWS.
15F2 4354 ;
15F2 4355 4$:   INX      H      ; BUMP STRING PTR PAST THE ':'
15F3 4356      MOV      A,M      ; GET THE CHARACTER
15F4 4357      CPI      <^X20>   ; IS IT A SPACE?
15F6 4358      JNZ     3$      ; BRANCH IF NO
15F9 4359      CALL     SKIP_SPACES ; SKIP OVER ANY SPACES
15FC 4360      JC       2$      ; EXIT IF CARRIAGE RETURN FOUND
15FF 4361 3$:   CPI      <^XD>   ; IS IT A CARRIAGE RETURN?
1601 4362      JZ       PARSE_ERR ; BRANCH IF YES
1604 4363      CALL     PARSE_HEX_NUMB ; PARSE THE HEX NUMBER
1607 4364      JC       PARSE_ERR  ; IF ILLEGAL NO., EXIT WITH C SET
160A 4365 ;
160A 4366 ; THROW AWAY SPACES
160A 4367 ;
160A 4368 5$:   CALL     SKIP_SPACES ; BUMP STRING POINTER PAST SPACES
160D 4369      JC       2$      ; EXIT IF CARRIAGE RETURN FOUND
1610 4370 ;
1610 4371 ; NEXT LEXIME COULD BE ANOTHER NUMERIC OR ANOTHER KEYWORD. USE PARSE_HEX_NUMB
1610 4372 ; ROUTINE TO SEE IF IT'S A NUMERIC. IF NOT NUMERIC, RESTORE INPUT BUFFER
1610 4373 ; POINTER AND SEE IF IT'S A KEYWORD.
1610 4374 ;
1610 4375      PUSH     H      ; SAVE INPUT STRING POINTER
1611 4376      CALL     PARSE_HEX_NUMB ; SEE IF LEXIME IS NUMERIC
1614 4377      POP      D      ; GET SAVED H AND L BACK
1615 4378      JNC     5$      ; BRANCH IF IT WAS A NUMBER
1618 4379      XCHG    ; RESTORE INPUT STRING POINTER
1619 4380      JMP     1$      ; SEE IF IT'S A KEYWORD
161C 4381 ;
161C 4382 ; PARSE ERROR. RETURN TO CALLER WITH C BIT SET
161C 4383 ;
161C 4384 PARSE_ERR:
161C 4385      STC      ; SET THE CARRY BIT
161D 4386      RET     ; RETURN TO CALLER
161E 4387 ;
161E 4388 ; PARSE DONE. EXIT WITH C BIT CLEAR
161E 4389 ;
161E 4390 PARSE_DONE:
161E 4391      STC      ; SET THE CARRY BIT
161F 4392      CMC      ; CLEAR THE CARRY BIT
1620 4393      RET     ; RETURN TO CALLER
1621 4394
1621 4395

```

```

1621 4397 .SBTTL " SEARCH COMMAND/KEYWORD LIST ROUTINE
1621 4398 ;++
1621 4399 ;
1621 4400 ; FUNCTIONAL DESCRIPTION:
1621 4401 ;
1621 4402 ; THIS ROUTINE SEARCHES THE COMMAND LIST OR A KEYWORD LIST FOR
1621 4403 ; A MATCH ON THE CURRENT TWO CHARACTERS OF THE INPUT COMMAND
1621 4404 ; STRING.
1621 4405 ;
1621 4406 ; CALLING SEQUENCE:
1621 4407 ;
1621 4408 ; CALL SEARCH_LIST
1621 4409 ;
1621 4410 ; INPUT PARAMETERS:
1621 4411 ;
1621 4412 ; REGISTER PAIR H & L - CONTAIN THE ADDRESS OF THE INPUT STRING.
1621 4413 ; REGISTER PAIR B & C - CONTAIN THE ADDRESS OF THE SEARCH LIST.
1621 4414 ;
1621 4415 ; IMPLICIT INPUTS:
1621 4416 ;
1621 4417 ; NONE
1621 4418 ;
1621 4419 ; OUTPUT PARAMETERS:
1621 4420 ;
1621 4421 ; REGISTER PAIR H & L - POINT AT FIRST CHARACTER PAST THE MATCHED STRING
1621 4422 ; REGISTER D - CONTAINS THE WORD POSITION IN THE LIST OF
1621 4423 ; THE MATCHED STRING.
1621 4424 ; C BIT SET - IF NO MATCH WAS FOUND IN THE LIST
1621 4425 ; C BIT CLEAR - IF A MATCH WAS FOUND.
1621 4426 ;
1621 4427 ;--
1621 4428
1621 4429 SEARCH_LIST:
1621 4430 MVI D,0 ; INIT THE WORD POSITION
1623 4431 2$: MVI E,2 ; WANT TO MATCH TWO CHARACTERS
1625 4432 1$: LDAX B ; GET CHARACTER FROM SEARCH LIST
1626 4433 ORA A ; CHECK IF LIST TERMINATOR
1627 4434 JM SEARCH_ERR ; IF TERMINATOR, EXIT WITH C SET
162A 4435 CMP H ; DOES CHARACTER MATCH?
162B 4436 JZ 3$ ; BRANCH IF YES
162E 4437 ;
162E 4438 ; CHARACTER DID NOT MATCH. IF LOOP COUNT = 2, BUMP MATCH STRING ADDRESS BY
162E 4439 ; TWO BYTES. INCREMENT WORD POSITION BY TWO AND RESTART.
162E 4440 ; IF LOOP COUNT = 1, BUMP MATCH STRING ADDRESS BY ONE, INCREMENT THE WORD
162E 4441 ; POSITION BY TWO, DECREMENT THE INPUT STRING ADDRESS BY ONE, AND RESTART.
162E 4442 ;
162E 4443 INR D ; INCREMENT WORD POSITION BY TWO
162F 4444 INR D ;
1630 4445 INX B ; INCREMENT MATCH STR ADR BY ONE
1631 4446 MOV A,E ; GET THE LOOP COUNT
1632 4447 CPI 2 ; LOOP COUNT = 2?
1634 4448 JNZ 4$ ; BRANCH IF NO
1637 4449 INX B ; INCREMENT MATCH STR ADR BY TWO
1638 4450 JMP 1$ ; KEEP TRYING TO MATCH
163B 4451 4$: DCX H ; DECREMENT INPUT STRING ADDRESS

```

```
163C 4452          JMP          2$          ; KEEP TRYING TO MATCH
163F 4453          ;
163F 4454          ; GOT A MATCH. IF LOOP COUNT = 2, BUMP MATCH STRING ADDRESS AND INPUT
163F 4455          ; STRING ADDRESS BY ONE, DECREMENT THE LOOP COUNT, AND TRY SECOND CHARAC.
163F 4456          ; IF LOOP COUNT = 1, BUMP THE INPUT STRING ADDRESS BY ONE AND RETURN
163F 4457          ; TO CALLER WITH C BIT CLEAR
163F 4458          ;
163F 4459 3$:      DCR          E          ; DECREMENT LOOP COUNT BY 1
1640 4460          JZ          SEARCH_DONE ; GOT TWO MATCHES, EXIT WITH C CLEAR
1643 4461          INX          B          ; INCREMENT MATCH STR ADDR
1644 4462          INX          H          ; INCREMENT INPUT STRING ADDR
1645 4463          JMP          1$          ; SEE IF SECOND CHAR MATCHES
1648 4464          ;
1648 4465          ; FOUND A MATCH. BUMP INPUT STRING POINTER TO NEXT CHARACTER AND RETURN
1648 4466          ; WITH C CLEAR
1648 4467          ;
1648 4468 SEARCH_DONE:
1648 4469          INX          H          ; INCREMENT INPUT STRING PTR
1649 4470          STC          ; SET C BIT
164A 4471          CMC          ; CLEAR THE C BIT
164B 4472          RET          ; RETURN TO CALLER
164C 4473          ;
164C 4474          ; SEARCH FAILED. RETURN WITH C BIT SET.
164C 4475          ;
164C 4476 SEARCH_ERR:
164C 4477          STC          ;
164D 4478          RET          ;
164E 4479          ;
```

```
164E 4481 .SBTTL " SKIP SPACES ROUTINE
164E 4482 ;++
164E 4483 ;
164E 4484 ; FUNCTIONAL DESCRIPTION:
164E 4485 ;
164E 4486 ; THIS ROUTINE INCREMENTS THE INPUT STRING POINTER TO THE FIRST
164E 4487 ; CHARACTER AFTER A STRING OF SPACES. IF A CARRIAGE RETURN IS
164E 4488 ; FOUND, RETURN IS MADE WITH THE C BIT SET.
164E 4489 ;
164E 4490 ; CALLING SEQUENCE:
164E 4491 ;
164E 4492 ; CALL SKIP_SPACES
164E 4493 ;
164E 4494 ; INPUT PARAMETERS:
164E 4495 ;
164E 4496 ; REGISTER PAIR H & L - CONTAINS ADDRESS OF INPUT STRING
164E 4497 ;
164E 4498 ; IMPLICIT INPUTS:
164E 4499 ;
164E 4500 ; NONE
164E 4501 ;
164E 4502 ; OUTPUT PARAMETERS:
164E 4503 ;
164E 4504 ; REGISTER PAIR H & L - CONTAINS ADDRESS OF FIRST CHARACTER AFTER A
164E 4505 ; STRING OF SPACES.
164E 4506 ;
164E 4507 ; IMPLICIT OUTPUTS:
164E 4508 ;
164E 4509 ; NONE
164E 4510 ;
164E 4511 ;--
164E 4512 ;
164E 4513 SKIP_SPACES:
164E 4514 MOV A,M ; GET CHARACTER IN INPUT BUFFER
164F 4515 CPI <^XD> ; IS IT A CARRIAGE RETURN?
1651 4516 JZ 2$ ; BRANCH IF YES
1654 4517 CPI <^X20> ; IS IT A SPACE?
1656 4518 INX H ;
1657 4519 JNZ SKIP_SPACES ; BRANCH IF NO
165A 4520 ;
165A 4521 ; FOUND THE FIRST SPACE. NOW SKIP UNTIL NO MORE SPACES.
165A 4522 ;
165A 4523 1$: MOV A,M ; GET THE CHARACTER
165B 4524 CPI <^X20> ; IS IT A SPACE?
165D 4525 JNZ 3$ ; BRANCH IF NO
1660 4526 INX H ; INCREMENT INPUT BUFFER PTR
1661 4527 JMP 1$ ; CONTINUE SKIPPING SPACES
1664 4528 ;
1664 4529 ; FOUND A CHARACTER THAT IS NOT A SPACE. CHECK IF IT'S A CARRIAGE RETURN.
1664 4530 ;
1664 4531 3$: CPI <^XD> ; IS CHAR. A CARRIAGE RETURN?
1666 4532 JZ 2$ ; BRANCH IF YES
1669 4533 STC ; SET THE C BIT
166A 4534 CMC ; CLEAR THE C BIT
166B 4535 RET ; RETURN TO CALLER
```

ZZ-ECKAA-8.7 V08.07
ECKAA
V08.07

F 10
SKIP SPACES R11-FEB-1986
VAX-11/750 MICRO DIAGNOSTIC MONITOR
SKIP SPACES ROUTINE

Fiche 1 Frame F10 Sequence 122
11-FEB-1986 10:07:52 VAX/VMS Macro V04-00 Page 97
11-FEB-1986 10:06:37 [VAX750.MONITOR]ECKAA.MAR;811 (37)

166C 4536 2\$: STC
166D 4537 RET
166E 4538

: SET THE C BIT
: RETURN TO CALLER

```
166E 4540 .SBTTL PUT D REGISTER IN PARSE TABLE ROUTINE
166E 4541 ;**
166E 4542 ;
166E 4543 ; FUNCTIONAL DESCRIPTION:
166E 4544 ;
166E 4545 ; THIS ROUTINE PUTS THE D REGISTER INTO THE CURRENT POSITION OF
166E 4546 ; THE PARSE TABLE.
166E 4547 ;
166E 4548 ; CALLING SEQUENCE:
166E 4549 ;
166E 4550 ; CALL PUT_D_IN_TBL
166E 4551 ;
166E 4552 ; INPUT PARAMETERS:
166E 4553 ;
166E 4554 ; D REGISTER - CONTAINS PARSE TABLE CODE
166E 4555 ;
166E 4556 ; IMPLICIT INPUTS:
166E 4557 ;
166E 4558 ; PARSE_TBL_PTR - CONTAINS THE ADDRESS OF THE CURRENT POSITION
166E 4559 ; IN THE PARSE TABLE
166E 4560 ;
166E 4561 ; IMPLICIT OUTPUTS:
166E 4562 ;
166E 4563 ; PARSE_TBL_PTR - IS INCREMENTED BY ONE
166E 4564 ;
166E 4565 ;--
166E 4566
166E 4567 PUT_D_IN_TBL:
166E 4568 PUSH H ; SAVE H & L REGISTERS
166F 4569 LHLD PARSE_TBL_PTR ; GET ADDR OF PARSE TABLE
1672 4570 MOV M,D ; STORE DATA IN PARSE TABLE
1673 4571 INX H ; INCREMENT THE TABLE ADDRESS
1674 4572 SHLD PARSE_TBL_PTR ; PUT ADDR BACK IN POINTER
1677 4573 POP H ; RESTORE H & L REGISTERS
1678 4574 RET ; RETURN TO CALLING SEQUENCE
1679 4575
```

```

1679 4577 .SBTTL PARSE HEX NUMBER ROUTINE
1679 4578 ;++
1679 4579 ;
1679 4580 ; FUNCTIONAL DESCRIPTION:
1679 4581 ;
1679 4582 ; THIS ROUTINE CONVERTS AN ASCII STRING OF CHARACTERS INTO A
1679 4583 ; BINARY NUMBER. IF THE ASCII STRING IS NOT COMPOSED OF LEGAL
1679 4584 ; HEXIDECIMAL DIGITS, RETURN IS MADE WITH THE C BIT SET. THE
1679 4585 ; STRING MUST BE TERMINATED WITH A SPACE OR CARRIAGE RETURN.
1679 4586 ;
1679 4587 ; THE BINARY NUMBER IS PLACED IN THE PARSE TABLE FROM MOST
1679 4588 ; SIGNIFICANT BYTE TO LEAST SIGNIFICANT BYTE. THE PARSE TABLE
1679 4589 ; ALWAYS RECEIVES A 32 BIT NUMBER.
1679 4590 ;
1679 4591 ; CALLING SEQUENCE:
1679 4592 ;
1679 4593 ; CALL PARSE_HEX_NUMB
1679 4594 ;
1679 4595 ; INPUT PARAMETERS:
1679 4596 ;
1679 4597 ; REGISTER PAIR H & L - CONTAINS THE ADDRESS OF THE INPUT STRING.
1679 4598 ;
1679 4599 ; IMPLICIT INPUTS:
1679 4600 ;
1679 4601 ; NONE
1679 4602 ;
1679 4603 ; IMPLICIT OUTPUTS:
1679 4604 ;
1679 4605 ; REGISTER PAIR H & L - CONTAINS THE ADDRESS OF THE INPUT CHARACTER
1679 4606 ; AFTER THE ASCII STRING OF DIGITS.
1679 4607 ;
1679 4608 ; COMPLETION CODES:
1679 4609 ;
1679 4610 ; C BIT SET - ILLEGAL HEXIDECIMAL STRING
1679 4611 ; C BIT CLEAR - STRING PARSE WAS SUCCESSFUL
1679 4612 ;
1679 4613 ;--
1679 4614 ;
1679 4615 PARSE_HEX_NUMB:
1679 4616 LXI B,PARSE_HEX_TEMP ; GET ADDRESS OF TEMP LOCATION
167C 4617 XRA A ; CLEAR THE A REGISTER
167D 4618 STAX B ; INITIALIZE THE TEMP LOCATION
167E 4619 INX B ; ...
167F 4620 STAX B ; ...
1680 4621 INX B ; ...
1681 4622 STAX B ; ...
1682 4623 INX B ; ...
1683 4624 STAX B ; ...
1684 4625 ;
1684 4626 ; NOW START CONVERTING THE INPUT STRING
1684 4627 ;
1684 4628 1$: MOV A,M ; GET A CHARACTER
1685 4629 CPI <^XD> ; IS IT A CARRIAGE RETURN?
1687 4630 JZ PARSE_HEX_DONE ; BRANCH IF YES
168A 4631 CPI <^X20> ; IS IT A SPACE?

```

```
168C 4632 JZ PARSE_HEX_DONE ; BRANCH IF YES
168F 4633 INX H ; INCREMENT POINTER TO INPUT STRING
1690 4634 SUI <^X30> ; IS DIGIT GREATER THAN 0?
1692 4635 JM PARSE_HEX_ERR ; BRANCH IF NO
1695 4636 CPI <^XA> ; IS DIGIT NUMERIC?
1697 4637 JM 2$ ; BRANCH IF YES
169A 4638 ;
169A 4639 ; CHECK IF DIGIT IS A LEGAL ALPHA CHARACTER.
169A 4640 ;
169A 4641 SUI <^X11> ; IS DIGIT ALPHA?
169C 4642 JM PARSE_HEX_ERR ; BRANCH IF NO
169F 4643 CPI 6 ; IS DIGIT LEGAL HEX?
16A1 4644 JP PARSE_HEX_ERR ; BRANCH IF NO
16A4 4645 ADI <^XA> ; MAKE DIGIT BINARY HEXIDECIMAL
16A6 4646 ;
16A6 4647 ; BINARY DIGIT IS IN 'A' REG. SHIFT WORKING 32 LOCATION AND INSERT DIGIT.
16A6 4648 ;
16A6 4649 2$: PUSH $PSW ; SAVE 'A' REG
16A7 4650 MVI E,4 ; SET A LOOP COUNT FOR 4 SHIFTS
16A9 4651 4$: MVI D,4 ; SET A LOOP COUNT FOR 4 BYTES
16AB 4652 LXI B,PARSE_HEX_TEMP ; GET ADDRESS OF TEMP LOCATION
16AE 4653 PUSH $PSW ; SAVE THE C BIT (THIS IS A DUMMY
16AF 4654 ; PUSH TO INIT THE FOLLOWING LOOP)
16AF 4655 3$: POP $PSW ; RESTORE THE C BIT
16B0 4656 LDAX B ; GET A BYTE OF TEMP LOCATION
16B1 4657 RAL ; SHIFT IT LEFT
16B2 4658 PUSH $PSW ; SAVE THE C BIT
16B3 4659 STAX B ; PUT IT BACK IN TEMP LOCATION
16B4 4660 INX B ; POINT AT NEXT BYTE
16B5 4661 DCR D ; DECREMENT THE COUNT
16B6 4662 JNZ 3$ ; CONTINUE FOR 4 BYTES
16B9 4663 ;
16B9 4664 ; FINISHED 4 BYTES. CHECK IF 4 SHIFTS HAVE BEEN DONE
16B9 4665 ;
16B9 4666 POP $PSW ; FIXUP STACK
16BA 4667 DCR E ; DECREMENT
16BB 4668 JNZ 4$ ; CONTINUE FOR 4 SHIFTS
16BE 4669 ;
16BE 4670 ; TEMP IS NOW SHIFTED. PUT THE NEW 4 BITS INTO THE TEMP LOCATION
16BE 4671 ;
16BE 4672 LXI B,PARSE_HEX_TEMP ; GET ADDR OF TEMP LOCATION
16C1 4673 LDAX B ; GET THE LOW BYTE
16C2 4674 ANI <^XF0> ; THROW AWAY GARBAGE CARRY BIT
16C4 4675 MOV D,A ; SAVE IN D REGISTER
16C5 4676 POP $PSW ; GET THE DATA TO INSERT
16C6 4677 ORA D ; INSERT INTO THE LOW BYTE
16C7 4678 STAX B ; PUT BACK IN TEMP LOCATION
16C8 4679 JMP 1$ ; GO GET NEXT CHARACTER
16CB 4680 ;
16CB 4681 ; FOUND A CARRIAGE RETURN OR SPACE. NOW INSERT THE TEMP LOCATION
16CB 4682 ; INTO THE PARSE TABLE.
16CB 4683 ;
16CB 4684 PARSE_HEX_DONE:
16CB 4685 LXI B,PARSE_HEX_TEMP+3 ; GET ADDR OF HIGH BYTE
16CE 4686 PUSH H ; SAVE H & L REGISTERS
```

```
16CF 4687      LHLD      PARSE_TBL_PTR ; GET ADDR OF PARSE TABLE
16D2 4688      MVI       D,4          ; SET A LOOP COUNT FOR 4 BYTES
16D4 4689 1$:  LDAX      B          ; GET A BYTE OF DATA
16D5 4690      MOV       M,A        ; PUT IN PARSE TABLE
16D6 4691      INX       H          ; INCREMENT PARSE TABLE ADDRESS
16D7 4692      DCX      B          ; DECREMENT TEMP ADDRESS
16D8 4693      DCR      D          ; DECREMENT THE LOOP COUNT
16D9 4694      JNZ      1$        ; MOVE 4 BYTES
16DC 4695      SHLD     PARSE_TBL_PTR ; SAVE THE CURRENT PARSE TBL ADDRESS
16DF 4696      POP      H          ; RESTORE THE H & L REGISTERS
16E0 4697      STC       ; SET THE C BIT
16E1 4698      CMC       ; CLEAR THE C BIT
16E2 4699      RET        ; RETURN TO THE CALLER
16E3 4700      ;
16E3 4701      ; AN ERROR WAS FOUND PARSING THE STRING. RETURN WITH C BIT SET.
16E3 4702      ;
16E3 4703      ; PARSE_HEX_ERR:
16E3 4704      STC       ;
16E4 4705      RET        ;
16E5 4706
```

```
16E5 4708      .SBTTL  ACTION ROUTINES
16E5 4709      .SBTTL  DIAGNOSE ACTION ROUTINE
16E5 4710      ;++
16E5 4711      ;
16E5 4712      ; FUNCTIONAL DESCRIPTION:
16E5 4713      ;
16E5 4714      ; THIS ROUTINE PEFORMS THE NECESSARY FUNCTIONS FOR A 'DIGNOSE'
16E5 4715      ; COMMAND. THE FOLLOWING FLAGS ARE INITIALIZED:
16E5 4716      ;
16E5 4717      ; LOST_FLAG  CONT_FLAG  TSTSPAN_FLAG
16E5 4718      ; LOOP_FLAG  MER_FLAG  BELL_FLAG  HALT_FLAG
16E5 4719      ;
16E5 4720      ; THE PARSE TABLE IS THEN INTERPRETED INTO THE APPROPRIATE ACTIONS.
16E5 4721      ;
16E5 4722      ; CALLING SEQUENCE:
16E5 4723      ;
16E5 4724      ; CALL    DIAGNOSE_ACTION
16E5 4725      ;
16E5 4726      ; INPUT PARAMETERS:
16E5 4727      ;
16E5 4728      ; REGISTER PAIR D & E - CONTAIN ADDRESS OF PARSE TABLE PAST THE
16E5 4729      ; COMMAND CODE.
16E5 4730      ;
16E5 4731      ; IMPLICIT INPUTS:
16E5 4732      ;
16E5 4733      ; NONE
16E5 4734      ;
16E5 4735      ; OUTPUT PARAMETERS:
16E5 4736      ;
16E5 4737      ; NONE
16E5 4738      ;
16E5 4739      ; IMPLICIT OUTPUTS:
16E5 4740      ;
16E5 4741      ; THE PROGRAM CONTROL FLAGS ARE INITIALIZED.
16E5 4742      ; THE 'DIAGNOSE_FLAG' IS SET IN THE SOFTWARE CONTROL REGISTER
16E5 4743      ;
16E5 4744      ; COMPLETION CODES:
16E5 4745      ;
16E5 4746      ; C BIT SET   - ILLEGAL QUALIFIER WITH THE 'DIAGNOSE' COMMAND
16E5 4747      ; C BIT CLEAR - DIAGNOSE COMMAND EXECUTION WAS SUCCESSFUL
16E5 4748      ;
16E5 4749      ;--
16E5 4750      ;
16E5 4751      ;
16E5 4752      ; DIAGNOSE_ACTION:
16E5 4753      ; LDA      PROG_CTRL_REG      ; GET PROGRAM CONTROL REG
16E8 4754      ; ANI      <^C<LOOP_FLAG!MER_FLAG!- ; CLEAR THE LOOP, MER,
16E8 4755      ; BELL_FLAG>>      ; AND BELL FLAGS
16EA 4756      ; ORI      HALT_FLAG      ; SET THE 'HALT' FLAG
16EC 4757      ; STA      PROG_CTRL_REG      ; ...
16EF 4758      ; LDA      SOFT_CTRL_REG      ; GET THE SOFTWARE CONTROL REG
16F2 4759      ; ANI      <^C<LOST_FLAG!- ; CLEAR THE LOST,
16F2 4760      ; CONT_FLAG!TSTSPAN_FLAG!- ; CONTINUE, TEST SPAN,
16F2 4761      ; DIAGNOSE_FLAG>> ; AND DIAGNOSE FLAGS
16F4 4762      ; STA      SOFT_CTRL_REG      ; ...
```

```

16F7 4763 LDA SOFT_CTRL_REGB ; SET THE ERROR LOG INIT FLAG
16FA 4764 ORI ERR_LOG_INIT ; ...
16FC 4765 ANI <^C^QV_FLAG!- ; AND CLEAR THE QV AND RDM FLAGS
16FC 4766 RDM_FLAG!- ; AND SBE FLAG
16FC 4767 SBE_FLAG>> ; ...
16FE 4768 STA SOFT_CTRL_REGB ; ...
1701 4769 MVI A,1 ; GET THE INITIAL PASS COUNT
1703 4770 STA USER_PASS_CNT ; INIT THE PASS COUNT
1706 4771 XCHG ; PUT PARSE TBL ADDR IN H & L REG'S
1707 4772 XRA A ; CLEAR THE A REG
1708 4773 STA TEST_NUMBER ; INIT THE TEST NUMBER
170B 4774 STA TYPE_TEST_FLAG ; INIT THE TYPE TEST FLAG
170E 4775 MOV D,A ; INITIALIZE THE TEST FLAG
170F 4776 MVI C,3 ; PUT CONSTANT 3 IN B & C REG'S
1711 4777 MOV B,A ; ...
1712 4778 ;
1712 4779 ; NOW START DECODING THE CODES IN THE PARSE TABLE
1712 4780 ;
1712 4781 DIAGNOSE_LOOP:
1712 4782 MOV A,M ; GET A KEYWORD
1713 4783 ANI <^X7F> ; DISCARD KEYWORD CODE BIT
1715 4784 INX H ; INCREMENT PARSE TABLE POINTER
1716 4785 CPI <^X55> ; END OF PARSE TABLE?
1718 4786 JZ DIAGNOSE_DONE ; BRANCH IF YES
171B 4787 ;
171B 4788 ; SEE WHICH KEYWORD IS IN PARSE TABLE.
171B 4789 ;
171B 4790 CPI TEST_KEY ; 'TEST' KEYWORD?
171D 4791 JZ TEST_SUB_ACTION ; BRANCH IF YES
1720 4792 CPI PASS_KEY ; 'PASS' KEYWORD?
1722 4793 JZ PASS_SUB_ACTION ; BRANCH IF YES
1725 4794 CPI CONT_KEY ; 'CONTINUE' KEYWORD?
1727 4795 JZ CONT_SUB_ACTION ; BRANCH IF YES
172A 4796 CPI QV_KEY ; 'QUICK VERIFY' KEYWORD?
172C 4797 JZ QV_SUB_ACTION ; BRANCH IF YES
172F 4798 CPI RDM_KEY ; 'RDM' KEYWORD?
1731 4799 JZ RDM_SUB_ACTION ; BRANCH IF YES
1734 4800 CPI SBE_KEY ; SINGLE BIT ERROR ENABLE?
1736 4801 JNZ DIAGNOSE_ERR ; BRANCH IF NO
1739 4802 ;
1739 4803 ; SBE KEYWORD. SET THE SBE FLAG.
1739 4804 ;
1739 4805 LDA SOFT_CTRL_REGB ; GET CURRENT FLAGS
173C 4806 ORI SBE_FLAG ; SET THE SBE FLAG
173E 4807 STA SOFT_CTRL_REGB ; ...
1741 4808 JMP DIAGNOSE_LOOP ; CONTINUE PARSING
1744 4809 ;
1744 4810 ; RDM KEYWORD. SET THE RDM FLAG.
1744 4811 ;
1744 4812 RDM_SUB_ACTION:
1744 4813 LDA SOFT_CTRL_REGB ; GET CURRENT FLAGS
1747 4814 ORI RDM_FLAG ; SET THE RDM FLAG
1749 4815 STA SOFT_CTRL_REGB ; ...
174C 4816 JMP DIAGNOSE_LOOP ; CONTINUE PARSING
174F 4817 ;

```

```
174F 4818 ; QUICK VERIFY KEYWORD. SET THE QV FLAG.
174F 4819 ;
174F 4820 QV_SUB_ACTION:
174F 4821     LDA          SOFT_CTRL_REGB ; GET CURRENT FLAGS
1752 4822     ORI          QV_FLAG       ; SET THE QV FLAG
1754 4823     STA          SOFT_CTRL_REGB ;
1757 4824     JMP          DIAGNOSE_LOOP ; CONTINUE PARSING
175A 4825 ;
175A 4826 ; CONTINUE KEYWORD WAS RECOGNIZED. SET THE 'CONT' FLAG AND KEEP PARSING.
175A 4827 ;
175A 4828 CONT_SUB_ACTION:
175A 4829     LDA          SOFT_CTRL_REG  ; GET CONTENTS OF SOFTWARE CTRL REG
175D 4830     ORI          CONT_FLAG   ; SET THE 'CONT' FLAG
175F 4831     STA          SOFT_CTRL_REG ; UPDATE SOFTWARE CONTROL REG
1762 4832     JMP          DIAGNOSE_LOOP ; CONTINUE SEARCHING PARSE TABLE
1765 4833 ;
1765 4834 ; 'TEST' KEYWORD WAS RECOGNIZED. SET THE 'LOST' FLAG AND GET THE TEST
1765 4835 ; NUMBER(S).
1765 4836 ;
1765 4837 TEST_SUB_ACTION:
1765 4838     MOV          A,D           ; GET THE TEST/SECTION FLAG
1766 4839     ORA          A           ; SET THE CONDITION CODES
1767 4840     JNZ          DIAGNOSE_ERR ; BRANCH IF SECTION ALREADY SPECIFIED
176A 4841     MVI          D,1       ; SET THE TEST/SECTION FLAG
176C 4842     DAD          B           ; POINT AT THE TEST NUMBER IN PARSE TBL
176D 4843     MOV          A,M       ; GET THE TEST NUMBER
176E 4844     STA          SPEC_TEST_NUMB ; SAVE
1771 4845     INX          H           ; POINT AT NEXT ENTRY IN PARSE TBL
1772 4846     MVI          E,LOST_FLAG ; GET THE 'LOST' FLAG IN 'E' REG
1774 4847     MOV          A,M       ; GET NEXT ELEMENT IN PARSE TBL
1775 4848     ORA          A           ; SET THE CONDITION CODES
1776 4849     MVI          A,0       ;
1778 4850     JM          1$         ; BRANCH IF IT'S A KEYWORD
177B 4851     MOV          A,M       ; GET NEXT ELEMENT IN PARSE TBL
177C 4852     CPI          <^X55>   ; END OF PARSE TABLE?
177E 4853     MVI          A,0       ; CLEAR THE A REG
1780 4854     JZ          1$         ; BRANCH IF YES
1783 4855 ;
1783 4856 ; AN END TEST HAS BEEN SPECIFIED. GET THE NUMBER OUT OF THE PARSE TABLE.
1783 4857 ;
1783 4858     DAD          B           ; POINT AT THE NUMBER
1784 4859     MOV          A,M       ; GET THE END TEST NUMBER
1785 4860     STA          TEST_SPAN_END ; SAVE
1788 4861     INX          H           ; UPDATE PARSE TBL POINTER
1789 4862     MVI          A,TSTSPAN_FLAG ; GET THE TEST SPAN FLAG
178B 4863 ;
178B 4864 ; 'E' REG HAS 'LOST' FLAG. 'A' REG HAS 'TSTSPAN' FLAG OR ZERO.
178B 4865 ; OR TOGETHER AND LOAD SOFT_CTRL_REG.
178B 4866 ;
178B 4867 1$:
178B 4868     ORA          E           ; OR IN THE 'LOST' FLAG
178C 4869     MOV          E,A       ; SAVE
178D 4870     LDA          SOFT_CTRL_REG ; GET THE CONTROL REGISTER
1790 4871     ORA          E           ; OR IN THE TWO FLAGS
1791 4872     STA          SOFT_CTRL_REG ; SAVE
```

```
1794 4873            JMP                    DIAGNOSE_LOOP    ; CONTINUE PARSING DIAGNOSE COMMAND
1797 4874            ;
1797 4875            ; 'PASS' KEYWORD WAS RECOGNIZED. GET THE USER PASS COUNT FROM THE
1797 4876            ; PARSE TABLE.
1797 4877            ;
1797 4878            PASS_SUB_ACTION:
1797 4879            DAD                    B                    ; POINT AT THE PASS COUNT
1798 4880            MOV                   A,M                ; GET THE PASS COUNT
1799 4881            STA                   USER_PASS_CNT    ; SAVE
179C 4882            INX                   H                    ; POINT AT NEXT ELEMENT IN PARSE TBL
179D 4883            JMP                   DIAGNOSE_LOOP    ; CONTINUE PARSING DIAGNOSE COMMAND
17A0 4884            ;
17A0 4885            ; AN ILLEGAL DIAGNOSE COMMAND WAS DETECTED. RETURN WITH C BIT SET
17A0 4886            ;
17A0 4887            DIAGNOSE_ERR:
17A0 4888            STC
17A1 4889            RET
17A2 4890            ;
17A2 4891            ; DIAGNOSE PARSE COMPLETE. SET THE 'DIAGNOSE_FLAG' AND RETURN.
17A2 4892            ;
17A2 4893            DIAGNOSE_DONE:
17A2 4894            LDA                   SOFT_CTRL_REG    ; GET THE CONTROL REG
17A5 4895            ORI                   DIAGNOSE_FLAG    ; SET THE 'DIAGNOSE' FLAG
17A7 4896            STA                   SOFT_CTRL_REG    ; ...
17AA 4897            STC
17AB 4898            CMC
17AC 4899            RET
17AD 4900
```

```
17AD 4902 .SBTTL SHOW ACTION ROUTINE
17AD 4903 ;++
17AD 4904 ;
17AD 4905 ; FUNCTIONAL DESCPTION:
17AD 4906 ;
17AD 4907 ; THIS ROUTINE DISPLAYS THE CURRENT VALUE OF THE 'PROGRAM' CONTROL
17AD 4908 ; FLAGS OR THE VISIBILITY BUS. THE PRINT OUT FOR THE PROGRAM CONTROL
17AD 4909 ; FLAGS IS:
17AD 4910 ;
17AD 4911 ; FLAGS SET: <FLAG NAMES>
17AD 4912 ; FLAGS CLEAR: <FLAG NAMES>
17AD 4913 ;
17AD 4914 ; THE VBUS PRINTOUT DISPLAYS THE VALUE OF EACH VBUS BIT IN BINARY.
17AD 4915 ; GROUPS OF 8 BITS ARE SEPARATED BY COMMAS. BIT 0 OF THE BUS IS
17AD 4916 ; ON THE RIGHT OF THE TYPEOUT.
17AD 4917 ;
17AD 4918 ; CALLING SEQUENCE:
17AD 4919 ;
17AD 4920 ; CALL SHOW_ACTION
17AD 4921 ;
17AD 4922 ; INPUT PARAMETERS:
17AD 4923 ;
17AD 4924 ; REGISTER PAIR D & E - CONTAIN THE ADDRESS OF THE PARSE_TABLE+1
17AD 4925 ;
17AD 4926 ; IMPLICIT INPUTS:
17AD 4927 ;
17AD 4928 ; PROG_CTRL_REG - CONTAINS THE CURRENT STATE OF THE FLAGS.
17AD 4929 ;
17AD 4930 ; OUTPUT PARAMETERS:
17AD 4931 ;
17AD 4932 ; NONE
17AD 4933 ;
17AD 4934 ; IMPLICIT OUTPUTS:
17AD 4935 ;
17AD 4936 ; NONE
17AD 4937 ;
17AD 4938 ; COMPLETION CODES:
17AD 4939 ;
17AD 4940 ; RETURN IS MADE WITH THE C BIT CLEAR IF NO ERROR OTHERWISE IT IS SET.
17AD 4941 ;
17AD 4942 ;--
17AD 4943 ;
17AD 4944 SHOW_ACTION:
17AD 4945 ;
17AD 4946 ; SEE IF THIS IS A FLAG SHOW
17AD 4947 ;
17AD 4948 LDAX D ; GET THE KEYWORD CODE
17AE 4949 CPI <FLAG_KEY!^X80> ; IS IT 'FLAG'?
17B0 4950 JNZ 10$ ; BRANCH IF NO
17B3 4951 ;
17B3 4952 ; DISPLAY THE PROGRAM CONTROL FLAGS
17B3 4953 ;
17B3 4954 TYPE SHOW_MESSAGE_1 ; TYPE THE 'FLAGS SET: ' MESSAGE
17BC 4955 LDA PROG_CTRL_REG ; GET THE FLAGS
17BF 4956 CALL TYPE_FLAGS ; TYPE THE FLAG NAMES THAT ARE SET
```

```
17C2 4957          TYPE      SHOW_MESSAGE_2 ; TYPE THE 'FLAGS CLEAR: ' MESSAGE
17CB 4958          LDA        PROG_CTRL_REG  ; GET THE FLAGS
17CE 4959          CMA        ; GET THE FLAGS THAT ARE CLEAR
17CF 4960          CALL       TYPE_FLAGS    ; TYPE THE FLAG NAMES THAT ARE CLEAR
17D2 4961          JMP        50$           ; EXIT
17D5 4962          ;
17D5 4963          ; THIS IS A VISIBILITY BUS 'SHOW'. READ THE V BUS.
17D5 4964          ;
17D5 4965          10$:      CPI        <VB_KEY!^X80> ; IS IT A VISIBILITY BUS SHOW?
17D7 4966          JNZ        40$           ; BRANCH IF NO
17DA 4967          ;
17DA 4968          ; FIRST CLEAR THE CONTROL FILE
17DA 4969          ;
17DA 4970          LDA        DCS_CTRL_RE_CPY ; GET CURRENT CONTENTS OF CTRL REG
17DD 4971          ORI        CLEAR_CTRL_FILE ; SET CONTROL FILE CLEAR BIT
17DF 4972          STA        DCS_CTRL_REG   ;
17E2 4973          ANI        <^C<CLEAR_CTRL_FILE>> ;
17E4 4974          STA        DCS_CTRL_REG   ; CLEAR CONTROL FILE CLEAR BIT
17E7 4975          LDA        FRONT_PNL_2   ; FIRST LOAD THE VBUS
17EA 4976          ANI        <FAULT!TEST!REMOTE> ; ...
17EC 4977          ORI        <MASTER_HALT_EN!-
17EC 4978          VBUS_LOAD!TRAP_HALT_EN>
17EE 4979          STA        RD_CTRL_REG    ; LOAD THE VBUS
17F1 4980          ANI        <^CVBUS_LOAD>   ; ...
17F3 4981          STA        RD_CTRL_REG    ;
17F6 4982          CALL       READ_V_BUS    ; READ THE V BUS INTO THE BUFFER
17F9 4983          TYPE      VBUS_MSG      ; TYPE 'VBUS='
1802 4984          ;
1802 4985          ; NOW TYPE THE BITS
1802 4986          ;
1802 4987          LXI        H,V_BUS_BUFFER+39 ; GET THE BUFFER ADDRESS
1805 4988          MVI        B,40           ; SET THE LOOP COUNT
1807 4989          20$:      MOV        A,M    ; GET THE BIT VALUE
1808 4990          DCX        H             ; POINT AT NEXT BIT
1809 4991          PUSH       H             ; SAVE REGISTERS
180A 4992          PUSH       B             ;
180B 4993          STA        SHOW_TEMP    ;
180E 4994          TYPEN      SHOW_TEMP    ; TYPE THE BIT VALUE
1819 4995          POP        B             ; RESTORE THE REGISTERS
181A 4996          POP        H             ;
181B 4997          DCR        B             ; DONE 40 BITS YET?
181C 4998          JZ        50$          ; BRANCH IF YES
181F 4999          MOV        A,B         ; GET THE LOOP COUNT
1820 5000          ANI        7           ; TIME FOR A COMMA?
1822 5001          JNZ        20$          ; BRANCH IF NO
1825 5002          PUSH       H             ; SAVE REGISTERS
1826 5003          PUSH       B             ;
1827 5004          TYPE      COMMA_MSG    ; TYPE A COMMA
1830 5005          POP        B             ; RESTORE THE REGISTERS
1831 5006          POP        H             ;
1832 5007          JMP        20$          ; CONTINUE
1835 5008          ;
1835 5009          ; RETURN WITH C BIT CLEAR
1835 5010          ;
1835 5011          50$:      STC
```

ZZ-ECKAA-8.7 V08.07
ECKAA
V08.07

D 11
SHOW ACTION R11-FEB-1986
VAX-11/750 MICRO DIAGNOSTIC MONITOR
SHOW ACTION ROUTINE

Fiche 1 Frame D11 Sequence 133
11-FEB-1986 10:07:52 VAX/VMS Macro V04-00 Page 108
11-FEB-1986 10:06:37 [VAX750.MONITOR]ECKAA.MAR;811 (41)

```
1836 5012          CMC
1837 5013          RET
1838 5014 ;
1838 5015 ; ILLEGAL KEYWORD. RETURN WITH C BIT SET
1838 5016 ;
1838 5017 40$:    STC
1839 5018          RET
183A 5019
```

```
183A 5021 .SBTTL CONTINUE ACTION ROUTINE
183A 5022 ;++
183A 5023 ;
183A 5024 ; FUNCTIONAL DESCRIPTION:
183A 5025 ;
183A 5026 ; THIS ROUTINE SETS THE 'CONTINUE' FLAG IN THE SOFTWARE CONTROL
183A 5027 ; REGISTER.
183A 5028 ;
183A 5029 ; CALLING SEQUENCE:
183A 5030 ;
183A 5031 ; CALL CONTINUE_ACTION
183A 5032 ;
183A 5033 ; INPUT PARAMETERS:
183A 5034 ;
183A 5035 ; NONE
183A 5036 ;
183A 5037 ; IMPLICIT INPUTS:
183A 5038 ;
183A 5039 ; NONE
183A 5040 ;
183A 5041 ; OUTPUT PARAMETERS:
183A 5042 ;
183A 5043 ; NONE
183A 5044 ;
183A 5045 ; IMPLICIT OUTPUTS:
183A 5046 ;
183A 5047 ; SOFT_CTRL_REG - 'CONTINUE_FLAG' SI SET
183A 5048 ;
183A 5049 ; COMPLETION CODES:
183A 5050 ;
183A 5051 ; RETURN IS MADE WITH THE C BIT CLEAR.
183A 5052 ;
183A 5053 ;--
183A 5054 ;
183A 5055 CONTINUE_ACTION:
183A 5056 LDA SOFT_CTRL_REG ; GET THE CURRENT FLAGS
183D 5057 ORI CONTINUE_FLAG ; SET THE 'CONTINUE' FLAG
183F 5058 STA SOFT_CTRL_REG ; ...
1842 5059 STC
1843 5060 CMC
1844 5061 RET
1845 5062
```

ZZ-ECKAA-8.7 V08.07
ECKAA
V08.07

F 11
RETURN ACTION 11-FEB-1986 Fiche 1 Frame F11 Sequence 135
VAX-11/750 MICRO DIAGNOSTIC MONITOR 11-FEB-1986 10:07:52 VAX/VMS Macro V04-00 Page 110
RETURN ACTION ROUTINE 11-FEB-1986 10:06:37 [VAX750.MONITOR]ECKAA.MAR;811 (43)

```
1845 5064 .SBTTL " RETURN ACTION ROUTINE
1845 5065 ;++
1845 5066 ;
1845 5067 ; FUNCTIONAL DESCRIPTION:
1845 5068 ;
1845 5069 ; THIS ROUTINE CAUSES A RETURN TO THE RDM SOFTWARE. THIS IS DONE
1845 5070 ; BY CALLING THE 'ABORT' ROUTINE.
1845 5071 ;
1845 5072 ; CALLING SEQUENCE:
1845 5073 ;
1845 5074 ; CALL RETURN_ACTION
1845 5075 ;
1845 5076 ;--
1845 5077 ;
1845 5078 RETURN_ACTION:
1845 5079 ABORT
1848 5080
```

```
1848 5082      .SBTTL      LOOP ACTION ROUTINE
1848 5083      ;**
1848 5084      :
1848 5085      : FUNCTION DESCRIPTION:
1848 5086      :
1848 5087      :     THIS ROUTINE SETS THE 'LOOP' AND 'NER' AND CLEARS THE 'HALT'
1848 5088      :     FLAG IN THE PROGRAM CONTROL REGISTER. IT ALSO TRANSFERS THE
1848 5089      :     THE CONTENTS OF 'CURRENT_PC' TO 'ERROR_PC' FOR THE IFERROR ROUTINE.
1848 5090      :
1848 5091      : CALLING SEQUENCE:
1848 5092      :
1848 5093      :     CALL      LOOP_ACTION
1848 5094      :
1848 5095      :--
1848 5096      :
1848 5097      LOOP_ACTION:
1848 5098          LDA          PROG_CTRL_REG      ; GET THE CURRENT FLAGS
1848 5099          ORI          <NER_FLAG!LOOP_FLAG> ; SET THE NER & LOOP FLAGS
184D 5100          ANI          <^CHALT_FLAG>     ; CLEAR THE 'HALT' FLAG
184F 5101          STA          PROG_CTRL_REG      ;
1852 5102          LDA          SOFT_CTRL_REG     ; NOW SET THE 'CONTINUE' FLAG
1855 5103          ORI          CONTINUE_FLAG    ; TO SIMULATE A 'CONTINUE'
1857 5104          STA          SOFT_CTRL_REG     ; COMMAND
185A 5105          LHLD         CURRENT_PC       ; GET THE CURRENT PC
185D 5106          SHLD        ERROR_PC         ; SAVE IN ERROR_PC
1860 5107          STC
1861 5108          CMC
1862 5109          RET
1863 5110
```

```

1863 5112      .SBTTL      SET ACTION ROUTINE
1863 5113      :++
1863 5114      :
1863 5115      : FUNCTIONAL DESCRIPTION:
1863 5116      :
1863 5117      : THIS ROUTINE EXECUTES THE MANY FORMS OF THE 'SET' COMMAND. THE
1863 5118      : FORMS SUPPORTED ARE:
1863 5119      :
1863 5120      :           SET FLAG
1863 5121      :           SET SOMM
1863 5122      :           SET CF
1863 5123      :           SET STEP
1863 5124      :
1863 5125      : INPUT PARAMETERS:
1863 5126      :
1863 5127      : REGISTER PAIR D & E - CONTAIN ADDRESS OF THE PARSE TABLE.
1863 5128      :
1863 5129      : IMPLICIT INPUTS:
1863 5130      :
1863 5131      : NONE
1863 5132      :
1863 5133      : OUTPUT PARAMETERS:
1863 5134      :
1863 5135      : NONE
1863 5136      :
1863 5137      : IMPLICIT OUTPUTS:
1863 5138      :
1863 5139      : NONE
1863 5140      :
1863 5141      : COMPLETION CODES:
1863 5142      :
1863 5143      : C BIT SET   - SYNTAX ERROR IN SET COMMAND
1863 5144      : C BIT CLEAR - SET COMMAND EXECUTED SUCCESSFULLY
1863 5145      :
1863 5146      :--
1863 5147      :
1863 5148      SET_ACTION:
1863 5149      XCHG          ; PUT PARSE TBL ADDR IN H & L
1864 5150      MVI          C,3      ; GET CONSTANT TO INCREMENT PARSE
1866 5151      MVI          B,0      ; TABLE ADDRESS
1868 5152      MOV          A,M      ; GET A KEYWORD
1869 5153      ANI          <^X7F> ; DISCARD KEYWORD FLAG BIT
186B 5154      INX          H        ; POINT AT NEXT ELEMENT OF PARSE TBL
186C 5155      CPI          <^X55> ; END OF PARSE TABLE?
186E 5156      JZ          SET_ERR  ; BRANCH IF YES
1871 5157      CPI          FLAG_KEY ; IS IT A 'FLAG' SET?
1873 5158      JZ          SFLAG_SUB_ACT ; BRANCH IF YES
1876 5159      CPI          STEP_KEY ; IS IT A 'STEP' SET?
1878 5160      JZ          SSTEP_SUB_ACT ; BRANCH IF YES
187B 5161      CPI          SOMM_KEY ; IS IT A 'SOMM' SET?
187D 5162      JZ          SSOMM_SUB_ACT ; BRANCH IF YES
1880 5163      CPI          CF_KEY   ; IS IT A 'CF' SET?
1882 5164      JNZ         SET_ERR   ; BRANCH IF NO
1885 5165      :
1885 5166      : CONTROL FILE (CF) KEYWORD RECOGNIZED. GET THE DCS ADDRESS AND BIT NUMBER

```

```
1885 5167 ; AND LOAD THE CONTROL FILE.
1885 5168 ;
1885 5169      DAD      B      ; POINT AT DCS ADDRESS
1886 5170      MOV      A,M    ; GET THE ADDRESS
1887 5171      STA      SETCLR_CF_ADDR ; SAVE
188A 5172      INX      H      ; INCREMENT PARSE TBL PTR
188B 5173      DAD      B      ; POINT AT BIT(S) TO SET
188C 5174      MOV      A,M    ; GET THE DATA
188D 5175      CALL     BIT_NMB_TO_MSK ; DECODE THE BIT NUMBER
1890 5176      STA      SETCLR_CF_DATA ; SAVE
1893 5177      CALL     SET_DCS_CF   ; SET THE BITS IN THE CONTROL FILE
1896 5178      JMP      SET_DONE    ; EXIT
1899 5179 ;
1899 5180 ; FLAG KEYWORD RECOGNIZED. SET THE SPECIFIED FLAG.
1899 5181 ;
1899 5182 SFLAG_SUB_ACT:
1899 5183      MOV      A,M      ; GET THE KEYWORD FLAG NAME
189A 5184      ANI      <^X7F> ; DISCARD KEYWORD FLAG BIT
189C 5185      INX      H      ; POINT AT NEXT ELEMENT OF PARSE TBL
189D 5186      CPI      <^X55> ; END OF PARSE TABLE?
189F 5187      JZ       SET_DONE  ; BRANCH IF YES
18A2 5188      CALL     DECODE_FLAG_NAM ; GO DECODE THE FLAG NAME
18A5 5189      JC       SET_ERR   ; EXIT WITH C SET IF ERROR
18A8 5190      LDA      PROG_CTRL_REG ; GET THE PROGRAM CONTROL FLAGS
18AB 5191      ORA      C        ; OR IN THE NEW FLAG
18AC 5192      STA      PROG_CTRL_REG ; SAVE
18AF 5193 ;
18AF 5194 ; CHECK IF QA FLAG.
18AF 5195 ;
18AF 5196      MOV      A,C      ; QA FLAG?
18B0 5197      CPI      QA_FLAG   ; QA FLAG?
18B2 5198      JNZ     SFLAG_SUB_ACT ; BRANCH IF NO
18B5 5199 ;
18B5 5200 ; QA FLAG, CHECK IF LOOP COUNT SPECIFIED
18B5 5201 ;
18B5 5202      MVI      A,-1      ; INIT QA COUNT
18B7 5203      STA      QA_COUNT   ; ...
18BA 5204      STA      QA_COUNT+1 ; ...
18BD 5205      MOV      A,M      ; GET NEXT ELEMENT IN PARSE TABLE
18BE 5206      ORA      A        ; SET THE CONDITION CODES
18BF 5207      JM      SFLAG_SUB_ACT ; BRANCH IF KEYWORD
18C2 5208      CPI      <^X55> ; END OF PARSE TABLE?
18C4 5209      JZ       SET_DONE  ; BRANCH IF YES
18C7 5210      INX      H      ; POINT TO BYTE 1 OF LOOP COUNT
18C8 5211      INX      H      ; ...
18C9 5212      MOV      A,M      ; GET HIGH BYTE OF LOOP COUNT
18CA 5213      STA      QA_COUNT+1 ; SAVE
18CD 5214      INX      H      ; ...
18CE 5215      MOV      A,M      ; GET BYTE 0 OF COUNT
18CF 5216      STA      QA_COUNT   ; SAVE
18D2 5217      INX      H      ; POINT TO NEXT ENTRY IN PARSE TABLE
18D3 5218      JMP      SFLAG_SUB_ACT ; SEE IF MORE FLAGS
18D6 5219 ;
18D6 5220 ; STEP KEYWORD WAS RECOGNIZED. DECODE THE NEXT KEYWORD AND SET THE
18D6 5221 ; APPROPRIATE BIT IN SOFTWARE CONTROL REG. 1.
```

```
18D6 5222 ;  
18D6 5223 SSTEP_SUB_ACT:  
18D6 5224     MOV      A,M          ; GET THE KEYWORD  
18D7 5225     ANI      <^X7F>      ; DISCARD KEYWORD FLAG BIT  
18D9 5226     INX      H          ; POINT AT NEXT ELEMENT IN PARSE TBL  
18DA 5227     CPI      <^X55>      ; END OF PARSE TABLE?  
18DC 5228     JZ       SET_ERR      ; BRANCH IF YES  
18DF 5229     MVI      D,CYCLE_FLAG ; INIT 'D' REG WITH CYCLE FLAG  
18E1 5230     CPI      CYCLE_KEY    ; IS IT 'CYCLE'?  
18E3 5231     JZ       1$          ; BRANCH IF YES  
18E6 5232     MVI      D,TICK_FLAG  ; INIT 'D' REG WITH TICK FLAG  
18E8 5233     CPI      TICK_KEY    ; IS IT 'TICK'?  
18EA 5234     JZ       1$          ; BRANCH IF YES  
18ED 5235     MVI      D,INSTR_FLAG ; INIT 'D' REG WITH INSTRUCTION FLAG  
18EF 5236     CPI      INSTR_KEY    ; IS IT 'INSTRUCTION'?  
18F1 5237     JNZ      SET_ERR      ; BRANCH IF NO  
18F4 5238 ;  
18F4 5239 ; SET STEP INSTRUCTION. SEE IF A SPECIAL TEST PC WAS SPECIFIED.  
18F4 5240 ;  
18F4 5241     MVI      A,-1          ; INIT THE SPECIAL TEST PC  
18F6 5242     STA      SPEC_TEST_PC+1 ;  
18F9 5243     MOV      A,M          ; GET NEXT ELEMENT IN PARSE TABLE  
18FA 5244     CPI      <^X55>      ; END OF PARSE TABLE?  
18FC 5245     JZ       1$          ; BRANCH IF YES  
18FF 5246     INX      H          ; POINT AT SPECIFIED TEST PC  
1900 5247     INX      H          ;  
1901 5248     MOV      A,M          ; GET UPPER ORDER BITS  
1902 5249     STA      SPEC_TEST_PC+1 ; SAVE  
1905 5250     INX      H          ; POINT AT LOW BITS  
1906 5251     MOV      A,M          ; GET THEM  
1907 5252     STA      SPEC_TEST_PC ; SAVE  
190A 5253 1$: LDA      SOFT_CTRL_REGA ; GET CONTROL REGISTER 1  
190D 5254     ANI      <^C<CYCLE_FLAG!-  
190D 5255     TICK_FLAG>>        ; CLEAR BOTH CYCLE AND TICK FLAGS  
190F 5256     ORA      D          ; OR IN THE APPROPRIATE BIT  
1910 5257     STA      SOFT_CTRL_REGA ; SAVE  
1913 5258     JMP      SET_DONE     ; EXIT  
1916 5259 ;  
1916 5260 ; SOMM KEYWORD WAS RECOGNIZED. SET THE STOP ON MATCH BIT AND LOAD THE  
1916 5261 ; ADDRESS INTO THE MATCH REGISTER IF IT WAS SPECIFIED.  
1916 5262 ;  
1916 5263 SSOMM_SUB_ACT:  
1916 5264     LDA      SOFT_CTRL_REGA    ; SET THE SOMM FLAG  
1919 5265     ORI      SOMM_FLAG      ; ...  
191B 5266     STA      SOFT_CTRL_REGA    ; ...  
191E 5267 ;  
191E 5268 ; SET THE STOP ON MATCH BIT  
191E 5269 ;  
191E 5270     LDA      DCS_CTRL_RE_COPY ;  
1921 5271     ORI      HALT_ON_MATCH ;  
1923 5272     STA      DCS_CTRL_RE_COPY ;  
1926 5273     STA      DCS_CTRL_REG      ;  
1929 5274 ;  
1929 5275 ; CHECK IF ADDRESS SPECIFIED  
1929 5276 ;
```

```
1929 5277      MOV      A,M          ; GET NEXT KEYWORD
192A 5278      CPI      <^X55>        ; END OF PARSE TABLE?
192C 5279      JZ       SET_DONE     ; BRANCH IF YES
192F 5280      ;
192F 5281      ; GET THE ADDRESS AND LOAD THE MATCH REGISTER
192F 5282      ;
192F 5283      INX      H          ; POINT TO THE ADDRESS
1930 5284      INX      H          ; BYTE 1
1931 5285      MOV      A,M          ; GET UPPER BYTE OF ADDRESS
1932 5286      STA      ADDR_MATCH_HI ; LOAD ADDRESS MATCH REG
1935 5287      STA      SOMM_ADDRESS+1 ; AND SAVE
1938 5288      INX      H          ; POINT AT BYTE 0
1939 5289      MOV      A,M          ; GET BYTE 0 OF ADDRESS
193A 5290      STA      ADDR_MATCH_LO ; LOAD ADDRESS MATCH LO
193D 5291      STA      SOMM_ADDRESS ; AND SAVE
1940 5292      ;
1940 5293      ; EXIT WITH C BIT CLEAR
1940 5294      ;
1940 5295      SET_DONE:
1940 5296      STC
1941 5297      CMC
1942 5298      RET
1943 5299      ;
1943 5300      ; EXIT WITH C BIT SET
1943 5301      ;
1943 5302      SET_ERR:
1943 5303      STC
1944 5304      RET
1945 5305
```

```
1945 5307      .SBTTL      CLEAR ACTION ROUTINE
1945 5308      ;++
1945 5309      ;
1945 5310      ; FUNCTIONAL DESCRIPTION:
1945 5311      ;
1945 5312      ;     THIS ROUTINE EXECUTES THE MANY FORMS OF THE 'CLEAR' COMMAND.
1945 5313      ;     THE FORMS SUPPORTED ARE:
1945 5314      ;
1945 5315      ;             CLEAR FLAG
1945 5316      ;             CLEAR SOMM
1945 5317      ;             CLEAR CF
1945 5318      ;
1945 5319      ; CALLING SEQUENCE:
1945 5320      ;
1945 5321      ;     CALL      CLEAR_ACTION
1945 5322      ;
1945 5323      ; INPUT PARAMETERS:
1945 5324      ;
1945 5325      ;     REGISTER PAIR D & E - CONTAINS THE ADDRESS OF THE PARSE TABLE.
1945 5326      ;
1945 5327      ; IMPLICIT INPUTS:
1945 5328      ;
1945 5329      ;     NONE
1945 5330      ;
1945 5331      ; OUTPUT PARAMETERS:
1945 5332      ;
1945 5333      ;     NONE
1945 5334      ;
1945 5335      ; IMPLICIT OUTPUTS:
1945 5336      ;
1945 5337      ;     NONE
1945 5338      ;
1945 5339      ; COMPLETION CODES:
1945 5340      ;
1945 5341      ;     C BIT SET  - SYNTAX ERROR IN CLEAR COMMAND
1945 5342      ;     C BIT CLEAR - CLEAR COMMAND EXECUTED SUCCESSFULLY
1945 5343      ;
1945 5344      ;--
1945 5345      ;
1945 5346      CLEAR_ACTION:
1945 5347      XCHG      ; PUT PARSE TABLE ADDR IN H & L
1946 5348      MVI      B,0      ; PUT A CONSTANT 3 IN B & C REG
1948 5349      MVI      C,3
194A 5350      MOV      A,M
194B 5351      ANI      <^X7F>    ; GET KEYWORD CODE
194D 5352      INX      ; DISCARD KEYWORD FLAG BIT
194E 5353      CPI      <^X55>    ; POINT AT NEXT ENTRY IN PARSE TBL
1950 5354      JZ      CLEAR_ERR  ; END OF PARSE TABLE?
1953 5355      CPI      FLAG_KEY  ; BRANCH IF YES
1955 5356      JZ      CFLAG_SUB_ACT ; IS IT A 'FLAG' CLEAR?
1958 5357      CPI      CF_KEY    ; BRANCH IF YES
195A 5358      JZ      CCF_SUB_ACT  ; IS IT A 'CF' CLEAR?
195D 5359      CPI      SOMM_KEY  ; BRANCH IF YES
195F 5360      JNZ     CLEAR_ERR  ; IS TI A 'SOMM' CLEAR?
1962 5361      ;           ; BRANCH IF NO
```

```
1962 5362 ; SOMM KEYWORD WAS DETECTED. CLEAR THE SOMM FLAG AND THE HALT ON MATCH BIT
1962 5363 ; AND LOAD THE MATCH REGISTER IF AN ADDRESS IS SPECIFIED
1962 5364 ;
1962 5365 LDA SOFT_CTRL_REGA ; GET THE SOMM FLAG
1965 5366 ANI <^CSOMM_FLAG> ; CLEAR IT
1967 5367 STA SOFT_CTRL_REGA ; ...
196A 5368 LDA DCS_CTRL_RE_CPY ; CLEAR THE HALT ON MATCH BIT
196D 5369 ANI <^CHALT_ON_MATCH> ; ...
196F 5370 STA DCS_CTRL_RE_CPY ; ...
1972 5371 STA DCS_CTRL_REG ; ...
1975 5372 ;
1975 5373 ; NOW LOAD THE MATCH REGISTER IF AN ADDRESS IS SPECIFIED.
1975 5374 ;
1975 5375 1$: MOV A,M ; GET NEXT PARSE TABLE ENTRY
1976 5376 CPI <^X55> ; END OF PARSE TABLE?
1978 5377 JZ CLEAR_DONE ; BRANCH IF YES
197B 5378 INX H
197C 5379 INX H
197D 5380 MOV A,M ; GET THE ADDRESS
197E 5381 STA ADDR_MATCH_HI ; LOAD THE HI ORDER BITS
1981 5382 STA SOMM_ADDRESS+1 ; AND SAVE
1984 5383 INX H
1985 5384 MOV A,M ; GET THE LOW ORDER BITS
1986 5385 STA ADDR_MATCH_LO ; LOAD THE LOW ORDER BITS
1989 5386 STA SOMM_ADDRESS ; AND SAVE
198C 5387 JMP CLEAR_DONE ; EXIT
198F 5388 ;
198F 5389 ; FLAG KEYWORD WAS DETECTED. DECODE AND CLEAR THE APPROPRIATE PROGRAM
198F 5390 ; FLAG.
198F 5391 ;
198F 5392 CFLAG_SUB_ACT:
198F 5393 MOV A,M ; GET THE KEYWORD FLAG NAME
1990 5394 ANI <^X7F> ; DISCARD KEYWORD FLAG BIT
1992 5395 INX H ; POINT AT NEXT PARSE TBL ENTRY
1993 5396 CPI <^X55> ; END OF PARSE TABLE?
1995 5397 JZ CLEAR_DONE ; BRANCH IF YES
1998 5398 CALL DECODE_FLAG_NAM ; DECODE THE FLAG NAME
199B 5399 JC CLEAR_ERR ; BRANCH IF ERROR
199E 5400 MOV A,C ; GET THE FLAG BIT
199F 5401 CMA ; INVERT FOR THE CLEAR OPERATION
19A0 5402 MOV C,A ; SAVE
19A1 5403 LDA PROG_CTRL_REG ; GET PROGRAM CONTROL FLAGS
19A4 5404 ANA C ; CLEAR THE FLAG
19A5 5405 STA PROG_CTRL_REG ; SAVE
19A8 5406 JMP CFLAG_SUB_ACT ; SEE IF MORE FLAGS TO CLEAR
19AB 5407 ;
19AB 5408 ; CONTROL FILE (CF) KEYWORD WAS RECOGNIZED. GET THE ADDRESS AND DATA AND
19AB 5409 ; CLEAR THE CONTROL FILE.
19AB 5410 ;
19AB 5411 CCF_SUB_ACT:
19AB 5412 DAD B ; POINT AT THE ADDRESS
19AC 5413 MOV A,M ; GET THE ADDRESS
19AD 5414 STA SETCLR_CF_ADDR ; SAVE
19B0 5415 INX H
19B1 5416 DAD B ; POINT AT THE DATA
```

```
19B2 5417      MOV      A,M      ; GET THE DATA
19B3 5418      CALL     BIT_NMB_TO_MSK ; DECODE THE BIT VALUE
19B6 5419      STA     SETCLR_CF_DATA ; SAVE
19B9 5420      CALL     CLEAR_DCS_CF ; CLEAR THE SPECIFIED BITS
19BC 5421      ;
19BC 5422      ; EXIT WITH C BIT CLEAR
19BC 5423      ;
19BC 5424      CLEAR_DONE:
19BC 5425      STC
19BD 5426      CMC
19BE 5427      RET
19BF 5428      ;
19BF 5429      ; EXIT WITH C BIT SET
19BF 5430      ;
19BF 5431      CLEAR_ERR:
19BF 5432      STC
19C0 5433      RET
19C1 5434
```

```

19C1 5436      .SBTTL      EXAMINE ACTION ROUTINE
19C1 5437      ;++
19C1 5438      ;
19C1 5439      ; FUNCTIONAL DESCRIPTION:
19C1 5440      ;
19C1 5441      ;     THIS ROUTINE EXECUTES THE MANY FORMS OF THE 'EXAMINE' COMMAND.
19C1 5442      ;     THE FORMS SUPPORTED ARE:
19C1 5443      ;
19C1 5444      ;           EXAMINE RTEMPS:<2F:0>           EXAMINE MTEMPS:<F:0>
19C1 5445      ;           EXAMINE PC           EXAMINE VA           EXAMINE MDR
19C1 5446      ;           EXAMINE PBACK      EXAMINE MA           EXAMINE WDR
19C1 5447      ;           EXAMINE SR:<n>      EXAMINE PSL           EXAMINE ST
19C1 5448      ;           EXAMINE SF
19C1 5449      ;
19C1 5450      ; CALLING SEQUENCE:
19C1 5451      ;
19C1 5452      ;     CALL      EXAMINE_ACTION
19C1 5453      ;
19C1 5454      ; INPUT PARAMETERS:
19C1 5455      ;
19C1 5456      ;     REGISTER PAIR D & E - CONTAINS THE ADDRESS OF THE PARSE TABLE.
19C1 5457      ;
19C1 5458      ; IMPLICIT INPUTS:
19C1 5459      ;
19C1 5460      ;     NONE
19C1 5461      ;
19C1 5462      ; OUTPUT PARAMETERS:
19C1 5463      ;
19C1 5464      ;     NONE
19C1 5465      ;
19C1 5466      ; IMPLICIT OUTPUTS:
19C1 5467      ;
19C1 5468      ;     NONE
19C1 5469      ;
19C1 5470      ; COMPLETION CODES:
19C1 5471      ;
19C1 5472      ;     C BIT SET   - SYNTAX ERROR IN COMMAND
19C1 5473      ;     C BIT CLEAR - COMMAND EXECUTED SUCCESSFULLY
19C1 5474      ;
19C1 5475      ; --
19C1 5476      ;
19C1 5477      EXAMINE_ACTION:
19C1 5478      PUSH      D ; SAVE D & E
19C2 5479      WRITES_DCS DCS_SCRATCH_ADR+1,-; INIT DCS SCRATCH LOCATION
19C2 5480      NOP_MICRO_WORD,1;
19D5 5481      LXI      H,0 ; INIT 'EXAMINE_TEMP'
19D8 5482      SHLD     EXAMINE_TEMP ; ...
19DB 5483      SHLD     EXAMINE_TEMP+2 ; ...
19DE 5484      POP      H ; PUT PARSE TABLE ADDRESS IN H & L
19DF 5485      MVI      B,0 ; PUT A CONSTANT 3 IN B & C REG
19E1 5486      MVI      C,3 ; ...
19E3 5487      MOV      A,M ; GET KEYWORD CODE
19E4 5488      ANI      <^X7F> ; DISCARD KEYWORD FLAG BIT
19E6 5489      INX      H ; POINT AT NEXT ENTRY IN PARSE TBL
19E7 5490      CPI      <^X55> ; END OF PARSE TABLE?
  
```

```

19E9 5491 JZ EXAMINE_ERR ; BRANCH IF YES
19EC 5492 CPI RT_KEY ; RTEMP?
19EE 5493 JZ EXAMINE_RT
19F1 5494 CPI MT_KEY ; MTEMP?
19F3 5495 JZ EXAMINE_MT
19F6 5496 CPI PC_KEY ; PC?
19F8 5497 JZ EXAMINE_PC
19FB 5498 CPI VA_KEY ; VA?
19FD 5499 JZ EXAMINE_VA
1A00 5500 CPI MD_KEY ; MDR?
1A02 5501 JZ EXAMINE_MD
1A05 5502 CPI PB_KEY ; PC BACKUP?
1A07 5503 JZ EXAMINE_PB
1A0A 5504 CPI MA_KEY ; MA?
1A0C 5505 JZ EXAMINE_MA
1A0F 5506 CPI WD_KEY ; WDR?
1A11 5507 JZ EXAMINE_WD
1A14 5508 CPI SR_KEY ; STATUS & CONTROL?
1A16 5509 JZ EXAMINE_SR
1A19 5510 CPI PS_KEY ; PSL?
1A1B 5511 JZ EXAMINE_PS
1A1E 5512 CPI ST_KEY ; STEP COUNTER?
1A20 5513 JZ EXAMINE_ST
1A23 5514 CPI SF_KEY ; FLAGS?
1A25 5515 JZ EXAMINE_SF
1A28 5516 JMP EXAMINE_ERR ; INVALID KEYWORD
1A2B 5517 ;
1A2B 5518 ; EXAMINE R TEMPS
1A2B 5519 ;
1A2B 5520 EXAMINE_RT:
1A2B 5521 MOV A,M ; GET NEXT ELEMENT IN PARSE TABLE
1A2C 5522 CPI <^X55> ; REGISTER NUMBER SPECIFIED?
1A2E 5523 JZ EXAMINE_ERR ; BRANCH IF NO
1A31 5524 DAD B ; POINT AT NUMBER
1A32 5525 MOV A,M ;
1A33 5526 CALL READ_RTEMP ; GET CONTENTS OF REGISTER
1A36 5527 JMP EX_DISP_WH ; DISPLAY WH REGISTER
1A39 5528 ;
1A39 5529 ; EXAMINE M TEMPS
1A39 5530 ;
1A39 5531 EXAMINE_MT:
1A39 5532 MOV A,M ; GET NEXT ELEMENT IN PARSE TABLE
1A3A 5533 CPI <^X55> ; REGISTER NUMBER SPECIFIED?
1A3C 5534 JZ EXAMINE_ERR ; BRANCH IF NO
1A3F 5535 DAD B ; POINT AT ADDRESS
1A40 5536 MOV A,M ; ...
1A41 5537 MTEMP_COMMON:
1A41 5538 CALL READ_MTEMP ; GET CONTENTS OF REGISTER
1A44 5539 JMP EX_DISP_WH ; DISPLAY WH REGISTER
1A47 5540 ;
1A47 5541 ; EXAMINE PC
1A47 5542 ;
1A47 5543 EXAMINE_PC:
1A47 5544 MVI A,<^X1A> ; GET MSRC FIELD VALUE
1A49 5545 JMP MTEMP_COMMON ; READ THE REGISTER

```

```

1A4C 5546 ;
1A4C 5547 ; EXAMINE VA:
1A4C 5548 ;
1A4C 5549 EXAMINE_VA:
1A4C 5550     MVI           A,<^X1B>      ; GET MSRC FIELD VALUE
1A4E 5551     JMP           MTEMP_COMMON ; READ THE REGISTER
1A51 5552 ;
1A51 5553 ; EXAMINE MDR
1A51 5554 ;
1A51 5555 EXAMINE_MD:
1A51 5556     MVI           A,<^X12>      ; GET MSRC FIELD VALUE
1A53 5557     JMP           MTEMP_COMMON ; READ THE REGISTER
1A56 5558 ;
1A56 5559 ; EXAMINE PC BACKUP
1A56 5560 ;
1A56 5561 EXAMINE_PB:
1A56 5562     MVI           A,<^X19>      ; GET MSRC FIELD VALUE
1A58 5563     JMP           MTEMP_COMMON ; READ THE REGISTER
1A5B 5564 ;
1A5B 5565 ; EXAMINE MA
1A5B 5566 ;
1A5B 5567 EXAMINE_MA:
1A5B 5568     MVI           A,<^X18>      ; GET MSRC FIELD VALUE
1A5D 5569     JMP           MTEMP_COMMON ; READ THE REGISTER
1A60 5570 ;
1A60 5571 ; EXAMINE WDR
1A60 5572 ;
1A60 5573 EXAMINE_WD:
1A60 5574     WRITE$DCS      DCS_SCRATCH_ADR,-
1A60 5575     RDM_WD_MIC_WORD.1 ; LOAD THE MICRO WORDS
1A73 5576     EXECUTE      DCS_SCRATCH_ADR.1 ; EXECUTE THE MICRO INSTRUCTIONS
1A86 5577     JMP           EX_DISP_WH     ; DISPLAY WH REGISTER
1A89 5578 ;
1A89 5579 ; EXAMINE STATUS AND CONTROL
1A89 5580 ;
1A89 5581 EXAMINE_SR:
1A89 5582     MOV           A,M             ; GET NEXT ELEMENT IN PARSE TABLE
1A8A 5583     CPI           <^X55>        ; REGISTER NUMBER SPECIFIED?
1A8C 5584     JZ           EXAMINE_ERR    ; BRANCH IF NO
1A8F 5585     DAD           B             ; POINT AT ADDRESS
1A90 5586     MOV           A,M             ;
1A91 5587     STA           WD_REG_BYTE_3 ; PUT IN WD REGISTER
1A94 5588     WRITE$DCS      DCS_SCRATCH_ADR,-
1A94 5589     RDM_SR_MIC_WORD.2 ; LOAD THE MICRO WORDS
1AA7 5590     WRITE$DCS      DCS_SCRATCH_ADR+2,-
1AA7 5591     NOP_MICRO_WORD.1 ; ...
1ABA 5592     EXECUTE      DCS_SCRATCH_ADR.2 ; EXECUTE THE MICRO INSTRUCTIONS
1ACD 5593     LDA           WH_REG_BYTE_3 ; GET RETURNED DATA
1AD0 5594     MVI           B,15         ; GET MASK
1AD2 5595     JMP           MASK_DISPLAY  ; DISPLAY WH REGISTER
1AD5 5596 ;
1AD5 5597 ; EXAMINE PSL
1AD5 5598 ;
1AD5 5599 EXAMINE_PS:
1AD5 5600     WRITE$DCS      DCS_SCRATCH_ADR,-

```

```
1AD5 5601 RDM_PS_MIC_WORD,1; LOAD THE MICRO WORDS
1AE8 5602 EXECUTE DCS_SCRATCH_ADR,1; EXECUTE THE MICRO INSTRUCTIONS
1AFB 5603 LDA WH_REG_BYTE_2 ; GET BYTE 2
1AFE 5604 ANI <^XDF> ; MASK
1B00 5605 STA EXAMINE_TEMP+2 ; SAVE FOR DISPLAY
1B03 5606 LDA WH_REG_BYTE_3 ; GET BYTE 3
1B06 5607 ANI <^XCF> ; MASK
1B08 5608 STA EXAMINE_TEMP+3 ; SAVE FOR DISPLAY
1B0B 5609 LDA WH_REG_BYTE_0 ; GET BYTE 0
1B0E 5610 MVI B,255 ; GET MASK
1B10 5611 JMP MASK_DISPLAY ; DISPLAY 'EXAMINE_TEMP'
1B13 5612 ;
1B13 5613 ; EXAMINE STEP COUNTER
1B13 5614 ;
1B13 5615 EXAMINE_ST:
1B13 5616 WRITE$DCS DCS_SCRATCH_ADR,-
1B13 5617 RDM_ST_MIC_WORD,1; LOAD THE MICRO WORDS
1B26 5618 EXECUTE DCS_SCRATCH_ADR,1; EXECUTE THE MICRO INSTRUCTIONS
1B39 5619 LDA WH_REG_BYTE_0 ; GET DATA READ
1B3C 5620 MVI B,31 ; ONLY NEED BITS<4:0>
1B3E 5621 JMP MASK_DISPLAY ; DISPLAY DATA
1B41 5622 ;
1B41 5623 ; EXAMINE FLAGS
1B41 5624 ;
1B41 5625 EXAMINE_SF:
1B41 5626 WRITE$DCS DCS_SCRATCH_ADR,-
1B41 5627 RDM_SF_MIC_WORD,1; LOAD THE MICRO WORDS
1B54 5628 EXECUTE DCS_SCRATCH_ADR,1; EXECUTE THE MICRO INSTRUCTIONS
1B67 5629 LDA WH_REG_BYTE_0 ; GET DATA READ
1B6A 5630 MVI B,63 ; ONLY NEED BITS<5:0>
1B6C 5631 ; JMP MASK_DISPLAY ; DISPLAY DATA
1B6C 5632 ;
1B6C 5633 ;
1B6C 5634 ; MASK AND DISPLAY A REGISTER
1B6C 5635 ;
1B6C 5636 MASK_DISPLAY:
1B6C 5637 ANA B ; MASK THE DATA
1B6D 5638 STA EXAMINE_TEMP ; SAVE
1B70 5639 LXI H,EXAMINE_TEMP ; GET ADDRESS OF DATA
1B73 5640 JMP EXAMINE_DISPLAY ; DISPLAY THE DATA
1B76 5641 ;
1B76 5642 ; DISPLAY WH REGISTER
1B76 5643 ;
1B76 5644 EX_DISP_WH:
1B76 5645 LXI H,WH_REG_BYTE_0 ; GET ADDRESS OF DATA
1B79 5646 ;
1B79 5647 ; DISPLAY REGISTER WHOSE POINTER IS IN H & L
1B79 5648 ;
1B79 5649 EXAMINE_DISPLAY:
1B79 5650 CALL DISPLAY_DATA
1B7C 5651 JMP EXAMINE_EX
1B7F 5652 EXAMINE_ERR:
1B7F 5653 STC
1B80 5654 RET
1B81 5655 EXAMINE_EX:
```

ZZ-ECKAA-8.7 V08.07
ECKAA
V08.07

F 12
" EXAMINE ACTION ROUTINE
VAX-11/750 MICRO DIAGNOSTIC MONITOR
EXAMINE ACTION ROUTINE

Fiche 1 Frame F12 Sequence 148
11-FEB-1986 10:07:52 VAX/VMS Macro V04-00 Page 123
11-FEB-1986 10:06:37 [VAX750.MONITOR]ECKAA.MAR;811 (47)

1881	5656	STC
1882	5657	CMC
1883	5658	RET
1884	5659	
1884	5660	DEPOSIT_ACTION:
1884	5661	RET
1885	5662	
1885	5663	

```
1B85 5665 .SBTTL DECODE FLAG NAME ROUTINE
1B85 5666 ;++
1B85 5667 ;
1B85 5668 ; FUNCTIONAL DESCRIPTION:
1B85 5669 ;
1B85 5670 ; THIS ROUTINE DECODES A PROGRAM FLAG NAME INTO A BIT POSITION.
1B85 5671 ; THE FLAG NAME MUST BE ONE OF THE FOLLOWING:
1B85 5672 ;
1B85 5673 ; HALT LOOP NER BELL SA IB QA TR
1B85 5674 ;
1B85 5675 ; CALLING SEQUENCE:
1B85 5676 ;
1B85 5677 ; CALL DECODE_FLAG_NAM
1B85 5678 ;
1B85 5679 ; INPUT PARAMETERS:
1B85 5680 ;
1B85 5681 ; REGISTER A - CONTAINS THE OFFSET POSITION OF THE FLAG NAME
1B85 5682 ; IN THE KEYWORD TABLE.
1B85 5683 ;
1B85 5684 ; IMPLICIT INPUTS:
1B85 5685 ;
1B85 5686 ; NONE
1B85 5687 ;
1B85 5688 ; OUTPUT PARAMETERS:
1B85 5689 ;
1B85 5690 ; REGISTER C - CONTAINS THE FLAG BIT DECODED FROM THE NAME
1B85 5691 ;
1B85 5692 ; IMPLICIT OUTPUTS:
1B85 5693 ;
1B85 5694 ; NONE
1B85 5695 ;
1B85 5696 ; COMPLETION CODES:
1B85 5697 ;
1B85 5698 ; C BIT SET - INDICATES AN ILLEGAL FLAG NAME
1B85 5699 ; C BIT CLEAR - CONVERSION IS COMPLETE
1B85 5700 ;
1B85 5701 ;--
1B85 5702 ;
1B85 5703 DECODE_FLAG_NAM:
1B85 5704 SUI HALT_KEY ; IS IT A LEGAL FLAG NAME?
1B87 5705 JM DECODE_ERR ; BRANCH IF NO
1B8A 5706 CPI TR_KEY+2 ; IS IT LEGAL FLAG NAME?
1B8C 5707 JP DECODE_ERR ; BRANCH IF NO
1B8F 5708 ;
1B8F 5709 ; CONVERT FLAG NAME OFFSET INTO A BIT POSITION.
1B8F 5710 ;
1B8F 5711 RRC ; DIVIDE OFFSET BY 2
1B90 5712 CALL BIT_NMB_TO_MSK ; GET THE BIT MASK
1B93 5713 ;
1B93 5714 ; DECODE COMPLETE. RETURN WITH C BIT CLEAR.
1B93 5715 ;
1B93 5716 2$: MOV C,A ; PUT FLAG BIT IN C REGISTER
1B94 5717 STC
1B95 5718 CMC
1B96 5719 RET
```

ZZ-ECKAA-8.7 V08.07
ECKAA
V08.07

H 12
DECODE FLAG N11-FEB-1986 Fiche 1 Frame H12 Sequence 150
VAX-11/750 MICRO DIAGNOSTIC MONITOR 11-FEB-1986 10:07:52 VAX/VMS Macro V04-00 Page 125
" DECODE FLAG NAME ROUTINE 11-FEB-1986 10:06:37 [VAX750.MONITOR]ECKAA.MAR;811 (48)

```
1B97 5720 ;  
1B97 5721 ; ILLEGAL FLAG NAME. RETURN WITH C BIT SET.  
1B97 5722 ;  
1B97 5723 DECODE_ERR:  
1B97 5724     STC  
1B98 5725     RET  
1B99 5726
```

```
1B99 5728 .SBTTL THE INTERPRETER
1B99 5729 .SBTTL " OP CODE DISPATCH ROUTINE
1B99 5730 ;**
1B99 5731 ;
1B99 5732 ; FUNCTIONAL DESCRIPTION:
1B99 5733 ;
1B99 5734 ; THIS ROUTINE USES THE TEST PC TO PICKUP AN OPCODE AND ITS
1B99 5735 ; ASSOCIATED ARGUMENTS AND CALL THE SUBROUTINE SPECIFIED BY THE
1B99 5736 ; OP CODE.
1B99 5737 ;
1B99 5738 ; BEGIN
1B99 5739 ; IF THE 'SINGLE_INSTR' FLAG IS SET
1B99 5740 ; THEN
1B99 5741 ; IF THE 'LOSS' FLAG IS SET OR THE 'LOST' FLAG IS SET AND THE
1B99 5742 ; CURRENT TEST_PC IS EQUAL TO OR GREATER THAN THE
1B99 5743 ; SPECIFIED TEST_PC OR THERE IS NO SPECIFIED
1B99 5744 ; TEST_PC
1B99 5745 ; THEN CALL THE SINGLE PSEUDO INSTRUCTION ROUTINE
1B99 5746 ; ELSE DISPATCH
1B99 5747 ; ELSE DISPATCH
1B99 5748 ; END
1B99 5749 ;
1B99 5750 ; CALLING SEQUENCE:
1B99 5751 ;
1B99 5752 ; CALL OP_CODE_DISP
1B99 5753 ;
1B99 5754 ; INPUT PARAMETERS:
1B99 5755 ;
1B99 5756 ; NONE
1B99 5757 ;
1B99 5758 ; IMPLICIT INPUTS:
1B99 5759 ;
1B99 5760 ; TEST_PC - CONTAINS THE POINTER TO THE CURRENT PSEUDO INSTR
1B99 5761 ; SPEC_TEST_PC - CONTAINS THE TEST PC TO BEGIN SINGLE INSTRUCTION
1B99 5762 ; SOFT_CTRL_REG - CONTAINS THE LOSS AND LOST FLAGS
1B99 5763 ; SPEC_TEST_NUMB - CONTAINS THE TEST NUMBER FOR LOOP IN SPECIAL TEST
1B99 5764 ; TEST_NUMBER - CONTAINS THE CURRENT TEST NUMBER
1B99 5765 ;
1B99 5766 ; OUTPUT PARAMETERS:
1B99 5767 ;
1B99 5768 ; NONE
1B99 5769 ;
1B99 5770 ; IMPLICIT OUTPUTS:
1B99 5771 ;
1B99 5772 ; NONE
1B99 5773 ;
1B99 5774 ; COMPLETION CODES:
1B99 5775 ;
1B99 5776 ; NONE
1B99 5777 ;
1B99 5778 ;--
1B99 5779 ;
1B99 5780 OP_CODE_DISP:
1B99 5781 ;
1B99 5782 ; CHECK THE SINGLE INSTRUCTION FLAG
```

```

1B99 5783 ;
1B99 5784 ; LDA SOFT_CTRL_REGA ; GET THE SINGLE INSTRUCTION FLAG
1B9C 5785 ; ANI INSTR_FLAG ; IS IT SET?
1B9E 5786 ; JZ 1$ ; BRANCH IF NO
1BA1 5787 ;
1BA1 5788 ; CHECK THE 'LOST' FLAG
1BA1 5789 ;
1BA1 5790 ; LDA SOFT_CTRL_REG ; GET THE LOST FLAG
1BA4 5791 ; ANI LOST_FLAG ; IS IT SET?
1BA6 5792 ; JZ 2$ ; BRANCH IF NO
1BA9 5793 ;
1BA9 5794 ; LOST FLAG IS SET. SEE IF TEST PC
1BA9 5795 ; IS THE SPECIFIED TEST PC.
1BA9 5796 ;
1BA9 5797 ; LHLD TEST_PC ; GET THE CURRENT TEST PC
1BAC 5798 ; XCHG ; SAVE IN D & E
1BAD 5799 ; LHLD C_INIT_TEST_PC ; GET TWO'S COMPLIMENT OF INIT TEST PC
1BB0 5800 ; DAD D ; H & L NOW CONTAIN VIRTUAL TEST PC
1BB1 5801 ; INX H ; ACCOUNT FOR LENGTH BYTE
1BB2 5802 ; XCHG ; SAVE IN D & E
1BB3 5803 ; LHLD TITLE_LENGTH ; GET LENGTH OF TEST TITLE
1BB6 5804 ; DAD D
1BB7 5805 ; XCHG
1BB8 5806 ; LHLD SPEC_TEST_PC ; GET THE SPECIAL TEST PC
1BBB 5807 ; INR H ; HAS ONE BEEN SPECIFIED?
1BBC 5808 ; JZ 2$ ; BRANCH IF NO
1BBF 5809 ; DCR H ; RESTORE SPECIAL TEST PC
1BC0 5810 6$ : MOV A,H ; NOW COMPARE THE SPECIAL TEST
1BC1 5811 ; CMP D ; TEST PC TO THE CURRENT TEST PC
1BC2 5812 ; JC 2$ ; BRANCH IF GREATER THAN
1BC5 5813 ; JNZ 1$ ; BRANCH IF NOT THE SAME
1BC8 5814 ; MOV A,L ; ...
1BC9 5815 ; CMP E ; ...
1BCA 5816 ; JC 2$ ; BRANCH IF GREATER THAN
1BCD 5817 ; JNZ 1$ ; BRANCH IF LESS THAN
1BD0 5818 2$ : CALL SINGLE_INSTR ; GO TO THE SINGLE INSTRUCTION ROUTINE
1BD3 5819 ; CC GET_CMD_LINE ; CALL CMD PARSER IF NOT A SPACE
1BD6 5820 ;
1BD6 5821 ; NOW PICKUP THE OP CODE AND ARGUMENTS TO THE PSEUDO INSTRUCTION
1BD6 5822 ;
1BD6 5823 1$ : LHLD TEST_PC ; GET THE CURRENT TEST PC
1BD9 5824 ; SHLD CURRENT_PC ; SAVE IN CASE IT'S AN IFERROR INSTR
1BDC 5825 ; MOV C,M ; GET THE OP CODE
1BDD 5826 ; INX H ; POINT AT THE ARGUMENT COUNT
1BDE 5827 ; LXI D,ARG_LIST ; GET POINTER TO ARGUMENT LIST
1BE1 5828 ; MOV A,M ; GET THE NUMBER OF ARGUMENTS
1BE2 5829 ; CPI MAX_ARGUMENTS+1 ; LESS THAN ARGUMENT LIST?
1BE4 5830 ; JM 5$ ; BRANCH IF YES
1BE7 5831 ; MVI A,MAX_ARGUMENTS ; SET AT MAX
1BE9 5832 5$ : MOV B,A ; PUT ARGUMENT COUNT IN B REG
1BEA 5833 ; INX H ; POINT AT FIRST ARGUMENT
1BEB 5834 3$ : DCR B ; DECREMENT ARGUMENT COUNT
1BEC 5835 ; JM 4$ ; BRANCH IF ALL ARGUMENTS PICKED UP
1BEF 5836 ; MOV A,M ; GET AN ARGUMENT
1BF0 5837 ; STAX D ; SAVE IN ARGUMENT LIST

```

```

1BF1 5838      INX      D      ; INCREMENT ARG LIST POINTER
1BF2 5839      INX      H      ; INCREMENT TEST STREAM POINTER
1BF3 5840      JMP      3$     ; CONTINUE
1BF6 5841      ;
1BF6 5842      ; GOT ALL THE ARGUMENTS. UPDATE THE TEST PC
1BF6 5843      ;
1BF6 5844      4$:   SHLD     TEST_PC      ; SAVE THE CURRENT TEST PC
1BF9 5845      ;
1BF9 5846      ; USE THE OPCODE TO GET THE ADDRESS OF THE APPROPRIATE SUB ROUTINE
1BF9 5847      ; AND CALL THE ROUTINE.
1BF9 5848      ;
1BF9 5849      LXI      H,INSTR_TABLE ; GET THE ADDRESS OF THE INSTRUCTION TABLE
1BFC 5850      MVI      B,0      ; INIT B REGISTER
1BFE 5851      DAD      B      ; INDEX TO THE ROUTINE ADDRESS
1BFF 5852      MOV      A,M     ; GET LOW BITS OF ROUTINE ADDRESS
1C00 5853      MOV      E,A     ; PUT IN 'E' REG
1C01 5854      INX      H      ; POINT AT UPPER BITS
1C02 5855      MOV      A,M     ; GET UPPER BITS OF ADDRESS
1C03 5856      MOV      D,A     ; PUT IN D REG
1C04 5857      LXI      B,OP_CODE_DISP ; GET RETURN ADDRESS FOR SIMULATED
1C07 5858      PUSH     B      ; SUB ROUTINE CALL
1C08 5859      XCHG     ; PUT ROUTINE ADDRESS IN H & L
1C09 5860      PCHL     ; JMP TO THE ROUTINE
1C0A 5861

```

```

1COA 5863      .SBTTL      LOAD DCS ROUTINE
1COA 5864      ;++
1COA 5865      ;
1COA 5866      ; FUNCTIONAL DESCRIPTION:
1COA 5867      ;
1COA 5868      ;     THIS ROUTINE INTERPRETS THE 'LOADDCS' PSEUDO INSTRUCTION.
1COA 5869      ;     IT MOVES <WRD CNT> NUMBER OF MICRO WORDS STARTING AT <SRC ADR>
1COA 5870      ;     (INDEXED BY <SRC INDEX> TIMES THE <WRD CNT>) TO
1COA 5871      ;     DCS STARTING AT <DCS ADR>. IF <SRC
1COA 5872      ;     INDEX> IS SPECIFIED, <SRC ADR> IS INDEXED BY MICRO WORDS (11 BYTES)
1COA 5873      ;     TIMES THE CURRENT VALUE-1 OF THE INDEX. THIS ROUTINE CALLS THE
1COA 5874      ;     'WRITE_DCS' ROUTINE.
1COA 5875      ;
1COA 5876      ; CALLING SEQUENCE:
1COA 5877      ;
1COA 5878      ;     LOADDCS <SRC ADR>,[<SRC INDEX>],<DCS ADR>,<WRD CNT>
1COA 5879      ;
1COA 5880      ; IMPLICIT INPUTS:
1COA 5881      ;
1COA 5882      ;     ARGUMENT LIST:
1COA 5883      ;
1COA 5884      ;         OFFSET 0 - SOURCE ADDRESS LOW
1COA 5885      ;         OFFSET 1 - SOURCE ADDRESS HIGH
1COA 5886      ;         OFFSET 2 - SOURCE INDE CODE
1COA 5887      ;         OFFSET 3 - DCS ADDRESS
1COA 5888      ;         OFFSET 4 - MICRO WORD COUNT
1COA 5889      ;
1COA 5890      ; OUTPUT PARAMETERS:
1COA 5891      ;
1COA 5892      ;     NONE
1COA 5893      ;
1COA 5894      ; IMPLICIT OUTPUTS:
1COA 5895      ;
1COA 5896      ;     NONE
1COA 5897      ;
1COA 5898      ; COMPLETION CODES:
1COA 5899      ;
1COA 5900      ;     NONE
1COA 5901      ;
1COA 5902      ;--
1COA 5903      ;
1COA 5904      LOAD_DCS:
1COA 5905      LDA          ARG_LIST+2      ; GET THE SRC INDEX BYTE
1COD 5906      CALL          GET_INDEX_VALUE ; GET THE CURRENT INDEX VALUE
1C10 5907      ;
1C10 5908      ; NOW MULTIPLY BY THE WORD COUNT
1C10 5909      ;
1C10 5910      LDA          ARG_LIST+4      ; GET THE WORD COUNT
1C13 5911      CALL          MULTIPLY_A_BC
1C16 5912      MOV          B,D           ; PUT RESULT IN B & C
1C17 5913      MOV          C,E           ; ...
1C18 5914      ;
1C18 5915      ; INDEX VALUE IS IN B REGISTER. GET THE SRC ADDRESS AND INDEX IT.
1C18 5916      ;
1C18 5917      MVI          A,11           ; MULTIPLY 11 BYTES TIMES THE
  
```

```
1C1A 5918      CALL      MULTIPLY_A_BC      ; CURRENT INDEX VALUE
1C1D 5919      LHLD     ARG_LIST         ; GET THE SOURCE DATA ADDRESS
1C20 5920      DAD      D                 ; INDEX IT
1C21 5921      XCHG
1C22 5922      LHLD     INIT_TEST_PC ; GET INITIAL TEST PC
1C25 5923      DAD      D                 ; RELOCATE SOURCE ADDRESS
1C26 5924      :
1C26 5925      : INDEXED SRC ADDRESS IS IN H & L. GET THE DCS ADDRESS AND WORD COUNT
1C26 5926      :
1C26 5927      3$: LDA      ARG_LIST+4      ; GET THE WORD COUNT
1C29 5928      MOV      B,A          ; PUT IN 'B' REG FOR MACRO CALL
1C2A 5929      LDA      ARG_LIST+3    ; GET THE DCS ADDRESS
1C2D 5930      WRITE$_DCS          ; WRITE THE DCS
1C3A 5931      RET
1C3B 5932
```

```
1C3B 5934 .SBTTL NEW TEST ROUTINE
1C3B 5935 :++
1C3B 5936 :
1C3B 5937 : FUNCTIONAL DESCRIPTION:
1C3B 5938 :
1C3B 5939 : IF THE CONTROL_C_FLAG IS SET
1C3B 5940 : THEN CALL THE GET_CMD_LINE ROUTINE
1C3B 5941 : ELSE
1C3B 5942 : IF THE TSTSPAN_FLAG IS SET AND THE LAST TEST WAS THE SPECIFIED
1C3B 5943 : TEST_SPAN_END TEST
1C3B 5944 : THEN CALL THE GET_CMD_LINE ROUTINE
1C3B 5945 : ELSE TAKE THE NORMAL_EXIT
1C3B 5946 :
1C3B 5947 : IF THE LOST_FLAG IS SET
1C3B 5948 : THEN
1C3B 5949 : IF THE LAST TEST WAS THE SPECIFIED SPEC_TEST_NUMB
1C3B 5950 : THEN
1C3B 5951 : IF THE CONT_FLAG IS CLEAR
1C3B 5952 : THEN
1C3B 5953 : SET THE TEST_PC TO THE VALUE OF LOOP_ADDRESS
1C3B 5954 : CLEAR THE ERROR_FLAG AND LOOP_ERR_FLAG AND EXIT
1C3B 5955 : ELSE CLEAR THE LOST_FLAG AND CONT_FLAG AND TAKE NORMAL_EXIT
1C3B 5956 : ELSE
1C3B 5957 :
1C3B 5958 : NORMAL_EXIT:
1C3B 5959 : ELSE SAVE THE TEST_PC IN LOOP_ADDRESS AND ERROR_LOOP_ADDR
1C3B 5960 : CLEAR THE ERROR_FLAG AND LOOP_ERROR_FLAG
1C3B 5961 : UPDATE THE TEST_NUMBER, AND EXIT
1C3B 5962 :
1C3B 5963 : CALLING SEQUENCE:
1C3B 5964 :
1C3B 5965 : NEWTEST
1C3B 5966 :
1C3B 5967 : IMPLICIT INPUTS:
1C3B 5968 :
1C3B 5969 : ARGUMENT LIST:
1C3B 5970 :
1C3B 5971 : OFFSET 0 - THE TEST NUMBER
1C3B 5972 :
1C3B 5973 : TEST_PC - CONTAINS THE CURRENT TEST PC
1C3B 5974 : TEST_NUMBER - CONTAINS THE TEST NUMBER OF THE PREVIOUS TEST
1C3B 5975 : SPEC_TEST_NUMB - CONTAINS THE TEST NUMBER REQUEST FOR 'LOOP ON TEST'
1C3B 5976 : TEST_SPAN_END - CONTAINS THE ENDING TEST NUMBER FOR A 'TEST SPAN' FUNCTION
1C3B 5977 : LOOP_ADDRESS - CONTAINS THE 'LOOP ADDRESS' OF THE PREVIOUS TEST
1C3B 5978 : ERROR_LOOP_ADDR - CONTAINS THE 'ERROR LOOP ADDRESS' OF THE PREVIOUS TEST
1C3B 5979 :
1C3B 5980 : IMPLICIT OUTPUTS:
1C3B 5981 :
1C3B 5982 : LOOP_ADDRESS - CONTAINS THE 'LOOP ADDRESS' OF THE CURRENT TEST
1C3B 5983 : ERROR_LOOP_ADDR - CONTAINS THE 'ERROR LOOP ADDRESS' OF THE CURRENT TEST
1C3B 5984 : TEST_NUMBER - CONTAINS THE TEST NUMBER OF THE CURRENT TEST
1C3B 5985 :
1C3B 5986 : COMPLETION CODES:
1C3B 5987 :
1C3B 5988 : NONE
```

```

1C3B 5989 ;
1C3B 5990 ;--
1C3B 5991
1C3B 5992 NEW_TEST:
1C3B 5993 ;
1C3B 5994 ; CHECK THE CONTROL_C_FLAG
1C3B 5995 ;
1C3B 5996         LDA          SOFT_CTRL_REG      ; GET THE CONTROL C FLAG
1C3E 5997         ANI          CONTROL_C_FLAG    ; IS IT SET?
1C40 5998         CNZ          GET_CMD_LINE     ; IF YES, CALL THE COMMAND PARSER
1C43 5999 ;
1C43 6000 ; INITIALIZE THE SUB TEST NUMBER
1C43 6001 ;
1C43 6002         XRA          A                  ; GET INITIAL DATA
1C44 6003         STA          SUB_TEST_NUMB    ; INIT THE SUB TEST NUMBER
1C47 6004 ;
1C47 6005 ; INITIALIZE THE CURRENT VALUE OF THE LOOP NAMES
1C47 6006 ;
1C47 6007         CALL         INIT_LOOP_VALUE ;
1C4A 6008 ;
1C4A 6009 ; INITIALIZE THE TRAP FLAG
1C4A 6010 ;
1C4A 6011         LDA          SOFT_CTRL_REGA    ; GET THE FLAGS REGISTER
1C4D 6012         ANI          <^CEXP_UTRAP_FLAG> ; CLEAR THE FLAG
1C4F 6013         STA          SOFT_CTRL_REGA    ; RESTORE THE FLAGS REGISTER
1C52 6014 ;
1C52 6015 ; CHECK THE TSTSPAN_FLAG
1C52 6016 ;
1C52 6017         LDA          SOFT_CTRL_REG      ; GET THE TSTSPAN_FLAG
1C55 6018         ANI          TSTSPAN_FLAG     ; IS IT SET?
1C57 6019         JZ          1$                ; BRANCH IF NO
1C5A 6020         LDA          TEST_NUMBER      ; GET THE NUMBER OF THE LAST TEST
1C5D 6021         MOV          B,A              ; SAVE
1C5E 6022         LDA          TEST_SPAN_END    ; GET THE SPECIFIED END TEST NUMBER
1C61 6023         CMP          B                ; SAME AS LAST TEST NUMBER?
1C62 6024         JNZ          40$             ; BRANCH IF NO
1C65 6025         CALL         GET_CMD_LINE     ; CALL THE COMMAND PARSER
1C68 6026         LDA          SOFT_CTRL_REG    ; CLEAR THE TEST SPAN FLAG
1C6B 6027         ANI          <^C<TSTSPAN_FLAG!- ; AND THE LOST FLAG
1C6B 6028         LOST_FLAG>>                ; ...
1C6D 6029         STA          SOFT_CTRL_REG    ; ...
1C70 6030         JMP          40$             ; IF CONTINUE CMD, TAKE NORMAL EXIT
1C73 6031 ;
1C73 6032 ; CHECK THE 'LOST_FLAG'
1C73 6033 ;
1C73 6034 1$:     LDA          SOFT_CTRL_REG      ; GET THE 'LOST_FLAG'
1C76 6035         ANI          LOST_FLAG        ; IS IT SET?
1C78 6036         JZ          40$             ; BRANCH IF NO
1C7B 6037 ;
1C7B 6038 ; CHECK THE LAST TEST NUMBER AGAINST SPECIFIED NUMBER
1C7B 6039 ;
1C7B 6040         LDA          TEST_NUMBER      ; GET THE NUMBER OF THE LAST TEST
1C7E 6041         MOV          B,A              ; SAVE
1C7F 6042         LDA          SPEC_TEST_NUMB   ; GET THE SPECIFIED TEST NUMBER
1C82 6043         CMP          B                ; IS LAST TEST SPECIFIED TEST?

```

```

1C83 6044          JNZ          40$          ; BRANCH IF NO
1C86 6045          ;
1C86 6046          ; LAST TEST WAS SPECIFIED TEST. CHECK IF 'CONT_FLAG' IS SET.
1C86 6047          ;
1C86 6048          LDA          SOFT_CTRL_REG ; GET THE CONT_FLAG
1C89 6049          ANI          CONT_FLAG   ; IS IT SET?
1C8B 6050          JNZ          5$          ; BRANCH IF YES
1C8E 6051          LHLD         LOOP_ADDRESS ; GET THE LOOP ADDRESS
1C91 6052          SHLD        TEST_PC     ; SET THE TEST PC
1C94 6053          LDA          SOFT_CTRL_REGA ; CLEAR THE ERROR AND LOOP
1C97 6054          ANI          <^C<LOOP_FLAG!- ; ERROR FLAGS
1C97 6055          LOOP_ERROR_FLAG>> ; ...
1C99 6056          STA          SOFT_CTRL_REGA ; ...
1C9C 6057          JMP          45$          ; EXIT
1C9F 6058          ;
1C9F 6059          ; CONT_FLAG WAS SET
1C9F 6060          ;
1C9F 6061          5$: LDA          SOFT_CTRL_REG ; GET THE LOST_FLAG & CONT_FLAG
1CA2 6062          ANI          <^C<LOST_FLAG!- ; CLEAR THEM
1CA2 6063          CONT_FLAG>> ; ...
1CA4 6064          STA          SOFT_CTRL_REG ; ...
1CA7 6065          ;
1CA7 6066          ; NORMAL EXIT
1CA7 6067          ;
1CA7 6068          40$: LHLD        TEST_PC   ; GET THE TEST PC
1CAA 6069          SHLD        LOOP_ADDRESS ; SAVE AS THE LOOP ADDRESS
1CAD 6070          SHLD        ERROR_LOOP_ADDR ; AND THE ERROR LOOP ADDRESS
1CB0 6071          LDA          SOFT_CTRL_REGA ; GET THE ERROR FLAG AND LOOP ERROR FLAG
1CB3 6072          ANI          <^C<ERROR_FLAG!- ; CLEAR THEM
1CB3 6073          LOOP_ERROR_FLAG>> ; ...
1CB5 6074          STA          SOFT_CTRL_REGA ; ...
1CB8 6075          LDA          ARG_LIST    ; GET THE TEST NUMBER
1CBB 6076          STA          TEST_NUMBER ; SAVE
1CBE 6077          45$: RET          ; EXIT
1CBF 6078

```

```

1CBF 6080      .SBTTL " INITIALIZE ROUTINE
1CBF 6081      ;++
1CBF 6082      ;
1CBF 6083      ; FUNCTIONAL DESCRIPTION:
1CBF 6084      ;
1CBF 6085      ;     THIS ROUTINE INITIALIZES THE 11/750 CPU. THIS IS DONE BY SETTING
1CBF 6086      ;     THE 'DC LOW' BIT IN THE RD CONTROL REGISTER WITH MICRO ADDRESS
1CBF 6087      ;     INHIBIT ASCERTED.
1CBF 6088      ;
1CBF 6089      ;     THE RDM CONTROL FILE IS CLEARED, AND THE MASTER_HALT_ENABLE IS
1CBF 6090      ;     TOGGLED. THE V BUS LOAD BIT IS ALSO CLEARED.
1CBF 6091      ;
1CBF 6092      ; CALLING SEQUENCE:
1CBF 6093      ;
1CBF 6094      ;     INITIALIZE
1CBF 6095      ;
1CBF 6096      ; IMPLICIT INPUTS:
1CBF 6097      ;
1CBF 6098      ;     NONE
1CBF 6099      ;
1CBF 6100      ; IMPLICIT OUTPUTS:
1CBF 6101      ;
1CBF 6102      ;     NONE
1CBF 6103      ;
1CBF 6104      ; COMPLETION CODES:
1CBF 6105      ;
1CBF 6106      ;     NONE
1CBF 6107      ;
1CBF 6108      ;--
1CBF 6109      ;
1CBF 6110      INIT_THE_WORLD:
1CBF 6111      CALL          CPU_INIT          ; INIT THE MAINT REG, FORCE DC LOW,
1CC2 6112      ;                                     ; AND FETCH CS LOCATION 0
1CC2 6113      ;
1CC2 6114      ; CHECK IF IN REMOTE MODE
1CC2 6115      ;
1CC2 6116      LDA          FRONT_PNL_2      ; GET REMOTE BIT
1CC5 6117      ANI          REMOTE          ;
1CC7 6118      JZ          10$              ; BRANCH IF NOT REMOTE
1CCA 6119      MVI          A,MAINT_DCS_ENABL; GET DATA TO ENABLE CLOCK INTERRUPTS
1CCC 6120      STA          MAINT_REG        ; ENABLE THE RDM CLOCK
1CCF 6121      ;
1CCF 6122      10$: CALL      DISABLE_CMI     ; DISABLE THE CMI
1CD2 6123      CALL      FETCH_0           ; FETCH CS LOCATION 0
1CD5 6124      LDA          FRONT_PNL_2      ; GET THE REMOTE, FAULT, AND TEST BITS
1CD8 6125      ANI          <REMOTE!FAULT!TEST> ;...
1CDA 6126      ORI          <MASTER_HALT_EN!- ; SET MASTER HALT ENABLE
1CDA 6127      TRAP_HALT_EN> ; AND TRAP HALT ENABLE
1CDC 6128      STA          RD_CTRL_REG      ;
1CDF 6129      RET                          ;
1CE0 6130      ; EXIT
  
```

```

1CEO 6132      .SBTTL      LOOP ROUTINE
1CEO 6133      ;++
1CEO 6134      ;
1CEO 6135      ; FUNCTIONAL DESCRIPTION:
1CEO 6136      ;
1CEO 6137      ; THIS ROUTINE INITIALIZES THE 'LOOP_NAME' PARAMETERS TO THE
1CEO 6138      ; 'STARTING_VALUE', 'END_VALUE', AND 'INCREMENT_VALUE'. IT ALSO
1CEO 6139      ; SAVES THE CURRENT TEST_PC FOR USE BY THE 'ENDLOOP' ROUTINE.
1CEO 6140      ;
1CEO 6141      ; THE 'INCREMENT_VALUE' IS SET TO +1 IF THE STARTING VALUE IS LESS
1CEO 6142      ; THAN THE ENDING VALUE OTHERWISE IT IS SET TO -1.
1CEO 6143      ;
1CEO 6144      ; CALLING SEQUENCE:
1CEO 6145      ;
1CEO 6146      ; LOOP      LOOP_NAME,STARTING_VALUE,ENDING_VALUE
1CEO 6147      ;
1CEO 6148      ; IMPLICIT INPUTS:
1CEO 6149      ;
1CEO 6150      ; ARGUMENT LIST:
1CEO 6151      ;
1CEO 6152      ;         OFFSET 0 - INDEX NAME CODE
1CEO 6153      ;         OFFSET 1 - STARTING VALUE LOW
1CEO 6154      ;         OFFSET 2 - STARTING VALUE HIGH
1CEO 6155      ;         OFFSET 3 - ENDING VALUE LOW
1CEO 6156      ;         OFFSET 4 - ENDING VALUE HIGH
1CEO 6157      ;
1CEO 6158      ; INDEX_NAM_TBL - TABLE THAT CONTAINS THE ADDRESS OF THE I, J, OR
1CEO 6159      ;                 K INDEX TABLES.
1CEO 6160      ; TEST_PC      - CONTAINS THE CURRENT TEST PC.
1CEO 6161      ;
1CEO 6162      ; IMPLICIT OUTPUTS:
1CEO 6163      ;
1CEO 6164      ; THE I, J, OR K INDEX TABLE IS INITIALIZED.
1CEO 6165      ;
1CEO 6166      ;--
1CEO 6167
1CEO 6168 BEGIN_LOOP:
1CEO 6169      LDA      ARG_LIST      ; GET THE INDEX NAME BYTE
1CEO 6170      CALL     GET_INDEX_VALUE ; GET ADDRESS OF INDEX TABLE
1CEO 6171      LXI     D,ARG_LIST+1    ; GET ADDRESS OF STARTING VALUE
1CEO 6172      MVI     B,4              ; SET A LOOP COUNT
1CEO 6173 10$:    LDAX   D           ; GET LOW BYTE
1CEO 6174      MOV     M,A            ; PUT IN TABLE
1CEO 6175      INX     D           ; INCREMENT POINTERS
1CEO 6176      INX     H           ;
1CEO 6177      DCR     B           ; DONE 4 BYTES?
1CEO 6178      JNZ     10$         ; BRANCH IF NO
1CEO 6179      ;
1CEO 6180      ; NOW CALCULATE THE INCREMENT VALUE
1CEO 6181      ;
1CEO 6182      LDA      ARG_LIST+2    ; GET HIGH BYTE OF START VALUE
1CEO 6183      MOV     B,A            ; SAVE
1CEO 6184      LDA      ARG_LIST+4    ; GET HIGH BYTE OF END VALUE
1CEO 6185      CMP     B           ; IS START LESS THAN END?
1CEO 6186      JNZ     20$         ; NOT EQUAL, CHECK IF POSITIVE OR NEG

```

```
1CFE 6187 ;  
1CFE 6188 ; HIGH BYTES ARE EQUAL, CHECK THE LOW BYTES  
1CFE 6189 ;  
1CFE 6190        LDA            ARG_LIST+1        ; GET START LOW BYTE  
1D01 6191        MOV            B,A            ; SAVE  
1D02 6192        LDA            ARG_LIST+3        ; GET END LOW BYTE  
1D05 6193        CMP            B            ; IS START LESS THAN END?  
1D06 6194        JNC            1$            ; BRANCH IF YES  
1D09 6195        JZ             1$            ; ...  
1D0C 6196 ;  
1D0C 6197 ; END IS LESS THAN START. SET THE INCREMENT VALUE TO MINUS 1  
1D0C 6198 ;  
1D0C 6199 3$:     MVI            A,<^XFF>        ; GET INCREMENT VALUE  
1D0E 6200        MOV            M,A            ; PUT IN INDEX TABLE  
1D0F 6201        JMP            2$            ; GO PUT TEST PC IN TABLE  
1D12 6202 ;  
1D12 6203 ; START IS LESS THAN END. SET THE INCREMENT VALUE TO PLUS 1  
1D12 6204 1$:     MVI            A,1            ; GET THE INCREMENT VALUE  
1D14 6205        MOV            M,A            ; PUT IN INDEX TABLE  
1D15 6206        JMP            2$            ; GO GET THE TEST PC  
1D18 6207 ;  
1D18 6208 ; HIGH BYTES ARE NOT EQUAL, BRANCH TO APPROPRIATE SECTION TO SET INCREMENT VALUE  
1D18 6209 ;  
1D18 6210 20$:     JNC            1$            ; START IS LESS THAN END  
1D18 6211        JMP            3$            ; END IS LESS THAN START  
1D1E 6212 ;  
1D1E 6213 ; NOW GET THE TEST PC AND PUT IT IN THE INDEX TABLE  
1D1E 6214 ;  
1D1E 6215 2$:     INX            H            ; POINT AT LOW BYTE OF TEST PC FIELD  
1D1F 6216        LDA            TEST_PC        ; GET LOW BYTE OF TEST PC  
1D22 6217        MOV            M,A            ; PUT IN INDEX TABLE  
1D23 6218        INX            H            ; POINT AT HIGH BYTE FIELD  
1D24 6219        LDA            TEST_PC+1       ; GET HIGH BYTE OF TEST PC  
1D27 6220        MOV            M,A            ; PUT IN INDEX TABLE  
1D28 6221        RET            ; EXIT  
1D29 6222
```

```

1D29 6224      .SBTTL      END LOOP ROUTINE
1D29 6225      :++
1D29 6226      :
1D29 6227      : FUNCTIONAL DESCRIPTION:
1D29 6228      :
1D29 6229      :     THE ERROR LOOP FLAG IS CLEARED
1D29 6230      :
1D29 6231      :     THE ROUTINE ADDS THE 'LOOP_INCREMENT' VALUE TO THE CURRENT VALUE
1D29 6232      :     OF THE SPECIFIED LOOP_NAME.
1D29 6233      :     THEN
1D29 6234      :         IF THE LOOP INCREMENT VALUE IS +1
1D29 6235      :         THEN
1D29 6236      :             IF THE CURRENT VALUE IS LESS THAN OR EQUAL TO THE END VALUE
1D29 6237      :             THEN LOAD THE TEST_PC WITH THE LOOP TEST PC AND EXIT
1D29 6238      :             ELSE SET THE CURRENT VALUE TO MINUS ONE AND EXIT
1D29 6239      :         ELSE
1D29 6240      :             IF THE CURRENT VALUE IS GREATER THAN OR EQUAL TO THE END VALUE
1D29 6241      :             THEN LOAD THE TEST_PC WITH THE LOOP TEST PC AND EXIT
1D29 6242      :             ELSE SET THE CURRENT VALUE TO MINUS ONE AND EXIT
1D29 6243      :
1D29 6244      : CALLING SEQUENCE:
1D29 6245      :
1D29 6246      :     ENDL00P LOOP_NAME
1D29 6247      :
1D29 6248      : IMPLICIT INPUTS:
1D29 6249      :
1D29 6250      :     ARGUMENT LIST:
1D29 6251      :
1D29 6252      :         OFFSET 0 - INDEX NAME CODE
1D29 6253      :
1D29 6254      :         I, J, OR K INDEX TABLE
1D29 6255      :
1D29 6256      : IMPLICIT OUTPUTS:
1D29 6257      :
1D29 6258      :     TEST_PC - CONTAINS THE ADDRESS OF THE NEXT PSEUDO INSTRUCTION TO EXECUTE
1D29 6259      :
1D29 6260      :--
1D29 6261
1D29 6262      END_LOOP:
1D29 6263          LDA          SOFT_CTRL_REGA      ; CLEAR THE LOOP ERROR FLAG
1D2C 6264          ANI          <^CL00P_ERROR_FLAG>; ...
1D2E 6265          STA          SOFT_CTRL_REGA      ; ...
1D31 6266          LDA          ARG_LIST           ; GET THE INDEX CODE
1D34 6267          CALL         GET_INDEX_VALUE    ; GET CURRENT VALUE AND TABLE ADDRESS
1D37 6268          INX          H                    ; POINT AT THE INCREMENT VALUE
1D38 6269          INX          H                    ; ...
1D39 6270          INX          H                    ; ...
1D3A 6271          INX          H                    ; ...
1D3B 6272          XCHG         ; PUT IN D & E REG'S
1D3C 6273          LDAX        D                    ; GET THE INCREMENT VALUE
1D3D 6274          MOV          L,A                 ; SAVE IN L REGISTER
1D3E 6275          MVI          H,0                 ; INIT H FOR POSITIVE INCREMENT
1D40 6276          ORA          A                    ; IS INCREMENT POSITIVE?
1D41 6277          JP          20$                 ; BRANCH IF YES
1D44 6278          MVI          H,<^XFF>           ; SET H FOR NEGATIVE INCREMENT
  
```

```
1D46 6279 20$: DAD B ; ADD TO CURRENT VALUE
1D47 6280 DCX D ; POINT AT CURRENT VALUE
1D48 6281 DCX D ; ...
1D49 6282 DCX D ; ...
1D4A 6283 MOV A,H ; SAVE NEW CURRENT VALUE IN TABLE
1D4B 6284 STAX D ; ...
1D4C 6285 DCX D ; ...
1D4D 6286 MOV A,L ; ...
1D4E 6287 STAX D ; ...
1D4F 6288 ;
1D4F 6289 ; NOW SEE IF THE LOOP IS DONE
1D4F 6290 ;
1D4F 6291 INX D ; POINT AT END VALUE
1D50 6292 INX D ;
1D51 6293 INX D ; HIGH BYTE
1D52 6294 LDAX D ; GET HIGH BYTE OF END VALUE
1D53 6295 INX D ; POINT AT INCREMENT VALUE
1D54 6296 CMP H ; COMPARE WITH NEW CURRENT VALUE HIGH BYTE
1D55 6297 PUSH $PSW ; SAVE CONDITION CODES
1D56 6298 JNZ 30$ ; BRANCH IF NOT EQUAL
1D59 6299 POP $PSW ; THROW AWAY CONDITION CODES
1D5A 6300 ;
1D5A 6301 ; HIGH BYTES ARE EQUAL. CHECK THE LOW BYTES
1D5A 6302 ;
1D5A 6303 DCX D ; POINT AT LOW BYTE OF END VALUE
1D5B 6304 DCX D ;
1D5C 6305 LDAX D ; GET THE LOW BYTE
1D5D 6306 INX D ; POINT AT INCREMENT VALUE
1D5E 6307 INX D ;
1D5F 6308 CMP L ; COMPARE WITH LOW BYTE OF CURRENT VALUE
1D60 6309 PUSH $PSW ; SAVE THE CONDITION CODES
1D61 6310 ;
1D61 6311 ; SEE IF INCREMENT VALUE IS POSITIVE OR NEGATIVE
1D61 6312 ;
1D61 6313 30$: LDAX D ; GET THE INCREMENT VALUE
1D62 6314 ORA A ; SET THE CONDITION CODES
1D63 6315 JM 1$ ; BRANCH IF INCREMENT WAS NEGATIVE
1D66 6316 ;
1D66 6317 ; INCREMENT VALUE IS +1, SEE IF LOOP IS DONE
1D66 6318 ;
1D66 6319 POP $PSW ; GET COND CODES FROM COMPARE
1D67 6320 JC 10$ ; BRANCH IF LOOP IS DONE
1D6A 6321 JMP 5$ ; SET THE LOOP TEST PC
1D6D 6322 ;
1D6D 6323 ; INCREMENT VALUE IS -1, SEE IF LOOP IS DONE
1D6D 6324 ;
1D6D 6325 1$: POP $PSW ; GET CONDITION CODES FROM COMPARE
1D6E 6326 JNC 10$ ; BRANCH IF LOOP IS DONE
1D71 6327 ;
1D71 6328 ; SET THE TEST_PC TO THE PC IN THE LOOP TABLE
1D71 6329 ;
1D71 6330 5$: INX D ; POINT AT LOW BYTE OF TEST PC
1D72 6331 LDAX D ; GET THE LOW BYTE
1D73 6332 STA TEST_PC ; PUT IN TEST PC
1D76 6333 INX D ; POINT AT HIGH BYTE OF TEST PC
```

```
1D77 6334 LDAX D ; GET IT
1D78 6335 STA TEST_PC+1 ; PUT IN TEST PC
1D7B 6336 RET ; EXIT
1D7C 6337 ;
1D7C 6338 ; LOOP IS DONE, SET THE CURRENT VALUE TO MINUS ONE
1D7C 6339 ;
1D7C 6340 10$: LDA ARG_LIST ; GET THE LOOP NAME CODE
1D7F 6341 CALL GET_INDEX_VALUE ; GET POINTER TO CURRENT VALUE
1D82 6342 MVI A,-1 ; GET DATA TO SET IN CURRENT VALUE
1D84 6343 MOV M,A ; SET THE CURRENT VALUE TO MINUS ONE
1D85 6344 INX H ; ...
1D86 6345 MOV M,A ; ...
1D87 6346 RET ; EXIT
1D88 6347
```

```
1D88 6349 .SBTTL ERROR LOOP ROUTINE
1D88 6350 ;**
1D88 6351 ;
1D88 6352 ; FUNCTIONAL DESCRIPTION:
1D88 6353 ;
1D88 6354 ; THIS ROUTINE SAVES THE CURRENT TEST PC IN LOCATION 'ERROR_LOOP_ADDR'.
1D88 6355 ; THIS TEST PC IS USED WHEN PERFORMING A 'SCOPE LOOP'.
1D88 6356 ;
1D88 6357 ; CALLING SEQUENCE:
1D88 6358 ;
1D88 6359 ; ERRLOOP
1D88 6360 ;
1D88 6361 ; IMPLICIT INPUTS:
1D88 6362 ;
1D88 6363 ; TEST_PC - CONTAINS THE CURRENT TEST PC
1D88 6364 ;
1D88 6365 ; IMPLICIT OUTPUTS:
1D88 6366 ;
1D88 6367 ; ERROR_LOOP_ADDR - GETS LOADED WITH THE CURRENT TEST PC
1D88 6368 ;
1D88 6369 ;--
1D88 6370
1D88 6371 ERROR_LOOP:
1D88 6372 LHLD TEST_PC ; GET THE TEST PC
1D88 6373 SHLD ERROR_LOOP_ADDR ; SAVE
1D8E 6374 RET ; EXIT
1D8F 6375
```

```
1D8F 6377            .SBTTL    PATTERN GENERATE ROUTINE
1D8F 6378    :++
1D8F 6379    :
1D8F 6380    : FUNCTIONAL DESCRIPTION:
1D8F 6381    :
1D8F 6382    :        THIS ROUTINE IS USED TO SELECT ONE OF SIX TEST PATTERNS AND MOVE
1D8F 6383    :        THE SELECTED PATTERN TO THE SPECIFIED LOCATIONS. IF 'LONGLIT' IS
1D8F 6384    :        SPECIFIED, THE PATTERN IS COMPLIMENTED AND PLACED IN BITS 31 THRU
1D8F 6385    :        62 OF THE MICRO WORD SPECIFIED BY THE DESTINATION ADDRESS.
1D8F 6386    :        IF DESTINATION ADDRESS_X IS SPECIFIED, THE PATTERN IS ALSO MOVED
1D8F 6387    :        TO THIS ADDRESS.
1D8F 6388    :
1D8F 6389    :        THE PATTERNS ARE:
1D8F 6390    :
1D8F 6391    :                    AAAAAAAA
1D8F 6392    :                    55555555
1D8F 6393    :                    33333333
1D8F 6394    :                    0F0F0F0F
1D8F 6395    :                    00FF00FF
1D8F 6396    :                    0000FFFF
1D8F 6397    :
1D8F 6398    : CALLING SEQUENCE:
1D8F 6399    :
1D8F 6400    :        SA01PAT LOOP_NAME,DESTINATION_ADDRESS,[LONGLIT],[<DESTINATION_ADDRESS_X>]
1D8F 6401    :
1D8F 6402    : IMPLICIT INPUTS:
1D8F 6403    :
1D8F 6404    :        ARGUMENT LIST:
1D8F 6405    :
1D8F 6406    :                    OFFSET 0 - INDEX NAME CODE
1D8F 6407    :                    OFFSET 1 - DESTINATION ADDRESS LOW
1D8F 6408    :                    OFFSET 2 - DESTINATION ADDRESS HIGH
1D8F 6409    :                    OFFSET 3 - LONG LITERAL CODE
1D8F 6410    :
1D8F 6411    :        PATTERN_ADDRESS - BASE ADDRESS OF THE SIX PATTERNS
1D8F 6412    :        I, J, OR K INDEX TABLE
1D8F 6413    :
1D8F 6414    : IMPLICIT OUTPUTS:
1D8F 6415    :
1D8F 6416    :        THE SPECIFIED PATTERN IS MOVED TO THE SPECIFIED ADDRESS
1D8F 6417    :
1D8F 6418    : --
1D8F 6419    :
1D8F 6420    : PATTERN_GEN:
1D8F 6421    :        LDA                    ARG_LIST            ; GET THE INDEX CODE
1D92 6422    :        CALL                  GET_INDEX_VALUE ; GET THE INDEX VALUE
1D95 6423    :
1D95 6424    :        NOW GENERATE THE ADDRESS OF THE PATTERN
1D95 6425    :
1D95 6426    :        MVI                    A,4                    ; MULTIPLY INDEX VALUE BY 4
1D97 6427    :        CALL                  MULTIPLY_A_BC       ; GET INDEX INTO PATTERN TABLE
1D9A 6428    :        LXI                    H,PATTERN_ADDRESS ; GET THE BASE ADDRESS OF THE PATTERNS
1D9D 6429    :        DAD                    D                    ; INDEX THE BASE ADDRESS
1D9E 6430    :
1D9E 6431    :        SEE IF DESTINATION_ADDRESS_X IS SPECIFIED
```

```
1D9E 6432 ;  
1D9E 6433 XCHG  
1D9F 6434 LHLD ARG_LIST+4 ; GET OPTIONAL DESTINATION ADDRESS  
1DA2 6435 INR H ; IS UPPER BYTE MINUS 1?  
1DA3 6436 JZ 5$ ; BRANCH IF YES, ADDRESS NOT SPECIFIED  
1DA6 6437 DCR H ; RESTORE UPPER BYTE OF ADDRESS  
1DA7 6438 PUSH D ; SAVE PATTERN ADDRESS  
1DA8 6439 XCHG  
1DA9 6440 LHLD INIT_TEST_PC ; GET INITIAL TEST PC  
1DAC 6441 DAD D ; RELOCATE DESTINATION ADDRESS  
1DAD 6442 POP D ; RESTORE PATTERN ADDRESS  
1DAE 6443 ;  
1DAE 6444 ; NOW MOVE THE PATTERN TO THE ALTERNATE DESTINATION  
1DAE 6445 ;  
1DAE 6446 MVI B,4 ; SET A BYTE LOOP COUNT  
1DB0 6447 3$: LDAX D ; GET BYTE OF PATTERN  
1DB1 6448 MOV M,A ; PUT IN DESTINATION  
1DB2 6449 INX D ; INCREMENT ADDRESSES  
1DB3 6450 INX H ;  
1DB4 6451 DCR B ; DONE 4 BYTES?  
1DB5 6452 JNZ 3$ ; BRANCH IF NO  
1DB8 6453 DCX D ; BACKUP PATTERN ADDRESS  
1DB9 6454 DCX D ; ...  
1DBA 6455 DCX D ; ...  
1DBB 6456 DCX D ; ...  
1DBC 6457 ;  
1DBC 6458 ; GET THE ADDRESS OF THE DESTINATION  
1DBC 6459 ;  
1DBC 6460 5$: PUSH D ; SAVE PATTERN ADDRESS  
1DBD 6461 LHLD ARG_LIST+1 ; GET THE DESTINATION ADDRESS  
1DC0 6462 XCHG  
1DC1 6463 LHLD INIT_TEST_PC ; GET INITIAL TEST PC  
1DC4 6464 DAD D ; RELOCATE DESTINATION ADDRESS  
1DC5 6465 POP D ; RESTORE PATTERN ADDRESS  
1DC6 6466 XCHG ; PUT DEST. ADDR IN D & E  
1DC7 6467 ;  
1DC7 6468 ; SEE IF THE DESTINATION IS THE LONGLIT FIELD OF A MICRO WORD  
1DC7 6469 ;  
1DC7 6470 LDA ARG_LIST+3 ; GET THE LONGLIT CODE  
1DCA 6471 ORA A ; SET THE CONDITION CODES  
1DCB 6472 JNZ 10$ ; BRANCH IF LONG LIT  
1DCE 6473 ;  
1DCE 6474 ; LONG LITERAL WAS NOT SPECIFIED, MOVE THE PATTERN TO THE DESTINATION  
1DCE 6475 ;  
1DCE 6476 MVI B,4 ; SET A LOOP COUNT FOR NO. OF BYTES  
1DD0 6477 6$: MOV A,M ; GET A BYTE OF THE PATTERN  
1DD1 6478 INX H ;  
1DD2 6479 STAX D ; PUT IN THE DESTINATION  
1DD3 6480 INX D ;  
1DD4 6481 DCR B ; DECREMENT LOOP COUNT  
1DD5 6482 JNZ 6$ ; BRANCH IF NOT DONE WITH 4 BYTES  
1DD8 6483 JMP 20$ ; EXIT  
1DDB 6484 ;  
1DDB 6485 ; LONG LITERAL WAS SPECIFIED. MUST PUT THE PATTERN IN BITS 31 THRU 62 OF  
1DDB 6486 ; THE DESTINATION. THIS IS DONE BY MOVING THE PATTERN TO A TEMPORARY
```

```
1DDB 6487 ; BUFFER, LEFT SHIFTING THE BUFFER 7 BITS, THEN MOVING THE BUFFER TO
1DDB 6488 ; THE DESTINATION.
1DDB 6489 ;
1DDB 6490 ; FIRST COMPLIMENT THE PATTERN FOR THE LONG LIT FIELD
1DDB 6491 ;
1DDB 6492 10$: MVI B,4 ; SET THE BYTE COUNT
1DDD 6493 PUSH D ; SAVE DESTINATION ADDRESS
1DDE 6494 LXI D,TEMP_BUFFER ; GET ADDRESS OF TEMPORARY BUFFER
1DE1 6495 11$: MOV A,M ; GET A BYTE OF THE PATTERN
1DE2 6496 CMA ; COMPLIMENT
1DE3 6497 STAX D ; PUT IN TEMP BUFFER
1DE4 6498 INX D
1DE5 6499 INX H
1DE6 6500 DCR B ; DONE 4 BYTES?
1DE7 6501 JNZ 11$ ; BRANCH IF NO
1DEA 6502 POP D ; RESTORE DESTINATION ADDRESS
1DEB 6503 LXI H,TEMP_BUFFER ; PASS ADDRESS OF PATTERN
1DEE 6504 MVI B,31 ; PUT STARTING BIT NUMBER IN B REG
1DF0 6505 MVI C,62 ; AND ENDING BIT NUMBER IN C REG
1DF2 6506 XRA A ; CLEAR INHIBIT PARITY FLAG
1DF3 6507 PUSH $PSW ; CLEAR INHIBIT SCRAMBLE FLAG
1DF4 6508 CALL INSERT_FIELD ; GENERATE THE FIELD
1DF7 6509 ;
1DF7 6510 ; ALL DONE !!
1DF7 6511 ;
1DF7 6512 20$: RET ; EXIT
1DF8 6513
```

```

1DF8 6515      .SBTTL      LOAD REGISTER ROUTINE
1DF8 6516      ;**
1DF8 6517      ;
1DF8 6518      ; FUNCTIONAL DESCRIPTION:
1DF8 6519      ;
1DF8 6520      ;       THIS ROUTINE LOADS AN RDM REGISTER WITH THE SPECIFIED DATA.
1DF8 6521      ;
1DF8 6522      ; CALLING SEQUENCE:
1DF8 6523      ;
1DF8 6524      ;       LOADREG REGISTER_ADDRESS, SRC_ADDRESS, [SRC_INDEX], [DATA_TYPE]
1DF8 6525      ;
1DF8 6526      ; IMPLICIT INPUTS:
1DF8 6527      ;
1DF8 6528      ;       ARGUMENT LIST:
1DF8 6529      ;
1DF8 6530      ;           OFFSET 0 - REGISTER ADDRESS LOW
1DF8 6531      ;           OFFSET 1 - REGISTER ADDRESS HIGH
1DF8 6532      ;           OFFSET 2 - SOURCE ADDRESS LOW
1DF8 6533      ;           OFFSET 3 - SOURCE ADDRESS HIGH
1DF8 6534      ;           OFFSET 4 - SOURCE INDEX CODE
1DF8 6535      ;           OFFSET 5 - DATA TYPE CODE
1DF8 6536      ;
1DF8 6537      ; IMPLICIT OUTPUTS:
1DF8 6538      ;
1DF8 6539      ;       NONE
1DF8 6540      ;
1DF8 6541      ;--
1DF8 6542      ;
1DF8 6543      LOAD_REG:
1DF8 6544      LDA          ARG_LIST+4      ; GET THE INDEX CODE
1DF8 6545      CALL        GET_INDEX_VALUE ; GET THE INDEX VALUE
1DF8 6546      LDA          ARG_LIST+5      ; GET THE DATA TYPE
1E01 6547      CALL        MULTIPLY_A_BC  ; MULTIPLY DATA TYPE BY INDEX VALUE
1E04 6548      ;
1E04 6549      ; D & E NOW CONTAIN THE INDEX VALUE FOR THE SPECIFIED DATA TYPE
1E04 6550      ;
1E04 6551      2$:      LHLD        ARG_LIST+2      ; GET THE SRC ADDRESS
1E07 6552      DAD          D                ; INDEX IT
1E08 6553      XCHG
1E09 6554      LHLD        INIT_TEST_PC      ; GET INITIAL TEST PC
1E0C 6555      DAD          D                ; RELOCATE SOURCE ADDRESS
1E0D 6556      ;
1E0D 6557      ; REAL SOURCE ADDRESS IS NOW IN H & L. NOW WRITE THE SPECIFIED REGISTER
1E0D 6558      ; ACCORDING TO THE DATA TYPE. IF THE REGISTER IS THE RD CONTROL REGISTER,
1E0D 6559      ; READ THE FRONT PANEL REGISTER AND OR IN THE STATE OF THE LIGHTS.
1E0D 6560      ;
1E0D 6561      XCHG
1E0E 6562      LDA          ARG_LIST+5      ; PUT SRC ADDR IN D & E
1E11 6563      MOV          B,A            ; GET THE DATA TYPE
1E12 6564      LHLD        ARG_LIST        ; SAVE
1E15 6565      3$:      LDAX         D                ; GET THE REGISTER ADDRESS
1E16 6566      ;
1E16 6567      ; CHECK IF RD_CTRL_REG
1E16 6568      ;
1E16 6569      PUSH        $PSW            ; GET A BYTE OF DATA
1E16 6569      ;
1E16 6569      PUSH        $PSW            ; SAVE DATA

```

1E17	6570	MOV	A,L	: GET LOW BYTE OF REGISTER ADDRESS
1E18	6571	CPI	<RD_CTRL_REG & 255>	
1E1A	6572			: RD CONTROL REG?
1E1A	6573	JNZ	5\$: BRANCH IF NO
1E1D	6574	LDA	FRONT_PNL_2	: GET STATE OF LIGHTS
1E20	6575	ANI	<REMOTE!FAULT!TEST>	
1E22	6576	MOV	C,A	: ..:
1E23	6577	POP	\$PSW	: GET DATA THAT IS BEING WRITTEN
1E24	6578	ORA	C	: INSERT FRONT PANEL CONDITIONS
1E25	6579	JMP	7\$: PUT DATA IN REGISTER
1E28	6580	5\$: POP	\$PSW	: RESTORE THE DATA
1E29	6581	7\$: INX	D	: ..:
1E2A	6582	MOV	M,A	: WRITE THE REGISTER
1E2B	6583	INX	H	: ..:
1E2C	6584	DCR	B	: DECREMENT THE BYTE COUNT
1E2D	6585	JNZ	3\$: CONTINUE FOR DATA TYPE NO OF BYTES
1E30	6586	RET		: EXIT
1E31	6587			

```

1E31 6589      .SBTTL      SKIP ROUTINE
1E31 6590      :++
1E31 6591      :
1E31 6592      : FUNCTIONAL DESCRIPTION:
1E31 6593      :
1E31 6594      : IF THE CONDITION IS SPECIFIED AND IS ONE OF 'ONERROR' OR 'NOERROR'
1E31 6595      : THE LOOP ERROR FLAG IS TESTED AND IF IN THE SPECIFIED STATE, THE
1E31 6596      : TEST PC IS LOADED WITH THE DESTINATION ADDRESS. IF THE CONDITION
1E31 6597      : IS 'ONEQUAL', THE CURRENT VALUES OF THE TWO INDEXES ARE COMPARED
1E31 6598      : AND IF EQUAL, THE TEST PC IS LOADED WITH THE SPECIFIED ADDRESS.
1E31 6599      :
1E31 6600      : IF THE CONDITION IS NOT SPECIFIED, THE TEST PC IS LOADED WITH
1E31 6601      : THE SPECIFIED ADDRESS.
1E31 6602      :
1E31 6603      : CALLING SEQUENCE:
1E31 6604      :
1E31 6605      : SKIP      [DESTINATION ADDRESS],[CONDITION],[INDEX NAME],[INDEX NAME]
1E31 6606      :
1E31 6607      : IMPLICIT INPUT:
1E31 6608      :
1E31 6609      : ARGUMENT LIST:
1E31 6610      :
1E31 6611      : OFFSET 0 - DESTINATION ADDRESS LOW
1E31 6612      : OFFSET 1 - DESTINATION ADDRESS HIGH
1E31 6613      : OFFSET 2 - CONDITION
1E31 6614      : OFFSET 3 - INDEX FLAG
1E31 6615      : OFFSET 4 - INDEX FLAG
1E31 6616      :
1E31 6617      : IMPLICIT OUTPUTS:
1E31 6618      :
1E31 6619      : TEST_PC - SEE FUNCTIONAL DESCRIPTION.
1E31 6620      :
1E31 6621      :--
1E31 6622      :
1E31 6623      SKIP:
1E31 6624      LDA      SOFT_CTRL_REGA ; GET THE LOOP ERROR FLAG
1E34 6625      ANI      LOOP_ERROR_FLAG ; ...
1E36 6626      MOV      B,A ; SAVE IN B
1E37 6627      LDA      ARG_LIST+2 ; GET THE CONDITION
1E3A 6628      ORA      A ; SET THE CONDITION CODES
1E3B 6629      JM      1$ ; BRANCH IF NONE SPECIFIED
1E3E 6630      CPI      ONEQUAL ; INDEX CONDITION?
1E40 6631      JZ      5$ ; BRANCH IF YES
1E43 6632      CPI      NOTEQUAL ; INDEX CONDITION?
1E45 6633      JZ      5$ ; BRANCH IF YES
1E48 6634      CPI      ONERROR ; ONERROR CONDITION?
1E4A 6635      JZ      15$ ; BRANCH IF ON
1E4D 6636      :
1E4D 6637      : MUST BE NOERROR CONDITION
1E4D 6638      :
1E4D 6639      MOV      A,B ; GET LOOP ERROR FLAG
1E4E 6640      ORA      A ; IS IT SET?
1E4F 6641      JZ      1$ ; BRANCH IF NO
1E52 6642      JMP      10$ ; EXIT
1E55 6643      :
  
```

```
1E55 6644 ; ONERROR CONDITION
1E55 6645 ;
1E55 6646 15$: MOV A,B ; GET LOOP ERROR FLAG
1E56 6647 ORA A ; IS TI SET?
1E57 6648 JNZ 1$ ; BRANCH IF YES
1E5A 6649 JMP 10$ ; EXIT
1E5D 6650 ;
1E5D 6651 ; ONEQUAL OR NOTEQUAL CONDITION
1E5D 6652 ;
1E5D 6653 5$: LDA ARG_LIST+3 ; GET THE FIRST INDEX CODE
1E60 6654 CALL GET_INDEX_VALUE ; GET THE CURRENT VALUE
1E63 6655 PUSH B ; SAVE CURRENT VALUE
1E64 6656 LDA ARG_LIST+4 ; GET THE SECOND INDEX CODE
1E67 6657 CALL GET_INDEX_VALUE ; GET THE CURRENT VALUE
1E6A 6658 ;
1E6A 6659 ; NOW COMPARE THE CURRENT VALUES OF THE TWO INDEXES
1E6A 6660 ;
1E6A 6661 POP D ; GET VALUE OF FIRST INDEX
1E6B 6662 MOV A,E ; CHECK LOWER ORDER BITS
1E6C 6663 CMP C ; EQUAL?
1E6D 6664 JNZ 7$ ; BRANCH IF NO
1E70 6665 MOV A,D ; CHECK UPPER ORDER BITS
1E71 6666 CMP B ; EQUAL?
1E72 6667 JNZ 7$ ; BRANCH IF NO
1E75 6668 ;
1E75 6669 ; INDEXES ARE EQUAL. CHECK THE CONDITION
1E75 6670 ;
1E75 6671 LDA ARG_LIST+2 ; GET THE CONDITION
1E78 6672 CPI ONEQUAL ; ONEQUAL CONDITION?
1E7A 6673 JZ 1$ ; BRANCH IF YES
1E7D 6674 JMP 10$ ; EXIT
1E80 6675 ;
1E80 6676 ; INDEXES ARE NOT EQUAL. CHECK THE CONDITION
1E80 6677 ;
1E80 6678 7$: LDA ARG_LIST+2 ; GET THE CONDITION
1E83 6679 CPI ONEQUAL ; ONEQUAL CONDITION?
1E85 6680 JZ 10$ ; BRANCH IF YES
1E88 6681 ;
1E88 6682 ; SET THE TEST PC TO THE SPECIFIED ADDRESS
1E88 6683 ;
1E88 6684 1$: LHLD INIT_TEST_PC ; GET BASE ADDRESS OF TEST
1E8B 6685 XCHG
1E8C 6686 LHLD ARG_LIST ; GET THE SPECIFIED ADDRESS
1E8F 6687 DAD D ; RELOCATE THE DESTINATION ADDRESS
1E90 6688 SHLD TEST_PC ; SET THE TEST PC TO THE DESIRED ADDRESS
1E93 6689 10$: RET ; EXIT
1E94 6690
```

```
1E94 6692      .SBTTL      SUB TEST ROUTINE
1E94 6693      ;++
1E94 6694      ;
1E94 6695      ; FUNCTIONAL DESCRIPTION:
1E94 6696      ;
1E94 6697      ;     THIS ROUTINE INCREMENTS THE SUBTEST NUMBER AND INITIALIZES THE
1E94 6698      ;     THE CURRENT VALUES OF THE INDEX NAMES.
1E94 6699      ;
1E94 6700      ; CALLING SEQUENCE:
1E94 6701      ;
1E94 6702      ;     SUBTEST
1E94 6703      ;
1E94 6704      ; IMPLICIT INPUTS:
1E94 6705      ;
1E94 6706      ;     ARGUMENT LIST:
1E94 6707      ;
1E94 6708      ;     OFFSET 0 - CONTAINS THE SUB TEST NUMBER
1E94 6709      ;
1E94 6710      ; IMPLICIT OUTPUTS:
1E94 6711      ;
1E94 6712      ;     SUB_TEST_NUMB - GETS THE NEW SUBTEST NUMBER
1E94 6713      ;
1E94 6714      ;--
1E94 6715      ;
1E94 6716      SUB_TEST:
1E94 6717      ;
1E94 6718      ; INITIALIZE THE CURRENT VALUE OF THE LOOP NAMES
1E94 6719      ;
1E94 6720      ;     CALL          INIT_LOOP_VALUE ;
1E97 6721      ;
1E97 6722      ; INITIALIZE THE TRAP FLAG
1E97 6723      ;
1E97 6724      ;     LDA          SOFT_CTRL_REGA ; GET THE FLAG
1E9A 6725      ;     ANI          <^CEXP_UTRAP_FLAG> ; CLEAR THE FLAG
1E9C 6726      ;     STA          SOFT_CTRL_REGA ; RESTORE THE FLAGS
1E9F 6727      ;
1E9F 6728      ; SET THE SUBTEST NUMBER
1E9F 6729      ;
1E9F 6730      ;     LDA          ARG_LIST      ; GET THE SUBTEST NUMBER
1EA2 6731      ;     STA          SUB_TEST_NUMB ; SAVE THE NEW SUB TEST NUMBER
1EA5 6732      ;     RET                          ; EXIT
1EA6 6733
```

```

1EA6 6735      .SBTTL      FETCH ROUTINE
1EA6 6736      ;++
1EA6 6737      ;
1EA6 6738      ; FUNCTIONAL DESCRIPTION:
1EA6 6739      ;
1EA6 6740      ;     THIS ROUTINE IS USED TO LOAD A DCS MICRO WORD INTO THE 11/750
1EA6 6741      ;     CONTROL STORE LATCHES. THIS IS DONE WITH THE FOLLOWING SEQUENCE:
1EA6 6742      ;
1EA6 6743      ;     1) ASSERT 'MICRO ADDRESS INHIBIT' AND 'PARITY CHECK'
1EA6 6744      ;     2) LOAD THE DCS ADDRESS REGISTER
1EA6 6745      ;     3) LOAD THE 'ADDRESS MATCH' REGISTER WITH 1800(X).
1EA6 6746      ;     4) STEP THE CLOCK ONE MICRO STATE.
1EA6 6747      ;     5) DEASCERT 'PARITY CHECK'
1EA6 6748      ;     6) IF 'INHIBIT' IS NOT SPECIFIED, DEASCERT 'MICRO ADDRESS INHIBIT'
1EA6 6749      ;
1EA6 6750      ;     THE CONTENTS OF SOMM ADDRESS IS REPLACED IN THE MATCH REGISTER
1EA6 6751      ;     AFTER THE FETCH.
1EA6 6752      ;
1EA6 6753      ; CALLING SEQUENCE:
1EA6 6754      ;
1EA6 6755      ;     FETCH   DCS_ADDRESS,INHIBIT
1EA6 6756      ;
1EA6 6757      ; IMPLICIT INPUTS:
1EA6 6758      ;
1EA6 6759      ;     ARGUMENT LIST:
1EA6 6760      ;
1EA6 6761      ;     OFFSET 0 - DCS ADDRESS
1EA6 6762      ;     OFFSET 1 - DESTINATION INHIBIT CODE
1EA6 6763      ;
1EA6 6764      ; IMPLICIT OUTPUTS:
1EA6 6765      ;
1EA6 6766      ;     NONE
1EA6 6767      ;
1EA6 6768      ;--
1EA6 6769      ;
1EA6 6770      ; FETCH_MIC_INSTR:
1EA6 6771      ;     LDA      SOFT_CTRL_REG      ; SET THE FETCH FLAG
1EA9 6772      ;     ORI      FETCH_FLAG      ; ...
1EAB 6773      ;     STA      SOFT_CTRL_REG      ; ...
1EAE 6774      ;     LDA      ARG_LIST      ; GET THE DCS ADDRESS
1EB1 6775      ;     CALL     START_EXECUTION ; FETCH THE MICRO WORD
1EB4 6776      ;
1EB4 6777      ; SEE IF MICRO ADDRESS INHIBIT SHOULD BE ASCERTED
1EB4 6778      ;
1EB4 6779      ;     LDA      ARG_LIST+1      ; GET THE INHIBIT FLAG
1EB7 6780      ;     ORA      A              ; SET THE CONDITION CODES
1EB8 6781      ;     LDA      DCS_CTRL_RE_CPY ; GET CTRL REG DATA
1EBB 6782      ;     JM      1$             ; BRANCH IF INHIBIT NOT SPECIFIED
1EBE 6783      ;     ORI      MICRO_ADDR_INH ; LEAVE MICRO ADDR INHIBIT SET
1EC0 6784      ;     1$:   STA      DCS_CTRL_REG ; LOAD THE CONTROL REGISTER
1EC3 6785      ;     STA      DCS_CTRL_RE_CPY ; SAVE A COPY
1EC6 6786      ;
1EC6 6787      ; LOAD THE SOMM ADDRESS IN THE MATCH REGISTER
1EC6 6788      ;
1EC6 6789      ;     LHLD     SOMM_ADDRESS      ; GET THE SOMM ADDRESS
  
```

ZZ-ECKAA-8.7 V08.07
ECKAA
V08.07

G 14
VAX-11/750 MICRO DIAGNOSTIC MONITOR
FETCH ROUTINE

11-FEB-1986 10:07:52 VAX/VMS Macro V04-00 Page 150
11-FEB-1986 10:06:37 [VAX750.MONITOR]ECKAA.MAR;811 (60)

1EC9 6790 SHLD
1ECC 6791 RET
1ECD 6792

ADDR_MATCH_LO . ; LOAD THE MATCH REGISTER
; EXIT

```

1ECD 6794      .SBTTL      MASK ROUTINE
1ECD 6795      ;++
1ECD 6796      ;
1ECD 6797      ; FUNCTIONAL DESCRIPTION:
1ECD 6798      ;
1ECD 6799      ;     THIS ROUTINE IS USED TO MASK AN ELEMENT OF DATA. THE MASK IS PERFORMED
1ECD 6800      ;     BY DOING A LOGICAL 'AND' OF THE DATA WITH THE MASK.
1ECD 6801      ;
1ECD 6802      ; CALLING SEQUENCE:
1ECD 6803      ;
1ECD 6804      ;     MASK      SOURCE ADDRESS,SOURCE INDEX,MASK ADDRESS,MASK INDEX
1ECD 6805      ;
1ECD 6806      ; IMPLICIT INPUTS:
1ECD 6807      ;
1ECD 6808      ;     ARGUMENT LIST:
1ECD 6809      ;
1ECD 6810      ;           OFFSET 0 - SOURCE ADDRESS LOW
1ECD 6811      ;           OFFSET 1 - SOURCE ADDRESS HIGH
1ECD 6812      ;           OFFSET 2 - SOURCE INDEX CODE
1ECD 6813      ;           OFFSET 3 - MASK ADDRESS LOW
1ECD 6814      ;           OFFSET 4 - MASK ADDRESS HIGH
1ECD 6815      ;           OFFSET 5 - MASK INDEX CODE
1ECD 6816      ;           OFFSET 6 - DATA TYPE CODE
1ECD 6817      ;
1ECD 6818      ; IMPLICIT OUTPUTS:
1ECD 6819      ;
1ECD 6820      ;     THE CONTENTS OF THE SOURCE ADDRESS IS MODIFIED BY THE MASK.
1ECD 6821      ;
1ECD 6822      ;--
1ECD 6823      ;
1ECD 6824      MASK:   LDA          ARG_LIST+2      ; GET THE SOURCE INDEX CODE
1ED0 6825          CALL         GET_INDEX_VALUE  ; GET THE CURRENT VALUE
1ED3 6826      ;
1ED3 6827      ; NOW MULTIPLY THE INDEX BY THE DATA TYPE
1ED3 6828      ;
1ED3 6829          LDA          ARG_LIST+6      ; GET THE DATA TYPE
1ED6 6830          CALL         MULTIPLY_A_BC   ; MULTIPLY DATA TYPE AND LOOP VALUE
1ED9 6831      ;
1ED9 6832      ; D & E REGISTER NOW HAS SOURCE INDEX
1ED9 6833      ;
1ED9 6834          LHL         ARG_LIST        ; GET THE SOURCE ADDRESS
1EDC 6835          DAD          D              ; ADD THE INDEX
1EDD 6836          XCHG
1EDE 6837          LHL         INIT_TEST_PC    ; GET INITIAL TEST PC
1EE1 6838          DAD          D              ; RELOCATE SOURCE ADDRESS
1EE2 6839          PUSH         H              ; SAVE THE SOURCE ADDRESS
1EE3 6840      ;
1EE3 6841      ; NOW GET THE MASK INDEX AND ADDRESS
1EE3 6842      ;
1EE3 6843          LDA          ARG_LIST+5      ; GET THE MASK INDEX CODE
1EE6 6844          CALL         GET_INDEX_VALUE  ; GET THE CURRENT VALUE
1EE9 6845          LDA          ARG_LIST+6      ; GET THE DATA TYPE
1EEC 6846          CALL         MULTIPLY_A_BC   ; MULTIPLY DATA TYPE TIMES INDEX VALUE
1EEF 6847          LHL         ARG_LIST+3      ; GET THE MASK ADDRESS
1EF2 6848          DAD          D              ; INDEX THE MASK ADDRESS
  
```

```
1EF3 6849      XCHG
1EF4 6850      LHLD      INIT_TEST_PC      ; GET INITIAL TEST PC
1EF7 6851      DAD        D                ; RELOCATE MASK ADDRESS
1EF8 6852      ;
1EF8 6853      ; NOW START MASKING THE SOURCE DATA
1EF8 6854      ;
1EF8 6855      POP        D                ; GET SOURCE ADDRESS
1EF9 6856      LDA        ARG_LIST+6      ; GET THE DATA TYPE
1EFC 6857      MOV        B,A            ; SAVE
1EFD 6858 1$:  LDAX       D                ; GET A BYTE OF THE DATA
1EFE 6859      ANA        M                ; MASK THE BYTE
1EFF 6860      STAX       D                ; PUT BACK IN SOURCE ADDRESS
1F00 6861      INX        D                ; INCREMENT THE SOURCE AND
1F01 6862      INX        H                ; MASK ADDRESSES
1F02 6863      DCR        B                ; DECREMENT THE BYTE COUNT
1F03 6864      JNZ       1$              ; CONTINUE FOR DATA TYPE NO. OF BYTES
1F06 6865      ;
1F06 6866      ; ALL DONE
1F06 6867      ;
1F06 6868      RET
1F07 6869
```

```

1F07 6871      .SBTTL      COMPARE REGISTER ROUTINE
1F07 6872      ;++
1F07 6873      ;
1F07 6874      ; FUNCTIONAL DESCRIPTION:
1F07 6875      ;
1F07 6876      ;     THIS ROUTINE COMPARES THE CONTENTS OF AN RDM REGISTER(S) WITH
1F07 6877      ;     THE CONTENTS OF A SPECIFIED LOCATION IN THE THE 'DATA' AREA OF THE
1F07 6878      ;     TEST. IF THE TWO DATA ELEMENTS DO NOT SATISFY THE MODE ARGUMENT,
1F07 6879      ;     THE ERROR FLAG AND LOOP ERROR FLAG ARE SET OTHERWISE, THE LOOP ERROR
1F07 6880      ;     FLAG IS CLEARED.
1F07 6881      ;
1F07 6882      ; CALLING SEQUENCE:
1F07 6883      ;
1F07 6884      ;     CMPREG  REG ADDR,DEST ADDR,[DEST INDEX],[DATA TYPE]
1F07 6885      ;
1F07 6886      ; IMPLICIT INPUTS:
1F07 6887      ;
1F07 6888      ;     ARGUMENT LIST:
1F07 6889      ;
1F07 6890      ;         OFFSET 0 - REGISTER ADDRESS LOW
1F07 6891      ;         OFFSET 1 - REGISTER ADDRESS HIGH
1F07 6892      ;         OFFSET 2 - DESTINATION ADDRESS LOW
1F07 6893      ;         OFFSET 3 - DESTINATION ADDRESS HIGH
1F07 6894      ;         OFFSET 4 - DESTINATION INDEX CODE
1F07 6895      ;         OFFSET 5 - DATA TYPE CODE
1F07 6896      ;         OFFSET 6 - MODE TYPE
1F07 6897      ;             0 = COMPARE FOR EQUALITY
1F07 6898      ;             FF = COMPARE FOR NOT EQUAL
1F07 6899      ;
1F07 6900      ; IMPLICIT OUTPUTS:
1F07 6901      ;
1F07 6902      ;     GOOD_DATA      - GETS THE CONTENTS OF THE 'DATA' ELEMENT
1F07 6903      ;     BAD_DATA       - GETS THE CONTENTS OF THE RDM REGISTER
1F07 6904      ;     DATA_TYPE    - GETS THE DATA TYPE USED IN THE COMPARISON
1F07 6905      ;
1F07 6906      ;--
1F07 6907
1F07 6908 COMPARE_REG:
1F07 6909      LHL      ARG_LIST      ; GET ADDRESS OF THE RDM REGISTER
1F0A 6910      LDA      ARG_LIST+5    ; GET THE DATA TYPE
1F0D 6911      CALL     SAVE_BAD_DATA ; PUT THE REGISTER DATA INTO IMPLICIT STOAGE
1F10 6912      ;
1F10 6913      ; GET THE DATA INDEX AND INDEX THE DATA ADDRESS
1F10 6914      ;
1F10 6915      ;     LDA      ARG_LIST+4      ; GET THE INDEX CODE
1F13 6916      ;     CALL     GET_INDEX_VALUE ; GET THE CURRENT VALUE
1F16 6917      ;     MOV      A,C          ; SAVE
1F17 6918      ;     STA      QA_INDEX      ; ...
1F1A 6919      ;     MOV      A,B          ; ...
1F1B 6920      ;     STA      QA_INDEX+1    ; ...
1F1E 6921      ;     LDA      ARG_LIST+5    ; GET THE DATA TYPE
1F21 6922      ;     CALL     MULTIPLY_A_BC ; ADJUST INDEX VALUE BY DATA TYPE
1F24 6923      ;     LHL      ARG_LIST+2    ; GET THE DATA ADDRESS
1F27 6924      ;     DAD      D          ; INDEX THE DATA ADDRESS
1F28 6925      ;     XCHG

```

```
1F29 6926            LHLD            INIT_TEST_PC        ; GET INITIAL TEST PC
1F2C 6927            DAD                D                    ; RELOCATE DATA ADDRESS
1F2D 6928            LDA                ARG_LIST+5          ; GET THE DATA TYPE
1F30 6929            CALL               SAVE_GOOD_DATA     ; SAVE THE GOOD DATA
1F33 6930            ;
1F33 6931            ; NOW COMPARE THE GOOD AND BAD DATA
1F33 6932            ;
1F33 6933            LDA                ARG_LIST+6          ; GET THE MODE TYPE
1F36 6934            STA                MODE_TYPE            ; SAVE FOR TYPEOUT ROUTINE
1F39 6935            MOV                B,A                ; PASS IN B REGISTER
1F3A 6936            LDA                ARG_LIST+5          ; GET THE DATA TYPE
1F3D 6937            CALL                CMP_GOOD_BAD        ; COMPARE THE DATA
1F40 6938            JNC                10$                ; BRANCH IF DATA IS EQUAL TO REGISTER
1F43 6939            LDA                SOFT_CTRL_REGA     ; GET THE CONTROL FLAGS
1F46 6940            ORI                <ERROR_FLAG!-     ; SET THE ERROR AND LOOP ERROR
1F46 6941                               LOOP_ERROR_FLAG>; FLAGS
1F48 6942            ANI                <^CVBUS_COMPARE>; ENSURE VBUS COMPARE FLAG CLEAR
1F4A 6943            STA                SOFT_CTRL_REGA     ;
1F4D 6944            RET                                    ; EXIT
1F4E 6945            ;
1F4E 6946            ; DATA WAS EQUAL TO REGISTER. CLEAR THE LOOP ERROR FLAG
1F4E 6947            ;
1F4E 6948            10$: LDA                SOFT_CTRL_REGA     ; GET THE CONTROL REG
1F51 6949            ANI                <^C<LOOP_ERROR_FLAG!-
1F51 6950                               VBUS_COMPARE>> ; ...
1F53 6951            STA                SOFT_CTRL_REGA     ; ...
1F56 6952            CALL                CHECK_QA_FLAG        ; CHECK IF QA FLAG SET
1F59 6953            RET                                    ; EXIT
1F5A 6954
```

```

1F5A 6956      .SBTTL      COMPARE REGISTER MASKED ROUTINE
1F5A 6957      ;++
1F5A 6958      ;
1F5A 6959      ; FUNCTIONAL DESCRIPTION:
1F5A 6960      ;
1F5A 6961      ; THIS ROUTINE COMPARES THE CONTENTS OF AN RDM REGISTER(S) WITH THE
1F5A 6962      ; CONTENTS OF A SPECIFIED DATA ELEMENT. BEFORE THE COMPARISON, THE
1F5A 6963      ; CONTENTS OF THE REGISTER(S) IS MASKED BY A SPECIFIED MASK. THE
1F5A 6964      ; MASK IS PERFORMED BY LOGICALLY 'ANDING' THE MASK AND THE CONTENTS
1F5A 6965      ; OF THE REGISTER(S). IF THE RESULTING COMPARISON DOES NOT SATISFY
1F5A 6966      ; THE MODE ARGUMENT, THE ERROR AND LOOP ERROR FLAGS ARE SET OTHERWISE,
1F5A 6967      ; THE LOOP ERROR FLAG IS CLEARED.
1F5A 6968      ;
1F5A 6969      ; CALLING SEQUENCE:
1F5A 6970      ;
1F5A 6971      ; CMPREGMSK REG ADDRESS,DST ADDRESS,[DST INDEX],MSK ADDRESS,[MSK INDEX],[DAT -

1F5A 6972      ;
1F5A 6973      ; IMPLICIT INPUTS:
1F5A 6974      ;
1F5A 6975      ; ARGUMENT LIST:
1F5A 6976      ;
1F5A 6977      ; OFFSET 0 - REGISTER ADDRESS LOW
1F5A 6978      ; OFFSET 1 - REGISTER ADDRESS HIGH
1F5A 6979      ; OFFSET 2 - DESTINATION ADDRESS LOW
1F5A 6980      ; OFFSET 3 - DESTINATION ADDRESS HIGH
1F5A 6981      ; OFFSET 4 - DESTINATION INDEX CODE
1F5A 6982      ; OFFSET 5 - MASK ADDRESS LOW
1F5A 6983      ; OFFSET 6 - MASK ADDRESS HIGH
1F5A 6984      ; OFFSET 7 - MASK INDEX CODE
1F5A 6985      ; OFFSET 8 - DATA TYPE CODE
1F5A 6986      ; OFFSET 9 - MODE TYPE
1F5A 6987      ; 0 = COMPARE FOR EQUALITY
1F5A 6988      ; FF = COMPARE FOR NOT EQUAL
1F5A 6989      ;
1F5A 6990      ; IMPLICIT OUTPUTS:
1F5A 6991      ;
1F5A 6992      ; GOOD_DATA - GETS THE CONTENTS OF THE DATA ELEMENT
1F5A 6993      ; BAD_DATA - GETS THE MASKED CONTENTS OF THE RDM REGISTER
1F5A 6994      ; DATA_TYPE - GETS THE SPECIFIED DATA TYPE
1F5A 6995      ;
1F5A 6996      ; --
1F5A 6997      ;
1F5A 6998      COMPARE_REG_MSK:
1F5A 6999      LHL      ARG_LIST      ; GET THE REGISTER ADDRESS
1F5D 7000      LDA      ARG_LIST+8    ; GET THE DATA TYPE
1F60 7001      CALL     SAVE_BAD_DATA ; SAVE THE REGISTER DATA
1F63 7002      ;
1F63 7003      ; NOW GET THE DESTINATION ADDRESS AND SAVE THE DATA
1F63 7004      ;
1F63 7005      LDA      ARG_LIST+4    ; GET THE INDEX CODE
1F66 7006      CALL     GET_INDEX_VALUE ; GET THE INDEX VALUE
1F69 7007      MOV      A,C          ; SAVE
1F6A 7008      STA      QA_INDEX      ; ...
1F6D 7009      MOV      A,B          ; ...
1F6E 7010      STA      QA_INDEX+1    ; ...
  
```

cont> A TYPE]

```
1F71 7011 LDA ARG_LIST+8 ; GET THE DATA TYPE
1F74 7012 CALL MULTIPLY_A_BC ; MULTIPLY DATA TYPE BY CURRENT INDEX VALUE
1F77 7013 LHLD ARG_LIST+2 ; GET THE DATA ADDRESS
1F7A 7014 DAD D ; INDEX THE ADDRESS
1F7B 7015 XCHG
1F7C 7016 LHLD INIT_TEST_PC ; GET INITIAL TEST PC
1F7F 7017 DAD D ; RELOATE DATA ADDRESS
1F80 7018 LDA ARG_LIST+8 ; GET THE DATA TYPE
1F83 7019 CALL SAVE_GOOD_DATA ; SAVE THE DATA ELEMENT
1F86 7020 ;
1F86 7021 ; NOW GET THE MASK ADDRESS
1F86 7022 ;
1F86 7023 LDA ARG_LIST+7 ; GET THE MASK INDEX CODE
1F89 7024 CALL GET_INDEX_VALUE ; GET THE CURRENT INDEX VALUE
1F8C 7025 LDA ARG_LIST+8 ; GET THE DATA TYPE
1F8F 7026 CALL MULTIPLY_A_BC ; GET THE INDEX VALUE TIMES THE DATA TYPE
1F92 7027 LHLD ARG_LIST+5 ; GET THE MASK ADDRESS
1F95 7028 DAD D ; INDEX THE MASK ADDRESS
1F96 7029 XCHG
1F97 7030 LHLD INIT_TEST_PC ; GET INITIAL TEST PC
1F9A 7031 DAD D ; RELOCATE MASK ADDRESS
1F9B 7032 ;
1F9B 7033 ; NOW MASK THE 'GOOD_DATA'
1F9B 7034 ;
1F9B 7035 PUSH H ; SAVE THE MASK ADDRESS
1F9C 7036 LXI D,GOOD_DATA ; GET ADDRESS OF GOOD DATA
1F9F 7037 LDA ARG_LIST+8 ; GET THE DATA TYPE
1FA2 7038 MOV B,A ; SAVE
1FA3 7039 1$: LDAX D ; GET A BYTE OF THE GOOD DATA
1FA4 7040 ANA M ; AND WITH THE MASK
1FA5 7041 STAX D ; PUT BACK IN GOOD DATA
1FA6 7042 INX D ; POINT AT NEXT BYTE
1FA7 7043 INX H ;
1FA8 7044 DCR B ; DECREMENT THE BYTE COUNT
1FA9 7045 JNZ 1$ ; CONTINUE FOR THE DATA TYPE NO. OF BYTES
1FAC 7046 ;
1FAC 7047 ; NOW MASK THE 'BAD_DATA'
1FAC 7048 ;
1FAC 7049 POP H ; GET ADDRESS OF MASK
1FAD 7050 LXI D,BAD_DATA ; GET ADDRESS OF BAD DATA
1FB0 7051 LDA ARG_LIST+8 ; GET THE DATA TYPE
1FB3 7052 MOV B,A ; SAVE
1FB4 7053 2$: LDAX D ; GET A BYTE OF THE BAD DATA
1FB5 7054 ANA M ; AND WITH THE MASK
1FB6 7055 STAX D ; PUT BACK IN BAD DATA
1FB7 7056 INX D ; POINT AT NEXT BYTE
1FB8 7057 INX H ;
1FB9 7058 DCR B ; DECREMENT THE BYTE COUNT
1FBA 7059 JNZ 2$ ; CONTINUE FOR THE DATA TYPE NO. OF BYTES
1FBD 7060 ;
1FBD 7061 ; DATA IS ALREADY IN THE 'GOOD' AND 'BAD' DATA LOCATIONS. NOW COMPARE IT.
1FBD 7062 ;
1FBD 7063 LDA ARG_LIST+9 ; GET THE MODE TYPE
1FC0 7064 STA MODE_TYPE ; SAVE FOR TYPEOUT ROUTINE
1FC3 7065 MOV B,A ; PASS IN B REGISTER
```

1FC4	7066	LDA	ARG_LIST+8	:	GET THE DATA TYPE
1FC7	7067	CALL	CMP_GOOD_BAD	:	COMPARE THE DATA
1FCA	7068	JNC	S\$:	BRANCH IF DATA COMPARED
1FCD	7069	LDA	SOFT_CTRL_REGA	:	GET THE CONTROL REGISTER
1FD0	7070	ORI	<ERROR_FLAG!-	:	SET THE ERROR AND LOOP ERROR
1FD0	7071		LOOP_ERROR_FLAG>	:	FLAGS
1FD2	7072	ANI	<^CVBUS_COMPARE>	:	ENSURE VBUS COMPARE FLAG CLEAR
1FD4	7073	STA	SOFT_CTRL_REGA	:	...
1FD7	7074	RET			
1FD8	7075	:			
1FD8	7076	:	DATA IS OK. CLEAR THE LOOP ERROR FLAG.		
1FD8	7077	:			
1FD8	7078	S\$:	LDA		SOFT_CTRL_REGA
1FDB	7079		ANI		<^C<LOOP_ERROR_FLAG!-
1FDB	7080				VBUS_COMPARE>>
1FDD	7081	STA	SOFT_CTRL_REGA		
1FE0	7082	CALL	CHECK_QA_FLAG		
1FE3	7083	RET		:	EXIT
1FE4	7084				

```
1FE4 7086 .SBTTL COMPARE V BUS ROUTINE
1FE4 7087 ;++
1FE4 7088 ;
1FE4 7089 ; FUNCTIONAL DESCRIPTION:
1FE4 7090 ;
1FE4 7091 ; THIS ROUTINE IS USED TO COMPARE VALUES OF BITS ON THE V BUS WITH
1FE4 7092 ; EXPECTED VALUES OF THE BITS. THE EXPECTED VALUES MUST BE PLACED IN
1FE4 7093 ; A TABLE WITH THE FOLLOWING FORMAT:
1FE4 7094 ;
1FE4 7095 ; .BYTE NUMBER OF BITS
1FE4 7096 ; .BYTE BIT<7>=VALUE AND BITS <6:0>=BIT NUMBER
1FE4 7097 ;
1FE4 7098 ;
1FE4 7099 ;
1FE4 7100 ; IF ANY BIT ON THE V BUS IS NOT IN THE SAME STATE AS SPECIFIED IN THE
1FE4 7101 ; TABLE, THE ERROR AND LOOP ERROR FLAGS ARE SET.
1FE4 7102 ;
1FE4 7103 ; CALLING SEQUENCE:
1FE4 7104 ;
1FE4 7105 ; CMPVBUS TABLE_ADDRESS,[INDEX NAME]
1FE4 7106 ;
1FE4 7107 ; IMPLICIT INPUTS:
1FE4 7108 ;
1FE4 7109 ; ARGUMENT LIST:
1FE4 7110 ;
1FE4 7111 ; OFFSET 0 - DATA TABLE ADDRESS LOW
1FE4 7112 ; OFFSET 1 - DATA TABLE ADDRESS HIGH
1FE4 7113 ; OFFSET 2 - INDEX NAME CODE
1FE4 7114 ;
1FE4 7115 ; IMPLICIT OUTPUTS:
1FE4 7116 ;
1FE4 7117 ; GOOD_DATA - CONTAINS THE EXPECTED BIT NO. AND VALUE IF A BIT FAILED
1FE4 7118 ; BAD_DATA - CONTAINS THE RECEIVED BIT NO. AND VALUE IF A BIT FAILED
1FE4 7119 ; ERROR AND LOOP ERROR FLAGS ARE SET IF ANY BIT IS NOT CORRECT
1FE4 7120 ;
1FE4 7121 ;--
1FE4 7122 ;
1FE4 7123 COMPARE_VBUS:
1FE4 7124 MVI A,WORD ; INITIALIZE THE DATA TYPE
1FE6 7125 STA DATA_TYPE ;
1FE9 7126 XRA A ; INITIALIZE THE MODE TYPE
1FEA 7127 STA MODE_TYPE ; 'EQUALS' FOR THIS COMPARE
1FED 7128 ;
1FED 7129 ; CLEAR THE GOOD AND BAD DATA LONG WORDS
1FED 7130 ;
1FED 7131 LXI H,0 ; GET DATA TO CLEAR WITH
1FF0 7132 SHLD GOOD_DATA ; CLEAR LOW 16 BITS
1FF3 7133 SHLD GOOD_DATA+2 ; AND UPPER 16 BITS
1FF6 7134 SHLD BAD_DATA ; CLEAR BAD DATA
1FF9 7135 SHLD BAD_DATA+2 ; ...
1FFC 7136 ;
1FFC 7137 ; FIRST READ THE V BUS INTO A BUFFER
1FFC 7138 ;
1FFC 7139 CALL READ_V_BUS ; READ THE V BUS
1FFF 7140 LHLD ARG_LIST ; GET ADDRESS OF EXPECTED TABLE
```

```
2002 7141      XCHG
2003 7142      LHLD          INIT_TEST_PC      ; GET INITIAL TEST PC
2006 7143      DAD           D              ; RELOCATE TABLE ADDRESS
2007 7144      ;
2007 7145      ; SEE IF TABLE ADDRESS IS INDEXED
2007 7146      ;
2007 7147      PUSH          H              ; SAVE H & L
2008 7148      LDA           ARG_LIST+2     ; GET THE INDEX FLAG
200B 7149      CALL          GET_INDEX_VALUE ; GET THE CURRENT VALUE
200E 7150      MOV           A,C           ; SAVE
200F 7151      STA           QA_INDEX      ; ...
2012 7152      MOV           A,B           ; ...
2013 7153      STA           QA_INDEX+1    ; ...
2016 7154      POP           H              ; RESTORE H & L
2017 7155      JC           5$            ; BRANCH IF NO INDEX
201A 7156      ;
201A 7157      ; GENRATE THE ADDRESS OF THE INDEXED TABLE
201A 7158      ;
201A 7159      INR           C              ; BUMP THE REGISTER FOR COUNT TO ZERO
201B 7160 3$:  DCR           C              ; DONE INDEXING?
201C 7161      JZ           5$            ; BRANCH IF YES
201F 7162      MOV           E,M          ; GET NUMBER OF BITS IN CURRENT TABLE
2020 7163      MVI           D,0          ; SETUP TO ADD TO TABLE ADDRESS
2022 7164      INX           D              ;
2023 7165      DAD           D              ; POINT TO NEXT TABLE
2024 7166      JMP           3$            ; CHECK IF DONE INDEXING
2027 7167      ;
2027 7168      ; GET NUMBER OF BITS TO COMPARE
2027 7169      ;
2027 7170 5$:  MOV           B,M          ; GET THE NUMBER OF BITS TO COMPARE
2028 7171      INX           H              ; POINT AT FIRST EXPECTED BIT
2029 7172      ;
2029 7173      ; NOW START CHECKING THE BITS
2029 7174      ;
2029 7175 1$:  CALL          CHECK_VBUS     ; CHECK THE VBUS BIT
202C 7176      JC           10$           ; BRANCH IF BIT DID NOT MATCH
202F 7177      INX           H              ; POINT AT NEXT EXPECTED BIT
2030 7178      DCR           B              ; CHECKED ALL THE BITS?
2031 7179      JNZ          1$            ; BRANCH IF NO
2034 7180      ;
2034 7181      ; ALL BITS ARE OK. CLEAR THE LOOP ERROR FLAG
2034 7182      ;
2034 7183      LDA           SOFT_CTRL_REGA
2037 7184      ANI           <^C<LOOP_ERROR_FLAG!-
2037 7185      VBUS_COMPARE>>
2039 7186      STA           SOFT_CTRL_REGA
203C 7187      CALL          CHECK_QA_FLAG  ; CHECK IF QA FLAG SET
203F 7188      RET                          ; EXIT
2040 7189      ;
2040 7190      ; A BIT ON THE V BUS IS INCORRECT. SET THE ERROR AND LOOP ERROR FLAGS.
2040 7191      ;
2040 7192 10$: LDA           SOFT_CTRL_REGA ; GET THE FLAGS
2043 7193      ORI           <ERROR_FLAG!- ; SET THE ERROR AND LOOP
2043 7194      LOOP_ERROR_FLAG!-
2043 7195      VBUS_COMPARE> ; ERROR FLAGS
```

ZZ-ECKAA-8.7 V08.07
ECKAA
V08.07

D 15
COMPARE V BUS11-FEB-1986
VAX-11/750 MICRO DIAGNOSTIC MONITOR
COMPARE V BUS ROUTINE

Fiche 1 Frame D15 Sequence 185
11-FEB-1986 10:07:52 VAX/VMS Macro V04-00 Page 160
11-FEB-1986 10:06:37 [VAX750.MONITOR]ECKAA.MAR;811 (64)

2045	7196	STA	SOFT_CTRL_REGA	:	:::
2048	7197	INX	H	:	POINT AT NEXT BIT
2049	7198	SHLD	VBUS_TBL_ADDR	:	SAVE THE TABLE ADDRESS
204C	7199	MOV	A,B	:	AND THE BIT COUNT
204D	7200	DCR	A	:	DECREMENT THE BIT COUNT
204E	7201	STA	VBUS_TBL_COUNT	:	:::
2051	7202	RET		:	EXIT
2052	7203			:	

```
2052 7205 .SBTTL IF ERROR ROUTINE
2052 7206 :++
2052 7207 :
2052 7208 : FUNCTIONAL DESCRIPTION:
2052 7209 :
2052 7210 : THIS ROUTINE IS USED TO TEST THE ERROR AND LOOP ERROR FLAGS. FOLLOWING
2052 7211 : IS THE ALGORITHM:
2052 7212 :
2052 7213 : IF THE ERROR FLAG IS CLEAR
2052 7214 : THEN EXIT
2052 7215 : ELSE
2052 7216 :
2052 7217 : IF THE CONTROL_C_FLAG IS SET
2052 7218 : THEN CALL THE COMMAND PARSER
2052 7219 : ELSE
2052 7220 :
2052 7221 : IF THE LOOP_ERROR_FLAG IS SET
2052 7222 : THEN
2052 7223 : IF THE BELL_FLAG IS SET
2052 7224 : THEN CALL THE RING_BELL ROUTINE
2052 7225 : ELSE
2052 7226 :
2052 7227 : IF THE NER_FLAG IS CLEAR
2052 7228 : THEN CALL THE TYPE ERROR ROUTINE
2052 7229 : ELSE
2052 7230 :
2052 7231 : IF THE HALT_FLAG IS SET
2052 7232 : THEN CALL THE COMMAND PARSER
2052 7233 : ELSE
2052 7234 :
2052 7235 : IF THE LOOP_FLAG IS SET AND THE SA_FLAG IS CLEAR
2052 7236 : THEN
2052 7237 : IF THE TEST_PC IS EQUAL TO THE ERROR_PC
2052 7238 : THEN SET THE TEST_PC TO THE ERROR LOOP ADDRESS
2052 7239 : CLEAR THE ERROR LOOP FLAG, AND EXIT
2052 7240 : ELSE EXIT
2052 7241 : ELSE EXIT
2052 7242 :
2052 7243 : CALLING SEQUENCE:
2052 7244 :
2052 7245 : IFERROR [DATA COUNT],[GATE ARRAY LIST],[MODULE LIST]
2052 7246 :
2052 7247 : IMPLICIT INPUTS:
2052 7248 :
2052 7249 : ARGUMENT LIST:
2052 7250 :
2052 7251 : OFFSET 0 - DATA COUNT
2052 7252 : OFFSET 1 - NUMBER OF GATE ARRAY CODES IN LIST
2052 7253 : OFFSET 2 - FIRST GATE ARRAY CODE
2052 7254 :
2052 7255 :
2052 7256 : OFFSET N - LAST GATE ARRAY CODE
2052 7257 : OFFSET N+1 - NUMBER OF MODULE NAME CODES IN LIST
2052 7258 : OFFSET N+2 - MODULE NAME CODE
2052 7259 :
```

```
2052 7260 ;  
2052 7261 ;          OFFSET N+M - LAST MODULE NAME CODE  
2052 7262 ;  
2052 7263 ;          ERROR_LOOP_ADDR - CONTAINS THE ADDRESS TO LOOP BACK TO IF THE LOOP  
2052 7264 ;          FLAG IS SET AND THE SA FLAG IS CLEAR.  
2052 7265 ;  
2052 7266 ; IMPLICIT OUTPUTS:  
2052 7267 ;  
2052 7268 ;          TEST_PC          - CONTAINS THE ADDRESS OF THE NEXT PSEUDO INSTRUCTION  
2052 7269 ;  
2052 7270 ;--  
2052 7271 ;  
2052 7272 IF_ERROR:  
2052 7273     LDA          SOFT_CTRL_REGA ; GET THE ERROR FLAG  
2055 7274     ANI          ERROR_FLAG  ; IS IT SET?  
2057 7275     JZ           50$          ; BRANCH IF NO (EXIT)  
205A 7276 ;  
205A 7277 ; CHECK THE CONTROL C FLAG  
205A 7278 ;  
205A 7279     LDA          SOFT_CTRL_REG  ; GET THE CONTROL C FLAG  
205D 7280     ANI          CONTROL_C_FLAG ; IS IT SET?  
205F 7281     CNZ          GET_CMD_LINE ; CALL COMMAND PARSER IF YES  
2062 7282 ;  
2062 7283 ; CHECK THE LOOP ERROR FLAG  
2062 7284 ;  
2062 7285     LDA          SOFT_CTRL_REGA ; GET THE LOOP ERROR FLAG  
2065 7286     ANI          LOOP_ERROR_FLAG ; IS IT SET?  
2067 7287     JZ           10$          ; BRANCH IF NO  
206A 7288 ;  
206A 7289 ; CHECK THE BELL FLAG  
206A 7290 ;  
206A 7291     LDA          PROG_CTRL_REG  ; GET THE BELL FLAG  
206D 7292     ANI          BELL_FLAG    ; IS IT SET?  
206F 7293     CNZ          RING_BELL    ; RING THE TERMINAL BELL IF YES  
2072 7294 ;  
2072 7295 ; CHECK THE NER FLAG  
2072 7296 ;  
2072 7297     LXI          H,ARG_LIST     ; GET ADDRESS OF ARGUMENT LIST  
2075 7298     LDA          PROG_CTRL_REG ; GET THE NER FLAG  
2078 7299     ANI          NER_FLAG     ; IS IT CLEAR?  
207A 7300     CZ           TYPE_ERROR  ; YES,TYPE THE ERROR MESSAGE  
207D 7301 ;  
207D 7302 ; CHECK THE HALT FLAG  
207D 7303 ;  
207D 7304 5$: LDA          PROG_CTRL_REG ; GET THE HALT FLAG  
2080 7305     ANI          HALT_FLAG    ; IS THE HALT FLAG SET?  
2082 7306     CNZ          GET_CMD_LINE ; CALL COMMAND PARSER IF YES  
2085 7307 ;  
2085 7308 ; CHECK THE LOOP FLAG AND THE SA FLAG  
2085 7309 ;  
2085 7310 10$: LDA          PROG_CTRL_REG ; GET THE LOOP AND SA FLAGS  
2088 7311     MOV          B,A          ; SAVE  
2089 7312     ANI          LOOP_FLAG   ; IS THE LOOP FLAG SET?  
208B 7313     JZ           50$          ; BRANCH IF NO (EXIT)  
208E 7314     MOV          A,B          ; GET THE SA FLAG
```

```
208F 7315      ANI      SA_FLAG      ; IS THE SA FLAG SET?
2091 7316      JNZ      50$          ; BRANCH IF YES (EXIT)
2094 7317      ;
2094 7318      ; SEE IF THIS IS THE CORRECT ERROR PC
2094 7319      ;
2094 7320      LXI      H,CURRENT_PC ; GET ADDRESS OF CURRENT PC
2097 7321      LDA      ERROR_PC    ; GET LOW BYTE OF ERROR PC
209A 7322      CMP      M           ; LOW BYTE THE SAME?
209B 7323      JNZ      50$          ; BRANCH IF NO
209E 7324      INX      H           ; POINT AT HIGH BYTE
209F 7325      LDA      ERROR_PC+1 ; GET HIGH BYTE
20A2 7326      CMP      M           ; HIGH BYTE THE SAME?
20A3 7327      JNZ      50$          ; BRANCH IF NO
20A6 7328      ;
20A6 7329      ; SET THE TEST PC TO THE ERROR LOOP ADDRESS
20A6 7330      ;
20A6 7331      LHLD     ERROR_LOOP_ADDR ; GET THE ERROR LOOP TEST PC
20A9 7332      SHLD     TEST_PC      ; SET THE TEST PC
20AC 7333      LDA      SOFT_CTRL_REGA ; GET THE CONTROL REGISTER
20AF 7334      ANI      <^CLOOP_ERROR_FLAG>; CLEAR THE ERROR LOOP FLAG
20B1 7335      STA      SOFT_CTRL_REGA ; ...
20B4 7336      ;
20B4 7337      ; EXIT
20B4 7338      ;
20B4 7339      50$:   RET          ; EXIT
20B5 7340
```

```
20B5 7342      .SBTTL      BURST CLOCK ROUTINE
20B5 7343      ;**
20B5 7344      ;
20B5 7345      ; FUNCTIONAL DESCRIPTION:
20B5 7346      ;
20B5 7347      ;     THIS ROUTINE IS USED TO BURST THE CPU CLOCK. THE ALGORITHM IS:
20B5 7348      ;
20B5 7349      ;     IF THE LOOP AND ERROR FLAGS ARE SET AND THE SA FLAG IS CLEAR AND
20B5 7350      ;     THE INHIBIT FLAG IS CLEAR
20B5 7351      ;     THEN CLEAR THE 'STOP CLOCK' BIT AND SET THE 'DCS ADDRESS CLEAR'
20B5 7352      ;     BIT AT THE SPECIFIED DCS ADDRESS
20B5 7353      ;     ELSE SET THE 'STOP CLOCK' BIT AND CLEAR THE 'DCS ADDRESS CLEAR'
20B5 7354      ;     BIT AT THE SPECIFIED DCS ADDRESS
20B5 7355      ;
20B5 7356      ;     IF THE CYCLE OR TICK FLAG IS SET
20B5 7357      ;     THEN WHILE THE OPERATOR TYPES SPACES, DO
20B5 7358      ; (15$)     BEGIN
20B5 7359      ;     SET THE TYPEOUT ADDRESS TO THE CURRENT DCS ADDRESS MINUS ONE
20B5 7360      ;     TYPE THE TYPEOUT ADDRESS
20B5 7361      ;     IF THE LOOP AND ERROR FLAGS ARE NOT SET
20B5 7362      ;     THEN
20B5 7363      ;     IF THE TYPEOUT ADDRESS EQUALS THE <DCS ADDRESS>
20B5 7364      ;     THEN CALL THE COMMAND PARSER
20B5 7365      ; (25$)     WAIT FOR OPERATOR INPUT
20B5 7366      ;     CHECK FOR SPACE
20B5 7367      ;     STEP THE CLOCK (CYCLE OR TICK)
20B5 7368      ;     END
20B5 7369      ;     CALL COMMAND PARSER
20B5 7370      ;     IF RETURN FROM COMMAND PARSER
20B5 7371      ;     THEN RESTART THIS ROUTINE
20B5 7372      ;     ELSE
20B5 7373      ;     ELSE
20B5 7374      ;
20B5 7375      ; (30$)     START THE CPU CLOCK
20B5 7376      ;     UNTIL OPERATOR TYPES 'CONTROL C' OR THE CLOCK STOPS, DO
20B5 7377      ;     BEGIN
20B5 7378      ;     WAIT HERE
20B5 7379      ;     END
20B5 7380      ;     IF THE OPERATOR TYPED CONTROL C
20B5 7381      ;     THEN CALL THE COMMAND PARSER. IF COMMAND PARSER RETURNS,
20B5 7382      ;     RESTART THIS ROUTINE
20B5 7383      ;     ELSE
20B5 7384      ; (40$)     IF THE 'EXP_UTRAP_FLAG' IS SET
20B5 7385      ;     THEN
20B5 7386      ;     IF THE TRAP REGISTER IS STILL IN DCS SPACE
20B5 7387      ;     THEN
20B5 7388      ; (45$)     IF THE SOMM FLAG IS SET AND THE STOP CLOCK BIT IN THE CURRENT DCS
20B5 7389      ;     ADDRESS IS NOT SET
20B5 7390      ;     THEN TYPE 'STOP ON MICRO MATCH' MESSAGE AND CALL THE COMMAND PARSER
20B5 7391      ;     IF THE COMMAND PARSER RETURNS
20B5 7392      ;     THEN RESTART THE CLOCK AND WAIT AGAIN
20B5 7393      ;     ELSE
20B5 7394      ;     ELSE
20B5 7395      ; (50$)     CLEAR THE CONTROL FILE
20B5 7396      ;     IF THE SOMM FLAG IS CLEAR AND THE CURRENT DCS ADDRESS MINUS 1 IS
```

```
20B5 7397 : EQUAL TO THE <DCS ADDRESS>
20B5 7398 : THEN EXIT THIS ROUTINE
20B5 7399 : (60$) ELSE TYPE 'UNEXPECTED CLOCK STOPPED' MESSAGE AND CALL THE COMMAND
20B5 7400 : PARSER. DON'T ALLOW CONTINUE.
20B5 7401 : ELSE EXIT THIS ROUTINE.
20B5 7402 : ELSE GO TO 45$
20B5 7403 :
20B5 7404 : CALLING SEQUENCE:
20B5 7405 :
20B5 7406 : BRSTCLK <DCS ADDRESS>
20B5 7407 :
20B5 7408 : IMPLICIT INPUTS:
20B5 7409 :
20B5 7410 : ARGUMENT LIST:
20B5 7411 :
20B5 7412 : OFFSET 0 - DCS ADDRESS TO STOP THE CLOCK AT
20B5 7413 : OFFSET 1 - INHIBIT RETURN TO ZERO FLAG
20B5 7414 : = -1 IF SPECIFIED
20B5 7415 : = 0 IF NOT SPECIFIED
20B5 7416 :
20B5 7417 : IMPLICIT OUTPUTS:
20B5 7418 :
20B5 7419 : NONE
20B5 7420 :
20B5 7421 : --
20B5 7422 :
20B5 7423 BURST_CLOCK:
20B5 7424 LDA ARG_LIST ; GET THE SPECIFIED DCS ADDRESS
20B8 7425 STA SETCLR_CF_ADDR ; SAVE FOR SET/CLR CF ROUTINES
20BB 7426 LXI H,CLK_STOP_MSG1 ; INITIALIZE THE ERROR MESSAGE
20BE 7427 SHLD CLK_MSG_ADDR ;
20C1 7428 LDA DCS_ADDR_REG_R0 ; SAVE THE CURRENT DCS ADDRESS
20C4 7429 ANI <^C<CLK_CTRL_1_R0!-
20C4 7430 CLK_CTRL_0_R0>> ; DISCARD CLOCK STATE BITS
20C6 7431 STA M_CLK_ADDR ;
20C9 7432 LDA SOFT_CTRL_REG ; CLEAR THE BURST_STOP FLAG
20CC 7433 ANI <^C<BURST_STOP_FLAG>>
20CE 7434 STA SOFT_CTRL_REG ; ...
20D1 7435 :
20D1 7436 : CHECK THE INHIBIT BURST FLAG
20D1 7437 :
20D1 7438 LDA PROG_CTRL_REG ; GET THE FLAG
20D4 7439 ANI IB_FLAG ; IS THE FLAG SET?
20D6 7440 JNZ 1$ ; BRANCH IF YES
20D9 7441 :
20D9 7442 : CHECK THE LOOP, ERROR, AND SA FLAGS
20D9 7443 :
20D9 7444 LDA SOFT_CTRL_REGA ; GET THE ERROR FLAG
20DC 7445 ANI ERROR_FLAG ; IS IT SET?
20DE 7446 JZ 1$ ; BRANCH IF NO
20E1 7447 LDA PROG_CTRL_REG ; GET THE LOOP FLAG
20E4 7448 MOV B,A ; SAVE THE SA FLAG
20E5 7449 ANI LOOP_FLAG ; IS IT SET?
20E7 7450 JZ 1$ ; BRANCH IF NO
20EA 7451 MOV A,B ; GET THE SA FLAG
```

```
20EB 7452      ANI      SA_FLAG      ; IS IT CLEAR?
20ED 7453      JNZ      1$          ; BRANCH IF NO
20F0 7454      LDA      ARG_LIST+1    ; GET THE INHIBIT FLAG
20F3 7455      ORA      A            ; SET THE CONDITION CODES
20F4 7456      JNZ      1$          ; BRANCH IF INHIBIT SPECIFIED
20F7 7457      ;
20F7 7458      ; LOOP AND ERROR FLAGS ARE SET, SA FLAG IS CLEAR. SET THE 'DCS ADDRESS
20F7 7459      ; CLEAR' BIT AND CLEAR THE 'STOP CLOCK' BIT IN THE SPECIFIED DCS ADDRESS
20F7 7460      ;
20F7 7461      MVI      A,DCS_ADDR_CLEAR ; GET THE ADDRESS CLEAR BIT
20F9 7462      STA      SETCLR_CF_DATA ; PASS TO SET/CLR CF ROUTINE
20FC 7463      CALL     SET_DCS_CF      ; SET THE BIT
20FF 7464      MVI      A,STOP_CLOCK    ; CLEAR THE STOP CLOCK BIT
2101 7465      STA      SETCLR_CF_DATA ; ...
2104 7466      CALL     CLEAR_DCS_CF     ; ...
2107 7467      JMP      10$         ; GO TO NEXT STEP
210A 7468      ;
210A 7469      ; FLAGS WERE NOT IN THE CORRECT STATE. SET THE 'STOP CLOCK' BIT AND CLEAR
210A 7470      ; THE 'DCS ADDRESS CLEAR' BIT IN THE CONTROL FILE.
210A 7471      ;
210A 7472      1$: MVI      A,STOP_CLOCK    ; SET THE CLOCK STOP BIT
210C 7473      STA      SETCLR_CF_DATA ; ...
210F 7474      CALL     SET_DCS_CF      ; ...
2112 7475      MVI      A,DCS_ADDR_CLEAR ; GET THE ADDRESS CLEAR BIT
2114 7476      STA      SETCLR_CF_DATA ; CLEAR IT IN THE CONTROL FILE
2117 7477      CALL     CLEAR_DCS_CF     ; ...
211A 7478      LDA      SOFT_CTRL_REG   ; SET THE BURST_STOP FLAG
211D 7479      ORI      BURST_STOP_FLAG ; ...
211F 7480      STA      SOFT_CTRL_REG   ; ...
2122 7481      ;
2122 7482      ; CHECK IF SINGLE CYCLE OR SINGLE TICK HAS BEEN SPECIFIED
2122 7483      ;
2122 7484      10$: LDA      SOFT_CTRL_REGA ; GET THE TWO FLAGS
2125 7485      ANI      <CYCLE_FLAG!TICK_FLAG> ; IS EITHER ONE SET?
2127 7486      JZ       30$         ; BRANCH IF NO
212A 7487      ;
212A 7488      ; TYPE THE CURRENT DCS ADDRESS
212A 7489      ;
212A 7490      15$: TYPE     CRLF          ;
2133 7491      TYPE     DCS_ADDR_MSG    ; TYPE 'DCS ADDR= '
213C 7492      LDA      DCS_ADDR_REG_RO ; GET CURRENT DCS ADDRESS
213F 7493      ANI      <<C<CLK_CTRL_1_RO!-
213F 7494      CLK_CTRL_0_RO>> ; DISCARD CLOCK STATE BITS
2141 7495      DCR      A            ; BACKUP BY ONE
2142 7496      STA      BURST_TEMP     ; SAVE FOR TYPEOUT
2145 7497      MOV      B,A           ; SAVE A REG
2146 7498      LDA      DCS_CTRL_FILE   ; SEE IF THE RETURN TO ZERO
2149 7499      ANI      DCS_ADDR_CLEAR  ; BIT IS ACTIVE IN THE CONTROL FILE
214B 7500      JNZ      16$         ; BRANCH IF NOT ACTIVE
214E 7501      LDA      STEP_ADDRESS  ; GET THE PREVIOUS ADDRESS
2151 7502      INR      A            ; INCREMENT TO CURRENT ADDRESS
2152 7503      STA      BURST_TEMP     ; SET FOR TYPEOUT
2155 7504      MOV      B,A           ; SAVE FOR NEXT INSTRUCTION
2156 7505      16$: MOV      A,B          ; RESTORE THE DCS ADDRESS
2157 7506      STA      STEP_ADDRESS  ; SAVE
```

```

215A 7507          TYPEB          BURST_TEMP          ; TYPE THE CURRENT ADDRESS
2165 7508          ;
2165 7509          ; SEE IF WE ARE STILL IN DCS SPACE. IF NOT, TERMINATE TICKING THE CLOCK.
2165 7510          ;
2165 7511          LDA          SOFT_CTRL_REG          ; GET THE FETCH FLAG
2168 7512          ANI          FETCH_FLAG            ;
216A 7513          JNZ          19$                   ; SKIP TEST IF JUST EXECUTED FETCH
216D 7514          LDA          CS_ADD_TRAP_HGH        ; GET HIGH 8 BITS OF TRAP REGISTER
2170 7515          ANI          <^X3F>                ; DISCARD UPPER TWO BITS
2172 7516          CPI          <^X18>                ; CHECK IF IN THE RANGE 18 TO 1F
2174 7517          JM          18$                   ; BRANCH IF LESS THAN 18
2177 7518          CPI          <^X20>                ;
2179 7519          JM          19$                   ; BRANCH IF LESS THAN 20
217C 7520          ;
217C 7521          ; WE LEFT DCS SPACE, TYPE THE TRAP REGISTER CONTENTS
217C 7522          ;
217C 7523          18$:          TYPE          CLK_STOP_MSG4          ; TYPE ' TRAP ADDRESS= '
2185 7524          LHLD         CS_ADD_TRAP_LOW        ; GET TRAP REGISTER
2188 7525          MOV          A,H                   ; GET HIGH BYTE
2189 7526          ANI          <^X3F>                ; DISCARD TOP TWOBITS
218B 7527          MOV          H,A                   ; PUT BACK
218C 7528          SHLD         BURST_TEMP            ; SAVE
218F 7529          TYPEW        BURST_TEMP            ; TYPE IT
219A 7530          CALL         GET_CMD_LINE          ; CALL THE COMMAND PARSER
219D 7531          RET                               ; EXIT THIS ROUTINE
219E 7532          ;
219E 7533          ; SEE IF THE LOOP AND ERROR FLAGS ARE SET
219E 7534          ;
219E 7535          19$:          LDA          ARG_LIST+1          ; GET THE INHIBIT ARGUMENT
21A1 7536          ORA          A                      ; SET THE CONDITION CODES
21A2 7537          JNZ          20$                   ; BRANCH IF INHIBIT SPECIFIED
21A5 7538          LDA          SOFT_CTRL_REGA        ; GET THE ERROR FLAG
21A8 7539          MOV          B,A                   ; SAVE
21A9 7540          ANI          ERROR_FLAG            ; IS IT SET?
21AB 7541          JZ          20$                   ; BRANCH IF NO
21AE 7542          LDA          PROG_CTRL_REG         ; GET THE LOOP FLAG
21B1 7543          ANI          LOOP_FLAG             ; IS IT SET?
21B3 7544          JNZ          25$                   ; BRANCH IF YES & WAIT FOR OPERATOR
21B6 7545          ;
21B6 7546          ;
21B6 7547          ; LOOP AND ERROR FLAGS WERE NOT BOTH SET OR THE INHIBIT ARGUMENT IS SPECIFIED.
21B6 7548          ; DON'T ALLOW OPERATOR TO STEP BEYOND THE 'STOP CLOCK' MICRO INSTRUCTION.
21B6 7549          ;
21B6 7550          20$:          LDA          STEP_ADDRESS          ; GET THE TYPEOUT ADDRESS
21B9 7551          MOV          B,A                   ; SAVE
21BA 7552          LDA          ARG_LIST              ; GET THE SPECIFIED ADDRESS
21BD 7553          CMP          B                      ; DOES TYPEOUT ADDR=<DCS ADDRESS>?
21BE 7554          JNZ          25$                   ; BRANCH IF NO
21C1 7555          CALL         GET_CMD_LINE          ; CALL THE COMMAND PARSER
21C4 7556          RET                               ; CONTINUE WITH NEXT PSEUDO
21C5 7557          ;
21C5 7558          ;
21C5 7559          ; WAIT FOR OPERATOR INPUT AND CHECK FOR A SPACE.
21C5 7560          ;
21C5 7561          25$:          $TERM_INIT            ; CLEAR ANY READ REQUESTS

```

```
21CA 7562      $TERM_READ      TERM_INP_BUFF,ONE ; READ ONE CHARACTER
21D5 7563      LDA              TERM_INP_BUFF+1 ; GET THE CHARACTER THAT WAS TYPED
21D8 7564      CPI              <^X20>      ; WAS IT A SPACE?
21DA 7565      JNZ              27$        ; BRANCH IF NO
21DD 7566      ;
21DD 7567      ; SPACE WAS TYPE, TICK THE CLOCK
21DD 7568      ;
21DD 7569      LDA              SOFT_CTRL_REG ; CLEAR THE FETCH FLAG
21E0 7570      ANI              <^CFETCH_FLAG> ; ...
21E2 7571      STA              SOFT_CTRL_REG ; ...
21E5 7572      LDA              SOFT_CTRL_REGA ; GET THE CYCLE FLAG
21E8 7573      ANI              CYCLE_FLAG ; IS THIS A CYCLE TICK?
21EA 7574      JZ              26$      ; BRANCH IF NO
21ED 7575      SGL_MIC_INSTR  1         ; STEP THE CLOCK ONE CYCLE
21F5 7576      JMP              15$     ; GO TYPE THE ADDRESS AGAIN
21F8 7577 26$: SGL_TICK      1         ; STEP THE CLOCK ONE TICK
2200 7578      JMP              15$     ; GO TYPE THE ADDRESS AGAIN
2203 7579      ;
2203 7580      ; SPACE WAS NOT TYPED. CALL THE COMMAND PARSER
2203 7581      ;
2203 7582 27$: LDA              SOFT_CTRL_REGA ; CLEAR THE TWO FLAGS
2206 7583      ANI              <^C<CYCLE_FLAG!- ; ...
2206 7584      TICK_FLAG>> ; ...
2208 7585      STA              SOFT_CTRL_REGA ; ...
220B 7586      CALL             GET_CMD_LINE ; CALL THE COMMAND PARSER
220E 7587      JMP              BURST_CLOCK ; RESTART THIS ROUTINE
2211 7588
```

```

2211 7590 :
2211 7591 ; NEITHER THE CYCLE NOR THE TICK FLAG WAS SET. START THE CLOCK
2211 7592 :
2211 7593 30$: LDA SOFT_CTRL_REG ; CLEAR THE FETCH FLAG
2214 7594 ANI <^CFETCH_FLAG> ; ...
2216 7595 STA SOFT_CTRL_REG ; ...
2219 7596 LDA DCS_CTRL_RE_CPY ; GET COPY OF DCS CONTROL REGISTER
221C 7597 ANI <^C<CLK_CTRL_0!CLK_CTRL_1>> ; ACTIVATE BITS TO START CLOCK
221E 7598 STA DCS_CTRL_REG ; START THE CLOCK
2221 7599 :
2221 7600 ; START INTERRUPT DRIVEN INPUT REQUEST AND WAIT FOR EITHER A CONTROL C OR
2221 7601 ; THE CLOCK TO STOP.
2221 7602 :
2221 7603 LDA SOFT_CTRL_REG ; MAKE SURE CONTROL C FLAG
2224 7604 STA BURST_TEMP ; SAVE IN CASE CTRL C FLAG SET
2227 7605 ANI <^CONTROL_C_FLAG>; IS CLEAR
2229 7606 STA SOFT_CTRL_REG ; ...
222C 7607 36$: LDA SOFT_CTRL_REG ; GET THE CONTROL C FLAG
222F 7608 ANI CONTROL_C_FLAG ; IS IT SET?
2231 7609 JNZ 37$ ; BRANCH IF YES
2234 7610 LDA DCS_ADDR_REG_RO ; READ THE CLOCK CONTROL BITS
2237 7611 MOV C,A ; SAVE DCS ADDRESS
2238 7612 ANI <CLK_CTRL_0_RO!- ; DISCARD OTHER BITS
2238 7613 CLK_CTRL_1_RO>
223A 7614 JNZ 37$ ; GET OUT IF CLOCK STOPPED
223D 7615 LDA SOFT_CTRL_REG ; SEE IF BURST_STOP FLAG IS SET
2240 7616 ANI BURST_STOP_FLAG ; ...
2242 7617 JZ 36$ ; BRANCH IF IT'S NOT SET
2245 7618 :
2245 7619 ; CHECK IF DCS ADDRESS REGISTER IS STILL CHANGING
2245 7620 :
2245 7621 LDA M_CLK_ADDR ; GET THE PREVIOUS DCS ADDRESS
2248 7622 MOV B,A ; PUT IN B REG
2249 7623 MOV A,C ; GET CURRENT DCS ADDRESS
224A 7624 ANI <^C<CLK_CTRL_1_RO!-
224A 7625 CLK_CTRL_0_RO>> ; DISCARD CLOCK STATE BITS
224C 7626 STA M_CLK_ADDR ; SAVE FOR NEXT CHECK
224F 7627 CMP B ; HAS IT CHANGED?
2250 7628 JNZ 36$ ; BRANCH IF YES (CLOCK STILL RUNNING)
2253 7629 LXI H,CLK_STOP_MSGS ; GET ADDRESS OF ERROR MESSAGE
2256 7630 SHLD CLK_MSG_ADDR ; SAVE FOR TIMEOUT ROUTINE
2259 7631 :
2259 7632 ; CONTROL C TYPED OR CLOCK STOPPED/HUNG. UNCONDITIONALLY STOP THE CLOCK
2259 7633 :
2259 7634 37$: LDA DCS_CTRL_RE_CPY ; GET COPY OF CONTROL REGISTER
225C 7635 STA DCS_CTRL_REG ; STOP THE CLOCK
225F 7636 :
225F 7637 ; SEE IF CONTROL C WAS TYPED
225F 7638 :
225F 7639 LDA SOFT_CTRL_REG ; GET THE CONTROL C FLAG
2262 7640 ANI CONTROL_C_FLAG ; IS IT SET?
2264 7641 JZ 40$ ; BRANCH IF NO (CLOCK STOPPED)
2267 7642 CALL GET_CMD_LINE ; CALL THE COMMAND PARSER
226A 7643 JMP 30$ ; IF 'CO', START THE CLOCK AGAIN
226D 7644 :

```

```

226D 7645 ; IF EXPECTED TRAP FLAG IS SET, SEE IF TRACE REGISTER IS OUTSIDE DCS SPACE.
226D 7646 ;
226D 7647 40$: LDA BURST_TEMP ; RESTORE THE CONTROL C FLAG
2270 7648 STA SOFT_CTRL_REG ;
2273 7649 LDA SOFT_CTRL_REGA ; GET THE EXPECTED TRAP FLAG
2276 7650 ANI EXP_UTRAP_FLAG ;
2278 7651 JZ 42$ ; BRANCH IF FLAG NOT SET
227B 7652 LDA CS_ADD_TRAP_HGH ; GET UPPER BYTE OF TRAP REGISTER
227E 7653 ANI <^X3F> ; MASK
2280 7654 CPI <^X18> ; GTR THAN 18?
2282 7655 JM 55$ ; BRANCH IF OUT OF DCS SPACE
2285 7656 CPI <^X20> ; LESS THAN 20?
2287 7657 JP 55$ ; BRANCH IF OUT OF DCS SPACE
228A 7658 ;
228A 7659 ; SEE IF 'EXPECT STALL' FLAG IS SET
228A 7660 ;
228A 7661 42$: LDA SOFT_CTRL_REGB ; GET THE EXPECTED STALL FLAG
228D 7662 ANI EXP_STALL_FLAG ;
228F 7663 JNZ 55$ ; BRANCH IF FLAG SET
2292 7664 ;
2292 7665 ; SEE IF THE CLOCK STOPPED BECAUSE OF A 'SOMM'
2292 7666 ;
2292 7667 45$: LDA SOFT_CTRL_REGA ; GET THE SOMM FLAG
2295 7668 ANI SOMM_FLAG ; IS IT SET?
2297 7669 JZ 50$ ; BRANCH IF NO
229A 7670 ;
229A 7671 ; SOMM FLAG IS SET. SEE IF CLOCK STOPPED AT THE CORRECT PLACE
229A 7672 ;
229A 7673 LDA DCS_CTRL_FILE ; CHECK STOP CLOCK BIT IN CONTROL FILE
229D 7674 ANI STOP_CLOCK ;
229F 7675 JZ 50$ ; BRANCH IF NOT SOMM STOP
22A2 7676 ;
22A2 7677 ; CLOCK STOPPED BECAUSE OF 'SOMM'. TYPE THE 'SOMM MESSAGE' AND CALL THE
22A2 7678 ; COMMAND PARSER
22A2 7679 ;
22A2 7680 TYPE CRLF ;
22AB 7681 TYPE SOMM_MSG ; TYPE THE SOMM MESSAGE
22B4 7682 TYPEW SOMM_ADDRESS ; TYPE THE ADDRESS
22BF 7683 LDA DCS_CTRL_FILE ; CHECK IF RETURN TO ZERO BIT IS SET
22C2 7684 ANI DCS_ADDR_CLEAR ;
22C4 7685 JZ 47$ ; BRANCH IF ACTIVE
22C7 7686 TYPE COMMA_MSG ; TYPE A COMMA
22D0 7687 TYPE DCS_ADDR_MSG ; TYPE DCS ADDR =
22D9 7688 LDA DCS_ADDR_REG_RO ; GET CURRENT DCS ADDR
22DC 7689 ANI <^C<CLK_CTRL_1_RO!-
22DC 7690 CLK_CTRL_0_RO>> ; DISCARD CLOCK STATE BITS
22DE 7691 DCR A ; DECREMENT BY 1
22DF 7692 STA BURST_TEMP ; SAVE THE ADDR
22E2 7693 TYPEB BURST_TEMP ; PRINT THE CURRENT DCS ADDR
22ED 7694 47$: CALL GET_CMD_LINE ; CALL THE COMMAND PARSER
22F0 7695 LDA FRONT_PNL_2 ; GET REMOTE, FAULT, & TEST
22F3 7696 ANI <FAULT!TEST!REMOTE>
22F5 7697 STA RD_CTRL_REG ; CLEAR MASTER HALT ENABLE
22F8 7698 ORI <TRAP_HALT_EN!-
22F8 7699 MASTER_HALT_EN> ; RESET MASTER HALT ENABLE

```

```

22FA 7700          STA          RD_CTRL_REG      ;
22FD 7701          JMP          30$              ; IF 'CO' RESTART THE CLOCK
2300 7702          ;
2300 7703          ; SEE IF CLOCK STOPPED AT <DCS ADDRESS>
2300 7704          ;
2300 7705 50$:     LDA          DCS_CTRL_RE_CPY  ; CLEAR THE CONTROL FILE
2303 7706          ORI          CLEAR_CTRL_FILE ; ...
2305 7707          STA          DCS_CTRL_REG    ; ...
2308 7708          ANI          <^CCLEAR_CTRL_FILE>; ...
230A 7709          STA          DCS_CTRL_REG    ; ...
230D 7710          LDA          DCS_ADDR_REG_RO ; GET THE CURRENT DCS ADDRESS
2310 7711          DCR          A              ; BACKUP BY ONE
2311 7712          ANI          <^C<CLK_CTRL_1_RO!-
2311 7713          CLK_CTRL_0_RO>> ; DISCARD CLOCK STATE BITS
2313 7714          MOV          B,A            ; SAVE
2314 7715          LDA          ARG_LIST       ; GET THE <DCS ADDRESS>
2317 7716          CMP          B              ; DID CLOCK STOP AT CORRECT PLACE?
2318 7717          JNZ          60$           ; BRANCH IF NO
231B 7718          ;
231B 7719          ; CLOCK STOPPED CORRECTLY. EXIT
231B 7720          ;
231B 7721 55$:     RET                      ; EXIT
231C 7722          ;
231C 7723          ; CLOCK STOPPED WHEN IT WAS NOT SUPPOSE TO. TYPE AN ERROR MESSAGE AND
231C 7724          ; CALL THE COMMAND PARSER
231C 7725          ;
231C 7726 60$:     TYPE          CRLF
2325 7727          LHLD         CLK_MSG_ADDR   ; GET THE ERROR MESSAGE ADDRESS
2328 7728          TYPE          ; TYPE THE ERROR MESSAGE
232E 7729          LDA          DCS_ADDR_REG_RO ; GET ADDRESS AT WHICH IT STOPPED
2331 7730          ; OR HUNG
2331 7731          ANI          <^C<CLK_CTRL_1_RO!-
2331 7732          CLK_CTRL_0_RO>> ; DISCARD CLOCK STATE BITS
2333 7733          DCR          A              ; ...
2334 7734          STA          BURST_TEMP     ; SAVE
2337 7735          TYPEB       BURST_TEMP    ; TYPE THE ADDRESS
2342 7736          TYPE          CLK_STOP_MSG2 ; TYPE ' TEST NUMBER= '
234B 7737          TYPEB       TEST_NUMBER   ; TYPE THE TEST NUMBER
2356 7738          TYPE          CRLF
235F 7739          TYPE          CLK_STOP_MSG3 ; TYPE 'TPC= '
2368 7740          LHLD         C_INIT_TEST_PC ; GET COMPLIMENT INITIAL TEST PC
236B 7741          XCHG
236C 7742          LHLD         CURRENT_PC    ; GET THE CURRENT TEST PC
236F 7743          DAD         D              ; SUBTRACT INITIAL TEST PC
2370 7744          XCHG
2371 7745          LHLD         TITLE_LENGTH  ; GET THE TITLE LENGTH
2374 7746          DAD         D              ; ...
2375 7747          INX          H              ; ACCOUNT FOR LENGTH BYTE
2376 7748          SHLD        TYPE_ERR_BUFFER ; SAVE
2379 7749          TYPEW       TYPE_ERR_BUFFER ; TYPE IT
2384 7750          TYPE          CLK_STOP_MSG4 ; TYPE ' TRAP ADDRESS= '
238D 7751          LHLD         CS_ADD_TRAP_LOW ; GET CONTENTS OF TRAP REGISTER
2390 7752          MOV          A,H           ; DISCARD UPPER TWO BITS
2391 7753          ANI          <^X3F>        ; OF TRAP REGISTER
2393 7754          MOV          H,A           ; ...

```

```
2394 7755      SHLD      TYPE_ERR_BUFFER ;
2397 7756      TYPEW     TYPE_ERR_BUFFER ; TYPE THE TRAP ADDRESS
23A2 7757      ;
23A2 7758      ; type the index values if they are in use
23A2 7759      ;
23A2 7760      TYPE      TAB           ; TYPE 8 SPACES
23AB 7761      LXI       H,0          ; CLEAR THE TEMP BUFFER
23AE 7762      SHLD     TYPE_ERR_BUFFER ; ...
23B1 7763      SHLD     TYPE_ERR_BUFFER+2 ; ...
23B4 7764      XRA       A           ; GET THE I INDEX CODE IN A REGISTER
23B5 7765      CALL     GET_INDEX_VALUE ; GET THE CURRENT VALUE OF I INDEX
23B8 7766      JC       70$         ; BRANCH IF I NOT BEING USED
23BB 7767      PUSH     B           ; PUT VALUE IN H & L
23BC 7768      POP      H           ; ...
23BD 7769      INX      H           ; MAKE LOOP COUNT START AT ONE
23BE 7770      SHLD     TYPE_ERR_BUFFER ; SAVE I INDEX VALUE
23C1 7771      TYPEW     TYPE_ERR_BUFFER ; TYPE THE I INDEX VALUE
23CC 7772
23CC 7773 70$:  MVI       A,2         ; GET J INDEX CODE
23CE 7774      CALL     GET_INDEX_VALUE ; GET THE CURRENT VALUE OF I INDEX
23D1 7775      JC       71$         ; BRANCH IF I NOT BEING USED
23D4 7776      PUSH     B           ; PUT VALUE IN H & L
23D5 7777      POP      H           ; ...
23D6 7778      INX      H           ; MAKE LOOP COUNT START AT ONE
23D7 7779      SHLD     TYPE_ERR_BUFFER ; SAVE I INDEX VALUE
23DA 7780      TYPE     COMMA_MSG    ; TYPE A COMMA SPACE
23E3 7781      TYPEW     TYPE_ERR_BUFFER ; TYPE THE I INDEX VALUE
23EE 7782
23EE 7783 71$:  MVI       A,4         ; GET K INDEX CODE
23F0 7784      CALL     GET_INDEX_VALUE ; GET THE CURRENT VALUE OF I INDEX
23F3 7785      JC       61$         ; BRANCH IF I NOT BEING USED
23F6 7786      PUSH     B           ; PUT VALUE IN H & L
23F7 7787      POP      H           ; ...
23F8 7788      INX      H           ; MAKE LOOP COUNT START AT ONE
23F9 7789      SHLD     TYPE_ERR_BUFFER ; SAVE I INDEX VALUE
23FC 7790      TYPE     COMMA_MSG    ; TYPE A COMMA SPACE
2405 7791      TYPEW     TYPE_ERR_BUFFER ; TYPE THE I INDEX VALUE
2410 7792 61$:  CALL     GET_CMD_LINE ; CALL THE COMMAND PARSER
2413 7793      JMP      61$         ; DON'T ALLOW CONTINUE
2416 7794
2416 7795
```

```
2416 7797 .SBTTL END TEST ROUTINE
2416 7798 ;++
2416 7799 ;
2416 7800 ; FUNCTIONAL DESCRIPTION:
2416 7801 ;
2416 7802 ; THIS ROUTINE IS USED TO TERMINATE A TEST. IT SHOULD ALWAYS BE
2416 7803 ; THE LAST INSTRUCTION OF THE TEST (EXCEPT FOR THE LAST TEST
2416 7804 ; OF A TEST SEQUENCE). FOLLOWING IS THE ALGORITHM FOR THIS ROUTINE:
2416 7805 ;
2416 7806 ; READ THE NEXT OVERLAY AND EXIT.
2416 7807 ;
2416 7808 ; CALLING SEQUENCE:
2416 7809 ;
2416 7810 ; ENDTST
2416 7811 ;
2416 7812 ; IMPLICIT INPUTS:
2416 7813 ;
2416 7814 ; TEST_PC - CONTAINS THE CURRENT TEST PC
2416 7815 ;
2416 7816 ; IMPLICIT OUTPUTS:
2416 7817 ;
2416 7818 ; NONE
2416 7819 ;
2416 7820 ;--
2416 7821
2416 7822 END_TEST:
2416 7823 LDA TEST_NUMBER ; GET THE CURRENT TEST NUMBER
2419 7824 MOV B,A ;
241A 7825 LDA TOTAL_NMB_TESTS ; GET THE TOTAL NUMBER OF TESTS
241D 7826 CMP B ; MORE TESTS TO EXECUTE?
241E 7827 JZ 10$ ; BRANCH IF NO
2421 7828 CALL CPU_INIT ; INIT THE CPU
2424 7829 CALL READ_OVERLAY ; READ IN THE NEXT TEST
2427 7830 LHLD INIT_TEST_PC ; INITIALIZE THE TEST PC
242A 7831 SHLD TEST_PC ;
242D 7832 10$: RET ; EXIT
242E 7833
```

```
242E 7835      .SBTTL      END PASS ROUTINE
242E 7836      ;**
242E 7837      ;
242E 7838      ; FUNCTIONAL DESCRIPTION:
242E 7839      ;
242E 7840      ;         THIS ROUTINE INCREMENTS THE PASS COUNT AND RETURNS TO THE
242E 7841      ;         PROGRAM_INIT ROUTINE.
242E 7842      ;
242E 7843      ; CALLING SEQUENCE:
242E 7844      ;
242E 7845      ;         ENDPASS
242E 7846      ;
242E 7847      ; IMPLICIT INPUTS:
242E 7848      ;
242E 7849      ;         PASS_COUNT      - CONTAINS THE CURRENT PASS COUNT
242E 7850      ;
242E 7851      ; IMPLICIT OUTPUTS:
242E 7852      ;
242E 7853      ;         PASS_COUNT      - CONTAINS THE CURRENT PASS COUNT PLUS 1
242E 7854      ;
242E 7855      ;--
242E 7856
242E 7857      END_PASS:
242E 7858          LDA      PASS_COUNT      ; GET THE PASS COUNT
2431 7859          INR      A                  ; INCREMENT
2432 7860          STA      PASS_COUNT      ; SAVE
2435 7861          POP      B                  ; THROW AWAY PC OF CALL
2436 7862          RET                          ; RETURN TO PROGRAM_INIT ROUTINE
2437 7863
```

```
2437 7865 .SBTTL LOAD FIELD ROUTINE
2437 7866 ;++
2437 7867 ;
2437 7868 ; FUNCTIONAL DESCRIPTION:
2437 7869 ;
2437 7870 ; THIS ROUTINE IS USED TO LOAD A FIELD OF A MICRO WORD. IF BIT 15
2437 7871 ; OF EITHER THE STARTING BIT NUMBER OR ENDING BIT NUMBER ARGUEMNT
2437 7872 ; IS SET, THE BIT NUMBER IS A POINTER (OPTIONALLY INDEXED) TO THE
2437 7873 ; REQUESTED BIT NUMBER.
2437 7874 ;
2437 7875 ; CALLING SEQUENCE:
2437 7876 ;
2437 7877 ; LDFIELD SRC_ADDRESS,[SRC_INDEX],[DATA_TYPE],DEST_ADDRESS,START_BIT,END_BIT -
cont> ,[START_BIT_INDEX],[END_BIT_INDEX],[INHIBIT_PARITY]
2437 7878 ;
2437 7879 ; IMPLICIT INPUTS:
2437 7880 ;
2437 7881 ; ARGUMENT LIST:
2437 7882 ;
2437 7883 ; OFFSET 0 - SOURCE ADDRESS LOW
2437 7884 ; OFFSET 1 - SOURCE ADDRESS HIGH
2437 7885 ; OFFSET 2 - SOURCE INDEX CODE
2437 7886 ; OFFSET 3 - DATA TYPE CODE
2437 7887 ; OFFSET 4 - DESTINATION ADDRESS LOW
2437 7888 ; OFFSET 5 - DESTINATION ADDRESS HIGH
2437 7889 ; OFFSET 6 - STARTING BIT NUMBER LOW
2437 7890 ; OFFSET 7 - STARTING BIT NUMBER HIGH
2437 7891 ; OFFSET 8 - ENDING BIT NUMBER LOW
2437 7892 ; OFFSET 9 - ENDING BIT NUMBER HIGH
2437 7893 ; OFFSET 10- LONLIT FLAG
2437 7894 ; OFFSET 11- STARTING BIT NUMBER INDEX CODE
2437 7895 ; OFFSET 12- ENDING BIT NUMBER INDEX CODE
2437 7896 ; OFFSET 13- INHIBIT PARITY CODE (-1 EQL INHIBIT)
2437 7897 ; OFFSET 14- INHIBIT SCRAMBLE CODE (-1 EQL INHIBIT)
2437 7898 ;
2437 7899 ; IMPLICIT OUTPUTS:
2437 7900 ;
2437 7901 ; THE SPECIFIED FIELD OF THE DESTINATION IS LOADED WITH THE SOURCE DATA
2437 7902 ;
2437 7903 ;--
2437 7904 ;
2437 7905 LOAD_FIELD:
2437 7906 LDA ARG_LIST+2 ; GET THE INDEX CODE
243A 7907 CALL GET_INDEX_VALUE ; GET THE CURRENT VALUE
243D 7908 LDA ARG_LIST+3 ; GET THE DATA TYPE
2440 7909 CALL MULTIPLY_A_BC ; MULTIPLY DATA TYPE BY INDEX VALUE
2443 7910 ;
2443 7911 ; D & E NOW CONTAIN THE INDEX VALUE FOR THE SPECIFIED DATA TYPE
2443 7912 ;
2443 7913 ; LHL D ARG_LIST ; GET THE SOURCE ADDRESS
2446 7914 DAD D ; INDEX IT
2447 7915 XCHG
2448 7916 LHL D INIT_TEST_PC ; GET INITIAL TEST PC
244B 7917 DAD D ; RELOCATE SOURCE ADDRESS
244C 7918 ;
244C 7919 ; CHECK IF LONLIT SPECIFIED
```

```

244C 7920 ;
244C 7921 LDA ARG_LIST+10 ; GET THE LONLIT FLAG
244F 7922 ORA A ; SET THE CONDITION CODES
2450 7923 JM 5$ ; BRANCH IF NO LONLIT
2453 7924 LXI D,TEMP_BUFFER ; PUT COMPLIMENT DATA IN TEMP BUFFER
2456 7925 MVI C,4 ; SETUP LOOP TO MOVE 4 BYTES
2458 7926 3$: MOV A,M ; GET A BYTE OF THE SOURCE
2459 7927 CMA ; COMPLIMENT IT
245A 7928 STAX D ; PUT IN TEMP BUFFER
245B 7929 INX D ; INCREMENT THE POINTERS
245C 7930 INX H ;
245D 7931 DCR C ; DONE 4 BYTES?
245E 7932 JNZ 3$ ; BRANCH IF NO
2461 7933 LXI H,TEMP_BUFFER ; PUT ADDRESS OF DATA IN H & L
2464 7934 ;
2464 7935 ; GET THE DESTINATION ADDRESS AND INSERT THE FIELD
2464 7936 ;
2464 7937 5$: PUSH ; SAVE SOURCE ADDRESS
2465 7938 LHL ARG_LIST+4 ; GET THE DESTINATION ADDRESS
2468 7939 XCHG
2469 7940 LHL INIT_TEST_PC ; GET INITIAL TEST PC
246C 7941 DAD D ; RELOCATE DESTINATION ADDRESS
246D 7942 XCHG
246E 7943 ;
246E 7944 ; D & E CONTAIN DESTINATION ADDRESS
246E 7945 ; (SP) CONTAINS THE SOURCE ADDRESS
246E 7946 ;
246E 7947 ; GET THE STARTING AND ENDING BIT NUMBERS
246E 7948 ;
246E 7949 PUSH D ; SAVE D & E
246F 7950 LHL ARG_LIST+6 ; GET THE STARTING BIT NUMBER
2472 7951 MOV A,H ; SEE IF BIT 15 SET
2473 7952 ANI <^X80> ;
2475 7953 JZ 10$ ; BRANCH IF ABSOLUTE DATA
2478 7954 MOV A,H ; CLEAR BIT 15
2479 7955 ANI <^X7F> ;
247B 7956 MOV H,A ;
247C 7957 PUSH H ; SAVE ADDRESS
247D 7958 LDA ARG_LIST+11 ; GET STARTING INDEX CODE
2480 7959 CALL GET_INDEX_VALUE ; GET THE CURRENT INDEX VALUE
2483 7960 POP H ; GET ADDRESS BACK
2484 7961 DAD B ; ADD CURRENT INDEX VALUE
2485 7962 XCHG
2486 7963 LHL INIT_TEST_PC ; GET THE INITIAL TEST PC
2489 7964 DAD D ; ADD TO INDEX VALUE ADDRESS
248A 7965 MOV L,M ; GET STARTING BIT NUMBER
248B 7966 10$: MOV B,L ; PUT STARTING BIT IN B REG
248C 7967 ;
248C 7968 ; GET THE END VALUE
248C 7969 ;
248C 7970 LHL ARG_LIST+8 ; GET THE ENDING BIT NUMBER
248F 7971 MOV A,H ; SEE IF BIT 15 SET
2490 7972 ANI <^X80> ;
2492 7973 JZ 15$ ; BRANCH IF ABSOLUTE DATA
2495 7974 MOV A,H ; CLEAR BIT 15

```

```
2496 7975 ANI <^X7F> ; ...
2498 7976 MOV H,A ; ...
2499 7977 PUSH B ; SAVE STARTING BIT VALUE
249A 7978 PUSH H ; SAVE ADDRESS
249B 7979 LDA ARG_LIST+12 ; GET STARTING INDEX CODE
249E 7980 CALL GET_INDEX_VALUE ; GET THE CURRENT INDEX VALUE
24A1 7981 POP H ; GET ADDRESS BACK
24A2 7982 DAD B ; ADD CURRENT INDEX VALUE
24A3 7983 XCHG
24A4 7984 LHLD INIT_TEST_PC ; GET THE INITIAL TEST PC
24A7 7985 DAD D ; ADD TO INDEX VALUE ADDRESS
24A8 7986 MOV L,M ; GET STARTING BIT NUMBER
24A9 7987 POP B ; RESTORE STARTING BIT VALUE
24AA 7988 15$: MOV C,L ; PUT ENDING BIT VALUE IN C REG
24AB 7989 POP D ; RESTORE DESTINATION ADDRESS
24AC 7990 POP H ; RESTORE SOURCE ADDRESS
24AD 7991 LDA ARG_LIST+14 ; GET THE INHIBIT SCRAMBLE FLAG
24B0 7992 PUSH $PSW ; PASS IT ON THE STACK
24B1 7993 LDA ARG_LIST+13 ; GET THE INHIBIT PARITY FLAG
24B4 7994 CALL INSERT_FIELD ; GENERATE THE FIELD
24B7 7995 RET
24B8 7996
```

```

24B8 7998      .SBTTL      LOAD INDEX ROUTINE
24B8 7999      ;++
24B8 8000      ;
24B8 8001      ; FUNCTIONAL DESCRIPTION:
24B8 8002      ;
24B8 8003      ;     THIS ROUTINE INITIALIZES THE 'STARTING_VALUE' OF THE 'LOOP_NAME'
24B8 8004      ;     PARAMETER.
24B8 8005      ;
24B8 8006      ; CALLING SEQUENCE:
24B8 8007      ;
24B8 8008      ;     LD INDEX      LOOP_NAME,STARTING_VALUE
24B8 8009      ;
24B8 8010      ; IMPLICIT INPUTS:
24B8 8011      ;
24B8 8012      ;     ARGUMENT LIST:
24B8 8013      ;
24B8 8014      ;     OFFSET 0 - INDEX NAME CODE
24B8 8015      ;     OFFSET 1 - STARTING VALUE LOW
24B8 8016      ;     OFFSET 2 - STARTING VALUE HIGH
24B8 8017      ;     OFFSET 3 - VALUE FLAG (0 = LITERAL OTHERWISE, ADDRESS)
24B8 8018      ;     OFFSET 4 - VALUE INDEX CODE (ONLY USED IF VALUE FLAG IS NON 0)
24B8 8019      ;
24B8 8020      ;     INDEX_NAM_TBL - TABLE THAT CONTAINS THE ADDRESS OF THE I, J, OR
24B8 8021      ;     K INDEX TABLES.
24B8 8022      ;
24B8 8023      ; IMPLICIT OUTPUTS:
24B8 8024      ;
24B8 8025      ;     THE I, J, OR K STARTING VALUE IS INITIALIZED.
24B8 8026      ;
24B8 8027      ;--
24B8 8028      ;
24B8 8029      LOAD_INDEX:
24B8 8030      LDA      ARG_LIST      ; GET THE INDEX NAME BYTE
24B8 8031      CALL     GET_INDEX_VALUE ; GET ADDRESS OF THE INDEX TABLE
24B8 8032      PUSH     H              ; SAVE ADDRESS OF TABLE
24B8 8033      LDA      ARG_LIST+3    ; GET THE VALUE FLAG
24C2 8034      ORA      A              ; SET THE CONDITION CODES
24C3 8035      JZ       10$          ; BRANCH IF VALUE IS A LITERAL
24C6 8036      ;
24C6 8037      ; VALUE TO LOAD IS IN A TABLE
24C6 8038      ;
24C6 8039      LDA      ARG_LIST+4    ; GET THE INDEX CODE FOR THE VALUE
24C9 8040      CALL     GET_INDEX_VALUE ; GET THE CURRENT VALUE
24CC 8041      JC       5$           ; BRANCH IF NO INDEX SPECIFIED
24CF 8042      MVI     A,2          ; MULTIPLY INDEX VALUE BY 2
24D1 8043      CALL     MULTIPLY_A_BC ;
24D4 8044      LHLD    ARG_LIST+1    ; GET THE TABLE ADDRESS
24D7 8045      DAD     D            ; INDEX BY INDEX VALUE
24D8 8046      XCHG    ;             ; PUT ADDRESS IN D & E
24D9 8047      LHLD    INIT_TEST_PC ; GET RELOCATION FACTOR
24DC 8048      DAD     D            ; RELOCATE TABLE ADDRESS
24DD 8049      XCHG    ;             ; PUT IN D & E
24DE 8050      POP     H            ; RESTORE ADDRESS OF INDEX TABLE
24DF 8051      JMP     20$          ; LOAD THE VALUE
24E2 8052      ;

```

```
24E2 8053 ; COME HERE WHEN THERE IS AND ADDRESS THAT POINTS TO THE VALUE TO BE LOADED
24E2 8054 ; AND
24E2 8055 ; THERE IS NO INDEX SPECIFIED TO INDEX INTO THAT ADDRESS WITH
24E2 8056 ;
24E2 8057 5$: LHL D ARG_LIST+1 ; GET THE ADDRESS
24E5 8058 XCHG ; PUT THE ADDRESS IN D & E
24E6 8059 LHL D INIT_TEST_PC ; GET THE RELOCATION FACTOR
24E9 8060 DAD D ; RELOCATE THE ADDRESS
24EA 8061 XCHG ; PUT IN D & E
24EB 8062 POP H ; RESTORE ADDRESS OF INDEX TABLE
24EC 8063 JMP 20$ ; LOAD THE VALUE
24EF 8064 ;
24EF 8065 ; (SP) - ADDRESS OF INDEX TABLE
24EF 8066 ; ARG LIST CONTAINS THE VALUE TO LOAD
24EF 8067 ;
24EF 8068 10$: POP H ; RESTORE ADDRESS OF INDEX TABLE
24F0 8069 LXI D,ARG_LIST+1 ; GET ADDRESS OF STARTING VALUE
24F3 8070 ;
24F3 8071 ; H & L - CONTAIN THE ADDRESS OF THE INDEX TABLE
24F3 8072 ; D & E - CONTAIN THE ADDRESS OF THE VALUE TO LOAD
24F3 8073 ;
24F3 8074 20$: LDAX D ; GET LOW BYTE OF STARTING VALUE
24F4 8075 MOV M,A ; PUT IN INDEX TABLE
24F5 8076 INX D ; INCREMENT THE POINTERS
24F6 8077 INX H ;
24F7 8078 LDAX D ; GET HIGH BYTE OF STARTING VALUE
24F8 8079 MOV M,A ; STORE IN INDEX TABLE
24F9 8080 RET ; EXIT
24FA 8081
```

```
24FA 8083 .SBTTL " INCREMENT INDEX ROUTINE
24FA 8084 ;**
24FA 8085 ;
24FA 8086 ; FUNCTIONAL DESCRIPTION:
24FA 8087 ;
24FA 8088 ; THIS ROUTINE INCREMENTS THE CURRENT VALUE OF THE 'LOOP_NAME'
24FA 8089 ; INDEX.
24FA 8090 ;
24FA 8091 ; CALLING SEQUENCE:
24FA 8092 ;
24FA 8093 ; INCINDEX            LOOP_NAME
24FA 8094 ;
24FA 8095 ; IMPLICIT INPUTS:
24FA 8096 ;
24FA 8097 ; ARGUMENT LIST:
24FA 8098 ;
24FA 8099 ;            OFFSET 0 - INDEX NAME CODE
24FA 8100 ;
24FA 8101 ; INDEX_NAME_TBL
24FA 8102 ;
24FA 8103 ; --
24FA 8104 ;
24FA 8105 INCREMENT_INDEX:
24FA 8106        LDA            ARG_LIST            ; GET THE INDEX NAME CODE
24FD 8107        CALL        GET_INDEX_VALUE       ; GET THE CURRENT INDEX VALUE
2500 8108        INX            B                ; INCREMENT THE CURRENT VALUE
2501 8109        MOV           M,C               ; PUT BACK IN THE TABLE
2502 8110        INX            H                ; ...
2503 8111        MOV           M,B               ; ...
2504 8112        RET                             ; EXIT
2505 8113
```

```
2505 8115      .SBTTL      SAVE INDEX ROUTINE
2505 8116      ;**
2505 8117      ;
2505 8118      ; FUNCTIONAL DESCRIPTION:
2505 8119      ;
2505 8120      ;     THIS ROUTINE SAVES THE CURRENT VALUE OF THE SPECIFIED
2505 8121      ;     INDEX NAME IN THE SPECIFIED ADDRESS.
2505 8122      ;
2505 8123      ; CALLING SEQUENCE:
2505 8124      ;
2505 8125      ;     SAVEINDEX      INDEX_NAME,ADDRESS
2505 8126      ;
2505 8127      ; IMPLICIT INPUTS:
2505 8128      ;
2505 8129      ;     ARGUMENT LIST:
2505 8130      ;
2505 8131      ;     OFFSET 0 - INDEX NAME CODE
2505 8132      ;     OFFSET 1 - ADDRESS LOW
2505 8133      ;     OFFSET 2 - ADDRESS HIGH
2505 8134      ;
2505 8135      ;--
2505 8136      ;
2505 8137      SAVE_INDEX:
2505 8138      LDA      ARG_LIST      ; GET INDEX NAME CODE
2508 8139      CALL     GET_INDEX_VALUE ; GET THE CURRENT VALUE
2508 8140      LHL     ARG_LIST+1      ; GET DESTINATION ADDRESS
250E 8141      XCHG      ; PUT IN D & E
250F 8142      LHL     INIT_TEST_PC   ; RELOCATE THE ADDRESS
2512 8143      DAD      D              ;
2513 8144      MOV     M,C           ; SAVE THE INDEX VALUE
2514 8145      INX     H              ; ...
2515 8146      MOV     M,B           ;
2516 8147      RET      ; EXIT
2517 8148
```

```
2517 8150      .SBTTL      ENABLE MICRO TRAP ROUTINE
2517 8151      ;++
2517 8152      ;
2517 8153      ; FUNCTIONAL DESCRIPTION:
2517 8154      ;
2517 8155      ;         THIS PSEUDO INSTRUCTION SETS A SOFTWARE FLAG THAT CONTROLS THE
2517 8156      ;         BRSTCLOCK ROUTINE.
2517 8157      ;
2517 8158      ; CALLING SEQUENCE:
2517 8159      ;
2517 8160      ;         EXPUTRAP
2517 8161      ;
2517 8162      ; IMPLICIT INPUTS:
2517 8163      ;
2517 8164      ;         ARGUMENT LIST:
2517 8165      ;
2517 8166      ;         NONE
2517 8167      ;
2517 8168      ;--
2517 8169      ;
2517 8170      ENABLE_TRAP:
2517 8171          LDA          SOFT_CTRL_REGA ; GET THE CONTROL REGISTER
251A 8172          ORI          EXP_UTRAP_FLAG ; SET THE FLAG
251C 8173          STA          SOFT_CTRL_REGA ; RESTORE THE CONTROL REGISTER
251F 8174          RET
2520 8175
```

```
2520 8177      .SBTTL      ENABLE MICRO STALL ROUTINE
2520 8178      ;++
2520 8179      ;
2520 8180      ; FUNCTIONAL DESCRIPTION:
2520 8181      ;
2520 8182      ;     THIS PSEUDO INSTRUCTION SETS A SOFTWARE FLAG THAT CONTROLS THE
2520 8183      ;     BRSTCLOCK ROUTINE.
2520 8184      ;
2520 8185      ; CALLING SEQUENCE:
2520 8186      ;
2520 8187      ;     EXPSTALL
2520 8188      ;
2520 8189      ; IMPLICIT INPUTS:
2520 8190      ;
2520 8191      ;     ARGUMENT LIST:
2520 8192      ;
2520 8193      ;     NONE
2520 8194      ;
2520 8195      ;--
2520 8196      ;
2520 8197      ENABLE_STALL:
2520 8198          LDA          SOFT_CTRL_REGB ; GET THE CONTROL REGISTER
2523 8199          ORI          EXP_STALL_FLAG ; SET THE FLAG
2525 8200          STA          SOFT_CTRL_REGB ; RESTORE THE CONTROL REGISTER
2528 8201          RET
2529 8202
```

```

2529 8204      .SBTTL      ERROR LOG ROUTINE
2529 8205      ;++
2529 8206      ;
2529 8207      ; FUNCTIONAL DESCRIPTION:
2529 8208      ;
2529 8209      ;     THIS ROUTINE IS USED TO INSERT ENTRIES IN THE ERROR LOG. THE ERROR
2529 8210      ;     LOG KEEPS TRACK OF SINGLE BIT ERRORS IN THE DYNAMIC MEMORY TEST.
2529 8211      ;
2529 8212      ;     THE LOG CONSISTS OF 260 BYTES LOCATED AT THE VERY TOP OF MEMORY.
2529 8213      ;     THE FIRST 4 BYTES CONTAINS THE NUMBER OF ENTRIES IN THE LOG.
2529 8214      ;
2529 8215      ;     IF THE INIT_ERR_LOG FLAG IS SET WHEN THIS ROUTINE IS CALLED, THE
2529 8216      ;     NUMBER OF ENTRIES IN THE LOG IS SET TO ZERO AND THE FLAG CLEARED
2529 8217      ;     BEFORE THE CURRENT DATA IS PLACED IN THE LOG. THE INIT_ERR_LOG FLAG
2529 8218      ;     IS SET BY A 'DIAGNOSE' COMMAND.
2529 8219      ;
2529 8220      ; CALLING SEQUENCE:
2529 8221      ;
2529 8222      ;     ERRLOG
2529 8223      ;
2529 8224      ; IMPLICIT INPUTS:
2529 8225      ;
2529 8226      ;     ARGUMENT LIST:
2529 8227      ;
2529 8228      ;     NONE
2529 8229      ;
2529 8230      ;     TEST_SIZE - CONTAINS THE NUMBER OF RECORDS (128 BYTES EACH) IN THE
2529 8231      ;     CURRENT TEST. THE TEST MUST NOT OVERFLOW INTO THE LOG.
2529 8232      ;
2529 8233      ;     WH-REGISTER - CONTAINS PHYSICAL ADDRESS OF ERROR
2529 8234      ;     MH-REGISTER - CONTAINS SYNDROME OF FAILURE IN BITS <6:0>
2529 8235      ;
2529 8236      ; ROUTINE VALUE:
2529 8237      ;
2529 8238      ;     ERROR_FLAG - SET IF ERROR LOG IS FULL.
2529 8239      ;--
2529 8240      ERROR_LOG:
2529 8241          LDA          TEST_SIZE      ; GET CURRENT TEST SIZE
2529 8242          MOV          C,A              ; SETUP TO CALCULATE # OF BYTES
2529 8243          XRA          A                ; ...
2529 8244          MOV          B,A              ; ...
2529 8245          MVI          A,128          ; ...
2531 8246          CALL          MULTIPLY_A_BC ; GET BYTE LENGTH OF TEST IN D & E
2534 8247          LXI          H,TEST_BUFFER ; GET STARTING ADDRESS OF TEST
2537 8248          DAD          D            ; GET ENDING ADDRESS OF TEST
2538 8249          MOV          A,H          ; CHECK IF ERROR LOG OVERFLOW
2539 8250          CPI          <^XB6>      ; CHECK UPPER 8 BITS
253B 8251          JZ           6$          ; BRANCH TO CHECK LOW HALF
253E 8252          JNC          1$          ; BRANCH IF TOO BIG
2541 8253          JMP          5$          ; OTHERWISE ITS OK
2544 8254 6$:          MOV          A,L          ; GET LOW HALF
2545 8255          CPI          <^XFC>      ; CHECK LOW HALF
2547 8256          JC           5$          ; BRANCH IF OK
254A 8257          ;
254A 8258      ; TEST OVERFLOWS ERROR LOG

```

```
254A 8259 ;  
254A 8260 1$: TYPE CRLF  
2553 8261 TYPE LOG_MSG_1 ; TYPE THE ERROR MESSAGE  
255C 8262 TYPE CRLF  
2565 8263 JMP LOG_FAIL ; EXIT  
2568 8264 ;  
2568 8265 ; CHECK IF ERR_LOG_INIT SET  
2568 8266 ;  
2568 8267 5$: LXI H,ERR_LOG_ADR ; GET START ADDRESS OF ERROR LOG B6FC(X)  
256B 8268 MVI E,4 ; SETUP ERROR LOG INCREMENT  
256D 8269 MVI D,0 ;  
256F 8270 LDA SOFT_CTRL_REGB ; GET THE FLAG  
2572 8271 MOV B,A ; SAVE  
2573 8272 ANI ERR_LOG_INIT ; IS FLAG SET?  
2575 8273 JZ 10$ ; BRANCH IF NO  
2578 8274 MOV M,D ; SET LOG COUNT TO ZERO  
2579 8275 MOV A,B ; CLEAR THE FLAG  
257A 8276 ANI <^C<ERR_LOG_INIT>>; ...  
257C 8277 STA SOFT_CTRL_REGB ; ...  
257F 8278 ;  
257F 8279 ; CHECK IF ERROR LOG FULL  
257F 8280 ;  
257F 8281 10$: MOV A,M ; GET NUMBER OF ENTRIES IN LOG  
2580 8282 CPI 64 ; LOG FULL?  
2582 8283 JZ LOG_FAIL ; BRANCH IF YES  
2585 8284 ;  
2585 8285 ; SEARCH THE ERROR LOG FOR THIS PHYSICAL ADDRESS  
2585 8286 ;  
2585 8287 MOV B,A ; GET NUMBER OF ENTRIES IN LOG  
2586 8288 LDA WH_REG_BYTE_0 ; GET THE PHYSICAL ADDRESS  
2589 8289 STA LOG_ADDRESS ; ...  
258C 8290 LDA WH_REG_BYTE_1 ; ...  
258F 8291 STA LOG_ADDRESS+1 ; ...  
2592 8292 LDA WH_REG_BYTE_2 ; ...  
2595 8293 STA LOG_ADDRESS+2 ; ...  
2598 8294 LDA MH_REG_BYTE_0 ; GET THE SYNDROME  
259B 8295 ANI <^X7F> ; ...  
259D 8296 STA LOG_SYNDROME ; SAVE  
25A0 8297 MOV A,B ; CHECK IF LOG EMPTY  
25A1 8298 ORA A ; ...  
25A2 8299 JZ 25$ ; BRANCH IF LOG EMPTY  
25A5 8300 ;  
25A5 8301 ; SCAN THE LOG  
25A5 8302 ;  
25A5 8303 15$: DAD D ; POINT AT NEXT ENTRY  
25A6 8304 LDA LOG_ADDRESS ; GET CURRENT LOG ADDRESS BYTE 0  
25A9 8305 CMP M ; THIS ITEM IN LOG?  
25AA 8306 JNZ 20$ ; BRANCH IF NO  
25AD 8307 LDA LOG_ADDRESS+1 ; GET NEXT BYTE OF CURRENT ADDRESS  
25B0 8308 INX H  
25B1 8309 CMP M ; STILL THIS ITEM?  
25B2 8310 DCX H  
25B3 8311 JNZ 20$ ; BRANCH IF NO  
25B6 8312 LDA LOG_ADDRESS+2 ; GET 3RD BYTE OF CURRENT ADDRESS  
25B9 8313 INX H
```

```

25BA 8314      INX      H
25BB 8315      CMP      M      ; STILL THIS ITEM?
25BC 8316      DCX      H
25BD 8317      DCX      H
25BE 8318      JNZ      20$    ; BRANCH IF NO
25C1 8319      LDA      LOG_SYNDROME ; GET CURRENT SYNDROME
25C4 8320      INX      H
25C5 8321      INX      H
25C6 8322      INX      H
25C7 8323      CMP      M      ; THIS SYNDROME?
25C8 8324      JZ       LOG_SUCCESS ; BRANCH IF YES
25CB 8325      DCX      H
25CC 8326      DCX      H
25CD 8327      DCX      H
25CE 8328      20$:    DCR      B      ; CHECKED ALL ENTRIES IN LOG?
25CF 8329      JNZ      15$    ; BRANCH IF NO
25D2 8330      ;
25D2 8331      ; CHECKED ALL ENTRIES IN ERROR LOG AND THIS ADDRESS AND SYNDROME ARE NOT
25D2 8332      ; THERE. PUT THEM IN THE ERROR LOG
25D2 8333      ;
25D2 8334      25$:    DAD      D      ; POINT AT FREE ELEMENT IN LOG
25D3 8335      LDA      LOG_ADDRESS ; PUT ADDRESS AND SYNDROME IN LOG
25D6 8336      MOV      M,A      ; ...
25D7 8337      INX      H
25D8 8338      LDA      LOG_ADDRESS+1 ; ...
25DB 8339      MOV      M,A      ; ...
25DC 8340      INX      H
25DD 8341      LDA      LOG_ADDRESS+2 ; ...
25E0 8342      MOV      M,A      ; ...
25E1 8343      INX      H
25E2 8344      LDA      LOG_SYNDROME ; ...
25E5 8345      MOV      M,A      ; ...
25E6 8346      LXI      H,ERROR_LOG_ADR ; GET ADDRESS OF LOG COUNT B6FC(X)
25E9 8347      MOV      A,M      ; GET LOG COUNT
25EA 8348      INR      A      ; COUNT THIS ENTRY
25EB 8349      MOV      M,A      ; SAVE IN LOG
25EC 8350      JMP      LOG_SUCCESS ; EXIT
25EF 8351      LOG_FAIL:
25EF 8352      LDA      SOFT_CTRL_REGA ; GET ERROR FLAGS
25F2 8353      ORI      <ERROR_FLAG!-
25F2 8354      LOOP_ERROR_FLAG>; SET THE FLAGS
25F4 8355      ANI      <^CVBUS_COMPARE>; ...
25F6 8356      STA      SOFT_CTRL_REGA ; ...
25F9 8357      JMP      LOG_EXIT
25FC 8358      LOG_SUCCESS:
25FC 8359      LDA      SOFT_CTRL_REGA ; GET THE FLAGS
25FF 8360      ANI      <^C<LOOP_ERROR_FLAG!-
25FF 8361      VBUS_COMPARE>>; CLEAR THEM
2601 8362      STA      SOFT_CTRL_REGA ; ...
2604 8363      CALL     CHECK_QA_FLAG ; CHECK IF QA FLAG SET
2607 8364      LOG_EXIT:
2607 8365      RET
2608 8366

```

```
2608 8368      .SBTTL      PRINT STRING ROUTINE
2608 8369      ;++
2608 8370      ;
2608 8371      ; FUNCTIONAL DESCRIPTION:
2608 8372      ;
2608 8373      ;     THIS ROUTINE IS USED TO PRINT A COUNTED ASCII STRING
2608 8374      ;
2608 8375      ; CALLING SEQUENCE:
2608 8376      ;
2608 8377      ;     PRINT_S      POINTER
2608 8378      ;
2608 8379      ; IMPLICIT INPUTS:
2608 8380      ;
2608 8381      ;     ARGUMENT LIST:
2608 8382      ;
2608 8383      ;           OFFSET 0 - POINTER OFFSET LOW
2608 8384      ;           OFFSET 1 - POINTER OFFSET HIGH
2608 8385      ;--
2608 8386      PRINT_S:
2608 8387      LHL      ARG_LIST      ; GET OFFSET TO STRING
2608 8388      XCHG      ; PUT IN D REG
2608 8389      LHL      INIT_TEST_PC  ; GET OFFSET TO TEST DATA
2608 8390      DAD      D              ; GET ADDRESS OF STRING
2610 8391      TYPE      ; TYPE THE STRING
2616 8392      RET              ; EXIT
2617 8393
```

```
2617 8395 .SBTTL PRINT NUMERIC ROUTINE
2617 8396 :++
2617 8397 :
2617 8398 : FUNCTIONAL DESCRIPTION:
2617 8399 :
2617 8400 : THIS ROUTINE IS USED TO CONVERT A BINARY NUMBER TO A HEXIDECIMAL
2617 8401 : ASCII STRING AND TYPE IT.
2617 8402 :
2617 8403 : CALLING SEQUENCE:
2617 8404 :
2617 8405 : PRINT_N REG ADDRESS,DATA_TYPE
2617 8406 :
2617 8407 : IMPLICIT INPUTS:
2617 8408 :
2617 8409 : ARGUMENT LIST:
2617 8410 :
2617 8411 : OFFSET 0 - REGISTER ADDRESS LOW
2617 8412 : OFFSET 1 - REGISTER ADDRESS HIGH
2617 8413 : OFFSET 2 - DATA TYPE
2617 8414 :--
2617 8415 PRINT_N:
2617 8416 LHLD ARG_LIST ; GET THE REGISTER ADDRESS
261A 8417 LDA ARG_LIST+2 ; GET THE DATA TYPE
261D 8418 CPI 1 ; BYTE?
261F 8419 JNZ 10$ ; BRANCH IF NO
2622 8420 TYPEB ; PRINT THE DATA
262A 8421 JMP PRINT_N_EX ; EXIT
262D 8422 10$: CPI 2 ; WORD?
262F 8423 JNZ 20$ ; BRANCH IF NO
2632 8424 TYPEW ; PRINT THE DATA
263A 8425 JMP PRINT_N_EX ; EXIT
263D 8426 20$: TYPEL ; TYPE THE DATA
2645 8427 PRINT_N_EX:
2645 8428 RET ; EXIT
```

```

2646 8430          .SBTTL  DUMP ERROR LOG ROUTINE
2646 8431      ;++
2646 8432      ;
2646 8433      ; FUNCTIONAL DESCRIPTION:
2646 8434      ;
2646 8435      ;     THIS ROUTINE IS USED TO TYPE THE NUMBER OF ERRORS IN THE
2646 8436      ;     ERROR LOG.
2646 8437      ;--
2646 8438  DUMP_LOG:
2646 8439          LDA          SOFT_CTRL_REGB ; GET ENABLE FLAG
2649 8440          ANI          SBE_FLAG      ; IS FLAG SET?
264B 8441          JZ          DUMP_RET      ; BRANCH IF NO
264E 8442          TYPE        CRLF
2657 8443          LXI          H,ERROR_LOG_ADR ; GET ADDRESS OF ERROR LOG COUNT B6FC(X)
265A 8444          LDA          SOFT_CTRL_REGB ; SEE IF ERROR LOG WAS INITIALIZED
265D 8445          ANI          ERR_LOG_INIT ; ...
265F 8446          JZ          2$          ; BRANCH IF IT WAS INITIALIZED
2662 8447          XRA          A          ; INIT IT
2663 8448          MOV          M,A        ; ...
2664 8449  2$:    MOV          A,M        ; GET NUMBER OF ERRORS
2665 8450          CPI          64        ; IS IT FULL?
2667 8451          JNZ          1$        ; BRANCH IF NO
266A 8452          TYPE        LOG_MSG_3  ; TYPE ERROR LOG FULL MESSAGE
2673 8453          JMP          DUMP_EXIT  ; EXIT
2676 8454  1$:    PUSH        H          ; SAVE ADDRESS OF LOG
2677 8455          TYPE        LOG_MSG_2  ; TYPE THE MESSAGE
2680 8456          POP         H          ; GET ADDRESS OF LOG
2681 8457          TYPEB       ; TYPE NUMBER OF ERRORS (H&L CONTAIN
2689 8458          ; ADDRESS)
2689 8459  DUMP_EXIT:
2689 8460          TYPE        CRLF
2692 8461  DUMP_RET:
2692 8462          LDA          SOFT_CTRL_REGB ; SET THE ERROR LOG INIT FLAG
2695 8463          ORI          ERR_LOG_INIT ; ...
2697 8464          STA          SOFT_CTRL_REGB ; ...
269A 8465          RET
269B 8466

```

```
269B 8468 .SBTTL INTERPRETER SUBROUTINES
269B 8469 .SBTTL INITIALIZE LOOP VALUE ROUTINE
269B 8470 ;++
269B 8471 ;
269B 8472 ; FUNCTIONAL DESCRIPTION:
269B 8473 ;
269B 8474 ; THIS ROUTINE SETS THE CURRENT VALUE OF ALL THE LOOP NAMES TO
269B 8475 ; MINUS 1.
269B 8476 ;
269B 8477 ; CALLING SEQUENCE:
269B 8478 ;
269B 8479 ; CALL INIT_LOOP_VALUE
269B 8480 ;
269B 8481 ; THIS ROUTINE IS CALLED BY:
269B 8482 ;
269B 8483 ; NEW_TEST
269B 8484 ; SUB_TEST
269B 8485 ;
269B 8486 ; IMPLICIT INPUTS:
269B 8487 ;
269B 8488 ; NONE
269B 8489 ;
269B 8490 ; IMPLICIT OUTPUTS:
269B 8491 ;
269B 8492 ; THE CURRENT VALUE OF ALL THE LOOP NAMES IS INITIALIZED
269B 8493 ;
269B 8494 ;--
269B 8495
269B 8496 INIT_LOOP_VALUE:
269B 8497 LXI H,I_INDEX_TBL ; GET ADDRESS OF I INDEX TABLE
269E 8498 MVI A,-1 ; GET INITIAL DATA
26A0 8499 MVI C,6 ; SETUP TO INCREMENT H & L
26A2 8500 MVI B,0 ;
26A4 8501 MVI D,3 ; SET A LOOP COUNT
26A6 8502 10$: MOV M,A ; INIT I CURRENT VALUE
26A7 8503 INX H ;
26A8 8504 MOV M,A ;
26A9 8505 DAD B ; POINT AT J INDEX TABLE
26AA 8506 DCR D ; DONE ALL 3 INDEXES?
26AB 8507 JNZ 10$ ; BRANCH IF NO
26AE 8508 RET ; EXIT
26AF 8509
```

```
26AF 8511            .SBTTL    GET INDEX VALUE ROUTINE
26AF 8512            ;++
26AF 8513            ;
26AF 8514            ; FUNCTIONAL DESCRIPTION:
26AF 8515            ;
26AF 8516            ;        THIS ROUTINE GETS THE CURRENT VALUE OF A SPECIFIED INDEX NAME.
26AF 8517            ;
26AF 8518            ; CALLING SEQUENCE:
26AF 8519            ;
26AF 8520            ;        CALL    GET_INDEX_VALUE
26AF 8521            ;
26AF 8522            ; INPUT PARAMETERS:
26AF 8523            ;
26AF 8524            ;        REGISTER A - CONTAINS THE INDEX NAME VALUE (I=0, J=2, K=4)
26AF 8525            ;
26AF 8526            ; IMPLICIT INPUTS:
26AF 8527            ;
26AF 8528            ;        NONE
26AF 8529            ;
26AF 8530            ; OUTPUT PARAMETERS:
26AF 8531            ;
26AF 8532            ;        REGISTER B AND C - CONTAINS THE CURRENT VALUE OF THE SPECIFIED INDEX NAME
26AF 8533            ;        REGISTER PAIR H & L - CONTAIN ADDRESS OF SPECIFIED INDEX TABLE
26AF 8534            ;
26AF 8535            ; IMPLICIT OUTPUTS:
26AF 8536            ;
26AF 8537            ;        NONE
26AF 8538            ;
26AF 8539            ; COMPLETION CODES:
26AF 8540            ;
26AF 8541            ;        C BIT SET IF CURRENT VALUE OF INDEX IS MINUS 1 (INDICATES UNUSED).
26AF 8542            ;        OTHERWISE, C BIT CLEAR.
26AF 8543            ;
26AF 8544            ;--
26AF 8545            ;
26AF 8546            GET_INDEX_VALUE:
26AF 8547            ;
26AF 8548            ; SEE IF AN INDEX NAME WAS SPECIFIED
26AF 8549            ;
26AF 8550                   MVI            B,0            ; INITIALIZE THE INDEX VALUE
26B1 8551                   MOV            C,B            ;
26B2 8552                   ORA            A            ; SET THE CONDITION CODES
26B3 8553                   JM            5$            ; EXIT IF INDEX NAME NOT SPECIFIED
26B6 8554            ;
26B6 8555            ; SETUP TO INDEX THRU THE INDEX_NAM_TBL WITH THE 'A' REGISTER
26B6 8556            ;
26B6 8557                   LXI            H,INDEX_NAM_TBL ; GET ADDRESS OF THE TABLE
26B9 8558                   MOV            E,A            ; SETUP TO DO THE ADD
26BA 8559                   MVI            D,0            ;
26BC 8560                   DAD            D            ; INCREMENT H & L TO CORRECT PLACE
26BD 8561            ;
26BD 8562            ; NOW GET THE ADDRESS OF THE SPECIFIED INDEX
26BD 8563            ;
26BD 8564                   MOV            E,M            ; GET LOW PART OF ADDRESS
26BE 8565                   INX            H            ; POINT AT HI PART OF ADDRESS
```

```
26BF 8566      MOV          D,M          ; GET HIGH PART OF ADDRESS
26C0 8567      XCHG          ; PUT ADDRESS IN H & L
26C1 8568      ;
26C1 8569      ; REGISTER PAIR D & E NOW CONTAINS ADDRESS OF THE INDEX TABLE. GET THE
26C1 8570      ; CURRENT VALUE FROM THE TABLE AND RETURN
26C1 8571      ;
26C1 8572      MOV          C,M          ; GET THE CURRENT VALUE (LOW BYTE)
26C2 8573      INX          H          ; POINT AT HIGH BYTE
26C3 8574      MOV          B,M          ; GET HIGH BYTE
26C4 8575      DCX          H          ; POINT H & L AT CURRENT VALUE
26C5 8576      MOV          A,B          ; TEST FOR MINUS ONE
26C6 8577      CPI          -1          ;
26C8 8578      JNZ          10$        ; HIGH BYTE NOT MINUS ONE
26CB 8579      MOV          A,C          ;
26CC 8580      CPI          -1          ;
26CE 8581      JNZ          10$        ; LOW BYTE NOT MINUS ONE
26D1 8582 5$:  STC          ; SET THE C BIT
26D2 8583      RET          ; RETURN WITH C BIT SET
26D3 8584 10$: STC          ; RETURN WITH C BIT CLEAR
26D4 8585      CMC          ;
26D5 8586      RET          ; RETURN TO CALLER
26D6 8587
```

```

26D6 8589          .SBTTL      MULTIPLY A TIMES B&C ROUTINE
26D6 8590          ;++
26D6 8591          ;
26D6 8592          ; FUNCTIONAL DESCRIPTION:
26D6 8593          ;
26D6 8594          ;     THIS ROUTINE MULTIPLIES THE CONTENTS OF THE A REGISTER BY THE
26D6 8595          ;     CONTENTS OF THE B AND C REGISTERS WHERE B CONTAINS THE HIGH ORDER
26D6 8596          ;     BITS. THE 16 LEAST SIGNIFICANT BITS OF THE PRODUCT ARE RETURNED IN
26D6 8597          ;     THE D & E REGISTER PAIR. THE LOW ORDER BITS ARE IN THE E REGISTER.
26D6 8598          ;
26D6 8599          ; CALLING SEQUENCE:
26D6 8600          ;
26D6 8601          ;     CALL      MULTIPLY_A_BC
26D6 8602          ;
26D6 8603          ; INPUT PARAMETERS:
26D6 8604          ;
26D6 8605          ;     REGISTER A - CONTAINS THE MULTIPLIER
26D6 8606          ;     REGISTER PAIR B & C - CONTAINS THE MULTIPLICAND
26D6 8607          ;
26D6 8608          ; IMPLICIT INPUTS:
26D6 8609          ;
26D6 8610          ;     NONE
26D6 8611          ;
26D6 8612          ; OUTPUT PARAMETERS:
26D6 8613          ;
26D6 8614          ;     REGISTER PAIR D & E - CONTAIN THE 16 BIT PRODUCT
26D6 8615          ;
26D6 8616          ; IMPLICIT OUTPUTS:
26D6 8617          ;
26D6 8618          ;     NONE
26D6 8619          ;
26D6 8620          ;--
26D6 8621          ;
26D6 8622          MULTIPLY_A_BC:
26D6 8623          MVI          H,0          ; INITIALIZE THE PRODUCT
26D8 8624          MOV          L,0          ; ...
26D9 8625          ;
26D9 8626          ; START THE MULTIPLY
26D9 8627          ;
26D9 8628          1$:      DCR          A          ; DECREMENT THE MULTIPLIER
26DA 8629          JM          5$          ; BRANCH IF DONE WITH MULTIPLY
26DD 8630          DAD          B          ; ADD MULTIPLICAND TO P.P.
26DE 8631          JMP          1$          ; CONTINUE
26E1 8632          5$:      XCHG         ; MOVE RESULT TO D & E
26E2 8633          RET          ; EXIT
26E3 8634
  
```

```
26E3 8636      .SBTTL      SAVE BAD DATA ROUTINE
26E3 8637      ;++
26E3 8638      ;
26E3 8639      ; FUNCTIONAL DESCRIPTION:
26E3 8640      ;
26E3 8641      ;     THIS ROUTINE SAVES THE CONTENTS OF THE LOCATION(S) POINTED TO BY
26E3 8642      ;     THE H & L REGISTERS IN IMPLICIT STORAGE.
26E3 8643      ;
26E3 8644      ; CALLING SEQUENCE:
26E3 8645      ;
26E3 8646      ;     CALL      SAVE_BAD_DATA
26E3 8647      ;
26E3 8648      ; INPUT PARAMETERS:
26E3 8649      ;
26E3 8650      ;     REGISTER PAIR H & L - CONTAIN THE ADDRESS OF THE DATA TO SAVE
26E3 8651      ;     REGISTER A       - CONTAINS THE NUMBER OF BYTES TO SAVE
26E3 8652      ;
26E3 8653      ; IMPLICIT INPUTS:
26E3 8654      ;
26E3 8655      ;     NONE
26E3 8656      ;
26E3 8657      ; OUTPUT PARAMETERS:
26E3 8658      ;
26E3 8659      ;     NONE
26E3 8660      ;
26E3 8661      ; IMPLICIT OUTPUTS:
26E3 8662      ;
26E3 8663      ;     BAD_DATA - GETS LOADED WITH THE SPECIFIED DATA
26E3 8664      ;     DATA_TYPE- IS LOADED WITH THE BYTE COUNT
26E3 8665      ;
26E3 8666      ;--
26E3 8667      ;
26E3 8668      SAVE_BAD_DATA:
26E3 8669      STA      DATA_TYPE      ; SAVE THE DATA TYPE
26E6 8670      MOV      B,A           ; SAVE THE BYTE COUNT
26E7 8671      LXI      D,BAD_DATA    ; GET ADDRESS OF BAD DATA LOCATION
26EA 8672      ;
26EA 8673      ; FIRST CLEAR THE ENTIRE LONG WORD
26EA 8674      ;
26EA 8675      PUSH     H             ; SAVE H & L
26EB 8676      LXI      H,0          ; GET DATA TO CLEAR WITH
26EE 8677      SHLD    BAD_DATA      ; CLEAR LOW 16 BITS
26F1 8678      SHLD    BAD_DATA+2    ; AND UPPER 16 BITS
26F4 8679      POP      H           ; RESTORE H & L
26F5 8680 1$:  MOV      A,M         ; GET A BYTE OF THE DATA TO SAVE
26F6 8681      STAX    D           ; SAVE THE DATA
26F7 8682      INX     D           ; POINT TO THE NEXT BYTE
26F8 8683      INX     H           ;
26F9 8684      DCR     B           ; DECREMENT THE BYTE COUNT
26FA 8685      JNZ    1$          ; CONTINUE FOR THE SPECIFIED NO. OF BYTES
26FD 8686      RET
26FE 8687
```

```
26FE 8689 .SBTTL SAVE GOOD DATA ROUTINE
26FE 8690 ;**
26FE 8691 ;
26FE 8692 ; FUNCTIONAL DESCRIPTION:
26FE 8693 ;
26FE 8694 ; THIS ROUTINE SAVES THE CONTENTS OF THE LOCATION(S) POINTED TO BY
26FE 8695 ; THE H & L REGISTERS IN IMPLICIT STORAGE.
26FE 8696 ;
26FE 8697 ; CALLING SEQUENCE:
26FE 8698 ;
26FE 8699 ; CALL SAVE_GOOD_DATA
26FE 8700 ;
26FE 8701 ; INPUT PARAMETERS:
26FE 8702 ;
26FE 8703 ; REGISTER PAIR H & L - CONTAIN THE ADDRFS OF THE DATA TO SAVE
26FE 8704 ; REGISTER A - CONTAINS THE NUMBER OF BYTES TO SAVE
26FE 8705 ;
26FE 8706 ; IMPLICIT INPUTS:
26FE 8707 ;
26FE 8708 ; NONE
26FE 8709 ;
26FE 8710 ; OUTPUT PARAMETERS:
26FE 8711 ;
26FE 8712 ; NONE
26FE 8713 ;
26FE 8714 ; IMPLICIT OUTPUTS:
26FE 8715 ;
26FE 8716 ; GOOD_DATA - GETS LOADED WITH THE SPECIFIED DATA
26FE 8717 ; DATA_TYPE - IS LOADED WITH THE BYTE COUNT
26FE 8718 ;
26FE 8719 ;--
26FE 8720
26FE 8721 SAVE_GOOD_DATA:
26FE 8722 STA DATA_TYPE ; SAVE DATA TYPE
2701 8723 MOV B,A ; SAVE THE BYTE COUNT
2702 8724 LXI D,GOOD_DATA ; GET ADDRESS OF GOOD DATA LOCATION
2705 8725 ;
2705 8726 ; FIRST CLEAR THE ENTIRE LONG WORD
2705 8727 ;
2705 8728 PUSH H ; SAVE H & L
2706 8729 LXI H,0 ; GET DATA TO CLEAR WITH
2709 8730 SHLD GOOD_DATA ; CLEAR LOW 16 BITS
270C 8731 SHLD GOOD_DATA+2 ; AND UPPER 16 BITS
270F 8732 POP H ; RESTORE H & L
2710 8733 1$: MOV A,M ; GET A BYTE OF THE DATA TO SAVE
2711 8734 STAX D ; SAVE THE DATA
2712 8735 INX D ; POINT TO THE NEXT BYTE
2713 8736 INX H ;
2714 8737 DCR B ; DECREMENT THE BYTE COUNT
2715 8738 JNZ 1$ ; CONTINUE FOR THE SPECIFIED NO. OF BYTES
2718 8739 RET ; EXIT
2719 8740
```

```

2719 8742      .SBTTL      COMPARE GOOD AND BAD ROUTINE
2719 8743      ;++
2719 8744      ;
2719 8745      ; FUNCTIONAL DESCRIPTION:
2719 8746      ;
2719 8747      ;     THIS ROUTINE COMPARES THE CONTENTS OF GOOD_DATA AND BAD_DATA.
2719 8748      ;
2719 8749      ; CALLING SEQUENCE:
2719 8750      ;
2719 8751      ;     CALL      CMP_GOOD_BAD
2719 8752      ;
2719 8753      ; INPUT PARAMETERS:
2719 8754      ;
2719 8755      ;     REGISTER A - CONTAINS THE NUMBER OF BYTES TO COMPARE
2719 8756      ;     REGISTER B - CONTAINS THE MODE FLAG
2719 8757      ;             0 = COMPARE FOR EQUALITY
2719 8758      ;             FF = COMPARE FOR NOT EQUAL
2719 8759      ;
2719 8760      ; INPUT PARAMETERS:
2719 8761      ;
2719 8762      ;     GOOD_DATA - CONTAINS THE EXPECTED DATA
2719 8763      ;     BAD_DATA - CONTAINS THE RECIEVED DATA
2719 8764      ;
2719 8765      ; OUTPUT PARAMETERS:
2719 8766      ;
2719 8767      ;     NONE
2719 8768      ;
2719 8769      ; IMPLICIT OUTPUTS:
2719 8770      ;
2719 8771      ;     NONE
2719 8772      ;
2719 8773      ; COMPLETION CODES:
2719 8774      ;
2719 8775      ;     C BIT SET   - ALL (OR ONE) BYTE FAILED THE COMPARE
2719 8776      ;     C BIT CLEAR - ALL (OR ONE) BYTE PASSED THE COMPARE
2719 8777      ;
2719 8778      ;--
2719 8779      ;
2719 8780      CMP_GOOD_BAD:
2719 8781          MOV          C,A          ; SAVE THE BYTE COUNT
271A 8782          LXI          D,GOOD_DATA ; GET ADDRESS OF GOOD DATA
271D 8783          LXI          H,BAD_DATA  ; AND ADDRESS OF BAD DATA
2720 8784          MOV          A,B          ; WHAT IS THE MODE?
2721 8785          ORA          A          ;
2722 8786          JNZ          10$         ; BRANCH IF NOT EQUAL
2725 8787 1$:      LDAX          D          ; GET A BYTE OF GOOD DATA
2726 8788          CMP          M          ; COMPARE WITH BAD DATA
2727 8789          JNZ          5$          ; EXIT WITH C BIT SET
272A 8790          INX          D          ; POINT AT NEXT BYTE
272B 8791          INX          H          ;
272C 8792          DCR          C          ; DECREMENT THE BYTE COUNT
272D 8793          JNZ          1$         ; CONTINUE FOR BYTE CNT NO OF TIMES
2730 8794 15$:    STC          ; EXIT WITH C BIT CLEAR
2731 8795          CMC          ;
2732 8796          RET          ;

```

ZZ-ECKAA-8.7 V08.07
ECKAA
V08.07

D 2
COMPARE GOOD 11-FEB-1986
VAX-11/750 MICRO DIAGNOSTIC MONITOR
COMPARE GOOD AND BAD ROUTINE

Fiche 2 Frame D2 Sequence 222
11-FEB-1986 10:07:52 VAX/VMS Macro V04-00 Page 197
11-FEB-1986 10:06:37 [VAX750.MONITOR]ECKAA.MAR;811 (85)

```
2733 8797 5$:      STC                ; EXIT WITH C BIT SET
2734 8798          RET                ; ...
2735 8799          ;
2735 8800          ; MODE IS 'NOT EQUAL'
2735 8801          ;
2735 8802 10$:    LDAX                D      ; GET A BYTE OF GOOD DATA
2736 8803          CMP                M      ; COMPARE WITH BAD DATA
2737 8804          JNZ                15$   ; EXIT WITH C BIT CLEAR
273A 8805          INX                D      ; POINT AT NEXT BYTE
273B 8806          INX                H      ;
273C 8807          DCR                C      ; DECREMENT THE BYTE COUNT
273D 8808          JNZ                10$   ; CONTINUE FOR BYTE CNT NO. OF TIMES
2740 8809          JMP                5$    ; EXIT WITH C SET (ALL BYTES EQUAL)
2743 8810
```

```
2743 8812 .SBTTL RING BELL ROUTINE
2743 8813 ;++
2743 8814 ;
2743 8815 ; FUNCTIONAL DESCRIPTION:
2743 8816 ;
2743 8817 ; THIS ROUTINE TRANSMITS AN ASCII 7 (BELL) TO THE TERMINAL THE FIRST
2743 8818 ; TIME IT IS CALLED AND ON EVERY 5TH CALL THERE AFTER.
2743 8819 ;
2743 8820 ; CALLING SEQUENCE:
2743 8821 ;
2743 8822 ; CALL RING_BELL
2743 8823 ;
2743 8824 ; IMPLICIT INPUTS:
2743 8825 ;
2743 8826 ; BELL_COUNT - CONTAINS THE NUMBER OF TIMES THE ROUTINE HAS BEEN CALLED.
2743 8827 ;
2743 8828 ; IMPLICIT OUTPUTS:
2743 8829 ;
2743 8830 ; BELL_COUNT - IS INCREMENTED BY ONE OR INITIALIZED TO ONE.
2743 8831 ;
2743 8832 ;--
2743 8833 ;
2743 8834 RING_BELL:
2743 8835 LDA BELL_COUNT ; GET THE CALL COUNT
2746 8836 ORA A ; CHECK FOR ZERO
2747 8837 JZ 1$ ; BRANCH IF FIRST CALL
274A 8838 CPI 5 ; IS IT EQUAL TO 5 ,ET?
274C 8839 JNZ 10$ ; EXIT IF NO
274F 8840 XRA A ; CLEAR A
2750 8841 STA BELL_COUNT ; INITIALIZE THE BELL COUNT
2753 8842 ;
2753 8843 ; RING THE BELL
2753 8844 ;
2753 8845 1$: TYPE BELL ; RING THE BELL
275C 8846 ;
275C 8847 ; INCREMENT THE BELL COUNT AND EXIT
275C 8848 ;
275C 8849 10$: LDA BELL_COUNT
275F 8850 INR A ; INCREMENT
2760 8851 STA BELL_COUNT ; ...
2763 8852 RET ;
2764 8853
```

```
2764 8855 .SBTTL TYPE ERROR ROUTINE
2764 8856 ;**
2764 8857 ;
2764 8858 ; FUNCTIONAL DESCRIPTION:
2764 8859 ;
2764 8860 ; THIS ROUTINE IS USED TO TYPE AN ERROR MESSAGE. THE ERROR MESSAGE
2764 8861 ; HAS THE FOLLOWING FORMAT:
2764 8862 ;
2764 8863 ; ?ERROR: <ERROR_PC> TEST: <TEST_NUMBER> SUBTEST: <SUB_TEST_NUMB>
2764 8864 ;
2764 8865 ; DATA: XXXXXXXX [NOT]
2764 8866 ; YYYYYYYY
2764 8867 ;
2764 8868 ;
2764 8869 ;
2764 8870 ; FAILING GATE ARRAYS: AAA,BBB,CCC,...
2764 8871 ; FAILING MODULES: M8NXX
2764 8872 ;
2764 8873 ; THE DATA THAT GETS TYPED OUT AFTER 'DATA:' COMES FROM LOCATIONS
2764 8874 ; 'GOOD_DATA', 'BAD_DATA', CURRENT LOOP VALUES, AND INTERNAL CPU
2764 8875 ; SCRATCH PADS. THE NUMBER OF SCRATCH PADS TO TYPE IS CONTAINED IN
2764 8876 ; THE FOLLOWING TABLE. IF THE MODE TYPE OF THE LAST COMPARISON WAS
2764 8877 ; 'NOT EQUAL', THE WORD 'NOT' IS TYPED AFTER THE GOOD DATA.
2764 8878 ;
2764 8879 ; THE NAMES OF THE GATE ARRAYS AND MODULES TO TYPE MUST BE IN A TABLE
2764 8880 ; WITH THE FOLLOWING FORMAT:
2764 8881 ;
2764 8882 ; .BYTE NUMBER OF SCRATCH PADS
2764 8883 ; .BYTE NUMBER OF GATE ARRAYS
2764 8884 ; .BYTE GATE ARRAY CODE
2764 8885 ;
2764 8886 ;
2764 8887 ; .BYTE NUMBER OF MODULES
2764 8888 ; .BYTE MODULE CODE
2764 8889 ;
2764 8890 ;
2764 8891 ; CALLING SEQUENCE:
2764 8892 ;
2764 8893 ; CALL TYPE_ERROR
2764 8894 ;
2764 8895 ; INPUT PARAMETERS:
2764 8896 ;
2764 8897 ; REGISTER PAIR H & L - CONTAIN THE ADDRESS OF THE GATE ARRAY/MODULE LIST
2764 8898 ;
2764 8899 ; IMPLICIT INPUTS:
2764 8900 ;
2764 8901 ; CURRENT_PC - CONTAINS THE ADDRESS OF THE 'IFERROR' INSTRUCTION
2764 8902 ; TEST_NUMBER - CONTAINS THE CURRENT TEST NUMBER
2764 8903 ; SUB_TEST_NUMB - CONTAINS THE CURRENT SUBTEST NUMBER
2764 8904 ; GOOD_DATA - CONTAINS THE EXPECTED DATA
2764 8905 ; BAD_DATA - CONTAINS THE RECEIVED DATA
2764 8906 ; CURRENT LOOP VALUES
2764 8907 ;
2764 8908 ; IMPLICIT OUTPUTS:
2764 8909 ;
```

```

2764 8910 : NONE
2764 8911 :
2764 8912 :--
2764 8913
2764 8914 TYPE_ERROR:
2764 8915 PUSH H ; SAVE POINTER TO LIST
2765 8916 TYPE CRLF ; START A NEW LINE
276E 8917 TYPE ERROR_MSG ; TYPE '?ERROR: '
2777 8918 :
2777 8919 : GET THE ERROR TEST PC, SUBTRACT THE BASE ADDRESS, AND TYPE IT
2777 8920 :
2777 8921 LHL D C_INIT_TEST_PC ; GET TWO'S COMPLIMENT INIT TEST PC
277A 8922 XCHG
277B 8923 LHL D CURRENT_PC ; GET THE ERROR PC
277E 8924 DAD D ; SUBTRACT THE BASE ADDRESS OF TEST
277F 8925 XCHG ; ADD IN THE TITLE LENGTH
2780 8926 LHL D TITLE_LENGTH ; ...
2783 8927 DAD D ; ...
2784 8928 INX H ; ACCOUNT FOR LENGTH BYTE
2785 8929 SHLD TYPE_ERR_BUFFER ; PUT ERROR PC IN BUFFER
2788 8930 TYPEW TYPE_ERR_BUFFER ; TYPE THE ERROR PC
2793 8931 :
2793 8932 : NOW TYPE THE TEST NUMBER
2793 8933 :
2793 8934 TYPE TEST_MSG ; TYPE ' TEST: '
279C 8935 TYPEB TEST_NUMBER ; TYPE THE TEST NUMBER
27A7 8936 :
27A7 8937 : NOW TYPE THE SUB TEST NUMBER
27A7 8938 :
27A7 8939 TYPE SUB_TEST_MSG ; TYPE ' SUBTEST: '
27B0 8940 LDA SUB_TEST_NUMB ; IF SUB TEST NUMBER IS ZERO, MAKE
27B3 8941 ORA A ; IT EQUAL TO ONE
27B4 8942 JNZ 1$ ; NOT ZERO
27B7 8943 MVI A,1 ; ...
27B9 8944 STA SUB_TEST_NUMB ; ...
27BC 8945 1$: TYPEB SUB_TEST_NUMB ; TYPE THE SUB TEST NUMBER
27C7 8946 TYPE CRLF ; TERMINATE THE LINE
27D0 8947 TYPE CRLF ; SKIP A LINE
27D9 8948 :
27D9 8949 : NOW TYPE THE DATA
27D9 8950 :
27D9 8951 TYPE DATA_MSG ; TYPE 'DATA: '
27E2 8952 TYPEL GOOD_DATA ; TYPE THE EXPECTED DATA
27ED 8953 LDA MODE_TYPE ; WAS THIS AN 'EQUALS' CHECK?
27F0 8954 ORA A ; ...
27F1 8955 JZ 2$ ; BRANCH IF YES
27F4 8956 TYPE NOT_MSG ; TYPE 'NOT' AFTER GOOD DATA
27FD 8957 2$: TYPE CRLF
2806 8958 LXI H,BAD_DATA
2809 8959 CALL DISPLAY_DATA ; TYPE THE RECEIVED DATA
280C 8960 :
280C 8961 : NOW TYPE THE LOOP VALUES OF I, J, AND K IF THEY ARE BEING USED.
280C 8962 :
280C 8963 LXI H,0 ; CLEAR THE TEMP BUFFER
280F 8964 SHLD TYPE_ERR_BUFFER ; ...

```

```

2812 8965      SHLD      TYPE_ERR_BUFFER+2 ;
2815 8966      XRA       A ; GET THE I INDEX CODE IN A REGISTER
2816 8967      CALL      GET_INDEX_VALUE ; GET THE CURRENT VALUE OF I INDEX
2819 8968      JC        5$ ; BRANCH IF I NOT BEING USED
281C 8969      PUSH      B ; PUT VALUE IN H & L
281D 8970      POP       H ;
281E 8971      INX       H ; MAKE LOOP COUNT START AT ONE
281F 8972      SHLD      TYPE_ERR_BUFFER ; SAVE I INDEX VALUE
2822 8973      LXI       H,TYPE_ERR_BUFFER
2825 8974      CALL      DISPLAY_DATA ; TYPE THE I INDEX VALUE
2828 8975
2828 8976 5$:    MVI       A,2 ; GET THE J INDEX CODE
282A 8977      CALL      GET_INDEX_VALUE ; GET THE CURRENT VALUE
282D 8978      JC        10$ ; BRANCH IF J INDEX NOT USED
2830 8979      PUSH      B ; MOVE VALUE TO H & L
2831 8980      POP       H ;
2832 8981      INX       H ; MAKE VALUE START AT ONE
2833 8982      SHLD      TYPE_ERR_BUFFER ; SAVE THE CURRENT VALUE
2836 8983      LXI       H,TYPE_ERR_BUFFER
2839 8984      CALL      DISPLAY_DATA ; TYPE THE J INDEX VALUE
283C 8985
283C 8986 10$:   MVI       A,4 ; GET THE K INDEX CODE
283E 8987      CALL      GET_INDEX_VALUE ; GET THE CURRENT VALUE
2841 8988      JC        15$ ; BRANCH IF NOT USED
2844 8989      PUSH      B ; PUT VALUE IN H & L
2845 8990      POP       H ;
2846 8991      INX       H ; MAKE VALUE START AT ONE
2847 8992      SHLD      TYPE_ERR_BUFFER ; SAVE CURRENT J INDEX VALUE
284A 8993      LXI       H,TYPE_ERR_BUFFER
284D 8994      CALL      DISPLAY_DATA ; TYPE THE K INDEX VALUE
2850 8995 ;
2850 8996 ; NOW TYPE THE SCRATCH PAD DATA IF THERE IS ANY
2850 8997 ;
2850 8998 15$:   POP       H ; GET POINTER TO LIST
2851 8999      MOV       A,M ; GET THE NUMBER OF SCRATCH PADS TO TYPE
2852 9000      ORA       A ; SET THE CONDITION CODES
2853 9001      JZ        30$ ; BRANCH IF NO SCRATCH PADS TO TYPE
2856 9002      PUSH      $PSW ; SAE THE COUNT
2857 9003      XRA       A ; INIT THE REGISTER NUMBER
2858 9004      STA       TYPE_ERR_TEMP ;
285B 9005 20$:   PUSH      H ; SAVE POINTER
285C 9006      LDA       TYPE_ERR_TEMP ; GET REGISTER NUMBER
285F 9007      CALL      READ_RTEMP ; GET CONTENTS OF REGISTER
2862 9008 ;
2862 9009 ; NOW TYPE THE CONTENTS OF THE WH REGISTER
2862 9010 ;
2862 9011      LXI       H,WH_REG_BYTE_0 ; GET ADDRESS OF REGISTER
2865 9012      CALL      DISPLAY_DATA ; TYPE THE DATA
2868 9013 ;
2868 9014 ; CHECK IF MORE DATA TO TYPE
2868 9015 ;
2868 9016      POP       H ; RESTORE H & L
2869 9017      LDA       TYPE_ERR_TEMP ; INCREMENT THE REGISTER NUMBER
286C 9018      INR       A ;
286D 9019      STA       TYPE_ERR_TEMP ;

```

```
2870 9020      POP      $PSW      ; GET THE WORD COUNT
2871 9021      DCR      A          ; DONE YET?
2872 9022      PUSH     $PSW      ; SAVE WORD COUNT
2873 9023      JNZ     20$      ; BRANCH IF NO
2876 9024      POP      $PSW      ; CLEANUP THE STACK
2877 9025      ;
2877 9026      ; SEE IF THIS WAS A VBUS COMPARE THAT FAILED
2877 9027      ;
2877 9028 30$:  LDA      SOFT_CTRL_REGA ; GET THE VBUS COMPARE FLAG
287A 9029      ANI      VBUS_COMPARE ; IS IT SET?
287C 9030      JZ       40$      ; BRANCH IF NO
287F 9031      ;
287F 9032      ; TYPE THE REMAINING VBUS BITS THAT WERE NOT CHECKED
287F 9033      ;
287F 9034      PUSH     H          ; SAVE THE H & L REGISTERS
2880 9035      LHLD    VBUS_TBL_ADDR ; GET THE ADDRESS OF THE VBUS TABLE
2883 9036      LDA     VBUS_TBL_COUNT ; AND THE REMAINING BITS
2886 9037      MOV     B,A          ;
2887 9038      DCR     B          ; MORE BITS TO TYPE?
2888 9039      JM      39$      ; BRANCH IF NO
288B 9040      PUSH     H          ;
288C 9041      PUSH     B          ;
288D 9042      TYPE    VBUS_MSG_2   ; TYPE THE VBUS MESSAGE
2896 9043      TYPE    CRLF        ;
289F 9044      POP     B          ;
28A0 9045      POP     H          ;
28A1 9046 33$: CALL    CHECK_VBUS   ; GET VBUS DATA INTO 'BAD_DATA'
28A4 9047      PUSH     H          ; SAVE H & L
28A5 9048      PUSH     B          ; AND B
28A6 9049      LXI    H,BAD_DATA   ;
28A9 9050      CALL    DISPLAY_DATA ; TYPE THE VBUS BIT AND VALUE
28AC 9051      POP     B          ;
28AD 9052      POP     H          ;
28AE 9053      INX    H          ; POINT AT NEXT VBUS ENTRY
28AF 9054      DCR     B          ; MORE TO TYPE?
28B0 9055      JP      33$      ; BRANCH IF YES
28B3 9056 39$: POP     H          ; RESTORE H & L
28B4 9057      ;
28B4 9058      ;
28B4 9059      ; NOW TYPE THE FAILING GATE ARRAY LIST
28B4 9060      ;
28B4 9061 40$: INX    H          ; POINT AT THE GATE ARRAY COUNT
28B5 9062      MOV     A,M        ; GET NUMBER OF GATE ARRAYS
28B6 9063      ORA    A          ; SET THE CODITION CODES
28B7 9064      CNZ    TYPE_GA_LIST ; DON'T CALL IF NO GATE ARRAYS IN CALLOUT
28BA 9065      ;
28BA 9066      ;
28BA 9067      ; NOW SEE IF THERE IS A MODULE LIST TO TYPE
28BA 9068      ;
28BA 9069 50$: INX    H          ; POINT AT THE MODULE COUNT
28BB 9070      MOV     A,M        ; GET THE COUNT
28BC 9071      ORA    A          ; SET THE CONDITION CODES
28BD 9072      JZ     60$      ; BRANCH IF NO MODULES TO TYPE
28C0 9073      PUSH     $PSW      ; SAVE THE COUNT
28C1 9074      INX    H          ; POINT AT THE MODULE CODE
```

```
28C2 9075      PUSH      H           ; SAVE THE POINTER
28C3 9076      TYPE      MODULE_MSG ; TYPE 'FAILING MODULES: '
28CC 9077      POP       H           ; GET THE POINTER
28CD 9078 51$:  MOV       A,M        ; GET THE MODULE CODE
28CE 9079      PUSH      H           ; SAVE THE POINTER
28CF 9080      RLC       H           ; MAKE CODE A WORD INDEX
28D0 9081      LXI      H,MODULE_NAM_TBL ; GET ADDRESS OF MODULE NAME TABLE
28D3 9082      MOV       C,A        ; SETUP TO INDEX THRU THE TABLE
28D4 9083      MVI      B,0         ; ...
28D6 9084      DAD      B           ; INDEX THE LIST ADDRESS
28D7 9085      MOV       E,M        ; GET ADDRESS OF ASCII STRING
28D8 9086      INX      H           ; ...
28D9 9087      MOV       D,M        ; ...
28DA 9088      XCHG      H           ; ...
28DB 9089      TYPE      CRLF       ; TYPE THE MODULE NAME
28E1 9090      TYPE      FAIL_MOD_INDEX ; TERMINATE THE LINE
28EA 9091      TYPE      H           ; SET CURSOR FOR NEXT MODULE PRINTOUT
28F3 9092      POP       H           ; GET LIST POINTER
28F4 9093      POP       $PSW       ; GET LOOP COUNT
28F5 9094      DCR      A           ; DECREMENT
28F6 9095      PUSH     H           ; SAVE
28F7 9096      INX      H           ; POINT AT NEXT MODULE CODE
28F8 9097      JNZ     51$         ; TYPE NEXT MODULE NAME
28FB 9098      POP       $PSW       ; THROW AWAY SAVED LOOP COUNT
28FC 9099      TYPE      CRLF       ; TERMINATE THE LAST LINE
2905 9100      :
2905 9101      : DONE
2905 9102      :
2905 9103 60$:  RET
2906 9104
```

```

2906 9106      .SBTTL      SINGLE PSEUDO INSTRUCTION ROUTINE
2906 9107      ;**
2906 9108      ;
2906 9109      ; FUNCTIONAL DESCRIPTION:
2906 9110      ;
2906 9111      ;     THIS ROUTINE IS USED TO STEP BY SINGLE PSEUDO INSTRUCTIONS. IT IS
2906 9112      ;     CALLED BY THE OP CODE DISPATCH ROUTINE. THE ROUTINE TYPES THE
2906 9113      ;     CURRENT TEST PC AND WAITS FOR OPERATOR INPUT. IF A 'SPACE' IS TYPED
2906 9114      ;     THE ROUTINE EXITS WITH THE C BIT CLEAR. IF ANY OTHER CHARACTER
2906 9115      ;     IS TYPED, THE SINGLE INSTRUCTION FLAG IS CLEARED AND RETURN IS MADE
2906 9116      ;     WITH THE C BIT SET.
2906 9117      ;
2906 9118      ; CALLING SEQUENCE:
2906 9119      ;
2906 9120      ;     CALL      SINGLE_INSTR
2906 9121      ;
2906 9122      ; IMPLICIT INPUTS:
2906 9123      ;
2906 9124      ;     TEST_PC - CONTAINS THE CURRENT TEST PC
2906 9125      ;
2906 9126      ; OUTPUT PARAMETERS:
2906 9127      ;
2906 9128      ;     NONE
2906 9129      ;
2906 9130      ; COMPLETION CODES:
2906 9131      ;
2906 9132      ;     C BIT SET   - A SPACE WAS NOT TYPED
2906 9133      ;     C BIT CLEAR - A SPACE WAS TYPED
2906 9134      ;
2906 9135      ;--
2906 9136      ;
2906 9137      SINGLE_INSTR:
2906 9138      $TERM_INIT          ; ABORT ANY INPUT REQUESTS
290B 9139      TYPE              CRLF          ; START A NEW LINE
2914 9140      1$: TYPE          TPC_MSG      ; TYPE '   TPC= '
291D 9141      LHL D             TEST_PC      ; GET THE TEST PC
2920 9142      XCHG
2921 9143      LHL D             C_INIT_TEST_PC ; GET TWO'S COMPLIMENT INIT TEST PC
2924 9144      D                D            ; SUBTRACT FROM TEST PC
2925 9145      INX              H            ; ACCOUNT FOR LENGTH BYTE
2926 9146      XCHG
2927 9147      LHL D             TITLE_LENGTH
292A 9148      D
292B 9149      SHLD             TYPE_ERR_BUFFER ; SAVE THE TEST PC
292E 9150      TYPEW           TYPE_ERR_BUFFER ; TYPE THE TEST PC
2939 9151      $TERM_READ      TERM_INP_BUFF,ONE ; GET ONE CHARACTER FROM TERMINAL
2944 9152      LDA              TERM_INP_BUFF+1 ; GET THE CHARACTER THAT WAS TYPED
2947 9153      CPI              <^X20>      ; WAS IT A SPACE?
2949 9154      JZ                S$          ; BRANCH IF YES
294C 9155      LDA              SOFT_CTRL_REGA ; GET SINGLE INSTR FLAG
294F 9156      ANI              <^CINSTR_FLAG> ; CLEAR THE FLAG
2951 9157      STA              SOFT_CTRL_REGA ; ...
2954 9158      STC
2955 9159      JMP              10$          ; RETURN WITH C BIT SET
2958 9160      ;

```

ZZ-ECKAA-8.7 V08.07
ECKAA
V08.07

L 2

SINGLE PSEUDO11-FEB-1986 Fiche 2 Frame L2 Sequence 230

VAX-11/750 MICRO DIAGNOSTIC MONITOR 11-FEB-1986 10:07:52 VAX/VMS Macro V04-00 Page 205

SINGLE PSEUDO INSTRUCTION ROUTINE 11-FEB-1986 10:06:37 [VAX750.MONITOR]ECKAA.MAR;611 (88)

```
2958 9161 ; EXIT
2958 9162 ;
2958 9163 5$:
2958 9164 $TERM_READ      TERM_INP_BUFF,- ; ISSUE AN INTERRUPT DRIVEN READ
2958 9165                    ONE,TERM_INTERRUPT; ...
2963 9166                    ; RETURN WITH C BIT CLEAR
2964 9167                    ; EXIT
2965 9168 10$:
2966 9169
```

```

2966 9171      .SBTTL      INSERT FIELD ROUTINE
2966 9172      :++
2966 9173      :
2966 9174      : FUNCTIONAL DESCRIPTION:
2966 9175      :
2966 9176      :           THIS ROUTINE IS USED TO INSERT A DATA ELEMENT INTO A FIELD OF A
2966 9177      :           MICRO WORD. IT USES THE TABLE 'DCS_BIT_MAP' TO SCRAMBLE THE BITS
2966 9178      :           IN THE MICRO WORD. AFTER THE FIELD IS INSERTED, THE 'CALC_PARITY'
2966 9179      :           ROUTINE IS CALLED TO GENERATE THE NEW PARITY BITS.
2966 9180      :
2966 9181      : CALLINE SEQUENCE:
2966 9182      :
2966 9183      :           CALL      INSERT_FIELD
2966 9184      :
2966 9185      : INPUT PARAMETERS:
2966 9186      :
2966 9187      :           REGISTER PAIR H & L - CONTAINS ADDRESS OF THE DATA TO INSERT
2966 9188      :           REGISTER PAIR D & E - CONTAINS ADDRESS OF BIT 0 OF THE ELEMENT
2966 9189      :                               TO INSER THE FIELD IN.
2966 9190      :           REGISTER B           - CONTAINS THE STARTING BIT NUMBER OF THE FIELD
2966 9191      :           REGISTER C           - CONTAINS THE ENDING BIT NUMBER OF THE FIELD
2966 9192      :           REGISTER A           - CONTAINS THE INHIBIT PARITY FLAG. NON ZERO
2966 9193      :                               MEANS DON'T SET THE PARITY.
2966 9194      :
2966 9195      : IMPLICIT INPUTS:
2966 9196      :
2966 9197      :           (SP)      - INHIBIT SCRAMBLE FLAG
2966 9198      :
2966 9199      : OUTPUT PARAMETERS:
2966 9200      :
2966 9201      :           NONE
2966 9202      :
2966 9203      : IMPLICIT OUTPUTS:
2966 9204      :
2966 9205      :           NONE
2966 9206      :
2966 9207      :--
2966 9208
2966 9209      INSERT_FIELD:
2966 9210      SHLD      INS_FLD_BUFF      ; SAVE H & L
2969 9211      STA      CURRENT_BIT      ; SAVE A REGISTER
296C 9212      POP      H              ; GET RETURN PC
296D 9213      POP      $PSW          ; GET SCRAMBLE FLAG
296E 9214      STA      SCRAMBLE_FLAG ; SAVE
2971 9215      PUSH     H              ; RESTORE THE RETURN PC
2972 9216      LHLD     INS_FLD_BUFF      ; RESTORE H & L
2975 9217      LDA      CURRENT_BIT      ; RESTORE THE A REGISTER
2978 9218      PUSH     $PSW          ; SAVE THE A REGISTER
2979 9219      PUSH     D              ; SAVE THE DESTINATION ADDRESS
297A 9220      PUSH     H              ; SAVE THE SOURCE DATA ADDRESS
297B 9221      PUSH     B              ; SAVE STARTING AND ENDING BIT NUMBERS
297C 9222      XRA      A              ; INIT THE CURRENT BIT NUMBER
297D 9223      STA      CURRENT_BIT      ;
2980 9224      LXI      H,INS_FLD_BUFF ; GET ADDRESS OF TEMP BUFFER
2983 9225      MVI      B,10             ; SET A LOOP COUNT FOR 10 BYTES

```

```
2985 9226 1$: LDAX D ; GET A BYTE OF THE MICRO WORD
2986 9227 MOV M,A ; PUT IN TEMP BUFFER
2987 9228 INX H ; ...
2988 9229 INX D ; ...
2989 9230 DCR B ; DECREMENT BYTE COUNT
298A 9231 JNZ 1$ ; CONTINUE FOR 10 BYTES
298D 9232 ;
298D 9233 ; CALCULATE THE NUMBER OF BITS IN THE FIELD
298D 9234 ;
298D 9235 POP B ; GET THE STARTING AND ENDING BIT #'S
298E 9236 MOV A,C ; SUBTRACT STARTING FROM THE
298F 9237 SUB B ; ENDING BIT NUMBER
2990 9238 INR A ; INCREMENT BY ONE
2991 9239 STA BIT_COUNT ;SAVE
2994 9240 ;
2994 9241 ; B = UNSCRAMBLED BIT NUMBER
2994 9242 ; GET THE SCRAMBLED BIT NUMBER
2994 9243 ;
2994 9244 2$: LDA SCRAMBLE_FLAG ; GET THE SCRAMBLE FLAG
2997 9245 ORA A ; IS IT SET?
2998 9246 JP 4$ ; BRANCH IF NO
299B 9247 MOV A,B ; PUT UNSCRAMBLED NUMBER IN A
299C 9248 JMP 6$ ; CONTINUE BELOW
299F 9249 4$: LXI H,DCS_BIT_MAP ; GET ADDRESS OF THE BIT MAP
29A2 9250 MOV E,B ; INCREMENT BY THE UNSCRAMBLED BIT #
29A3 9251 MVI D,0 ; ...
29A5 9252 DAD D ; ...
29A6 9253 MOV A,M ; GET THE SCRAMBLED BIT NUMBER
29A7 9254 ;
29A7 9255 ; A = SCRAMBLED BIT NUMBER
29A7 9256 ; B = UNSCRAMBLED BIT NUMBER
29A7 9257 ; CALCULATE THE BYTE AND BIT NUMBER OF THE MICRO WORD
29A7 9258 ;
29A7 9259 6$: LXI D,0 ; INIT D & E
29AA 9260 3$: SUI 8 ; DIVIDE SCRAMBLED BIT NUMBER BY 8
29AC 9261 JM 5$ ; BRANCH IF DIVIDE DONE
29AF 9262 INR E ; INCREMENT THE QUOTIENT
29B0 9263 JMP 3$ ; CONTINUE THE DIVIDE
29B3 9264 5$: ADI 8 ; GET THE REMAINDER
29B5 9265 ;
29B5 9266 ; A = BIT NUMBER WITHIN THE BYTE OF THE MICRO WORD
29B5 9267 ; B = UNSCRAMBLED BIT NUMBER
29B5 9268 ; D = 0
29B5 9269 ; E = BYTE NUMBER WITHIN THE MICRO WORD
29B5 9270 ; GET THE BYTE ADDRESS AND BIT MASK FOR THE MICRO WORD
29B5 9271 ;
29B5 9272 LXI H,INS_FLD_BUFF ; GET ADDRESS OF BUFFER
29B8 9273 DAD D ; GET ADDRESS OF BYTE TO MODIFY
29B9 9274 CALL BIT_NMB_TO_MSK ; GET BIT MASK
29BC 9275 ;
29BC 9276 ; A = MACRO WORD BIT MASK
29BC 9277 ; B = UNSCRAMBLED BIT NUMBER
29BC 9278 ; H & L = MICRO WORD BYTE ADDRESS
29BC 9279 ; NOW GET THE BYTE ADDRESS AND BIT MASK FOR THE SOURCE DATA
29BC 9280 ;
```

```

29BC 9281      POP      D      ; GET THE SOURCE ADDRESS
29BD 9282      PUSH     D      ; SAVE IT AGAIN
29BE 9283      PUSH     H      ; SAVE MICRO WORD ADDRESS
29BF 9284      XCHG
29C0 9285      PUSH     $PSW   ; SAVE THE UNSCRAMBLD BIT NUMBER
29C1 9286      LDA      CURRENT_BIT ; GET THE CURRENT BIT NUMBER
29C4 9287      LXI      D,0     ; INIT D & E
29C7 9288 15$: SUI      8      ; DIVIDE THE CURRENT BIT NUMBER BY 8
29C9 9289      JM       20$    ; BRANCH IF DIVIDE DONE
29CC 9290      INR      E      ; INCREMENT THE QUOTIENT
29CD 9291      JMP      15$    ; CONTINUE THE DIVIDE
29D0 9292 20$: ADI      8      ; BACKUP THE REMAINDER
29D2 9293      CALL    BIT_NMB_TO_MSK ; GET A MASK FOR THE BIT NUMBER
29D5 9294      MOV      C,A    ; PUT MASK IN C REG
29D6 9295      ;
29D6 9296      ; C = SOURCE DATA BIT MASK
29D6 9297      ; D = 0
29D6 9298      ; E = SOURCE DATA BYTE NUMBER
29D6 9299      ; H & L = SOURCE DATA ADDRESS
29D6 9300      ;
29D6 9301      DAD      D      ; GET BYTE ADDRESS OF SOURCE DATA
29D7 9302      POP      $PSW  ; GET MICRO WORD BIT MASK
29D8 9303      POP      D      ; GET MICRO WORD BYTE ADDRESS
29D9 9304      ;
29D9 9305      ; NO INSERT THE BIT FROM THE SOURCE INTO THE MICRO WORD
29D9 9306      ;
29D9 9307      PUSH     B      ; SAVE B & C
29DA 9308      MOV      B,A    ; SAVE MICRO WORD BIT MASK
29DB 9309      LDAX   D      ; GET THE BYTE OF MICRO WORD DATA
29DC 9310      ORA      B      ; SET THE SPECIFIED BIT
29DD 9311      PUSH     B      ; SAVE THE BIT MASK
29DE 9312      MOV      B,A    ; PUT THE DATA IN B
29DF 9313      MOV      A,M    ; GET THE BYTE OF SOURCE DATA
29E0 9314      ANA      C      ; SEE IF BIT SHOULD BE SET
29E1 9315      MOV      A,B    ; PUT MICRO WORD BYTE BACK INTO A
29E2 9316      POP      B      ; RESTORE B & C
29E3 9317      JNZ     25$    ; BRANCH IF BIT SHOULD BE SET
29E6 9318      XRA      B      ; CLEAR THE BIT
29E7 9319 25$: POP      B      ; RESTORE THE UNSCRAMBLD BIT NUMBER
29E8 9320      STAX   D      ; PUT THE BYTE IN THE MICRO WORD
29E9 9321      ;
29E9 9322      ; NOW CHECK IF THERE ARE MORE BITS TO INSERT
29E9 9323      ;
29E9 9324      LDA      CURRENT_BIT ; INCREMENT TH CURRENT BIT NUMBER
29EC 9325      INR      A      ; ...
29ED 9326      STA      CURRENT_BIT ; ...
29F0 9327      INR      B      ; INCREMENT THE UNSCRAMBLD BIT #
29F1 9328      LDA      BIT_COUNT ; DECREMENT THE BIT COUNT
29F4 9329      DCR      A      ; ...
29F5 9330      STA      BIT_COUNT ; ...
29F8 9331      JNZ     2$      ; BRANCH IF MORE BITS TO INSERT
29FB 9332      ;
29FB 9333      ; SETUP THE REGISTERS TO MOVE THE MICRO WORD TO ITS ORIGINAL DESTINATION.
29FB 9334      ;
29FB 9335      POP      D      ; CLEANUP THE STACK

```

```
29FC 9336      POP      D      ; GET THE DESTINATION ADDRESS
29FD 9337      LXI      H,INS_FLD_BUFF ; GET ADDRESS OF THE MICRO WORD
2A00 9338      MVI      C,10      ; SET THE BYTE COUNT
2A02 9339      ;
2A02 9340      ; NOW MOVE THE TEMP BUFFER TO THE DESTINATION
2A02 9341      ;
2A02 9342      PUSH     D      ; SAVE ADDRESS OF MICRO WORD
2A03 9343 30$:  MOV     A,M      ; GET A BYTE FROM TEMP BUFFER
2A04 9344      STAX     D      ; PUT IN DESTINATION
2A05 9345      INX     D      ; ...
2A06 9346      INX     H      ; ...
2A07 9347      DCR     C      ; DECREMENT THE BYTE COUNT
2A08 9348      JNZ     30$    ; CONTINUE FOR 10 BYTES
2A0B 9349      POP     H      ; GET MICRO WORD ADDRESS
2A0C 9350      POP     $PSW   ; GET THE INHIBIT PARITY FLAG
2A0D 9351      ORA     A      ; SET THE CONDITION CODES
2A0E 9352      JNZ     INSERT_FIELD_X ; BRANCH IF INHIBIT
2A11 9353      LDA     SCRAMBLE_FLAG ; GET THE SCRAMBLE FLAG
2A14 9354      ORA     A      ; IS IT SET?
2A15 9355      JM      INSERT_FIELD_X ; BRANCH IF YES
2A18 9356      CALL    CALC_PARITY ; CALCULATE THE NEW PARITY BITS
2A1B 9357  INSERT_FIELD_X:
2A1B 9358      RET
2A1C 9359
```

```

2A1C 9361      .SBTTL      CALCULATE PARITY ROUTINE
2A1C 9362      ;++
2A1C 9363      ;
2A1C 9364      ; FUNCTIONAL DESCRIPTION:
2A1C 9365      ;
2A1C 9366      ;     THIS ROUTINE CALCULATES AND LOADS THE P0 AND P1 PARITY BITS OF
2A1C 9367      ;     A MICRO WORD. A BIT MASK IS USED TO DETERMINE THE PARITY FOR THE
2A1C 9368      ;     CORRESPONDING PARITY BITS. P0 IS SET TO MAKE EVEN PARITY AND P1
2A1C 9369      ;     IS SET TO MAKE ODD PARITY.
2A1C 9370      ;
2A1C 9371      ; CALLING SEQUENCE:
2A1C 9372      ;
2A1C 9373      ;     CALL      CALC_PARITY
2A1C 9374      ;
2A1C 9375      ; INPUT PARAMETERS:
2A1C 9376      ;
2A1C 9377      ;     REGISTERS H & L - CONTAIN THE ADDRESS OF THE MICRO WORD TO CALCULATE
2A1C 9378      ;     THE PARITY BITS ON.
2A1C 9379      ;
2A1C 9380      ; IMPLICIT INPUTS:
2A1C 9381      ;
2A1C 9382      ;     P0_PARITY_MSK - CONTAINS THE 10 BYTE BIT MAP OF THE P0 PARITY BIT
2A1C 9383      ;     P1_PARITY_MSK - CONTAINS THE 10 BYTE BIT MAP OF THE P1 PARITY BIT
2A1C 9384      ;
2A1C 9385      ; IMPLICIT OUTPUTS:
2A1C 9386      ;
2A1C 9387      ;     THE MICRO WORD'S PARITY BITS ARE SET TO THE CORRECT VALUE.
2A1C 9388      ;
2A1C 9389      ;--
2A1C 9390      ;
2A1C 9391      CALC_PARITY:
2A1C 9392      PUSH          H                ; SAVE ADDRESS OF DATA
2A1D 9393      ;
2A1D 9394      ; FIRST CALCULATE THE P0 PARITY BIT
2A1D 9395      ;
2A1D 9396      LXI          D,P0_PARITY_MSK ; GET ADDRESS OF P0 BIT MAP
2A20 9397      MVI          B,0             ; INIT A PARITY FLAG
2A22 9398      MVI          C,10           ; SET A LOOP COUNT
2A24 9399      1$: LDAX      D             ; GET A BYTE OF THE BIT MAP
2A25 9400      ANA          M             ; AND WITH THE MICRO WORD
2A26 9401      JPE          2$           ; BRANCH IF PARITY EVEN
2A29 9402      MOV          A,B          ; GET THE PARITY FLAG
2A2A 9403      CMA          ;             ; COMPLIMENT IT
2A2B 9404      MOV          B,A          ; SAVE
2A2C 9405      2$: INX      D             ; INCREMENT BIT MAP ADDRESS
2A2D 9406      INX      H             ; INCREMENT DATA ADDRESS
2A2E 9407      DCR          C             ; DONE 10 BYTES?
2A2F 9408      JNZ          1$          ; BRANCH IF NO
2A32 9409      ;
2A32 9410      ; NOW SET OR CLEAR THE P0 BIT
2A32 9411      ;
2A32 9412      POP          H             ; GET THE MICRO WORD ADDRESS
2A33 9413      PUSH         H             ; SAVE AGAIN
2A34 9414      MVI          E,5          ; POINT AT BYTE 5
2A36 9415      MVI          D,0          ; ...
  
```

```
2A38 9416      DAD      D      ;  
2A39 9417      MOV      A,M    ; GET THE BYTE  
2A3A 9418      ANI      <^XBF> ; CLEAR THE P0 BIT  
2A3C 9419      MOV      C,A    ; SAVE  
2A3D 9420      MOV      A,B    ; GET THE PARITY FLAG  
2A3E 9421      ORA      A      ; SEE IF ITS ZERO  
2A3F 9422      MOV      A,C    ; GET DATA BACK  
2A40 9423      JZ       5$     ; BRANCH IF ALREADY ZERO  
2A43 9424      ORI      <^X40> ; SET THE P0 BIT  
2A45 9425 5$:   MOV      M,A    ; LOAD THE MICRO WORD  
2A46 9426      ;  
2A46 9427      ; NOW CALCULATE THE P1 PARITY BIT  
2A46 9428      ;  
2A46 9429      POP      H      ; GET ADDRESS OF DATA AGAIN  
2A47 9430      PUSH     H      ; SAVE  
2A48 9431      LXI      D,P1_PARITY_MSK ; GET ADDRESS OF P1 BIT MAP  
2A4B 9432      MVI      B,0    ; INIT A PARITY FLAG  
2A4D 9433      MVI      C,10   ; SET A LOOP COUNT  
2A4F 9434 10$:  LDAX     D      ; GET A BYTE OF THE BIT MAP  
2A50 9435      ANA      M      ; AND WITH THE MICRO WORD  
2A51 9436      JPE      12$    ; BRANCH IF PARITY EVEN  
2A54 9437      MOV      A,B    ; GET THE PARITY FLAG  
2A55 9438      CMA      ; COMPLIMENT IT  
2A56 9439      MOV      B,A    ; SAVE  
2A57 9440 12$:  INX      D      ; INCREMENT BIT MAP ADDRESS  
2A58 9441      INX      H      ; INCREMENT DATA ADDRESS  
2A59 9442      DCR      C      ; DONE 10 BYTES?  
2A5A 9443      JNZ      10$   ; BRANCH IF NO  
2A5D 9444      ;  
2A5D 9445      ; NOW SET OR CLEAR THE P1 BIT  
2A5D 9446      ;  
2A5D 9447      POP      H      ; GET THE MICRO WORD ADDRESS  
2A5E 9448      INX      H      ; POINT AT BYTE 2  
2A5F 9449      INX      H      ;  
2A60 9450      MOV      A,M    ; GET THE BYTE  
2A61 9451      ANI      <^XF7> ; CLEAR THE P1 BIT  
2A63 9452      MOV      C,A    ; SAVE  
2A64 9453      MOV      A,B    ; GET THE PARITY FLAG  
2A65 9454      ORA      A      ; SEE IF ITS ZERO  
2A66 9455      MOV      A,C    ; GET DATA BACK  
2A67 9456      JNZ      15$    ; BRANCH IF NOT ZERO  
2A6A 9457      ORI      8      ; SET THE P1 BIT  
2A6C 9458 15$:  MOV      M,A    ; LOAD THE MICRO WORD  
2A6D 9459      RET      ; EXIT  
2A6E 9460
```

```
2A6E 9462      .SBTTL      CONVERT BIT NUMBER TO BIT MASK ROUTINE
2A6E 9463      ;++
2A6E 9464      ;
2A6E 9465      ; FUNCTIONAL DESCRIPTION:
2A6E 9466      ;
2A6E 9467      ;     THIS ROUTINE CONVERTS A BIT NUMBER (BETWEEN 0 AND 7) INTO A MASK
2A6E 9468      ;     WITH ONE BIT SET IN THE POSITION DEFINED BY THE NUMBER
2A6E 9469      ;
2A6E 9470      ; INPUT PARAMETERS:
2A6E 9471      ;
2A6E 9472      ;     A REGISTER - CONTAINS THE BIT NUMBER
2A6E 9473      ;
2A6E 9474      ; OUTPUT PARAMETERS:
2A6E 9475      ;
2A6E 9476      ;     A REGISTER - CONTAINS THE BIT MASK
2A6E 9477      ;
2A6E 9478      ;--
2A6E 9479      ;
2A6E 9480      BIT_NMB_TO_MSK:
2A6E 9481      PUSH      B           ; SAVE THE B REGISTER
2A6F 9482      MOV      B,A       ; PUT BIT NUMBER IN B REG
2A70 9483      MVI      A,1       ; INIT THE BIT MASK
2A72 9484      1$:      DCR      B           ; FOUND THE BIT YET?
2A73 9485      JM      5$         ; BRANCH IF YES
2A76 9486      RLC      R0        ; ROTATE THE MASK
2A77 9487      JMP      1$        ; CONTINUE
2A7A 9488      5$:      POP      B           ; RESTORE THE B REGISTER
2A7B 9489      RET
2A7C 9490
2A7C 9491
```

```

2A7C 9493      .SBTTL      CHECK VBUS ROUTINE
2A7C 9494      :++
2A7C 9495      :
2A7C 9496      : FUNCTIONAL DESCRIPTION:
2A7C 9497      :
2A7C 9498      :     THIS ROUTINE MOVES A BIT FROM THE VBUS BUFFER TO 'BAD_DATA' AND
2A7C 9499      :     THE EXPECTED VALUE FROM A VBUS TABLE TO 'GOOD_DATA', COMPARES
2A7C 9500      :     THE TWO AND RETURNS WITH THE RESULT OF THE COMPARE.
2A7C 9501      :
2A7C 9502      : IMPLICIT INPUTS:
2A7C 9503      :
2A7C 9504      :     V_BUS_BUFFER      - CONTAINS THE CURRENT STATE OF THE VBUS
2A7C 9505      :
2A7C 9506      : EXPLICIT INPUTS:
2A7C 9507      :
2A7C 9508      :     H & L REGISTERS - CONTAINS THE ADDRESS OF THE EXPECTED VBUS BIT
2A7C 9509      :     AND VALUE.
2A7C 9510      :
2A7C 9511      : ROUTINE VALUE:
2A7C 9512      :
2A7C 9513      :     C BIT CLEAR IF COMPARE IS OK
2A7C 9514      :     C BIT SET IF COMPARE FAILED.
2A7C 9515      :
2A7C 9516      :--
2A7C 9517      :
2A7C 9518      CHECK_VBUS:
2A7C 9519      LXI          D,V_BUS_BUFFER      ; GET ADDRESS OF V BUS BUFFER
2A7F 9520      MOV          A,M              ; GET EXPECTED BIT NUMBER AND VALUE
2A80 9521      MOV          C,A              ; SAVE
2A81 9522      ANI          <^X7F>          ; DISCARD VALUE
2A83 9523      STA          GOOD_DATA       ; SAVE BIT NUMBER
2A86 9524      STA          BAD_DATA       ; SAVE BIT NUMBER IN BAD DATA
2A89 9525      MOV          A,C              ; GET BIT VALUE
2A8A 9526      RLC                      ; PUT IN POSITION 0
2A8B 9527      ANI          1                ; DISCARD GARBAGE
2A8D 9528      STA          GOOD_DATA+1    ; SAVE VALUE IN GOOD DATA
2A90 9529      MOV          A,M              ; GET BIT NUMBER AGAIN
2A91 9530      ANI          <^X7F>          ; THROW AWAY THE VALUE
2A93 9531      ADD          E              ; INDEX THE BUFFER ADDRESS TO
2A94 9532      MOV          E,A            ; THE SPECIFIED BIT NUMBER
2A95 9533      MVI          A,0            ; ...
2A97 9534      ADC          D              ; ...
2A98 9535      MOV          D,A            ; ...
2A99 9536      LDAX         D              ; GET THE RECEIVED VALUE OF THE BIT
2A9A 9537      STA          BAD_DATA+1    ; SAVE AS BAD DATA
2A9D 9538      LDA          GOOD_DATA+1    ; GET EXPECTED VALUE
2AA0 9539      MOV          C,A            ; SAVE
2AA1 9540      LDA          BAD_DATA+1    ; GET RECEIVED VALUE
2AA4 9541      CMP          C              ; DOES EXPECTED = RECEIVED?
2AA5 9542      JNZ          10$            ; BRANCH IF NO
2AA8 9543      STC                      ;
2AA9 9544      CMC                      ;
2AAA 9545      RET                          ; RETURN WITH C BIT CLEAR
2AAB 9546      STC 10$:                  ;
2AAC 9547      RET                          ; RETURN WITH C BIT SET
  
```

ZZ-ECKAA-8.7 V08.07
ECKAA
V08.07

H 3
CHECK VBUS R011-FEB-1986
VAX-11/750 MICRO DIAGNOSTIC MONITOR
CHECK VBUS ROUTINE

Fiche 2 Frame H3 Sequence 239
11-FEB-1986 10:07:52 VAX/VMS Macro V04-00 Page 214
11-FEB-1986 10:06:37 [VAX750.MONITOR]ECKAA.MAR;811 (92)

2AAD 9548
2AAD 9549

```

2AAD 9551      .SBTTL      CHECK QA FLAG
2AAD 9552      ;++
2AAD 9553      ;
2AAD 9554      ; FUNCTIONAL DESCRIPTION:
2AAD 9555      ;
2AAD 9556      ;         THIS ROUTINE CHECKS IF THE QA FLAG IS SET. IF II IS, AND THE
2AAD 9557      ;         QA COUNT HAS BEEN REACHED, THE ERROR FLAG AND LOOP ERROR FLAG
2AAD 9558      ;         IS SET.
2AAD 9559      ;
2AAD 9560      ; INPUT PARAMETERS:
2AAD 9561      ;
2AAD 9562      ;         NONE
2AAD 9563      ;
2AAD 9564      ; IMPLICIT INPUTS:
2AAD 9565      ;
2AAD 9566      ;         QA_COUNT - CONTAINS THE USER SPECIFIED COUNT TO START THE QA FUNCTION
2AAD 9567      ;         QA_INDEX - CONTAINS THE CURRENT INDEX USED IN THE 'COMPARE' ROUTINES
2AAD 9568      ;
2AAD 9569      ;--
2AAD 9570      CHECK_QA_FLAG:
2AAD 9571      LDA          PROG_CTRL_REG      ; GET THE QA FLAG
2AB0 9572      ANI          QA_FLAG           ; IS THE FLAG SET?
2AB2 9573      JZ          10$              ; BRANCH IF NO
2AB5 9574      LDA          QA_COUNT+1       ; GET HIGH BYTE OF COUNT
2AB8 9575      CPI          -1              ; WAS IT SPECIFIED?
2ABA 9576      JNZ         2$              ; BRANCH IF YES
2ABD 9577      LDA          QA_COUNT         ; GET LOW BYTE
2AC0 9578      CPI          -1              ; WAS IT SPECIFIED?
2AC2 9579      JZ          5$              ; BRANCH IF NO
2AC5 9580      ;
2AC5 9581      ; QA COUNT WAS SPECIFIED. CHECK IF WERE THERE YET
2AC5 9582      ;
2AC5 9583      2$:      LHLD         QA_COUNT      ; GET THE USER SPECIFIED COUNT
2AC8 9584      DCX          H                  ; ADJUST TO INDEX RANGE
2AC9 9585      LDA          QA_INDEX+1       ; GET HIGH BYTE OF CURRENT INDEX VALUE
2ACC 9586      CMP          H                  ; HIGH BYTE THERE YET?
2ACD 9587      JC          10$              ; BRANCH IF NO
2AD0 9588      JNZ         5$              ; BRANCH IF YES
2AD3 9589      LDA          QA_INDEX         ; GET LOW BYTE OF INDEX
2AD6 9590      CMP          L                  ; LOW BYTE THERE YET?
2AD7 9591      JC          10$              ; BRANCH IF NO
2ADA 9592      ;
2ADA 9593      ; SET THE ERROR AND LOOP ERROR FLAGS
2ADA 9594      ;
2ADA 9595      5$:      LDA          SOFT_CTRL_REGA ; GET THE FLAGS
2ADD 9596      ORI          <ERROR_FLAG!-
2ADD 9597      LOOP_ERROR_FLAG>
2ADF 9598      STA          SOFT_CTRL_REGA ; SET THE ERROR FLAGS
2AE2 9599      10$:     RET
2AE3 9600
2AE3 9601

```

```
2AE3 9603 .SBTTL TYPE GATE ARRAY LIST
2AE3 9604 ;**
2AE3 9605 ;
2AE3 9606 ; FUNCTIONAL DESCRIPTION:
2AE3 9607 ;
2AE3 9608 ; THIS ROUTINE IS USED TO TYPE A LIST OF GATE ARRAY NAMES.
2AE3 9609 ;
2AE3 9610 ; IMPLICIT INPUTS:
2AE3 9611 ;
2AE3 9612 ; H & L - CONTAIN ADDRESS OF LIST OF GATE ARRAY CODES
2AE3 9613 ; A - CONTAINS THE NUMBER OF CODES IN THE LIST
2AE3 9614 ;
2AE3 9615 ; IMPLICIT OUTPUTS:
2AE3 9616 ;
2AE3 9617 ; H & L - CONTAIN ADDRESS OF END OF LIST OF GATE ARRAY CODES
2AE3 9618 ;
2AE3 9619 ;--
2AE3 9620 TYPE_GA_LIST:
2AE3 9621 ;
2AE3 9622 ;
2AE3 9623 ; TYPE THE GATE ARRAY MESSAGE AND THE LIST
2AE3 9624 ;
2AE3 9625 PUSH $PSW ; SAVE THE GATE ARRAY COUNT
2AE4 9626 INX H ; POINT AT FIRST GATE ARRAY CODE
2AE5 9627 PUSH H ; SAVE LIST POINTER
2AE6 9628 TYPE CRLF
2AEF 9629 TYPE FAIL_ARRAY_MSG ; TYPE 'FAILING GATE ARRAYS: '
2AF8 9630 POP H ; GET THE GATE ARRAY LIST POINTER
2AF9 9631 1$: PUSH H ; SAVE THE POINTER
2AFA 9632 PUSH $PSW ; SAVE CODE
2AFB 9633 TYPE DC_MSG ; TYPE THE DC PREFIX
2B04 9634 POP $PSW
2B05 9635 POP H
2B06 9636 MOV A,M ; GET THE GATE ARRAY CODE
2B07 9637 CPI ALP ; CHECK IF BIT SLICE GATE ARRAY
2B09 9638 JC S$ ; BRANCH IF NOT
2B0C 9639 CPI END_SLICE ; CHECK IF BIT SLICE CODE
2B0E 9640 JNC S$ ; BRANCH IF NOT
2B11 9641 PUSH H
2B12 9642 CALL BIT_SLICE ; TYPE THE DC NUMBER
2B15 9643 JMP 10$ ; CONTINUE
2B18 9644 5$: PUSH H ; SAVE THE POINTER
2B19 9645 MOV C,A ; SAVE IN C REG
2B1A 9646 RLC ; MAKE CODE A 9 BYTE INDEX
2B1B 9647 RLC
2B1C 9648 RLC
2B1D 9649 ADD C
2B1E 9650 MOV C,A
2B1F 9651 MVI B,0
2B21 9652 LXI H,GATE_ARRAY_LST ; GET ADDRESS OF ASCII GATE ARRAY NAMES
2B24 9653 DAD B ; POINT AT ASCII NAME
2B25 9654 TYPE ; TYPE THE NAME
2B2B 9655 10$: TYPE CRLF ; TERMINATE THE LINE
2B34 9656 TYPE FAIL_ARR_INDEX ; SET CURSOR FOR NEXT ARRAY NAME
2B3D 9657 POP H ; GET THE LOOP COUNT FROM THE STACK
```

ZZ-ECKAA-8.7 V08.07
ECKAA
V08.07

K 3

TYPE GATE ARR11-FEB-1986

VAX-11/750 MICRO DIAGNOSTIC MONITOR
TYPE GATE ARRAY LIST

Fiche 2 Frame K3 Sequence 242
11-FEB-1986 10:07:52 VAX/VMS Macro V04-00 Page 217
11-FEB-1986 10:06:37 [VAX750.MONITOR]ECKAA.MAR;811 (94)

2B3E	9658	POP	\$PSW	:	...
2B3F	9659	DCR	A	:	DÉCREMENT
2B40	9660	PUSH	\$PSW	:	SAVE
2B41	9661	INX	H	:	POINT AT THE NEXT CODE
2B42	9662	JNZ	1\$:	CONTINUE LOOP
2B45	9663	POP	\$PSW	:	THROW AWAY LOOP COUNT
2B46	9664	PUSH	H	:	SAVE THE POINTER
2B47	9665	TYPE	CRLF	:	TERMINATE THE LINE
2B50	9666	POP	H	:	RESTORE THE POINTER
2B51	9667	DCX	H	:	BACKUP TO LAST GATE ARRAY CODE
2B52	9668	RET		:	EXIT
2B53	9669			:	

```
.SBTTL BIT SLICE ALGORITHM
2B53 9671 ;++
2B53 9672 ;
2B53 9673 ;
2B53 9674 ; FUNCTIONAL DESCRIPTION:
2B53 9675 ;
2B53 9676 ; THIS ROUTINE CHECKS TO SEE IF THE XOR OF THE GOOD AND BAD DATA
2B53 9677 ; FITS THE BIT SLICE FOR THE SPECIFIED GATE ARRAY.
2B53 9678 ;
2B53 9679 ; EXPLICIT INPUTS:
2B53 9680 ;
2B53 9681 ; A - CONTAINS GATE ARRAY CODE
2B53 9682 ;
2B53 9683 ; IMPLICIT INPUTS:
2B53 9684 ;
2B53 9685 ; BIT_SLICE_TBL - TABLE OF POINTERS TO BIT SLICE STRINGS
2B53 9686 ; INDEXED BY GATE ARRAY NUMBER
2B53 9687 ;
2B53 9688 ; THE BIT SLICE STRINGS HAVE THE FOLLOWING FORMAT:
2B53 9689 ;
2B53 9690 ; .BYTE Number of Slices
2B53 9691 ; .LONG Mask for slice # 1
2B53 9692 ; .LONG Mask for slice # 2
2B53 9693 ;
2B53 9694 ; .LONG Mask for slice # n
2B53 9695 ; .ASCIC /Chip name string for slice # 1/
2B53 9696 ; .ASCIC /Chip name string for slice # 2/
2B53 9697 ;
2B53 9698 ; .ASCIC /Chip name string for slice # n/
2B53 9699 ;--
2B53 9700 ;
2B53 9701 BIT_SLICE:
2B53 9702 ;
2B53 9703 ; CREATE THE INDEX TO THE SPECIFIED GATE ARRAY TABLE
2B53 9704 ;
2B53 9705 ; SUI ALP ; NORMALIZE GATE ARRAY NUMBER
2B55 9706 ; LXI H,BIT_SLICE_TBL ; GET ADDRESS OF POINTER TABLE
2B58 9707 ; RLC ; MAKE GATE ARRAY A WORD INDEX
2B59 9708 ; MOV E,A
2B5A 9709 ; MVI D,0
2B5C 9710 ; DAD D ; INDEX THE TABLE ADDRESS
2B5D 9711 ;
2B5D 9712 ; NOW GET THE ADDRESS OF THE SPECIFIED GATE ARRAY TABLE
2B5D 9713 ;
2B5D 9714 ; MOV E,M ; GET ADDRESS OF GATE ARRAY BIT SLICE
2B5E 9715 ; INX H ;
2B5F 9716 ; MOV D,M ; TABLE
2B60 9717 ; XCHG ; PUT IN H & L
2B61 9718 ;
2B61 9719 ; MOV B,M ; GET NUMBER OF SLICES IN THIS ARRAY
2B62 9720 ; INX H ; POINT AT FIRST SLICE MASK
2B63 9721 ;
2B63 9722 ; NOW GET THE XOR OF THE GOOD AND BAD DATA
2B63 9723 ;
2B63 9724 ; PUSH B
2B64 9725 ; PUSH H
```

```

2B65 9726 LXI D,BIT_SLICE_TEMP; GET ADDRESS OF TEMP STORAGE
2B68 9727 LXI B,BAD_DATA ; GET ADDRESS OF BAD DATA
2B6B 9728 LXI H,GOOD_DATA ; GET ADDRESS OF GOOD DATA
2B6E 9729 LDAX B ; GET BAD DATA
2B6F 9730 XRA M ; GET XOR WITH GOOD DATA
2B70 9731 STAX D ; SAVE (BYTE 0)
2B71 9732 INX B
2B72 9733 INX D
2B73 9734 INX H
2B74 9735 LDAX B ; BYTE 1
2B75 9736 XRA M
2B76 9737 STAX D
2B77 9738 INX B
2B78 9739 INX D
2B79 9740 INX H
2B7A 9741 LDAX B ; BYTE 2
2B7B 9742 XRA M
2B7C 9743 STAX D
2B7D 9744 INX B
2B7E 9745 INX D
2B7F 9746 INX H
2B80 9747 LDAX B ; BYTE 3
2B81 9748 XRA M
2B82 9749 STAX D
2B83 9750 POP H
2B84 9751 POP B
2B85 9752 DCX D ; BACKUP POINTER TO XOR DATA
2B86 9753 DCX D
2B87 9754 DCX D
2B88 9755 ;
2B88 9756 ; NOW CHECK XOR DATA WITH THE BIT SLICE MASKS
2B88 9757 ;
2B88 9758 ; B REG CONTAINS NUMBER OF SLICES
2B88 9759 ; H & L CONTAIN POINTER TO MASKS
2B88 9760 ; D & E CONTAIN POINTER TO XOR DATA
2B88 9761 ;
2B88 9762 ; PUSH B ; SAVE SLICE COUNT
2B89 9763 ; PUSH H ; AND POINTER
2B8A 9764 XCHG
2B8B 9765 1$: MVI C,4 ; SET COUNTER FOR NUMBER OF BYTES
2B8D 9766 5$: LDAX D ; GET A BYTE OF THE MASK
2B8E 9767 INX D
2B8F 9768 ANA M ; AND WITH XOR DATA
2B90 9769 INX H
2B91 9770 JNZ 10$ ; BRANCH IF FOUND THE FAILING SLICE
2B94 9771 DCR C ; CHECK ALL FOUR BYTES
2B95 9772 JNZ 5$ ; ...
2B98 9773 ;
2B98 9774 ; CHECK NEXT BIT SLICE
2B98 9775 ;
2B98 9776 DCX H ; BACKUP POINTER TO XOR DATA
2B99 9777 DCX H
2B9A 9778 DCX H
2B9B 9779 DCX H
2B9C 9780 DCR B ; CHECK ALL BIT SLICES

```

```
2B9D 9781      JNZ      1$
2BA0 9782      INR      B          ; FORCE LAST SLICE OUTPUT
2BA1 9783      :
2BA1 9784      : FOUND THE FAILING BIT SLICE
2BA1 9785      : B REG CONTAINS THE FAILING LOOP COUNT
2BA1 9786      :
2BA1 9787      10$: POP      H          ; GET POINTER TO SLICE COUNT
2BA2 9788      POP      D          ; AND NUMBER OF SLICES
2BA3 9789      MOV      A,D        ; GENERATE POINTER TO FIRST MESSAGE STRING
2BA4 9790      RLC
2BA5 9791      RLC
2BA6 9792      ADD      L          ; ...
2BA7 9793      MOV      L,A        ; ...
2BA8 9794      MVI      A,0        ; ...
2BAA 9795      ADC      H          ; ...
2BAB 9796      MOV      H,A        ; ...
2BAC 9797      MOV      A,D        ; GET SLICE NUMBER THAT FAILED
2BAD 9798      SUB      B          ; ...
2BAE 9799      MOV      C,A        ; PUT IN B & C
2BAF 9800      MVI      B,0        ; ...
2BB1 9801      MOV      A,M        ; GET LENGTH OF STRING
2BB2 9802      PUSH     H          ; SAVE H & L
2BB3 9803      PUSH     B          ; SAVE SLICE NUMBER
2BB4 9804      CALL    MULTIPLY_A_BC ; GET POINTER TO APPROPRIATE STRING
2BB7 9805      POP      H          ; GET SLICE NUMBER
2BB8 9806      DAD     D          ; ADD TO STRING INDEX
2BB9 9807      XCHG
2BBA 9808      POP      H          ; RESTORE H & L
2BBB 9809      DAD     D          ; POINT TO SPECIFIED STRING
2BBC 9810      TYPE
2BC2 9811      RET          ; EXIT
```

```
2BC3 9813      .SBTTL      READ RTEMP REGISTER
2BC3 9814      ;++
2BC3 9815      ;
2BC3 9816      ; FUNCTIONAL DESCRIPTION:
2BC3 9817      ;
2BC3 9818      ;     THIS ROUTINE IS USED TO READ THE CONTENTS OF AN RTEMP REGISTER
2BC3 9819      ;     IN THE 11/750 CPU.
2BC3 9820      ;
2BC3 9821      ; EXPLICIT INPUTS:
2BC3 9822      ;
2BC3 9823      ;     A REG - CONTAINS THE REGISTER NUMBER TO READ
2BC3 9824      ;
2BC3 9825      ; EXPLICIT OUTPUTS:
2BC3 9826      ;
2BC3 9827      ;     WH REG - CONTAINS THE CONTENTS OF THE SPECIFIED R TEMP REGISTER
2BC3 9828      ;
2BC3 9829      ; SIDE EFFECTS:
2BC3 9830      ;
2BC3 9831      ;     ALL 8085 REGISTERS ARE LOST
2BC3 9832      ;--
2BC3 9833      READ_RTEMP:
2BC3 9834          STA          READ_R_TEMP      ; SAVE REGISTER NUMBER
2BC6 9835          LXI          D,RDM_R_MIC_WORD; GET ADDRESS OF MICRO WORD
2BC9 9836          LXI          H,READ_R_TEMP    ; GET ADDRESS OF REGISTER NUMBER
2BCC 9837          MVI          B,34            ; SET STARTING BIT NUMBER
2BCE 9838          MVI          C,39            ; AND ENDING BIT NUMBER
2BD0 9839          XRA          A              ; CLEAR INHIBIT PARITY FLAG
2BD1 9840          PUSH         $PSW          ; CLEAR INHIBIT SCRAMBLE FLAG
2BD2 9841          CALL         INSERT_FIELD   ; PUT REGISTER NUMBER IN MICRO WORD
2BD5 9842          WRITE$_DCS  DCS_SCRATCH_ADR,-
2BD5 9843          RDM_R_MIC_WORD,1; LOAD THE MICRO WORD
2BE8 9844          WRITE$_DCS  DCS_SCRATCH_ADR+1,-
2BE8 9845          NOP_MICRO_WORD,1; ...
2BFB 9846          EXECUTE     DCS_SCRATCH_ADR,1; GET CONTENTS OF REGISTER
2C0E 9847          RET                      ; EXIT
```

```
2COF 9849 .SBTTL " DISPLAY DATA
2COF 9850 :++
2COF 9851 :
2COF 9852 : FUNCTIONAL DESCRIPTION:
2COF 9853 :
2COF 9854 : THIS ROUTINE IS USED TO TYPE A <TAB> FOLLOWED BY THE LONG WORD
2COF 9855 : CONTENTS OF THE SPECIFIED LOCATION, FOLLOWED BY A CRLF.
2COF 9856 :
2COF 9857 : EXPLICIT INPUTS:
2COF 9858 :
2COF 9859 : H & L - CONTAIN ADDRESS OF DATA TO TYPE
2COF 9860 :
2COF 9861 :--
2COF 9862 DISPLAY_DATA:
2COF 9863 PUSH H ; SAVE ADDRESS
2C10 9864 TYPE TAB ; TYPE THE TAB
2C19 9865 POP H
2C1A 9866 TYPEL ; TYPE THE DATA
2C22 9867 TYPE CRLF ; TERMINATE THE LINE
2C2B 9868 RET ; EXIT
```

```
2C2C 9870      .SBTTL      READ MSRC FIELD
2C2C 9871      :++
2C2C 9872      :
2C2C 9873      : FUNCTIONAL DESCRIPTION:
2C2C 9874      :
2C2C 9875      :           THIS ROUTINE IS USED TO READ THE CONTENTS OF AN MSRC FIELD
2C2C 9876      :           IN THE 11/750 CPU.
2C2C 9877      :
2C2C 9878      : EXPLICIT INPUTS:
2C2C 9879      :
2C2C 9880      :           A REG - CONTAINS THE REGISTER NUMBER TO READ
2C2C 9881      :
2C2C 9882      : EXPLICIT OUTPUTS:
2C2C 9883      :
2C2C 9884      :           WH REG - CONTAINS THE CONTENTS OF THE SPECIFIED M SRC REGISTER
2C2C 9885      :
2C2C 9886      : SIDE EFFECTS:
2C2C 9887      :
2C2C 9888      :           ALL 8085 REGISTERS ARE LOST
2C2C 9889      :--
2C2C 9890      READ_MTEMP:
2C2C 9891      STA          READ_R_TEMP      ; SAVE REGISTER NUMBER
2C2F 9892      LXI          D,RDM_M_MIC_WORD; GET ADDRESS OF MICRO WORD
2C32 9893      LXI          H,READ_R_TEMP  ; GET ADDRESS OF REGISTER NUMBER
2C35 9894      MVI          B,64           ; SET STARTING BIT NUMBER
2C37 9895      MVI          C,68           ; AND ENDING BIT NUMBER
2C39 9896      XRA          A              ; CLEAR INHIBIT PARITY FLAG
2C3A 9897      PUSH        $PSW           ; CLEAR INHIBIT SCRAMBLE FLAG
2C3B 9898      CALL        INSERT_FIELD   ; PUT REGISTER NUMBER IN MICRO WORD
2C3E 9899      WRITE$_DCS  DCS_SCRATCH_ADR,-
2C3E 9900      RDM_M_MIC_WORD,1; LOAD THE MICRO WORD
2C51 9901      WRITE$_DCS  DCS_SCRATCH_ADR+1,-
2C51 9902      NOP_MICRO_WORD,1; ...
2C64 9903      EXECUTE   DCS_SCRATCH_ADR,1; GET CONTENTS OF REGISTER
2C77 9904      RET              ; EXIT
2C78 9905
2C78 9906      ;
2C78 9907      ; THE TEST OVERLAYS GET LOADED STARTING HERE
2C78 9908      ;
2C78 9909
2C78 9910
2C78 9911
2C78 9912      TEST_BUFFER:
2C78 9913
2C78 9914      .END
```

\$CONTROL_C_CODE=	0000000A			CLA	=	00000003			CS_ADD_TRAP_HGH=	00007423		
\$FILE_NOT_FOUND=	00000009			CLEAR_ACTION	=	00001945	R	01	CS_ADD_TRAP_LOW=	00007422		
\$PSW	=	00000006		CLEAR_CTRL_FILE=	=	00000002			CURRENT_BIT	=	00000C9A	R 01
\$SP	=	00000006		CLEAR_DCS_CF	=	00000FD7	R	01	CURRENT_PC	=	000003B8	R 01
A	=	00000007		CLEAR_DONE	=	000019BC	R	01	CURRENT_REC_NMB	=	0000013B	R 01
ACV	=	0000000F		CLEAR_ERR	=	000019BF	R	01	CYCLE_FLAG	=	00000008	
AC_LOW	=	00000080		CLKDCS	=	00007427			CYCLE_KEY	=	00000020	
ADD	=	00000019		CLK_CTRL_0	=	00000020			C_INIT_TEST_PC	=	000001B7	R 01
ADDR_MATCH_HI	=	00007405		CLK_CTRL_0_RO	=	00000040			D	=	00000002	
ADDR_MATCH_LO	=	00007404		CLK_CTRL_1	=	00000040			DATA_MSG	=	000005B7	R 01
ADD_NAME	=	00000963	RG 01	CLK_CTRL_1_RO	=	00000080			DATA_TYPE	=	00000447	R 01
ADK	=	0000000B		CLK_MSG_ADDR	=	00000453	R	01	DCSAD	=	00007401	
ALK	=	00000000		CLK_STOP_MSG1	=	00000481	R	01	DCSCF	=	00007403	
ALP	=	00000016		CLK_STOP_MSG2	=	000004A3	R	01	DCSCR	=	00007402	
ALP_NAME	=	00000713	RG 01	CLK_STOP_MSG3	=	000004B4	R	01	DCSDA	=	00007400	
ARG_LIST	=	000003BE	R 01	CLK_STOP_MSG4	=	000004BA	R	01	DCS_ADDR_CLEAR	=	00000002	
A_REG_BYTE_0	=	00007410		CLK_STOP_MSG5	=	000004CC	R	01	DCS_ADDR_MSG	=	00000456	R 01
A_REG_BYTE_1	=	00007411		CLOCK_STOPPED	=	00000080			DCS_ADDR_REG_RO	=	00007427	
A_REG_BYTE_2	=	00007412		CMD_DISPATCH	=	00000304	R	01	DCS_ADDR_REG_WO	=	00007401	
A_REG_BYTE_3	=	00007413		CMD_ENTRY	=	0000000D	R	01	DCS_BIT_MAP	=	00000C9E	R 01
B	=	00000000		CMD_LINE_LENGTH	=	000002F1	R	01	DCS_CTRL_FILE	=	00007403	
BADD1	=	00007426		CMI	=	00000007			DCS_CTRL_REG	=	00007402	
BAD_DATA	=	00000443	P 01	CMICTL	=	0000741C			DCS_CTRL_RE_CPY	=	0000005C	R 01
BEGIN_LOOP	=	00001CE0	R 01	CMI_COMPLETE	=	00000020			DCS_DATA_REG	=	00007400	
BELL	=	00000592	R 01	CMI_CTRL_CLEAR	=	00000001			DCS_SCRATCH_ADR	=	00000036	
BELL_COUNT	=	00000591	R 01	CMI_CTRL_REG	=	0000741C			DCS_START	=	00001800	
BELL_FLAG	=	00000008		CMI_GO	=	00000008			DCS_WRITE_ADR	=	000001AC	R 01
BELL_KEY	=	0000000E		CMI_NAME	=	000000F2	R	01	DCS_WRITE_PTR	=	000001AE	R 01
BIT_COUNT	=	00000C9B	R 01	CMI_READ	=	00000040			DCS_WRITE_WDCNT	=	000001B0	R 01
BIT_NMB_TO_MSK	=	00002A6E	R 01	CMI_STATUS_0	=	00000008			DC_LOW	=	00000040	
BIT_SLICE	=	00002B53	R 01	CMI_STATUS_1	=	00000010			DC_MSG	=	0000062C	R 01
BIT_SLICE_TBL	=	000006F5	RG 01	CMI_WRITE	=	00000020			DECODE_ERR	=	00001B97	R 01
BIT_SLICE_TEMP	=	000002C4	R 01	CML	=	0000000E			DECODE_FLAG_NAM	=	00001B85	R 01
BREAK_ACTION	=	000015A1	R 01	CMP_GOOD_BAD	=	00002719	R	01	DEPOSIT_ACTION	=	00001B84	R 01
BUFF_ADDR_LOW	=	00007426		COMMAND_LIST	=	0000032E	R	01	DESELECT_DCS	=	00001383	R 01
BURST_CLOCK	=	000020B5	R 01	COMMAND_PARSE	=	000015A8	R	01	DIAGNOSE_ACTION	=	000016E5	R 01
BURST_STOP_FLAG	=	00000080		COMMA_MSG	=	0000062A	R	01	DIAGNOSE_DONE	=	000017A2	R 01
BURST_TEMP	=	00000451	R 01	COMPARE_REG	=	00001F07	R	01	DIAGNOSE_ERR	=	000017A0	R 01
BYTE	=	00000001		COMPARE_REG_MSK	=	00001F5A	R	01	DIAGNOSE_FLAG	=	00000004	
C	=	00000001		COMPARE_VBUS	=	00001FE4	R	01	DIAGNOSE_LOOP	=	00001712	R 01
CACHE_ENA_ADR	=	00000275	R 01	CON	=	00000011			DIRECT_MSG	=	00000076	R 01
CACHE_ENA_DATA	=	00000280	R 01	CONTINUE_ACTION	=	0000183A	R	01	DIRECT_SEARCH	=	0000121B	R 01
CACHE_WRITE	=	0000028B	R 01	CONTINUE_FLAG	=	00000002			DISABLE_CMI	=	000013D5	R 01
CAK	=	0000000A		CONTROL_C_FLAG	=	00000001			DISPLAY_DATA	=	00002C0F	R 01
CALC_PARITY	=	00002A1C	R 01	CONT_FLAG	=	00000020			DPM	=	00000000	
CCC	=	00000004		CONT_KEY	=	0000001C			DPH_NAME	=	000000A5	R 01
CCF_SUB_ACT	=	000019AB	R 01	CONT_SUB_ACTION	=	0000175A	R	01	DUMP_EXIT	=	00002689	R 01
CCS	=	00000005		CPU_INIT	=	00001459	R	01	DUMP_LOG	=	00002646	R 01
CCS_NAME	=	000000DC	R 01	CPU_RUN	=	00000010			DUMP_RET	=	00002692	R 01
CFLAG_SUB_ACT	=	0000198F	R 01	CRLF	=	00000CEE	R	01	E	=	00000003	
CF_KEY	=	00000024		CSAMR	=	00007404			ENABLE_CMI	=	00001417	R 01
CHECK_QA_FLAG	=	00002AAD	R 01	CSBUF	=	00007424			ENABLE_STALL	=	00002520	R 01
CHECK_QV	=	000011D4	R 01	CSTRP	=	00007422			ENABLE_TRAP	=	00002517	R 01
CHECK_VBUS	=	00002A7C	R 01	CS_ADD_BUFF_HGH	=	00007425			ENDPASS_MSG	=	00000584	R 01
CHR	=	00000005		CS_ADD_BUFF_LOW	=	00007424			END_LOOP	=	00001D29	R 01

END_PASS	0000242E	R	01	FILE_FOR_MIC	00000154	R	01	J_INDEX_TBL	00000415	R	01
END_SLICE	= 00000025			FILE_FOR_RDM	00000166	R	01	KEYWORD_LIST	00000343	R	01
END_TEST	00002416	R	01	FILE_NAME_HEAD	= 0000014E	R	01	K_INDEX_TBL	0000041C	R	01
EN_TRACE_REG	= 00000008			FILE_NAME_TAIL	= 0000016C	R	01	L	= 00000005		
ERROR_FLAG	= 00000010			FILE_NAM_INDEX	00000063	R	01	LITERAL	= 00000001		
ERROR_LOG	00002529	R	01	FILE_NAM_PTR	00000139	R	01	LOAD_DCS	00001C0A	R	01
ERRGR_LOG_ADR	= 000032FC			FILL_DCS	000014D1	R	01	LOAD_FIELD	00002437	R	01
ERROR_LOOP	00001D88	R	01	FIO	= 0000001D			LOAD_INDEX	000024B8	R	01
ERROR_LOOP_ADDR	00000425	R	01	FIO_NAME	00000AF1	RG	01	LOAD_MEMSCAR	00000228	R	01
ERROR_MSG	0000059B	R	01	FLAG_KEY	= 00000004			LOAD_MEMSCR	00000249	R	01
ERROR_PC	00000594	R	01	FMR	= 00000022			LOAD_REG	00001DF8	R	01
ERR_LOG_INIT	= 00000001			FMR_NAME	00000BED	RG	01	LOG_ADDRESS	000004E5	R	01
EXAMINE_ACTION	000019C1	R	01	FPA	= 00000002			LOG_EXIT	00002607	R	01
EXAMINE_DISPLAY	00001B79	R	01	FPA_NAME	000000BB	R	01	LOG_FAIL	000025EF	R	01
EXAMINE_ERR	00001B7F	R	01	FP_BOOT_0	= 00000001			LOG_MSG_1	000004E9	R	01
EXAMINE_EX	00001B81	R	01	FP_BOOT_1	= 00000002			LOG_MSG_2	00000507	R	01
EXAMINE_MA	00001A5B	R	01	FP_START_0	= 00000004			LOG_MSG_3	00000533	R	01
EXAMINE_MD	00001A51	R	01	FP_START_1	= 00000008			LOG_SUCCESS	000025FC	R	01
EXAMINE_MT	00001A39	R	01	FQA	= 00000014			LOG_SYNDROME	000004E8	R	01
EXAMINE_PB	00001A56	R	01	FRONT1	= 00007420			LONG	= 00000004		
EXAMINE_PC	00001A47	R	01	FRONT2	= 00007421			LOOP_ACTION	00001848	R	01
EXAMINE_PS	00001AD5	R	01	FRONT_PNL_1	= 00007420			LOOP_ADDRESS	00000423	R	01
EXAMINE_RT	00001A2B	R	01	FRONT_PNL_2	= 00007421			LOOP_ERROR_FLAG	= 00000020		
EXAMINE_SF	00001B41	R	01	FRONT_PNL_LOCK	= 00000080			LOOP_FLAG	= 00000002		
EXAMINE_SR	00001A89	R	01	GATE_ARRAY_LST	0000062F	R	01	LOOP_KEY	= 0000000A		
EXAMINE_ST	00001B13	R	01	GET_CMD_LINE	000014EF	R	01	LOST_FLAG	= 00000010		
EXAMINE_TEMP	000003BC	R	01	GET_INDEX_VALUE	000026AF	R	01	LOST_KEY	= 0000001A		
EXAMINE_VA	00001A4C	R	01	GOOD_DATA	0000043F	R	01	LOW	= 00000000		
EXAMINE_WD	00001A60	R	01	H	= 00000004			M	= 00000006		
EXP_STALL_FLAG	= 00000010			HALT_FLAG	= 00000001			M\$TEST_SIZE	= 000007A2		
EXP_UTRAP_FLAG	= 00000040			HALT_FLAG_NAME	0000034B	R	01	M\$TEST_SIZE_D	= 000003E2		
EX_DISP_WH	00001B76	R	01	HALT_KEY	= 00000008			MA0	= 00000008		
FAC	= 00000020			HALT_ON_MATCH	= 00000001			MA0_NAME	000000FD	R	01
FAC_NAME	00000B9E	RG	01	HEAD	= FFFF7C00			MA1	= 0000000A		
FAIL_ARRAY_MSG	000005DA	R	01	HIGH	= 00000001			MA1_NAME	00000113	R	01
FAIL_ARR_INDEX	000005F0	R	01	HYPHEN	000005CE	R	01	MA2	= 0000000C		
FAIL_MOD_INDEX	00000618	R	01	H_FROM_WBUS	= 00000010			MA2_NAME	00000129	R	01
FAULT	= 00000002			IB_FLAG	= 00000020			MAD	= 00000013		
FCC	= 00000015			IB_KEY	= 00000012			MAIN32	= 0000741D		
FCS	= 00000021			IF_ERROR	00002052	R	01	MAINT_A_TO_CMI	= 00000080		
FCS_NAME	00000BB0	RG	01	INCREMENT_INDEX	000024FA	R	01	MAINT_CMI_TO_MH	= 00000020		
FETCH_0	0000147B	R	01	INDEX_NAM_TBL	00000408	R	01	MAINT_DCS_ENABL	= 00000004		
FETCH_FLAG	= 00000008			INIT_LOOP_VALUE	0000269B	R	01	MAINT_MD_TO_CMI	= 00000040		
FETCH_MIC INSTR	00001EA6	R	01	INIT_TEST_PC	000001B5	R	01	MAINT_REG	= 0000741D		
FEX	= 0000001E			INIT_THE_WORLD	00001CBF	R	01	MAINT_STB_INH	= 00000002		
FEX_NAME	00000B6A	RG	01	INSERT_FIELD	00002966	R	01	MAINT_TRAP_OFF	= 00000001		
FFH	= 00000024			INSERT_FIELD_X	00002A1B	R	01	MAINT_WB_TO_WH	= 00000008		
FFH_NAME	00000C49	RG	01	INSTR_FLAG	= 00000004			MAINT_WD_TO_WB	= 00000010		
FFL	= 00000023			INSTR_KEY	= 00000018			MAP	= 00000012		
FFL_NAME	00000C0C	RG	01	INSTR_TABLE	000003CE	R	01	MAREG	= 00007410		
FIC	= 0000001F			INS_FLD_BUFF	00000C90	R	01	MASK	00001ECD	R	01
FIC_NAME	00000B89	RG	01	INT	= 00000010			MASK_DISPLAY	00001B6C	R	01
FILE_FOR_CMC	00000160	R	01	INVALID_COMMAND	000002F3	R	01	MASTER_HALT_EN	= 00000008		
FILE_FOR_DPM	0000014E	R	01	IRD	= 00000005			MAX_ARGUMENTS	= 00000010		
FILE_FOR_FPA	0000015A	R	01	I_INDEX_TBL	0000040E	R	01	MA_KEY	= 00000038		

MCO	=	00000003		ONEQUAL	=	00000002		PRIM_REV	=	00000008		
MCO_NAME	=	000000C6	R	ONERROR	=	00000000		PRINT_N	=	00002617	R	01
MC1	=	00000009		OPEN_TU58	=	00000E2F	R	PRINT_N_EX	=	00002645	R	01
MC1_NAME	=	00000108	R	OP_BEGIN_LOOP	=	00000008		PRINT_S	=	00002608	R	01
MC2	=	0000000B		OP_BURST_CLOCK	=	00000002		PRK	=	0000000C		
MC2_NAME	=	0000011E	R	OP_CODE_DISP	=	00001B99	R	PROGRAM_INIT	=	00000CF1	R	01
MDL	=	0000001B		OP_COMPARE_REG	=	00000004		PROGRAM_TITLE	=	00000067	R	01
MDL_NAME	=	000009EF	RG	OP_COMPARE_REGM	=	00000006		PROG_CTRL_REG	=	0000005D	R	01
MDR	=	00000018		OP_COMPARE_VB	=	00000020		PROMPT	=	000002EC	R	01
MDREG	=	00007414		OP_DUMPLOG	=	00000030		PS_KEY	=	0000003E		
MDR_NAME	=	0000087A	RG	OP_ENABL_STALL	=	00000038		PUT_D_IN_TBL	=	0000166E	R	01
MD_KEY	=	00000034		OP_ENABL_TRAP	=	0000002E		QA_COUNT	=	00000449	R	01
MD_REG_BYTE_0	=	00007414		OP_END_FILE	=	00000024		QA_FLAG	=	00000040		
MD_REG_BYTE_1	=	00007415		OP_END_LOOP	=	0000000A		QA_INDEX	=	0000044B	R	01
MD_REG_BYTE_2	=	00007416		OP_END_TEST	=	0000000C		QA_KEY	=	00000014		
MD_REG_BYTE_3	=	00007417		OP_ERRLOG	=	00000032		QV_FLAG	=	00000002		
MEC	=	0000001A		OP_ERROR_LOOP	=	0000000E		QV_KEY	=	00000028		
MEC_NAME	=	000009C0	RG	OP_FETCH	=	00000022		QV_SUB_ACTION	=	0000174F	R	01
MESSAGE_LENGTH	=	0000016E	R	OP_IF_ERROR	=	00000012		RDCTRL	=	0000741E		
MESSAGE_POINTER	=	0000016C	R	OP_INC_INDEX	=	0000002A		RDM	=	00000004		
MHREG	=	00007418		OP_INIT	=	00000014		RDM_FLAG	=	00000004		
MH_REG_BYTE_0	=	00007418		OP_LOAD_DCS	=	00000000		RDM_INDEX	=	00000004		
MH_REG_BYTE_1	=	00007419		OP_LOAD_FIELD	=	00000026		RDM_KEY	=	0000002A		
MH_REG_BYTE_2	=	0000741A		OP_LOAD_INDEX	=	00000028		RDM_LAST	=	00000008		
MH_REG_BYTE_3	=	0000741B		OP_LOAD_REG	=	00000016		RDM_M_MIC_WORD	=	000001DB	R	01
MIC	=	00000001		OP_MASK	=	00000018		RDM_NAME	=	000000D1	R	01
MICROWORD	=	0000000A		OP_NEW_TEST	=	0000001A		RDM_PS_MIC_WORD	=	00000207	R	01
MICRO_ADDR_INH	=	00000080		OP_PATTERN_GEN	=	00000010		RDM_R_MIC_WORD	=	000001D0	R	01
MICRO_CODE_HEAD	=	000001BA	R	OP_PRINT_N	=	00000036		RDM_SF_MIC_WORD	=	0000021D	R	01
MICRO_CODE_TAIL	=	000002B7	R	OP_PRINT_S	=	00000034		RDM_SR_MIC_WORD	=	000001F1	R	01
MIC_INSTR_CNT	=	000001B1	R	OP_SAVE_INDEX	=	0000002C		RDM_ST_MIC_WORD	=	00000212	R	01
MIC_NAME	=	000000B0	R	OP_SKIP	=	0000001C		RDM_SUB_ACTION	=	00001744	R	01
MIC_TICK_CNT	=	000001B2	R	OP_SUB_TEST	=	0000001E		RDM_TAIL	=	0000016C	R	01
MME_DIS_ADR	=	00000280	R	PO_PARITY_MSK	=	00000570	R	RDM_WD_MIC_WORD	=	000001E6	R	01
MME_DIS_DATA	=	00000280	R	P1_PARITY_MSK	=	0000057A	R	RD_CTRL_REG	=	0000741E		
MODE_TYPE	=	00000448	R	PARITY_ERROR	=	00000020		READ_MTEMP	=	00002C2C	R	01
MODULE_LST	=	000000A5	R	PARSE_DONE	=	0000161E	R	READ_OVERLAY	=	0000100E	R	01
MODULE_MSG	=	00000606	R	PARSE_ERR	=	0000161C	R	READ_RTEMP	=	00002BC3	R	01
MODULE_NAM_TBL	=	0000008B	R	PARSE_HEX_DONE	=	000016CB	R	READ_R_TEMP	=	00000C9D	R	01
MONITOR_SIZE	=	00002C5E		PARSE_HEX_ERR	=	000016E3	R	READ_TU58	=	00000DFC	R	01
MSQ	=	00000008		PARSE_HEX_NUMB	=	00001679	R	READ_V_BUS	=	000011FB	R	01
MTEMP_COMMON	=	00001A41	R	PARSE_HEX_TEMP	=	00000388	R	REMOTE	=	00000001		
MT_KEY	=	0000002E		PARSE_TABLE	=	0000031A	R	RETURN_ACTION	=	00001845	R	01
MULTIPLY_A_BC	=	000026D6	R	PARSE_TBL_PTR	=	00000318	R	RETURN_TO_RDM	=	00000F81	R	01
M_CLK_ADDR	=	00000455	R	PAR_CHK_ENABL	=	00000008		RING_BELL	=	00002743	R	01
NER_FLAG	=	00000004		PAR_STOP_ENABL	=	00000010		RT_KEY	=	0000002C		
NER_KEY	=	0000000C		PASS_COUNT	=	00000062	R	R_RDM_MIC_WORD	=	000001C5	R	01
NEW_TEST	=	00001C3B	R	PASS_KEY	=	00000002		SAC	=	00000009		
NOERROR	=	00000001		PASS_SUB_ACTION	=	00001797	R	SAVED_SP	=	0000005E	R	01
NOP_MICRO_WORD	=	000001BA	R	PATTERN_ADDRESS	=	00000427	R	SAVE_BAD_DATA	=	000026E3	R	01
NOTEQUAL	=	00000003		PATTERN_GEN	=	00001DBF	R	SAVE_GOOD_DATA	=	000026FE	R	01
NOT_MSG	=	000005C0	R	PB_INIT	=	00000040		SAVE_INDEX	=	00002505	R	01
NO_MICRO_WORDS	=	00000017		PB_KEY	=	00000036		SA_CLOCK	=	00000080		
NUMBER_OF_FILES	=	00000005		PC_KEY	=	00000030		SA_FLAG	=	00000010		
ONE	=	000002F2	R	PHB	=	00000007		SA_KEY	=	00000010		

SA_START_STOP	=	00000080			SUB_TEST_NUMB	00000061	R	01	UTR	=	0000000D			
SBE_FLAG	=	00000020			TAB	000005C5	R	01	VAL	=	00002305			
SBE_KEY	=	00000042			TEMP_BUFFER	000001A3	R	01	VA_KEY	=	00000032			
SCRAMBLE_FLAG		00000C9C	R	01	TERM_ERROR	0000018E	R	01	VBUS_CLOCK	=	00000080			
SEARCH_DONE		00001648	R	01	TERM_ERR_CODE	0000018D	R	01	VBUS_COMPARE	=	00000080			
SEARCH_ERR		0000164C	R	01	TERM_INP_BUFF	0000016F	R	01	VBUS_LOAD	=	00000010			
SEARCH_LIST		00001621	R	01	TERM_INTERRUPT	00000ED5	R	01	VBUS_MSG		000003AC	R	01	
SEC_REV	=	00000007			TEST	=	00000004		VBUS_MSG_2		000005D0	R	01	
SELECT_DCS		00001392	R	01	TESTS_PER_LINE		000002B8	R	01	VBUS_SERIAL_IN	=	00000040		
SETCLR_CF_ADDR		000001B3	R	01	TEST_BUFFER		00002C78	R	01	VBUS_SERIAL_OUT	=	00000001		
SETCLR_CF_DATA		000001B4	R	01	TEST_KEY	=	00000000		VBUS_TBL_ADDR		0000044D	R	01	
SET_ACTION		00001863	R	01	TEST_MSG		000005A4	R	01	VBUS_TBL_COUNT		0000044F	R	01
SET_DCS_CF		00000F9F	R	01	TEST_MSG_1		000002BB	R	01	VB_KEY	=	00000026		
SET_DONE		00001940	R	01	TEST_MSG_2		000002C1	R	01	V_BUS_BUFFER		000002C4	R	01
SET_ERR		00001943	R	01	TEST_NUMBER		00000060	R	01	V_BUS_STROBE	=	00000001		
SFLAG_SUB_ACT		00001899	R	01	TEST_PC		000003B6	R	01	WBUS_FROM_D	=	00000020		
SF_KEY	=	00000040			TEST_SIZE		000001B9	R	01	WDREG	=	0000742C		
SHOW_ACTION		000017AD	R	01	TEST_SPAN_END		0000038D	R	01	WD_KEY	=	0000003A		
SHOW_MESSAGE_1		00000391	R	01	TEST_SUB_ACTION		00001765	R	01	WD_REG_BYTE_0	=	0000742C		
SHOW_MESSAGE_2		0000039D	R	01	TICK_FLAG	=	00000002			WD_REG_BYTE_1	=	0000742D		
SHOW_TEMP		000003AB	R	01	TICK_KEY	=	00000022			WD_REG_BYTE_2	=	0000742E		
SINGLE_INSTR		00002906	R	01	TITLE_LENGTH		000002B9	R	01	WD_REG_BYTE_3	=	0000742F		
SINGLE_MIC_INST		00000F30	R	01	TOK	=	00000006			WHREG	=	00007430		
SINGLE_TICK		00000F57	R	01	TOTAL_BLOCKS		0000013D	R	01	WH_REG_BYTE_0	=	00007430		
SKIP		00001E31	R	01	TOTAL_NMB_TESTS		00000066	R	01	WH_REG_BYTE_1	=	00007431		
SKIP_SPACES		0000164E	R	01	TPC_MSG		00000C86	R	01	WH_REG_BYTE_2	=	00007432		
SOFT_CTRL_REG		00000059	R	01	TRACE_ENABL	=	00000004			WH_REG_BYTE_3	=	00007433		
SOFT_CTRL_REGA		0000005A	R	01	TRAP_HALT_EN	=	00000020			WORD	=	00000002		
SOFT_CTRL_REGB		0000005B	R	01	TRAP_OFF	=	00000002			WRITE_DCS		00000EF4	R	01
SOMM_ADDRESS		0000038F	R	01	TR_FLAG	=	00000080			X	=	00000003		
SOMM_FLAG	=	00000001			TR_KEY	=	00000016			Y	=	0000004E		
SOMM_KEY	=	00000006			TSTSPAN_FLAG	=	00000040							
SOMM_MSG		00000464	R	01	TU58_BUFF_ADDR		00000134	R	01					
SPA	=	00000002			TU58_ERROR		00000140	R	01					
SPEC_TEST_NUMB		0000038C	R	01	TU58_ERR_CODE		0000013F	R	01					
SPEC_TEST_PC		000003BA	R	01	TU58_REC_COUNT		00000136	R	01					
SRK	=	00000001			TU58_REC_NUMB		00000137	R	01					
SRM	=	00000017			TYPEN_DATA_PTR		000001A0	R	01					
SRM_NAME		000007CC	RG	01	TYPEN_DATA_TYPE		000001A2	R	01					
SR_KEY	=	0000003C			TYPE_ASCII		00000E82	R	01					
SSOMM_SUB_ACT		00001916	R	01	TYPE_ASCII_U		00000E8D	R	01					
SSTEP_SUB_ACT		000018D6	R	01	TYPE_ERROR		00002764	R	01					
STACK_SIZE	=	00000400			TYPE_ERR_BUFFER		00000596	R	01					
START1		0000000E	R	01	TYPE_ERR_TEMP		0000059A	R	01					
STARTING_RECORD		00000064	R	01	TYPE_FLAGS		0000149A	R	01					
START_EXECUTION		0000139F	R	01	TYPE_GA_LIST		00002AE3	R	01					
STATUS	=	00007406			TYPE_NUMERIC		00000E9F	R	01					
STATUS_REG	=	00007406			TYPE_TEST_FLAG		000002B7	R	01					
STEP_ADDRESS		00000450	R	01	TYPE_TEST_NUMB		000012C4	R	01					
STEP_KEY	=	0000001E			TYPE_TU58_ERROR		00000E50	R	01					
STOP_CLOCK	=	00000004			UBI	=	00000006							
STROBE_CMI	=	00000040			UBI_NAME		000000E7	R	01					
ST_KEY	=	0000001E			UDP	=	0000001C							
SUB_TEST		00001E94	R	01	UDP_NAME		00000A4C	RG	01					
SUB_TEST_MSG		000005AC	R	01	USER_PASS_CNT		0000038E	R	01					

! Psect synopsis !

<u>PSECT name</u>	<u>Allocation</u>	<u>PSECT No.</u>	<u>Attributes</u>										
. ABS .	00000000 (0.)	00 (0.)	NOPIC	USR	CON	ABS	LCL	NOSHR	NOEXE	NORD	NOWRT	NOVEC	BYTE
. BLANK .	00002C78 (11384.)	01 (1.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE

 ! Symbol Cross Reference !

SYMBOL	VALUE	DEFINITION	REFERENCES...
\$CONTROL_C_CODE	=0000000A	863 (4)	2746 (13) 4214 (34)
\$FILE_NOT_FOUND	=00000009	863 (4)	2521 (9)
\$PSW	=00000006	856 (3)	1384 (5) 1395 (5) 1402 (5) 1408 (5) 1429 (5) 2695 (12) 2703 (12) 2800 (14) 2848 (14) 3043 (18) 3055 (18) 3058 (18) 3063 (18) 3103 (19) 3114 (19) 3117 (19) 3122 (19) 3213 (20) 3224 (20) 3225 (20) 3248 (20) 3257 (20) 3282 (20) 3284 (20) 3286 (20) 3297 (20) 3337 (20) 3339 (20) 3341 (20) 3374 (20) 3375 (20) 3380 (20) 3406 (20) 3462 (21) 3466 (21) 3467 (21) 3482 (21) 3485 (21) 3643 (23) 3654 (23) 3655 (23) 3709 (23) 3717 (23) 4007 (28) 4013 (28) 4038 (29) 4044 (29) 4127 (32) 4130 (32) 4649 (39) 4653 (39) 4655 (39) 4658 (39) 4666 (39) 4676 (39) 6297 (54) 6299 (54) 6309 (54) 6319 (54) 6325 (54) 6507 (56) 6569 (57) 6577 (57) 6580 (57) 7992 (70) 9002 (87) 9020 (87) 9022 (87) 9024 (87) 9073 (87) 9093 (87) 9095 (87) 9098 (87) 9213 (89) 9218 (89) 9285 (89) 9302 (89) 9350 (89) 9625 (94) 9632 (94) 9634 (94) 9658 (94) 9660 (94) 9663 (94) 9840 (96) 9897 (98)
\$SP	=00000006	856 (3)	1375 (5) 1384 (5) 1386 (5) 1395 (5) 1402 (5) 1408 (5) 1429 (5) 2695 (12) 2703 (12) 2800 (14) 2848 (14) 3043 (18) 3055 (18) 3058 (18) 3063 (18) 3103 (19) 3114 (19) 3117 (19) 3122 (19) 3213 (20) 3224 (20) 3225 (20) 3248 (20) 3257 (20) 3282 (20) 3284 (20) 3286 (20) 3297 (20) 3337 (20) 3339 (20) 3341 (20) 3374 (20) 3375 (20) 3380 (20) 3406 (20) 3462 (21) 3466 (21) 3467 (21) 3482 (21) 3485 (21) 3643 (23) 3654 (23) 3655 (23) 3709 (23) 3717 (23) 4007 (28) 4013 (28) 4038 (29) 4044 (29) 4127 (32) 4130 (32) 4649 (39) 4653 (39) 4655 (39) 4658 (39) 4666 (39) 4676 (39) 6297 (54) 6299 (54) 6309 (54) 6319 (54) 6325 (54) 6507 (56) 6569 (57) 6577 (57) 6580 (57) 7992 (70) 9002 (87) 9020 (87) 9022 (87) 9024 (87) 9073 (87) 9093 (87) 9095 (87) 9098 (87) 9213 (89) 9218 (89) 9285 (89) 9302 (89) 9350 (89) 9625 (94) 9632 (94) 9634 (94) 9658 (94) 9660 (94) 9663 (94) 9840 (96) 9897 (98)
A	=00000007	856 (3)	1394 (5) 1401 (5) 1409 (5) 1414 (5) 1423 (5) 2271 (7) 2277 (7) 2280 (7) 2283 (7) 2288 (7) 2294 (7) 2303 (7) 2309 (7) 2318 (7) 2339 (7) 2380 (7) 2383 (7) 2387 (7) 2467 (8) 2570 (10) 2572 (10) 2628 (11) 2706 (12) 2805 (14) 2821 (14) 2833 (14) 2892 (15) 2897 (15) 2899 (15) 2941 (16) 2946 (16) 2948 (16) 2949 (16) 2951 (16) 3000 (17) 3042 (18) 3050 (18) 3061 (18) 3064 (18) 3067 (18) 3102 (19) 3109 (19) 3120 (19) 3123 (19) 3126 (19) 3207 (20) 3212 (20) 3217 (20) 3218 (20) 3223 (20)

3228	(20)	3243	(20)	3249	(20)	3256	(20)	3260	(20)
3262	(20)	3269	(20)	3278	(20)	3280	(20)	3283	(20)
3292	(20)	3305	(20)	3307	(20)	3308	(20)	3313	(20)
3317	(20)	3324	(20)	3333	(20)	3338	(20)	3349	(20)
3350	(20)	3354	(20)	3361	(20)	3363	(20)	3364	(20)
3366	(20)	3369	(20)	3370	(20)	3372	(20)	3373	(20)
3382	(20)	3384	(20)	3400	(20)	3401	(20)	3417	(20)
3419	(20)	3420	(20)	3422	(20)	3524	(22)	3526	(22)
3528	(22)	3606	(23)	3619	(23)	3629	(23)	3642	(23)
3645	(23)	3656	(23)	3678	(23)	3685	(23)	3686	(23)
3688	(23)	3699	(23)	3700	(23)	3711	(23)	3718	(23)
3780	(24)	3796	(24)	3799	(24)	3801	(24)	3802	(24)
3805	(24)	3812	(24)	3831	(24)	3833	(24)	3842	(24)
3844	(24)	3857	(24)	3862	(24)	3971	(27)	4009	(28)
4011	(28)	4012	(28)	4040	(29)	4042	(29)	4043	(29)
4067	(30)	4099	(31)	4119	(32)	4128	(32)	4156	(33)
4157	(33)	4205	(34)	4219	(34)	4239	(34)	4247	(34)
4250	(34)	4266	(34)	4269	(34)	4319	(35)	4321	(35)
4337	(35)	4339	(35)	4344	(35)	4349	(35)	4356	(35)
4433	(36)	4446	(36)	4514	(37)	4523	(37)	4617	(39)
4628	(39)	4675	(39)	4690	(39)	4769	(40)	4772	(40)
4775	(40)	4777	(40)	4782	(40)	4838	(40)	4839	(40)
4843	(40)	4847	(40)	4848	(40)	4849	(40)	4851	(40)
4853	(40)	4859	(40)	4862	(40)	4869	(40)	4880	(40)
4989	(41)	4994	(41)	4999	(41)	5152	(45)	5170	(45)
5174	(45)	5183	(45)	5196	(45)	5202	(45)	5205	(45)
5206	(45)	5212	(45)	5215	(45)	5224	(45)	5241	(45)
5243	(45)	5248	(45)	5251	(45)	5277	(45)	5285	(45)
5289	(45)	5350	(46)	5375	(46)	5380	(46)	5384	(46)
5393	(46)	5400	(46)	5402	(46)	5413	(46)	5417	(46)
5480	(47)	5487	(47)	5521	(47)	5525	(47)	5532	(47)
5536	(47)	5544	(47)	5550	(47)	5556	(47)	5562	(47)
5568	(47)	5575	(47)	5576	(47)	5582	(47)	5586	(47)
5589	(47)	5591	(47)	5592	(47)	5601	(47)	5602	(47)
5617	(47)	5618	(47)	5627	(47)	5628	(47)	5716	(48)
5810	(49)	5814	(49)	5828	(49)	5831	(49)	5832	(49)
5836	(49)	5852	(49)	5853	(49)	5855	(49)	5856	(49)
5917	(50)	5928	(50)	5930	(50)	6002	(51)	6021	(51)
6041	(51)	6119	(52)	6174	(53)	6183	(53)	6191	(53)
6199	(53)	6200	(53)	6204	(53)	6205	(53)	6217	(53)
6220	(53)	6274	(54)	6276	(54)	6283	(54)	6286	(54)
6314	(54)	6342	(54)	6343	(54)	6345	(54)	6426	(56)
6448	(56)	6471	(56)	6477	(56)	6495	(56)	6506	(56)
6563	(57)	6570	(57)	6576	(57)	6582	(57)	6626	(58)
6628	(58)	6639	(58)	6640	(58)	6646	(58)	6647	(58)
6662	(58)	6665	(58)	6780	(60)	6857	(61)	6917	(62)
6919	(62)	6935	(62)	7007	(63)	7009	(63)	7038	(63)
7052	(63)	7065	(63)	7124	(64)	7126	(64)	7150	(64)
7152	(64)	7199	(64)	7200	(64)	7311	(65)	7314	(65)
7448	(66)	7451	(66)	7455	(66)	7461	(66)	7464	(66)
7472	(66)	7475	(66)	7495	(66)	7497	(66)	7502	(66)
7504	(66)	7505	(66)	7507	(66)	7525	(66)	7527	(66)
7529	(66)	7536	(66)	7539	(66)	7551	(66)	7575	(66)
7577	(66)	7611	(67)	7622	(67)	7623	(67)	7682	(67)
7691	(67)	7693	(67)	7711	(67)	7714	(67)	7733	(67)

				7735	(67)	7737	(67)	7749	(67)	7752	(67)	7754	(67)
				7756	(67)	7764	(67)	7771	(67)	7773	(67)	7781	(67)
				7783	(67)	7791	(67)	7824	(68)	7859	(69)	7922	(70)
				7926	(70)	7951	(70)	7954	(70)	7956	(70)	7971	(70)
				7974	(70)	7976	(70)	8034	(71)	8042	(71)	8075	(71)
				8079	(71)	8242	(76)	8243	(76)	8244	(76)	8245	(76)
				8249	(76)	8254	(76)	8271	(76)	8275	(76)	8281	(76)
				8287	(76)	8297	(76)	8298	(76)	8336	(76)	8339	(76)
				8342	(76)	8345	(76)	8347	(76)	8348	(76)	8349	(76)
				8420	(78)	8424	(78)	8426	(78)	8447	(79)	8448	(79)
				8449	(79)	8457	(79)	8498	(80)	8502	(80)	8504	(80)
				8552	(81)	8558	(81)	8576	(81)	8579	(81)	8628	(82)
				8670	(83)	8680	(83)	8723	(84)	8733	(84)	8781	(85)
				8784	(85)	8785	(85)	8836	(86)	8840	(86)	8850	(86)
				8930	(87)	8935	(87)	8941	(87)	8943	(87)	8945	(87)
				8952	(87)	8954	(87)	8966	(87)	8976	(87)	8986	(87)
				8999	(87)	9000	(87)	9003	(87)	9018	(87)	9021	(87)
				9037	(87)	9062	(87)	9063	(87)	9070	(87)	9071	(87)
				9078	(87)	9082	(87)	9094	(87)	9150	(88)	9222	(89)
				9227	(89)	9236	(89)	9238	(89)	9245	(89)	9247	(89)
				9253	(89)	9294	(89)	9308	(89)	9312	(89)	9313	(89)
				9315	(89)	9325	(89)	9329	(89)	9343	(89)	9351	(89)
				9354	(89)	9402	(90)	9404	(90)	9417	(90)	9419	(90)
				9420	(90)	9421	(90)	9422	(90)	9425	(90)	9437	(90)
				9439	(90)	9450	(90)	9452	(90)	9453	(90)	9454	(90)
				9455	(90)	9458	(90)	9482	(91)	9483	(91)	9520	(92)
				9521	(92)	9525	(92)	9529	(92)	9532	(92)	9533	(92)
				9535	(92)	9539	(92)	9636	(94)	9645	(94)	9650	(94)
				9659	(94)	9708	(95)	9789	(95)	9793	(95)	9794	(95)
				9796	(95)	9797	(95)	9799	(95)	9801	(95)	9839	(96)
				9843	(96)	9845	(96)	9846	(96)	9866	(97)	9896	(98)
				9900	(98)	9902	(98)	9903	(98)				
ACV	=0000000F	863	(4)										
AC_LOW	=00000080	863	(4)										
ADD	=00000019	863	(4)										
ADDR_MATCH_HI	=00007405	863	(4)	3527	(22)	3529	(22)	3536	(22)	5286	(45)	5381	(46)
ADDR_MATCH_LO	=00007404	863	(4)	3895	(25)	3918	(26)	3970	(27)	3984	(27)	5290	(45)
				5385	(46)	6790	(60)	863	(4)				
ADD_NAME	00000963-R	2055	(6)	1993	(6)								
ADK	=0000000B	863	(4)										
ALK	=00000000	863	(4)										
ALP	=00000016	863	(4)	9637	(94)	9705	(95)						
ALP_NAME	00000713-R	2006	(6)	1990	(6)								
ARG_LIST	000003BE-R	1803	(6)	3800	(24)	3812	(24)	3862	(24)	5827	(49)	5905	(50)
				5910	(50)	5919	(50)	5927	(50)	5929	(50)	6075	(51)
				6169	(53)	6171	(53)	6182	(53)	6184	(53)	6190	(53)
				6192	(53)	6266	(54)	6340	(54)	6421	(56)	6434	(56)
				6461	(56)	6470	(56)	6544	(57)	6546	(57)	6551	(57)
				6562	(57)	6564	(57)	6627	(58)	6653	(58)	6656	(58)
				6671	(58)	6678	(58)	6686	(58)	6730	(59)	6774	(60)
				6779	(60)	6824	(61)	6829	(61)	6834	(61)	6843	(61)
				6845	(61)	6847	(61)	6856	(61)	6909	(62)	6910	(62)
				6915	(62)	6921	(62)	6923	(62)	6928	(62)	6933	(62)
				6936	(62)	6999	(63)	7000	(63)	7005	(63)	7011	(63)
				7013	(63)	7018	(63)	7023	(63)	7025	(63)	7027	(63)

				7037	(63)	7051	(63)	7063	(63)	7066	(63)	7140	(64)
				7148	(64)	7297	(65)	7424	(66)	7454	(66)	7535	(66)
				7552	(66)	7715	(67)	7906	(70)	7908	(70)	7913	(70)
				7921	(70)	7938	(70)	7950	(70)	7958	(70)	7970	(70)
				7979	(70)	7991	(70)	7993	(70)	8030	(71)	8033	(71)
				8039	(71)	8044	(71)	8057	(71)	8069	(71)	8106	(72)
				8138	(73)	8140	(73)	8387	(77)	8416	(78)	8417	(78)
A_REG_BYTE_0	=00007410	863	(4)	863	(4)								
A_REG_BYTE_1	=00007411	863	(4)										
A_REG_BYTE_2	=00007412	863	(4)										
A_REG_BYTE_3	=00007413	863	(4)										
B	=00000000	856	(3)	1375	(5)	1384	(5)	1386	(5)	1387	(5)	1388	(5)
				1395	(5)	1396	(5)	1397	(5)	1398	(5)	1400	(5)
				1402	(5)	1404	(5)	1407	(5)	1408	(5)	1429	(5)
				2277	(7)	2280	(7)	2288	(7)	2294	(7)	2320	(7)
				2383	(7)	2385	(7)	2468	(8)	2469	(8)	2630	(11)
				2695	(12)	2703	(12)	2800	(14)	2805	(14)	2817	(14)
				2819	(14)	2820	(14)	2822	(14)	2824	(14)	2825	(14)
				2826	(14)	2827	(14)	2834	(14)	2835	(14)	2841	(14)
				2842	(14)	2848	(14)	2892	(15)	2900	(15)	2941	(16)
				2950	(16)	2951	(16)	3043	(18)	3050	(18)	3052	(18)
				3055	(18)	3058	(18)	3063	(18)	3103	(19)	3109	(19)
				3111	(19)	3114	(19)	3117	(19)	3122	(19)	3213	(20)
				3216	(20)	3223	(20)	3224	(20)	3225	(20)	3228	(20)
				3229	(20)	3231	(20)	3236	(20)	3238	(20)	3242	(20)
				3247	(20)	3248	(20)	3251	(20)	3254	(20)	3257	(20)
				3263	(20)	3270	(20)	3271	(20)	3272	(20)	3278	(20)
				3282	(20)	3284	(20)	3286	(20)	3287	(20)	3291	(20)
				3296	(20)	3297	(20)	3298	(20)	3308	(20)	3317	(20)
				3318	(20)	3322	(20)	3325	(20)	3326	(20)	3327	(20)
				3328	(20)	3333	(20)	3337	(20)	3339	(20)	3341	(20)
				3342	(20)	3348	(20)	3350	(20)	3353	(20)	3354	(20)
				3359	(20)	3367	(20)	3373	(20)	3374	(20)	3375	(20)
				3378	(20)	3380	(20)	3384	(20)	3386	(20)	3394	(20)
				3400	(20)	3404	(20)	3406	(20)	3412	(20)	3413	(20)
				3414	(20)	3415	(20)	3423	(20)	3425	(20)	3462	(21)
				3466	(21)	3467	(21)	3473	(21)	3475	(21)	3476	(21)
				3477	(21)	3478	(21)	3479	(21)	3481	(21)	3482	(21)
				3485	(21)	3520	(22)	3525	(22)	3530	(22)	3605	(23)
				3607	(23)	3608	(23)	3611	(23)	3612	(23)	3618	(23)
				3626	(23)	3631	(23)	3633	(23)	3641	(23)	3643	(23)
				3644	(23)	3647	(23)	3648	(23)	3654	(23)	3655	(23)
				3678	(23)	3681	(23)	3683	(23)	3684	(23)	3690	(23)
				3691	(23)	3693	(23)	3694	(23)	3706	(23)	3707	(23)
				3709	(23)	3710	(23)	3717	(23)	3720	(23)	3722	(23)
				3723	(23)	3801	(24)	3829	(24)	3834	(24)	3836	(24)
				3967	(27)	3985	(27)	4004	(28)	4005	(28)	4006	(28)
				4007	(28)	4012	(28)	4013	(28)	4014	(28)	4015	(28)
				4016	(28)	4035	(29)	4036	(29)	4037	(29)	4038	(29)
				4043	(29)	4044	(29)	4045	(29)	4046	(29)	4047	(29)
				4121	(32)	4125	(32)	4126	(32)	4127	(32)	4130	(32)
				4131	(32)	4132	(32)	4133	(32)	4134	(32)	4135	(32)
				4153	(33)	4155	(33)	4158	(33)	4160	(33)	4241	(34)
				4243	(34)	4246	(34)	4248	(34)	4249	(34)	4251	(34)
				4253	(34)	4281	(34)	4312	(35)	4313	(35)	4334	(35)

4355	(35)	4375	(35)	4377	(35)	4432	(36)	4445	(36)
4449	(36)	4451	(36)	4461	(36)	4462	(36)	4469	(36)
4518	(37)	4526	(37)	4568	(38)	4571	(38)	4573	(38)
4616	(39)	4618	(39)	4619	(39)	4620	(39)	4621	(39)
4622	(39)	4623	(39)	4624	(39)	4633	(39)	4649	(39)
4652	(39)	4653	(39)	4655	(39)	4656	(39)	4658	(39)
4659	(39)	4660	(39)	4666	(39)	4672	(39)	4673	(39)
4676	(39)	4678	(39)	4685	(39)	4686	(39)	4689	(39)
4691	(39)	4692	(39)	4696	(39)	4777	(40)	4784	(40)
4842	(40)	4845	(40)	4858	(40)	4861	(40)	4879	(40)
4882	(40)	4948	(41)	4988	(41)	4990	(41)	4991	(41)
4992	(41)	4995	(41)	4996	(41)	4997	(41)	4999	(41)
5002	(41)	5003	(41)	5005	(41)	5006	(41)	5151	(45)
5154	(45)	5169	(45)	5172	(45)	5173	(45)	5185	(45)
5210	(45)	5211	(45)	5214	(45)	5217	(45)	5226	(45)
5246	(45)	5247	(45)	5250	(45)	5287	(45)	5284	(45)
5288	(45)	5348	(46)	5352	(46)	5378	(46)	5379	(46)
5383	(46)	5395	(46)	5412	(46)	5415	(46)	5416	(46)
5478	(47)	5484	(47)	5485	(47)	5489	(47)	5524	(47)
5535	(47)	5576	(47)	5585	(47)	5592	(47)	5594	(47)
5602	(47)	5610	(47)	5618	(47)	5670	(47)	5628	(47)
5630	(47)	5637	(47)	5800	(49)	5801	(49)	5804	(49)
5826	(49)	5832	(49)	5833	(49)	5834	(49)	5837	(49)
5838	(49)	5839	(49)	5850	(49)	5851	(49)	5854	(49)
5857	(49)	5858	(49)	5912	(50)	5920	(50)	5923	(50)
5928	(50)	5930	(50)	6021	(51)	6023	(51)	6041	(51)
6043	(51)	6172	(53)	6173	(53)	6175	(53)	6176	(53)
6177	(53)	6183	(53)	6185	(53)	6191	(53)	6193	(53)
6215	(53)	6218	(53)	6268	(54)	6269	(54)	6270	(54)
6271	(54)	6273	(54)	6279	(54)	6280	(54)	6281	(54)
6282	(54)	6284	(54)	6285	(54)	6287	(54)	6291	(54)
6292	(54)	6293	(54)	6294	(54)	6295	(54)	6297	(54)
6299	(54)	6303	(54)	6304	(54)	6305	(54)	6306	(54)
6307	(54)	6309	(54)	6313	(54)	6319	(54)	6325	(54)
6330	(54)	6331	(54)	6333	(54)	6334	(54)	6344	(54)
6429	(56)	6438	(56)	6441	(56)	6442	(56)	6446	(56)
6447	(56)	6449	(56)	6450	(56)	6451	(56)	6453	(56)
6454	(56)	6455	(56)	6456	(56)	6460	(56)	6464	(56)
6465	(56)	6476	(56)	6478	(56)	6479	(56)	6480	(56)
6481	(56)	6492	(56)	6493	(56)	6497	(56)	6498	(56)
6499	(56)	6500	(56)	6502	(56)	6504	(56)	6507	(56)
6552	(57)	6555	(57)	6563	(57)	6565	(57)	6569	(57)
6577	(57)	6580	(57)	6581	(57)	6583	(57)	6584	(57)
6626	(58)	6639	(58)	6646	(58)	6655	(58)	6661	(58)
6666	(58)	6687	(58)	6835	(61)	6838	(61)	6839	(61)
6848	(61)	6851	(61)	6855	(61)	6857	(61)	6858	(61)
6860	(61)	6861	(61)	6862	(61)	6863	(61)	6919	(62)
6924	(62)	6927	(62)	6935	(62)	7009	(63)	7014	(63)
7017	(63)	7028	(63)	7031	(63)	7035	(63)	7038	(63)
7039	(63)	7041	(63)	7042	(63)	7043	(63)	7044	(63)
7049	(63)	7052	(63)	7053	(63)	7055	(63)	7056	(63)
7057	(63)	7058	(63)	7065	(63)	7143	(64)	7147	(64)
7152	(64)	7154	(64)	7164	(64)	7165	(64)	7170	(64)
7171	(64)	7177	(64)	7178	(64)	7197	(64)	7199	(64)
7311	(65)	7314	(65)	7324	(65)	7448	(66)	7451	(66)

7497	(66)	7504	(66)	7505	(66)	7539	(66)	7551	(66)
7553	(66)	7622	(67)	7627	(67)	7714	(67)	7716	(67)
7743	(67)	7746	(67)	7747	(67)	7767	(67)	7768	(67)
7769	(67)	7776	(67)	7777	(67)	7778	(67)	7786	(67)
7787	(67)	7788	(67)	7824	(68)	7826	(68)	7861	(69)
7914	(70)	7917	(70)	7928	(70)	7929	(70)	7930	(70)
7937	(70)	7941	(70)	7949	(70)	7957	(70)	7960	(70)
7961	(70)	7964	(70)	7966	(70)	7977	(70)	7978	(70)
7981	(70)	7982	(70)	7985	(70)	7987	(70)	7989	(70)
7990	(70)	7992	(70)	8032	(71)	8045	(71)	8048	(71)
8050	(71)	8060	(71)	8062	(71)	8068	(71)	8074	(71)
8076	(71)	8077	(71)	8078	(71)	8108	(72)	8110	(72)
8111	(72)	8143	(73)	8145	(73)	8146	(73)	8244	(76)
8248	(76)	8271	(76)	8275	(76)	8287	(76)	8297	(76)
8303	(76)	8308	(76)	8310	(76)	8313	(76)	8314	(76)
8316	(76)	8317	(76)	8320	(76)	8321	(76)	8322	(76)
8325	(76)	8326	(76)	8327	(76)	8328	(76)	8334	(76)
8337	(76)	8340	(76)	8343	(76)	8390	(77)	8454	(79)
8456	(79)	8500	(80)	8503	(80)	8505	(80)	8550	(81)
8551	(81)	8560	(81)	8565	(81)	8573	(81)	8574	(81)
8575	(81)	8576	(81)	8630	(82)	8670	(83)	8675	(83)
8679	(83)	8681	(83)	8682	(83)	8683	(83)	8684	(83)
8723	(84)	8728	(84)	8732	(84)	8734	(84)	8735	(84)
8736	(84)	8737	(84)	8784	(85)	8787	(85)	8790	(85)
8791	(85)	8802	(85)	8805	(85)	8806	(85)	8915	(87)
8924	(87)	8927	(87)	8928	(87)	8969	(87)	8970	(87)
8971	(87)	8979	(87)	8980	(87)	8981	(87)	8989	(87)
8990	(87)	8991	(87)	8998	(87)	9002	(87)	9005	(87)
9016	(87)	9020	(87)	9022	(87)	9024	(87)	9034	(87)
9037	(87)	9038	(87)	9040	(87)	9041	(87)	9044	(87)
9045	(87)	9047	(87)	9048	(87)	9051	(87)	9052	(87)
9053	(87)	9054	(87)	9056	(87)	9061	(87)	9069	(87)
9073	(87)	9074	(87)	9075	(87)	9077	(87)	9079	(87)
9083	(87)	9084	(87)	9086	(87)	9092	(87)	9093	(87)
9095	(87)	9096	(87)	9098	(87)	9144	(88)	9145	(88)
9148	(88)	9212	(89)	9213	(89)	9215	(89)	9218	(89)
9219	(89)	9220	(89)	9221	(89)	9225	(89)	9226	(89)
9228	(89)	9229	(89)	9230	(89)	9235	(89)	9237	(89)
9247	(89)	9250	(89)	9252	(89)	9273	(89)	9281	(89)
9282	(89)	9283	(89)	9285	(89)	9301	(89)	9302	(89)
9303	(89)	9307	(89)	9308	(89)	9309	(89)	9310	(89)
9311	(89)	9312	(89)	9315	(89)	9316	(89)	9318	(89)
9319	(89)	9320	(89)	9327	(89)	9335	(89)	9336	(89)
9342	(89)	9344	(89)	9345	(89)	9346	(89)	9349	(89)
9350	(89)	9392	(90)	9397	(90)	9399	(90)	9402	(90)
9404	(90)	9405	(90)	9406	(90)	9412	(90)	9413	(90)
9416	(90)	9420	(90)	9429	(90)	9430	(90)	9432	(90)
9434	(90)	9437	(90)	9439	(90)	9440	(90)	9441	(90)
9447	(90)	9448	(90)	9449	(90)	9453	(90)	9481	(91)
9482	(91)	9484	(91)	9488	(91)	9536	(92)	9584	(93)
9625	(94)	9626	(94)	9627	(94)	9630	(94)	9631	(94)
9632	(94)	9634	(94)	9635	(94)	9641	(94)	9644	(94)
9651	(94)	9653	(94)	9657	(94)	9658	(94)	9660	(94)
9661	(94)	9663	(94)	9664	(94)	9666	(94)	9667	(94)
9710	(95)	9715	(95)	9719	(95)	9720	(95)	9724	(95)

				9725 (95)	9727 (95)	9729 (95)	9731 (95)	9732 (95)		
				9733 (95)	9734 (95)	9735 (95)	9737 (95)	9738 (95)		
				9739 (95)	9740 (95)	9741 (95)	9743 (95)	9744 (95)		
				9745 (95)	9746 (95)	9747 (95)	9749 (95)	9750 (95)		
				9751 (95)	9752 (95)	9753 (95)	9754 (95)	9762 (95)		
				9763 (95)	9766 (95)	9767 (95)	9769 (95)	9776 (95)		
				9777 (95)	9778 (95)	9779 (95)	9780 (95)	9782 (95)		
				9787 (95)	9788 (95)	9798 (95)	9800 (95)	9802 (95)		
				9803 (95)	9805 (95)	9806 (95)	9808 (95)	9809 (95)		
				9837 (96)	9840 (96)	9846 (96)	9863 (97)	9865 (97)		
				9894 (98)	9897 (98)	9903 (98)				
BADD1	=00007426	863	(4)							
BAD_DATA	00000443-R	1870	(6)	7050 (63)	7134 (64)	7135 (64)	8671 (83)	8677 (83)		
				8678 (83)	8783 (85)	8958 (87)	9049 (87)	9524 (92)		
				9537 (92)	9540 (92)	9727 (95)				
BEGIN_LOOP	00001CE0-R	6168	(53)	1808 (6)						
BELL	00000592-R	1930	(6)	8845 (86)						
BELL_COUNT	00000591-R	1928	(6)	4206 (34)	8835 (86)	8841 (86)	8849 (86)	8851 (86)		
BELL_FLAG	=00000008	863	(4)	4755 (40)	7292 (65)					
BELL_KEY	=0000000E	863	(4)							
BIT_COUNT	00000C9B-R	2183	(6)	9239 (89)	9328 (89)	9330 (89)				
BIT_NMB_TO_MSK	00002A6E-R	9480	(91)	5175 (45)	5418 (46)	5712 (48)	9274 (89)	9293 (89)		
BIT_SLICE	00002B53-R	9701	(95)	9642 (94)						
BIT_SLICE_TBL	000006F5-R	1990	(6)	9706 (95)						
BIT_SLICE_TEMP	000002C4-R	1685	(6)	9726 (95)						
BREAK_ACTION	000015A1-R	4279	(34)	1705 (6)						
BUFF_ADDR_LOW	=00007426	863	(4)	863 (4)						
BURST_CLOCK	000020B5-R	7423	(66)	1805 (6)	7587 (66)					
BURST_STOP_FLAG	=00000080	863	(4)	7433 (66)	7479 (66)	7616 (67)				
BURST_TEMP	00000451-R	1889	(6)	7496 (66)	7503 (66)	7507 (66)	7528 (66)	7529 (66)		
				7604 (67)	7647 (67)	7692 (67)	7693 (67)	7734 (67)		
				7735 (67)						
BYTE	=00000001	863	(4)							
C	=0000C001	856	(3)	1399 (5)	2467 (8)	2818 (14)	2823 (14)	2828 (14)		
				2836 (14)	3255 (20)	3268 (20)	3273 (20)	3476 (21)		
				3824 (24)	3832 (24)	3833 (24)	4154 (33)	4156 (33)		
				4159 (33)	4776 (40)	5150 (45)	5191 (45)	5196 (45)		
				5349 (46)	5400 (46)	5402 (46)	5404 (46)	5486 (47)		
				5716 (48)	5825 (49)	5913 (50)	6505 (56)	6576 (57)		
				6578 (57)	6663 (58)	6917 (62)	7007 (63)	7150 (64)		
				7159 (64)	7160 (64)	7611 (67)	7623 (67)	7925 (70)		
				7931 (70)	7988 (70)	8109 (72)	8144 (73)	8242 (76)		
				8499 (80)	8551 (81)	8572 (81)	8579 (81)	8781 (85)		
				8792 (85)	8807 (85)	9082 (87)	9236 (89)	9294 (89)		
				9314 (89)	9338 (89)	9347 (89)	9398 (90)	9407 (90)		
				9419 (90)	9422 (90)	9433 (90)	9442 (90)	9452 (90)		
				9455 (90)	9521 (92)	9525 (92)	9539 (92)	9541 (92)		
				9645 (94)	9649 (94)	9650 (94)	9765 (95)	9771 (95)		
				9799 (95)	9838 (96)	9895 (98)				
CACHE_ENA_ADR	00000275-R	1658	(6)	3305 (20)						
CACHE_ENA_DATA	00000280-R	1661	(6)	3307 (20)						
CACHE_WRITE	0000028B-R	1663	(6)	3313 (20)						
CAK	=0000000A	863	(4)							
CALC_PARITY	00002A1C-R	9391	(90)	9356 (89)						
CCC	=00000004	863	(4)							

CON	=00000011	863	(4)										
CONTINUE_ACTION	0000183A-R	5055	(42)	1698	(6)								
CONTINUE_FLAG	=00000002	863	(4)	4203	(34)	4267	(34)	5057	(42)	5103	(44)		
CONTROL_C_FLAG	=00000001	863	(4)	2750	(13)	4203	(34)	5997	(51)	7280	(65)	7605	(67)
				7608	(67)	7640	(67)						
CONT_FLAG	=00000020	863	(4)	4761	(40)	4830	(40)	6049	(51)	6063	(51)		
CONT_KEY	=0000001C	863	(4)	4794	(40)								
CONT_SUB_ACTION	0000175A-R	4828	(40)	4795	(40)								
CPU_INIT	00001459-R	4065	(30)	2997	(17)	6111	(52)	7828	(68)				
CPU_RUN	=00000010	863	(4)										
CRLF	00000CEE-R	2212	(6)	2269	(7)	2362	(7)	2378	(7)	2381	(7)	2568	(10)
				3650	(23)	3817	(24)	3856	(24)	4137	(32)	4209	(34)
				4217	(34)	7490	(66)	7680	(67)	7726	(67)	7738	(67)
				8260	(76)	8262	(76)	8442	(79)	8460	(79)	8916	(87)
				8946	(87)	8947	(87)	8957	(87)	9043	(87)	9090	(87)
				9099	(87)	9139	(88)	9628	(94)	9655	(94)	9665	(94)
				9867	(97)								
CSAMR	=00007404	863	(4)										
CSBUF	=00007424	863	(4)										
CSTRP	=00007422	863	(4)										
CS_ADD_BUFF_HGH	=00007425	863	(4)										
CS_ADD_BUFF_LOW	=00007424	863	(4)	863	(4)								
CS_ADD_TRAP_HGH	=00007423	863	(4)	7514	(66)	7652	(67)						
CS_ADD_TRAP_LOW	=00007422	863	(4)	7524	(66)	7751	(67)	863	(4)				
CURRENT_BIT	00000C9A-R	2182	(6)	9211	(89)	9217	(89)	9223	(89)	9286	(89)	9324	(89)
				9326	(89)								
CURRENT_PC	000003B8-R	1798	(6)	5105	(44)	5824	(49)	7320	(65)	7742	(67)	8923	(87)
CURRENT_REC_NMB	0000013B-R	1512	(6)	2302	(7)	2331	(7)	2465	(8)	2470	(8)	2518	(9)
				3474	(21)	3480	(21)	3604	(23)	3724	(23)		
CYCLE_FLAG	=00000008	863	(4)	5229	(45)	5255	(45)	7485	(66)	7573	(66)	7584	(66)
CYCLE_KEY	=00000020	863	(4)	5230	(45)								
C_INIT_TEST_PC	000001B7-R	1607	(6)	3368	(20)	3424	(20)	3835	(24)	3837	(24)	5799	(49)
				7740	(67)	8921	(87)	9143	(88)				
D	=00000002	856	(3)	1375	(5)	1384	(5)	1386	(5)	1387	(5)	1388	(5)
				1395	(5)	1396	(5)	1397	(5)	1400	(5)	1402	(5)
				1404	(5)	1406	(5)	1407	(5)	1408	(5)	1429	(5)
				2315	(7)	2320	(7)	2630	(11)	2695	(12)	2703	(12)
				2800	(14)	2807	(14)	2822	(14)	2826	(14)	2834	(14)
				2835	(14)	2841	(14)	2848	(14)	3043	(18)	3055	(18)
				3058	(18)	3063	(18)	3103	(19)	3114	(19)	3117	(19)
				3122	(19)	3213	(20)	3216	(20)	3224	(20)	3225	(20)
				3229	(20)	3231	(20)	3236	(20)	3238	(20)	3242	(20)
				3247	(20)	3248	(20)	3251	(20)	3252	(20)	3257	(20)
				3263	(20)	3267	(20)	3270	(20)	3271	(20)	3272	(20)
				3278	(20)	3282	(20)	3284	(20)	3286	(20)	3287	(20)
				3291	(20)	3296	(20)	3297	(20)	3298	(20)	3308	(20)
				3317	(20)	3318	(20)	3323	(20)	3325	(20)	3326	(20)
				3327	(20)	3333	(20)	3337	(20)	3339	(20)	3341	(20)
				3342	(20)	3348	(20)	3353	(20)	3359	(20)	3367	(20)
				3374	(20)	3375	(20)	3380	(20)	3394	(20)	3406	(20)
				3412	(20)	3413	(20)	3414	(20)	3415	(20)	3423	(20)
				3425	(20)	3462	(21)	3466	(21)	3467	(21)	3473	(21)
				3481	(21)	3482	(21)	3485	(21)	3525	(22)	3605	(23)
				3607	(23)	3608	(23)	3611	(23)	3612	(23)	3618	(23)
				3626	(23)	3630	(23)	3631	(23)	3633	(23)	3634	(23)

3641	(23)	3643	(23)	3644	(23)	3647	(23)	3648	(23)
3654	(23)	3655	(23)	3671	(23)	3680	(23)	3683	(23)
3684	(23)	3689	(23)	3690	(23)	3691	(23)	3693	(23)
3694	(23)	3706	(23)	3707	(23)	3709	(23)	3710	(23)
3717	(23)	3720	(23)	3722	(23)	3723	(23)	3829	(24)
3967	(27)	3985	(27)	4005	(28)	4006	(28)	4007	(28)
4012	(28)	4013	(28)	4014	(28)	4015	(28)	4036	(29)
4037	(29)	4038	(29)	4043	(29)	4044	(29)	4045	(29)
4046	(29)	4126	(32)	4127	(32)	4130	(32)	4131	(32)
4133	(32)	4134	(32)	4246	(34)	4248	(34)	4249	(34)
4254	(34)	4281	(34)	4312	(35)	4319	(35)	4321	(35)
4328	(35)	4337	(35)	4339	(35)	4355	(35)	4375	(35)
4377	(35)	4430	(36)	4443	(36)	4444	(36)	4451	(36)
4462	(36)	4469	(36)	4518	(37)	4526	(37)	4568	(38)
4570	(38)	4571	(38)	4573	(38)	4633	(39)	4649	(39)
4651	(39)	4653	(39)	4655	(39)	4658	(39)	4661	(39)
4666	(39)	4675	(39)	4676	(39)	4677	(39)	4686	(39)
4688	(39)	4691	(39)	4693	(39)	4696	(39)	4775	(40)
4784	(40)	4838	(40)	4841	(40)	4845	(40)	4861	(40)
4882	(40)	4948	(41)	4990	(41)	4991	(41)	4996	(41)
5002	(41)	5006	(41)	5154	(45)	5172	(45)	5185	(45)
5210	(45)	5211	(45)	5214	(45)	5217	(45)	5226	(45)
5229	(45)	5232	(45)	5235	(45)	5246	(45)	5247	(45)
5250	(45)	5256	(45)	5283	(45)	5284	(45)	5288	(45)
5352	(46)	5378	(46)	5379	(46)	5383	(46)	5395	(46)
5415	(46)	5478	(47)	5484	(47)	5489	(47)	5576	(47)
5592	(47)	5602	(47)	5618	(47)	5628	(47)	5800	(49)
5801	(49)	5804	(49)	5811	(49)	5826	(49)	5827	(49)
5833	(49)	5837	(49)	5838	(49)	5839	(49)	5854	(49)
5856	(49)	5912	(50)	5920	(50)	5923	(50)	6171	(53)
6173	(53)	6175	(53)	6176	(53)	6215	(53)	6218	(53)
6268	(54)	6269	(54)	6270	(54)	6271	(54)	6273	(54)
6280	(54)	6281	(54)	6282	(54)	6284	(54)	6285	(54)
6287	(54)	6291	(54)	6292	(54)	6293	(54)	6294	(54)
6295	(54)	6297	(54)	6299	(54)	6303	(54)	6304	(54)
6305	(54)	6306	(54)	6307	(54)	6309	(54)	6313	(54)
6319	(54)	6325	(54)	6330	(54)	6331	(54)	6333	(54)
6334	(54)	6344	(54)	6429	(56)	6438	(56)	6441	(56)
6442	(56)	6447	(56)	6449	(56)	6450	(56)	6453	(56)
6454	(56)	6455	(56)	6456	(56)	6460	(56)	6464	(56)
6465	(56)	6478	(56)	6479	(56)	6480	(56)	6493	(56)
6494	(56)	6497	(56)	6498	(56)	6499	(56)	6502	(56)
6507	(56)	6552	(57)	6555	(57)	6565	(57)	6569	(57)
6577	(57)	6580	(57)	6581	(57)	6583	(57)	6661	(58)
6665	(58)	6687	(58)	6835	(61)	6838	(61)	6839	(61)
6848	(61)	6851	(61)	6855	(61)	6858	(61)	6860	(61)
6861	(61)	6862	(61)	6924	(62)	6927	(62)	7014	(63)
7017	(63)	7028	(63)	7031	(63)	7035	(63)	7036	(63)
7039	(63)	7041	(63)	7042	(63)	7043	(63)	7049	(63)
7050	(63)	7053	(63)	7055	(63)	7056	(63)	7057	(63)
7143	(64)	7147	(64)	7154	(64)	7163	(64)	7164	(64)
7165	(64)	7171	(64)	7177	(64)	7197	(64)	7324	(65)
7743	(67)	7746	(67)	7747	(67)	7768	(67)	7769	(67)
7777	(67)	7778	(67)	7787	(67)	7788	(67)	7914	(70)
7917	(70)	7924	(70)	7928	(70)	7929	(70)	7930	(70)

7937	(70)	7941	(70)	7949	(70)	7957	(70)	7960	(70)
7964	(70)	7978	(70)	7981	(70)	7985	(70)	7989	(70)
7990	(70)	7992	(70)	8032	(71)	8045	(71)	8048	(71)
8050	(71)	8060	(71)	8062	(71)	8068	(71)	8069	(71)
8074	(71)	8076	(71)	8077	(71)	8078	(71)	8110	(72)
8143	(73)	8145	(73)	8248	(76)	8269	(76)	8274	(76)
8303	(76)	8308	(76)	8310	(76)	8313	(76)	8314	(76)
8316	(76)	8317	(76)	8320	(76)	8321	(76)	8322	(76)
8325	(76)	8326	(76)	8327	(76)	8334	(76)	8337	(76)
8340	(76)	8343	(76)	8390	(77)	8454	(79)	8456	(79)
8501	(80)	8503	(80)	8506	(80)	8559	(81)	8560	(81)
8565	(81)	8566	(81)	8573	(81)	8575	(81)	8671	(83)
8675	(83)	8679	(83)	8681	(83)	8682	(83)	8683	(83)
8724	(84)	8728	(84)	8732	(84)	8734	(84)	8735	(84)
8736	(84)	8782	(85)	8787	(85)	8790	(85)	8791	(85)
8802	(85)	8805	(85)	8806	(85)	8915	(87)	8924	(87)
8927	(87)	8928	(87)	8970	(87)	8971	(87)	8980	(87)
8981	(87)	8990	(87)	8991	(87)	8998	(87)	9002	(87)
9005	(87)	9016	(87)	9020	(87)	9022	(87)	9024	(87)
9034	(87)	9040	(87)	9045	(87)	9047	(87)	9052	(87)
9053	(87)	9056	(87)	9061	(87)	9069	(87)	9073	(87)
9074	(87)	9075	(87)	9077	(87)	9079	(87)	9086	(87)
9087	(87)	9092	(87)	9093	(87)	9095	(87)	9096	(87)
9098	(87)	9144	(88)	9145	(88)	9148	(88)	9212	(89)
9213	(89)	9215	(89)	9218	(89)	9219	(89)	9220	(89)
9226	(89)	9228	(89)	9229	(89)	9251	(89)	9252	(89)
9259	(89)	9273	(89)	9281	(89)	9282	(89)	9283	(89)
9285	(89)	9287	(89)	9301	(89)	9302	(89)	9303	(89)
9309	(89)	9320	(89)	9335	(89)	9336	(89)	9342	(89)
9344	(89)	9345	(89)	9346	(89)	9349	(89)	9350	(89)
9392	(90)	9396	(90)	9399	(90)	9405	(90)	9406	(90)
9412	(90)	9413	(90)	9415	(90)	9416	(90)	9429	(90)
9430	(90)	9431	(90)	9434	(90)	9440	(90)	9441	(90)
9447	(90)	9448	(90)	9449	(90)	9519	(92)	9534	(92)
9535	(92)	9536	(92)	9584	(93)	9625	(94)	9626	(94)
9627	(94)	9630	(94)	9631	(94)	9632	(94)	9634	(94)
9635	(94)	9641	(94)	9644	(94)	9657	(94)	9658	(94)
9660	(94)	9661	(94)	9663	(94)	9664	(94)	9666	(94)
9667	(94)	9709	(95)	9710	(95)	9715	(95)	9716	(95)
9720	(95)	9725	(95)	9726	(95)	9731	(95)	9733	(95)
9734	(95)	9737	(95)	9739	(95)	9740	(95)	9743	(95)
9745	(95)	9746	(95)	9749	(95)	9750	(95)	9752	(95)
9753	(95)	9754	(95)	9763	(95)	9766	(95)	9767	(95)
9769	(95)	9776	(95)	9777	(95)	9778	(95)	9779	(95)
9787	(95)	9788	(95)	9789	(95)	9797	(95)	9802	(95)
9805	(95)	9806	(95)	9808	(95)	9809	(95)	9835	(96)
9840	(96)	9846	(96)	9863	(97)	9865	(97)	9892	(98)
9897	(98)	9903	(98)						

DATA_MSG	000005B7-R	1943	(6)
DATA_TYPE	00000447-R	1871	(6)
DCSAD	=00007401	863	(4)
DCSCF	=00007403	863	(4)
DCSCR	=00007402	863	(4)
DCSDA	=00007400	863	(4)
DCS_ADDR_CLEAR	=00000002	863	(4)

7461	(66)	7475	(66)	7499	(66)	7684	(67)
------	------	------	------	------	------	------	------

ZZ-ECKAA-8.7
ECKAA
Cross reference

Cross reference

VAX-11/750 MICRO DIAGNOSTIC MONITOR

I 5
11-FEB-1986

Fiche 2 Frame 15

Sequence 266

11-FEB-1986 10:07:52 VAX/VMS Macro V04-00 Page 241
11-FEB-1986 10:06:37 [VAX750.MONITOR]ECKAA.MAR;811 (98)

DIRECT_MSG	00000076-R	1469	(6)	2361	(7)								
DIRECT_SEARCH	0000121B-R	3603	(23)	2349	(7)								
DISABLE_CMI	000013D5-R	4003	(28)	3299	(20)	6122	(52)						
DISPLAY_DATA	00002C0F-R	9862	(97)	5650	(47)	8959	(87)	8974	(87)	8984	(87)	8994	(87)
				9012	(87)	9050	(87)						
DPM	=00000000	863	(4)										
DPM_NAME	000000A5-R	1485	(6)	1470	(6)								
DUMP_EXIT	00002689-R	8459	(79)	8453	(79)								
DUMP_LOG	00002646-R	8438	(79)	1828	(6)								
DUMP_RET	00002692-R	8461	(79)	8441	(79)								
E	=00000003	856	(3)	1405	(5)	2316	(7)	3629	(23)	3632	(23)	3672	(23)
				3688	(23)	4431	(36)	4446	(36)	4459	(36)	4650	(39)
				4667	(39)	4846	(40)	4868	(40)	4869	(40)	4871	(40)
				5815	(49)	5853	(49)	5913	(50)	6662	(58)	7162	(64)
				8268	(76)	8558	(81)	8564	(81)	9085	(87)	9250	(89)
				9262	(89)	9290	(89)	9414	(90)	9531	(92)	9532	(92)
				9708	(95)	9714	(95)						
ENABLE_CMI	00001417-R	4034	(29)	2998	(17)								
ENABLE_STALL	00002520-R	8197	(75)	1832	(6)								
ENABLE_TRAP	00002517-R	8170	(74)	1827	(6)								
ENDPASS_MSG	00000584-R	1922	(6)	2379	(7)								
END_LOOP	00001D29-R	6262	(54)	1809	(6)								
END_PASS	0000242E-R	7857	(69)	1822	(6)								
END_SLICE	=00000025	863	(4)	9639	(94)								
END_TEST	00002416-R	7822	(68)	1810	(6)								
EN TRACE_REG	=00000008	863	(4)										
ERROR_FLAG	=00000010	863	(4)	6073	(51)	6941	(62)	7071	(63)	7195	(64)	7274	(65)
				7445	(66)	7540	(66)	8354	(76)	9597	(93)		
ERROR_LOG	00002529-R	8240	(76)	1829	(6)								
ERROR_LOG_ADR	=000032FC	863	(4)	8267	(76)	8346	(76)	8443	(79)				
ERROR_LOOP	00001D88-R	6371	(55)	1811	(6)								
ERROR_LOOP_ADDR	00000425-R	1852	(6)	6070	(51)	6373	(55)	7331	(65)				
ERROR_MSG	0000059B-R	1940	(6)	8917	(87)								
ERROR_PC	00000594-R	1936	(6)	5106	(44)	7321	(65)	7325	(65)				
ERR_LOG_INIT	=00000001	863	(4)	1445	(6)	4764	(40)	8272	(76)	8276	(76)	8445	(79)
				8463	(79)								
EXAMINE_ACTION	000019C1-R	5477	(47)	1703	(6)								
EXAMINE_DISPLAY	00001B79-R	5649	(47)	5640	(47)								
EXAMINE_ERR	00001B7F-R	5652	(47)	5491	(47)	5516	(47)	5523	(47)	5534	(47)	5584	(47)
EXAMINE_EX	00001B81-R	5655	(47)	5651	(47)								
EXAMINE_MA	00001A5B-R	5567	(47)	5505	(47)								
EXAMINE_MD	00001A51-R	5555	(47)	5501	(47)								
EXAMINE_MT	00001A39-R	5531	(47)	5495	(47)								
EXAMINE_PB	00001A56-R	5561	(47)	5503	(47)								
EXAMINE_PC	00001A47-R	5543	(47)	5497	(47)								
EXAMINE_PS	00001AD5-R	5599	(47)	5511	(47)								
EXAMINE_RT	00001A2B-R	5520	(47)	5493	(47)								
EXAMINE_SF	00001B41-R	5625	(47)	5515	(47)								
EXAMINE_SR	00001A89-R	5581	(47)	5509	(47)								
EXAMINE_ST	00001B13-R	5615	(47)	5513	(47)								
EXAMINE_TEMP	000003BC-R	1801	(6)	5482	(47)	5483	(47)	5605	(47)	5608	(47)	5638	(47)
				5639	(47)								
EXAMINE_VA	00001A4C-R	5549	(47)	5499	(47)								
EXAMINE_WD	00001A60-R	5573	(47)	5507	(47)								
EXP_STALL_FLAG	=00000010	863	(4)	7662	(67)	8199	(75)						

EXP_UTRAP_FLAG	=00000040	863	(4)	6012	(51)	6725	(59)	7650	(67)	8172	(74)		
EX_DISP_WH	00001B76-R	5644	(47)	5527	(47)	5539	(47)	5577	(47)				
FAC	=00000020	863	(4)										
FAC_NAME	00000B9E-R	2126	(6)	2000	(6)								
FAIL_ARRAY_MSG	000005DA-R	1948	(6)	9629	(94)								
FAIL_ARR_INDEX	000005F0-R	1949	(6)	9656	(94)								
FAIL_MOD_INDEX	00000618-R	1951	(6)	9091	(87)								
FAULT	=00000002	863	(4)	1418	(5)	2888	(15)	2903	(15)	2938	(16)	2954	(16)
				3004	(17)	4076	(30)	4976	(41)	6125	(52)	6575	(57)
				7696	(67)								
FCC	=00000015	863	(4)										
FCS	=00000021	863	(4)										
FCS_NAME	00000BB0-R	2131	(6)	2001	(6)								
FETCH_0	0000147B-R	4093	(31)	2999	(17)	6123	(52)						
FETCH_FLAG	=00000008	863	(4)	6772	(60)	7512	(66)	7570	(66)	7594	(67)		
FETCH_MIC_INSTR	00001EA6-R	6770	(60)	1821	(6)								
FEX	=0000001E	863	(4)										
FEX_NAME	00000B6A-R	2114	(6)	1998	(6)								
FFH	=00000024	863	(4)										
FFH_NAME	00000C49-R	2160	(6)	2004	(6)								
FFL	=00000023	863	(4)										
FFL_NAME	00000C0C-R	2149	(6)	2003	(6)								
FIC	=0000001F	863	(4)										
FIC_NAME	00000B89-R	2121	(6)	1999	(6)								
FILE_FOR_CMC	00000160-R	1535	(6)										
FILE_FOR_DPM	0000014E-R	1526	(6)	2314	(7)								
FILE_FOR_FPA	0000015A-R	1532	(6)										
FILE_FOR_MIC	00000154-R	1529	(6)										
FILE_FOR_RDM	00000166-R	1538	(6)										
FILE_NAME_HEAD	=0000014E-R	1525	(6)	1543	(6)	1546	(6)						
FILE_NAME_TAIL	=0000016C-R	1545	(6)	1546	(6)								
FILE_NAM_INDEX	00000063-R	1463	(6)	2310	(7)	2317	(7)	2338	(7)	2340	(7)	3609	(23)
FILE_NAM_PTR	00000139-R	1511	(6)	2325	(7)	2518	(9)						
FILL_DCS	000014D1-R	4152	(33)	1425	(5)	3237	(20)						
FIO	=0000001D	863	(4)										
FIO_NAME	00000AF1-R	2095	(6)	1997	(6)								
FLAG_KEY	=00000004	863	(4)	4949	(41)	5157	(45)	5355	(46)				
FMR	=00000022	863	(4)										
FMR_NAME	00000BED-R	2142	(6)	2002	(6)								
FPA	=00000002	863	(4)										
FPA_NAME	000000BB-R	1487	(6)	1472	(6)								
FP_BOOT_0	=00000001	863	(4)										
FP_BOOT_1	=00000002	863	(4)										
FP_START_0	=00000004	863	(4)										
FP_START_1	=00000008	863	(4)										
FQA	=00000014	863	(4)										
FRONT1	=00007420	863	(4)										
FRONT2	=00007421	863	(4)										
FRONT_PNL_1	=00007420	863	(4)	863	(4)								
FRONT_PNL_2	=00007421	863	(4)	1417	(5)	2887	(15)	2902	(15)	2937	(16)	2953	(16)
				3003	(17)	4075	(30)	4975	(41)	6116	(52)	6124	(52)
				6574	(57)	7695	(67)	863	(4)				
FRONT_PNL_LOCK	=00000080	863	(4)										
GATE_ARRAY_LST	0000062F-R	1959	(6)	9652	(94)								
GET_CMD_LINE	000014EF-R	4200	(34)	1430	(5)	2344	(7)	2398	(7)	2475	(8)	2524	(9)

			4282 (34)	5819 (49)	5998 (51)	6025 (51)	7281 (65)
			7306 (65)	7530 (66)	7555 (66)	7586 (66)	7642 (67)
			7694 (67)	7792 (67)			
GET_INDEX_VALUE	000026AF-R	8546 (81)	5906 (50)	6170 (53)	6267 (54)	6341 (54)	6422 (56)
			6545 (57)	6654 (58)	6657 (58)	6825 (61)	6844 (61)
			6916 (62)	7006 (63)	7024 (63)	7149 (64)	7765 (67)
			7774 (67)	7784 (67)	7907 (70)	7959 (70)	7980 (70)
			8031 (71)	8040 (71)	8107 (72)	8139 (73)	8967 (87)
			8977 (87)	8987 (87)			
GOOD_DATA	0000043F-R	1869 (6)	7036 (63)	7132 (64)	7133 (64)	8724 (84)	8730 (84)
			8731 (84)	8782 (85)	8952 (87)	9523 (92)	9528 (92)
			9538 (92)	9728 (95)			
H	=00000004	856 (3)	1374 (5)	1375 (5)	1384 (5)	1385 (5)	1386 (5)
			1387 (5)	1388 (5)	1393 (5)	1395 (5)	1396 (5)
			1400 (5)	1402 (5)	1404 (5)	1408 (5)	1429 (5)
			2269 (7)	2270 (7)	2304 (7)	2314 (7)	2361 (7)
			2362 (7)	2378 (7)	2379 (7)	2380 (7)	2381 (7)
			2568 (10)	2569 (10)	2570 (10)	2571 (10)	2572 (10)
			2630 (11)	2695 (12)	2703 (12)	2708 (12)	2710 (12)
			2800 (14)	2826 (14)	2835 (14)	2841 (14)	2848 (14)
			2893 (15)	2942 (16)	3043 (18)	3055 (18)	3058 (18)
			3063 (18)	3103 (19)	3114 (19)	3117 (19)	3122 (19)
			3207 (20)	3213 (20)	3216 (20)	3224 (20)	3225 (20)
			3229 (20)	3230 (20)	3231 (20)	3236 (20)	3238 (20)
			3242 (20)	3247 (20)	3248 (20)	3251 (20)	3253 (20)
			3257 (20)	3260 (20)	3262 (20)	3263 (20)	3271 (20)
			3278 (20)	3282 (20)	3284 (20)	3286 (20)	3287 (20)
			3291 (20)	3296 (20)	3297 (20)	3298 (20)	3305 (20)
			3307 (20)	3308 (20)	3313 (20)	3314 (20)	3317 (20)
			3318 (20)	3326 (20)	3333 (20)	3337 (20)	3339 (20)
			3341 (20)	3342 (20)	3348 (20)	3353 (20)	3358 (20)
			3359 (20)	3364 (20)	3366 (20)	3367 (20)	3369 (20)
			3374 (20)	3375 (20)	3380 (20)	3394 (20)	3403 (20)
			3406 (20)	3412 (20)	3413 (20)	3414 (20)	3415 (20)
			3420 (20)	3422 (20)	3423 (20)	3425 (20)	3462 (21)
			3466 (21)	3467 (21)	3473 (21)	3481 (21)	3482 (21)
			3485 (21)	3521 (22)	3525 (22)	3605 (23)	3606 (23)
			3607 (23)	3608 (23)	3611 (23)	3612 (23)	3618 (23)
			3626 (23)	3627 (23)	3633 (23)	3637 (23)	3641 (23)
			3643 (23)	3644 (23)	3647 (23)	3648 (23)	3650 (23)
			3654 (23)	3655 (23)	3661 (23)	3679 (23)	3683 (23)
			3690 (23)	3693 (23)	3694 (23)	3706 (23)	3707 (23)
			3709 (23)	3710 (23)	3717 (23)	3719 (23)	3722 (23)
			3811 (24)	3812 (24)	3813 (24)	3817 (24)	3825 (24)
			3856 (24)	3862 (24)	3863 (24)	3894 (25)	3967 (27)
			3969 (27)	3985 (27)	4006 (28)	4007 (28)	4009 (28)
			4011 (28)	4012 (28)	4013 (28)	4014 (28)	4037 (29)
			4038 (29)	4040 (29)	4042 (29)	4043 (29)	4044 (29)
			4045 (29)	4122 (32)	4126 (32)	4127 (32)	4129 (32)
			4130 (32)	4131 (32)	4133 (32)	4134 (32)	4137 (32)
			4157 (33)	4209 (34)	4210 (34)	4217 (34)	4218 (34)
			4219 (34)	4231 (34)	4240 (34)	4250 (34)	4265 (34)
			4272 (34)	4281 (34)	4309 (35)	4311 (35)	4312 (35)
			4355 (35)	4375 (35)	4451 (36)	4462 (36)	4469 (36)
			4518 (37)	4526 (37)	4568 (38)	4571 (38)	4573 (38)

4633	(39)	4649	(39)	4653	(39)	4655	(39)	4658	(39)
4666	(39)	4676	(39)	4686	(39)	4691	(39)	4696	(39)
4784	(40)	4845	(40)	4861	(40)	4882	(40)	4954	(41)
4957	(41)	4983	(41)	4987	(41)	4990	(41)	4991	(41)
4994	(41)	4996	(41)	5002	(41)	5004	(41)	5006	(41)
5154	(45)	5172	(45)	5185	(45)	5210	(45)	5211	(45)
5214	(45)	5217	(45)	5226	(45)	5246	(45)	5247	(45)
5250	(45)	5283	(45)	5284	(45)	5288	(45)	5352	(46)
5378	(46)	5379	(46)	5383	(46)	5395	(46)	5415	(46)
5480	(47)	5481	(47)	5484	(47)	5489	(47)	5575	(47)
5576	(47)	5589	(47)	5591	(47)	5592	(47)	5601	(47)
5602	(47)	5617	(47)	5618	(47)	5627	(47)	5628	(47)
5639	(47)	5645	(47)	5801	(49)	5807	(49)	5809	(49)
5810	(49)	5826	(49)	5833	(49)	5839	(49)	5849	(49)
5854	(49)	6176	(53)	6215	(53)	6218	(53)	6268	(54)
6269	(54)	6270	(54)	6271	(54)	6275	(54)	6278	(54)
6283	(54)	6296	(54)	6297	(54)	6299	(54)	6309	(54)
6319	(54)	6325	(54)	6344	(54)	6428	(56)	6435	(56)
6437	(56)	6450	(56)	6478	(56)	6499	(56)	6503	(56)
6507	(56)	6569	(57)	6577	(57)	6580	(57)	6583	(57)
6839	(61)	6862	(61)	7035	(63)	7043	(63)	7049	(63)
7057	(63)	7131	(64)	7147	(64)	7154	(64)	7171	(64)
7177	(64)	7197	(64)	7297	(65)	7320	(65)	7324	(65)
7426	(66)	7490	(66)	7491	(66)	7507	(66)	7523	(66)
7525	(66)	7527	(66)	7529	(66)	7629	(67)	7680	(67)
7681	(67)	7682	(67)	7686	(67)	7687	(67)	7693	(67)
7726	(67)	7735	(67)	7736	(67)	7737	(67)	7738	(67)
7739	(67)	7747	(67)	7749	(67)	7750	(67)	7752	(67)
7754	(67)	7756	(67)	7760	(67)	7761	(67)	7768	(67)
7769	(67)	7771	(67)	7777	(67)	7778	(67)	7780	(67)
7781	(67)	7787	(67)	7788	(67)	7790	(67)	7791	(67)
7930	(70)	7933	(70)	7937	(70)	7951	(70)	7954	(70)
7956	(70)	7957	(70)	7960	(70)	7971	(70)	7974	(70)
7976	(70)	7978	(70)	7981	(70)	7990	(70)	7992	(70)
8032	(71)	8050	(71)	8062	(71)	8068	(71)	8077	(71)
8110	(72)	8145	(73)	8247	(76)	8249	(76)	8260	(76)
8261	(76)	8262	(76)	8267	(76)	8308	(76)	8310	(76)
8313	(76)	8314	(76)	8316	(76)	8317	(76)	8320	(76)
8321	(76)	8322	(76)	8325	(76)	8326	(76)	8327	(76)
8337	(76)	8340	(76)	8343	(76)	8346	(76)	8442	(79)
8443	(79)	8452	(79)	8454	(79)	8455	(79)	8456	(79)
8460	(79)	8497	(80)	8503	(80)	8557	(81)	8565	(81)
8573	(81)	8575	(81)	8623	(82)	8675	(83)	8676	(83)
8679	(83)	8683	(83)	8728	(84)	8729	(84)	8732	(84)
8736	(84)	8783	(85)	8791	(85)	8806	(85)	8845	(86)
8915	(87)	8916	(87)	8917	(87)	8928	(87)	8930	(87)
8934	(87)	8935	(87)	8939	(87)	8945	(87)	8946	(87)
8947	(87)	8951	(87)	8952	(87)	8956	(87)	8957	(87)
8958	(87)	8963	(87)	8970	(87)	8971	(87)	8973	(87)
8980	(87)	8981	(87)	8983	(87)	8990	(87)	8991	(87)
8993	(87)	8998	(87)	9002	(87)	9005	(87)	9011	(87)
9016	(87)	9020	(87)	9022	(87)	9024	(87)	9034	(87)
9040	(87)	9042	(87)	9043	(87)	9045	(87)	9047	(87)
9049	(87)	9052	(87)	9053	(87)	9056	(87)	9061	(87)
9069	(87)	9073	(87)	9074	(87)	9075	(87)	9076	(87)

				9077	(87)	9079	(87)	9081	(87)	9086	(87)	9090	(87)
				9091	(87)	9092	(87)	9093	(87)	9095	(87)	9096	(87)
				9098	(87)	9099	(87)	9139	(88)	9140	(88)	9145	(88)
				9150	(88)	9212	(89)	9213	(89)	9215	(89)	9218	(89)
				9220	(89)	9224	(89)	9228	(89)	9249	(89)	9272	(89)
				9283	(89)	9285	(89)	9302	(89)	9337	(89)	9346	(89)
				9349	(89)	9350	(89)	9392	(90)	9406	(90)	9412	(90)
				9413	(90)	9429	(90)	9430	(90)	9441	(90)	9447	(90)
				9448	(90)	9449	(90)	9584	(93)	9586	(93)	9625	(94)
				9626	(94)	9627	(94)	9628	(94)	9629	(94)	9630	(94)
				9631	(94)	9632	(94)	9633	(94)	9634	(94)	9635	(94)
				9641	(94)	9644	(94)	9652	(94)	9655	(94)	9656	(94)
				9657	(94)	9658	(94)	9660	(94)	9661	(94)	9663	(94)
				9664	(94)	9665	(94)	9666	(94)	9667	(94)	9706	(95)
				9715	(95)	9720	(95)	9725	(95)	9728	(95)	9734	(95)
				9740	(95)	9746	(95)	9750	(95)	9763	(95)	9769	(95)
				9776	(95)	9777	(95)	9778	(95)	9779	(95)	9787	(95)
				9795	(95)	9796	(95)	9802	(95)	9805	(95)	9808	(95)
				9836	(96)	9840	(96)	9843	(96)	9845	(96)	9846	(96)
				9863	(97)	9864	(97)	9865	(97)	9867	(97)	9893	(98)
				9897	(98)	9900	(98)	9902	(98)	9903	(98)		
HALT_FLAG	=00000001	863	(4)	1453	(6)	4756	(40)	5100	(44)	7305	(65)		
HALT_FLAG_NAME	0000034B-R	1730	(6)	4122	(32)								
HALT_KEY	=00000008	863	(4)	5704	(48)								
HALT_ON_MATCH	=00000001	863	(4)	5271	(45)	5369	(46)						
HEAD	=FFFF7C00	863	(4)	1373	(5)	1376	(5)	1377	(5)	1389	(5)	1393	(5)
				1403	(5)	1410	(5)	1415	(5)	1416	(5)	1417	(5)
				1419	(5)	1421	(5)	1424	(5)	1425	(5)	1430	(5)
				1431	(5)	1470	(6)	1471	(6)	1472	(6)	1473	(6)
				1474	(6)	1475	(6)	1476	(6)	1477	(6)	1478	(6)
				1479	(6)	1480	(6)	1481	(6)	1482	(6)	1696	(6)
				1697	(6)	1698	(6)	1699	(6)	1700	(6)	1701	(6)
				1702	(6)	1703	(6)	1704	(6)	1705	(6)	1711	(6)
				1804	(6)	1805	(6)	1806	(6)	1807	(6)	1808	(6)
				1809	(6)	1810	(6)	1811	(6)	1812	(6)	1813	(6)
				1814	(6)	1815	(6)	1816	(6)	1817	(6)	1818	(6)
				1819	(6)	1820	(6)	1821	(6)	1822	(6)	1823	(6)
				1824	(6)	1825	(6)	1826	(6)	1827	(6)	1828	(6)
				1829	(6)	1830	(6)	1831	(6)	1832	(6)	1837	(6)
				1838	(6)	1839	(6)	1990	(6)	1991	(6)	1992	(6)
				1993	(6)	1994	(6)	1995	(6)	1996	(6)	1997	(6)
				1998	(6)	1999	(6)	2000	(6)	2001	(6)	2002	(6)
				2003	(6)	2004	(6)	2264	(7)	2266	(7)	2267	(7)
				2269	(7)	2270	(7)	2272	(7)	2276	(7)	2279	(7)
				2282	(7)	2284	(7)	2290	(7)	2296	(7)	2297	(7)
				2301	(7)	2302	(7)	2305	(7)	2310	(7)	2314	(7)
				2317	(7)	2319	(7)	2321	(7)	2325	(7)	2329	(7)
				2330	(7)	2331	(7)	2332	(7)	2333	(7)	2338	(7)
				2340	(7)	2342	(7)	2343	(7)	2344	(7)	2345	(7)
				2349	(7)	2351	(7)	2357	(7)	2361	(7)	2362	(7)
				2363	(7)	2368	(7)	2369	(7)	2370	(7)	2374	(7)
				2378	(7)	2379	(7)	2380	(7)	2381	(7)	2382	(7)
				2384	(7)	2386	(7)	2388	(7)	2389	(7)	2390	(7)
				2391	(7)	2392	(7)	2397	(7)	2398	(7)	2399	(7)
				2463	(8)	2464	(8)	2465	(8)	2466	(8)	2470	(8)

2471	(8)	2472	(8)	2473	(8)	2474	(8)	2475	(8)
2476	(8)	2477	(8)	2518	(9)	2519	(9)	2520	(9)
2522	(9)	2523	(9)	2524	(9)	2525	(9)	2568	(10)
2569	(10)	2570	(10)	2571	(10)	2572	(10)	2627	(11)
2629	(11)	2631	(11)	2634	(11)	2635	(11)	2636	(11)
2697	(12)	2699	(12)	2702	(12)	2705	(12)	2707	(12)
2708	(12)	2709	(12)	2710	(12)	2745	(13)	2747	(13)
2748	(13)	2749	(13)	2751	(13)	2753	(13)	2799	(14)
2801	(14)	2802	(14)	2803	(14)	2804	(14)	2806	(14)
2807	(14)	2837	(14)	2843	(14)	2847	(14)	2849	(14)
2887	(15)	2889	(15)	2891	(15)	2893	(15)	2895	(15)
2901	(15)	2902	(15)	2906	(15)	2937	(16)	2939	(16)
2940	(16)	2942	(16)	2944	(16)	2952	(16)	2953	(16)
2957	(16)	2997	(17)	2998	(17)	2999	(17)	3001	(17)
3002	(17)	3003	(17)	3005	(17)	3006	(17)	3041	(18)
3044	(18)	3045	(18)	3046	(18)	3047	(18)	3048	(18)
3051	(18)	3053	(18)	3054	(18)	3056	(18)	3057	(18)
3062	(18)	3065	(18)	3068	(18)	3101	(19)	3104	(19)
3105	(19)	3106	(19)	3107	(19)	3108	(19)	3110	(19)
3112	(19)	3113	(19)	3115	(19)	3116	(19)	3121	(19)
3124	(19)	3127	(19)	3207	(20)	3215	(20)	3219	(20)
3226	(20)	3227	(20)	3230	(20)	3237	(20)	3245	(20)
3250	(20)	3252	(20)	3253	(20)	3258	(20)	3260	(20)
3262	(20)	3267	(20)	3274	(20)	3278	(20)	3279	(20)
3281	(20)	3285	(20)	3294	(20)	3299	(20)	3305	(20)
3307	(20)	3308	(20)	3313	(20)	3315	(20)	3316	(20)
3317	(20)	3323	(20)	3329	(20)	3333	(20)	3340	(20)
3352	(20)	3354	(20)	3358	(20)	3360	(20)	3368	(20)
3369	(20)	3371	(20)	3376	(20)	3377	(20)	3379	(20)
3381	(20)	3383	(20)	3385	(20)	3387	(20)	3391	(20)
3392	(20)	3396	(20)	3402	(20)	3403	(20)	3405	(20)
3416	(20)	3424	(20)	3426	(20)	3463	(21)	3465	(21)
3469	(21)	3474	(21)	3480	(21)	3484	(21)	3521	(22)
3522	(22)	3527	(22)	3529	(22)	3531	(22)	3535	(22)
3536	(22)	3604	(23)	3606	(23)	3609	(23)	3613	(23)
3627	(23)	3636	(23)	3637	(23)	3646	(23)	3649	(23)
3650	(23)	3657	(23)	3661	(23)	3665	(23)	3667	(23)
3673	(23)	3677	(23)	3679	(23)	3682	(23)	3687	(23)
3695	(23)	3701	(23)	3702	(23)	3724	(23)	3779	(24)
3781	(24)	3785	(24)	3787	(24)	3791	(24)	3792	(24)
3797	(24)	3798	(24)	3800	(24)	3803	(24)	3804	(24)
3806	(24)	3807	(24)	3808	(24)	3810	(24)	3811	(24)
3812	(24)	3813	(24)	3814	(24)	3815	(24)	3816	(24)
3817	(24)	3821	(24)	3826	(24)	3828	(24)	3830	(24)
3835	(24)	3837	(24)	3841	(24)	3843	(24)	3845	(24)
3850	(24)	3852	(24)	3856	(24)	3858	(24)	3862	(24)
3863	(24)	3864	(24)	3890	(25)	3893	(25)	3895	(25)
3915	(26)	3916	(26)	3917	(26)	3918	(26)	3953	(27)
3955	(27)	3959	(27)	3961	(27)	3962	(27)	3970	(27)
3971	(27)	3975	(27)	3978	(27)	3979	(27)	3983	(27)
3984	(27)	4009	(28)	4011	(28)	4012	(28)	4040	(29)
4042	(29)	4043	(29)	4068	(30)	4069	(30)	4073	(30)
4074	(30)	4075	(30)	4078	(30)	4080	(30)	4094	(31)
4095	(31)	4098	(31)	4099	(31)	4100	(31)	4103	(31)
4104	(31)	4120	(32)	4122	(32)	4124	(32)	4128	(32)

4129	(32)	4136	(32)	4137	(32)	4157	(33)	4161	(33)
4201	(34)	4204	(34)	4206	(34)	4207	(34)	4208	(34)
4209	(34)	4210	(34)	4212	(34)	4213	(34)	4215	(34)
4216	(34)	4217	(34)	4218	(34)	4219	(34)	4220	(34)
4226	(34)	4228	(34)	4229	(34)	4230	(34)	4231	(34)
4232	(34)	4237	(34)	4241	(34)	4251	(34)	4254	(34)
4259	(34)	4265	(34)	4268	(34)	4271	(34)	4272	(34)
4275	(34)	4280	(34)	4282	(34)	4309	(35)	4310	(35)
4311	(35)	4313	(35)	4317	(35)	4318	(35)	4322	(35)
4326	(35)	4327	(35)	4329	(35)	4330	(35)	4334	(35)
4335	(35)	4336	(35)	4340	(35)	4346	(35)	4347	(35)
4348	(35)	4351	(35)	4358	(35)	4359	(35)	4360	(35)
4362	(35)	4363	(35)	4364	(35)	4368	(35)	4369	(35)
4376	(35)	4378	(35)	4380	(35)	4434	(36)	4436	(36)
4448	(36)	4450	(36)	4452	(36)	4460	(36)	4463	(36)
4516	(37)	4519	(37)	4525	(37)	4527	(37)	4532	(37)
4569	(38)	4572	(38)	4616	(39)	4630	(39)	4632	(39)
4635	(39)	4637	(39)	4642	(39)	4644	(39)	4652	(39)
4662	(39)	4668	(39)	4672	(39)	4679	(39)	4685	(39)
4687	(39)	4694	(39)	4695	(39)	4753	(40)	4757	(40)
4758	(40)	4762	(40)	4763	(40)	4768	(40)	4770	(40)
4773	(40)	4774	(40)	4786	(40)	4791	(40)	4793	(40)
4795	(40)	4797	(40)	4799	(40)	4801	(40)	4805	(40)
4807	(40)	4808	(40)	4813	(40)	4815	(40)	4816	(40)
4821	(40)	4823	(40)	4824	(40)	4829	(40)	4831	(40)
4832	(40)	4840	(40)	4844	(40)	4850	(40)	4854	(40)
4860	(40)	4870	(40)	4872	(40)	4873	(40)	4881	(40)
4883	(40)	4894	(40)	4896	(40)	4950	(41)	4954	(41)
4955	(41)	4956	(41)	4957	(41)	4958	(41)	4960	(41)
4961	(41)	4966	(41)	4970	(41)	4972	(41)	4974	(41)
4975	(41)	4979	(41)	4981	(41)	4982	(41)	4983	(41)
4987	(41)	4993	(41)	4994	(41)	4998	(41)	5001	(41)
5004	(41)	5007	(41)	5056	(42)	5058	(42)	5079	(43)
5098	(44)	5101	(44)	5102	(44)	5104	(44)	5105	(44)
5106	(44)	5156	(45)	5158	(45)	5160	(45)	5162	(45)
5164	(45)	5171	(45)	5175	(45)	5176	(45)	5177	(45)
5178	(45)	5187	(45)	5188	(45)	5189	(45)	5190	(45)
5192	(45)	5198	(45)	5203	(45)	5204	(45)	5207	(45)
5209	(45)	5213	(45)	5216	(45)	5218	(45)	5228	(45)
5231	(45)	5234	(45)	5237	(45)	5242	(45)	5245	(45)
5249	(45)	5252	(45)	5253	(45)	5257	(45)	5258	(45)
5264	(45)	5266	(45)	5270	(45)	5272	(45)	5273	(45)
5279	(45)	5286	(45)	5287	(45)	5290	(45)	5291	(45)
5354	(46)	5356	(46)	5358	(46)	5360	(46)	5365	(46)
5367	(46)	5368	(46)	5370	(46)	5371	(46)	5377	(46)
5381	(46)	5382	(46)	5385	(46)	5386	(46)	5387	(46)
5397	(46)	5398	(46)	5399	(46)	5403	(46)	5405	(46)
5406	(46)	5414	(46)	5418	(46)	5419	(46)	5420	(46)
5480	(47)	5482	(47)	5483	(47)	5491	(47)	5493	(47)
5495	(47)	5497	(47)	5499	(47)	5501	(47)	5503	(47)
5505	(47)	5507	(47)	5509	(47)	5511	(47)	5513	(47)
5515	(47)	5516	(47)	5523	(47)	5526	(47)	5527	(47)
5534	(47)	5538	(47)	5539	(47)	5545	(47)	5551	(47)
5557	(47)	5563	(47)	5569	(47)	5575	(47)	5576	(47)
5577	(47)	5584	(47)	5587	(47)	5589	(47)	5591	(47)

5592	(47)	5593	(47)	5595	(47)	5601	(47)	5602	(47)
5603	(47)	5605	(47)	5606	(47)	5608	(47)	5609	(47)
5611	(47)	5617	(47)	5618	(47)	5619	(47)	5621	(47)
5627	(47)	5628	(47)	5629	(47)	5638	(47)	5639	(47)
5640	(47)	5645	(47)	5650	(47)	5651	(47)	5705	(48)
5707	(48)	5712	(48)	5784	(49)	5786	(49)	5790	(49)
5792	(49)	5797	(49)	5799	(49)	5803	(49)	5806	(49)
5808	(49)	5812	(49)	5813	(49)	5816	(49)	5817	(49)
5818	(49)	5819	(49)	5823	(49)	5824	(49)	5827	(49)
5830	(49)	5835	(49)	5840	(49)	5844	(49)	5849	(49)
5857	(49)	5905	(50)	5906	(50)	5910	(50)	5911	(50)
5918	(50)	5919	(50)	5922	(50)	5927	(50)	5929	(50)
5930	(50)	5996	(51)	5998	(51)	6003	(51)	6007	(51)
6011	(51)	6013	(51)	6017	(51)	6019	(51)	6020	(51)
6022	(51)	6024	(51)	6025	(51)	6026	(51)	6029	(51)
6030	(51)	6034	(51)	6036	(51)	6040	(51)	6042	(51)
6044	(51)	6048	(51)	6050	(51)	6051	(51)	6052	(51)
6053	(51)	6056	(51)	6057	(51)	6061	(51)	6064	(51)
6068	(51)	6069	(51)	6070	(51)	6071	(51)	6074	(51)
6075	(51)	6076	(51)	6111	(52)	6116	(52)	6118	(52)
6120	(52)	6122	(52)	6123	(52)	6124	(52)	6128	(52)
6169	(53)	6170	(53)	6171	(53)	6178	(53)	6182	(53)
6184	(53)	6186	(53)	6190	(53)	6192	(53)	6194	(53)
6195	(53)	6201	(53)	6206	(53)	6210	(53)	6211	(53)
6216	(53)	6219	(53)	6263	(54)	6265	(54)	6266	(54)
6267	(54)	6277	(54)	6298	(54)	6315	(54)	6320	(54)
6321	(54)	6326	(54)	6332	(54)	6335	(54)	6340	(54)
6341	(54)	6372	(55)	6373	(55)	6421	(56)	6422	(56)
6427	(56)	6428	(56)	6434	(56)	6436	(56)	6440	(56)
6452	(56)	6461	(56)	6463	(56)	6470	(56)	6472	(56)
6482	(56)	6483	(56)	6494	(56)	6501	(56)	6503	(56)
6508	(56)	6544	(57)	6545	(57)	6546	(57)	6547	(57)
6551	(57)	6554	(57)	6562	(57)	6564	(57)	6573	(57)
6574	(57)	6579	(57)	6585	(57)	6624	(58)	6627	(58)
6629	(58)	6631	(58)	6633	(58)	6635	(58)	6641	(58)
6642	(58)	6648	(58)	6649	(58)	6653	(58)	6654	(58)
6656	(58)	6657	(58)	6664	(58)	6667	(58)	6671	(58)
6673	(58)	6674	(58)	6678	(58)	6680	(58)	6684	(58)
6686	(58)	6688	(58)	6720	(59)	6724	(59)	6726	(59)
6730	(59)	6731	(59)	6771	(60)	6773	(60)	6774	(60)
6775	(60)	6779	(60)	6781	(60)	6782	(60)	6784	(60)
6785	(60)	6789	(60)	6790	(60)	6824	(61)	6825	(61)
6829	(61)	6830	(61)	6834	(61)	6837	(61)	6843	(61)
6844	(61)	6845	(61)	6846	(61)	6847	(61)	6850	(61)
6856	(61)	6864	(61)	6909	(62)	6910	(62)	6911	(62)
6915	(62)	6916	(62)	6918	(62)	6920	(62)	6921	(62)
6922	(62)	6923	(62)	6926	(62)	6928	(62)	6929	(62)
6933	(62)	6934	(62)	6936	(62)	6937	(62)	6938	(62)
6939	(62)	6943	(62)	6948	(62)	6951	(62)	6952	(62)
6999	(63)	7000	(63)	7001	(63)	7005	(63)	7006	(63)
7008	(63)	7010	(63)	7011	(63)	7012	(63)	7013	(63)
7016	(63)	7018	(63)	7019	(63)	7023	(63)	7024	(63)
7025	(63)	7026	(63)	7027	(63)	7030	(63)	7036	(63)
7037	(63)	7045	(63)	7050	(63)	7051	(63)	7059	(63)
7063	(63)	7064	(63)	7066	(63)	7067	(63)	7068	(63)

7069	(63)	7073	(63)	7078	(63)	7081	(63)	7082	(63)
7125	(64)	7127	(64)	7132	(64)	7133	(64)	7134	(64)
7135	(64)	7139	(64)	7140	(64)	7142	(64)	7148	(64)
7149	(64)	7151	(64)	7153	(64)	7155	(64)	7161	(64)
7166	(64)	7175	(64)	7176	(64)	7179	(64)	7183	(64)
7186	(64)	7187	(64)	7192	(64)	7196	(64)	7198	(64)
7201	(64)	7273	(65)	7275	(65)	7279	(65)	7281	(65)
7285	(65)	7287	(65)	7291	(65)	7293	(65)	7297	(65)
7298	(65)	7300	(65)	7304	(65)	7306	(65)	7310	(65)
7313	(65)	7316	(65)	7320	(65)	7321	(65)	7323	(65)
7325	(65)	7327	(65)	7331	(65)	7332	(65)	7333	(65)
7335	(65)	7424	(66)	7425	(66)	7426	(66)	7427	(66)
7428	(66)	7431	(66)	7432	(66)	7434	(66)	7438	(66)
7440	(66)	7444	(66)	7446	(66)	7447	(66)	7450	(66)
7453	(66)	7454	(66)	7456	(66)	7462	(66)	7463	(66)
7465	(66)	7466	(66)	7467	(66)	7473	(66)	7474	(66)
7476	(66)	7477	(66)	7478	(66)	7480	(66)	7484	(66)
7486	(66)	7490	(66)	7491	(66)	7492	(66)	7496	(66)
7498	(66)	7500	(66)	7501	(66)	7503	(66)	7506	(66)
7507	(66)	7511	(66)	7513	(66)	7514	(66)	7517	(66)
7519	(66)	7523	(66)	7524	(66)	7528	(66)	7529	(66)
7530	(66)	7535	(66)	7537	(66)	7538	(66)	7541	(66)
7542	(66)	7544	(66)	7550	(66)	7552	(66)	7554	(66)
7555	(66)	7561	(66)	7562	(66)	7563	(66)	7565	(66)
7569	(66)	7571	(66)	7572	(66)	7574	(66)	7575	(66)
7576	(66)	7577	(66)	7578	(66)	7582	(66)	7585	(66)
7586	(66)	7587	(66)	7593	(67)	7595	(67)	7596	(67)
7598	(67)	7603	(67)	7604	(67)	7606	(67)	7607	(67)
7609	(67)	7610	(67)	7614	(67)	7615	(67)	7617	(67)
7621	(67)	7626	(67)	7628	(67)	7629	(67)	7630	(67)
7634	(67)	7635	(67)	7639	(67)	7641	(67)	7642	(67)
7643	(67)	7647	(67)	7648	(67)	7649	(67)	7651	(67)
7652	(67)	7655	(67)	7657	(67)	7661	(67)	7663	(67)
7667	(67)	7669	(67)	7673	(67)	7675	(67)	7680	(67)
7681	(67)	7682	(67)	7683	(67)	7685	(67)	7686	(67)
7687	(67)	7688	(67)	7692	(67)	7693	(67)	7694	(67)
7695	(67)	7697	(67)	7700	(67)	7701	(67)	7705	(67)
7707	(67)	7709	(67)	7710	(67)	7715	(67)	7717	(67)
7726	(67)	7727	(67)	7728	(67)	7729	(67)	7734	(67)
7735	(67)	7736	(67)	7737	(67)	7738	(67)	7739	(67)
7740	(67)	7742	(67)	7745	(67)	7748	(67)	7749	(67)
7750	(67)	7751	(67)	7755	(67)	7756	(67)	7760	(67)
7762	(67)	7763	(67)	7765	(67)	7766	(67)	7770	(67)
7771	(67)	7774	(67)	7775	(67)	7779	(67)	7780	(67)
7781	(67)	7784	(67)	7785	(67)	7789	(67)	7790	(67)
7791	(67)	7792	(67)	7793	(67)	7823	(68)	7825	(68)
7827	(68)	7828	(68)	7829	(68)	7830	(68)	7831	(68)
7858	(69)	7860	(69)	7906	(70)	7907	(70)	7908	(70)
7909	(70)	7913	(70)	7916	(70)	7921	(70)	7923	(70)
7924	(70)	7932	(70)	7933	(70)	7938	(70)	7940	(70)
7950	(70)	7953	(70)	7958	(70)	7959	(70)	7963	(70)
7970	(70)	7973	(70)	7979	(70)	7980	(70)	7984	(70)
7991	(70)	7993	(70)	7994	(70)	8030	(71)	8031	(71)
8033	(71)	8035	(71)	8039	(71)	8040	(71)	8041	(71)
8043	(71)	8044	(71)	8047	(71)	8051	(71)	8057	(71)

8059	(71)	8063	(71)	8069	(71)	8106	(72)	8107	(72)
8138	(73)	8139	(73)	8140	(73)	8142	(73)	8171	(74)
8173	(74)	8198	(75)	8200	(75)	8241	(76)	8246	(76)
8247	(76)	8251	(76)	8252	(76)	8253	(76)	8256	(76)
8260	(76)	8261	(76)	8262	(76)	8263	(76)	8267	(76)
8270	(76)	8273	(76)	8277	(76)	8283	(76)	8288	(76)
8289	(76)	8290	(76)	8291	(76)	8292	(76)	8293	(76)
8294	(76)	8296	(76)	8299	(76)	8304	(76)	8306	(76)
8307	(76)	8311	(76)	8312	(76)	8318	(76)	8319	(76)
8324	(76)	8329	(76)	8335	(76)	8338	(76)	8341	(76)
8344	(76)	8346	(76)	8350	(76)	8352	(76)	8356	(76)
8357	(76)	8359	(76)	8362	(76)	8363	(76)	8387	(77)
8389	(77)	8391	(77)	8416	(78)	8417	(78)	8419	(78)
8420	(78)	8421	(78)	8423	(78)	8424	(78)	8425	(78)
8426	(78)	8439	(79)	8441	(79)	8442	(79)	8443	(79)
8444	(79)	8446	(79)	8451	(79)	8452	(79)	8453	(79)
8455	(79)	8457	(79)	8460	(79)	8462	(79)	8464	(79)
8497	(80)	8507	(80)	8553	(81)	8557	(81)	8578	(81)
8581	(81)	8629	(82)	863	(4)	8631	(82)	8669	(83)
8671	(83)	8677	(83)	8678	(83)	8685	(83)	8722	(84)
8724	(84)	8730	(84)	8731	(84)	8738	(84)	8782	(85)
8783	(85)	8786	(85)	8789	(85)	8793	(85)	8804	(85)
8808	(85)	8809	(85)	8835	(86)	8837	(86)	8839	(86)
8841	(86)	8845	(86)	8849	(86)	8851	(86)	8916	(87)
8917	(87)	8921	(87)	8923	(87)	8926	(87)	8929	(87)
8930	(87)	8934	(87)	8935	(87)	8939	(87)	8940	(87)
8942	(87)	8944	(87)	8945	(87)	8946	(87)	8947	(87)
8951	(87)	8952	(87)	8953	(87)	8955	(87)	8956	(87)
8957	(87)	8958	(87)	8959	(87)	8964	(87)	8965	(87)
8967	(87)	8968	(87)	8972	(87)	8973	(87)	8974	(87)
8977	(87)	8978	(87)	8982	(87)	8983	(87)	8984	(87)
8987	(87)	8988	(87)	8992	(87)	8993	(87)	8994	(87)
9001	(87)	9004	(87)	9006	(87)	9007	(87)	9011	(87)
9012	(87)	9017	(87)	9019	(87)	9023	(87)	9028	(87)
9030	(87)	9035	(87)	9036	(87)	9039	(87)	9042	(87)
9043	(87)	9046	(87)	9049	(87)	9050	(87)	9055	(87)
9064	(87)	9072	(87)	9076	(87)	9081	(87)	9089	(87)
9090	(87)	9091	(87)	9097	(87)	9099	(87)	9138	(88)
9139	(88)	9140	(88)	9141	(88)	9143	(88)	9147	(88)
9149	(88)	9150	(88)	9151	(88)	9152	(88)	9154	(88)
9155	(88)	9157	(88)	9159	(88)	9165	(88)	9210	(89)
9211	(89)	9214	(89)	9216	(89)	9217	(89)	9223	(89)
9224	(89)	9231	(89)	9239	(89)	9244	(89)	9246	(89)
9248	(89)	9249	(89)	9261	(89)	9263	(89)	9272	(89)
9274	(89)	9286	(89)	9289	(89)	9291	(89)	9293	(89)
9317	(89)	9324	(89)	9326	(89)	9328	(89)	9330	(89)
9331	(89)	9337	(89)	9348	(89)	9352	(89)	9353	(89)
9355	(89)	9356	(89)	9396	(90)	9401	(90)	9408	(90)
9423	(90)	9431	(90)	9436	(90)	9443	(90)	9456	(90)
9485	(91)	9487	(91)	9519	(92)	9523	(92)	9524	(92)
9528	(92)	9537	(92)	9538	(92)	9540	(92)	9542	(92)
9571	(93)	9573	(93)	9574	(93)	9576	(93)	9577	(93)
9579	(93)	9583	(93)	9585	(93)	9587	(93)	9588	(93)
9589	(93)	9591	(93)	9595	(93)	9598	(93)	9628	(94)
9629	(94)	9633	(94)	9638	(94)	9640	(94)	9642	(94)

				9643	(94)	9652	(94)	9654	(94)	9655	(94)	9656	(94)
				9662	(94)	9665	(94)	9706	(95)	9726	(95)	9727	(95)
				9728	(95)	9770	(95)	9772	(95)	9781	(95)	9804	(95)
				9810	(95)	9834	(96)	9835	(96)	9836	(96)	9841	(96)
				9843	(96)	9845	(96)	9846	(96)	9864	(97)	9866	(97)
				9867	(97)	9891	(98)	9892	(98)	9893	(98)	9898	(98)
				9900	(98)	9902	(98)	9903	(98)				
HIGH	=00000001	863	(4)										
HYPHEN	000005CE-R	1946	(6)	3637	(23)								
H FROM WBUS	=00000010	863	(4)										
IB_FLAG	=00000020	863	(4)	7439	(66)								
IB_KEY	=00000012	863	(4)										
IF_ERROR	00002052-R	7272	(65)	1813	(6)								
INCREMENT_INDEX	000024FA-R	8105	(72)	1825	(6)								
INDEX_NAM_TBL	00000408-R	1837	(6)	8557	(81)								
INIT_LOOP_VALUE	0000269B-R	8496	(80)	6007	(51)	6720	(59)						
INIT_TEST_PC	000001B5-R	1606	(6)	2369	(7)	3360	(20)	3391	(20)	3416	(20)	3814	(24)
				3821	(24)	3828	(24)	3830	(24)	5922	(50)	6440	(56)
				6463	(56)	6554	(57)	6684	(58)	6837	(61)	6850	(61)
				6926	(62)	7016	(63)	7030	(63)	7142	(64)	7830	(68)
				7916	(70)	7940	(70)	7963	(70)	7984	(70)	8047	(71)
				8059	(71)	8142	(73)	8389	(77)				
INIT THE WORLD	00001CBF-R	6110	(52)	1814	(6)								
INSERT_FIELD	00002966-R	9209	(89)	1403	(5)	3258	(20)	6508	(56)	7994	(70)	9841	(96)
				9898	(98)								
INSERT_FIELD_X	00002A1B-R	9357	(89)	9352	(89)	9355	(89)						
INSTR_FLAG	=00000004	863	(4)	5235	(45)	5785	(49)	9156	(88)				
INSTR_KEY	=00000018	863	(4)	5236	(45)								
INSTR_TABLE	000003CE-R	1804	(6)	5849	(49)								
INS_FLD_BUFF	00000C90-R	2181	(6)	9210	(89)	9216	(89)	9224	(89)	9272	(89)	9337	(89)
INT	=00000010	863	(4)										
INVALID_COMMAND	000002F3-R	1695	(6)	4231	(34)								
IRD	-00000005	863	(4)										
I_INDEX_TBL	0000040E-R	1840	(6)	1837	(6)	8497	(80)						
J_INDEX_TBL	00000415-R	1844	(6)	1838	(6)								
KEYWORD_LIST	00000343-R	1725	(6)	4334	(35)								
K_INDEX_TBL	0000041C-R	1845	(6)	1839	(6)								
L	=00000005	856	(3)	3361	(20)	3363	(20)	3417	(20)	3419	(20)	3718	(23)
				3822	(24)	3823	(24)	3824	(24)	3968	(27)	4239	(34)
				4247	(34)	5814	(49)	6274	(54)	6286	(54)	6308	(54)
				6570	(57)	7965	(70)	7966	(70)	7986	(70)	7988	(70)
				8254	(76)	8624	(82)	9590	(93)	9792	(95)	9793	(95)
LITERAL	=00000001	9902	(98)	1374	(5)	1385	(5)	1393	(5)	2269	(7)	2270	(7)
				2314	(7)	2361	(7)	2362	(7)	2378	(7)	2379	(7)
				2380	(7)	2381	(7)	2568	(10)	2569	(10)	2570	(10)
				2571	(10)	2572	(10)	2708	(12)	2710	(12)	2807	(14)
				2893	(15)	2942	(16)	3207	(20)	3230	(20)	3252	(20)
				3253	(20)	3260	(20)	3262	(20)	3267	(20)	3305	(20)
				3307	(20)	3313	(20)	3314	(20)	3323	(20)	3358	(20)
				3369	(20)	3403	(20)	3521	(22)	3606	(23)	3627	(23)
				3637	(23)	3650	(23)	3661	(23)	3679	(23)	3680	(23)
				3811	(24)	3812	(24)	3813	(24)	3817	(24)	3856	(24)
				3862	(24)	3863	(24)	3894	(25)	4009	(28)	4011	(28)
				4040	(29)	4042	(29)	4122	(32)	4129	(32)	4137	(32)
				4157	(33)	4209	(34)	4210	(34)	4217	(34)	4218	(34)

LOST_FLAG	=00000010	863	(4)	3666	(23)	3786	(24)	4761	(40)	4846	(40)	5791	(49)
				6028	(51)	6035	(51)	6063	(51)				
LOST_KEY	=0000001A	863	(4)										
LOW	=00000000	863	(4)										
M	=00000006	856	(3)	2628	(11)	2821	(14)	2833	(14)	2897	(15)	2899	(15)
				2946	(16)	2948	(16)	3212	(20)	3217	(20)	3243	(20)
				3269	(20)	3292	(20)	3324	(20)	3349	(20)	3370	(20)
				3393	(20)	3395	(20)	3524	(22)	3610	(23)	3632	(23)
				3634	(23)	3642	(23)	3645	(23)	3685	(23)	3699	(23)
				3822	(24)	4266	(34)	4269	(34)	4344	(35)	4349	(35)
				4356	(35)	4435	(36)	4514	(37)	4523	(37)	4570	(38)
				4628	(39)	4690	(39)	4782	(40)	4843	(40)	4847	(40)
				4851	(40)	4859	(40)	4880	(40)	4989	(41)	5152	(45)
				5170	(45)	5174	(45)	5183	(45)	5205	(45)	5212	(45)
				5215	(45)	5224	(45)	5243	(45)	5248	(45)	5251	(45)
				5277	(45)	5285	(45)	5289	(45)	5350	(46)	5375	(46)
				5380	(46)	5384	(46)	5393	(46)	5413	(46)	5417	(46)
				5487	(47)	5521	(47)	5525	(47)	5532	(47)	5536	(47)
				5582	(47)	5586	(47)	5825	(49)	5828	(49)	5836	(49)
				5852	(49)	5855	(49)	6174	(53)	6200	(53)	6205	(53)
				6217	(53)	6220	(53)	6343	(54)	6345	(54)	6448	(56)
				6477	(56)	6495	(56)	6582	(57)	6859	(61)	7040	(63)
				7054	(63)	7162	(64)	7170	(64)	7322	(65)	7326	(65)
				7926	(70)	7965	(70)	7986	(70)	8075	(71)	8079	(71)
				8109	(72)	8111	(72)	8144	(73)	8146	(73)	8274	(76)
				8281	(76)	8305	(76)	8309	(76)	8315	(76)	8323	(76)
				8336	(76)	8339	(76)	8342	(76)	8345	(76)	8347	(76)
				8349	(76)	8448	(79)	8449	(79)	8502	(80)	8504	(80)
				8564	(81)	8566	(81)	8572	(81)	8574	(81)	8680	(83)
				8733	(84)	8788	(85)	8803	(85)	8999	(87)	9062	(87)
				9070	(87)	9078	(87)	9085	(87)	9087	(87)	9227	(89)
				9253	(89)	9313	(89)	9343	(89)	9400	(90)	9417	(90)
				9425	(90)	9435	(90)	9450	(90)	9458	(90)	9520	(92)
				9529	(92)	9636	(94)	9714	(95)	9716	(95)	9719	(95)
				9730	(95)	9736	(95)	9742	(95)	9748	(95)	9768	(95)
				9801	(95)								
M\$TEST_SIZE	=000007A2	863	(4)	863	(4)								
M\$TEST_SIZE_D	=000003E2	863	(4)										
MA0	=00000008	863	(4)										
MA0_NAME	000000FD-R	1493	(6)	1478	(6)								
MA1	=0000000A	863	(4)										
MA1_NAME	00000113-R	1495	(6)	1480	(6)								
MA2	=0000000C	863	(4)										
MA2_NAME	00000129-R	1497	(6)	1482	(6)								
MAD	=00000013	863	(4)										
MAIN32	=0000741D	863	(4)										
MAINT_A_TO_CMI	=00000080	863	(4)										
MAINT_CMI_TO_MH	=00000020	863	(4)										
MAINT_DCS_ENABL	=00000004	863	(4)	1423	(5)	3061	(18)	3067	(18)	3120	(19)	3126	(19)
				4067	(30)	6119	(52)						
MAINT_MD_TO_CMI	=00000040	863	(4)										
MAINT_REG	=0000741D	863	(4)	1424	(5)	3001	(17)	3062	(18)	3068	(18)	3121	(19)
				3127	(19)	4068	(30)	6120	(52)	863	(4)		
MAINT_STB_INH	=00000002	863	(4)	3061	(18)	3120	(19)						
MAINT_TRAP_OFF	=00000001	863	(4)	1423	(5)	3061	(18)	3067	(18)	3120	(19)	3126	(19)

ZZ-ECKAA-8.7		Cross reference		M 6		11-FEB-1986		Fiche 2 Frame M6		Sequence 283		Page 258	
ECKAA		VAX-11/750 MICRO DIAGNOSTIC MONITOR		11-FEB-1986 10:07:52		VAX/VMS Macro V04-00		11-FEB-1986 10:06:37		[VAX750.MONITOR]ECKAA.MAR;811		(98)	
Cross reference													
SELECT_DCS	00001392-R	3914	(26)	2847	(14)	3054	(18)	3113	(19)				
SETCLR_CF_ADDR	000001B3-R	1599	(6)	3045	(18)	3105	(19)	5171	(45)	5414	(46)	7425	(66)
SETCLR_CF_DATA	000001B4-R	1600	(6)	3048	(18)	3108	(19)	5176	(45)	5419	(46)	7462	(66)
				7465	(66)	7473	(66)	7476	(66)				
SET_ACTION	00001863-R	5148	(45)	1700	(6)								
SET_DCS_CF	00000F9F-R	3040	(18)	5177	(45)	7463	(66)	7474	(66)				
SET_DONE	00001940-R	5295	(45)	5178	(45)	5187	(45)	5209	(45)	5258	(45)	5279	(45)
SET_ERR	00001943-R	5302	(45)	5156	(45)	5164	(45)	5189	(45)	5228	(45)	5237	(45)
SFLAG_SUB_ACT	00001899-R	5182	(45)	5158	(45)	5198	(45)	5207	(45)	5218	(45)		
SF_KEY	=00000040	863	(4)	5514	(47)								
SHOW_ACTION	000017AD-R	4944	(41)	1697	(6)								
SHOW_MESSAGE_1	00000391-R	1787	(6)	4954	(41)								
SHOW_MESSAGE_2	0000039D-R	1788	(6)	4957	(41)								
SHOW_TEMP	000003AB-R	1789	(6)	4993	(41)	4994	(41)						
SINGLE_INSTR	00002906-R	9137	(88)	5818	(49)								
SINGLE_MIC_INST	00000F30-R	2883	(15)	3278	(20)	3308	(20)	3317	(20)	3333	(20)	3971	(27)
				4012	(28)	4043	(29)	4099	(31)	5576	(47)	5592	(47)
				5602	(47)	5618	(47)	5628	(47)	7575	(66)	9846	(96)
				9903	(98)								
SINGLE_TICK	00000F57-R	2936	(16)	7577	(66)								
SKIP	00001E31-R	6623	(58)	1818	(6)								
SKIP_SPACES	0000164E-R	4513	(37)	4326	(35)	4347	(35)	4359	(35)	4368	(35)	4519	(37)
SOFT_CTRL_REG	00000059-R	1443	(6)	2749	(13)	2751	(13)	3665	(23)	3785	(24)	4201	(34)
				4204	(34)	4265	(34)	4758	(40)	4762	(40)	4829	(40)
				4831	(40)	4870	(40)	4872	(40)	4894	(40)	4896	(40)
				5056	(42)	5058	(42)	5102	(44)	5104	(44)	5790	(49)
				5996	(51)	6017	(51)	6026	(51)	6029	(51)	6034	(51)
				6048	(51)	6061	(51)	6064	(51)	6771	(60)	6773	(60)
				7279	(65)	7432	(66)	7434	(66)	7478	(66)	7480	(66)
				7511	(66)	7569	(66)	7571	(66)	7593	(67)	7595	(67)
				7603	(67)	7606	(67)	7607	(67)	7615	(67)	7639	(67)
				7648	(67)								
SOFT_CTRL_REGA	0000005A-R	1444	(6)	4069	(30)	5253	(45)	5257	(45)	5264	(45)	5266	(45)
				5365	(46)	5367	(46)	5784	(49)	6011	(51)	6013	(51)
				6053	(51)	6056	(51)	6071	(51)	6074	(51)	6263	(54)
				6265	(54)	6624	(58)	6724	(59)	6726	(59)	6939	(62)
				6943	(62)	6948	(62)	6951	(62)	7069	(63)	7073	(63)
				7078	(63)	7081	(63)	7183	(64)	7186	(64)	7192	(64)
				7196	(64)	7273	(65)	7285	(65)	7333	(65)	7335	(65)
				7444	(66)	7484	(66)	7538	(66)	7572	(66)	7582	(66)
				7585	(66)	7649	(67)	7667	(67)	8171	(74)	8173	(74)
				8352	(76)	8356	(76)	8359	(76)	8362	(76)	9028	(87)
				9155	(88)	9157	(88)	9595	(93)	9598	(93)		
SOFT_CTRL_REGB	0000005B-R	1445	(6)	2276	(7)	2282	(7)	2296	(7)	3463	(21)	4763	(40)
				4768	(40)	4805	(40)	4807	(40)	4813	(40)	4815	(40)
				4821	(40)	4823	(40)	7661	(67)	8198	(75)	8200	(75)
				8270	(76)	8277	(76)	8439	(79)	8444	(79)	8462	(79)
				8464	(79)								
SOMM_ADDRESS	0000038F-R	1781	(6)	3535	(22)	3917	(26)	3983	(27)	5287	(45)	5291	(45)
				5382	(46)	5386	(46)	6789	(60)	7682	(67)		
SOMM_FLAG	=00000001	863	(4)	4070	(30)	5265	(45)	5366	(46)	7668	(67)		
SOMM_KEY	=00000006	863	(4)	5161	(45)	5359	(46)						
SOMM_MSG	00000464-R	1893	(6)	7681	(67)								
SPA	=00000002	863	(4)										
SPEC_TEST_NUMB	0000038C-R	1773	(6)	3677	(23)	3791	(24)	4844	(40)	6042	(51)		

ZZ-ECKAA-8.7
ECKAA
Cross reference

Cross reference

VAX-11/750 MICRO DIAGNOSTIC MONITOR

B 7

11-FEB-1986

Fiche 2 Frame B7

Sequence 285

11-FEB-1986 10:07:52 VAX/VMS Macro V04-00

Page 260

11-FEB-1986 10:06:37 [VAX750.MONITOR]ECKAA.MAR;811 (98)

TITLE_LENGTH	000002B9-R	1677	(6)	3826	(24)	5803	(49)	7745	(67)	8926	(87)	9147	(88)
TOK	=00000006	863	(4)										
TOTAL_BLOCKS	0000013D-R	1513	(6)	2518	(9)								
TOTAL_NMB_TESTS	00000066-R	1466	(6)	3385	(20)	3646	(23)	7825	(68)				
TPC_MSG	00000C86-R	2175	(6)	9140	(88)								
TRACE_ENABL	=00000004	863	(4)										
TRAP_HALT_EN	=00000020	863	(4)	2905	(15)	2956	(16)	4978	(41)	6127	(52)	7699	(67)
TRAP_OFF	=00000002	863	(4)										
TR_FLAG	=00000080	863	(4)	3809	(24)								
TR_KEY	=00000016	863	(4)	5706	(48)								
TSTSPAN_FLAG	=00000040	863	(4)	4761	(40)	4862	(40)	6018	(51)	6028	(51)		
TU58_BUFF_ADDR	00000134-R	1503	(6)	2463	(8)	2471	(8)	2477	(8)				
TU58_ERROR	00000140-R	1520	(6)	2569	(10)								
TU58_ERR_CODE	0000013F-R	1519	(6)	2473	(8)	2520	(9)	2570	(10)				
TU58_REC_COUNT	00000136-R	1504	(6)	2464	(8)	2471	(8)						
TU58_REC_NUMB	00000137-R	1505	(6)	2466	(8)	2471	(8)						
TYPEN_DATA_PTR	000001A0-R	1571	(6)	2380	(7)	2570	(10)	2572	(10)	2702	(12)	3812	(24)
				3862	(24)	4219	(34)	4994	(41)	7507	(66)	7529	(66)
				7682	(67)	7693	(67)	7735	(67)	7737	(67)	7749	(67)
				7756	(67)	7771	(67)	7781	(67)	7791	(67)	8420	(78)
				8424	(78)	8426	(78)	8457	(79)	8930	(87)	8935	(87)
				8945	(87)	8952	(87)	9150	(88)	9866	(97)		
TYPEN_DATA_TYPE	000001A2-R	1572	(6)	2697	(12)	2702	(12)						
TYPE_ASCII	00000E82-R	2626	(11)	2269	(7)	2270	(7)	2361	(7)	2362	(7)	2378	(7)
				2379	(7)	2381	(7)	2568	(10)	2569	(10)	2571	(10)
				2708	(12)	2710	(12)	3636	(23)	3637	(23)	3649	(23)
				3650	(23)	3811	(24)	3813	(24)	3816	(24)	3817	(24)
				3856	(24)	3863	(24)	4129	(32)	4137	(32)	4209	(34)
				4210	(34)	4217	(34)	4218	(34)	4231	(34)	4954	(41)
				4957	(41)	4983	(41)	5004	(41)	7490	(66)	7491	(66)
				7523	(66)	7680	(67)	7681	(67)	7686	(67)	7687	(67)
				7726	(67)	7728	(67)	7736	(67)	7738	(67)	7739	(67)
				7750	(67)	7760	(67)	7780	(67)	7790	(67)	8260	(76)
				8261	(76)	8262	(76)	8391	(77)	8442	(79)	8452	(79)
				8455	(79)	8460	(79)	8845	(86)	8916	(87)	8917	(87)
				8934	(87)	8939	(87)	8946	(87)	8947	(87)	8951	(87)
				8956	(87)	8957	(87)	9042	(87)	9043	(87)	9076	(87)
				9089	(87)	9090	(87)	9091	(87)	9099	(87)	9139	(88)
				9140	(88)	9628	(94)	9629	(94)	9633	(94)	9654	(94)
				9655	(94)	9656	(94)	9665	(94)	9810	(95)	9864	(97)
				9867	(97)								
TYPE_ASCII_U	00000E8D-R	2632	(11)	4128	(32)								
TYPE_ERROR	00002764-R	8914	(87)	7300	(65)								
TYPE_ERR_BUFFER	00000596-R	1938	(6)	7748	(67)	7749	(67)	7755	(67)	7756	(67)	7762	(67)
				7763	(67)	7770	(67)	7771	(67)	7779	(67)	7781	(67)
				7789	(67)	7791	(67)	8929	(87)	8930	(87)	8964	(87)
				8965	(87)	8972	(87)	8973	(87)	8982	(87)	8983	(87)
				8992	(87)	8993	(87)	9149	(88)	9150	(88)		
TYPE_ERR_TEMP	0000059A R	1939	(6)	3250	(20)	3253	(20)	3279	(20)	3281	(20)	9004	(87)
				9006	(87)	9017	(87)	9019	(87)				
TYPE_FLAGS	0000149A-R	4118	(32)	4956	(41)	4960	(41)						
TYPE_GA_LIST	00002AE3 R	9620	(94)	9064	(87)								
TYPE_NUMERIC	00000E9F-R	2694	(12)	2380	(7)	2570	(10)	2572	(10)	3812	(24)	3862	(24)
				4219	(34)	4994	(41)	7507	(66)	7529	(66)	7682	(67)
				7693	(67)	7735	(67)	7737	(67)	7749	(67)	7756	(67)

				7771 (67)	7781 (67)	7791 (67)	8420 (78)	8424 (78)
				8426 (78)	8457 (79)	8930 (87)	8935 (87)	8945 (87)
				8952 (87)	9150 (88)	9866 (97)		
TYPE_TEST_FLAG	000002B7-R	1673 (6)		2389 (7)	3779 (24)	3845 (24)	4774 (40)	
TYPE_TEST_NUMB	000012C4-R	3774 (24)		3379 (20)	3405 (20)			
TYPE_TUSB_ERROR	00000E50-R	2567 (10)		2343 (7)	2397 (7)	2474 (8)	2523 (9)	
UBI	=00000006	863 (4)						
UBI_NAME	000000E7-R	1491 (6)		1476 (6)				
UDP	=0000001C	863 (4)						
UDP_NAME	00000A4C-R	2084 (6)		1996 (6)				
USER_PASS_CNT	0000038E-R	1775 (6)		2384 (7)	4770 (40)	4881 (40)		
UTR	=0000000D	863 (4)						
VAL	=00002305	1540 (6)		1526 (6)	1527 (6)	1528 (6)	1529 (6)	1530 (6)
				1531 (6)	1532 (6)	1533 (6)	1534 (6)	1535 (6)
				1536 (6)	1537 (6)	1538 (6)	1539 (6)	1540 (6)
VA_KEY	=00000032	863 (4)		5498 (47)				
VBUS_CLOCK	=00000080	863 (4)		3526 (22)				
VBUS_COMPARE	=00000080	863 (4)		6942 (62)	6950 (62)	7072 (63)	7080 (63)	7185 (64)
				7195 (64)	8355 (76)	8361 (76)	9029 (87)	
VBUS_LOAD	=00000010	863 (4)		4978 (41)	4980 (41)			
VBUS_MSG	000003AC-R	1790 (6)		4983 (41)				
VBUS_MSG_2	000005D0-R	1947 (6)		9042 (87)				
VBUS_SERIAL_IN	=00000040	863 (4)						
VBUS_SERIAL_OUT	=00000001	863 (4)		3523 (22)				
VBUS_TBL_ADDR	0000044D-R	1878 (6)		7198 (64)	9035 (87)			
VBUS_TBL_COUNT	0000044F-R	1880 (6)		7201 (64)	9036 (87)			
VB_KEY	=00000026	863 (4)		4965 (41)				
V_BUS_BUFFER	000002C4-R	1686 (6)		3521 (22)	4987 (41)	9519 (92)		
V_BUS_STROBE	=00000001	863 (4)						
WBUS_FROM_D	=00000020	863 (4)						
WDREG	=0000742C	863 (4)						
WD_KEY	=0000003A	863 (4)		5506 (47)				
WD_REG_BYTE_0	=0000742C	863 (4)		3267 (20)	3315 (20)	3323 (20)	863 (4)	
WD_REG_BYTE_1	=0000742D	863 (4)						
WD_REG_BYTE_2	=0000742E	863 (4)		3316 (20)				
WD_REG_BYTE_3	=0000742F	863 (4)		5587 (47)				
WHREG	=00007430	863 (4)						
WH_REG_BYTE_0	=00007430	863 (4)		5609 (47)	5619 (47)	5629 (47)	5645 (47)	8288 (76)
				863 (4)	9011 (87)			
WH_REG_BYTE_1	=00007431	863 (4)		8290 (76)				
WH_REG_BYTE_2	=00007432	863 (4)		5603 (47)	8292 (76)			
WH_REG_BYTE_3	=00007433	863 (4)		5593 (47)	5606 (47)			
WORD	=00000002	863 (4)		7124 (64)				
WRITE_DCS	00000EF4-R	2798 (14)		3260 (20)	3262 (20)	3305 (20)	3307 (20)	3313 (20)
				3354 (20)	4009 (28)	4011 (28)	4040 (29)	4042 (29)
				4157 (33)	5480 (47)	5575 (47)	5589 (47)	5591 (47)
				5601 (47)	5617 (47)	5627 (47)	5930 (50)	9843 (96)
				9845 (96)	9900 (98)	9902 (98)		
X	=00000003	1540 (6)		1526 (6)	1527 (6)	1528 (6)	1529 (6)	1530 (6)
				1531 (6)	1532 (6)	1533 (6)	1534 (6)	1535 (6)
				1536 (6)	1537 (6)	1538 (6)	1539 (6)	1540 (6)
Y	=0000004E	9902 (98)		1374 (5)	1385 (5)	1393 (5)	1526 (6)	1527 (6)
				1528 (6)	1529 (6)	1530 (6)	1531 (6)	1532 (6)
				1533 (6)	1534 (6)	1535 (6)	1536 (6)	1537 (6)
				1538 (6)	1539 (6)	1540 (6)	2269 (7)	2270 (7)

2314	(7)	2361	(7)	2362	(7)	2378	(7)	2379	(7)
2380	(7)	2381	(7)	2568	(10)	2569	(10)	2570	(10)
2571	(10)	2572	(10)	2708	(12)	2710	(12)	2807	(14)
2893	(15)	2942	(16)	3207	(20)	3230	(20)	3252	(20)
3253	(20)	3260	(20)	3262	(20)	3267	(20)	3305	(20)
3307	(20)	3313	(20)	3314	(20)	3323	(20)	3358	(20)
3369	(20)	3403	(20)	3521	(22)	3606	(23)	3627	(23)
3637	(23)	3650	(23)	3661	(23)	3679	(23)	3680	(23)
3811	(24)	3812	(24)	3813	(24)	3817	(24)	3856	(24)
3862	(24)	3863	(24)	3894	(25)	4009	(28)	4011	(28)
4040	(29)	4042	(29)	4122	(32)	4129	(32)	4137	(32)
4157	(33)	4209	(34)	4210	(34)	4217	(34)	4218	(34)
4219	(34)	4231	(34)	4241	(34)	4251	(34)	4254	(34)
4265	(34)	4272	(34)	4309	(35)	4311	(35)	4313	(35)
4334	(35)	4616	(39)	4652	(39)	4672	(39)	4685	(39)
4954	(41)	4957	(41)	4983	(41)	4987	(41)	4994	(41)
5004	(41)	5480	(47)	5481	(47)	5575	(47)	5589	(47)
5591	(47)	5601	(47)	5617	(47)	5627	(47)	5639	(47)
5645	(47)	5827	(49)	5849	(49)	5857	(49)	6171	(53)
6428	(56)	6494	(56)	6503	(56)	7036	(63)	7050	(63)
7131	(64)	7297	(65)	7320	(65)	7426	(66)	7490	(66)
7491	(66)	7507	(66)	7523	(66)	7529	(66)	7629	(67)
7680	(67)	7681	(67)	7682	(67)	7686	(67)	7687	(67)
7693	(67)	7726	(67)	7735	(67)	7736	(67)	7737	(67)
7738	(67)	7739	(67)	7749	(67)	7750	(67)	7756	(67)
7760	(67)	7761	(67)	7771	(67)	7780	(67)	7781	(67)
7790	(67)	7791	(67)	7924	(70)	7933	(70)	8069	(71)
8247	(76)	8260	(76)	8261	(76)	8262	(76)	8267	(76)
8346	(76)	8442	(79)	8443	(79)	8452	(79)	8455	(79)
8460	(79)	8497	(80)	8557	(81)	8671	(83)	8676	(83)
8724	(84)	8729	(84)	8782	(85)	8783	(85)	8845	(86)
8916	(87)	8917	(87)	8930	(87)	8934	(87)	8935	(87)
8939	(87)	8945	(87)	8946	(87)	8947	(87)	8951	(87)
8952	(87)	8956	(87)	8957	(87)	8958	(87)	8963	(87)
8973	(87)	8983	(87)	8993	(87)	9011	(87)	9042	(87)
9043	(87)	9049	(87)	9076	(87)	9081	(87)	9090	(87)
9091	(87)	9099	(87)	9139	(88)	9140	(88)	9150	(88)
9224	(89)	9249	(89)	9259	(89)	9272	(89)	9287	(89)
9337	(89)	9396	(90)	9431	(90)	9519	(92)	9628	(94)
9629	(94)	9633	(94)	9652	(94)	9655	(94)	9656	(94)
9665	(94)	9706	(95)	9726	(95)	9727	(95)	9728	(95)
9835	(96)	9836	(96)	9843	(96)	9845	(96)	9864	(97)
9867	(97)	9892	(98)	9893	(98)	9900	(98)	9902	(98)

 ! Macros Cross Reference !

MACRO	SIZE	DEFINITION	REFERENCES...
\$CONVERT	2	2700 (12)	2700 (12)
\$TERM_INIT	1	2264 (7)	2264 (7) 4208 (34) 7561 (66) 9138 (88)
\$TERM_READ	3	2265 (7)	2265 (7) 2752 (13) 4211 (34) 4274 (34) 7562 (66)
			9151 (88) 9164 (88)
\$TERM_WRITE	3	2633 (11)	2633 (11)
\$TU58_OPEN	2	2518 (9)	2518 (9)
\$TU58_READ	3	2471 (8)	2471 (8)
ABORT	1	754 (2)	1377 (5) 2636 (11) 2748 (13) 5079 (43)
ACI	1	856 (3)	
ADC	1	856 (3)	9534 (92) 9795 (95)
ADD	1	856 (3)	2950 (16) 9531 (92) 9649 (94) 9792 (95)
ADDRESS	1	767 (2)	1470 (6) 1471 (6) 1472 (6) 1473 (6) 1474 (6)
			1475 (6) 1476 (6) 1477 (6) 1478 (6) 1479 (6)
			1480 (6) 1481 (6) 1482 (6) 1483 (6) 1484 (6)
			1698 (6) 1699 (6) 1700 (6) 1701 (6) 1702 (6)
			1703 (6) 1704 (6) 1705 (6) 1711 (6) 1804 (6)
			1805 (6) 1806 (6) 1807 (6) 1808 (6) 1809 (6)
			1810 (6) 1811 (6) 1812 (6) 1813 (6) 1814 (6)
			1815 (6) 1816 (6) 1817 (6) 1818 (6) 1819 (6)
			1820 (6) 1821 (6) 1822 (6) 1823 (6) 1824 (6)
			1825 (6) 1826 (6) 1827 (6) 1828 (6) 1829 (6)
			1830 (6) 1831 (6) 1832 (6) 1837 (6) 1838 (6)
			1839 (6) 1990 (6) 1991 (6) 1992 (6) 1993 (6)
			1994 (6) 1995 (6) 1996 (6) 1997 (6) 1998 (6)
			1999 (6) 2000 (6) 2001 (6) 2002 (6) 2003 (6)
			2004 (6)
ADI	1	856 (3)	4645 (39) 9264 (89) 9292 (89)
ANA	1	856 (3)	3052 (18) 5404 (46) 5637 (47) 6859 (61) 7040 (63)
			7054 (63) 9314 (89) 9400 (90) 9435 (90) 9768 (95)
ANI	1	1418 (5)	1418 (5) 2278 (7) 2289 (7) 2295 (7) 2696 (12)
			2704 (12) 2888 (15) 2896 (15) 2903 (15) 2938 (16)
			2945 (16) 2954 (16) 3004 (17) 3214 (20) 3464 (21)
			3468 (21) 3523 (22) 3666 (23) 3786 (24) 3809 (24)
			3960 (27) 3976 (27) 4070 (30) 4076 (30) 4101 (31)
			4202 (34) 4238 (34) 4267 (34) 4270 (34) 4674 (39)
			4754 (40) 4759 (40) 4765 (40) 4783 (40) 4973 (41)
			4976 (41) 4980 (41) 5000 (41) 5100 (44) 5153 (45)
			5184 (45) 5225 (45) 5254 (45) 5351 (46) 5366 (46)
			5369 (46) 5394 (46) 5488 (47) 5604 (47) 5607 (47)
			5785 (49) 5791 (49) 5997 (51) 6012 (51) 6018 (51)
			6027 (51) 6035 (51) 6049 (51) 6054 (51) 6062 (51)
			6072 (51) 6117 (52) 6125 (52) 6264 (54) 6575 (57)
			6625 (58) 6725 (59) 6942 (62) 6949 (62) 7072 (63)
			7079 (63) 7184 (64) 7274 (65) 7280 (65) 7286 (65)
			7292 (65) 7299 (65) 7305 (65) 7312 (65) 7315 (65)
			7334 (65) 7429 (66) 7433 (66) 7439 (66) 7445 (66)
			7449 (66) 7452 (66) 7485 (66) 7493 (66) 7499 (66)
			7512 (66) 7515 (66) 7526 (66) 7540 (66) 7543 (66)

ZZ-ECKAA-8.7 Cross reference
ECKAA
Cross reference

VAX-11/750 MICRO DIAGNOSTIC MONITOR

F 7
11-FEB-1986

Fiche 2 Frame F7

Sequence 289

11-FEB-1986 10:07:52 VAX/VMS Macro V04-00 Page 264
11-FEB-1986 10:06:37 [VAX750.MONITOR]ECKAA.MAR;811 (98)

CALL 1 856 (3)

7570 (66)	7573 (66)	7583 (66)	7594 (67)	7597 (67)
7605 (67)	7608 (67)	7612 (67)	7616 (67)	7624 (67)
7640 (67)	7650 (67)	7653 (67)	7662 (67)	7668 (67)
7674 (67)	7684 (67)	7689 (67)	7696 (67)	7708 (67)
7712 (67)	7731 (67)	7753 (67)	7952 (70)	7955 (70)
7972 (70)	7975 (70)	8272 (76)	8276 (76)	8295 (76)
8355 (76)	8360 (76)	8440 (79)	8445 (79)	9029 (87)
9156 (88)	9418 (90)	9451 (90)	9522 (92)	9527 (92)
9530 (92)	9572 (93)			
1377 (5)	1403 (5)	1425 (5)	2264 (7)	2266 (7)
2269 (7)	2270 (7)	2329 (7)	2343 (7)	2344 (7)
2349 (7)	2361 (7)	2362 (7)	2368 (7)	2374 (7)
2378 (7)	2379 (7)	2380 (7)	2381 (7)	2391 (7)
2397 (7)	2398 (7)	2471 (8)	2474 (8)	2475 (8)
2518 (9)	2523 (9)	2524 (9)	2568 (10)	2569 (10)
2570 (10)	2571 (10)	2572 (10)	2634 (11)	2636 (11)
2702 (12)	2708 (12)	2710 (12)	2748 (13)	2753 (13)
2801 (14)	2847 (14)	2997 (17)	2998 (17)	2999 (17)
3044 (18)	3054 (18)	3104 (19)	3113 (19)	3207 (20)
3226 (20)	3230 (20)	3237 (20)	3258 (20)	3260 (20)
3262 (20)	3278 (20)	3299 (20)	3305 (20)	3307 (20)
3308 (20)	3313 (20)	3317 (20)	3333 (20)	3354 (20)
3369 (20)	3376 (20)	3379 (20)	3403 (20)	3405 (20)
3606 (23)	3636 (23)	3637 (23)	3649 (23)	3650 (23)
3661 (23)	3811 (24)	3812 (24)	3813 (24)	3816 (24)
3817 (24)	3856 (24)	3862 (24)	3863 (24)	3971 (27)
4009 (28)	4011 (28)	4012 (28)	4040 (29)	4042 (29)
4043 (29)	4094 (31)	4099 (31)	4128 (32)	4129 (32)
4137 (32)	4157 (33)	4208 (34)	4209 (34)	4210 (34)
4212 (34)	4217 (34)	4218 (34)	4219 (34)	4229 (34)
4231 (34)	4275 (34)	4280 (34)	4317 (35)	4322 (35)
4326 (35)	4329 (35)	4335 (35)	4340 (35)	4347 (35)
4359 (35)	4363 (35)	4368 (35)	4376 (35)	4954 (41)
4956 (41)	4957 (41)	4960 (41)	4982 (41)	4983 (41)
4994 (41)	5004 (41)	5079 (43)	5175 (45)	5177 (45)
5188 (45)	5398 (46)	5418 (46)	5420 (46)	5480 (47)
5526 (47)	5538 (47)	5575 (47)	5576 (47)	5589 (47)
5591 (47)	5592 (47)	5601 (47)	5602 (47)	5617 (47)
5618 (47)	5627 (47)	5628 (47)	5650 (47)	5712 (48)
5818 (49)	5906 (50)	5911 (50)	5918 (50)	5930 (50)
6007 (51)	6025 (51)	6111 (52)	6122 (52)	6123 (52)
6170 (53)	6267 (54)	6341 (54)	6422 (56)	6427 (56)
6508 (56)	6545 (57)	6547 (57)	6654 (58)	6657 (58)
6720 (59)	6775 (60)	6825 (61)	6830 (61)	6844 (61)
6846 (61)	6911 (62)	6916 (62)	6922 (62)	6929 (62)
6937 (62)	6952 (62)	7001 (63)	7006 (63)	7012 (63)
7019 (63)	7024 (63)	7026 (63)	7067 (63)	7082 (63)
7139 (64)	7149 (64)	7175 (64)	7187 (64)	7463 (66)
7466 (66)	7474 (66)	7477 (66)	7490 (66)	7491 (66)
7507 (66)	7523 (66)	7529 (66)	7530 (66)	7555 (66)
7561 (66)	7562 (66)	7575 (66)	7577 (66)	7586 (66)
7642 (67)	7680 (67)	7681 (67)	7682 (67)	7686 (67)
7687 (67)	7693 (67)	7694 (67)	7726 (67)	7728 (67)
7735 (67)	7736 (67)	7737 (67)	7738 (67)	7739 (67)
7749 (67)	7750 (67)	7756 (67)	7760 (67)	7765 (67)

CC 1
 CHECKRP 1 856 (3)
 1375 (5)

7771	(67)	7774	(67)	7780	(67)	7781	(67)	7784	(67)
7790	(67)	7791	(67)	7792	(67)	7828	(68)	7829	(68)
7907	(70)	7909	(70)	7959	(70)	7980	(70)	7994	(70)
8031	(71)	8040	(71)	8043	(71)	8107	(72)	8139	(73)
8246	(76)	8260	(76)	8261	(76)	8262	(76)	8363	(76)
8391	(77)	8420	(78)	8424	(78)	8426	(78)	8442	(79)
8452	(79)	8455	(79)	8457	(79)	8460	(79)	8845	(86)
8916	(87)	8917	(87)	8930	(87)	8934	(87)	8935	(87)
8939	(87)	8945	(87)	8946	(87)	8947	(87)	8951	(87)
8952	(87)	8956	(87)	8957	(87)	8959	(87)	8967	(87)
8974	(87)	8977	(87)	8984	(87)	8987	(87)	8994	(87)
9007	(87)	9012	(87)	9042	(87)	9043	(87)	9046	(87)
9050	(87)	9076	(87)	9089	(87)	9090	(87)	9091	(87)
9099	(87)	9138	(88)	9139	(88)	9140	(88)	9150	(88)
9151	(88)	9165	(88)	9274	(89)	9293	(89)	9356	(89)
9628	(94)	9629	(94)	9633	(94)	9642	(94)	9654	(94)
9655	(94)	9656	(94)	9665	(94)	9804	(95)	9810	(95)
9841	(96)	9843	(96)	9845	(96)	9846	(96)	9864	(97)
9866	(97)	9867	(97)	9898	(98)	9900	(98)	9902	(98)
9903	(98)								
1430	(5)	5819	(49)						
1375	(5)	1384	(5)	1386	(5)	1387	(5)	1388	(5)
1395	(5)	1396	(5)	1397	(5)	1400	(5)	1402	(5)
1404	(5)	1407	(5)	1408	(5)	1429	(5)	2320	(7)
2469	(8)	2630	(11)	2695	(12)	2703	(12)	2800	(14)
2817	(14)	2820	(14)	2825	(14)	2826	(14)	2827	(14)
2835	(14)	2841	(14)	2848	(14)	3043	(18)	3055	(18)
3058	(18)	3063	(18)	3103	(19)	3114	(19)	3117	(19)
3122	(19)	3213	(20)	3216	(20)	3224	(20)	3225	(20)
3229	(20)	3231	(20)	3236	(20)	3238	(20)	3242	(20)
3247	(20)	3248	(20)	3251	(20)	3257	(20)	3263	(20)
3271	(20)	3272	(20)	3278	(20)	3282	(20)	3284	(20)
3286	(20)	3287	(20)	3291	(20)	3296	(20)	3297	(20)
3298	(20)	3308	(20)	3317	(20)	3318	(20)	3326	(20)
3327	(20)	3333	(20)	3337	(20)	3339	(20)	3341	(20)
3342	(20)	3348	(20)	3353	(20)	3359	(20)	3367	(20)
3374	(20)	3375	(20)	3380	(20)	3394	(20)	3406	(20)
3412	(20)	3413	(20)	3414	(20)	3415	(20)	3423	(20)
3425	(20)	3462	(21)	3466	(21)	3467	(21)	3473	(21)
3475	(21)	3478	(21)	3479	(21)	3481	(21)	3482	(21)
3485	(21)	3525	(22)	3605	(23)	3607	(23)	3608	(23)
3611	(23)	3612	(23)	3618	(23)	3626	(23)	3631	(23)
3633	(23)	3641	(23)	3643	(23)	3644	(23)	3647	(23)
3648	(23)	3654	(23)	3655	(23)	3683	(23)	3684	(23)
3690	(23)	3691	(23)	3693	(23)	3694	(23)	3706	(23)
3707	(23)	3709	(23)	3710	(23)	3717	(23)	3720	(23)
3722	(23)	3723	(23)	3829	(24)	3836	(24)	3967	(27)
3985	(27)	4004	(28)	4005	(28)	4006	(28)	4007	(28)
4012	(28)	4013	(28)	4014	(28)	4015	(28)	4016	(28)
4035	(29)	4036	(29)	4037	(29)	4038	(29)	4043	(29)
4044	(29)	4045	(29)	4046	(29)	4047	(29)	4125	(32)
4126	(32)	4127	(32)	4130	(32)	4131	(32)	4132	(32)
4133	(32)	4134	(32)	4155	(33)	4158	(33)	4243	(34)
4248	(34)	4253	(34)	4281	(34)	4312	(35)	4355	(35)
4375	(35)	4377	(35)	4445	(36)	4449	(36)	4451	(36)

ZZ-ECKAA-8.7 Cross reference
ECKAA
Cross reference

H 7
11-FEB-1986
VAX-11/750 MICRO DIAGNOSTIC MONITOR

Fiche 2 Frame H7
11-FEB-1986 10:07:52 VAX/VMS Macro V04-00
11-FEB-1986 10:06:37 [VAX750.MONITOR]ECKAA.MAR;811
Sequence 291
Page 266
(98)

4461	(36)	4462	(36)	4469	(36)	4518	(37)	4526	(37)
4568	(38)	4571	(38)	4573	(38)	4619	(39)	4621	(39)
4623	(39)	4633	(39)	4649	(39)	4653	(39)	4655	(39)
4658	(39)	4660	(39)	4666	(39)	4676	(39)	4686	(39)
4691	(39)	4692	(39)	4696	(39)	4784	(40)	4842	(40)
4845	(40)	4858	(40)	4861	(40)	4879	(40)	4882	(40)
4990	(41)	4991	(41)	4992	(41)	4995	(41)	4996	(41)
5002	(41)	5003	(41)	5005	(41)	5006	(41)	5154	(45)
5169	(45)	5172	(45)	5173	(45)	5185	(45)	5210	(45)
5211	(45)	5214	(45)	5217	(45)	5226	(45)	5246	(45)
5247	(45)	5250	(45)	5283	(45)	5284	(45)	5288	(45)
5352	(46)	5378	(46)	5379	(46)	5383	(46)	5395	(46)
5412	(46)	5415	(46)	5416	(46)	5478	(47)	5484	(47)
5489	(47)	5524	(47)	5535	(47)	5576	(47)	5585	(47)
5592	(47)	5602	(47)	5618	(47)	5628	(47)	5800	(49)
5801	(49)	5804	(49)	5826	(49)	5833	(49)	5838	(49)
5839	(49)	5851	(49)	5854	(49)	5858	(49)	5920	(50)
5923	(50)	6175	(53)	6176	(53)	6215	(53)	6218	(53)
6268	(54)	6269	(54)	6270	(54)	6271	(54)	6279	(54)
6280	(54)	6281	(54)	6282	(54)	6285	(54)	6291	(54)
6292	(54)	6293	(54)	6295	(54)	6297	(54)	6299	(54)
6303	(54)	6304	(54)	6306	(54)	6307	(54)	6309	(54)
6319	(54)	6325	(54)	6330	(54)	6333	(54)	6344	(54)
6429	(56)	6438	(56)	6441	(56)	6442	(56)	6449	(56)
6450	(56)	6453	(56)	6454	(56)	6455	(56)	6456	(56)
6460	(56)	6464	(56)	6465	(56)	6478	(56)	6480	(56)
6493	(56)	6498	(56)	6499	(56)	6502	(56)	6507	(56)
6552	(57)	6555	(57)	6569	(57)	6577	(57)	6580	(57)
6581	(57)	6583	(57)	6655	(58)	6661	(58)	6687	(58)
6835	(61)	6838	(61)	6839	(61)	6848	(61)	6851	(61)
6855	(61)	6861	(61)	6862	(61)	6924	(62)	6927	(62)
7014	(63)	7017	(63)	7028	(63)	7031	(63)	7035	(63)
7042	(63)	7043	(63)	7049	(63)	7056	(63)	7057	(63)
7143	(64)	7147	(64)	7154	(64)	7164	(64)	7165	(64)
7171	(64)	7177	(64)	7197	(64)	7324	(65)	7743	(67)
7746	(67)	7747	(67)	7767	(67)	7768	(67)	7769	(67)
7776	(67)	7777	(67)	7778	(67)	7786	(67)	7787	(67)
7788	(67)	7861	(69)	7914	(70)	7917	(70)	7929	(70)
7930	(70)	7937	(70)	7941	(70)	7949	(70)	7957	(70)
7960	(70)	7961	(70)	7964	(70)	7977	(70)	7978	(70)
7981	(70)	7982	(70)	7985	(70)	7987	(70)	7989	(70)
7990	(70)	7992	(70)	8032	(71)	8045	(71)	8048	(71)
8050	(71)	8060	(71)	8062	(71)	8068	(71)	8076	(71)
8077	(71)	8108	(72)	8110	(72)	8143	(73)	8145	(73)
8248	(76)	8303	(76)	8308	(76)	8310	(76)	8313	(76)
8314	(76)	8316	(76)	8317	(76)	8320	(76)	8321	(76)
8322	(76)	8325	(76)	8326	(76)	8327	(76)	8334	(76)
8337	(76)	8340	(76)	8343	(76)	8390	(77)	8454	(79)
8456	(79)	8503	(80)	8505	(80)	8560	(81)	8565	(81)
8573	(81)	8575	(81)	8630	(82)	8675	(83)	8679	(83)
8682	(83)	8683	(83)	8728	(84)	8732	(84)	8735	(84)
8736	(84)	8790	(85)	8791	(85)	8805	(85)	8806	(85)
8915	(87)	8924	(87)	8927	(87)	8928	(87)	8969	(87)
8970	(87)	8971	(87)	8979	(87)	8980	(87)	8981	(87)
8989	(87)	8990	(87)	8991	(87)	8998	(87)	9002	(87)

				9005 (87)	9016 (87)	9020 (87)	9022 (87)	9024 (87)
				9034 (87)	9040 (87)	9041 (87)	9044 (87)	9045 (87)
				9047 (87)	9048 (87)	9051 (87)	9052 (87)	9053 (87)
				9056 (87)	9061 (87)	9069 (87)	9073 (87)	9074 (87)
				9075 (87)	9077 (87)	9079 (87)	9084 (87)	9086 (87)
				9092 (87)	9093 (87)	9095 (87)	9096 (87)	9098 (87)
				9144 (88)	9145 (88)	9148 (88)	9212 (89)	9213 (89)
				9215 (89)	9218 (89)	9219 (89)	9220 (89)	9221 (89)
				9228 (89)	9229 (89)	9235 (89)	9252 (89)	9273 (89)
				9281 (89)	9282 (89)	9283 (89)	9285 (89)	9301 (89)
				9302 (89)	9303 (89)	9307 (89)	9311 (89)	9316 (89)
				9319 (89)	9335 (89)	9336 (89)	9342 (89)	9345 (89)
				9346 (89)	9349 (89)	9350 (89)	9392 (90)	9405 (90)
				9406 (90)	9412 (90)	9413 (90)	9416 (90)	9429 (90)
				9430 (90)	9440 (90)	9441 (90)	9447 (90)	9448 (90)
				9449 (90)	9481 (91)	9488 (91)	9584 (93)	9625 (94)
				9626 (94)	9627 (94)	9630 (94)	9631 (94)	9632 (94)
				9634 (94)	9635 (94)	9641 (94)	9644 (94)	9653 (94)
				9657 (94)	9658 (94)	9660 (94)	9661 (94)	9663 (94)
				9664 (94)	9666 (94)	9667 (94)	9710 (95)	9715 (95)
				9720 (95)	9724 (95)	9725 (95)	9732 (95)	9733 (95)
				9734 (95)	9738 (95)	9739 (95)	9740 (95)	9744 (95)
				9745 (95)	9746 (95)	9750 (95)	9751 (95)	9752 (95)
				9753 (95)	9754 (95)	9762 (95)	9763 (95)	9767 (95)
				9769 (95)	9776 (95)	9777 (95)	9778 (95)	9779 (95)
				9787 (95)	9788 (95)	9802 (95)	9803 (95)	9805 (95)
				9806 (95)	9808 (95)	9809 (95)	9840 (96)	9846 (96)
				9863 (97)	9865 (97)	9897 (98)	9903 (98)	
CM	1	856	(3)					
CMA	1	856	(3)	3049 (18)	3362 (20)	3365 (20)	3418 (20)	3421 (20)
				4959 (41)	5401 (46)	6496 (56)	7927 (70)	9403 (90)
				9438 (90)				
CMC	1	856	(3)	3487 (21)	3726 (23)	4392 (35)	4471 (36)	4534 (37)
				4698 (39)	4898 (40)	5012 (41)	5060 (42)	5108 (44)
				5297 (45)	5426 (46)	5657 (47)	5718 (48)	8585 (81)
				8795 (85)	9167 (88)	9544 (92)		
CMP	1	856	(3)	2385 (7)	3386 (20)	3610 (23)	4435 (36)	5811 (49)
				5815 (49)	6023 (51)	6043 (51)	6185 (53)	6193 (53)
				6296 (54)	6308 (54)	6663 (58)	6666 (58)	7322 (65)
				7326 (65)	7553 (66)	7627 (67)	7716 (67)	7826 (68)
				8305 (76)	8309 (76)	8315 (76)	8323 (76)	8788 (85)
				8803 (85)	9541 (92)	9586 (93)	9590 (93)	
CNC	1	856	(3)					
CNZ	1	856	(3)	5998 (51)	7281 (65)	7293 (65)	7306 (65)	9064 (87)
CP	1	856	(3)					
CPE	1	856	(3)					
CPI	1	856	(3)	2341 (7)	2355 (7)	2521 (9)	2746 (13)	3244 (20)
				3293 (20)	3351 (20)	3851 (24)	4214 (34)	4227 (34)
				4345 (35)	4350 (35)	4357 (35)	4361 (35)	4447 (36)
				4515 (37)	4517 (37)	4524 (37)	4531 (37)	4629 (39)
				4631 (39)	4636 (39)	4643 (39)	4785 (40)	4790 (40)
				4792 (40)	4794 (40)	4796 (40)	4798 (40)	4800 (40)
				4852 (40)	4949 (41)	4965 (41)	5155 (45)	5157 (45)
				5159 (45)	5161 (45)	5163 (45)	5186 (45)	5197 (45)
				5208 (45)	5227 (45)	5230 (45)	5233 (45)	5236 (45)

				5244	(45)	5278	(45)	5353	(46)	5355	(46)	5357	(46)
				5359	(46)	5376	(46)	5396	(46)	5490	(47)	5492	(47)
				5494	(47)	5496	(47)	5498	(47)	5500	(47)	5502	(47)
				5504	(47)	5506	(47)	5508	(47)	5510	(47)	5512	(47)
				5514	(47)	5522	(47)	5533	(47)	5583	(47)	5706	(48)
				5829	(49)	6571	(57)	6630	(58)	6632	(58)	6634	(58)
				6672	(58)	6679	(58)	7516	(66)	7518	(66)	7564	(66)
				7654	(67)	7656	(67)	8250	(76)	8255	(76)	8282	(76)
				8418	(78)	8422	(78)	8450	(79)	8577	(81)	8580	(81)
				8838	(86)	9153	(88)	9575	(93)	9578	(93)	9637	(94)
				9639	(94)								
CPO	1	856	(3)										
CZ	1	856	(3)	7300	(65)								
DAD	1	856	(3)	1375	(5)	1386	(5)	1407	(5)	2320	(7)	2469	(8)
				2820	(14)	2825	(14)	3478	(21)	3631	(23)	3691	(23)
				3720	(23)	3723	(23)	3829	(24)	3836	(24)	4243	(34)
				4842	(40)	4858	(40)	4879	(40)	5169	(45)	5173	(45)
				5412	(46)	5416	(46)	5524	(47)	5535	(47)	5585	(47)
				5800	(49)	5804	(49)	5851	(49)	5920	(50)	5923	(50)
				6279	(54)	6429	(56)	6441	(56)	6464	(56)	6552	(57)
				6555	(57)	6687	(58)	6835	(61)	6838	(61)	6848	(61)
				6851	(61)	6924	(62)	6927	(62)	7014	(63)	7017	(63)
				7028	(63)	7031	(63)	7143	(64)	7165	(64)	7743	(67)
				7746	(67)	7914	(70)	7917	(70)	7941	(70)	7961	(70)
				7964	(70)	7982	(70)	7985	(70)	8045	(71)	8048	(71)
				8060	(71)	8143	(73)	8248	(76)	8303	(76)	8334	(76)
				8390	(77)	8505	(80)	8560	(81)	8630	(82)	8924	(87)
				8927	(87)	9084	(87)	9144	(88)	9148	(88)	9252	(89)
				9273	(89)	9301	(89)	9416	(90)	9653	(94)	9710	(95)
				9806	(95)	9809	(95)						
DCR	1	856	(3)	1409	(5)	2318	(7)	2836	(14)	2842	(14)	2900	(15)
				3042	(18)	3102	(19)	3218	(20)	3273	(20)	3283	(20)
				3328	(20)	3338	(20)	3372	(20)	3530	(22)	3656	(23)
				3681	(23)	3805	(24)	4135	(32)	4160	(33)	4459	(36)
				4661	(39)	4667	(39)	4693	(39)	4997	(41)	5809	(49)
				5834	(49)	6177	(53)	6437	(56)	6451	(56)	6481	(56)
				6500	(56)	6584	(57)	6863	(61)	7044	(63)	7058	(63)
				7160	(64)	7178	(64)	7200	(64)	7495	(66)	7691	(67)
				7711	(67)	7733	(67)	7931	(70)	8328	(76)	8506	(80)
				8628	(82)	8684	(83)	8737	(84)	8792	(85)	8807	(85)
				9021	(87)	9038	(87)	9054	(87)	9094	(87)	9230	(89)
				9329	(89)	9347	(89)	9407	(90)	9442	(90)	9484	(91)
				9659	(94)	9771	(95)	9780	(95)				
DCX	1	856	(3)	3287	(20)	3342	(20)	3611	(23)	3612	(23)	4451	(36)
				4692	(39)	4990	(41)	6280	(54)	6281	(54)	6282	(54)
				6285	(54)	6303	(54)	6304	(54)	6453	(56)	6454	(56)
				6455	(56)	6456	(56)	8310	(76)	8316	(76)	8317	(76)
				8325	(76)	8326	(76)	8327	(76)	8575	(81)	9584	(93)
				9667	(94)	9752	(95)	9753	(95)	9754	(95)	9776	(95)
				9777	(95)	9778	(95)	9779	(95)				
EQUATE	2	856	(3)	856	(3)								
EQUATE SYMBOLS	21	863	(4)	863	(4)								
EXECUTE	2	818	(2)	3278	(20)	3308	(20)	3317	(20)	3333	(20)	4012	(28)
				4043	(29)	5576	(47)	5592	(47)	5602	(47)	5618	(47)
				5628	(47)	9846	(96)	9903	(98)				

GENERATE_TITLE	1	843	(2)	1468	(6)								
HLT	1	856	(3)										
INR	1	856	(3)	2339	(7)	3064	(18)	3123	(19)	3280	(20)	3382	(20)
				3799	(24)	3823	(24)	3842	(24)	4159	(33)	4443	(36)
				4444	(36)	5807	(49)	6435	(56)	7159	(64)	7502	(66)
				7859	(69)	8348	(76)	8850	(86)	9018	(87)	9238	(89)
				9262	(89)	9290	(89)	9325	(89)	9327	(89)	9782	(95)
INX	1	856	(3)	1387	(5)	1388	(5)	2630	(11)	2826	(14)	2835	(14)
				2841	(14)	3216	(20)	3242	(20)	3247	(20)	3271	(20)
				3272	(20)	3291	(20)	3296	(20)	3326	(20)	3327	(20)
				3348	(20)	3353	(20)	3359	(20)	3367	(20)	3394	(20)
				3412	(20)	3413	(20)	3415	(20)	3423	(20)	3525	(22)
				3607	(23)	3608	(23)	3633	(23)	3644	(23)	3647	(23)
				3648	(23)	3694	(23)	4133	(32)	4134	(32)	4248	(34)
				4312	(35)	4355	(35)	4445	(36)	4449	(36)	4461	(36)
				4462	(36)	4469	(36)	4518	(37)	4526	(37)	4571	(38)
				4619	(39)	4621	(39)	4623	(39)	4633	(39)	4660	(39)
				4691	(39)	4784	(40)	4845	(40)	4861	(40)	4882	(40)
				5154	(45)	5172	(45)	5185	(45)	5210	(45)	5211	(45)
				5214	(45)	5217	(45)	5226	(45)	5246	(45)	5247	(45)
				5250	(45)	5283	(45)	5284	(45)	5288	(45)	5352	(46)
				5378	(46)	5379	(46)	5383	(46)	5395	(46)	5415	(46)
				5489	(47)	5801	(49)	5826	(49)	5833	(49)	5838	(49)
				5839	(49)	5854	(49)	6175	(53)	6176	(53)	6215	(53)
				6218	(53)	6268	(54)	6269	(54)	6270	(54)	6271	(54)
				6291	(54)	6292	(54)	6293	(54)	6295	(54)	6306	(54)
				6307	(54)	6330	(54)	6333	(54)	6344	(54)	6449	(56)
				6450	(56)	6478	(56)	6480	(56)	6498	(56)	6499	(56)
				6581	(57)	6583	(57)	6861	(61)	6862	(61)	7042	(63)
				7043	(63)	7056	(63)	7057	(63)	7164	(64)	7171	(64)
				7177	(64)	7197	(64)	7324	(65)	7747	(67)	7769	(67)
				7778	(67)	7788	(67)	7929	(70)	7930	(70)	8076	(71)
				8077	(71)	8108	(72)	8110	(72)	8145	(73)	8308	(76)
				8313	(76)	8314	(76)	8320	(76)	8321	(76)	8322	(76)
				8337	(76)	8340	(76)	8343	(76)	8503	(80)	8565	(81)
				8573	(81)	8682	(83)	8683	(83)	8735	(84)	8736	(84)
				8790	(85)	8791	(85)	8805	(85)	8806	(85)	8928	(87)
				8971	(87)	8981	(87)	8991	(87)	9053	(87)	9061	(87)
				9069	(87)	9074	(87)	9086	(87)	9096	(87)	9145	(88)
				9228	(89)	9229	(89)	9345	(89)	9346	(89)	9405	(90)
				9406	(90)	9440	(90)	9441	(90)	9448	(90)	9449	(90)
				9626	(94)	9661	(94)	9715	(95)	9720	(95)	9732	(95)
				9733	(95)	9734	(95)	9738	(95)	9739	(95)	9740	(95)
				9744	(95)	9745	(95)	9746	(95)	9767	(95)	9769	(95)
JC	1	856	(3)	2330	(7)	3227	(20)	4259	(34)	4318	(35)	4336	(35)
				4348	(35)	4360	(35)	4364	(35)	4369	(35)	5189	(45)
				5399	(46)	5812	(49)	5816	(49)	6320	(54)	7155	(64)
				7176	(64)	7766	(67)	7775	(67)	7785	(67)	8041	(71)
				8256	(76)	8968	(87)	8978	(87)	8988	(87)	9587	(93)
				9591	(93)	9638	(94)						
JM	1	856	(3)	2319	(7)	3687	(23)	4434	(36)	4635	(39)	4637	(39)
				4642	(39)	4850	(40)	5207	(45)	5705	(48)	5830	(49)
				5835	(49)	6315	(54)	6629	(58)	6782	(60)	7517	(66)
				7519	(66)	7655	(67)	7923	(70)	8553	(81)	8629	(82)
				9039	(87)	9261	(89)	9289	(89)	9355	(89)	9485	(91)

JMP	1	856	(3)	1373 (5)	1431 (5)	2284 (7)	2297 (7)	2321 (7)
				2333 (7)	2345 (7)	2363 (7)	2399 (7)	2476 (8)
				2525 (9)	2709 (12)	3396 (20)	3426 (20)	3484 (21)
				3673 (23)	3695 (23)	3702 (23)	3792 (24)	3797 (24)
				3807 (24)	3864 (24)	4220 (34)	4232 (34)	4282 (34)
				4330 (35)	4380 (35)	4450 (36)	4452 (36)	4463 (36)
				4527 (37)	4679 (39)	4808 (40)	4816 (40)	4824 (40)
				4832 (40)	4873 (40)	4883 (40)	4961 (41)	5007 (41)
				5178 (45)	5218 (45)	5258 (45)	5387 (46)	5406 (46)
				5516 (47)	5527 (47)	5539 (47)	5545 (47)	5551 (47)
				5557 (47)	5563 (47)	5569 (47)	5577 (47)	5595 (47)
				5611 (47)	5621 (47)	5640 (47)	5651 (47)	5840 (49)
				6030 (51)	6057 (51)	6201 (53)	6206 (53)	6211 (53)
				6321 (54)	6483 (56)	6579 (57)	6642 (58)	6649 (58)
				6674 (58)	7166 (64)	7467 (66)	7576 (66)	7578 (66)
				7587 (66)	7643 (67)	7701 (67)	7793 (67)	8051 (71)
				8063 (71)	8253 (76)	8263 (76)	8350 (76)	8357 (76)
				8421 (78)	8425 (78)	8453 (79)	8631 (82)	8809 (85)
				9159 (88)	9248 (89)	9263 (89)	9291 (89)	9487 (91)
				9643 (94)				
JNC	1	856	(3)	2351 (7)	2392 (7)	2472 (8)	2519 (9)	2635 (11)
				2745 (13)	3377 (20)	4124 (32)	4213 (34)	4230 (34)
				4327 (35)	4378 (35)	6194 (53)	6210 (53)	6326 (54)
				6938 (62)	7068 (63)	8252 (76)	9640 (94)	
JNZ	1	856	(3)	1410 (5)	2342 (7)	2837 (14)	2843 (14)	2901 (15)
				2952 (16)	3215 (20)	3274 (20)	3285 (20)	3329 (20)
				3340 (20)	3387 (20)	3469 (21)	3531 (22)	3667 (23)
				3781 (24)	3852 (24)	4136 (32)	4161 (33)	4268 (34)
				4351 (35)	4358 (35)	4448 (36)	4519 (37)	4525 (37)
				4662 (39)	4668 (39)	4694 (39)	4801 (40)	4840 (40)
				4950 (41)	4966 (41)	5001 (41)	5164 (45)	5198 (45)
				5237 (45)	5360 (46)	5813 (49)	5817 (49)	6024 (51)
				6044 (51)	6050 (51)	6178 (53)	6186 (53)	6298 (54)
				6452 (56)	6472 (56)	6482 (56)	6501 (56)	6573 (57)
				6585 (57)	6648 (58)	6664 (58)	6667 (58)	6864 (61)
				7045 (63)	7059 (63)	7179 (64)	7316 (65)	7323 (65)
				7327 (65)	7440 (66)	7453 (66)	7456 (66)	7500 (66)
				7513 (66)	7537 (66)	7544 (66)	7554 (66)	7565 (66)
				7609 (67)	7614 (67)	7628 (67)	7663 (67)	7717 (67)
				7932 (70)	8306 (76)	8311 (76)	8318 (76)	8329 (76)
				8419 (78)	8423 (78)	8451 (79)	8507 (80)	8578 (81)
				8581 (81)	8685 (83)	8738 (84)	8786 (85)	8789 (85)
				8793 (85)	8804 (85)	8808 (85)	8839 (86)	8942 (87)
				9023 (87)	9097 (87)	9231 (89)	9317 (89)	9331 (89)
				9348 (89)	9352 (89)	9408 (90)	9443 (90)	9456 (90)
				9542 (92)	9576 (93)	9588 (93)	9662 (94)	9770 (95)
				9772 (95)	9781 (95)			
JP	1	856	(3)	2305 (7)	3701 (23)	4644 (39)	5707 (48)	6277 (54)
				7657 (67)	9055 (87)	9246 (89)		
JPE	1	856	(3)	9401 (90)	9436 (90)			
JPO	1	856	(3)					
JZ	1	856	(3)	2279 (7)	2290 (7)	2357 (7)	2386 (7)	2522 (9)
				2705 (12)	2747 (13)	3219 (20)	3245 (20)	3294 (20)
				3352 (20)	3402 (20)	3465 (21)	3613 (23)	3657 (23)
				3682 (23)	3787 (24)	3803 (24)	3810 (24)	4120 (32)

LDA 1 856 (3)

4215	(34)	4228	(34)	4271	(34)	4346	(35)	4362	(35)
4436	(36)	4460	(36)	4516	(37)	4532	(37)	4630	(39)
4632	(39)	4786	(40)	4791	(40)	4793	(40)	4795	(40)
4797	(40)	4799	(40)	4854	(40)	4998	(41)	5156	(45)
5158	(45)	5160	(45)	5162	(45)	5187	(45)	5209	(45)
5228	(45)	5231	(45)	5234	(45)	5245	(45)	5279	(45)
5354	(46)	5356	(46)	5358	(46)	5377	(46)	5397	(46)
5491	(47)	5493	(47)	5495	(47)	5497	(47)	5499	(47)
5501	(47)	5503	(47)	5505	(47)	5507	(47)	5509	(47)
5511	(47)	5513	(47)	5515	(47)	5523	(47)	5534	(47)
5584	(47)	5786	(49)	5792	(49)	5808	(49)	6019	(51)
6036	(51)	6118	(52)	6195	(53)	6436	(56)	6631	(58)
6633	(58)	6635	(58)	6641	(58)	6673	(58)	6680	(58)
7161	(64)	7275	(65)	7287	(65)	7313	(65)	7446	(66)
7450	(66)	7486	(66)	7541	(66)	7574	(66)	7617	(67)
7641	(67)	7651	(67)	7669	(67)	7675	(67)	7685	(67)
7827	(68)	7953	(70)	7973	(70)	8035	(71)	8251	(76)
8273	(76)	8283	(76)	8299	(76)	8324	(76)	8441	(79)
8446	(79)	8837	(86)	8955	(87)	9001	(87)	9030	(87)
9072	(87)	9154	(88)	9423	(90)	9573	(93)	9579	(93)
1417	(5)	2276	(7)	2317	(7)	2338	(7)	2382	(7)
2384	(7)	2749	(13)	2799	(14)	2802	(14)	2804	(14)
2887	(15)	2891	(15)	2895	(15)	2902	(15)	2937	(16)
2940	(16)	2944	(16)	2953	(16)	3003	(17)	3041	(18)
3045	(18)	3048	(18)	3051	(18)	3101	(19)	3105	(19)
3108	(19)	3110	(19)	3279	(20)	3381	(20)	3385	(20)
3463	(21)	3522	(22)	3535	(22)	3609	(23)	3665	(23)
3677	(23)	3779	(24)	3785	(24)	3791	(24)	3798	(24)
3804	(24)	3808	(24)	3841	(24)	3850	(24)	3890	(25)
3915	(26)	3955	(27)	3975	(27)	4069	(30)	4075	(30)
4095	(31)	4100	(31)	4201	(34)	4226	(34)	4237	(34)
4753	(40)	4758	(40)	4763	(40)	4805	(40)	4813	(40)
4821	(40)	4829	(40)	4870	(40)	4894	(40)	4955	(41)
4958	(41)	4970	(41)	4975	(41)	5056	(42)	5098	(44)
5102	(44)	5190	(45)	5253	(45)	5264	(45)	5270	(45)
5365	(46)	5368	(46)	5403	(46)	5593	(47)	5603	(47)
5606	(47)	5609	(47)	5619	(47)	5629	(47)	5784	(49)
5790	(49)	5905	(50)	5910	(50)	5927	(50)	5929	(50)
5996	(51)	6011	(51)	6017	(51)	6020	(51)	6022	(51)
6026	(51)	6034	(51)	6040	(51)	6042	(51)	6048	(51)
6053	(51)	6061	(51)	6071	(51)	6075	(51)	6116	(52)
6124	(52)	6169	(53)	6182	(53)	6184	(53)	6190	(53)
6192	(53)	6216	(53)	6219	(53)	6263	(54)	6266	(54)
6340	(54)	6421	(56)	6470	(56)	6544	(57)	6546	(57)
6562	(57)	6574	(57)	6624	(58)	6627	(58)	6653	(58)
6656	(58)	6671	(58)	6678	(58)	6724	(59)	6730	(59)
6771	(60)	6774	(60)	6779	(60)	6781	(60)	6824	(61)
6829	(61)	6843	(61)	6845	(61)	6856	(61)	6910	(62)
6915	(62)	6921	(62)	6928	(62)	6933	(62)	6936	(62)
6939	(62)	6948	(62)	7000	(63)	7005	(63)	7011	(63)
7018	(63)	7023	(63)	7025	(63)	7037	(63)	7051	(63)
7063	(63)	7066	(63)	7069	(63)	7078	(63)	7148	(64)
7183	(64)	7192	(64)	7273	(65)	7279	(65)	7285	(65)
7291	(65)	7298	(65)	7304	(65)	7310	(65)	7321	(65)
7325	(65)	7333	(65)	7424	(66)	7428	(66)	7432	(66)

				7438	(66)	7444	(66)	7447	(66)	7454	(66)	7478	(66)
				7484	(66)	7492	(66)	7498	(66)	7501	(66)	7511	(66)
				7514	(66)	7535	(66)	7538	(66)	7542	(66)	7550	(66)
				7552	(66)	7563	(66)	7569	(66)	7572	(66)	7582	(66)
				7593	(67)	7596	(67)	7603	(67)	7607	(67)	7610	(67)
				7615	(67)	7621	(67)	7634	(67)	7639	(67)	7647	(67)
				7649	(67)	7652	(67)	7661	(67)	7667	(67)	7673	(67)
				7683	(67)	7688	(67)	7695	(67)	7705	(67)	7710	(67)
				7715	(67)	7729	(67)	7823	(68)	7825	(68)	7858	(69)
				7906	(70)	7908	(70)	7921	(70)	7958	(70)	7979	(70)
				7991	(70)	7993	(70)	8030	(71)	8033	(71)	8039	(71)
				8106	(72)	8138	(73)	8171	(74)	8198	(75)	8241	(76)
				8270	(76)	8288	(76)	8290	(76)	8292	(76)	8294	(76)
				8304	(76)	8307	(76)	8312	(76)	8319	(76)	8335	(76)
				8338	(76)	8341	(76)	8344	(76)	8352	(76)	8359	(76)
				8417	(78)	8439	(79)	8444	(79)	8462	(79)	8835	(86)
				8849	(86)	8940	(87)	8953	(87)	9006	(87)	9017	(87)
				9028	(87)	9036	(87)	9152	(88)	9155	(88)	9217	(89)
				9244	(89)	9286	(89)	9324	(89)	9328	(89)	9353	(89)
				9538	(92)	9540	(92)	9571	(93)	9574	(93)	9577	(93)
				9585	(93)	9589	(93)	9595	(93)				
LDAX	1	856	(3)	4246	(34)	4249	(34)	4432	(36)	4656	(39)	4673	(39)
				4689	(39)	4948	(41)	6173	(53)	6273	(54)	6294	(54)
				6305	(54)	6313	(54)	6331	(54)	6334	(54)	6447	(56)
				6565	(57)	6858	(61)	7039	(63)	7053	(63)	8074	(71)
				8078	(71)	8787	(85)	8802	(85)	9226	(89)	9309	(89)
				9399	(90)	9434	(90)	9536	(92)	9729	(95)	9735	(95)
				9741	(95)	9747	(95)	9766	(95)				
LHLD	1	856	(3)	2267	(7)	2301	(7)	2331	(7)	2369	(7)	2465	(8)
				2477	(8)	2627	(11)	2806	(14)	3006	(17)	3391	(20)
				3474	(21)	3604	(23)	3814	(24)	3821	(24)	3828	(24)
				3835	(24)	3917	(26)	3983	(27)	4569	(38)	4687	(39)
				5105	(44)	5797	(49)	5799	(49)	5803	(49)	5806	(49)
				5823	(49)	5919	(50)	5922	(50)	6051	(51)	6068	(51)
				6372	(55)	6434	(56)	6440	(56)	6461	(56)	6463	(56)
				6551	(57)	6554	(57)	6564	(57)	6684	(58)	6686	(58)
				6789	(60)	6834	(61)	6837	(61)	6847	(61)	6850	(61)
				6909	(62)	6923	(62)	6926	(62)	6999	(63)	7013	(63)
				7016	(63)	7027	(63)	7030	(63)	7140	(64)	7142	(64)
				7331	(65)	7524	(66)	7727	(67)	7740	(67)	7742	(67)
				7745	(67)	7751	(67)	7830	(68)	7913	(70)	7916	(70)
				7938	(70)	7940	(70)	7950	(70)	7963	(70)	7970	(70)
				7984	(70)	8044	(71)	8047	(71)	8057	(71)	8059	(71)
				8140	(73)	8142	(73)	8387	(77)	8389	(77)	8416	(78)
				8921	(87)	8923	(87)	8926	(87)	9035	(87)	9141	(88)
				9143	(88)	9147	(88)	9216	(89)	9583	(93)		
LXI	1	856	(3)	1374	(5)	1385	(5)	1393	(5)	2269	(7)	2270	(7)
				2314	(7)	2361	(7)	2362	(7)	2378	(7)	2379	(7)
				2380	(7)	2381	(7)	2568	(10)	2569	(10)	2570	(10)
				2571	(10)	2572	(10)	2708	(12)	2710	(12)	2807	(14)
				2893	(15)	2942	(16)	3207	(20)	3230	(20)	3252	(20)
				3253	(20)	3260	(20)	3262	(20)	3267	(20)	3305	(20)
				3307	(20)	3313	(20)	3314	(20)	3323	(20)	3358	(20)
				3369	(20)	3403	(20)	3521	(22)	3606	(23)	3627	(23)
				3637	(23)	3650	(23)	3661	(23)	3679	(23)	3680	(23)

MOV

1

856 (3)

3811	(24)	3812	(24)	3813	(24)	3817	(24)	3856	(24)
3862	(24)	3863	(24)	3894	(25)	4009	(28)	4011	(28)
4040	(29)	4042	(29)	4122	(32)	4129	(32)	4137	(32)
4157	(33)	4209	(34)	4210	(34)	4217	(34)	4218	(34)
4219	(34)	4231	(34)	4241	(34)	4251	(34)	4254	(34)
4265	(34)	4272	(34)	4309	(35)	4311	(35)	4313	(35)
4334	(35)	4616	(39)	4652	(39)	4672	(39)	4685	(39)
4954	(41)	4957	(41)	4983	(41)	4987	(41)	4994	(41)
5004	(41)	5480	(47)	5481	(47)	5575	(47)	5589	(47)
5591	(47)	5601	(47)	5617	(47)	5627	(47)	5639	(47)
5645	(47)	5827	(49)	5849	(49)	5857	(49)	6171	(53)
6428	(56)	6494	(56)	6503	(56)	7036	(63)	7050	(63)
7131	(64)	7297	(65)	7320	(65)	7426	(66)	7490	(66)
7491	(66)	7507	(66)	7523	(66)	7529	(66)	7629	(67)
7680	(67)	7681	(67)	7682	(67)	7686	(67)	7687	(67)
7693	(67)	7726	(67)	7735	(67)	7736	(67)	7737	(67)
7738	(67)	7739	(67)	7749	(67)	7750	(67)	7756	(67)
7760	(67)	7761	(67)	7771	(67)	7780	(67)	7781	(67)
7790	(67)	7791	(67)	7924	(70)	7933	(70)	8069	(71)
8247	(76)	8260	(76)	8261	(76)	8262	(76)	8267	(76)
8346	(76)	8442	(79)	8443	(79)	8452	(79)	8455	(79)
8460	(79)	8497	(80)	8557	(81)	8671	(83)	8676	(83)
8724	(84)	8729	(84)	8782	(85)	8783	(85)	8845	(86)
8916	(87)	8917	(87)	8930	(87)	8934	(87)	8935	(87)
8939	(87)	8945	(87)	8946	(87)	8947	(87)	8951	(87)
8952	(87)	8956	(87)	8957	(87)	8958	(87)	8963	(87)
8973	(87)	8983	(87)	8993	(87)	9011	(87)	9042	(87)
9043	(87)	9049	(87)	9076	(87)	9081	(87)	9090	(87)
9091	(87)	9099	(87)	9139	(88)	9140	(88)	9150	(88)
9224	(89)	9249	(89)	9259	(89)	9272	(89)	9287	(89)
9337	(89)	9396	(90)	9431	(90)	9519	(92)	9628	(94)
9629	(94)	9633	(94)	9652	(94)	9655	(94)	9656	(94)
9665	(94)	9706	(95)	9726	(95)	9727	(95)	9728	(95)
9835	(96)	9836	(96)	9843	(96)	9845	(96)	9864	(97)
9867	(97)	9892	(98)	9893	(98)	9900	(98)	9902	(98)
2277	(7)	2280	(7)	2288	(7)	2294	(7)	2383	(7)
2467	(8)	2628	(11)	2805	(14)	2821	(14)	2833	(14)
2892	(15)	2897	(15)	2899	(15)	2941	(16)	2946	(16)
2948	(16)	2951	(16)	3050	(18)	3109	(19)	3212	(20)
3217	(20)	3223	(20)	3228	(20)	3243	(20)	3269	(20)
3292	(20)	3324	(20)	3349	(20)	3350	(20)	3354	(20)
3361	(20)	3363	(20)	3364	(20)	3366	(20)	3370	(20)
3373	(20)	3384	(20)	3400	(20)	3417	(20)	3419	(20)
3420	(20)	3422	(20)	3476	(21)	3524	(22)	3629	(23)
3632	(23)	3634	(23)	3642	(23)	3645	(23)	3678	(23)
3685	(23)	3688	(23)	3699	(23)	3718	(23)	3801	(24)
3822	(24)	3824	(24)	3833	(24)	4156	(33)	4239	(34)
4247	(34)	4250	(34)	4266	(34)	4269	(34)	4319	(35)
4321	(35)	4337	(35)	4339	(35)	4344	(35)	4349	(35)
4356	(35)	4446	(36)	4514	(37)	4523	(37)	4570	(38)
4628	(39)	4675	(39)	4690	(39)	4775	(40)	4777	(40)
4782	(40)	4838	(40)	4843	(40)	4847	(40)	4851	(40)
4859	(40)	4869	(40)	4880	(40)	4989	(41)	4999	(41)
5152	(45)	5170	(45)	5174	(45)	5183	(45)	5196	(45)
5205	(45)	5212	(45)	5215	(45)	5224	(45)	5243	(45)

5248	(45)	5251	(45)	5277	(45)	5285	(45)	5289	(45)
5350	(46)	5375	(46)	5380	(46)	5384	(46)	5393	(46)
5400	(46)	5402	(46)	5413	(46)	5417	(46)	5487	(47)
5521	(47)	5525	(47)	5532	(47)	5536	(47)	5582	(47)
5586	(47)	5716	(48)	5810	(49)	5814	(49)	5825	(49)
5828	(49)	5832	(49)	5836	(49)	5852	(49)	5853	(49)
5855	(49)	5856	(49)	5912	(50)	5913	(50)	5928	(50)
5930	(50)	6021	(51)	6041	(51)	6174	(53)	6183	(53)
6191	(53)	6200	(53)	6205	(53)	6217	(53)	6220	(53)
6274	(54)	6283	(54)	6286	(54)	6343	(54)	6345	(54)
6448	(56)	6477	(56)	6495	(56)	6563	(57)	6570	(57)
6576	(57)	6582	(57)	6626	(58)	6639	(58)	6646	(58)
6662	(58)	6665	(58)	6857	(61)	6917	(62)	6919	(62)
6935	(62)	7007	(63)	7009	(63)	7038	(63)	7052	(63)
7065	(63)	7150	(64)	7152	(64)	7162	(64)	7170	(64)
7199	(64)	7311	(65)	7314	(65)	7448	(66)	7451	(66)
7497	(66)	7504	(66)	7505	(66)	7525	(66)	7527	(66)
7539	(66)	7551	(66)	7611	(67)	7622	(67)	7623	(67)
7714	(67)	7752	(67)	7754	(67)	7824	(68)	7926	(70)
7951	(70)	7954	(70)	7956	(70)	7965	(70)	7966	(70)
7971	(70)	7974	(70)	7976	(70)	7986	(70)	7988	(70)
8075	(71)	8079	(71)	8109	(72)	8111	(72)	8144	(73)
8146	(73)	8242	(76)	8244	(76)	8249	(76)	8254	(76)
8271	(76)	8274	(76)	8275	(76)	8281	(76)	8287	(76)
8297	(76)	8336	(76)	8339	(76)	8342	(76)	8345	(76)
8347	(76)	8349	(76)	8448	(79)	8449	(79)	8502	(80)
8504	(80)	8551	(81)	8558	(81)	8564	(81)	8566	(81)
8572	(81)	8574	(81)	8576	(81)	8579	(81)	8624	(82)
8670	(83)	8680	(83)	8723	(84)	8733	(84)	8781	(85)
8784	(85)	8999	(87)	9037	(87)	9062	(87)	9070	(87)
9078	(87)	9082	(87)	9085	(87)	9087	(87)	9227	(89)
9236	(89)	9247	(89)	9250	(89)	9253	(89)	9294	(89)
9308	(89)	9312	(89)	9313	(89)	9315	(89)	9343	(89)
9402	(90)	9404	(90)	9417	(90)	9419	(90)	9420	(90)
9422	(90)	9425	(90)	9437	(90)	9439	(90)	9450	(90)
9452	(90)	9453	(90)	9455	(90)	9458	(90)	9482	(91)
9520	(92)	9521	(92)	9525	(92)	9529	(92)	9532	(92)
9535	(92)	9539	(92)	9636	(94)	9645	(94)	9650	(94)
9708	(95)	9714	(95)	9716	(95)	9719	(95)	9789	(95)
9793	(95)	9796	(95)	9797	(95)	9799	(95)	9801	(95)
1394	(5)	1398	(5)	1399	(5)	1405	(5)	1406	(5)
1414	(5)	1422	(5)	2283	(7)	2315	(7)	2316	(7)
2380	(7)	2468	(8)	2570	(10)	2572	(10)	2706	(12)
2818	(14)	2819	(14)	2823	(14)	2824	(14)	2828	(14)
2949	(16)	3000	(17)	3059	(18)	3066	(18)	3118	(19)
3125	(19)	3207	(20)	3254	(20)	3255	(20)	3260	(20)
3262	(20)	3268	(20)	3278	(20)	3305	(20)	3307	(20)
3308	(20)	3313	(20)	3317	(20)	3322	(20)	3333	(20)
3354	(20)	3369	(20)	3378	(20)	3393	(20)	3395	(20)
3404	(20)	3477	(21)	3520	(22)	3526	(22)	3606	(23)
3619	(23)	3630	(23)	3671	(23)	3672	(23)	3689	(23)
3719	(23)	3796	(24)	3812	(24)	3825	(24)	3834	(24)
3844	(24)	3862	(24)	3968	(27)	3969	(27)	3971	(27)
4009	(28)	4011	(28)	4012	(28)	4040	(29)	4042	(29)
4043	(29)	4066	(30)	4099	(31)	4121	(32)	4128	(32)

MVI

1

856 (3)

				4153	(33)	4154	(33)	4157	(33)	4219	(34)	4240	(34)
				4328	(35)	4430	(36)	4431	(36)	4650	(39)	4651	(39)
				4688	(39)	4769	(40)	4776	(40)	4841	(40)	4846	(40)
				4849	(40)	4853	(40)	4862	(40)	4988	(41)	4994	(41)
				5150	(45)	5151	(45)	5202	(45)	5229	(45)	5232	(45)
				5235	(45)	5241	(45)	5348	(46)	5349	(46)	5480	(47)
				5485	(47)	5486	(47)	5544	(47)	5550	(47)	5556	(47)
				5562	(47)	5568	(47)	5575	(47)	5576	(47)	5589	(47)
				5591	(47)	5592	(47)	5594	(47)	5601	(47)	5602	(47)
				5610	(47)	5617	(47)	5618	(47)	5620	(47)	5627	(47)
				5628	(47)	5630	(47)	5831	(49)	5850	(49)	5917	(50)
				6119	(52)	6172	(53)	6199	(53)	6204	(53)	6275	(54)
				6278	(54)	6342	(54)	6426	(56)	6446	(56)	6476	(56)
				6492	(56)	6504	(56)	6505	(56)	7124	(64)	7163	(64)
				7461	(66)	7464	(66)	7472	(66)	7475	(66)	7507	(66)
				7529	(66)	7575	(66)	7577	(66)	7682	(67)	7693	(67)
				7735	(67)	7737	(67)	7749	(67)	7756	(67)	7771	(67)
				7773	(67)	7781	(67)	7783	(67)	7791	(67)	7925	(70)
				8042	(71)	8245	(76)	8268	(76)	8269	(76)	8420	(78)
				8424	(78)	8426	(78)	8457	(79)	8498	(80)	8499	(80)
				8500	(80)	8501	(80)	8550	(81)	8559	(81)	8623	(82)
				8930	(87)	8935	(87)	8943	(87)	8945	(87)	8952	(87)
				8976	(87)	8986	(87)	9083	(87)	9150	(88)	9225	(89)
				9251	(89)	9338	(89)	9397	(90)	9398	(90)	9414	(90)
				9415	(90)	9432	(90)	9433	(90)	9483	(91)	9533	(92)
				9651	(94)	9709	(95)	9765	(95)	9794	(95)	9800	(95)
				9837	(96)	9838	(96)	9843	(96)	9845	(96)	9846	(96)
				9866	(97)	9894	(98)	9895	(98)	9900	(98)	9902	(98)
				9903	(98)								
NOP	1	856	(3)										
ORA	1	856	(3)	2304	(7)	3111	(19)	3401	(20)	3686	(23)	3700	(23)
				3780	(24)	3802	(24)	4119	(32)	4433	(36)	4677	(39)
				4839	(40)	4848	(40)	4868	(40)	4871	(40)	5191	(45)
				5206	(45)	5256	(45)	6276	(54)	6314	(54)	6471	(56)
				6578	(57)	6628	(58)	6640	(58)	6647	(58)	6780	(60)
				7455	(66)	7536	(66)	7922	(70)	8034	(71)	8298	(76)
				8552	(81)	8785	(85)	8836	(86)	8941	(87)	8954	(87)
				9000	(87)	9063	(87)	9071	(87)	9245	(89)	9310	(89)
				9351	(89)	9354	(89)	9421	(90)	9454	(90)		
ORI	1	856	(3)	1420	(5)	2281	(7)	2750	(13)	2904	(15)	2947	(16)
				2955	(16)	3891	(25)	3956	(27)	4071	(30)	4077	(30)
				4096	(31)	4320	(35)	4338	(35)	4756	(40)	4764	(40)
				4806	(40)	4814	(40)	4822	(40)	4830	(40)	4895	(40)
				4971	(41)	4977	(41)	5057	(42)	5099	(44)	5103	(44)
				5265	(45)	5271	(45)	6126	(52)	6772	(60)	6783	(60)
				6940	(62)	7070	(63)	7193	(64)	7479	(66)	7698	(67)
				7706	(67)	8172	(74)	8199	(75)	8353	(76)	8463	(79)
				9424	(90)	9457	(90)	9596	(93)				
PCHL	1	856	(3)	4255	(34)	5860	(49)						
POP	1	856	(3)	1397	(5)	1404	(5)	1408	(5)	1429	(5)	2703	(12)
				2827	(14)	2848	(14)	3055	(18)	3063	(18)	3114	(19)
				3122	(19)	3224	(20)	3231	(20)	3238	(20)	3263	(20)
				3278	(20)	3282	(20)	3286	(20)	3308	(20)	3317	(20)
				3318	(20)	3333	(20)	3337	(20)	3341	(20)	3374	(20)
				3380	(20)	3406	(20)	3425	(20)	3466	(21)	3479	(21)

ZZ-ECKAA-8.7	Cross reference	F 8											
ECKAA		11-FEB-1986						Fiche 2 Frame F8		Sequence 302			
Cross reference		VAX-11/750 MICRO DIAGNOSTIC MONITOR						11-FEB-1986 10:07:52		VAX/VMS Macro V04-00			
								11-FEB-1986 10:06:37		[VAX750.MONITOR]ECKAA.MAR;811			
													Page 277
													(98)
RAD50	2	610	(2)	1526	(6)	1527	(6)	1528	(6)	1529	(6)	1530	(6)
				1531	(6)	1532	(6)	1533	(6)	1534	(6)	1535	(6)
				1536	(6)	1537	(6)	1538	(6)	1539	(6)	1540	(6)
RAL	1	856	(3)	4657	(39)								
RAR	1	856	(3)	4123	(32)								
RC	1	856	(3)										
READ_RECORD	2	786	(2)	3207	(20)	3230	(20)	3369	(20)	3403	(20)	3606	(23)
				3661	(23)								
RET	1	856	(3)	1381	(5)	2478	(8)	2527	(9)	2573	(10)	2637	(11)
				2711	(12)	2754	(13)	2850	(14)	2907	(15)	2958	(16)
				3008	(17)	3069	(18)	3128	(19)	3407	(20)	3488	(21)
				3537	(22)	3621	(23)	3713	(23)	3727	(23)	3846	(24)
				3896	(25)	3919	(26)	3986	(27)	4017	(28)	4048	(29)
				4081	(30)	4105	(31)	4138	(32)	4162	(33)	4276	(34)
				4386	(35)	4393	(35)	4472	(36)	4478	(36)	4535	(37)
				4537	(37)	4574	(38)	4699	(39)	4705	(39)	4889	(40)
				4899	(40)	5013	(41)	5018	(41)	5061	(42)	5109	(44)
				5298	(45)	5304	(45)	5427	(46)	5433	(46)	5654	(47)
				5658	(47)	5661	(47)	5719	(48)	5725	(48)	5931	(50)
				6077	(51)	6129	(52)	6221	(53)	6336	(54)	6346	(54)
				6374	(55)	6512	(56)	6586	(57)	6689	(58)	6732	(59)
				6791	(60)	6868	(61)	6944	(62)	6953	(62)	7074	(63)
				7083	(63)	7188	(64)	7202	(64)	7339	(65)	7531	(66)
				7556	(66)	7721	(67)	7832	(68)	7862	(69)	7995	(70)
				8080	(71)	8112	(72)	8147	(73)	8174	(74)	8201	(75)
				8365	(76)	8392	(77)	8428	(78)	8465	(79)	8508	(80)
				8583	(81)	8586	(81)	8633	(82)	8686	(83)	8739	(84)
				8796	(85)	8798	(85)	8852	(86)	9103	(87)	9168	(88)
				9358	(89)	9459	(90)	9489	(91)	9545	(92)	9547	(92)
				9599	(93)	9668	(94)	9811	(95)	9847	(96)	9868	(97)
				9904	(98)								
RIM	1	856	(3)										
RLC	1	856	(3)	2698	(12)	3628	(23)	9080	(87)	9486	(91)	9526	(92)
				9646	(94)	9647	(94)	9648	(94)	9707	(95)	9790	(95)
				9791	(95)								
RM	1	856	(3)										
RNC	1	856	(3)										
RNZ	1	856	(3)										
RP	1	856	(3)										
RPE	1	856	(3)										
RPO	1	856	(3)										
RRC	1	856	(3)	5711	(48)								
RST	1	856	(3)										
RZ	1	856	(3)										
SBB	1	856	(3)										
SBI	1	856	(3)										
SGL_MIC_INSTR	1	707	(2)	3278	(20)	3308	(20)	3317	(20)	3333	(20)	3971	(27)
				4012	(28)	4043	(29)	4099	(31)	5576	(47)	5592	(47)
				5602	(47)	5618	(47)	5628	(47)	7575	(66)	9846	(96)
				9903	(98)								
SGL_TICK	1	730	(2)	7577	(66)								
SHLD	1	856	(3)	1376	(5)	1389	(5)	2269	(7)	2270	(7)	2302	(7)
				2325	(7)	2332	(7)	2361	(7)	2362	(7)	2370	(7)
				2378	(7)	2379	(7)	2380	(7)	2381	(7)	2463	(8)
				2466	(8)	2470	(8)	2568	(10)	2569	(10)	2570	(10)

2571	(10)	2572	(10)	2631	(11)	2708	(12)	2710	(12)
3260	(20)	3262	(20)	3305	(20)	3307	(20)	3313	(20)
3315	(20)	3316	(20)	3354	(20)	3360	(20)	3368	(20)
3392	(20)	3416	(20)	3424	(20)	3450	(21)	3636	(23)
3637	(23)	3649	(23)	3650	(23)	3724	(23)	3811	(24)
3812	(24)	3813	(24)	3815	(24)	3817	(24)	3826	(24)
3830	(24)	3837	(24)	3856	(24)	3862	(24)	3863	(24)
3895	(25)	3918	(26)	3970	(27)	3984	(27)	4009	(28)
4011	(28)	4040	(29)	4042	(29)	4128	(32)	4129	(32)
4137	(32)	4157	(33)	4209	(34)	4210	(34)	4217	(34)
4218	(34)	4219	(34)	4231	(34)	4310	(35)	4572	(38)
4695	(39)	4954	(41)	4957	(41)	4983	(41)	4994	(41)
5004	(41)	5106	(44)	5480	(47)	5482	(47)	5483	(47)
5575	(47)	5589	(47)	5591	(47)	5601	(47)	5617	(47)
5627	(47)	5824	(49)	5844	(49)	5930	(50)	6052	(51)
6069	(51)	6070	(51)	6373	(55)	6688	(58)	6790	(60)
7132	(64)	7133	(64)	7134	(64)	7135	(64)	7198	(64)
7332	(65)	7427	(66)	7490	(66)	7491	(66)	7507	(66)
7523	(66)	7528	(66)	7529	(66)	7630	(67)	7680	(67)
7681	(67)	7682	(67)	7686	(67)	7687	(67)	7693	(67)
7726	(67)	7728	(67)	7735	(67)	7736	(67)	7737	(67)
7738	(67)	7739	(67)	7748	(67)	7749	(67)	7750	(67)
7755	(67)	7756	(67)	7760	(67)	7762	(67)	7763	(67)
7770	(67)	7771	(67)	7779	(67)	7780	(67)	7781	(67)
7789	(67)	7790	(67)	7791	(67)	7831	(68)	8260	(76)
8261	(76)	8262	(76)	8391	(77)	8420	(78)	8424	(78)
8426	(78)	8442	(79)	8452	(79)	8455	(79)	8457	(79)
8460	(79)	8677	(83)	8678	(83)	8730	(84)	8731	(84)
8845	(86)	8916	(87)	8917	(87)	8929	(87)	8930	(87)
8934	(87)	8935	(87)	8939	(87)	8945	(87)	8946	(87)
8947	(87)	8951	(87)	8952	(87)	8956	(87)	8957	(87)
8964	(87)	8965	(87)	8972	(87)	8982	(87)	8992	(87)
9042	(87)	9043	(87)	9076	(87)	9089	(87)	9090	(87)
9091	(87)	9099	(87)	9139	(88)	9140	(88)	9149	(88)
9150	(88)	9210	(89)	9628	(94)	9629	(94)	9633	(94)
9654	(94)	9655	(94)	9656	(94)	9665	(94)	9810	(95)
9843	(96)	9845	(96)	9864	(97)	9866	(97)	9867	(97)
9900	(98)	9902	(98)						

SIM 1 856 (3)
 SPHL 1 856 (3)
 STA 1 856 (3)

2268	(7)	3007	(17)						
1415	(5)	1416	(5)	1419	(5)	1421	(5)	1424	(5)
2272	(7)	2282	(7)	2296	(7)	2310	(7)	2340	(7)
2388	(7)	2389	(7)	2390	(7)	2464	(8)	2473	(8)
2520	(9)	2629	(11)	2697	(12)	2699	(12)	2707	(12)
2751	(13)	2803	(14)	2849	(14)	2889	(15)	2906	(15)
2939	(16)	2957	(16)	3001	(17)	3002	(17)	3005	(17)
3046	(18)	3047	(18)	3053	(18)	3056	(18)	3057	(18)
3062	(18)	3065	(18)	3068	(18)	3106	(19)	3107	(19)
3112	(19)	3115	(19)	3116	(19)	3121	(19)	3124	(19)
3127	(19)	3250	(20)	3260	(20)	3262	(20)	3278	(20)
3281	(20)	3305	(20)	3307	(20)	3308	(20)	3313	(20)
3317	(20)	3333	(20)	3354	(20)	3371	(20)	3383	(20)
3527	(22)	3529	(22)	3536	(22)	3646	(23)	3800	(24)
3806	(24)	3843	(24)	3845	(24)	3858	(24)	3893	(25)
3916	(26)	3953	(27)	3959	(27)	3961	(27)	3962	(27)

				3971	(27)	3978	(27)	3979	(27)	4009	(28)	4011	(28)
				4012	(28)	4040	(29)	4042	(29)	4043	(29)	4068	(30)
				4073	(30)	4074	(30)	4078	(30)	4080	(30)	4098	(31)
				4099	(31)	4103	(31)	4104	(31)	4128	(32)	4157	(33)
				4204	(34)	4206	(34)	4207	(34)	4216	(34)	4757	(40)
				4762	(40)	4768	(40)	4770	(40)	4773	(40)	4774	(40)
				4807	(40)	4815	(40)	4823	(40)	4831	(40)	4844	(40)
				4860	(40)	4872	(40)	4881	(40)	4896	(40)	4972	(41)
				4974	(41)	4979	(41)	4981	(41)	4993	(41)	5058	(42)
				5101	(44)	5104	(44)	5171	(45)	5176	(45)	5192	(45)
				5203	(45)	5204	(45)	5213	(45)	5216	(45)	5242	(45)
				5249	(45)	5252	(45)	5257	(45)	5266	(45)	5272	(45)
				5273	(45)	5286	(45)	5287	(45)	5290	(45)	5291	(45)
				5367	(46)	5370	(46)	5371	(46)	5381	(46)	5382	(46)
				5385	(46)	5386	(46)	5405	(46)	5414	(46)	5419	(46)
				5480	(47)	5575	(47)	5576	(47)	5587	(47)	5589	(47)
				5591	(47)	5592	(47)	5601	(47)	5602	(47)	5605	(47)
				5608	(47)	5617	(47)	5618	(47)	5627	(47)	5628	(47)
				5638	(47)	5930	(50)	6003	(51)	6013	(51)	6029	(51)
				6056	(51)	6064	(51)	6074	(51)	6076	(51)	6120	(52)
				6128	(52)	6265	(54)	6332	(54)	6335	(54)	6726	(59)
				6731	(59)	6773	(60)	6784	(60)	6785	(60)	6918	(62)
				6920	(62)	6934	(62)	6943	(62)	6951	(62)	7008	(63)
				7010	(63)	7064	(63)	7073	(63)	7081	(63)	7125	(64)
				7127	(64)	7151	(64)	7153	(64)	7186	(64)	7196	(64)
				7201	(64)	7335	(65)	7425	(66)	7431	(66)	7434	(66)
				7462	(66)	7465	(66)	7473	(66)	7476	(66)	7480	(66)
				7496	(66)	7503	(66)	7506	(66)	7571	(66)	7575	(66)
				7577	(66)	7585	(66)	7595	(67)	7598	(67)	7604	(67)
				7606	(67)	7626	(67)	7635	(67)	7648	(67)	7692	(67)
				7697	(67)	7700	(67)	7707	(67)	7709	(67)	7734	(67)
				7860	(69)	8173	(74)	8200	(75)	8277	(76)	8289	(76)
				8291	(76)	8293	(76)	8296	(76)	8356	(76)	8362	(76)
				8464	(79)	8669	(83)	8722	(84)	8841	(86)	8851	(86)
				8944	(87)	9004	(87)	9019	(87)	9157	(88)	9211	(89)
				9214	(89)	9223	(89)	9239	(89)	9326	(89)	9330	(89)
				9523	(92)	9524	(92)	9528	(92)	9537	(92)	9598	(93)
				9834	(96)	9843	(96)	9845	(96)	9846	(96)	9891	(98)
				9900	(98)	9902	(98)	9903	(98)				
STAX	1	856	(3)	2822	(14)	2834	(14)	3270	(20)	3325	(20)	4618	(39)
				4620	(39)	4622	(39)	4624	(39)	4659	(39)	4678	(39)
				5837	(49)	6284	(54)	6287	(54)	6479	(56)	6497	(56)
				6860	(61)	7041	(63)	7055	(63)	7928	(70)	8681	(83)
				8734	(84)	9320	(89)	9344	(89)	9731	(95)	9737	(95)
				9743	(95)	9749	(95)						
STC	1	856	(3)	2526	(9)	3483	(21)	3486	(21)	3620	(23)	3712	(23)
				3725	(23)	4385	(35)	4391	(35)	4470	(36)	4477	(36)
				4533	(37)	4536	(37)	4697	(39)	4704	(39)	4888	(40)
				4897	(40)	5011	(41)	5017	(41)	5059	(42)	5107	(44)
				5296	(45)	5303	(45)	5425	(46)	5432	(46)	5653	(47)
				5656	(47)	5717	(48)	5724	(48)	8582	(81)	8584	(81)
				8794	(85)	8797	(85)	9158	(88)	9166	(88)	9543	(92)
				9546	(92)								
SUB	1	856	(3)	3832	(24)	9237	(89)	9798	(95)				
SUI	1	856	(3)	4634	(39)	4641	(39)	5704	(48)	9260	(89)	9288	(89)

TYPE	2	432	(2)	9705 (95)		2270 (7)	2361 (7)	2362 (7)	2378 (7)
				2269 (7)		2381 (7)	2568 (10)	2569 (10)	2571 (10)
				2379 (7)		2710 (12)	3636 (23)	3637 (23)	3649 (23)
				2708 (12)		3811 (24)	3813 (24)	3817 (24)	3856 (24)
				3650 (23)		4129 (32)	4137 (32)	4209 (34)	4210 (34)
				3863 (24)		4218 (34)	4231 (34)	4954 (41)	4957 (41)
				4217 (34)		5004 (41)	7490 (66)	7491 (66)	7523 (66)
				4983 (41)		7681 (67)	7686 (67)	7687 (67)	7726 (67)
				7680 (67)		7736 (67)	7738 (67)	7739 (67)	7750 (67)
				7728 (67)		7780 (67)	7790 (67)	8260 (76)	8261 (76)
				7760 (67)		8391 (77)	8442 (79)	8452 (79)	8455 (79)
				8262 (76)		8845 (86)	8916 (87)	8917 (87)	8934 (87)
				8460 (79)		8946 (87)	8947 (87)	8951 (87)	8956 (87)
				8939 (87)		9042 (87)	9043 (87)	9076 (87)	9089 (87)
				8957 (87)		9091 (87)	9099 (87)	9139 (88)	9140 (88)
				9090 (87)		9629 (94)	9633 (94)	9654 (94)	9655 (94)
				9628 (94)		9665 (94)	9810 (95)	9864 (97)	9867 (97)
TYPEB	1	518	(2)	9656 (94)		2380 (7)	2570 (10)	2572 (10)	3812 (24)
				2380 (7)		4219 (34)	7507 (66)	7693 (67)	7735 (67)
				4219 (34)		8420 (78)	8457 (79)	8935 (87)	8945 (87)
				8420 (78)		8426 (78)	8952 (87)	9866 (97)	
TYPEL	1	580	(2)	8426 (78)		4994 (41)			
TYPEN	1	490	(2)	4994 (41)		7529 (66)	7682 (67)	7749 (67)	7756 (67)
TYPEW	2	549	(2)	7529 (66)		7781 (67)	7791 (67)	8424 (78)	8930 (87)
				7781 (67)		4128 (32)			9150 (88)
TYPE_UNCOUNTED	1	464	(2)	4128 (32)		3259 (20)	3261 (20)	3304 (20)	3306 (20)
WRITE\$_DCS	3	656	(2)	3259 (20)		3354 (20)	4008 (28)	4010 (28)	4039 (29)
				3354 (20)		4157 (33)	5479 (47)	5574 (47)	5588 (47)
				4157 (33)		5600 (47)	5616 (47)	5626 (47)	5930 (50)
				5600 (47)		5844 (96)	9899 (98)	9901 (98)	9842 (96)
XCHG	1	856	(3)	9844 (96)		3635 (23)	3692 (23)	3721 (23)	3827 (24)
				3635 (23)		4379 (35)	4771 (40)	5149 (45)	5347 (46)
				4379 (35)		5802 (49)	5805 (49)	5859 (49)	5921 (50)
				5802 (49)		6433 (56)	6439 (56)	6462 (56)	6466 (56)
				6433 (56)		6561 (57)	6685 (58)	6836 (61)	6849 (61)
				6561 (57)		7015 (63)	7029 (63)	7141 (64)	7741 (67)
				7015 (63)		7915 (70)	7939 (70)	7942 (70)	7962 (70)
				7915 (70)		8046 (71)	8049 (71)	8058 (71)	8061 (71)
				8046 (71)		8388 (77)	8567 (81)	8632 (82)	8922 (87)
				8388 (77)		9088 (87)	9142 (88)	9146 (88)	9284 (89)
				9088 (87)		9764 (95)	9807 (95)		9717 (95)
				9764 (95)		1401 (5)	2271 (7)	2303 (7)	2309 (7)
XRA	1	856	(3)	1401 (5)		3249 (20)	3256 (20)	3528 (22)	3711 (23)
				3249 (20)		3857 (24)	4205 (34)	4617 (39)	4772 (40)
				3857 (24)		6506 (56)	7126 (64)	7764 (67)	8243 (76)
				6506 (56)		8840 (86)	8966 (87)	9003 (87)	9222 (89)
				8840 (86)		9730 (95)	9736 (95)	9742 (95)	9748 (95)
				9730 (95)		9896 (98)			9839 (96)
				9896 (98)		2898 (15)	4079 (30)		
XRI	1	856	(3)	2898 (15)		4273 (34)			
XTHL	1	856	(3)	4273 (34)					

 ! Directives Cross Reference !

DIRECTIVE	REFERENCES...																
.ASCIC	1468	(6)	1469	(6)	1485	(6)	1486	(6)	1487	(6)	1488	(6)	1489	(6)	1490	(6)	
	1491	(6)	1492	(6)	1493	(6)	1494	(6)	1495	(6)	1496	(6)	1497	(6)	1520	(6)	
	1566	(6)	1678	(6)	1679	(6)	1692	(6)	1695	(6)	1787	(6)	1788	(6)	1790	(6)	
	1892	(6)	1893	(6)	1894	(6)	1895	(6)	1896	(6)	1897	(6)	1898	(6)	1906	(6)	
	1907	(6)	1908	(6)	1922	(6)	1940	(6)	1941	(6)	1942	(6)	1943	(6)	1944	(6)	
	1945	(6)	1946	(6)	1947	(6)	1948	(6)	1949	(6)	1950	(6)	1951	(6)	1952	(6)	
	1958	(6)	1959	(6)	1960	(6)	1961	(6)	1962	(6)	1963	(6)	1964	(6)	1965	(6)	
	1966	(6)	1967	(6)	1968	(6)	1970	(6)	1971	(6)	1972	(6)	1973	(6)	1974	(6)	
	1975	(6)	1977	(6)	1978	(6)	1980	(6)	1981	(6)	1983	(6)	1984	(6)	2016	(6)	
	2017	(6)	2018	(6)	2019	(6)	2020	(6)	2021	(6)	2022	(6)	2023	(6)	2031	(6)	
	2032	(6)	2033	(6)	2034	(6)	2046	(6)	2047	(6)	2048	(6)	2049	(6)	2050	(6)	
	2051	(6)	2052	(6)	2053	(6)	2061	(6)	2062	(6)	2063	(6)	2064	(6)	2070	(6)	
	2071	(6)	2079	(6)	2080	(6)	2081	(6)	2082	(6)	2090	(6)	2091	(6)	2092	(6)	
	2093	(6)	2105	(6)	2106	(6)	2107	(6)	2108	(6)	2109	(6)	2110	(6)	2111	(6)	
	2112	(6)	2118	(6)	2119	(6)	2124	(6)	2129	(6)	2137	(6)	2138	(6)	2139	(6)	
	2140	(6)	2146	(6)	2147	(6)	2155	(6)	2156	(6)	2157	(6)	2158	(6)	2166	(6)	
	2167	(6)	2168	(6)	2169	(6)	2175	(6)									
	.ASCII	1713	(6)	1714	(6)	1715	(6)	1716	(6)	1717	(6)	1718	(6)	1719	(6)	1720	(6)
		1721	(6)	1722	(6)	1725	(6)	1726	(6)	1727	(6)	1728	(6)	1730	(6)	1731	(6)
		1732	(6)	1733	(6)	1734	(6)	1735	(6)	1736	(6)	1737	(6)	1739	(6)	1740	(6)
1741		(6)	1742	(6)	1743	(6)	1744	(6)	1745	(6)	1746	(6)	1747	(6)	1748	(6)	
1749		(6)	1750	(6)	1751	(6)	1752	(6)	1753	(6)	1754	(6)	1755	(6)	1756	(6)	
1757		(6)	1758	(6)	1759	(6)	1760	(6)									
.BLKB	1559	(6)	1573	(6)	1686	(6)	1712	(6)	1803	(6)	1844	(6)	1845	(6)	1869	(6)	
	1870	(6)	1904	(6)	1936	(6)	1938	(6)	2181	(6)							
.BYTE	1373	(5)	1374	(5)	1375	(5)	1376	(5)	1377	(5)	1381	(5)	1384	(5)	1385	(5)	
	1386	(5)	1387	(5)	1388	(5)	1389	(5)	1393	(5)	1394	(5)	1395	(5)	1396	(5)	
	1397	(5)	1398	(5)	1399	(5)	1400	(5)	1401	(5)	1402	(5)	1403	(5)	1404	(5)	
	1405	(5)	1406	(5)	1407	(5)	1408	(5)	1409	(5)	1410	(5)	1414	(5)	1415	(5)	
	1416	(5)	1417	(5)	1418	(5)	1419	(5)	1420	(5)	1421	(5)	1423	(5)	1424	(5)	
	1425	(5)	1429	(5)	1430	(5)	1431	(5)	1443	(6)	1444	(6)	1445	(6)	1446	(6)	
	1453	(6)	1460	(6)	1461	(6)	1462	(6)	1463	(6)	1466	(6)	1504	(6)	1519	(6)	
	1553	(6)	1565	(6)	1572	(6)	1581	(6)	1587	(6)	1593	(6)	1599	(6)	1600	(6)	
	1609	(6)	1616	(6)	1617	(6)	1618	(6)	1619	(6)	1620	(6)	1621	(6)	1622	(6)	
	1623	(6)	1624	(6)	1625	(6)	1638	(6)	1639	(6)	1640	(6)	1641	(6)	1642	(6)	
	1643	(6)	1644	(6)	1658	(6)	1661	(6)	1663	(6)	1664	(6)	1665	(6)	1666	(6)	
	1673	(6)	1675	(6)	1693	(6)	1694	(6)	1723	(6)	1761	(6)	1773	(6)	1774	(6)	
	1775	(6)	1789	(6)	1842	(6)	1871	(6)	1873	(6)	1880	(6)	1887	(6)	1891	(6)	
	1905	(6)	1915	(6)	1916	(6)	1928	(6)	1930	(6)	1939	(6)	2006	(6)	2025	(6)	
	2036	(6)	2055	(6)	2066	(6)	2073	(6)	2084	(6)	2095	(6)	2114	(6)	2121	(6)	
	2126	(6)	2131	(6)	2142	(6)	2149	(6)	2160	(6)	2182	(6)	2183	(6)	2184	(6)	
	2189	(6)	2197	(6)	2198	(6)	2199	(6)	2200	(6)	2201	(6)	2202	(6)	2203	(6)	
	2204	(6)	2205	(6)	2206	(6)	2212	(6)	2264	(7)	2266	(7)	2267	(7)	2268	(7)	
	2269	(7)	2270	(7)	2271	(7)	2272	(7)	2276	(7)	2277	(7)	2278	(7)	2279	(7)	
	2280	(7)	2281	(7)	2282	(7)	2283	(7)	2284	(7)	2288	(7)	2289	(7)	2290	(7)	
2294	(7)	2295	(7)	2296	(7)	2297	(7)	2301	(7)	2302	(7)	2303	(7)	2304	(7)		
2305	(7)	2309	(7)	2310	(7)	2314	(7)	2315	(7)	2316	(7)	2317	(7)	2318	(7)		
2319	(7)	2320	(7)	2321	(7)	2325	(7)	2329	(7)	2330	(7)	2331	(7)	2332	(7)		

2333	(7)	2338	(7)	2339	(7)	2340	(7)	2341	(7)	2342	(7)	2343	(7)	2344	(7)
2345	(7)	2349	(7)	2351	(7)	2355	(7)	2357	(7)	2361	(7)	2362	(7)	2363	(7)
2368	(7)	2369	(7)	2370	(7)	2374	(7)	2378	(7)	2379	(7)	2380	(7)	2381	(7)
2382	(7)	2383	(7)	2384	(7)	2385	(7)	2386	(7)	2387	(7)	2388	(7)	2389	(7)
2390	(7)	2391	(7)	2392	(7)	2397	(7)	2398	(7)	2399	(7)	2463	(8)	2464	(8)
2465	(8)	2466	(8)	2467	(8)	2468	(8)	2469	(8)	2470	(8)	2471	(8)	2472	(8)
2473	(8)	2474	(8)	2475	(8)	2476	(8)	2477	(8)	2478	(8)	2518	(9)	2519	(9)
2520	(9)	2521	(9)	2522	(9)	2523	(9)	2524	(9)	2525	(9)	2526	(9)	2527	(9)
2568	(10)	2569	(10)	2570	(10)	2571	(10)	2572	(10)	2573	(10)	2627	(11)	2628	(11)
2629	(11)	2630	(11)	2631	(11)	2634	(11)	2635	(11)	2636	(11)	2637	(11)	2695	(12)
2696	(12)	2697	(12)	2698	(12)	2699	(12)	2702	(12)	2703	(12)	2704	(12)	2705	(12)
2706	(12)	2707	(12)	2708	(12)	2709	(12)	2710	(12)	2711	(12)	2745	(13)	2746	(13)
2747	(13)	2748	(13)	2749	(13)	2750	(13)	2751	(13)	2753	(13)	2754	(13)	2799	(14)
2800	(14)	2801	(14)	2802	(14)	2803	(14)	2804	(14)	2805	(14)	2806	(14)	2807	(14)
2817	(14)	2818	(14)	2819	(14)	2820	(14)	2821	(14)	2822	(14)	2823	(14)	2824	(14)
2825	(14)	2826	(14)	2827	(14)	2828	(14)	2833	(14)	2834	(14)	2835	(14)	2836	(14)
2837	(14)	2841	(14)	2842	(14)	2843	(14)	2847	(14)	2848	(14)	2849	(14)	2850	(14)
2887	(15)	2888	(15)	2889	(15)	2891	(15)	2892	(15)	2893	(15)	2895	(15)	2896	(15)
2897	(15)	2898	(15)	2899	(15)	2900	(15)	2901	(15)	2902	(15)	2903	(15)	2905	(15)
2906	(15)	2907	(15)	2937	(16)	2938	(16)	2939	(16)	2940	(16)	2941	(16)	2942	(16)
2944	(16)	2945	(16)	2946	(16)	2947	(16)	2948	(16)	2949	(16)	2950	(16)	2951	(16)
2952	(16)	2953	(16)	2954	(16)	2956	(16)	2957	(16)	2958	(16)	2997	(17)	2998	(17)
2999	(17)	3000	(17)	3001	(17)	3002	(17)	3003	(17)	3004	(17)	3005	(17)	3006	(17)
3007	(17)	3008	(17)	3041	(18)	3042	(18)	3043	(18)	3044	(18)	3045	(18)	3046	(18)
3047	(18)	3048	(18)	3049	(18)	3050	(18)	3051	(18)	3052	(18)	3053	(18)	3054	(18)
3055	(18)	3056	(18)	3057	(18)	3058	(18)	3061	(18)	3062	(18)	3063	(18)	3064	(18)
3065	(18)	3067	(18)	3068	(18)	3069	(18)	3101	(19)	3102	(19)	3103	(19)	3104	(19)
3105	(19)	3106	(19)	3107	(19)	3108	(19)	3109	(19)	3110	(19)	3111	(19)	3112	(19)
3113	(19)	3114	(19)	3115	(19)	3116	(19)	3117	(19)	3120	(19)	3121	(19)	3122	(19)
3123	(19)	3124	(19)	3126	(19)	3127	(19)	3128	(19)	3207	(20)	3212	(20)	3213	(20)
3214	(20)	3215	(20)	3216	(20)	3217	(20)	3218	(20)	3219	(20)	3223	(20)	3224	(20)
3225	(20)	3226	(20)	3227	(20)	3228	(20)	3229	(20)	3230	(20)	3231	(20)	3236	(20)
3237	(20)	3238	(20)	3242	(20)	3243	(20)	3244	(20)	3245	(20)	3247	(20)	3248	(20)
3249	(20)	3250	(20)	3251	(20)	3252	(20)	3253	(20)	3254	(20)	3255	(20)	3256	(20)
3257	(20)	3258	(20)	3260	(20)	3262	(20)	3263	(20)	3267	(20)	3268	(20)	3269	(20)
3270	(20)	3271	(20)	3272	(20)	3273	(20)	3274	(20)	3278	(20)	3279	(20)	3280	(20)
3281	(20)	3282	(20)	3283	(20)	3284	(20)	3285	(20)	3286	(20)	3287	(20)	3291	(20)
3292	(20)	3293	(20)	3294	(20)	3296	(20)	3297	(20)	3298	(20)	3299	(20)	3305	(20)
3307	(20)	3308	(20)	3313	(20)	3314	(20)	3315	(20)	3316	(20)	3317	(20)	3318	(20)
3322	(20)	3323	(20)	3324	(20)	3325	(20)	3326	(20)	3327	(20)	3328	(20)	3329	(20)
3333	(20)	3337	(20)	3338	(20)	3339	(20)	3340	(20)	3341	(20)	3342	(20)	3348	(20)
3349	(20)	3350	(20)	3351	(20)	3352	(20)	3353	(20)	3354	(20)	3358	(20)	3359	(20)
3360	(20)	3361	(20)	3362	(20)	3363	(20)	3364	(20)	3365	(20)	3366	(20)	3367	(20)
3368	(20)	3369	(20)	3370	(20)	3371	(20)	3372	(20)	3373	(20)	3374	(20)	3375	(20)
3376	(20)	3377	(20)	3378	(20)	3379	(20)	3380	(20)	3381	(20)	3382	(20)	3383	(20)
3384	(20)	3385	(20)	3386	(20)	3387	(20)	3391	(20)	3392	(20)	3393	(20)	3394	(20)
3395	(20)	3396	(20)	3400	(20)	3401	(20)	3402	(20)	3403	(20)	3404	(20)	3405	(20)
3406	(20)	3407	(20)	3412	(20)	3413	(20)	3414	(20)	3415	(20)	3416	(20)	3417	(20)
3418	(20)	3419	(20)	3420	(20)	3421	(20)	3422	(20)	3423	(20)	3424	(20)	3425	(20)
3426	(20)	3462	(21)	3463	(21)	3464	(21)	3465	(21)	3466	(21)	3467	(21)	3468	(21)
3469	(21)	3473	(21)	3474	(21)	3475	(21)	3476	(21)	3477	(21)	3478	(21)	3479	(21)
3480	(21)	3481	(21)	3482	(21)	3483	(21)	3484	(21)	3485	(21)	3486	(21)	3487	(21)
3488	(21)	3520	(22)	3521	(22)	3522	(22)	3523	(22)	3524	(22)	3525	(22)	3526	(22)
3527	(22)	3528	(22)	3529	(22)	3530	(22)	3531	(22)	3535	(22)	3536	(22)	3537	(22)
3604	(23)	3605	(23)	3606	(23)	3607	(23)	3608	(23)	3609	(23)	3610	(23)	3611	(23)

3612	(23)	3613	(23)	3618	(23)	3619	(23)	3620	(23)	3621	(23)	3626	(23)	3627	(23)
3628	(23)	3629	(23)	3630	(23)	3631	(23)	3632	(23)	3633	(23)	3634	(23)	3635	(23)
3636	(23)	3637	(23)	3641	(23)	3642	(23)	3643	(23)	3644	(23)	3645	(23)	3646	(23)
3647	(23)	3648	(23)	3649	(23)	3650	(23)	3654	(23)	3655	(23)	3656	(23)	3657	(23)
3661	(23)	3665	(23)	3666	(23)	3667	(23)	3671	(23)	3672	(23)	3673	(23)	3677	(23)
3678	(23)	3679	(23)	3680	(23)	3681	(23)	3682	(23)	3683	(23)	3684	(23)	3685	(23)
3686	(23)	3687	(23)	3688	(23)	3689	(23)	3690	(23)	3691	(23)	3692	(23)	3693	(23)
3694	(23)	3695	(23)	3699	(23)	3700	(23)	3701	(23)	3702	(23)	3706	(23)	3707	(23)
3709	(23)	3710	(23)	3711	(23)	3712	(23)	3713	(23)	3717	(23)	3718	(23)	3719	(23)
3720	(23)	3721	(23)	3722	(23)	3723	(23)	3724	(23)	3725	(23)	3726	(23)	3727	(23)
3779	(24)	3780	(24)	3781	(24)	3785	(24)	3786	(24)	3787	(24)	3791	(24)	3792	(24)
3796	(24)	3797	(24)	3798	(24)	3799	(24)	3800	(24)	3801	(24)	3802	(24)	3803	(24)
3804	(24)	3805	(24)	3806	(24)	3807	(24)	3808	(24)	3809	(24)	3810	(24)	3811	(24)
3812	(24)	3813	(24)	3814	(24)	3815	(24)	3816	(24)	3817	(24)	3821	(24)	3822	(24)
3823	(24)	3824	(24)	3825	(24)	3826	(24)	3827	(24)	3828	(24)	3829	(24)	3830	(24)
3831	(24)	3832	(24)	3833	(24)	3834	(24)	3835	(24)	3836	(24)	3837	(24)	3841	(24)
3842	(24)	3843	(24)	3844	(24)	3845	(24)	3846	(24)	3850	(24)	3851	(24)	3852	(24)
3856	(24)	3857	(24)	3858	(24)	3862	(24)	3863	(24)	3864	(24)	3890	(25)	3892	(25)
3893	(25)	3894	(25)	3895	(25)	3896	(25)	3915	(26)	3916	(26)	3917	(26)	3918	(26)
3919	(26)	3953	(27)	3955	(27)	3958	(27)	3959	(27)	3960	(27)	3961	(27)	3962	(27)
3967	(27)	3968	(27)	3969	(27)	3970	(27)	3971	(27)	3975	(27)	3977	(27)	3978	(27)
3979	(27)	3983	(27)	3984	(27)	3985	(27)	3986	(27)	4004	(28)	4005	(28)	4006	(28)
4007	(28)	4009	(28)	4011	(28)	4012	(28)	4013	(28)	4014	(28)	4015	(28)	4016	(28)
4017	(28)	4035	(29)	4036	(29)	4037	(29)	4038	(29)	4040	(29)	4042	(29)	4043	(29)
4044	(29)	4045	(29)	4046	(29)	4047	(29)	4048	(29)	4067	(30)	4068	(30)	4069	(30)
4070	(30)	4072	(30)	4073	(30)	4074	(30)	4075	(30)	4076	(30)	4077	(30)	4078	(30)
4079	(30)	4080	(30)	4081	(30)	4094	(31)	4095	(31)	4097	(31)	4098	(31)	4099	(31)
4100	(31)	4102	(31)	4103	(31)	4104	(31)	4105	(31)	4119	(32)	4120	(32)	4121	(32)
4122	(32)	4123	(32)	4124	(32)	4125	(32)	4126	(32)	4127	(32)	4128	(32)	4129	(32)
4130	(32)	4131	(32)	4132	(32)	4133	(32)	4134	(32)	4135	(32)	4136	(32)	4137	(32)
4138	(32)	4153	(33)	4154	(33)	4155	(33)	4156	(33)	4157	(33)	4158	(33)	4159	(33)
4160	(33)	4161	(33)	4162	(33)	4201	(34)	4203	(34)	4204	(34)	4205	(34)	4206	(34)
4207	(34)	4208	(34)	4209	(34)	4210	(34)	4212	(34)	4213	(34)	4214	(34)	4215	(34)
4216	(34)	4217	(34)	4218	(34)	4219	(34)	4220	(34)	4226	(34)	4227	(34)	4228	(34)
4229	(34)	4230	(34)	4231	(34)	4232	(34)	4237	(34)	4238	(34)	4239	(34)	4240	(34)
4241	(34)	4243	(34)	4245	(34)	4246	(34)	4247	(34)	4248	(34)	4249	(34)	4250	(34)
4251	(34)	4253	(34)	4254	(34)	4255	(34)	4259	(34)	4265	(34)	4266	(34)	4267	(34)
4268	(34)	4269	(34)	4270	(34)	4271	(34)	4272	(34)	4273	(34)	4275	(34)	4276	(34)
4280	(34)	4281	(34)	4282	(34)	4309	(35)	4310	(35)	4311	(35)	4312	(35)	4313	(35)
4317	(35)	4318	(35)	4319	(35)	4320	(35)	4321	(35)	4322	(35)	4326	(35)	4327	(35)
4328	(35)	4329	(35)	4330	(35)	4334	(35)	4335	(35)	4336	(35)	4337	(35)	4338	(35)
4339	(35)	4340	(35)	4344	(35)	4345	(35)	4346	(35)	4347	(35)	4348	(35)	4349	(35)
4350	(35)	4351	(35)	4355	(35)	4356	(35)	4357	(35)	4358	(35)	4359	(35)	4360	(35)
4361	(35)	4362	(35)	4363	(35)	4364	(35)	4368	(35)	4369	(35)	4375	(35)	4376	(35)
4377	(35)	4378	(35)	4379	(35)	4380	(35)	4385	(35)	4386	(35)	4391	(35)	4392	(35)
4393	(35)	4430	(36)	4431	(36)	4432	(36)	4433	(36)	4434	(36)	4435	(36)	4436	(36)
4443	(36)	4444	(36)	4445	(36)	4446	(36)	4447	(36)	4448	(36)	4449	(36)	4450	(36)
4451	(36)	4452	(36)	4459	(36)	4460	(36)	4461	(36)	4462	(36)	4463	(36)	4469	(36)
4470	(36)	4471	(36)	4472	(36)	4477	(36)	4478	(36)	4514	(37)	4515	(37)	4516	(37)
4517	(37)	4518	(37)	4519	(37)	4523	(37)	4524	(37)	4525	(37)	4526	(37)	4527	(37)
4531	(37)	4532	(37)	4533	(37)	4534	(37)	4535	(37)	4536	(37)	4537	(37)	4568	(38)
4569	(38)	4570	(38)	4571	(38)	4572	(38)	4573	(38)	4574	(38)	4616	(39)	4617	(39)
4618	(39)	4619	(39)	4620	(39)	4621	(39)	4622	(39)	4623	(39)	4624	(39)	4628	(39)
4629	(39)	4630	(39)	4631	(39)	4632	(39)	4633	(39)	4634	(39)	4635	(39)	4636	(39)
4637	(39)	4641	(39)	4642	(39)	4643	(39)	4644	(39)	4645	(39)	4649	(39)	4650	(39)

4651	(39)	4652	(39)	4653	(39)	4655	(39)	4656	(39)	4657	(39)	4658	(39)	4659	(39)
4660	(39)	4661	(39)	4662	(39)	4666	(39)	4667	(39)	4668	(39)	4672	(39)	4673	(39)
4674	(39)	4675	(39)	4676	(39)	4677	(39)	4678	(39)	4679	(39)	4685	(39)	4686	(39)
4687	(39)	4688	(39)	4689	(39)	4690	(39)	4691	(39)	4692	(39)	4693	(39)	4694	(39)
4695	(39)	4696	(39)	4697	(39)	4698	(39)	4699	(39)	4704	(39)	4705	(39)	4753	(40)
4755	(40)	4756	(40)	4757	(40)	4758	(40)	4761	(40)	4762	(40)	4763	(40)	4764	(40)
4767	(40)	4768	(40)	4769	(40)	4770	(40)	4771	(40)	4772	(40)	4773	(40)	4774	(40)
4775	(40)	4776	(40)	4777	(40)	4782	(40)	4783	(40)	4784	(40)	4785	(40)	4786	(40)
4790	(40)	4791	(40)	4792	(40)	4793	(40)	4794	(40)	4795	(40)	4796	(40)	4797	(40)
4798	(40)	4799	(40)	4800	(40)	4801	(40)	4805	(40)	4806	(40)	4807	(40)	4808	(40)
4813	(40)	4814	(40)	4815	(40)	4816	(40)	4821	(40)	4822	(40)	4823	(40)	4824	(40)
4829	(40)	4830	(40)	4831	(40)	4832	(40)	4838	(40)	4839	(40)	4840	(40)	4841	(40)
4842	(40)	4843	(40)	4844	(40)	4845	(40)	4846	(40)	4847	(40)	4848	(40)	4849	(40)
4850	(40)	4851	(40)	4852	(40)	4853	(40)	4854	(40)	4858	(40)	4859	(40)	4860	(40)
4861	(40)	4862	(40)	4868	(40)	4869	(40)	4870	(40)	4871	(40)	4872	(40)	4873	(40)
4879	(40)	4880	(40)	4881	(40)	4882	(40)	4883	(40)	4888	(40)	4889	(40)	4894	(40)
4895	(40)	4896	(40)	4897	(40)	4898	(40)	4899	(40)	4948	(41)	4949	(41)	4950	(41)
4954	(41)	4955	(41)	4956	(41)	4957	(41)	4958	(41)	4959	(41)	4960	(41)	4961	(41)
4965	(41)	4966	(41)	4970	(41)	4971	(41)	4972	(41)	4973	(41)	4974	(41)	4975	(41)
4976	(41)	4978	(41)	4979	(41)	4980	(41)	4981	(41)	4982	(41)	4983	(41)	4987	(41)
4988	(41)	4989	(41)	4990	(41)	4991	(41)	4992	(41)	4993	(41)	4994	(41)	4995	(41)
4996	(41)	4997	(41)	4998	(41)	4999	(41)	5000	(41)	5001	(41)	5002	(41)	5003	(41)
5004	(41)	5005	(41)	5006	(41)	5007	(41)	5011	(41)	5012	(41)	5013	(41)	5017	(41)
5018	(41)	5056	(42)	5057	(42)	5058	(42)	5059	(42)	5060	(42)	5061	(42)	5079	(43)
5098	(44)	5099	(44)	5100	(44)	5101	(44)	5102	(44)	5103	(44)	5104	(44)	5105	(44)
5106	(44)	5107	(44)	5108	(44)	5109	(44)	5149	(45)	5150	(45)	5151	(45)	5152	(45)
5153	(45)	5154	(45)	5155	(45)	5156	(45)	5157	(45)	5158	(45)	5159	(45)	5160	(45)
5161	(45)	5162	(45)	5163	(45)	5164	(45)	5169	(45)	5170	(45)	5171	(45)	5172	(45)
5173	(45)	5174	(45)	5175	(45)	5176	(45)	5177	(45)	5178	(45)	5183	(45)	5184	(45)
5185	(45)	5186	(45)	5187	(45)	5188	(45)	5189	(45)	5190	(45)	5191	(45)	5192	(45)
5196	(45)	5197	(45)	5198	(45)	5202	(45)	5203	(45)	5204	(45)	5205	(45)	5206	(45)
5207	(45)	5208	(45)	5209	(45)	5210	(45)	5211	(45)	5212	(45)	5213	(45)	5214	(45)
5215	(45)	5216	(45)	5217	(45)	5218	(45)	5224	(45)	5225	(45)	5226	(45)	5227	(45)
5228	(45)	5229	(45)	5230	(45)	5231	(45)	5232	(45)	5233	(45)	5234	(45)	5235	(45)
5236	(45)	5237	(45)	5241	(45)	5242	(45)	5243	(45)	5244	(45)	5245	(45)	5246	(45)
5247	(45)	5248	(45)	5249	(45)	5250	(45)	5251	(45)	5252	(45)	5253	(45)	5255	(45)
5256	(45)	5257	(45)	5258	(45)	5264	(45)	5265	(45)	5266	(45)	5270	(45)	5271	(45)
5272	(45)	5273	(45)	5277	(45)	5278	(45)	5279	(45)	5283	(45)	5284	(45)	5285	(45)
5286	(45)	5287	(45)	5288	(45)	5289	(45)	5290	(45)	5291	(45)	5296	(45)	5297	(45)
5298	(45)	5303	(45)	5304	(45)	5347	(46)	5348	(46)	5349	(46)	5350	(46)	5351	(46)
5352	(46)	5353	(46)	5354	(46)	5355	(46)	5356	(46)	5357	(46)	5358	(46)	5359	(46)
5360	(46)	5365	(46)	5366	(46)	5367	(46)	5368	(46)	5369	(46)	5370	(46)	5371	(46)
5375	(46)	5376	(46)	5377	(46)	5378	(46)	5379	(46)	5380	(46)	5381	(46)	5382	(46)
5383	(46)	5384	(46)	5385	(46)	5386	(46)	5387	(46)	5393	(46)	5394	(46)	5395	(46)
5396	(46)	5397	(46)	5398	(46)	5399	(46)	5400	(46)	5401	(46)	5402	(46)	5403	(46)
5404	(46)	5405	(46)	5406	(46)	5412	(46)	5413	(46)	5414	(46)	5415	(46)	5416	(46)
5417	(46)	5418	(46)	5419	(46)	5420	(46)	5425	(46)	5426	(46)	5427	(46)	5432	(46)
5433	(46)	5478	(47)	5480	(47)	5481	(47)	5482	(47)	5483	(47)	5484	(47)	5485	(47)
5486	(47)	5487	(47)	5488	(47)	5489	(47)	5490	(47)	5491	(47)	5492	(47)	5493	(47)
5494	(47)	5495	(47)	5496	(47)	5497	(47)	5498	(47)	5499	(47)	5500	(47)	5501	(47)
5502	(47)	5503	(47)	5504	(47)	5505	(47)	5506	(47)	5507	(47)	5508	(47)	5509	(47)
5510	(47)	5511	(47)	5512	(47)	5513	(47)	5514	(47)	5515	(47)	5516	(47)	5521	(47)
5522	(47)	5523	(47)	5524	(47)	5525	(47)	5526	(47)	5527	(47)	5532	(47)	5533	(47)
5534	(47)	5535	(47)	5536	(47)	5538	(47)	5539	(47)	5544	(47)	5545	(47)	5550	(47)
5551	(47)	5556	(47)	5557	(47)	5562	(47)	5563	(47)	5568	(47)	5569	(47)	5575	(47)

5576	(47)	5577	(47)	5582	(47)	5583	(47)	5584	(47)	5585	(47)	5586	(47)	5587	(47)
5589	(47)	5591	(47)	5592	(47)	5593	(47)	5594	(47)	5595	(47)	5601	(47)	5602	(47)
5603	(47)	5604	(47)	5605	(47)	5606	(47)	5607	(47)	5608	(47)	5609	(47)	5610	(47)
5611	(47)	5617	(47)	5618	(47)	5619	(47)	5620	(47)	5621	(47)	5627	(47)	5628	(47)
5629	(47)	5630	(47)	5637	(47)	5638	(47)	5639	(47)	5640	(47)	5645	(47)	5650	(47)
5651	(47)	5653	(47)	5654	(47)	5656	(47)	5657	(47)	5658	(47)	5661	(47)	5704	(48)
5705	(48)	5706	(48)	5707	(48)	5711	(48)	5712	(48)	5716	(48)	5717	(48)	5718	(48)
5719	(48)	5724	(48)	5725	(48)	5784	(49)	5785	(49)	5786	(49)	5790	(49)	5791	(49)
5792	(49)	5797	(49)	5798	(49)	5799	(49)	5800	(49)	5801	(49)	5802	(49)	5803	(49)
5804	(49)	5805	(49)	5806	(49)	5807	(49)	5808	(49)	5809	(49)	5810	(49)	5811	(49)
5812	(49)	5813	(49)	5814	(49)	5815	(49)	5816	(49)	5817	(49)	5818	(49)	5819	(49)
5823	(49)	5824	(49)	5825	(49)	5826	(49)	5827	(49)	5828	(49)	5829	(49)	5830	(49)
5831	(49)	5832	(49)	5833	(49)	5834	(49)	5835	(49)	5836	(49)	5837	(49)	5838	(49)
5839	(49)	5840	(49)	5844	(49)	5849	(49)	5850	(49)	5851	(49)	5852	(49)	5853	(49)
5854	(49)	5855	(49)	5856	(49)	5857	(49)	5858	(49)	5859	(49)	5860	(49)	5905	(50)
5906	(50)	5910	(50)	5911	(50)	5912	(50)	5913	(50)	5917	(50)	5918	(50)	5919	(50)
5920	(50)	5921	(50)	5922	(50)	5923	(50)	5927	(50)	5928	(50)	5929	(50)	5930	(50)
5931	(50)	5996	(51)	5997	(51)	5998	(51)	6002	(51)	6003	(51)	6007	(51)	6011	(51)
6012	(51)	6013	(51)	6017	(51)	6018	(51)	6019	(51)	6020	(51)	6021	(51)	6022	(51)
6023	(51)	6024	(51)	6025	(51)	6026	(51)	6028	(51)	6029	(51)	6030	(51)	6034	(51)
6035	(51)	6036	(51)	6040	(51)	6041	(51)	6042	(51)	6043	(51)	6044	(51)	6048	(51)
6049	(51)	6050	(51)	6051	(51)	6052	(51)	6053	(51)	6055	(51)	6056	(51)	6057	(51)
6061	(51)	6063	(51)	6064	(51)	6068	(51)	6069	(51)	6070	(51)	6071	(51)	6073	(51)
6074	(51)	6075	(51)	6076	(51)	6077	(51)	6111	(52)	6116	(52)	6117	(52)	6118	(52)
6119	(52)	6120	(52)	6122	(52)	6123	(52)	6124	(52)	6125	(52)	6127	(52)	6128	(52)
6129	(52)	6169	(53)	6170	(53)	6171	(53)	6172	(53)	6173	(53)	6174	(53)	6175	(53)
6176	(53)	6177	(53)	6178	(53)	6182	(53)	6183	(53)	6184	(53)	6185	(53)	6186	(53)
6190	(53)	6191	(53)	6192	(53)	6193	(53)	6194	(53)	6195	(53)	6199	(53)	6200	(53)
6201	(53)	6204	(53)	6205	(53)	6206	(53)	6210	(53)	6211	(53)	6215	(53)	6216	(53)
6217	(53)	6218	(53)	6219	(53)	6220	(53)	6221	(53)	6263	(54)	6264	(54)	6265	(54)
6266	(54)	6267	(54)	6268	(54)	6269	(54)	6270	(54)	6271	(54)	6272	(54)	6273	(54)
6274	(54)	6275	(54)	6276	(54)	6277	(54)	6278	(54)	6279	(54)	6280	(54)	6281	(54)
6282	(54)	6283	(54)	6284	(54)	6285	(54)	6286	(54)	6287	(54)	6291	(54)	6292	(54)
6293	(54)	6294	(54)	6295	(54)	6296	(54)	6297	(54)	6298	(54)	6299	(54)	6303	(54)
6304	(54)	6305	(54)	6306	(54)	6307	(54)	6308	(54)	6309	(54)	6313	(54)	6314	(54)
6315	(54)	6319	(54)	6320	(54)	6321	(54)	6325	(54)	6326	(54)	6330	(54)	6331	(54)
6332	(54)	6333	(54)	6334	(54)	6335	(54)	6336	(54)	6340	(54)	6341	(54)	6342	(54)
6343	(54)	6344	(54)	6345	(54)	6346	(54)	6372	(55)	6373	(55)	6374	(55)	6421	(56)
6422	(56)	6426	(56)	6427	(56)	6428	(56)	6429	(56)	6433	(56)	6434	(56)	6435	(56)
6436	(56)	6437	(56)	6438	(56)	6439	(56)	6440	(56)	6441	(56)	6442	(56)	6446	(56)
6447	(56)	6448	(56)	6449	(56)	6450	(56)	6451	(56)	6452	(56)	6453	(56)	6454	(56)
6455	(56)	6456	(56)	6460	(56)	6461	(56)	6462	(56)	6463	(56)	6464	(56)	6465	(56)
6466	(56)	6470	(56)	6471	(56)	6472	(56)	6476	(56)	6477	(56)	6478	(56)	6479	(56)
6480	(56)	6481	(56)	6482	(56)	6483	(56)	6492	(56)	6493	(56)	6494	(56)	6495	(56)
6496	(56)	6497	(56)	6498	(56)	6499	(56)	6500	(56)	6501	(56)	6502	(56)	6503	(56)
6504	(56)	6505	(56)	6506	(56)	6507	(56)	6508	(56)	6512	(56)	6544	(57)	6545	(57)
6546	(57)	6547	(57)	6551	(57)	6552	(57)	6553	(57)	6554	(57)	6555	(57)	6561	(57)
6562	(57)	6563	(57)	6564	(57)	6565	(57)	6569	(57)	6570	(57)	6571	(57)	6573	(57)
6574	(57)	6575	(57)	6576	(57)	6577	(57)	6578	(57)	6579	(57)	6580	(57)	6581	(57)
6582	(57)	6583	(57)	6584	(57)	6585	(57)	6586	(57)	6624	(58)	6625	(58)	6626	(58)
6627	(58)	6628	(58)	6629	(58)	6630	(58)	6631	(58)	6632	(58)	6633	(58)	6634	(58)
6635	(58)	6639	(58)	6640	(58)	6641	(58)	6642	(58)	6646	(58)	6647	(58)	6648	(58)
6649	(58)	6653	(58)	6654	(58)	6655	(58)	6656	(58)	6657	(58)	6661	(58)	6662	(58)
6663	(58)	6664	(58)	6665	(58)	6666	(58)	6667	(58)	6671	(58)	6672	(58)	6673	(58)
6674	(58)	6678	(58)	6679	(58)	6680	(58)	6684	(58)	6685	(58)	6686	(58)	6687	(58)

6688	(58)	6689	(58)	6720	(59)	6724	(59)	6725	(59)	6726	(59)	6730	(59)	6731	(59)
6732	(59)	6771	(60)	6772	(60)	6773	(60)	6774	(60)	6775	(60)	6779	(60)	6780	(60)
6781	(60)	6782	(60)	6783	(60)	6784	(60)	6785	(60)	6789	(60)	6790	(60)	6791	(60)
6824	(61)	6825	(61)	6829	(61)	6830	(61)	6834	(61)	6835	(61)	6836	(61)	6837	(61)
6838	(61)	6839	(61)	6843	(61)	6844	(61)	6845	(61)	6846	(61)	6847	(61)	6848	(61)
6849	(61)	6850	(61)	6851	(61)	6855	(61)	6856	(61)	6857	(61)	6858	(61)	6859	(61)
6860	(61)	6861	(61)	6862	(61)	6863	(61)	6864	(61)	6868	(61)	6909	(62)	6910	(62)
6911	(62)	6915	(62)	6916	(62)	6917	(62)	6918	(62)	6919	(62)	6920	(62)	6921	(62)
6922	(62)	6923	(62)	6924	(62)	6925	(62)	6926	(62)	6927	(62)	6928	(62)	6929	(62)
6933	(62)	6934	(62)	6935	(62)	6936	(62)	6937	(62)	6938	(62)	6939	(62)	6941	(62)
6942	(62)	6943	(62)	6944	(62)	6948	(62)	6950	(62)	6951	(62)	6952	(62)	6953	(62)
6999	(63)	7000	(63)	7001	(63)	7005	(63)	7006	(63)	7007	(63)	7008	(63)	7009	(63)
7010	(63)	7011	(63)	7012	(63)	7013	(63)	7014	(63)	7015	(63)	7016	(63)	7017	(63)
7018	(63)	7019	(63)	7023	(63)	7024	(63)	7025	(63)	7026	(63)	7027	(63)	7028	(63)
7029	(63)	7030	(63)	7031	(63)	7035	(63)	7036	(63)	7037	(63)	7038	(63)	7039	(63)
7040	(63)	7041	(63)	7042	(63)	7043	(63)	7044	(63)	7045	(63)	7049	(63)	7050	(63)
7051	(63)	7052	(63)	7053	(63)	7054	(63)	7055	(63)	7056	(63)	7057	(63)	7058	(63)
7059	(63)	7063	(63)	7064	(63)	7065	(63)	7066	(63)	7067	(63)	7068	(63)	7069	(63)
7071	(63)	7072	(63)	7073	(63)	7074	(63)	7078	(63)	7080	(63)	7081	(63)	7082	(63)
7083	(63)	7124	(64)	7125	(64)	7126	(64)	7127	(64)	7131	(64)	7132	(64)	7133	(64)
7134	(64)	7135	(64)	7139	(64)	7140	(64)	7141	(64)	7142	(64)	7143	(64)	7147	(64)
7148	(64)	7149	(64)	7150	(64)	7151	(64)	7152	(64)	7153	(64)	7154	(64)	7155	(64)
7159	(64)	7160	(64)	7161	(64)	7162	(64)	7163	(64)	7164	(64)	7165	(64)	7166	(64)
7170	(64)	7171	(64)	7175	(64)	7176	(64)	7177	(64)	7178	(64)	7179	(64)	7183	(64)
7185	(64)	7186	(64)	7187	(64)	7188	(64)	7192	(64)	7195	(64)	7196	(64)	7197	(64)
7198	(64)	7199	(64)	7200	(64)	7201	(64)	7202	(64)	7273	(65)	7274	(65)	7275	(65)
7279	(65)	7280	(65)	7281	(65)	7285	(65)	7286	(65)	7287	(65)	7291	(65)	7292	(65)
7293	(65)	7297	(65)	7298	(65)	7299	(65)	7300	(65)	7304	(65)	7305	(65)	7306	(65)
7310	(65)	7311	(65)	7312	(65)	7313	(65)	7314	(65)	7315	(65)	7316	(65)	7320	(65)
7321	(65)	7322	(65)	7323	(65)	7324	(65)	7325	(65)	7326	(65)	7327	(65)	7331	(65)
7332	(65)	7333	(65)	7334	(65)	7335	(65)	7339	(65)	7424	(66)	7425	(66)	7426	(66)
7427	(66)	7428	(66)	7430	(66)	7431	(66)	7432	(66)	7433	(66)	7434	(66)	7438	(66)
7439	(66)	7440	(66)	7444	(66)	7445	(66)	7446	(66)	7447	(66)	7448	(66)	7449	(66)
7450	(66)	7451	(66)	7452	(66)	7453	(66)	7454	(66)	7455	(66)	7456	(66)	7461	(66)
7462	(66)	7463	(66)	7464	(66)	7465	(66)	7466	(66)	7467	(66)	7472	(66)	7473	(66)
7474	(66)	7475	(66)	7476	(66)	7477	(66)	7478	(66)	7479	(66)	7480	(66)	7484	(66)
7485	(66)	7486	(66)	7490	(66)	7491	(66)	7492	(66)	7494	(66)	7495	(66)	7496	(66)
7497	(66)	7498	(66)	7499	(66)	7500	(66)	7501	(66)	7502	(66)	7503	(66)	7504	(66)
7505	(66)	7506	(66)	7507	(66)	7511	(66)	7512	(66)	7513	(66)	7514	(66)	7515	(66)
7516	(66)	7517	(66)	7518	(66)	7519	(66)	7523	(66)	7524	(66)	7525	(66)	7526	(66)
7527	(66)	7528	(66)	7529	(66)	7530	(66)	7531	(66)	7535	(66)	7536	(66)	7537	(66)
7538	(66)	7539	(66)	7540	(66)	7541	(66)	7542	(66)	7543	(66)	7544	(66)	7550	(66)
7551	(66)	7552	(66)	7553	(66)	7554	(66)	7555	(66)	7556	(66)	7561	(66)	7562	(66)
7563	(66)	7564	(66)	7565	(66)	7569	(66)	7570	(66)	7571	(66)	7572	(66)	7573	(66)
7574	(66)	7575	(66)	7576	(66)	7577	(66)	7578	(66)	7582	(66)	7584	(66)	7585	(66)
7586	(66)	7587	(66)	7593	(67)	7594	(67)	7595	(67)	7596	(67)	7597	(67)	7598	(67)
7603	(67)	7604	(67)	7605	(67)	7606	(67)	7607	(67)	7608	(67)	7609	(67)	7610	(67)
7611	(67)	7613	(67)	7614	(67)	7615	(67)	7616	(67)	7617	(67)	7621	(67)	7622	(67)
7623	(67)	7625	(67)	7626	(67)	7627	(67)	7628	(67)	7629	(67)	7630	(67)	7634	(67)
7635	(67)	7639	(67)	7640	(67)	7641	(67)	7642	(67)	7643	(67)	7647	(67)	7648	(67)
7649	(67)	7650	(67)	7651	(67)	7652	(67)	7653	(67)	7654	(67)	7655	(67)	7656	(67)
7657	(67)	7661	(67)	7662	(67)	7663	(67)	7667	(67)	7668	(67)	7669	(67)	7673	(67)
7674	(67)	7675	(67)	7680	(67)	7681	(67)	7682	(67)	7683	(67)	7684	(67)	7685	(67)
7686	(67)	7687	(67)	7688	(67)	7690	(67)	7691	(67)	7692	(67)	7693	(67)	7694	(67)
7695	(67)	7696	(67)	7697	(67)	7699	(67)	7700	(67)	7701	(67)	7705	(67)	7706	(67)

7707	(67)	7708	(67)	7709	(67)	7710	(67)	7711	(67)	7713	(67)	7714	(67)	7715	(67)
7716	(67)	7717	(67)	7721	(67)	7726	(67)	7727	(67)	7728	(67)	7729	(67)	7732	(67)
7733	(67)	7734	(67)	7735	(67)	7736	(67)	7737	(67)	7738	(67)	7739	(67)	7740	(67)
7741	(67)	7742	(67)	7743	(67)	7744	(67)	7745	(67)	7746	(67)	7747	(67)	7748	(67)
7749	(67)	7750	(67)	7751	(67)	7752	(67)	7753	(67)	7754	(67)	7755	(67)	7756	(67)
7760	(67)	7761	(67)	7762	(67)	7763	(67)	7764	(67)	7765	(67)	7766	(67)	7767	(67)
7768	(67)	7769	(67)	7770	(67)	7771	(67)	7773	(67)	7774	(67)	7775	(67)	7776	(67)
7777	(67)	7778	(67)	7779	(67)	7780	(67)	7781	(67)	7783	(67)	7784	(67)	7785	(67)
7786	(67)	7787	(67)	7788	(67)	7789	(67)	7790	(67)	7791	(67)	7792	(67)	7793	(67)
7823	(68)	7824	(68)	7825	(68)	7826	(68)	7827	(68)	7828	(68)	7829	(68)	7830	(68)
7831	(68)	7832	(68)	7858	(69)	7859	(69)	7860	(69)	7861	(69)	7862	(69)	7906	(70)
7907	(70)	7908	(70)	7909	(70)	7913	(70)	7914	(70)	7915	(70)	7916	(70)	7917	(70)
7921	(70)	7922	(70)	7923	(70)	7924	(70)	7925	(70)	7926	(70)	7927	(70)	7928	(70)
7929	(70)	7930	(70)	7931	(70)	7932	(70)	7933	(70)	7937	(70)	7938	(70)	7939	(70)
7940	(70)	7941	(70)	7942	(70)	7949	(70)	7950	(70)	7951	(70)	7952	(70)	7953	(70)
7954	(70)	7955	(70)	7956	(70)	7957	(70)	7958	(70)	7959	(70)	7960	(70)	7961	(70)
7962	(70)	7963	(70)	7964	(70)	7965	(70)	7966	(70)	7970	(70)	7971	(70)	7972	(70)
7973	(70)	7974	(70)	7975	(70)	7976	(70)	7977	(70)	7978	(70)	7979	(70)	7980	(70)
7981	(70)	7982	(70)	7983	(70)	7984	(70)	7985	(70)	7986	(70)	7987	(70)	7988	(70)
7989	(70)	7990	(70)	7991	(70)	7992	(70)	7993	(70)	7994	(70)	7995	(70)	8030	(71)
8031	(71)	8032	(71)	8033	(71)	8034	(71)	8035	(71)	8039	(71)	8040	(71)	8041	(71)
8042	(71)	8043	(71)	8044	(71)	8045	(71)	8046	(71)	8047	(71)	8048	(71)	8049	(71)
8050	(71)	8051	(71)	8057	(71)	8058	(71)	8059	(71)	8060	(71)	8061	(71)	8062	(71)
8063	(71)	8068	(71)	8069	(71)	8074	(71)	8075	(71)	8076	(71)	8077	(71)	8078	(71)
8079	(71)	8080	(71)	8106	(72)	8107	(72)	8108	(72)	8109	(72)	8110	(72)	8111	(72)
8112	(72)	8138	(73)	8139	(73)	8140	(73)	8141	(73)	8142	(73)	8143	(73)	8144	(73)
8145	(73)	8146	(73)	8147	(73)	8171	(74)	8172	(74)	8173	(74)	8174	(74)	8198	(75)
8199	(75)	8200	(75)	8201	(75)	8241	(76)	8242	(76)	8243	(76)	8244	(76)	8245	(76)
8246	(76)	8247	(76)	8248	(76)	8249	(76)	8250	(76)	8251	(76)	8252	(76)	8253	(76)
8254	(76)	8255	(76)	8256	(76)	8260	(76)	8261	(76)	8262	(76)	8263	(76)	8267	(76)
8268	(76)	8269	(76)	8270	(76)	8271	(76)	8272	(76)	8273	(76)	8274	(76)	8275	(76)
8276	(76)	8277	(76)	8281	(76)	8282	(76)	8283	(76)	8287	(76)	8288	(76)	8289	(76)
8290	(76)	8291	(76)	8292	(76)	8293	(76)	8294	(76)	8295	(76)	8296	(76)	8297	(76)
8298	(76)	8299	(76)	8303	(76)	8304	(76)	8305	(76)	8306	(76)	8307	(76)	8308	(76)
8309	(76)	8310	(76)	8311	(76)	8312	(76)	8313	(76)	8314	(76)	8315	(76)	8316	(76)
8317	(76)	8318	(76)	8319	(76)	8320	(76)	8321	(76)	8322	(76)	8323	(76)	8324	(76)
8325	(76)	8326	(76)	8327	(76)	8328	(76)	8329	(76)	8334	(76)	8335	(76)	8336	(76)
8337	(76)	8338	(76)	8339	(76)	8340	(76)	8341	(76)	8342	(76)	8343	(76)	8344	(76)
8345	(76)	8346	(76)	8347	(76)	8348	(76)	8349	(76)	8350	(76)	8352	(76)	8354	(76)
8355	(76)	8356	(76)	8357	(76)	8359	(76)	8361	(76)	8362	(76)	8363	(76)	8365	(76)
8387	(77)	8388	(77)	8389	(77)	8390	(77)	8391	(77)	8392	(77)	8416	(78)	8417	(78)
8418	(78)	8419	(78)	8420	(78)	8421	(78)	8422	(78)	8423	(78)	8424	(78)	8425	(78)
8426	(78)	8428	(78)	8439	(79)	8440	(79)	8441	(79)	8442	(79)	8443	(79)	8444	(79)
8445	(79)	8446	(79)	8447	(79)	8448	(79)	8449	(79)	8450	(79)	8451	(79)	8452	(79)
8453	(79)	8454	(79)	8455	(79)	8456	(79)	8457	(79)	8460	(79)	8462	(79)	8463	(79)
8464	(79)	8465	(79)	8497	(80)	8498	(80)	8499	(80)	8500	(80)	8501	(80)	8502	(80)
8503	(80)	8504	(80)	8505	(80)	8506	(80)	8507	(80)	8508	(80)	8550	(81)	8551	(81)
8552	(81)	8553	(81)	8557	(81)	8558	(81)	8559	(81)	8560	(81)	8564	(81)	8565	(81)
8566	(81)	8567	(81)	8572	(81)	8573	(81)	8574	(81)	8575	(81)	8576	(81)	8577	(81)
8578	(81)	8579	(81)	8580	(81)	8581	(81)	8582	(81)	8583	(81)	8584	(81)	8585	(81)
8586	(81)	8623	(82)	8624	(82)	8628	(82)	8629	(82)	8630	(82)	8631	(82)	8632	(82)
8633	(82)	8669	(83)	8670	(83)	8671	(83)	8675	(83)	8676	(83)	8677	(83)	8678	(83)
8679	(83)	8680	(83)	8681	(83)	8682	(83)	8683	(83)	8684	(83)	8685	(83)	8686	(83)
8722	(84)	8723	(84)	8724	(84)	8728	(84)	8729	(84)	8730	(84)	8731	(84)	8732	(84)
8733	(84)	8734	(84)	8735	(84)	8736	(84)	8737	(84)	8738	(84)	8739	(84)	8781	(85)

8782	(85)	8783	(85)	8784	(85)	8785	(85)	8786	(85)	8787	(85)	8788	(85)	8789	(85)
8790	(85)	8791	(85)	8792	(85)	8793	(85)	8794	(85)	8795	(85)	8796	(85)	8797	(85)
8798	(85)	8802	(85)	8803	(85)	8804	(85)	8805	(85)	8806	(85)	8807	(85)	8808	(85)
8809	(85)	8835	(86)	8836	(86)	8837	(86)	8838	(86)	8839	(86)	8840	(86)	8841	(86)
8845	(86)	8849	(86)	8850	(86)	8851	(86)	8852	(86)	8915	(87)	8916	(87)	8917	(87)
8921	(87)	8922	(87)	8923	(87)	8924	(87)	8925	(87)	8926	(87)	8927	(87)	8928	(87)
8929	(87)	8930	(87)	8934	(87)	8935	(87)	8939	(87)	8940	(87)	8941	(87)	8942	(87)
8943	(87)	8944	(87)	8945	(87)	8946	(87)	8947	(87)	8951	(87)	8952	(87)	8953	(87)
8954	(87)	8955	(87)	8956	(87)	8957	(87)	8958	(87)	8959	(87)	8963	(87)	8964	(87)
8965	(87)	8966	(87)	8967	(87)	8968	(87)	8969	(87)	8970	(87)	8971	(87)	8972	(87)
8973	(87)	8974	(87)	8976	(87)	8977	(87)	8978	(87)	8979	(87)	8980	(87)	8981	(87)
8982	(87)	8983	(87)	8984	(87)	8986	(87)	8987	(87)	8988	(87)	8989	(87)	8990	(87)
8991	(87)	8992	(87)	8993	(87)	8994	(87)	8998	(87)	8999	(87)	9000	(87)	9001	(87)
9002	(87)	9003	(87)	9004	(87)	9005	(87)	9006	(87)	9007	(87)	9011	(87)	9012	(87)
9016	(87)	9017	(87)	9018	(87)	9019	(87)	9020	(87)	9021	(87)	9022	(87)	9023	(87)
9024	(87)	9028	(87)	9029	(87)	9030	(87)	9034	(87)	9035	(87)	9036	(87)	9037	(87)
9038	(87)	9039	(87)	9040	(87)	9041	(87)	9042	(87)	9043	(87)	9044	(87)	9045	(87)
9046	(87)	9047	(87)	9048	(87)	9049	(87)	9050	(87)	9051	(87)	9052	(87)	9053	(87)
9054	(87)	9055	(87)	9056	(87)	9061	(87)	9062	(87)	9063	(87)	9064	(87)	9069	(87)
9070	(87)	9071	(87)	9072	(87)	9073	(87)	9074	(87)	9075	(87)	9076	(87)	9077	(87)
9078	(87)	9079	(87)	9080	(87)	9081	(87)	9082	(87)	9083	(87)	9084	(87)	9085	(87)
9086	(87)	9087	(87)	9088	(87)	9089	(87)	9090	(87)	9091	(87)	9092	(87)	9093	(87)
9094	(87)	9095	(87)	9096	(87)	9097	(87)	9098	(87)	9099	(87)	9103	(87)	9138	(88)
9139	(88)	9140	(88)	9141	(88)	9142	(88)	9143	(88)	9144	(88)	9145	(88)	9146	(88)
9147	(88)	9148	(88)	9149	(88)	9150	(88)	9151	(88)	9152	(88)	9153	(88)	9154	(88)
9155	(88)	9156	(88)	9157	(88)	9158	(88)	9159	(88)	9165	(88)	9166	(88)	9167	(88)
9168	(88)	9210	(89)	9211	(89)	9212	(89)	9213	(89)	9214	(89)	9215	(89)	9216	(89)
9217	(89)	9218	(89)	9219	(89)	9220	(89)	9221	(89)	9222	(89)	9223	(89)	9224	(89)
9225	(89)	9226	(89)	9227	(89)	9228	(89)	9229	(89)	9230	(89)	9231	(89)	9235	(89)
9236	(89)	9237	(89)	9238	(89)	9239	(89)	9244	(89)	9245	(89)	9246	(89)	9247	(89)
9248	(89)	9249	(89)	9250	(89)	9251	(89)	9252	(89)	9253	(89)	9259	(89)	9260	(89)
9261	(89)	9262	(89)	9263	(89)	9264	(89)	9272	(89)	9273	(89)	9274	(89)	9281	(89)
9282	(89)	9283	(89)	9284	(89)	9285	(89)	9286	(89)	9287	(89)	9288	(89)	9289	(89)
9290	(89)	9291	(89)	9292	(89)	9293	(89)	9294	(89)	9301	(89)	9302	(89)	9303	(89)
9307	(89)	9308	(89)	9309	(89)	9310	(89)	9311	(89)	9312	(89)	9313	(89)	9314	(89)
9315	(89)	9316	(89)	9317	(89)	9318	(89)	9319	(89)	9320	(89)	9324	(89)	9325	(89)
9326	(89)	9327	(89)	9328	(89)	9329	(89)	9330	(89)	9331	(89)	9335	(89)	9336	(89)
9337	(89)	9338	(89)	9342	(89)	9343	(89)	9344	(89)	9345	(89)	9346	(89)	9347	(89)
9348	(89)	9349	(89)	9350	(89)	9351	(89)	9352	(89)	9353	(89)	9354	(89)	9355	(89)
9356	(89)	9358	(89)	9392	(90)	9396	(90)	9397	(90)	9398	(90)	9399	(90)	9400	(90)
9401	(90)	9402	(90)	9403	(90)	9404	(90)	9405	(90)	9406	(90)	9407	(90)	9408	(90)
9412	(90)	9413	(90)	9414	(90)	9415	(90)	9416	(90)	9417	(90)	9418	(90)	9419	(90)
9420	(90)	9421	(90)	9422	(90)	9423	(90)	9424	(90)	9425	(90)	9429	(90)	9430	(90)
9431	(90)	9432	(90)	9433	(90)	9434	(90)	9435	(90)	9436	(90)	9437	(90)	9438	(90)
9439	(90)	9440	(90)	9441	(90)	9442	(90)	9443	(90)	9447	(90)	9448	(90)	9449	(90)
9450	(90)	9451	(90)	9452	(90)	9453	(90)	9454	(90)	9455	(90)	9456	(90)	9457	(90)
9458	(90)	9459	(90)	9481	(91)	9482	(91)	9483	(91)	9484	(91)	9485	(91)	9486	(91)
9487	(91)	9488	(91)	9489	(91)	9519	(92)	9520	(92)	9521	(92)	9522	(92)	9523	(92)
9524	(92)	9525	(92)	9526	(92)	9527	(92)	9528	(92)	9529	(92)	9530	(92)	9531	(92)
9532	(92)	9533	(92)	9534	(92)	9535	(92)	9536	(92)	9537	(92)	9538	(92)	9539	(92)
9540	(92)	9541	(92)	9542	(92)	9543	(92)	9544	(92)	9545	(92)	9546	(92)	9547	(92)
9571	(93)	9572	(93)	9573	(93)	9574	(93)	9575	(93)	9576	(93)	9577	(93)	9578	(93)
9579	(93)	9583	(93)	9584	(93)	9585	(93)	9586	(93)	9587	(93)	9588	(93)	9589	(93)
9590	(93)	9591	(93)	9595	(93)	9597	(93)	9598	(93)	9599	(93)	9625	(94)	9626	(94)
9627	(94)	9628	(94)	9629	(94)	9630	(94)	9631	(94)	9632	(94)	9633	(94)	9634	(94)

9635	(94)	9636	(94)	9637	(94)	9638	(94)	9639	(94)	9640	(94)	9641	(94)	9642	(94)
9643	(94)	9644	(94)	9645	(94)	9646	(94)	9647	(94)	9648	(94)	9649	(94)	9650	(94)
9651	(94)	9652	(94)	9653	(94)	9654	(94)	9655	(94)	9656	(94)	9657	(94)	9658	(94)
9659	(94)	9660	(94)	9661	(94)	9662	(94)	9663	(94)	9664	(94)	9665	(94)	9666	(94)
9667	(94)	9668	(94)	9705	(95)	9706	(95)	9707	(95)	9708	(95)	9709	(95)	9710	(95)
9714	(95)	9715	(95)	9716	(95)	9717	(95)	9719	(95)	9720	(95)	9724	(95)	9725	(95)
9726	(95)	9727	(95)	9728	(95)	9729	(95)	9730	(95)	9731	(95)	9732	(95)	9733	(95)
9734	(95)	9735	(95)	9736	(95)	9737	(95)	9738	(95)	9739	(95)	9740	(95)	9741	(95)
9742	(95)	9743	(95)	9744	(95)	9745	(95)	9746	(95)	9747	(95)	9748	(95)	9749	(95)
9750	(95)	9751	(95)	9752	(95)	9753	(95)	9754	(95)	9762	(95)	9763	(95)	9764	(95)
9765	(95)	9766	(95)	9767	(95)	9768	(95)	9769	(95)	9770	(95)	9771	(95)	9772	(95)
9776	(95)	9777	(95)	9778	(95)	9779	(95)	9780	(95)	9781	(95)	9782	(95)	9787	(95)
9788	(95)	9789	(95)	9790	(95)	9791	(95)	9792	(95)	9793	(95)	9794	(95)	9795	(95)
9796	(95)	9797	(95)	9798	(95)	9799	(95)	9800	(95)	9801	(95)	9802	(95)	9803	(95)
9804	(95)	9805	(95)	9806	(95)	9807	(95)	9808	(95)	9809	(95)	9810	(95)	9811	(95)
9834	(96)	9835	(96)	9836	(96)	9837	(96)	9838	(96)	9839	(96)	9840	(96)	9841	(96)
9843	(96)	9845	(96)	9846	(96)	9847	(96)	9863	(97)	9864	(97)	9865	(97)	9866	(97)
9867	(97)	9868	(97)	9891	(98)	9892	(98)	9893	(98)	9894	(98)	9895	(98)	9896	(98)
9897	(98)	9898	(98)	9900	(98)	9902	(98)	9903	(98)	9904	(98)				
9914	(98)														
1374	(5)	1375	(5)	1382	(5)	1384	(5)	1385	(5)	1386	(5)	1387	(5)	1388	(5)
1393	(5)	1395	(5)	1396	(5)	1397	(5)	1400	(5)	1402	(5)	1404	(5)	1407	(5)
1408	(5)	1429	(5)	1468	(6)	1526	(6)	1527	(6)	1528	(6)	1529	(6)	1530	(6)
1531	(6)	1532	(6)	1533	(6)	1534	(6)	1535	(6)	1536	(6)	1537	(6)	1538	(6)
1539	(6)	1540	(6)	2264	(7)	2266	(7)	2269	(7)	2270	(7)	2277	(7)	2280	(7)
2288	(7)	2294	(7)	2314	(7)	2320	(7)	2361	(7)	2362	(7)	2378	(7)	2379	(7)
2380	(7)	2381	(7)	2383	(7)	2467	(8)	2469	(8)	2471	(8)	2518	(9)	2568	(10)
2569	(10)	2570	(10)	2571	(10)	2572	(10)	2628	(11)	2630	(11)	2634	(11)	2695	(12)
2702	(12)	2703	(12)	2708	(12)	2710	(12)	2753	(13)	2800	(14)	2805	(14)	2807	(14)
2817	(14)	2820	(14)	2821	(14)	2822	(14)	2825	(14)	2826	(14)	2827	(14)	2833	(14)
2834	(14)	2835	(14)	2841	(14)	2848	(14)	2892	(15)	2893	(15)	2897	(15)	2899	(15)
2941	(16)	2942	(16)	2946	(16)	2948	(16)	2951	(16)	3043	(18)	3050	(18)	3055	(18)
3058	(18)	3063	(18)	3103	(19)	3109	(19)	3114	(19)	3117	(19)	3122	(19)	3207	(20)
3212	(20)	3213	(20)	3216	(20)	3217	(20)	3223	(20)	3224	(20)	3225	(20)	3228	(20)
3229	(20)	3230	(20)	3231	(20)	3236	(20)	3238	(20)	3242	(20)	3243	(20)	3247	(20)
3248	(20)	3251	(20)	3252	(20)	3253	(20)	3257	(20)	3260	(20)	3262	(20)	3263	(20)
3267	(20)	3269	(20)	3270	(20)	3271	(20)	3272	(20)	3278	(20)	3282	(20)	3284	(20)
3286	(20)	3287	(20)	3291	(20)	3292	(20)	3296	(20)	3297	(20)	3298	(20)	3305	(20)
3307	(20)	3308	(20)	3313	(20)	3314	(20)	3317	(20)	3318	(20)	3323	(20)	3324	(20)
3325	(20)	3326	(20)	3327	(20)	3333	(20)	3337	(20)	3339	(20)	3341	(20)	3342	(20)
3348	(20)	3349	(20)	3350	(20)	3353	(20)	3354	(20)	3358	(20)	3359	(20)	3361	(20)
3363	(20)	3364	(20)	3366	(20)	3367	(20)	3369	(20)	3370	(20)	3373	(20)	3374	(20)
3375	(20)	3380	(20)	3384	(20)	3394	(20)	3400	(20)	3403	(20)	3406	(20)	3412	(20)
3413	(20)	3414	(20)	3415	(20)	3417	(20)	3419	(20)	3420	(20)	3422	(20)	3423	(20)
3425	(20)	3462	(21)	3466	(21)	3467	(21)	3473	(21)	3475	(21)	3476	(21)	3478	(21)
3479	(21)	3481	(21)	3482	(21)	3485	(21)	3521	(22)	3524	(22)	3525	(22)	3605	(23)
3606	(23)	3607	(23)	3608	(23)	3611	(23)	3612	(23)	3618	(23)	3626	(23)	3627	(23)
3629	(23)	3631	(23)	3632	(23)	3633	(23)	3634	(23)	3636	(23)	3637	(23)	3641	(23)
3642	(23)	3643	(23)	3644	(23)	3645	(23)	3647	(23)	3648	(23)	3649	(23)	3650	(23)
3654	(23)	3655	(23)	3661	(23)	3678	(23)	3679	(23)	3680	(23)	3683	(23)	3684	(23)
3685	(23)	3688	(23)	3690	(23)	3691	(23)	3693	(23)	3694	(23)	3699	(23)	3706	(23)
3707	(23)	3709	(23)	3710	(23)	3717	(23)	3718	(23)	3720	(23)	3722	(23)	3723	(23)
3801	(24)	3811	(24)	3812	(24)	3813	(24)	3817	(24)	3822	(24)	3824	(24)	3829	(24)
3833	(24)	3836	(24)	3856	(24)	3862	(24)	3863	(24)	3894	(25)	3967	(27)	3971	(27)
3985	(27)	4004	(28)	4005	(28)	4006	(28)	4007	(28)	4009	(28)	4011	(28)	4012	(28)

.END
.ENDC

4013	(28)	4014	(28)	4015	(28)	4016	(28)	4035	(29)	4036	(29)	4037	(29)	4038	(29)
4040	(29)	4042	(29)	4043	(29)	4044	(29)	4045	(29)	4046	(29)	4047	(29)	4099	(31)
4122	(32)	4125	(32)	4126	(32)	4127	(32)	4129	(32)	4130	(32)	4131	(32)	4132	(32)
4133	(32)	4134	(32)	4137	(32)	4155	(33)	4156	(33)	4157	(33)	4158	(33)	4208	(34)
4209	(34)	4210	(34)	4212	(34)	4217	(34)	4218	(34)	4219	(34)	4231	(34)	4239	(34)
4241	(34)	4243	(34)	4246	(34)	4247	(34)	4248	(34)	4249	(34)	4250	(34)	4251	(34)
4253	(34)	4254	(34)	4265	(34)	4266	(34)	4269	(34)	4272	(34)	4275	(34)	4281	(34)
4309	(35)	4311	(35)	4312	(35)	4313	(35)	4319	(35)	4321	(35)	4334	(35)	4337	(35)
4339	(35)	4344	(35)	4349	(35)	4355	(35)	4356	(35)	4375	(35)	4377	(35)	4432	(36)
4445	(36)	4446	(36)	4449	(36)	4451	(36)	4461	(36)	4462	(36)	4469	(36)	4514	(37)
4518	(37)	4523	(37)	4526	(37)	4568	(38)	4570	(38)	4571	(38)	4573	(38)	4616	(39)
4618	(39)	4619	(39)	4620	(39)	4621	(39)	4622	(39)	4623	(39)	4624	(39)	4628	(39)
4633	(39)	4649	(39)	4652	(39)	4653	(39)	4655	(39)	4656	(39)	4658	(39)	4659	(39)
4660	(39)	4666	(39)	4672	(39)	4673	(39)	4675	(39)	4676	(39)	4678	(39)	4685	(39)
4686	(39)	4689	(39)	4690	(39)	4691	(39)	4692	(39)	4696	(39)	4775	(40)	4777	(40)
4782	(40)	4784	(40)	4838	(40)	4842	(40)	4843	(40)	4845	(40)	4847	(40)	4851	(40)
4858	(40)	4859	(40)	4861	(40)	4869	(40)	4879	(40)	4880	(40)	4882	(40)	4948	(41)
4954	(41)	4957	(41)	4983	(41)	4987	(41)	4989	(41)	4990	(41)	4991	(41)	4992	(41)
4994	(41)	4995	(41)	4996	(41)	4999	(41)	5002	(41)	5003	(41)	5004	(41)	5005	(41)
5006	(41)	5152	(45)	5154	(45)	5169	(45)	5170	(45)	5172	(45)	5173	(45)	5174	(45)
5183	(45)	5185	(45)	5196	(45)	5205	(45)	5210	(45)	5211	(45)	5212	(45)	5214	(45)
5215	(45)	5217	(45)	5224	(45)	5226	(45)	5243	(45)	5246	(45)	5247	(45)	5248	(45)
5250	(45)	5251	(45)	5277	(45)	5283	(45)	5284	(45)	5285	(45)	5288	(45)	5289	(45)
5350	(46)	5352	(46)	5375	(46)	5378	(46)	5379	(46)	5380	(46)	5383	(46)	5384	(46)
5393	(46)	5395	(46)	5400	(46)	5402	(46)	5412	(46)	5413	(46)	5415	(46)	5416	(46)
5417	(46)	5478	(47)	5480	(47)	5481	(47)	5484	(47)	5487	(47)	5489	(47)	5521	(47)
5524	(47)	5525	(47)	5532	(47)	5535	(47)	5536	(47)	5575	(47)	5576	(47)	5582	(47)
5585	(47)	5586	(47)	5589	(47)	5591	(47)	5592	(47)	5601	(47)	5602	(47)	5617	(47)
5618	(47)	5627	(47)	5628	(47)	5639	(47)	5645	(47)	5716	(48)	5800	(49)	5801	(49)
5804	(49)	5810	(49)	5814	(49)	5825	(49)	5826	(49)	5827	(49)	5828	(49)	5832	(49)
5833	(49)	5836	(49)	5837	(49)	5838	(49)	5839	(49)	5849	(49)	5851	(49)	5852	(49)
5853	(49)	5854	(49)	5855	(49)	5856	(49)	5857	(49)	5858	(49)	5912	(50)	5913	(50)
5920	(50)	5923	(50)	5928	(50)	5930	(50)	6021	(51)	6041	(51)	6171	(53)	6173	(53)
6174	(53)	6175	(53)	6176	(53)	6183	(53)	6191	(53)	6200	(53)	6205	(53)	6215	(53)
6217	(53)	6218	(53)	6220	(53)	6268	(54)	6269	(54)	6270	(54)	6271	(54)	6273	(54)
6274	(54)	6279	(54)	6280	(54)	6281	(54)	6282	(54)	6283	(54)	6284	(54)	6285	(54)
6286	(54)	6287	(54)	6291	(54)	6292	(54)	6293	(54)	6294	(54)	6295	(54)	6297	(54)
6299	(54)	6303	(54)	6304	(54)	6305	(54)	6306	(54)	6307	(54)	6309	(54)	6313	(54)
6319	(54)	6325	(54)	6330	(54)	6331	(54)	6333	(54)	6334	(54)	6343	(54)	6344	(54)
6345	(54)	6428	(56)	6429	(56)	6438	(56)	6441	(56)	6442	(56)	6447	(56)	6448	(56)
6449	(56)	6450	(56)	6453	(56)	6454	(56)	6455	(56)	6456	(56)	6460	(56)	6464	(56)
6465	(56)	6477	(56)	6478	(56)	6479	(56)	6480	(56)	6493	(56)	6494	(56)	6495	(56)
6497	(56)	6498	(56)	6499	(56)	6502	(56)	6503	(56)	6507	(56)	6552	(57)	6555	(57)
6563	(57)	6565	(57)	6569	(57)	6570	(57)	6576	(57)	6577	(57)	6580	(57)	6581	(57)
6582	(57)	6583	(57)	6626	(58)	6639	(58)	6646	(58)	6655	(58)	6661	(58)	6662	(58)
6665	(58)	6687	(58)	6835	(61)	6838	(61)	6839	(61)	6848	(61)	6851	(61)	6855	(61)
6857	(61)	6858	(61)	6860	(61)	6861	(61)	6862	(61)	6917	(62)	6919	(62)	6924	(62)
6927	(62)	6935	(62)	7007	(63)	7009	(63)	7014	(63)	7017	(63)	7028	(63)	7031	(63)
7035	(63)	7036	(63)	7038	(63)	7039	(63)	7041	(63)	7042	(63)	7043	(63)	7049	(63)
7050	(63)	7052	(63)	7053	(63)	7055	(63)	7056	(63)	7057	(63)	7065	(63)	7131	(64)
7143	(64)	7147	(64)	7150	(64)	7152	(64)	7154	(64)	7162	(64)	7164	(64)	7165	(64)
7170	(64)	7171	(64)	7177	(64)	7197	(64)	7199	(64)	7297	(65)	7311	(65)	7314	(65)
7320	(65)	7324	(65)	7426	(66)	7448	(66)	7451	(66)	7490	(66)	7491	(66)	7497	(66)
7504	(66)	7505	(66)	7507	(66)	7523	(66)	7525	(66)	7527	(66)	7529	(66)	7539	(66)
7551	(66)	7561	(66)	7562	(66)	7575	(66)	7577	(66)	7611	(67)	7622	(67)	7623	(67)

7629	(67)	7680	(67)	7681	(67)	7682	(67)	7686	(67)	7687	(67)	7693	(67)	7714	(67)
7726	(67)	7728	(67)	7735	(67)	7736	(67)	7737	(67)	7738	(67)	7739	(67)	7743	(67)
7746	(67)	7747	(67)	7749	(67)	7750	(67)	7752	(67)	7754	(67)	7756	(67)	7760	(67)
7761	(67)	7767	(67)	7768	(67)	7769	(67)	7771	(67)	7776	(67)	7777	(67)	7778	(67)
7780	(67)	7781	(67)	7786	(67)	7787	(67)	7788	(67)	7790	(67)	7791	(67)	7824	(68)
7861	(69)	7914	(70)	7917	(70)	7924	(70)	7926	(70)	7928	(70)	7929	(70)	7930	(70)
7933	(70)	7937	(70)	7941	(70)	7949	(70)	7951	(70)	7954	(70)	7956	(70)	7957	(70)
7960	(70)	7961	(70)	7964	(70)	7965	(70)	7966	(70)	7971	(70)	7974	(70)	7976	(70)
7977	(70)	7978	(70)	7981	(70)	7982	(70)	7985	(70)	7986	(70)	7987	(70)	7988	(70)
7989	(70)	7990	(70)	7992	(70)	8032	(71)	8045	(71)	8048	(71)	8050	(71)	8060	(71)
8062	(71)	8068	(71)	8069	(71)	8074	(71)	8075	(71)	8076	(71)	8077	(71)	8078	(71)
8079	(71)	8108	(72)	8109	(72)	8110	(72)	8111	(72)	8143	(73)	8144	(73)	8145	(73)
8146	(73)	8242	(76)	8244	(76)	8247	(76)	8248	(76)	8249	(76)	8254	(76)	8260	(76)
8261	(76)	8262	(76)	8267	(76)	8271	(76)	8274	(76)	8275	(76)	8281	(76)	8287	(76)
8297	(76)	8303	(76)	8308	(76)	8310	(76)	8313	(76)	8314	(76)	8316	(76)	8317	(76)
8320	(76)	8321	(76)	8322	(76)	8325	(76)	8326	(76)	8327	(76)	8334	(76)	8336	(76)
8337	(76)	8339	(76)	8340	(76)	8342	(76)	8343	(76)	8345	(76)	8346	(76)	8347	(76)
8349	(76)	8390	(77)	8391	(77)	8420	(78)	8424	(78)	8426	(78)	8442	(79)	8443	(79)
8448	(79)	8449	(79)	8452	(79)	8454	(79)	8455	(79)	8456	(79)	8457	(79)	8460	(79)
8497	(80)	8502	(80)	8503	(80)	8504	(80)	8505	(80)	8551	(81)	8557	(81)	8558	(81)
8560	(81)	8564	(81)	8565	(81)	8566	(81)	8572	(81)	8573	(81)	8574	(81)	8575	(81)
8576	(81)	8579	(81)	8624	(82)	863	(4)	8630	(82)	8670	(83)	8671	(83)	8675	(83)
8676	(83)	8679	(83)	8680	(83)	8681	(83)	8682	(83)	8683	(83)	8723	(84)	8724	(84)
8728	(84)	8729	(84)	8732	(84)	8733	(84)	8734	(84)	8735	(84)	8736	(84)	8781	(85)
8782	(85)	8783	(85)	8784	(85)	8787	(85)	8790	(85)	8791	(85)	8802	(85)	8805	(85)
8806	(85)	8845	(86)	8915	(87)	8916	(87)	8917	(87)	8924	(87)	8927	(87)	8928	(87)
8930	(87)	8934	(87)	8935	(87)	8939	(87)	8945	(87)	8946	(87)	8947	(87)	8951	(87)
8952	(87)	8956	(87)	8957	(87)	8958	(87)	8963	(87)	8969	(87)	8970	(87)	8971	(87)
8973	(87)	8979	(87)	8980	(87)	8981	(87)	8983	(87)	8989	(87)	8990	(87)	8991	(87)
8993	(87)	8998	(87)	8999	(87)	9002	(87)	9005	(87)	9011	(87)	9016	(87)	9020	(87)
9022	(87)	9024	(87)	9034	(87)	9037	(87)	9040	(87)	9041	(87)	9042	(87)	9043	(87)
9044	(87)	9045	(87)	9047	(87)	9048	(87)	9049	(87)	9051	(87)	9052	(87)	9053	(87)
9056	(87)	9061	(87)	9062	(87)	9069	(87)	9070	(87)	9073	(87)	9074	(87)	9075	(87)
9076	(87)	9077	(87)	9078	(87)	9079	(87)	9081	(87)	9082	(87)	9084	(87)	9085	(87)
9086	(87)	9087	(87)	9089	(87)	9090	(87)	9091	(87)	9092	(87)	9093	(87)	9095	(87)
9096	(87)	9098	(87)	9099	(87)	9138	(88)	9139	(88)	9140	(88)	9144	(88)	9145	(88)
9148	(88)	9150	(88)	9151	(88)	9165	(88)	9212	(89)	9213	(89)	9215	(89)	9218	(89)
9219	(89)	9220	(89)	9221	(89)	9224	(89)	9226	(89)	9227	(89)	9228	(89)	9229	(89)
9235	(89)	9236	(89)	9247	(89)	9249	(89)	9250	(89)	9252	(89)	9253	(89)	9259	(89)
9272	(89)	9273	(89)	9281	(89)	9282	(89)	9283	(89)	9285	(89)	9287	(89)	9294	(89)
9301	(89)	9302	(89)	9303	(89)	9307	(89)	9308	(89)	9309	(89)	9311	(89)	9312	(89)
9313	(89)	9315	(89)	9316	(89)	9319	(89)	9320	(89)	9335	(89)	9336	(89)	9337	(89)
9342	(89)	9343	(89)	9344	(89)	9345	(89)	9346	(89)	9349	(89)	9350	(89)	9392	(90)
9396	(90)	9399	(90)	9402	(90)	9404	(90)	9405	(90)	9406	(90)	9412	(90)	9413	(90)
9416	(90)	9417	(90)	9419	(90)	9420	(90)	9422	(90)	9425	(90)	9429	(90)	9430	(90)
9431	(90)	9434	(90)	9437	(90)	9439	(90)	9440	(90)	9441	(90)	9447	(90)	9448	(90)
9449	(90)	9450	(90)	9452	(90)	9453	(90)	9455	(90)	9458	(90)	9481	(91)	9482	(91)
9488	(91)	9519	(92)	9520	(92)	9521	(92)	9525	(92)	9529	(92)	9532	(92)	9535	(92)
9536	(92)	9539	(92)	9584	(93)	9625	(94)	9626	(94)	9627	(94)	9628	(94)	9629	(94)
9630	(94)	9631	(94)	9632	(94)	9633	(94)	9634	(94)	9635	(94)	9636	(94)	9641	(94)
9644	(94)	9645	(94)	9650	(94)	9652	(94)	9653	(94)	9654	(94)	9655	(94)	9656	(94)
9657	(94)	9658	(94)	9660	(94)	9661	(94)	9663	(94)	9664	(94)	9665	(94)	9666	(94)
9667	(94)	9706	(95)	9708	(95)	9710	(95)	9714	(95)	9715	(95)	9716	(95)	9719	(95)
9720	(95)	9724	(95)	9725	(95)	9726	(95)	9727	(95)	9728	(95)	9729	(95)	9731	(95)
9732	(95)	9733	(95)	9734	(95)	9735	(95)	9737	(95)	9738	(95)	9739	(95)	9740	(95)

	9741	(95)	9743	(95)	9744	(95)	9745	(95)	9746	(95)	9747	(95)	9749	(95)	9750	(95)
	9751	(95)	9752	(95)	9753	(95)	9754	(95)	9762	(95)	9763	(95)	9766	(95)	9767	(95)
	9769	(95)	9776	(95)	9777	(95)	9778	(95)	9779	(95)	9787	(95)	9788	(95)	9789	(95)
	9793	(95)	9796	(95)	9797	(95)	9799	(95)	9801	(95)	9802	(95)	9803	(95)	9805	(95)
	9806	(95)	9808	(95)	9809	(95)	9810	(95)	9835	(96)	9836	(96)	9840	(96)	9843	(96)
	9845	(96)	9846	(96)	9863	(97)	9864	(97)	9865	(97)	9866	(97)	9867	(97)	9892	(98)
	9893	(98)	9897	(98)	9900	(98)	9902	(98)	9903	(98)						
.ENDM	1375	(5)	1418	(5)	2264	(7)	2265	(7)	2471	(8)	2518	(9)	2633	(11)	2700	(12)
	462	(2)	488	(2)	516	(2)	547	(2)	578	(2)	608	(2)	654	(2)	705	(2)
	728	(2)	752	(2)	765	(2)	783	(2)	816	(2)	841	(2)	849	(2)	856	(3)
	863	(4)														
.ENDR	1374	(5)	1385	(5)	1393	(5)	1526	(6)	1527	(6)	1528	(6)	1529	(6)	1530	(6)
	1531	(6)	1532	(6)	1533	(6)	1534	(6)	1535	(6)	1536	(6)	1537	(6)	1538	(6)
	1539	(6)	1540	(6)	2269	(7)	2270	(7)	2314	(7)	2361	(7)	2362	(7)	2378	(7)
	2379	(7)	2380	(7)	2381	(7)	2568	(10)	2569	(10)	2570	(10)	2571	(10)	2572	(10)
	2708	(12)	2710	(12)	2807	(14)	2893	(15)	2942	(16)	3207	(20)	3230	(20)	3252	(20)
	3253	(20)	3260	(20)	3262	(20)	3267	(20)	3305	(20)	3307	(20)	3313	(20)	3314	(20)
	3323	(20)	3358	(20)	3369	(20)	3403	(20)	3521	(22)	3606	(23)	3627	(23)	3637	(23)
	3650	(23)	3661	(23)	3679	(23)	3680	(23)	3811	(24)	3812	(24)	3813	(24)	3817	(24)
	3856	(24)	3862	(24)	3863	(24)	3894	(25)	4009	(28)	4011	(28)	4040	(29)	4042	(29)
	4122	(32)	4129	(32)	4137	(32)	4157	(33)	4209	(34)	4210	(34)	4217	(34)	4218	(34)
	4219	(34)	4231	(34)	4241	(34)	4251	(34)	4254	(34)	4265	(34)	4272	(34)	4309	(35)
	4311	(35)	4313	(35)	4334	(35)	4616	(39)	4652	(39)	4672	(39)	4685	(39)	4954	(41)
	4957	(41)	4983	(41)	4987	(41)	4994	(41)	5004	(41)	5480	(47)	5481	(47)	5575	(47)
	5589	(47)	5591	(47)	5601	(47)	5617	(47)	5627	(47)	5639	(47)	5645	(47)	5827	(49)
	5849	(49)	5857	(49)	6171	(53)	6428	(56)	6494	(56)	6503	(56)	7036	(63)	7050	(63)
	7131	(64)	7297	(65)	7320	(65)	7426	(66)	7490	(66)	7491	(66)	7507	(66)	7523	(66)
	7529	(66)	7629	(67)	7680	(67)	7681	(67)	7682	(67)	7686	(67)	7687	(67)	7693	(67)
	7726	(67)	7735	(67)	7736	(67)	7737	(67)	7738	(67)	7739	(67)	7749	(67)	7750	(67)
	7756	(67)	7760	(67)	7761	(67)	7771	(67)	7780	(67)	7781	(67)	7790	(67)	7791	(67)
	7924	(70)	7933	(70)	8069	(71)	8247	(76)	8260	(76)	8261	(76)	8262	(76)	8267	(76)
	8346	(76)	8442	(79)	8443	(79)	8452	(79)	8455	(79)	8460	(79)	8497	(80)	8557	(81)
	8671	(83)	8676	(83)	8724	(84)	8729	(84)	8782	(85)	8783	(85)	8845	(86)	8916	(87)
	8917	(87)	8930	(87)	8934	(87)	8935	(87)	8939	(87)	8945	(87)	8946	(87)	8947	(87)
	8951	(87)	8952	(87)	8956	(87)	8957	(87)	8958	(87)	8963	(87)	8973	(87)	8983	(87)
	8993	(87)	9011	(87)	9042	(87)	9043	(87)	9049	(87)	9076	(87)	9081	(87)	9090	(87)
	9091	(87)	9099	(87)	9139	(88)	9140	(88)	9150	(88)	9224	(89)	9249	(89)	9259	(89)
	9272	(89)	9287	(89)	9337	(89)	9396	(90)	9431	(90)	9519	(92)	9628	(94)	9629	(94)
	9633	(94)	9652	(94)	9655	(94)	9656	(94)	9665	(94)	9706	(95)	9726	(95)	9727	(95)
	9728	(95)	9835	(96)	9836	(96)	9843	(96)	9845	(96)	9864	(97)	9867	(97)	9892	(98)
	9893	(98)	9900	(98)	9902	(98)										
.IDENT	3	(1)														
.IF	1374	(5)	1375	(5)	1384	(5)	1385	(5)	1386	(5)	1387	(5)	1388	(5)	1393	(5)
	1395	(5)	1396	(5)	1397	(5)	1400	(5)	1402	(5)	1404	(5)	1407	(5)	1408	(5)
	1429	(5)	1468	(6)	1526	(6)	1527	(6)	1528	(6)	1529	(6)	1530	(6)	1531	(6)
	1532	(6)	1533	(6)	1534	(6)	1535	(6)	1536	(6)	1537	(6)	1538	(6)	1539	(6)
	1540	(6)	2264	(7)	2266	(7)	2269	(7)	2270	(7)	2277	(7)	2280	(7)	2288	(7)
	2294	(7)	2314	(7)	2320	(7)	2361	(7)	2362	(7)	2378	(7)	2379	(7)	2380	(7)
	2381	(7)	2383	(7)	2467	(8)	2469	(8)	2471	(8)	2518	(9)	2568	(10)	2569	(10)
	2570	(10)	2571	(10)	2572	(10)	2628	(11)	2630	(11)	2634	(11)	2695	(12)	2702	(12)
	2703	(12)	2708	(12)	2710	(12)	2753	(13)	2800	(14)	2805	(14)	2807	(14)	2817	(14)
	2820	(14)	2821	(14)	2822	(14)	2825	(14)	2826	(14)	2827	(14)	2833	(14)	2834	(14)
	2835	(14)	2841	(14)	2848	(14)	2892	(15)	2893	(15)	2897	(15)	2899	(15)	2941	(16)
	2942	(16)	2946	(16)	2948	(16)	2951	(16)	3043	(18)	3050	(18)	3055	(18)	3058	(18)
	3063	(18)	3103	(19)	3109	(19)	3114	(19)	3117	(19)	3122	(19)	3207	(20)	3212	(20)

3213	(20)	3216	(20)	3217	(20)	3223	(20)	3224	(20)	3225	(20)	3228	(20)	3229	(20)
3230	(20)	3231	(20)	3236	(20)	3238	(20)	3242	(20)	3243	(20)	3247	(20)	3248	(20)
3251	(20)	3252	(20)	3253	(20)	3257	(20)	3260	(20)	3262	(20)	3263	(20)	3267	(20)
3269	(20)	3270	(20)	3271	(20)	3272	(20)	3278	(20)	3282	(20)	3284	(20)	3286	(20)
3287	(20)	3291	(20)	3292	(20)	3296	(20)	3297	(20)	3298	(20)	3305	(20)	3307	(20)
3308	(20)	3313	(20)	3314	(20)	3317	(20)	3318	(20)	3323	(20)	3324	(20)	3325	(20)
3326	(20)	3327	(20)	3333	(20)	3337	(20)	3339	(20)	3341	(20)	3342	(20)	3348	(20)
3349	(20)	3350	(20)	3353	(20)	3354	(20)	3358	(20)	3359	(20)	3361	(20)	3363	(20)
3364	(20)	3366	(20)	3367	(20)	3369	(20)	3370	(20)	3373	(20)	3374	(20)	3375	(20)
3380	(20)	3384	(20)	3394	(20)	3400	(20)	3403	(20)	3406	(20)	3412	(20)	3413	(20)
3414	(20)	3415	(20)	3417	(20)	3419	(20)	3420	(20)	3422	(20)	3423	(20)	3425	(20)
3462	(21)	3466	(21)	3467	(21)	3473	(21)	3475	(21)	3476	(21)	3478	(21)	3479	(21)
3481	(21)	3482	(21)	3485	(21)	3521	(22)	3524	(22)	3525	(22)	3605	(23)	3606	(23)
3607	(23)	3608	(23)	3611	(23)	3612	(23)	3618	(23)	3626	(23)	3627	(23)	3629	(23)
3631	(23)	3632	(23)	3633	(23)	3634	(23)	3636	(23)	3637	(23)	3641	(23)	3642	(23)
3643	(23)	3644	(23)	3645	(23)	3647	(23)	3648	(23)	3649	(23)	3650	(23)	3654	(23)
3655	(23)	3661	(23)	3678	(23)	3679	(23)	3680	(23)	3683	(23)	3684	(23)	3685	(23)
3688	(23)	3690	(23)	3691	(23)	3693	(23)	3694	(23)	3699	(23)	3706	(23)	3707	(23)
3709	(23)	3710	(23)	3717	(23)	3718	(23)	3720	(23)	3722	(23)	3723	(23)	3801	(24)
3811	(24)	3812	(24)	3813	(24)	3817	(24)	3822	(24)	3824	(24)	3829	(24)	3833	(24)
3836	(24)	3856	(24)	3862	(24)	3863	(24)	3894	(25)	3967	(27)	3971	(27)	3985	(27)
4004	(28)	4005	(28)	4006	(28)	4007	(28)	4009	(28)	4011	(28)	4012	(28)	4013	(28)
4014	(28)	4015	(28)	4016	(28)	4035	(29)	4036	(29)	4037	(29)	4038	(29)	4040	(29)
4042	(29)	4043	(29)	4044	(29)	4045	(29)	4046	(29)	4047	(29)	4099	(31)	4122	(32)
4125	(32)	4126	(32)	4127	(32)	4129	(32)	4130	(32)	4131	(32)	4132	(32)	4133	(32)
4134	(32)	4137	(32)	4155	(33)	4156	(33)	4157	(33)	4158	(33)	4208	(34)	4209	(34)
4210	(34)	4212	(34)	4217	(34)	4218	(34)	4219	(34)	4231	(34)	4239	(34)	4241	(34)
4243	(34)	4246	(34)	4247	(34)	4248	(34)	4249	(34)	4250	(34)	4251	(34)	4253	(34)
4254	(34)	4265	(34)	4266	(34)	4269	(34)	4272	(34)	4275	(34)	4281	(34)	4309	(35)
4311	(35)	4312	(35)	4313	(35)	4319	(35)	4321	(35)	4334	(35)	4337	(35)	4339	(35)
4344	(35)	4349	(35)	4355	(35)	4356	(35)	4375	(35)	4377	(35)	4432	(36)	4445	(36)
4446	(36)	4449	(36)	4451	(36)	4461	(36)	4462	(36)	4469	(36)	4514	(37)	4518	(37)
4523	(37)	4526	(37)	4568	(38)	4570	(38)	4571	(38)	4573	(38)	4616	(39)	4618	(39)
4619	(39)	4620	(39)	4621	(39)	4622	(39)	4623	(39)	4624	(39)	4628	(39)	4633	(39)
4649	(39)	4652	(39)	4653	(39)	4655	(39)	4656	(39)	4658	(39)	4659	(39)	4660	(39)
4666	(39)	4672	(39)	4673	(39)	4675	(39)	4676	(39)	4678	(39)	4685	(39)	4686	(39)
4689	(39)	4690	(39)	4691	(39)	4692	(39)	4696	(39)	4775	(40)	4777	(40)	4782	(40)
4784	(40)	4838	(40)	4842	(40)	4843	(40)	4845	(40)	4847	(40)	4851	(40)	4858	(40)
4859	(40)	4861	(40)	4869	(40)	4879	(40)	4880	(40)	4882	(40)	4948	(41)	4954	(41)
4957	(41)	4983	(41)	4987	(41)	4989	(41)	4990	(41)	4991	(41)	4992	(41)	4994	(41)
4995	(41)	4996	(41)	4999	(41)	5002	(41)	5003	(41)	5004	(41)	5005	(41)	5006	(41)
5152	(45)	5154	(45)	5169	(45)	5170	(45)	5172	(45)	5173	(45)	5174	(45)	5183	(45)
5185	(45)	5196	(45)	5205	(45)	5210	(45)	5211	(45)	5212	(45)	5214	(45)	5215	(45)
5217	(45)	5224	(45)	5226	(45)	5243	(45)	5246	(45)	5247	(45)	5248	(45)	5250	(45)
5251	(45)	5277	(45)	5283	(45)	5284	(45)	5285	(45)	5288	(45)	5289	(45)	5350	(46)
5352	(46)	5375	(46)	5378	(46)	5379	(46)	5380	(46)	5383	(46)	5384	(46)	5393	(46)
5395	(46)	5400	(46)	5402	(46)	5412	(46)	5413	(46)	5415	(46)	5416	(46)	5417	(46)
5478	(47)	5480	(47)	5481	(47)	5484	(47)	5487	(47)	5489	(47)	5521	(47)	5524	(47)
5525	(47)	5532	(47)	5535	(47)	5536	(47)	5575	(47)	5576	(47)	5582	(47)	5585	(47)
5586	(47)	5589	(47)	5591	(47)	5592	(47)	5601	(47)	5602	(47)	5617	(47)	5618	(47)
5627	(47)	5628	(47)	5639	(47)	5645	(47)	5716	(48)	5800	(49)	5801	(49)	5804	(49)
5810	(49)	5814	(49)	5825	(49)	5826	(49)	5827	(49)	5828	(49)	5832	(49)	5833	(49)
5836	(49)	5837	(49)	5838	(49)	5839	(49)	5849	(49)	5851	(49)	5852	(49)	5853	(49)
5854	(49)	5855	(49)	5856	(49)	5857	(49)	5858	(49)	5912	(50)	5913	(50)	5920	(50)
5923	(50)	5928	(50)	5930	(50)	6021	(51)	6041	(51)	6171	(53)	6173	(53)	6174	(53)

6175	(53)	6176	(53)	6183	(53)	6191	(53)	6200	(53)	6205	(53)	6215	(53)	6217	(53)
6218	(53)	6220	(53)	6268	(54)	6269	(54)	6270	(54)	6271	(54)	6273	(54)	6274	(54)
6279	(54)	6280	(54)	6281	(54)	6282	(54)	6283	(54)	6284	(54)	6285	(54)	6286	(54)
6287	(54)	6291	(54)	6292	(54)	6293	(54)	6294	(54)	6295	(54)	6297	(54)	6299	(54)
6303	(54)	6304	(54)	6305	(54)	6306	(54)	6307	(54)	6309	(54)	6313	(54)	6319	(54)
6325	(54)	6330	(54)	6331	(54)	6333	(54)	6334	(54)	6343	(54)	6344	(54)	6345	(54)
6428	(56)	6429	(56)	6438	(56)	6441	(56)	6442	(56)	6447	(56)	6448	(56)	6449	(56)
6450	(56)	6453	(56)	6454	(56)	6455	(56)	6456	(56)	6460	(56)	6464	(56)	6465	(56)
6477	(56)	6478	(56)	6479	(56)	6480	(56)	6493	(56)	6494	(56)	6495	(56)	6497	(56)
6498	(56)	6499	(56)	6502	(56)	6503	(56)	6507	(56)	6552	(57)	6555	(57)	6563	(57)
6565	(57)	6569	(57)	6570	(57)	6576	(57)	6577	(57)	6580	(57)	6581	(57)	6582	(57)
6583	(57)	6626	(58)	6639	(58)	6646	(58)	6655	(58)	6661	(58)	6662	(58)	6665	(58)
6687	(58)	6635	(61)	6838	(61)	6839	(61)	6848	(61)	6851	(61)	6855	(61)	6857	(61)
6858	(61)	6860	(61)	6861	(61)	6862	(61)	6917	(62)	6919	(62)	6924	(62)	6927	(62)
6935	(62)	7007	(63)	7009	(63)	7014	(63)	7017	(63)	7028	(63)	7031	(63)	7035	(63)
7036	(63)	7038	(63)	7039	(63)	7041	(63)	7042	(63)	7043	(63)	7049	(63)	7050	(63)
7052	(63)	7053	(63)	7055	(63)	7056	(63)	7057	(63)	7065	(63)	7131	(64)	7143	(64)
7147	(64)	7150	(64)	7152	(64)	7154	(64)	7162	(64)	7164	(64)	7165	(64)	7170	(64)
7171	(64)	7177	(64)	7197	(64)	7199	(64)	7297	(65)	7311	(65)	7314	(65)	7320	(65)
7324	(65)	7426	(66)	7448	(66)	7451	(66)	7490	(66)	7491	(66)	7497	(66)	7504	(66)
7505	(66)	7507	(66)	7523	(66)	7525	(66)	7527	(66)	7529	(66)	7539	(66)	7551	(66)
7561	(66)	7562	(66)	7575	(66)	7577	(66)	7611	(67)	7622	(67)	7623	(67)	7629	(67)
7680	(67)	7681	(67)	7682	(67)	7686	(67)	7687	(67)	7693	(67)	7714	(67)	7726	(67)
7728	(67)	7735	(67)	7736	(67)	7737	(67)	7738	(67)	7739	(67)	7743	(67)	7746	(67)
7747	(67)	7749	(67)	7750	(67)	7752	(67)	7754	(67)	7756	(67)	7760	(67)	7761	(67)
7767	(67)	7768	(67)	7769	(67)	7771	(67)	7776	(67)	7777	(67)	7778	(67)	7780	(67)
7781	(67)	7786	(67)	7787	(67)	7788	(67)	7790	(67)	7791	(67)	7824	(68)	7861	(69)
7914	(70)	7917	(70)	7924	(70)	7926	(70)	7928	(70)	7929	(70)	7930	(70)	7933	(70)
7937	(70)	7941	(70)	7949	(70)	7951	(70)	7954	(70)	7956	(70)	7957	(70)	7960	(70)
7961	(70)	7964	(70)	7965	(70)	7966	(70)	7971	(70)	7974	(70)	7976	(70)	7977	(70)
7978	(70)	7981	(70)	7982	(70)	7985	(70)	7986	(70)	7987	(70)	7988	(70)	7989	(70)
7990	(70)	7992	(70)	8032	(71)	8045	(71)	8048	(71)	8050	(71)	8060	(71)	8062	(71)
8068	(71)	8069	(71)	8074	(71)	8075	(71)	8076	(71)	8077	(71)	8078	(71)	8079	(71)
8108	(72)	8109	(72)	8110	(72)	8111	(72)	8143	(73)	8144	(73)	8145	(73)	8146	(73)
8242	(76)	8244	(76)	8247	(76)	8248	(76)	8249	(76)	8254	(76)	8260	(76)	8261	(76)
8262	(76)	8267	(76)	8271	(76)	8274	(76)	8275	(76)	8281	(76)	8287	(76)	8297	(76)
8303	(76)	8308	(76)	8310	(76)	8313	(76)	8314	(76)	8316	(76)	8317	(76)	8320	(76)
8321	(76)	8322	(76)	8325	(76)	8326	(76)	8327	(76)	8334	(76)	8336	(76)	8337	(76)
8339	(76)	8340	(76)	8342	(76)	8343	(76)	8345	(76)	8346	(76)	8347	(76)	8349	(76)
8390	(77)	8391	(77)	8420	(78)	8424	(78)	8426	(78)	8442	(79)	8443	(79)	8448	(79)
8449	(79)	8452	(79)	8454	(79)	8455	(79)	8456	(79)	8457	(79)	8460	(79)	8497	(80)
8502	(80)	8503	(80)	8504	(80)	8505	(80)	8551	(81)	8557	(81)	8558	(81)	8560	(81)
8564	(81)	8565	(81)	8566	(81)	8572	(81)	8573	(81)	8574	(81)	8575	(81)	8576	(81)
8579	(81)	8624	(82)	863	(4)	8630	(82)	8670	(83)	8671	(83)	8675	(83)	8676	(83)
8679	(83)	868	(5)	8680	(83)	8681	(83)	8682	(83)	8683	(83)	8723	(84)	8724	(84)
8728	(84)	8729	(84)	8732	(84)	8733	(84)	8734	(84)	8735	(84)	8736	(84)	8781	(85)
8782	(85)	8783	(85)	8784	(85)	8787	(85)	8790	(85)	8791	(85)	8802	(85)	8805	(85)
8806	(85)	8845	(86)	8915	(87)	8916	(87)	8917	(87)	8924	(87)	8927	(87)	8928	(87)
8930	(87)	8934	(87)	8935	(87)	8939	(87)	8945	(87)	8946	(87)	8947	(87)	8951	(87)
8952	(87)	8956	(87)	8957	(87)	8958	(87)	8963	(87)	8969	(87)	8970	(87)	8971	(87)
8973	(87)	8979	(87)	8980	(87)	8981	(87)	8983	(87)	8989	(87)	8990	(87)	8991	(87)
8993	(87)	8998	(87)	8999	(87)	9002	(87)	9005	(87)	9011	(87)	9016	(87)	9020	(87)
9022	(87)	9024	(87)	9034	(37)	9037	(87)	9040	(87)	9041	(87)	9042	(87)	9043	(87)
9044	(87)	9045	(87)	9047	(87)	9048	(87)	9049	(87)	9051	(87)	9052	(87)	9053	(87)
9056	(87)	9061	(87)	9062	(87)	9069	(87)	9070	(87)	9073	(87)	9074	(87)	9075	(87)

9076	(87)	9077	(87)	9078	(87)	9079	(87)	9081	(87)	9082	(87)	9084	(87)	9085	(87)
9086	(87)	9087	(87)	9089	(87)	9090	(87)	9091	(87)	9092	(87)	9093	(87)	9095	(87)
9096	(87)	9098	(87)	9099	(87)	9138	(88)	9139	(88)	9140	(88)	9144	(88)	9145	(88)
9148	(88)	9150	(88)	9151	(88)	9165	(88)	9212	(89)	9213	(89)	9215	(89)	9218	(89)
9219	(89)	9220	(89)	9221	(89)	9224	(89)	9226	(89)	9227	(89)	9228	(89)	9229	(89)
9235	(89)	9236	(89)	9247	(89)	9249	(89)	9250	(89)	9252	(89)	9253	(89)	9259	(89)
9272	(89)	9273	(89)	9281	(89)	9282	(89)	9283	(89)	9285	(89)	9287	(89)	9294	(89)
9301	(89)	9302	(89)	9303	(89)	9307	(89)	9308	(89)	9309	(89)	9311	(89)	9312	(89)
9313	(89)	9315	(89)	9316	(89)	9319	(89)	9320	(89)	9335	(89)	9336	(89)	9337	(89)
9342	(89)	9343	(89)	9344	(89)	9345	(89)	9346	(89)	9349	(89)	9350	(89)	9392	(90)
9396	(90)	9399	(90)	9402	(90)	9404	(90)	9405	(90)	9406	(90)	9412	(90)	9413	(90)
9416	(90)	9417	(90)	9419	(90)	9420	(90)	9422	(90)	9425	(90)	9429	(90)	9430	(90)
9431	(90)	9434	(90)	9437	(90)	9439	(90)	9440	(90)	9441	(90)	9447	(90)	9448	(90)
9449	(90)	9450	(90)	9452	(90)	9453	(90)	9455	(90)	9458	(90)	9481	(91)	9482	(91)
9488	(91)	9519	(92)	9520	(92)	9521	(92)	9525	(92)	9529	(92)	9532	(92)	9535	(92)
9536	(92)	9539	(92)	9584	(93)	9625	(94)	9626	(94)	9627	(94)	9628	(94)	9629	(94)
9630	(94)	9631	(94)	9632	(94)	9633	(94)	9634	(94)	9635	(94)	9636	(94)	9641	(94)
9644	(94)	9645	(94)	9650	(94)	9652	(94)	9653	(94)	9654	(94)	9655	(94)	9656	(94)
9657	(94)	9658	(94)	9660	(94)	9661	(94)	9663	(94)	9664	(94)	9665	(94)	9666	(94)
9667	(94)	9706	(95)	9708	(95)	9710	(95)	9714	(95)	9715	(95)	9716	(95)	9719	(95)
9720	(95)	9724	(95)	9725	(95)	9726	(95)	9727	(95)	9728	(95)	9729	(95)	9731	(95)
9732	(95)	9733	(95)	9734	(95)	9735	(95)	9737	(95)	9738	(95)	9739	(95)	9740	(95)
9741	(95)	9743	(95)	9744	(95)	9745	(95)	9746	(95)	9747	(95)	9749	(95)	9750	(95)
9751	(95)	9752	(95)	9753	(95)	9754	(95)	9762	(95)	9763	(95)	9766	(95)	9767	(95)
9769	(95)	9776	(95)	9777	(95)	9778	(95)	9779	(95)	9787	(95)	9788	(95)	9789	(95)
9793	(95)	9796	(95)	9797	(95)	9799	(95)	9801	(95)	9802	(95)	9803	(95)	9805	(95)
9806	(95)	9808	(95)	9809	(95)	9810	(95)	9835	(96)	9836	(96)	9840	(96)	9843	(96)
9845	(96)	9846	(96)	9863	(97)	9864	(97)	9865	(97)	9866	(97)	9867	(97)	9892	(98)
9893	(98)	9897	(98)	9900	(98)	9902	(98)	9903	(98)						
1367	(5)	1374	(5)	1385	(5)	1393	(5)	1468	(6)	1526	(6)	1527	(6)	1528	(6)
1529	(6)	1530	(6)	1531	(6)	1532	(6)	1533	(6)	1534	(6)	1535	(6)	1536	(6)
1537	(6)	1538	(6)	1539	(6)	1540	(6)	2264	(7)	2266	(7)	2269	(7)	2270	(7)
2314	(7)	2361	(7)	2362	(7)	2378	(7)	2379	(7)	2380	(7)	2381	(7)	2471	(8)
2518	(9)	2568	(10)	2569	(10)	2570	(10)	2571	(10)	2572	(10)	2634	(11)	2702	(12)
2708	(12)	2710	(12)	2753	(13)	2807	(14)	2893	(15)	2942	(16)	3207	(20)	3230	(20)
3252	(20)	3253	(20)	3260	(20)	3262	(20)	3267	(20)	3305	(20)	3307	(20)	3313	(20)
3314	(20)	3323	(20)	3354	(20)	3358	(20)	3369	(20)	3403	(20)	3521	(22)	3606	(23)
3627	(23)	3637	(23)	3650	(23)	3661	(23)	3679	(23)	3680	(23)	3811	(24)	3812	(24)
3813	(24)	3817	(24)	3856	(24)	3862	(24)	3863	(24)	3894	(25)	4009	(28)	4011	(28)
4040	(29)	4042	(29)	4122	(32)	4129	(32)	4137	(32)	4157	(33)	4208	(34)	4209	(34)
4210	(34)	4212	(34)	4217	(34)	4218	(34)	4219	(34)	4231	(34)	4241	(34)	4251	(34)
4254	(34)	4265	(34)	4272	(34)	4275	(34)	4309	(35)	4311	(35)	4313	(35)	4334	(35)
4616	(39)	4652	(39)	4672	(39)	4685	(39)	4954	(41)	4957	(41)	4983	(41)	4987	(41)
4994	(41)	5004	(41)	5480	(47)	5481	(47)	5575	(47)	5589	(47)	5591	(47)	5601	(47)
5617	(47)	5627	(47)	5639	(47)	5645	(47)	5827	(49)	5849	(49)	5857	(49)	5930	(50)
6171	(53)	6428	(56)	6494	(56)	6503	(56)	7036	(63)	7050	(63)	7131	(64)	7297	(65)
7320	(65)	7426	(66)	7490	(66)	7491	(66)	7507	(66)	7523	(66)	7529	(66)	7561	(66)
7562	(66)	7629	(67)	7680	(67)	7681	(67)	7682	(67)	7686	(67)	7687	(67)	7693	(67)
7726	(67)	7735	(67)	7736	(67)	7737	(67)	7738	(67)	7739	(67)	7749	(67)	7750	(67)
7756	(67)	7760	(67)	7761	(67)	7771	(67)	7780	(67)	7781	(67)	7790	(67)	7791	(67)
7924	(70)	7933	(70)	8069	(71)	8247	(76)	8260	(76)	8261	(76)	8262	(76)	8267	(76)
8346	(76)	8442	(79)	8443	(79)	8452	(79)	8455	(79)	8460	(79)	8497	(80)	8557	(81)
863	(4)	8671	(83)	8676	(83)	8724	(84)	8729	(84)	8782	(85)	8783	(85)	8845	(86)
8916	(87)	8917	(87)	8930	(87)	8934	(87)	8935	(87)	8939	(87)	8945	(87)	8946	(87)
8947	(87)	8951	(87)	8952	(87)	8956	(87)	8957	(87)	8958	(87)	8963	(87)	8973	(87)

.IFF

.IRPC

8983	(87)	8993	(87)	9011	(87)	9042	(87)	9043	(87)	9049	(87)	9076	(87)	9081	(87)
9090	(87)	9091	(87)	9099	(87)	9138	(88)	9139	(88)	9140	(88)	9150	(88)	9151	(88)
9165	(88)	9224	(89)	9249	(89)	9259	(89)	9272	(89)	9287	(89)	9337	(89)	9396	(90)
9431	(90)	9519	(92)	9628	(94)	9629	(94)	9633	(94)	9652	(94)	9655	(94)	9656	(94)
9665	(94)	9706	(95)	9726	(95)	9727	(95)	9728	(95)	9835	(96)	9836	(96)	9843	(96)
9845	(96)	9864	(97)	9867	(97)	9892	(98)	9893	(98)	9900	(98)	9902	(98)		
1374	(5)	1385	(5)	1393	(5)	1526	(6)	1527	(6)	1528	(6)	1529	(6)	1530	(6)
1531	(6)	1532	(6)	1533	(6)	1534	(6)	1535	(6)	1536	(6)	1537	(6)	1538	(6)
1539	(6)	1540	(6)	2269	(7)	2270	(7)	2314	(7)	2361	(7)	2362	(7)	2378	(7)
2379	(7)	2380	(7)	2381	(7)	2568	(10)	2569	(10)	2570	(10)	2571	(10)	2572	(10)
2708	(12)	2710	(12)	2807	(14)	2893	(15)	2942	(16)	3207	(20)	3230	(20)	3252	(20)
3253	(20)	3260	(20)	3262	(20)	3267	(20)	3305	(20)	3307	(20)	3313	(20)	3314	(20)
3323	(20)	3358	(20)	3369	(20)	3403	(20)	3521	(22)	3606	(23)	3627	(23)	3637	(23)
3650	(23)	3661	(23)	3679	(23)	3680	(23)	3811	(24)	3812	(24)	3813	(24)	3817	(24)
3856	(24)	3862	(24)	3863	(24)	3894	(25)	4009	(28)	4011	(28)	4040	(29)	4042	(29)
4122	(32)	4129	(32)	4137	(32)	4157	(33)	4209	(34)	4210	(34)	4217	(34)	4218	(34)
4219	(34)	4231	(34)	4241	(34)	4251	(34)	4254	(34)	4265	(34)	4272	(34)	4309	(35)
4311	(35)	4313	(35)	4334	(35)	4616	(39)	4652	(39)	4672	(39)	4685	(39)	4954	(41)
4957	(41)	4983	(41)	4987	(41)	4994	(41)	5004	(41)	5480	(47)	5481	(47)	5575	(47)
5589	(47)	5591	(47)	5601	(47)	5617	(47)	5627	(47)	5639	(47)	5645	(47)	5827	(49)
5849	(49)	5857	(49)	6171	(53)	6428	(56)	6494	(56)	6503	(56)	7036	(63)	7050	(63)
7131	(64)	7297	(65)	7320	(65)	7426	(66)	7490	(66)	7491	(66)	7507	(66)	7523	(66)
7529	(66)	7629	(67)	7680	(67)	7681	(67)	7682	(67)	7686	(67)	7687	(67)	7693	(67)
7726	(67)	7735	(67)	7736	(67)	7737	(67)	7738	(67)	7739	(67)	7749	(67)	7750	(67)
7756	(67)	7760	(67)	7761	(67)	7771	(67)	7780	(67)	7781	(67)	7790	(67)	7791	(67)
7924	(70)	7933	(70)	8069	(71)	8247	(76)	8260	(76)	8261	(76)	8262	(76)	8267	(76)
8346	(76)	8442	(79)	8443	(79)	8452	(79)	8455	(79)	8460	(79)	8497	(80)	8557	(81)
8671	(83)	8676	(83)	8724	(84)	8729	(84)	8782	(85)	8783	(85)	8845	(86)	8916	(87)
8917	(87)	8930	(87)	8934	(87)	8935	(87)	8939	(87)	8945	(87)	8946	(87)	8947	(87)
8951	(87)	8952	(87)	8956	(87)	8957	(87)	8958	(87)	8963	(87)	8973	(87)	8983	(87)
8993	(87)	9011	(87)	9042	(87)	9043	(87)	9049	(87)	9076	(87)	9081	(87)	9090	(87)
9091	(87)	9099	(87)	9139	(88)	9140	(88)	9150	(88)	9224	(89)	9249	(89)	9259	(89)
9272	(89)	9287	(89)	9337	(89)	9396	(90)	9431	(90)	9519	(92)	9628	(94)	9629	(94)
9633	(94)	9652	(94)	9655	(94)	9656	(94)	9665	(94)	9706	(95)	9726	(95)	9727	(95)
9728	(95)	9835	(96)	9836	(96)	9843	(96)	9845	(96)	9864	(97)	9867	(97)	9892	(98)
9893	(98)	9900	(98)	9902	(98)										

.LIBRARY
.LIST

424	(2)	425	(2)	426	(2)										
1375	(5)	1384	(5)	1386	(5)	1387	(5)	1388	(5)	1395	(5)	1396	(5)	1397	(5)
1400	(5)	1402	(5)	1404	(5)	1407	(5)	1408	(5)	1429	(5)	2320	(7)	2469	(8)
2630	(11)	2695	(12)	2703	(12)	2800	(14)	2817	(14)	2820	(14)	2825	(14)	2826	(14)
2827	(14)	2835	(14)	2841	(14)	2848	(14)	3043	(18)	3055	(18)	3058	(18)	3063	(18)
3103	(19)	3114	(19)	3117	(19)	3122	(19)	3213	(20)	3216	(20)	3224	(20)	3225	(20)
3229	(20)	3231	(20)	3236	(20)	3238	(20)	3242	(20)	3247	(20)	3248	(20)	3251	(20)
3257	(20)	3263	(20)	3271	(20)	3272	(20)	3278	(20)	3282	(20)	3284	(20)	3286	(20)
3287	(20)	3291	(20)	3296	(20)	3297	(20)	3298	(20)	3308	(20)	3317	(20)	3318	(20)
3326	(20)	3327	(20)	3333	(20)	3337	(20)	3339	(20)	3341	(20)	3342	(20)	3348	(20)
3353	(20)	3359	(20)	3367	(20)	3374	(20)	3375	(20)	3380	(20)	3394	(20)	3406	(20)
3412	(20)	3413	(20)	3414	(20)	3415	(20)	3423	(20)	3425	(20)	3462	(21)	3466	(21)
3467	(21)	3473	(21)	3475	(21)	3478	(21)	3479	(21)	3481	(21)	3482	(21)	3485	(21)
3525	(22)	3605	(23)	3607	(23)	3608	(23)	3611	(23)	3612	(23)	3618	(23)	3626	(23)
3631	(23)	3633	(23)	3641	(23)	3643	(23)	3644	(23)	3647	(23)	3648	(23)	3654	(23)
3655	(23)	3683	(23)	3684	(23)	3690	(23)	3691	(23)	3693	(23)	3694	(23)	3706	(23)
3707	(23)	3709	(23)	3710	(23)	3717	(23)	3720	(23)	3722	(23)	3723	(23)	3829	(24)
3836	(24)	3967	(27)	3985	(27)	4004	(28)	4005	(28)	4006	(28)	4007	(28)	4012	(28)
4013	(28)	4014	(28)	4015	(28)	4016	(28)	4035	(29)	4036	(29)	4037	(29)	4038	(29)

4043	(29)	4044	(29)	4045	(29)	4046	(29)	4047	(29)	4125	(32)	4126	(32)	4127	(32)
4130	(32)	4131	(32)	4132	(32)	4133	(32)	4134	(32)	4155	(33)	4158	(33)	417	(2)
4243	(34)	4248	(34)	4253	(34)	4281	(34)	4312	(35)	4355	(35)	4375	(35)	4377	(35)
4445	(36)	4449	(36)	4451	(36)	4461	(36)	4462	(36)	4469	(36)	4518	(37)	4526	(37)
4568	(38)	4571	(38)	4573	(38)	4619	(39)	4621	(39)	4623	(39)	4633	(39)	4649	(39)
4653	(39)	4655	(39)	4658	(39)	4660	(39)	4666	(39)	4676	(39)	4686	(39)	4691	(39)
4692	(39)	4696	(39)	4784	(40)	4842	(40)	4845	(40)	4858	(40)	4861	(40)	4879	(40)
4882	(40)	4990	(41)	4991	(41)	4992	(41)	4995	(41)	4996	(41)	5002	(41)	5003	(41)
5005	(41)	5006	(41)	5154	(45)	5169	(45)	5172	(45)	5173	(45)	5185	(45)	5210	(45)
5211	(45)	5214	(45)	5217	(45)	5226	(45)	5246	(45)	5247	(45)	5250	(45)	5283	(45)
5284	(45)	5288	(45)	5352	(46)	5378	(46)	5379	(46)	5383	(46)	5395	(46)	5412	(46)
5415	(46)	5416	(46)	5478	(47)	5484	(47)	5489	(47)	5524	(47)	5535	(47)	5576	(47)
5585	(47)	5592	(47)	5602	(47)	5618	(47)	5628	(47)	5800	(49)	5801	(49)	5804	(49)
5826	(49)	5833	(49)	5838	(49)	5839	(49)	5851	(49)	5854	(49)	5858	(49)	5920	(50)
5923	(50)	6175	(53)	6176	(53)	6215	(53)	6218	(53)	6268	(54)	6269	(54)	6270	(54)
6271	(54)	6279	(54)	6280	(54)	6281	(54)	6282	(54)	6285	(54)	6291	(54)	6292	(54)
6293	(54)	6295	(54)	6297	(54)	6299	(54)	6303	(54)	6304	(54)	6306	(54)	6307	(54)
6309	(54)	6319	(54)	6325	(54)	6330	(54)	6333	(54)	6344	(54)	6429	(56)	6438	(56)
6441	(56)	6442	(56)	6449	(56)	6450	(56)	6453	(56)	6454	(56)	6455	(56)	6456	(56)
6460	(56)	6464	(56)	6465	(56)	6478	(56)	6480	(56)	6493	(56)	6498	(56)	6499	(56)
6502	(56)	6507	(56)	6552	(57)	6555	(57)	6569	(57)	6577	(57)	6580	(57)	6581	(57)
6583	(57)	6655	(58)	6661	(58)	6687	(58)	6835	(61)	6838	(61)	6839	(61)	6848	(61)
6851	(61)	6855	(61)	6861	(61)	6862	(61)	6924	(62)	6927	(62)	7014	(63)	7017	(63)
7028	(63)	7031	(63)	7035	(63)	7042	(63)	7043	(63)	7049	(63)	7056	(63)	7057	(63)
7143	(64)	7147	(64)	7154	(64)	7164	(64)	7165	(64)	7171	(64)	7177	(64)	7197	(64)
7324	(65)	7743	(67)	7746	(67)	7747	(67)	7767	(67)	7768	(67)	7769	(67)	7776	(67)
7777	(67)	7778	(67)	7786	(67)	7787	(67)	7788	(67)	7861	(69)	7914	(70)	7917	(70)
7929	(70)	7930	(70)	7937	(70)	7941	(70)	7949	(70)	7957	(70)	7960	(70)	7961	(70)
7964	(70)	7977	(70)	7978	(70)	7981	(70)	7982	(70)	7985	(70)	7987	(70)	7989	(70)
7990	(70)	7992	(70)	8032	(71)	8045	(71)	8048	(71)	8050	(71)	8060	(71)	8062	(71)
8068	(71)	8076	(71)	8077	(71)	8108	(72)	8110	(72)	8143	(73)	8145	(73)	8248	(76)
8303	(76)	8308	(76)	8310	(76)	8313	(76)	8314	(76)	8316	(76)	8317	(76)	8320	(76)
8321	(76)	8322	(76)	8325	(76)	8326	(76)	8327	(76)	8334	(76)	8337	(76)	8340	(76)
8343	(76)	8390	(77)	8454	(79)	8456	(79)	8503	(80)	8505	(80)	856	(3)	8560	(81)
8565	(81)	8573	(81)	8575	(81)	863	(4)	8630	(82)	8675	(83)	8679	(83)	8682	(83)
8683	(83)	8728	(84)	8732	(84)	8735	(84)	8736	(84)	8790	(85)	8791	(85)	8805	(85)
8806	(85)	8915	(87)	8924	(87)	8927	(87)	8928	(87)	8969	(87)	8970	(87)	8971	(87)
8979	(87)	8980	(87)	8981	(87)	8989	(87)	8990	(87)	8991	(87)	8998	(87)	9002	(87)
9005	(87)	9016	(87)	9020	(87)	9022	(87)	9024	(87)	9034	(87)	9040	(87)	9041	(87)
9044	(87)	9045	(87)	9047	(87)	9048	(87)	9051	(87)	9052	(87)	9053	(87)	9056	(87)
9061	(87)	9069	(87)	9073	(87)	9074	(87)	9075	(87)	9077	(87)	9079	(87)	9084	(87)
9086	(87)	9092	(87)	9093	(87)	9095	(87)	9096	(87)	9098	(87)	9144	(88)	9145	(88)
9148	(88)	9212	(89)	9213	(89)	9215	(89)	9218	(89)	9219	(89)	9220	(89)	9221	(89)
9228	(89)	9229	(89)	9235	(89)	9252	(89)	9273	(89)	9281	(89)	9282	(89)	9283	(89)
9285	(89)	9301	(89)	9302	(89)	9303	(89)	9307	(89)	9311	(89)	9316	(89)	9319	(89)
9335	(89)	9336	(89)	9342	(89)	9345	(89)	9346	(89)	9349	(89)	9350	(89)	9392	(90)
9405	(90)	9406	(90)	9412	(90)	9413	(90)	9416	(90)	9429	(90)	9430	(90)	9440	(90)
9441	(90)	9447	(90)	9448	(90)	9449	(90)	9481	(91)	9488	(91)	9584	(93)	9625	(94)
9626	(94)	9627	(94)	9630	(94)	9631	(94)	9632	(94)	9634	(94)	9635	(94)	9641	(94)
9644	(94)	9653	(94)	9657	(94)	9658	(94)	9660	(94)	9661	(94)	9663	(94)	9664	(94)
9666	(94)	9667	(94)	9710	(95)	9715	(95)	9720	(95)	9724	(95)	9725	(95)	9732	(95)
9733	(95)	9734	(95)	9738	(95)	9739	(95)	9740	(95)	9744	(95)	9745	(95)	9746	(95)
9750	(95)	9751	(95)	9752	(95)	9753	(95)	9754	(95)	9762	(95)	9763	(95)	9767	(95)
9769	(95)	9776	(95)	9777	(95)	9778	(95)	9779	(95)	9787	(95)	9788	(95)	9802	(95)
9803	(95)	9805	(95)	9806	(95)	9808	(95)	9809	(95)	9840	(96)	9846	(96)	9863	(97)

	9865	(97)	9897	(98)	9903	(98)													
.LONG	1767	(6)	1858	(6)	1859	(6)	1860	(6)	1861	(6)	1862	(6)	1863	(6)	2007	(6)			
	2008	(6)	2009	(6)	2010	(6)	2011	(6)	2012	(6)	2013	(6)	2014	(6)	2026	(6)			
	2027	(6)	2028	(6)	2029	(6)	2037	(6)	2038	(6)	2039	(6)	2040	(6)	2041	(6)			
	2042	(6)	2043	(6)	2044	(6)	2056	(6)	2057	(6)	2058	(6)	2059	(6)	2067	(6)			
	2068	(6)	2074	(6)	2075	(6)	2076	(6)	2077	(6)	2085	(6)	2086	(6)	2087	(6)			
	2088	(6)	2096	(6)	2097	(6)	2098	(6)	2099	(6)	2100	(6)	2101	(6)	2102	(6)			
	2103	(6)	2115	(6)	2116	(6)	2122	(6)	2127	(6)	2132	(6)	2133	(6)	2134	(6)			
	2135	(6)	2143	(6)	2144	(6)	2150	(6)	2151	(6)	2152	(6)	2153	(6)	2161	(6)			
	2162	(6)	2163	(6)	2164	(6)													
.MACRO	432	(2)	464	(2)	490	(2)	518	(2)	549	(2)	580	(2)	610	(2)	656	(2)			
	707	(2)	730	(2)	754	(2)	767	(2)	786	(2)	818	(2)	843	(2)					
.MCALL	856	(3)																	
.MEXIT	1393	(5)	2269	(7)	2270	(7)	2314	(7)	2361	(7)	2362	(7)	2378	(7)	2379	(7)			
	2380	(7)	2381	(7)	2568	(10)	2569	(10)	2570	(10)	2571	(10)	2572	(10)	2708	(12)			
	2710	(12)	2807	(14)	2893	(15)	2942	(16)	3207	(20)	3230	(20)	3252	(20)	3253	(20)			
	3260	(20)	3262	(20)	3267	(20)	3305	(20)	3307	(20)	3313	(20)	3323	(20)	3358	(20)			
	3369	(20)	3403	(20)	3521	(22)	3606	(23)	3627	(23)	3637	(23)	3650	(23)	3661	(23)			
	3679	(23)	3811	(24)	3812	(24)	3813	(24)	3817	(24)	3856	(24)	3862	(24)	3863	(24)			
	4009	(28)	4011	(28)	4040	(29)	4042	(29)	4122	(32)	4129	(32)	4137	(32)	4157	(33)			
	4209	(34)	4210	(34)	4217	(34)	4218	(34)	4219	(34)	4231	(34)	4241	(34)	4251	(34)			
	4254	(34)	4265	(34)	4272	(34)	4309	(35)	4311	(35)	4313	(35)	4334	(35)	4616	(39)			
	4652	(39)	4672	(39)	4685	(39)	4954	(41)	4957	(41)	4983	(41)	4987	(41)	4994	(41)			
	5004	(41)	5480	(47)	5575	(47)	5589	(47)	5591	(47)	5601	(47)	5617	(47)	5627	(47)			
	5639	(47)	5645	(47)	5827	(49)	5849	(49)	5857	(49)	6171	(53)	6428	(56)	6494	(56)			
	6503	(56)	7036	(63)	7050	(63)	7297	(65)	7320	(65)	7426	(66)	7490	(66)	7491	(66)			
	7507	(66)	7523	(66)	7529	(66)	7629	(67)	7680	(67)	7681	(67)	7682	(67)	7686	(67)			
	7687	(67)	7693	(67)	7726	(67)	7735	(67)	7736	(67)	7737	(67)	7738	(67)	7739	(67)			
	7749	(67)	7750	(67)	7756	(67)	7760	(67)	7771	(67)	7780	(67)	7781	(67)	7790	(67)			
	7791	(67)	7924	(70)	7933	(70)	8069	(71)	8247	(76)	8260	(76)	8261	(76)	8262	(76)			
	8267	(76)	8346	(76)	8442	(79)	8443	(79)	8452	(79)	8455	(79)	8460	(79)	8497	(80)			
	8557	(81)	8671	(83)	8724	(84)	8782	(85)	8783	(85)	8845	(86)	8916	(87)	8917	(87)			
	8930	(87)	8934	(87)	8935	(87)	8939	(87)	8945	(87)	8946	(87)	8947	(87)	8951	(87)			
	8952	(87)	8956	(87)	8957	(87)	8958	(87)	8973	(87)	8983	(87)	8993	(87)	9011	(87)			
	9042	(87)	9043	(87)	9049	(87)	9076	(87)	9081	(87)	9090	(87)	9091	(87)	9099	(87)			
	9139	(88)	9140	(88)	9150	(88)	9224	(89)	9249	(89)	9272	(89)	9337	(89)	9396	(90)			
	9431	(90)	9519	(92)	9628	(94)	9629	(94)	9633	(94)	9652	(94)	9655	(94)	9656	(94)			
	9665	(94)	9706	(95)	9726	(95)	9727	(95)	9728	(95)	9835	(96)	9836	(96)	9843	(96)			
	9845	(96)	9864	(97)	9867	(97)	9892	(98)	9893	(98)	9900	(98)	9902	(98)					
.NCHR	1526	(6)	1527	(6)	1528	(6)	1529	(6)	1530	(6)	1531	(6)	1532	(6)	1533	(6)			
	1534	(6)	1535	(6)	1536	(6)	1537	(6)	1538	(6)	1539	(6)	1540	(6)					
.NLIST	1375	(5)	1384	(5)	1386	(5)	1387	(5)	1388	(5)	1395	(5)	1396	(5)	1397	(5)			
	1400	(5)	1402	(5)	1404	(5)	1407	(5)	1408	(5)	1429	(5)	2320	(7)	2469	(8)			
	2630	(11)	2695	(12)	2703	(12)	2800	(14)	2817	(14)	2820	(14)	2825	(14)	2826	(14)			
	2827	(14)	2835	(14)	2841	(14)	2848	(14)	3043	(18)	3055	(18)	3058	(18)	3063	(18)			
	3103	(19)	3114	(19)	3117	(19)	3122	(19)	3213	(20)	3216	(20)	3224	(20)	3225	(20)			
	3229	(20)	3231	(20)	3236	(20)	3238	(20)	3242	(20)	3247	(20)	3248	(20)	3251	(20)			
	3257	(20)	3263	(20)	3271	(20)	3272	(20)	3278	(20)	3282	(20)	3284	(20)	3286	(20)			
	3287	(20)	3291	(20)	3296	(20)	3297	(20)	3298	(20)	3308	(20)	3317	(20)	3318	(20)			
	3326	(20)	3327	(20)	3333	(20)	3337	(20)	3339	(20)	3341	(20)	3342	(20)	3348	(20)			
	3353	(20)	3359	(20)	3367	(20)	3374	(20)	3375	(20)	3380	(20)	3394	(20)	3406	(20)			
	3412	(20)	3413	(20)	3414	(20)	3415	(20)	3423	(20)	3425	(20)	3462	(21)	3466	(21)			
	3467	(21)	3473	(21)	3475	(21)	3478	(21)	3479	(21)	3481	(21)	3482	(21)	3485	(21)			
	3525	(22)	3605	(23)	3607	(23)	3608	(23)	3611	(23)	3612	(23)	3618	(23)	3626	(23)			
	3631	(23)	3633	(23)	3641	(23)	3643	(23)	3644	(23)	3647	(23)	3648	(23)	3654	(23)			

3655	(23)	3683	(23)	3684	(23)	3690	(23)	3691	(23)	3693	(23)	3694	(23)	3706	(23)
3707	(23)	3709	(23)	3710	(23)	3717	(23)	3720	(23)	3722	(23)	3723	(23)	3829	(24)
3836	(24)	3967	(27)	3985	(27)	4004	(28)	4005	(28)	4006	(28)	4007	(28)	4012	(28)
4013	(28)	4014	(28)	4015	(28)	4016	(28)	4035	(29)	4036	(29)	4037	(29)	4038	(29)
4043	(29)	4044	(29)	4045	(29)	4046	(29)	4047	(29)	4125	(32)	4126	(32)	4127	(32)
4130	(32)	4131	(32)	4132	(32)	4133	(32)	4134	(32)	4155	(33)	4158	(33)	418	(2)
4243	(34)	4248	(34)	4253	(34)	4281	(34)	4312	(35)	4355	(35)	4375	(35)	4377	(35)
4445	(36)	4449	(36)	4451	(36)	4461	(36)	4462	(36)	4469	(36)	4518	(37)	4526	(37)
4568	(38)	4571	(38)	4573	(38)	4619	(39)	4621	(39)	4623	(39)	4633	(39)	4649	(39)
4653	(39)	4655	(39)	4658	(39)	4660	(39)	4666	(39)	4676	(39)	4686	(39)	4691	(39)
4692	(39)	4696	(39)	4784	(40)	4842	(40)	4845	(40)	4858	(40)	4861	(40)	4879	(40)
4882	(40)	4990	(41)	4991	(41)	4992	(41)	4995	(41)	4996	(41)	5002	(41)	5003	(41)
5005	(41)	5006	(41)	5154	(45)	5169	(45)	5172	(45)	5173	(45)	5185	(45)	5210	(45)
5211	(45)	5214	(45)	5217	(45)	5226	(45)	5246	(45)	5247	(45)	5250	(45)	5283	(45)
5284	(45)	5288	(45)	5352	(46)	5378	(46)	5379	(46)	5383	(46)	5395	(46)	5412	(46)
5415	(46)	5416	(46)	5478	(47)	5484	(47)	5489	(47)	5524	(47)	5535	(47)	5576	(47)
5585	(47)	5592	(47)	5602	(47)	5618	(47)	5628	(47)	5800	(49)	5801	(49)	5804	(49)
5826	(49)	5833	(49)	5838	(49)	5839	(49)	5851	(49)	5854	(49)	5858	(49)	5920	(50)
5923	(50)	6175	(53)	6176	(53)	6215	(53)	6218	(53)	6268	(54)	6269	(54)	6270	(54)
6271	(54)	6279	(54)	6280	(54)	6281	(54)	6282	(54)	6285	(54)	6291	(54)	6292	(54)
6293	(54)	6295	(54)	6297	(54)	6299	(54)	6303	(54)	6304	(54)	6306	(54)	6307	(54)
6309	(54)	6319	(54)	6325	(54)	6330	(54)	6333	(54)	6344	(54)	6429	(56)	6438	(56)
6441	(56)	6442	(56)	6449	(56)	6450	(56)	6453	(56)	6454	(56)	6455	(56)	6456	(56)
6460	(56)	6464	(56)	6465	(56)	6478	(56)	6480	(56)	6493	(56)	6498	(56)	6499	(56)
6502	(56)	6507	(56)	6552	(57)	6555	(57)	6569	(57)	6577	(57)	6580	(57)	6581	(57)
6583	(57)	6655	(58)	6661	(58)	6687	(58)	6835	(61)	6838	(61)	6839	(61)	6848	(61)
6851	(61)	6855	(61)	6861	(61)	6862	(61)	6924	(62)	6927	(62)	7014	(63)	7017	(63)
7028	(63)	7031	(63)	7035	(63)	7042	(63)	7043	(63)	7049	(63)	7056	(63)	7057	(63)
7143	(64)	7147	(64)	7154	(64)	7164	(64)	7165	(64)	7171	(64)	7177	(64)	7197	(64)
7324	(65)	7743	(67)	7746	(67)	7747	(67)	7767	(67)	7768	(67)	7769	(67)	7776	(67)
7777	(67)	7778	(67)	7786	(67)	7787	(67)	7788	(67)	7861	(69)	7914	(70)	7917	(70)
7929	(70)	7930	(70)	7937	(70)	7941	(70)	7949	(70)	7957	(70)	7960	(70)	7961	(70)
7964	(70)	7977	(70)	7978	(70)	7981	(70)	7982	(70)	7985	(70)	7987	(70)	7989	(70)
7990	(70)	7992	(70)	8032	(71)	8045	(71)	8048	(71)	8050	(71)	8060	(71)	8062	(71)
8068	(71)	8076	(71)	8077	(71)	8108	(72)	8110	(72)	8143	(73)	8145	(73)	8248	(76)
8303	(76)	8308	(76)	8310	(76)	8313	(76)	8314	(76)	8316	(76)	8317	(76)	8320	(76)
8321	(76)	8322	(76)	8325	(76)	8326	(76)	8327	(76)	8334	(76)	8337	(76)	8340	(76)
8343	(76)	8390	(77)	8454	(79)	8456	(79)	8503	(80)	8505	(80)	856	(3)	8560	(81)
8565	(81)	8573	(81)	8575	(81)	863	(4)	8630	(82)	8675	(83)	8679	(83)	8682	(83)
8683	(83)	8728	(84)	8732	(84)	8735	(84)	8736	(84)	8790	(85)	8791	(85)	8805	(85)
8806	(85)	8915	(87)	8924	(87)	8927	(87)	8928	(87)	8969	(87)	8970	(87)	8971	(87)
8979	(87)	8980	(87)	8981	(87)	8989	(87)	8990	(87)	8991	(87)	8998	(87)	9002	(87)
9005	(87)	9016	(87)	9020	(87)	9022	(87)	9024	(87)	9034	(87)	9040	(87)	9041	(87)
9044	(87)	9045	(87)	9047	(87)	9048	(87)	9051	(87)	9052	(87)	9053	(87)	9056	(87)
9061	(87)	9069	(87)	9073	(87)	9074	(87)	9075	(87)	9077	(87)	9079	(87)	9084	(87)
9086	(87)	9092	(87)	9093	(87)	9095	(87)	9096	(87)	9098	(87)	9144	(88)	9145	(88)
9148	(88)	9212	(89)	9213	(89)	9215	(89)	9218	(89)	9219	(89)	9220	(89)	9221	(89)
9228	(89)	9229	(89)	9235	(89)	9252	(89)	9273	(89)	9281	(89)	9282	(89)	9283	(89)
9285	(89)	9301	(89)	9302	(89)	9303	(89)	9307	(89)	9311	(89)	9316	(89)	9319	(89)
9335	(89)	9336	(89)	9342	(89)	9345	(89)	9346	(89)	9349	(89)	9350	(89)	9392	(90)
9405	(90)	9406	(90)	9412	(90)	9413	(90)	9416	(90)	9429	(90)	9430	(90)	9440	(90)
9441	(90)	9447	(90)	9448	(90)	9449	(90)	9481	(91)	9488	(91)	9584	(93)	9625	(94)
9626	(94)	9627	(94)	9630	(94)	9631	(94)	9632	(94)	9634	(94)	9635	(94)	9641	(94)
9644	(94)	9653	(94)	9657	(94)	9658	(94)	9660	(94)	9661	(94)	9663	(94)	9664	(94)
9666	(94)	9667	(94)	9710	(95)	9715	(95)	9720	(95)	9724	(95)	9725	(95)	9732	(95)

.SBTTL

9733	(95)	9734	(95)	9738	(95)	9739	(95)	9740	(95)	9744	(95)	9745	(95)	9746	(95)
9750	(95)	9751	(95)	9752	(95)	9753	(95)	9754	(95)	9762	(95)	9763	(95)	9767	(95)
9769	(95)	9776	(95)	9777	(95)	9778	(95)	9779	(95)	9787	(95)	9788	(95)	9802	(95)
9803	(95)	9805	(95)	9806	(95)	9808	(95)	9809	(95)	9840	(96)	9846	(96)	9863	(97)
9865	(97)	9897	(98)	9903	(98)										
1434	(6)	2215	(7)	2402	(8)	2409	(8)	2481	(9)	2530	(10)	2576	(11)	2640	(12)
2714	(13)	2757	(14)	2853	(15)	2910	(16)	2961	(17)	3011	(18)	3072	(19)	3131	(20)
3429	(21)	3491	(22)	3540	(23)	3730	(24)	3867	(25)	3899	(26)	3922	(27)	3989	(28)
4020	(29)	4051	(30)	4084	(31)	4108	(32)	4140	(33)	4164	(34)	4165	(34)	419	(2)
4286	(35)	4397	(36)	4481	(37)	4540	(38)	4577	(39)	4708	(40)	4709	(40)	4902	(41)
5021	(42)	5064	(43)	5082	(44)	5112	(45)	5307	(46)	5436	(47)	5665	(48)	5728	(49)
5729	(49)	5863	(50)	5934	(51)	6080	(52)	6132	(53)	6224	(54)	6349	(55)	6377	(56)
6515	(57)	6589	(58)	6692	(59)	6735	(60)	6794	(61)	6871	(62)	6956	(63)	7086	(64)
7205	(65)	7342	(66)	7797	(68)	7835	(69)	7865	(70)	7998	(71)	8083	(72)	8115	(73)
8150	(74)	8177	(75)	8204	(76)	8368	(77)	8395	(78)	8430	(79)	8468	(80)	8469	(80)
8511	(81)	852	(3)	8589	(82)	863	(4)	8636	(83)	866	(5)	8689	(84)	8742	(85)
8812	(86)	8855	(87)	9106	(88)	9171	(89)	9361	(90)	9462	(91)	9493	(92)	9551	(93)
9603	(94)	9671	(95)	9813	(96)	9849	(97)	9870	(98)						

.TITLE
.WORD

2	(1)														
1373	(5)	1376	(5)	1377	(5)	1389	(5)	1403	(5)	1410	(5)	1415	(5)	1416	(5)
1417	(5)	1419	(5)	1421	(5)	1424	(5)	1425	(5)	1430	(5)	1431	(5)	1459	(6)
1464	(6)	1470	(6)	1471	(6)	1472	(6)	1473	(6)	1474	(6)	1475	(6)	1476	(6)
1477	(6)	1478	(6)	1479	(6)	1480	(6)	1481	(6)	1482	(6)	1503	(6)	1505	(6)
1511	(6)	1512	(6)	1513	(6)	1526	(6)	1527	(6)	1528	(6)	1529	(6)	1530	(6)
1531	(6)	1532	(6)	1533	(6)	1534	(6)	1535	(6)	153	(6)	1537	(6)	1538	(6)
1539	(6)	1540	(6)	1552	(6)	1571	(6)	1579	(6)	158	(6)	1606	(6)	1607	(6)
1677	(6)	1696	(6)	1697	(6)	1698	(6)	1699	(6)	1700	(6)	1701	(6)	1702	(6)
1703	(6)	1704	(6)	1705	(6)	1711	(6)	1781	(6)	1796	(6)	1798	(6)	1800	(6)
1801	(6)	1804	(6)	1805	(6)	1806	(6)	1807	(6)	1808	(6)	1809	(6)	1810	(6)
1811	(6)	1812	(6)	1813	(6)	1814	(6)	1815	(6)	1816	(6)	1817	(6)	1818	(6)
1819	(6)	1820	(6)	1821	(6)	1822	(6)	1823	(6)	1824	(6)	1825	(6)	1826	(6)
1827	(6)	1828	(6)	1829	(6)	1830	(6)	1831	(6)	1832	(6)	1837	(6)	1838	(6)
1839	(6)	1840	(6)	1841	(6)	1843	(6)	1851	(6)	1852	(6)	1875	(6)	1876	(6)
1878	(6)	1889	(6)	1890	(6)	1990	(6)	1991	(6)	1992	(6)	1993	(6)	1994	(6)
1995	(6)	1996	(6)	1997	(6)	1998	(6)	1999	(6)	2000	(6)	2001	(6)	2002	(6)
2003	(6)	2004	(6)	2264	(7)	2266	(7)	2267	(7)	2269	(7)	2270	(7)	2272	(7)
2276	(7)	2279	(7)	2282	(7)	2284	(7)	2290	(7)	2296	(7)	2297	(7)	2301	(7)
2302	(7)	2305	(7)	2310	(7)	2317	(7)	2319	(7)	2321	(7)	2325	(7)	2329	(7)
2330	(7)	2331	(7)	2332	(7)	2333	(7)	2338	(7)	2340	(7)	2342	(7)	2343	(7)
2344	(7)	2345	(7)	2349	(7)	2351	(7)	2357	(7)	2361	(7)	2362	(7)	2363	(7)
2368	(7)	2369	(7)	2370	(7)	2374	(7)	2378	(7)	2379	(7)	2380	(7)	2381	(7)
2382	(7)	2384	(7)	2386	(7)	2388	(7)	2389	(7)	2390	(7)	2391	(7)	2392	(7)
2397	(7)	2398	(7)	2399	(7)	2463	(8)	2464	(8)	2465	(8)	2466	(8)	2470	(8)
2471	(8)	2472	(8)	2473	(8)	2474	(8)	2475	(8)	2476	(8)	2477	(8)	2518	(9)
2519	(9)	2520	(9)	2522	(9)	2523	(9)	2524	(9)	2525	(9)	2568	(10)	2569	(10)
2570	(10)	2571	(10)	2572	(10)	2627	(11)	2629	(11)	2631	(11)	2634	(11)	2635	(11)
2636	(11)	2697	(12)	2699	(12)	2702	(12)	2705	(12)	2707	(12)	2708	(12)	2709	(12)
2710	(12)	2745	(13)	2747	(13)	2748	(13)	2749	(13)	2751	(13)	2753	(13)	2799	(14)
2801	(14)	2802	(14)	2803	(14)	2804	(14)	2806	(14)	2837	(14)	2843	(14)	2847	(14)
2849	(14)	2887	(15)	2889	(15)	2891	(15)	2895	(15)	2901	(15)	2902	(15)	2906	(15)
2937	(16)	2939	(16)	2940	(16)	2944	(16)	2952	(16)	2953	(16)	2957	(16)	2997	(17)
2998	(17)	2999	(17)	3001	(17)	3002	(17)	3003	(17)	3005	(17)	3006	(17)	3041	(18)
3044	(18)	3045	(18)	3046	(18)	3047	(18)	3048	(18)	3051	(18)	3053	(18)	3054	(18)
3056	(18)	3057	(18)	3062	(18)	3065	(18)	3068	(18)	3101	(19)	3104	(19)	3105	(19)
3106	(19)	3107	(19)	3108	(19)	3110	(19)	3112	(19)	3113	(19)	3115	(19)	3116	(19)

3121	(19)	3124	(19)	3127	(19)	3207	(20)	3215	(20)	3219	(20)	3226	(20)	3227	(20)
3230	(20)	3237	(20)	3245	(20)	3250	(20)	3258	(20)	3260	(20)	3262	(20)	3274	(20)
3278	(20)	3279	(20)	3281	(20)	3285	(20)	3294	(20)	3299	(20)	3305	(20)	3307	(20)
3308	(20)	3313	(20)	3315	(20)	3316	(20)	3317	(20)	3329	(20)	3333	(20)	3340	(20)
3352	(20)	3354	(20)	3360	(20)	3368	(20)	3369	(20)	3371	(20)	3376	(20)	3377	(20)
3379	(20)	3381	(20)	3383	(20)	3385	(20)	3387	(20)	3391	(20)	3392	(20)	3396	(20)
3402	(20)	3403	(20)	3405	(20)	3416	(20)	3424	(20)	3426	(20)	3463	(21)	3465	(21)
3469	(21)	3474	(21)	3480	(21)	3484	(21)	3522	(22)	3527	(22)	3529	(22)	3531	(22)
3535	(22)	3536	(22)	3604	(23)	3606	(23)	3609	(23)	3613	(23)	3636	(23)	3637	(23)
3646	(23)	3649	(23)	3650	(23)	3657	(23)	3661	(23)	3665	(23)	3667	(23)	3673	(23)
3677	(23)	3682	(23)	3687	(23)	3695	(23)	3701	(23)	3702	(23)	3724	(23)	3779	(24)
3781	(24)	3785	(24)	3787	(24)	3791	(24)	3792	(24)	3797	(24)	3798	(24)	3800	(24)
3803	(24)	3804	(24)	3806	(24)	3807	(24)	3808	(24)	3810	(24)	3811	(24)	3812	(24)
3813	(24)	3814	(24)	3815	(24)	3816	(24)	3817	(24)	3821	(24)	3826	(24)	3828	(24)
3830	(24)	3835	(24)	3837	(24)	3841	(24)	3843	(24)	3845	(24)	3850	(24)	3852	(24)
3856	(24)	3858	(24)	3862	(24)	3863	(24)	3864	(24)	3890	(25)	3893	(25)	3895	(25)
3915	(26)	3916	(26)	3917	(26)	3918	(26)	3953	(27)	3955	(27)	3959	(27)	3961	(27)
3962	(27)	3970	(27)	3971	(27)	3975	(27)	3978	(27)	3979	(27)	3983	(27)	3984	(27)
4009	(28)	4011	(28)	4012	(28)	4040	(29)	4042	(29)	4043	(29)	4068	(30)	4069	(30)
4073	(30)	4074	(30)	4075	(30)	4078	(30)	4080	(30)	4094	(31)	4095	(31)	4098	(31)
4099	(31)	4100	(31)	4103	(31)	4104	(31)	4120	(32)	4124	(32)	4128	(32)	4129	(32)
4136	(32)	4137	(32)	4157	(33)	4161	(33)	4201	(34)	4204	(34)	4206	(34)	4207	(34)
4208	(34)	4209	(34)	4210	(34)	4212	(34)	4213	(34)	4215	(34)	4216	(34)	4217	(34)
4218	(34)	4219	(34)	4220	(34)	4226	(34)	4228	(34)	4229	(34)	4230	(34)	4231	(34)
4232	(34)	4237	(34)	4259	(34)	4268	(34)	4271	(34)	4275	(34)	4280	(34)	4282	(34)
4310	(35)	4317	(35)	4318	(35)	4322	(35)	4326	(35)	4327	(35)	4329	(35)	4330	(35)
4335	(35)	4336	(35)	4340	(35)	4346	(35)	4347	(35)	4348	(35)	4351	(35)	4358	(35)
4359	(35)	4360	(35)	4362	(35)	4363	(35)	4364	(35)	4368	(35)	4369	(35)	4376	(35)
4378	(35)	4380	(35)	4434	(36)	4436	(36)	4448	(36)	4450	(36)	4452	(36)	4460	(36)
4463	(36)	4516	(37)	4519	(37)	4525	(37)	4527	(37)	4532	(37)	4569	(38)	4572	(38)
4630	(39)	4632	(39)	4635	(39)	4637	(39)	4642	(39)	4644	(39)	4662	(39)	4668	(39)
4679	(39)	4687	(39)	4694	(39)	4695	(39)	4753	(40)	4757	(40)	4758	(40)	4762	(40)
4763	(40)	4768	(40)	4770	(40)	4773	(40)	4774	(40)	4786	(40)	4791	(40)	4793	(40)
4795	(40)	4797	(40)	4799	(40)	4801	(40)	4805	(40)	4807	(40)	4808	(40)	4813	(40)
4815	(40)	4816	(40)	4821	(40)	4823	(40)	4824	(40)	4829	(40)	4831	(40)	4832	(40)
4840	(40)	4844	(40)	4850	(40)	4854	(40)	4860	(40)	4870	(40)	4872	(40)	4873	(40)
4881	(40)	4883	(40)	4894	(40)	4896	(40)	4950	(41)	4954	(41)	4955	(41)	4956	(41)
4957	(41)	4958	(41)	4960	(41)	4961	(41)	4966	(41)	4970	(41)	4972	(41)	4974	(41)
4975	(41)	4979	(41)	4981	(41)	4982	(41)	4983	(41)	4993	(41)	4994	(41)	4998	(41)
5001	(41)	5004	(41)	5007	(41)	5056	(42)	5058	(42)	5079	(43)	5098	(44)	5101	(44)
5102	(44)	5104	(44)	5105	(44)	5106	(44)	5156	(45)	5158	(45)	5160	(45)	5162	(45)
5164	(45)	5171	(45)	5175	(45)	5176	(45)	5177	(45)	5178	(45)	5187	(45)	5188	(45)
5189	(45)	5190	(45)	5192	(45)	5198	(45)	5203	(45)	5204	(45)	5207	(45)	5209	(45)
5213	(45)	5216	(45)	5218	(45)	5228	(45)	5231	(45)	5234	(45)	5237	(45)	5242	(45)
5245	(45)	5249	(45)	5252	(45)	5253	(45)	5257	(45)	5258	(45)	5264	(45)	5266	(45)
5270	(45)	5272	(45)	5273	(45)	5279	(45)	5286	(45)	5287	(45)	5290	(45)	5291	(45)
5354	(46)	5356	(46)	5358	(46)	5360	(46)	5365	(46)	5367	(46)	5368	(46)	5370	(46)
5371	(46)	5377	(46)	5381	(46)	5382	(46)	5385	(46)	5386	(46)	5387	(46)	5397	(46)
5398	(46)	5399	(46)	5403	(46)	5405	(46)	5406	(46)	5414	(46)	5418	(46)	5419	(46)
5420	(46)	5480	(47)	5482	(47)	5483	(47)	5491	(47)	5493	(47)	5495	(47)	5497	(47)
5499	(47)	5501	(47)	5503	(47)	5505	(47)	5507	(47)	5509	(47)	5511	(47)	5513	(47)
5515	(47)	5516	(47)	5523	(47)	5526	(47)	5527	(47)	5534	(47)	5538	(47)	5539	(47)
5545	(47)	5551	(47)	5557	(47)	5563	(47)	5569	(47)	5575	(47)	5576	(47)	5577	(47)
5584	(47)	5587	(47)	5589	(47)	5591	(47)	5592	(47)	5593	(47)	5595	(47)	5601	(47)
5602	(47)	5603	(47)	5605	(47)	5606	(47)	5608	(47)	5609	(47)	5611	(47)	5617	(47)

5618	(47)	5619	(47)	5621	(47)	5627	(47)	5628	(47)	5629	(47)	5638	(47)	5640	(47)
5650	(47)	5651	(47)	5705	(48)	5707	(48)	5712	(48)	5784	(49)	5786	(49)	5790	(49)
5792	(49)	5797	(49)	5799	(49)	5803	(49)	5806	(49)	5808	(49)	5812	(49)	5813	(49)
5816	(49)	5817	(49)	5818	(49)	5819	(49)	5823	(49)	5824	(49)	5830	(49)	5835	(49)
5840	(49)	5844	(49)	5905	(50)	5906	(50)	5910	(50)	5911	(50)	5918	(50)	5919	(50)
5922	(50)	5927	(50)	5929	(50)	5930	(50)	5996	(51)	5998	(51)	6003	(51)	6007	(51)
6011	(51)	6013	(51)	6017	(51)	6019	(51)	6020	(51)	6022	(51)	6024	(51)	6025	(51)
6026	(51)	6029	(51)	6030	(51)	6034	(51)	6036	(51)	6040	(51)	6042	(51)	6044	(51)
6048	(51)	6050	(51)	6051	(51)	6052	(51)	6053	(51)	6056	(51)	6057	(51)	6061	(51)
6064	(51)	6068	(51)	6069	(51)	6070	(51)	6071	(51)	6074	(51)	6075	(51)	6076	(51)
6111	(52)	6116	(52)	6118	(52)	6120	(52)	6122	(52)	6123	(52)	6124	(52)	6128	(52)
6169	(53)	6170	(53)	6178	(53)	6182	(53)	6184	(53)	6186	(53)	6190	(53)	6192	(53)
6194	(53)	6195	(53)	6201	(53)	6206	(53)	6210	(53)	6211	(53)	6216	(53)	6219	(53)
6263	(54)	6265	(54)	6266	(54)	6267	(54)	6277	(54)	6298	(54)	6315	(54)	6320	(54)
6321	(54)	6326	(54)	6332	(54)	6335	(54)	6340	(54)	6341	(54)	6372	(55)	6373	(55)
6421	(56)	6422	(56)	6427	(56)	6434	(56)	6436	(56)	6440	(56)	6452	(56)	6461	(56)
6463	(56)	6470	(56)	6472	(56)	6482	(56)	6483	(56)	6501	(56)	6508	(56)	6544	(57)
6545	(57)	6546	(57)	6547	(57)	6551	(57)	6554	(57)	6562	(57)	6564	(57)	6573	(57)
6574	(57)	6579	(57)	6585	(57)	6624	(58)	6627	(58)	6629	(58)	6631	(58)	6633	(58)
6635	(58)	6641	(58)	6642	(58)	6648	(58)	6649	(58)	6653	(58)	6654	(58)	6656	(58)
6657	(58)	6664	(58)	6667	(58)	6671	(58)	6673	(58)	6674	(58)	6678	(58)	6680	(58)
6684	(58)	6686	(58)	6688	(58)	6720	(59)	6724	(59)	6726	(59)	6730	(59)	6731	(59)
6771	(60)	6773	(60)	6774	(60)	6775	(60)	6779	(60)	6781	(60)	6782	(60)	6784	(60)
6785	(60)	6789	(60)	6790	(60)	6824	(61)	6825	(61)	6829	(61)	6830	(61)	6834	(61)
6837	(61)	6843	(61)	6844	(61)	6845	(61)	6846	(61)	6847	(61)	6850	(61)	6856	(61)
6864	(61)	6909	(62)	6910	(62)	6911	(62)	6915	(62)	6916	(62)	6918	(62)	6920	(62)
6921	(62)	6922	(62)	6923	(62)	6926	(62)	6928	(62)	6929	(62)	6933	(62)	6934	(62)
6936	(62)	6937	(62)	6938	(62)	6939	(62)	6943	(62)	6948	(62)	6951	(62)	6952	(62)
6999	(63)	7000	(63)	7001	(63)	7005	(63)	7006	(63)	7008	(63)	7010	(63)	7011	(63)
7012	(63)	7013	(63)	7016	(63)	7018	(63)	7019	(63)	7023	(63)	7024	(63)	7025	(63)
7026	(63)	7027	(63)	7030	(63)	7037	(63)	7045	(63)	7051	(63)	7059	(63)	7063	(63)
7064	(63)	7066	(63)	7067	(63)	7068	(63)	7069	(63)	7073	(63)	7078	(63)	7081	(63)
7082	(63)	7125	(64)	7127	(64)	7132	(64)	7133	(64)	7134	(64)	7135	(64)	7139	(64)
7140	(64)	7142	(64)	7148	(64)	7149	(64)	7151	(64)	7153	(64)	7155	(64)	7161	(64)
7166	(64)	7175	(64)	7176	(64)	7179	(64)	7183	(64)	7186	(64)	7187	(64)	7192	(64)
7196	(64)	7198	(64)	7201	(64)	7273	(65)	7275	(65)	7279	(65)	7281	(65)	7285	(65)
7287	(65)	7291	(65)	7293	(65)	7298	(65)	7300	(65)	7304	(65)	7306	(65)	7310	(65)
7313	(65)	7316	(65)	7321	(65)	7323	(65)	7325	(65)	7327	(65)	7331	(65)	7332	(65)
7333	(65)	7335	(65)	7424	(66)	7425	(66)	7427	(66)	7428	(66)	7431	(66)	7432	(66)
7434	(66)	7438	(66)	7440	(66)	7444	(66)	7446	(66)	7447	(66)	7450	(66)	7453	(66)
7454	(66)	7456	(66)	7462	(66)	7463	(66)	7465	(66)	7466	(66)	7467	(66)	7473	(66)
7474	(66)	7476	(66)	7477	(66)	7478	(66)	7480	(66)	7484	(66)	7486	(66)	7490	(66)
7491	(66)	7492	(66)	7496	(66)	7498	(66)	7500	(66)	7501	(66)	7503	(66)	7506	(66)
7507	(66)	7511	(66)	7513	(66)	7514	(66)	7517	(66)	7519	(66)	7523	(66)	7524	(66)
7528	(66)	7529	(66)	7530	(66)	7535	(66)	7537	(66)	7538	(66)	7541	(66)	7542	(66)
7544	(66)	7550	(66)	7552	(66)	7554	(66)	7555	(66)	7561	(66)	7562	(66)	7563	(66)
7565	(66)	7569	(66)	7571	(66)	7572	(66)	7574	(66)	7575	(66)	7576	(66)	7577	(66)
7578	(66)	7582	(66)	7585	(66)	7586	(66)	7587	(66)	7593	(67)	7595	(67)	7596	(67)
7598	(67)	7603	(67)	7604	(67)	7606	(67)	7607	(67)	7609	(67)	7610	(67)	7614	(67)
7615	(67)	7617	(67)	7621	(67)	7626	(67)	7628	(67)	7630	(67)	7634	(67)	7635	(67)
7639	(67)	7641	(67)	7642	(67)	7643	(67)	7647	(67)	7648	(67)	7649	(67)	7651	(67)
7652	(67)	7655	(67)	7657	(67)	7661	(67)	7663	(67)	7667	(67)	7669	(67)	7673	(67)
7675	(67)	7680	(67)	7681	(67)	7682	(67)	7683	(67)	7685	(67)	7686	(67)	7687	(67)
7688	(67)	7692	(67)	7693	(67)	7694	(67)	7695	(67)	7697	(67)	7700	(67)	7701	(67)
7705	(67)	7707	(67)	7709	(67)	7710	(67)	7715	(67)	7717	(67)	7726	(67)	7727	(67)

7728	(67)	7729	(67)	7734	(67)	7735	(67)	7736	(67)	7737	(67)	7738	(67)	7739	(67)
7740	(67)	7742	(67)	7745	(67)	7748	(67)	7749	(67)	7750	(67)	7751	(67)	7755	(67)
7756	(67)	7760	(67)	7762	(67)	7763	(67)	7765	(67)	7766	(67)	7770	(67)	7771	(67)
7774	(67)	7775	(67)	7779	(67)	7780	(67)	7781	(67)	7784	(67)	7785	(67)	7789	(67)
7790	(67)	7791	(67)	7792	(67)	7793	(67)	7823	(68)	7825	(68)	7827	(68)	7828	(68)
7829	(68)	7830	(68)	7831	(68)	7858	(69)	7860	(69)	7906	(70)	7907	(70)	7908	(70)
7909	(70)	7913	(70)	7916	(70)	7921	(70)	7923	(70)	7932	(70)	7938	(70)	7940	(70)
7950	(70)	7953	(70)	7958	(70)	7959	(70)	7963	(70)	7970	(70)	7973	(70)	7979	(70)
7980	(70)	7984	(70)	7991	(70)	7993	(70)	7994	(70)	8030	(71)	8031	(71)	8033	(71)
8035	(71)	8039	(71)	8040	(71)	8041	(71)	8043	(71)	8044	(71)	8047	(71)	8051	(71)
8057	(71)	8059	(71)	8063	(71)	8106	(72)	8107	(72)	8138	(73)	8139	(73)	8140	(73)
8142	(73)	8171	(74)	8173	(74)	8198	(75)	8200	(75)	8241	(76)	8246	(76)	8251	(76)
8252	(76)	8253	(76)	8256	(76)	8260	(76)	8261	(76)	8262	(76)	8263	(76)	8270	(76)
8273	(76)	8277	(76)	8283	(76)	8288	(76)	8289	(76)	8290	(76)	8291	(76)	8292	(76)
8293	(76)	8294	(76)	8296	(76)	8299	(76)	8304	(76)	8306	(76)	8307	(76)	8311	(76)
8312	(76)	8318	(76)	8319	(76)	8324	(76)	8329	(76)	8335	(76)	8338	(76)	8341	(76)
8344	(76)	8350	(76)	8352	(76)	8356	(76)	8357	(76)	8359	(76)	8362	(76)	8363	(76)
8387	(77)	8389	(77)	8391	(77)	8416	(78)	8417	(78)	8419	(78)	8420	(78)	8421	(78)
8423	(78)	8424	(78)	8425	(78)	8426	(78)	8439	(79)	8441	(79)	8442	(79)	8444	(79)
8446	(79)	8451	(79)	8452	(79)	8453	(79)	8455	(79)	8457	(79)	8460	(79)	8462	(79)
8464	(79)	8507	(80)	8553	(81)	8578	(81)	8581	(81)	8629	(82)	8631	(82)	8669	(83)
8677	(83)	8678	(83)	8685	(83)	8722	(84)	8730	(84)	8731	(84)	8738	(84)	8786	(85)
8789	(85)	8793	(85)	8804	(85)	8808	(85)	8809	(85)	8835	(86)	8837	(86)	8839	(86)
8841	(86)	8845	(86)	8849	(86)	8851	(86)	8916	(87)	8917	(87)	8921	(87)	8923	(87)
8926	(87)	8929	(87)	8930	(87)	8934	(87)	8935	(87)	8939	(87)	8940	(87)	8942	(87)
8944	(87)	8945	(87)	8946	(87)	8947	(87)	8951	(87)	8952	(87)	8953	(87)	8955	(87)
8956	(87)	8957	(87)	8959	(87)	8964	(87)	8965	(87)	8967	(87)	8968	(87)	8972	(87)
8974	(87)	8977	(87)	8978	(87)	8982	(87)	8984	(87)	8987	(87)	8988	(87)	8992	(87)
8994	(87)	9001	(87)	9004	(87)	9006	(87)	9007	(87)	9012	(87)	9017	(87)	9019	(87)
9023	(87)	9028	(87)	9030	(87)	9035	(87)	9036	(87)	9039	(87)	9042	(87)	9043	(87)
9046	(87)	9050	(87)	9055	(87)	9064	(87)	9072	(87)	9076	(87)	9089	(87)	9090	(87)
9091	(87)	9097	(87)	9099	(87)	9138	(88)	9139	(88)	9140	(88)	9141	(88)	9143	(88)
9147	(88)	9149	(88)	9150	(88)	9151	(88)	9152	(88)	9154	(88)	9155	(88)	9157	(88)
9159	(88)	9165	(88)	9210	(89)	9211	(89)	9214	(89)	9216	(89)	9217	(89)	9223	(89)
9231	(89)	9239	(89)	9244	(89)	9246	(89)	9248	(89)	9261	(89)	9263	(89)	9274	(89)
9286	(89)	9289	(89)	9291	(89)	9293	(89)	9317	(89)	9324	(89)	9326	(89)	9328	(89)
9330	(89)	9331	(89)	9348	(89)	9352	(89)	9353	(89)	9355	(89)	9356	(89)	9401	(90)
9408	(90)	9423	(90)	9436	(90)	9443	(90)	9456	(90)	9485	(91)	9487	(91)	9523	(92)
9524	(92)	9528	(92)	9537	(92)	9538	(92)	9540	(92)	9542	(92)	9571	(93)	9573	(93)
9574	(93)	9576	(93)	9577	(93)	9579	(93)	9583	(93)	9585	(93)	9587	(93)	9588	(93)
9589	(93)	9591	(93)	9595	(93)	9598	(93)	9628	(94)	9629	(94)	9633	(94)	9638	(94)
9640	(94)	9642	(94)	9643	(94)	9654	(94)	9655	(94)	9656	(94)	9662	(94)	9665	(94)
9770	(95)	9772	(95)	9781	(95)	9804	(95)	9810	(95)	9834	(96)	9841	(96)	9843	(96)
9845	(96)	9846	(96)	9864	(97)	9866	(97)	9867	(97)	9891	(98)	9898	(98)	9900	(98)
9902	(98)	9903	(98)												

 ! Performance indicators !

Phase	Page faults	CPU Time	Elapsed Time
-----	-----	-----	-----
Initialization	38	00:00:00.22	00:00:03.36
Command processing	866	00:00:00.83	00:00:02.37
Pass 1	4745	00:08:31.90	00:13:44.38

Symbol table sort	10	00:00:01.15	00:00:01.79
Pass 2	1311	00:00:29.52	00:00:46.90
Symbol table output	1	00:00:00.37	00:00:00.43
Psect synopsis output	3	00:00:00.02	00:00:00.06
Cross-reference output	1553	00:01:27.47	00:01:48.91
Assembler run totals	8529	00:10:31.48	00:16:28.23

The working set limit was 2048 pages.

688857 bytes (1346 pages) of virtual memory were used to buffer the intermediate code.

There were 40 pages of symbol table space allocated to hold 636 non-local and 245 local symbols.

9914 source lines were read in Pass 1, producing 87 object records in Pass 2.

135 pages of virtual memory were used to define 99 macros.

 ! Macro library statistics !

Macro library name

Macros defined

DISK\$UCODPACK00:[VAX750]RDMMAC.MLB;10	6
DISK\$UCODPACK00:[VAX750]COMETMAC.MLB;1	1
DISK\$UCODPACK00:[VAX750]8085MAC.MLB;32	77
SYS\$COMMON:[SYSLIB]STARLET.MLB;2	0
TOTALS (all libraries)	84

1105 GETS were required to define 84 macros.

There were no errors, warnings or information messages.

MAC/LIS=ECKAA.LIS/OBJ=ECKAA.OBJ/CROSS_REFERENCE=(ALL) ECKAA