VAX 8600 CONSOLE SPECIFICATIONS

The following are trademarks of Digital Equipment Corporation:

ALL-IN-1	EDGRIN	PRO/RMS
BAL-8	EduSystem	PROSE
CDP	FLIP CHIP	QUICKPOINT
COMPUTER LAB	FOCAL	RAD-8
COMSYST	GLC-8	Rainbow
COMTEX	IAS	RSTS
CTBUS	IDAC	RSX
DATATRIEVE	IDACS	RT-11
DDT	INDAC	RTM
DEC	KA10	SABR
DECCOM	KI10	Tool Kit
DECmail	LAB-K	TYPESET-8
DECmate	MASSBUS	TYPESET-10
DECnet	OMNIBUS	TYPESET-11
DECsystem-10	OS 8	UNIBUS
DECSYSTEM-20	PDP	VAX
DECtape	PDT	VMS
DECUS	PHA	VT
DECWORD	PS 8	Work Processor
DECwriter	P/OS	digital
DIBOL	Professional	
DIGITAL	PRO/BASIC	
DNC	PRO/FMS	

CONTENTS

CHAPTER 1

INTRODUCTION AND OVERVIEW

This document describes the design and operation of the 8600 console software and is intended to be used, in part, as training material for internal and customer use. The chapter describing console I/O mode operation can also be used as a command set reference.

The console software consists of three parts: the console PROM code, a set of console subsystem diagnostics, and a console program (ED0AA.SAV). They are designed to work together to provide support for running and troubleshooting the VAX 8600 CPU and console.

Note that console software V9.0 and above support the VAX 8650 CPU as well as the 8600. Throughout the remainder of this document, 8600 implies 8600 and/or 8650.

The term 'console' is used often in this document and refers to the console program which is booted from the console disk (ED0AA.SAV). The console program is the main portion of the console software package and provides a variety of services to the operator and VAX operating system.

Other documents referenced to by this one include DEC STD 032 (chapter 11), the EMM INTERFACE SPECIFICATION, the CSM FUNCTIONAL SPECIFICATION, the DSM FUNCTIONAL SPECIFICATION, the RCP DESIGN SPECIFICATION, the XXNET PROTOCOL SPECIFICATION and the 8600 REGISTER SPECIFICATION. Note that some of these documents may be for internal use only.

1.1 GLOSSARY OF TERMS

Terms used throughout this document are define here.

o CBUS -- The CPU Bus is a 32x8-bit dual access RAM used for CPU/console communication. The CBUS RAM is the means by which the RX, TX, STX, and TODR register functions are provided to the CPU.

o   CCS -- Console Command Strings are diagnostic commands issued
    from the VAX processor, under program control, to the console
    program. The console interprets the string as a series of
    console commands to be executed.

o   CIO mode -- Console I/O mode. One of two modes that the
    console program can be running in (see also PIO mode). In
    CIO mode the console program accepts command input from the
    CTY and RTY ports.

o   CPU -- In this document, CPU refers to the VAX Central
    processor running Macro-instructions.

o   CSM -- Console Support Microcode runs in the Ebox while the
    console program is in CIO mode (MACRO context) and provides
    the console program with access to various address spaces in
    the CPU.

o   CTY PORT -- This is a PCI port used for console
    communications with the local terminal device.

o   DSM -- Diagnostic Support Microcode runs in the Ebox while
    the console program is in CIO mode (Diagnostic context) and
    provides the console with access to various address spaces in
    the CPU.

o   EMM -- Environmental Monitoring Module. This
    microprocessor-controlled unit is part of the power system
    and communicates to the console program to provide control
    and status of the power system.

o   EMM PORT -- This is a PCI port used for EMM/console
    communication.

o   KAF -- Refers to a CPU KEEP ALIVE FAILURE. This is a
    condition detected in PIO mode where the CPU stops executing
    instructions. In most cases the only recourse for the
    console program is to reinitialize and reboot the CPU. A
    restart of the CPU is attempted if the KAF was one of those
    defined in DEC STD 032 as CPU ERROR HALTS.

o   KEYWORD -- An argument on a console command line (that is not
    a switch).

o   LOGICAL REGISTER -- A set of visibility bits grouped together
    to form a register. The EXAM/SDB command is one way of
    displaying the state of a logical register.

o   MPS -- Modular Power System.

o   PAMM -- The Physical Address Memory Map is a RAM in the Mbox
    that is initialized by the INIT/PAMM command. The PAMM
    assigns memory arrays and I/O adapters to specific ranges of
    physical memory addresses.

o PCI -- Programmable Communications Interface (generic term
   for USART).

o PIO mode -- Program I/O mode. One of two modes that the
   console program can be running in (see also CIO mode). In
   PIO mode the console is a dedicated I/O device to the CPU.

o QBA -- The Qbus Adapter interfaces the 16-bit Qbus with the
   8-bit console memory bus, and also performs the necessary
   handshaking with the Qbus device (RLV12 controller) for DMA
   and register transfers.

o RCP -- Remote Console Protocol. This is the name of the
   remote protocol used by the console to communicate with
   remote host systems.

o REGISTER ID --   A   unique   16-bit   number (with the
   most-significant bit set) assigned to a LOGICAL REGISTER.
   May be used as an argument to the EXAMINE/SDB command, but
   mainly intended as an "tag" used internally by the console
   program.

o RTY PORT -- This is the PCI port used for console
   communications with the remote (RD) terminal device.

o SCP -- The System Control Panel contains the 4 state LEDs and
   2 rotary switches.

o SDB -- The Serial Diagnostic Bus provides the console with
   visibility into the CPU and also is used for loading control
   stores and state information into the CPU.

o SDB ID -- A unique 16-bit number (with the most-significant
   bit clear) assigned by the CHASER utility to single
   visibility signal.

o SNAPSHOT -- The process of the console collecting data from
   the CPU after a KAF and saving it on the console's disk.

o TODR -- The Time Of Day Register is a 32-bit integer in the
   CBUS that the console increments once each 10 milliseconds to
   provide the CPU with Time-of- Year information.

o TOY -- Time of Year, in terms of 10 millisecond increments
   from the beginning of the year.

o TRANSPARENT MODE --   Term   used   to   describe   normal,
   non-protocol, remote terminal activity.

o UPC -- Short for micro PC.

o VISIBILITY REGISTER -- See LOGICAL REGISTER.

o VTERM -- Visibility Terminator. Special logic device used to
   terminate up to 8 ECL signals and provide visibility to each
   individual bit.

1.2 HARDWARE OVERVIEW
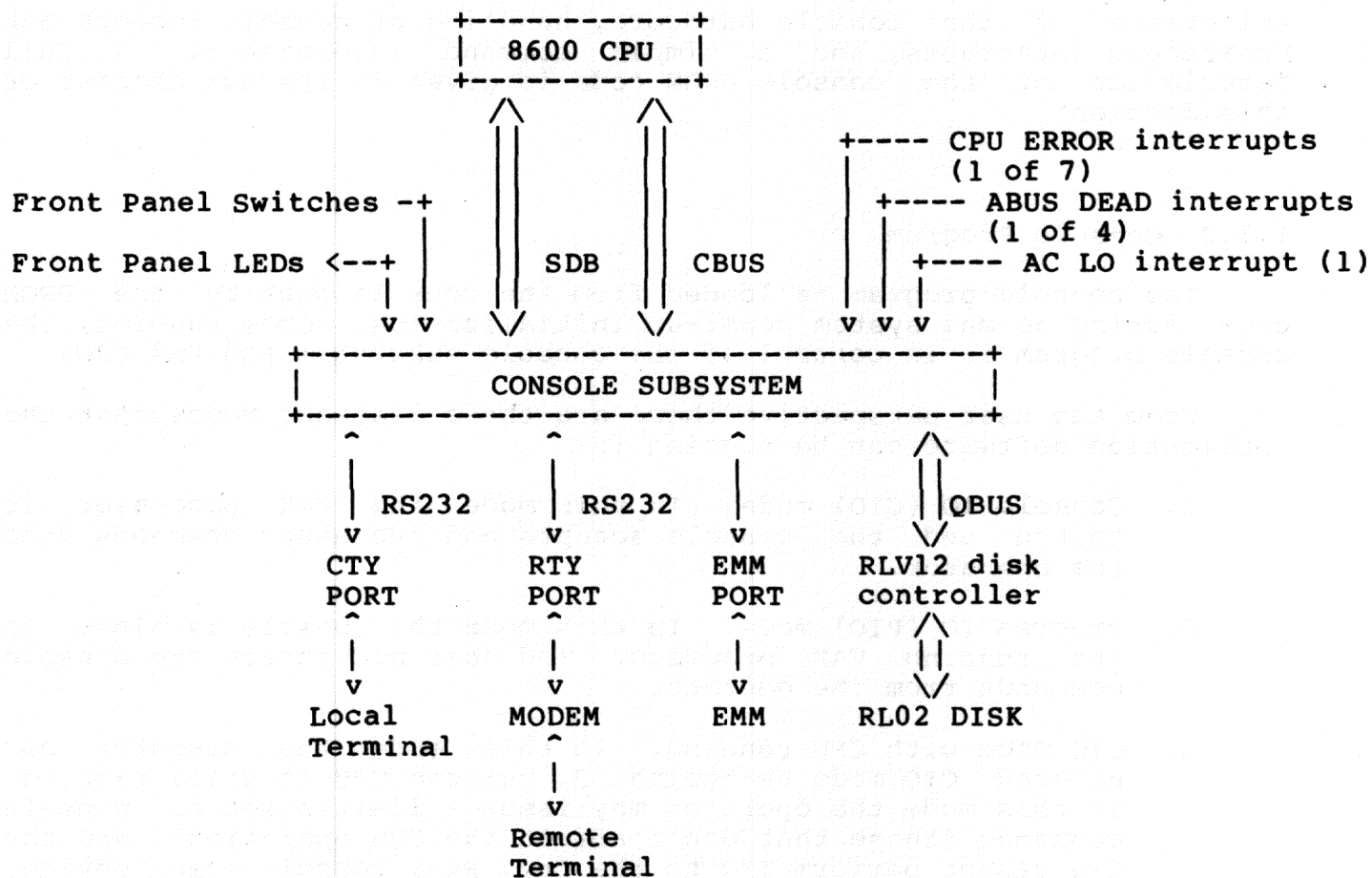
The hardware environment is discussed in detail in the
specification for the console hardware subsystem.   The following
points highlight the hardware characteristics of main interest.

o DEC's own T-11 chip is used as the microprocessor for the
   console subsystem, supporting the LSI-11 instruction set less
   MUL, DIV, SXT, and MARK instructions.

o The console subsystem has 256kb of on-board, parity assisted,
   dynamic RAM.   There is a custom section of mapping logic to
   facilitate virtual memory operation of the console program.

o The console subsystem has 3 on-board PCI's for handling Cty,
   Rty, and EMM communications.

o All I/O device CSR base addresses and vector addresses are
   non-standard, in terms of other PDP-11 products.

o An 8-bit by 32 location dual-ported RAM (CBUS) is used for
   console/CPU communication.

o A Serial Diagnostic Bus (SDB) provides control and visibility
   to the CPU and is used to access machine control stores.

o A single-chip timer (Am9513) provides 1ms timing for the
   console program, which in turn is used to handle the 10ms
   TODR register.

The console software has been designed to run only on the 8600
console subsystem and is not intended to be transportable to other
PDP-11 based systems.

1.2.1   Console Subsystem Illustration

        The following diagram shows the relationship between the console
subsystem and the rest of the 8600 system.

```
                        +------------+
                        | 8600 CPU  |
                        +------------+

                                              +---- CPU ERROR interrupts
                                              |        (1 of 7)
Front Panel Switches -+                       | +---- ABUS DEAD interrupts
                                              | |      (1 of 4)
Front Panel LEDs <--+ |       SDB     CBUS    | | +---- AC LO interrupt (1)
                    | |                       | | |
                    v v     v         v       v v v
          +---------------------------------------------------+
          |                 CONSOLE SUBSYSTEM                 |
          +---------------------------------------------------+
               ^           ^           ^              ^
               |           |           |              ||
               | RS232     | RS232     |              ||QBUS
               v           v           v              \/
             CTY         RTY         EMM         RLV12 disk
             PORT        PORT        PORT        controller
               ^           ^           ^              /\
               |           |           |              ||
               v           v           v              \/
             Local       MODEM       EMM         RL02 DISK
             Terminal      ^
                           |
                           v
                         Remote
                         Terminal
```

1.3   CONSOLE SOFTWARE OVERVIEW

1.3.1 Console PROM

The console hardware includes an 8KB PROM which provides power up self-tests of the console hardcore, handling of console reboots and unexpected interrupts, and a simple command interpreter. A full description of the console PROM code is given in its own chapter of this document.

1.3.2 Console Program

The console program is loaded from the console pack by the PROM code during normal system power-up initialization. Once running, the console program is in control of the console subsystem and VAX CPU.

From the user perspective there are three distinct modes that the application software can be running in:

1. Console IO (CIO) mode. In this mode the VAX processor is halted and the console accepts and processes commands from the operator.

2. Program IO (PIO) mode. In this mode the console is slave to the running VAX processor and does not accept any console commands from the operator.

3. CIO mode with CPU running. In this mode the operator has entered CIO mode by typing ^P, but the CPU is still running. In this mode the operator may issue a limited set of console commands (those that don't affect the CPU operation), and the CPU cannot perform I/O to the Cty, Rty, console load device, or EMM).

While in CIO mode (with the CPU not running) there are three separate command contexts that can be run, each having a specific purpose and command set. A general command set is available under all of these CIO mode contexts, and a microcode/hardware debugger command set can be enabled under two of the three contexts.

1.3.3 RT-11 Operating System

A conditionalized version of RT-11 is used to provide the console program with the general system-software requirements. This includes a disk file structure, CTY port services, timer functions, disk I/O handling and bootstrapping.

The presence of RT-11 is invisible to the operator; its commands and utilities are not available. Any console pack file maintenance (e.g., updates) must be done from the timesharing system via the console/CPU interface (CBUS). The RT-11 software is considered a part of the console program kernel.

/A special command, PROM/RT, is provided for INTERNAL use only, and will transfer control from the console program to the RT-11 operating system./

1.3.4 Console Subsystem Diagnostics

The console diagnostic (ED0BA) is loaded from the RL02 by the console PROM's 'T' command. Other diagnostics may be loaded using the T command with a filename argument. Instructions on how to enter the PROM command mode and run console diagnostics is given in the section describing the PROM code.

1.3.5 On-line Utilities

There are two preparation utilities responsible for console program data file generation. UCBLD creates machine loadable microcode files, and CHASER creates SDB visibility data "lookup" tables.

1.3.5.1 UCBLD Utility

UCBLD is maintained by the Large Vax Diagnostics group. It runs under VMS and is written in FORTRAN. Its purpose is to convert assembled microcode (.ULD output from the MICRO2 assembler) into a binary formatted output file (.BPN for Binary Physical Normalized). The .BPN file conserves space on the console pack and also allows more time-efficient loading of control stores.

The .BPN file consists of fixed-length records, each file having its own record size depending on the size needed for the largest record in that file. Records are padded with null-byte characters where necessary to make them all the same size.

All .BPN files begin with a header record that contains control and revision information and uses a minimum of 6 bytes. The header contains the number of data records following it. The last data record can be followed by another header record if another type of microcode data is present in the file, or by a null record which indicates the end of the file has been reached.

The format of the header record is:

byte	contents
1	letter code
2	version, minor
3	version, major
4	record size
5	record count, low byte
6	record count, high byte
x	padding if necessary

The "letter code" is the unique letter assigned to the control store and is used in determining the size and destination of the control store data.   The letter code assignments are:

Code	Description	Size
B	Fbox "A" control store	(48 bits x 512 locations)
D	Ibox decode Ram	(20 bits x 4096 locations)
E	Ebox scratch Pad Ram	(32 bits x 1024 locations)
G	Fbox decode Ram	(8 bits x 1024 locations)
F	Fbox "M" control store	(40 bits x 512 locations)
H	Access control Ram	(1 bits x 128 locations)
I	Ibox control store	(52 bits x 256 locations)
M	Mbox control store	(80 bits x 256 locations)
N	Cycle parameter Ram	(20 bits x 256 locations)
U	Ebox control store	(92 bits x 8192 locations)
X	Ebox context Ram	(4 bits x 512 locations)
Y	Ebox MCF Ram	(16 bits x 256 locations)

The second and third bytes in the header contain microcode version information divided into a major level and minor level. The major rev number is an 8-bit integer independent of the minor rev, which is also an 8-bit integer. This allows version numbers up to 255.255. This version number is extracted from the second line of the .ULD file and is forced to zero if no version number is found. The .ULD version number must be on the second line of the file and be in one of the following formats:

    ;Vmajor
    ;V.minor
    ;Vmajor.minor

Note that the following version numbers are equivalent:

    ;V1.01 =   ;V1.1
    ;V02   =   ;V2

The next header byte contains the record size and is the size in bytes of all records in the file, including the header record. Records are padded with null-byte characters where necessary.

The last two header bytes contain the record count, which is the number of data records following the header. The record count does not included the header itself.

The header record is followed by the first data record, which begins at byte offset "filestart+recordsize". The format of a data record is:

```
byte      contents
----      --------
1         load address, low byte
2         load address, high byte
3         ecc code
4         null byte
5         data, least-significant byte
6         data
.         data
.         data
.         data
n         data, most-significant byte
x         padding if necessary
```

The first two bytes of a data record contains the load address at which to load the data. This is followed by the ECC code calculated for the data field of this record and is used for single-bit parity error correction. Following the ECC byte is a null byte used to word-align the data field.

The fifth byte in the data record is the least-significant byte of the data to be deposited into the control store ram.

1.3.5.2 CHASER Utility

This utility, maintained by the CAD group, analyzes the CAD data base and produces a set of ASCII files which the console program uses to identify SDB visibility-bit addresses. At console initialization time the file CDF860(or 865).DAT is opened (on the console pack) and a list of .CDF filenames extracted from it. The .CDF files are then opened and processed, yielding a T11-memory resident set of SDB translation tables tailored to the particular system configuration at hand.

The header record is followed by the first data record, which begins at byte offset "filedata+recordsize". The format of a data record is:

byte	contents
1	load address, low byte
2	load address, high byte
3	ecc copy
4	null byte
5	data, least-significant byte
6	data
.	data
.	data
.	data
n	data, most-significant byte
n+	padding if necessary

The first two bytes of a data record contains the load address at which to load the data. This is followed by the ecc code calculated for the data field of this record and is used for single-bit parity error correction. Following the ecc byte is a null byte used to word-align the data field.

The fifth byte in the data record is the least-significant byte of the data to be deposited into the control store ram.

1.3.5.? CHBROX utility

This utility, maintained by the CAD group, analyzes the CAD data base and produces a set of ASCII files, which the console program uses to identify ROM viability-bit addresses. At console initialization time the file CONSOLE.ASCII.DAT is passed to the console pack, and a list of CDF filenames extracted from it. The .CDF files are then opened and processed, yielding a microcode resident set of ECB translation tables tailored to the particular system configuration at hand.

CHAPTER 2

STANDARDS

This chapter lists the DEC standards which apply to the console software.   Unless stated otherwise, compliance with all aspects of these standards have been met.

2.1   DEC STD 032 - "VAX ARCHITECTURE"

There are two chapters in DEC STD 032 which affect console software directly.   All areas of non-compliance must be sited since this standard deals heavily with operator and VAX processor interfaces to the console.

The first area of concern is Chapter 11, Console and Bootstrap. This defines most of the commands implemented in the CIO mode MACRO context. It defines the means by which console devices, such as the disk and terminals, should be communicated with by native mode programs.

Another area of concern is Chapter 9, privileged registers, which is relevant to the console as the names and addresses of these registers as well as whether they are read only or read/write is detailed. Also the format and nature of the RXCS, TXCS, STXCS, STXDB, TODR, and SID is affected by the design and structure of the console program.

2.1.1   Noncompliance

The major areas of non-compliance with this standard are listed below. Only MACRO context commands are included here.

- o   The EXAMINE command displays a full 32-bit number even if the /BYTE or /WORD switch is used. The unspecified portion of the data is forced to 0.

- o   The /C switch on the LOAD, EXAMINE, and DEPOSIT commands is not supported.

    o An additional switch is provided on the BOOT command
       (/NOSTART) to allow easy modification of boot parameters.

    o The character typed to exit from space-bar-step-mode is
       thrown away.

    o There is no way to set a default data type or address space
       for the EXAMINE and DEPOSIT commands. The defaults are
       always longword, physical.

    o BOOT, START, and CONTINUE do not cause a command file to
       suspend execution.

    o The %O, %X, and %D prefixes can be used to override the
       default radix.

    o There is no TEST command.

    o The START and CONTINUE commands are used to re-enter PIO mode
       after a ^P break to CIO mode, regardless of whether or not
       the CPU is already running.

2.2 DEC STD 051 - "ASCII DATA STANDARD"

The area of major concern with this standard is in the use of
control characters. Since only certain DEC products will be supported
as terminals the impact of a variant is slight.

2.3 DEC STD 052 - "MODEM SUPPORT"

This standard defines MODEM support necessary to meet
requirements of all known modem devices. All requirements of this
specification have been met.

2.4 DEC STD 172 - "LEGAL NOTICES"

This standard defines copyright and liability information to be
included in all source files for the console program.

2.5 DEC STD 112 - "FORMATTING DATE AND TIME"

This standard defines how the time and date is to be expressed in
messages and commands visible to the users. In all aspects of
customer visible functions it has been complied with fully.

2.6   DEC STD 145 - "FORMATTING DATA"

This standard defines internal data formats as well as ways of expressing floating point and integer values. Compliance with this standard makes borrowing and interfacing with existing software easier.

2.4. DOD STD 1AS - "FORMATTING DATA."

This standard defines internal data formats as well as type of expressing floating point and integer values. Compliance with this standard makes borrowing and interfacing with existing software easier.

CHAPTER 3

CONSOLE PROM OPERATION

PROM code initialization is performed whenever the machine power is turned on, or whenever a power failure recovery is being attempted. It is not performed entirely when a console reboot is requested by the CPU interrupt (TSTRT) or by the REBOOT command. Initialialization begins with the execution of several self-tests and ends with the booting of the console program from the RL.

Optionally, the user may wish to use the limited PROM command set to test the console further or perform some I/O register manipulation.

Invisible to the user is the fact that the console PROM is segmented into two 4kb sections, only one of which is enabled at a time. When a segment is enabled it occupies the T11 addresses between 164000 and 173776. This reduces the amount of console memory space needed to run the 8kb PROM to only 4kb. Switching between segments is handled mainly by the PROM code itself, with the exception of a power on condition.

3.1 POWER-UP OPERATION

On power-up the T11 issues a console INIT pulse to clear most of the console hardware registers (which disables all T11 interrupts and enables the lower segment of the PROM) and vectors to the PROM address 172000. Immediately the PROM begins a series of self tests designed to verify logic critical to operation. It is assumed, however, that the T11 chip and its instruction set is fully operational. The following subsection describes the self-tests in detail.

When a self-test fails the console loops indefinately. In some cases the front panel LEDs are used to indicate the failing test number, but mostly there will be an error message displayed on the CTY.

If none of the self-tests fail the PROM code turns off the 4 front panel LEDs and sets up all T11 interrupt vectors to point to the "unexpected interrupt handler" entry to the PROM (172010). It then transfers control to the upper 4kb segment. The upper segment begins by sending a <CR><LF><BELL> sequence to the CTY to let the operator

know that the self-tests have completed and that the console program
is about to be booted. A pause of 3-4 seconds occurs between the
ringing of the CTY bell and the beginning of the RL boot sequence,
during which any input typed on the CTY causes the PROM to enter its
command input loop.

The RL boot sequence consists of a series of RL-access tests
prior to the actual boot attempt. These tests are explained in the
section describing the B command.

3.1.1 Self-test Execution

The execution of the PROM self-tests occurs immediately after
power is supplied to the T11 processor and surrounding circuitry. It
is assumed that the T11 internal logic is operational.

Test loop controls similar to those found in diagnostic programs
are used. A test will loop on error, respond to ^S, ^Q, and ^O (if
the test outputs anything), and will exit the loop in response to ^E
(only for some tests). Furthermore, a special trigger signal is SET
immediately after detection of the error and cleared at the end of
each test loop. This serves as a trigger input for oscillascopes and
analyzers and is located as the signal CL09 DIAG TRIGGER H.

The tests are arranged in a building block fashion so that logic
is not used before it is tested. Because the CTY interface is not
known to be good the initial tests rely on the front panel LEDs to
contain test numbers. When one of the primitive tests fails the front
panel LEDs will contain the test's number. As soon as the CTY
interface is tested the PROM Vn.n banner is printed. The CTY is used
to report errors beyond that point while the LEDs remain in there last
set state, which happens to be 1001 (test 9).

Errors reported to the CTY are of the following form:

          CONSOLE SELF-TEST DETECTED ERROR: TEST #n   SUBTEST #n
          "descriptive message"
          "data header"
          "data ..."

The first couple of self-tests verify the integrity of the PROM
code, the front panel LEDs, and much of the T11 addressing and data
paths.

Upon receiveing power the front panel LEDs are forced to the ON
state. At this time the PROM code begins its first test, which is to
calculate the additive checksum of the entire 8kb PROM. If
unsuccessful the code falls into a "BR ." instruction and the LEDs
remain in the 1111 state. Note that the 1111 state will also remain
if some other critical T11 logic is broken.

Below is a list of the PROM tests executed up to the displaying
of the PROM Vn.n banner.

Test #0   PROM CHECKSUM TEST
                   LED state = 1111
Description: An additive checksum of both the lower and upper segments
                   of the PROM is calculated. The result should be 377(8).
                   On error a "BR ." is executed.

Test #1   SCP SDB CHANNEL TEST
                   LED state = 0001 -> 0010 -> 0100 -> 1000
Description: A floating 1 pattern is shifted through the LED's at a fairly
                   slow speed so the operator can tell if there is a LED
                   failure. The test checks most of the SDB control logic and
                   the SCP LED logic which is used by following tests to report
                   test numbers. The test is open ended and cannot fail.

Test #2   CTY INTERFACE MODE REGISTER 1 BIT TEST
                   LED state = 0010
Description: Alternating 01010101 and 10101010 patterns are written and
                   read from the CTY's PCI_MODE_REGISTER_1.

Test #3   CTY INTERFACE MODE REGISTER 2 BIT TEST
                   LED state = 0011
Description: Alternating 01010101 and 10101010 patterns are written and
                   read from the CTY's PCI_MODE_REGISTER_2.

Test #4   CTY INTERFACE COMMAND REGISTER BIT TEST
                   LED state = 0100
Description: Both a 01010101 and a 10101010 pattern is written and
                   read from the CTY's PCI_COMMAND_REGISTER.

Test #5   CTY INTERFACE RESET TEST
                   LED state = 0101
Description: A pattern is loaded into the PCI_COMMAND_REGISTER and a
                   PCI RESET is performed. If the command register clears then
                   the PCI's RESET input is connected and working.

Test #6   CTY INTERFACE TxRDY BIT TEST
                   LED state = 0110
Description: The local PCI is configured to run in its normal operating
                   mode (9600baud, 8-bit, noparity, 1 stop). The TxRDY bit in
                   the PCI status register is then expected to set within 100ms.
                   If ok, the PCI transmit register is loaded and the TxRDY
                   bit is expected to clear immediately.

Test #7   CTY INTERFACE RxRDY BIT TEST
                   LED state = 0111
Description: The local PCI is reinitialized but this time in local loop-
                   back mode. A character is transmitted and the RxRDY bit in
                   the PCI status register is expected to set. When the
                   RECEIVED_CHARACTER_REGISTER is read the RxRDY bit is
                   expected to clear.

Test #10   CTY INTERFACE LOOP BACK TEST

	LED state = 1000
Description:	This is essentially the same as the previous test ex-
	cept that several loopback transmissions are perform-
	ed to make certain the PCI and crystal can handle it.

Test #11   CTY BANNER TEST
                  LED state = 1001
Description: This is an open ended test that relies on the operator
                  to observe the PROM Vn.n banner.

     At this point the CTY interface, cable, and printer are assumed
to be operational. Test failures from here on are reported on the CTY
and CTY input may also be recognized where indicated.   The remaining
tests in the PROM initialization are listed below. Tests that allow
the ^E input character can be exited if they fail, otherwise the test
will loop indefinately.   The other input control characters can be
used stop (^S) or resume (^Q) test execution, or suppress (^O) error
report output.

Test #12   PARITY ERROR LATCH TEST
                  Allowable Control Characters are ^S ^Q ^O ^E.
Description: Tests that the latch responsible for indicating parity
                  errors can be set and cleared directly.

Test #13   PARITY CIRCUIT TEST, PART 1
                  Allowable Control Characters are ^S ^Q ^O ^E.
Description: Simultaneously tests that RAM location 0 will hold a
                  01010101 pattern. If ok, location 0 of the parity RAM
                  is expected to contain 1.

Test #14   PARITY CIRCUIT TEST, PART 2
                  Allowable Control Characters are ^S ^Q ^O ^E.
Description: Simultaneously tests that RAM location 0 will hold a
                  10101010 pattern. If ok, location 0 of the parity RAM
                  is expected to contain 0 (since the "force parity error"
                  bit in MCSR0 was set prior to depositing the pattern).

Test #15   58KB RAM DATA/ADDRESS TEST, BOTTOM-UP
                  Allowable Control Characters are ^S ^Q ^O ^E.
Description: A modified moving-inversions test is performed on the
                  first 58KB of physical RAM. The test verifies all data
                  stuck-at faults likely to occur in the console RAM con-
                  figuration as well as all addressing faults in the
                  positive (incrementing) direction. On error the address,
                  expected data, and received data are displayed.

Test #16   58KB RAM DATA/ADDRESS TEST, TOP-DOWN
                  Allowable Control Characters are ^S ^Q ^O ^E.
Description: A modified moving-inversions test is performed on the
                  first 58KB of physical RAM. The test verifies all data
                  stuck-at faults likely to occur in the console RAM con-
                  figuration as well as all addressing faults in the
                  negative (decrementing) direction. On error the address,
                  expected data, and received data are displayed.

Test #17

MAP RAM LOCATION 0 QV TEST
Allowable Control Characters are ^S ^Q ^O ^E.

Description:

A loop is first performed that uses the first mapping RAM
location (MAPR00) to initialize all of physical memory with
zero's and good parity, and placing each 4kb-page number
in the first location of its own 4kb page boundary. The
process is repeated to check that the first byte of each
page does contain the correct value.

Test #20

MAPPING RAM DATA TEST, BOTTOM-UP
Allowable Control Characters are ^S ^Q ^O ^E.

Description:

This test uses the memory pattern verified by the previous
test to check that all mapping RAM locations can access
pages uniquely, in the positive direction.

Test #21

MAPPING RAM DATA TEST, TOP-DOWN
Allowable Control Characters are ^S ^Q ^O ^E.

Description:

This test uses the memory pattern verified by the previous
test to check that all mapping RAM locations can access
pages uniquely, in the negative direction.

Test #22

MAPPING RAM ADDRESSING TEST
Allowable Control Characters are ^S ^Q ^O ^E.

Description:

This test uses the memory pattern verified by the previous
tests to check that all mapping RAM locations are themselves
uniquely addressable. The memory mapper is then turned off.

Test #23

TOY CHIP ACCESS TEST
Allowable Control Characters are ^S ^Q ^O ^E.

Description:

This test checks that registers in the console's TOY chip
can be accessed. This verifies that the TOY chip is
receiving power from the +5 B signal input from the BBU.

Test #24

RTY INTERFACE MODE REGISTER 1 BIT TEST
Allowable Control Characters are ^S ^Q ^O ^E.

Description:

Normal operation patterns are loaded into the RTY's PCI_
MODE_REGISTER_1 and checked.

Test #25

RTY INTERFACE MODE REGISTER 2 BIT TEST
Allowable Control Characters are ^S ^Q ^O ^E.

Description:

Normal operation patterns are loaded into the RTY's PCI_
MODE_REGISTER_2 and checked.

Test #26

RTY INTERFACE COMMAND REGISTER BIT TEST
Allowable Control Characters are ^S ^Q ^O ^E.

Description:

Both a 01000101 and a 10101010 pattern is written and
read from the RTY's PCI_COMMAND_REGISTER.

Test #27

RTY INTERFACE RESET TEST
Allowable Control Characters are ^S ^Q ^O ^E.

Description:

A pattern is loaded into the PCI_COMMAND_REGISTER and a
PCI RESET is performed. If the command register clears
then the PCI's RESET input is connected and working.

Test #30   RTY INTERFACE TxRDY BIT TEST
                    Allowable Control Characters are ^S ^Q ^O ^E.
Description: The remote PCI is configured to run in its normal oper-
                    ating mode (1200baud, 8-bit, noparity, 1 stop). The
                    TxRDY bit in the PCI status register is then expected
                    to set within 100ms. If ok, the PCI transmit register
                    is loaded and the TxRDY bit is expected to clear
                    immediately.

Test #31   RTY INTERFACE RxRDY BIT TEST
                    Allowable Control Characters are ^S ^Q ^O ^E.
Description: The remote PCI is reinitialized but this time in local
                    loop-back mode. A character is transmitted and the
                    RxRDY bit in the PCI status register is expected to set.
                    When the RECEIVED_CHARACTER_REGISTER is read the RxRDY
                    bit is expected to clear.

Test #32   RTY INTERFACE LOOP BACK TEST
                    Allowable Control Characters are ^S ^Q ^O ^E.
Description: This test duplicates the function of the previous test
                    to verify that the remote PCI can handle consecutive
                    character transmissions.

Test #33   SCP SDB LOGIC TEST
                    Allowable Control Characters are ^S ^Q ^O ^E.
Description: This test checks the continuity of the SDB control chan-
                    nel on the SCP. This test will affect the state of the
                    front panel LEDs but is done so fast it is unnoticable.

Test #34   "CL15 TSTRT" INTERRUPT TEST
                    Allowable Control Characters are ^S ^Q ^O.
Description: This test checks that the TSTRT interrupt input to the
                    console is not asserted. This interrupt is not maskable
                    through PSW so it must be cleared before the console
                    program will run.

Test #35   UNEXPECTED INTERRUPT TEST
                    Allowable Control Characters are ^S ^Q ^O.
Description: This test checks that after a full console reset there
                    are no pending (or stuck) interrupts to the T11. This
                    is to ensure the proper startup of RT and the console
                    kernel.

3.2 PROM COMMAND LOOP

    When the PROM code signals the completion of the selt-tests by
sending the bell character to the CTY the operator can type any input
and cause the PROM to enter its command null loop. While in the null
loop the PROM code may service the RTY device as well as the CTY for
possible input (provided the front panel switch is in the REMOTE
position and a remote connection has been made). The PROM code
indicates its readyness for input with the "ROM>" prompt.

The command parser in the PROM is extremly simple.   It accepts
single character commands and alphanumeric arguments. The commands
available to the user are listed below.

Command:	Description:
B	Boot the console software from the RL. This command begins by testing the interface between the console and the RL controller. The sequence of tests are:

        Check for AC LOW and DC LOW conditions in
            the BA11.
        Check that console generated BUS INIT reaches
            the RL controller and clears all
            RL controller registers.
        Check that RLCS, RLBA, and RLDA will retain
            a full compliment of patterns.
      Perform an RL MAINTENANCE TRANSFER to verify
            DMA logic between the console and
            RL, and other logic on the RLV12.
        Check that the first word read in from the RL
            boot block contains "240" (NOP).

If any of the above checks fail a message is reported
to both the CTY and RTY and the B command aborts.
There is no test looping capability. If a drive or
controller status error is detected during the actual
booting of the device a number of retries is done.

Command:	Description:
D addr data	Deposit the data word to the specified address. The address must be an even number.
E addr	Examine the data word at the specified address. Odd addresses are made even by dropping the low order bit.
S addr	Start the T11 execution at the specified address. The address must be an even number. The command-argument delimiter is optional.
T [filename]	Without any argument this command re-runs the PROM self-tests and then loads and runs the console diagnostic program (ED0BA) for 2 passes. If a filename is specified the self-tests are not run and the file is loaded and started at location 200. The default filename extension is .SAV.
Q addr data	This command allows direct deposits to Qbus registers. The addresses 174400, 174402, 174404, and 174406 are the RLCS, RLBA, RLDA, and RLMPR registers, respectively. All other addresses are rejected.
R addr	This command allows direct examines of Qbus registers. The addresses 174400, 174402, 174404, and 174406 are the RLCS, RLBA, RLDA, and RLMPR registers,

respectively. All other addresses are rejected.

V   This command causes the PROM code to enter a
                loop designed to allow characters to pass di-
                rectly between the CTY and RTY. This is com-
                monly used in manufacturing to allow the
                operator to communicate with VMS in order to
                initiate the down-line loading of test code into
                the console (using the X command). The escape
                sequence ^P^X^P can be entered from either input
                device to force the PROM code to exit this mode.

X   This command complies (less any switches) with
                the description of same in chapter 11 of DEC
                STD 032. It is intended to be used for binary
                data transfers between T11 memory and a compu-
                ter using the remote port input. It is not in-
                tended for use by humans and will work
                properly only when issued from the RTY device.

3.3 PROM RTY SUPPORT

The PROM code services the RTY port with a much simpler procedure
than the one used by the console program. This is necessary since the
PROM code has no timer or interrupt capability. No remote protocol
support is offered by the PROM, only character mode operation exists.

After the PROM code completes its power-up self-tests, which
includes loop back tests on the RTY PCI, it configures the PCI to
operate with default characteristics (same defaults used by the SET
TERMINAL command).   Once the PCI is set up, and if the front panel
terminal control switch is in the REMOTE position, the PROM asserts
the DTR signal to the MODEM, thus allowing the MODEM to turn on-line.
If the DSR and CD signals from the MODEM are sensed as TRUE then all
character output to the CTY will also be sent to the RTY port, and
input will be accepted from either. Note that because the DTR signal
to the MODEM is not asserted until the end of the self-tests it will
be virtually impossible to establish a MODEM connect to the RTY port
during system power-on intialization while he PROM is still running.
However, a terminal connected locally with a NULL MODEM cable should
not have any problem getting control of the PROM code, and it is for
this reason that this scheme becomes useful.

Once the PROM asserts DTR to the MODEM the dial-in capability is
activated.   When the console program starts running it begins to
service the RTY port immediately, thus giving the remote user control
throughout PROM-to- console program transitions, and visa-versa. This
means that if T11 control transfers to the PROM because of some
unexpected condition, such as for a T-11 HALT or unexpected interrupt,
the PROM can continue to service the remote connection.

During the handling of connections and disconnections of the
remote terminal by the PROM code the state of the front panel
indicators is updated in accordance with the definitions of those

indicators within this document.

Because the RTY interface is tested by the prom self tests and by
the console diagnostic (ED0BA) the T command cannot be used by the RTY
user. It is therefore an assumption that the console subsystem is
fully operable before a remote session can be properly handled.

3.4   SPECIAL SERVICES

The console PROM code provides two special services to the
console program.   The first is to handle unexpected interrupts, and
the second to handle console reboot requests.   These services are
described below.

3.4.1   Unexpected Interrupt Handling

Whenever the PROM code receives control of the T11, whether by
first-time power up, a console reboot, or an unexpected interrupt, it
disables T11 interrupts by raising the interrupt priority level (IPL)
in the PSW to 7 (highest) and loads all T11 interrupt vectors with a
pointer to the PROM's unexpected interrupt service routine (172010).
The "new PSW", contained in the location following each vector
location, contains a 4-bit id in the condition code field such that a
single service routine entry can be used for all interrupt vectors.

There are 16 possible vectored interrupts on the console module,
2 of which are unused and can never occur. Each vector is dedicated
for one specific device.   The following table lists the vectors,
priority levels, and id codes for all possible console interrupts.

Vector	IPL	ID	I/O Device
24	*	00	PROM RESTART (*UNMASKABLE by PSW)
60	4	01	REMOTE PCI TRANSMIT/RECEIVE/MODEM
64	4	02	LOCAL PCI TRANSMIT/RECEIVE
70	4	03	QBUS REPLY TIMEOUT
100	6	04	UNUSED
104	6	05	TOY 1MS INTERRUPT
110	6	06	CPU CONTROL STORE PARITY ERROR
114	6	07	STOR RDY
120	5	10	TXCS RDY
124	5	11	RXCS DNE
130	5	12	QBUS ADAPTER
134	5	13	EMM PCI TRANSMIT/RECEIVE
140	7	14	CONSOLE RAM PARITY ERROR
144	7	15	UNUSED
150	7	16	SYSTEM AC LOW
154	7	17	ABUS DEAD

The devices handled by vectors 24, 70, and 140 are always handled by the PROM code as unexpected interrupts.

When the PROM receives control at its unexpected interrupt service routine it decodes the four-bit ID number in the condition code field of the PSW and reports the vector through which the transfer was made. This is followed by the contents of the T11 registers (R0 through R7 and PSW) at the time of the interrupt. In the case of vector 140, console ram parity error, the PROM code constructs the address of the parity error (from the PECAS and PERAS registers) and displays this, along with the contents of the location and whether or not the parity error is hard or soft.

Because most of the interrupts on the console are level-sensitive rather than edge-sensitive there is no easy way for the PROM to dismiss the interrupt and return to the previous T11 instruction stream. Thus, the PROM code enters its command loop and waits for operator intervention, or a CPU reboot request, to restart the console program.

3.4.2 Console Reboot Handling

In addition to the 16 vectored interrupts described in the previous section there is another which forces the console to begin execution at the PROM entry responsible for handling console reboots (172004). When the T11 interrupt facility is enabled (MCSR0 bit 0 = 1) it treats this interrupt as an unmaskable edge-sensitive input which saves the PC and PSW, forces the IPL to priority 7, and vectors the T11 directly to the reboot address. This same operation occurs when the T11 executes a HALT instruction.

If the cause of the entry into the reboot/halt routine was a TSTRT interrupt, which can be generated by the CPU's MTPR CRBT instruction or the console's REBOOT command, the message "?Console Reboot Initiated" is printed and the PROM reboots the RL. Note: VMS issues a TSTRT interrupt if the console fails to update the Time-Of-Day register (TODR) for a considerable period of time (1 - 2 minutes).

If the cause of the entry was due to a T11 HALT instruction the ?T11 HLT message is printed followed by the T11 registers saved prior. The PROM code then enters its command loop and waits for input. While the PROM code waits in its command loop it is still sensitive to TSTRT interrupt requests from the CPU.

The PROM code has no responsibility for saving or restoring console state information during a reboot. This is done entirely by the console program.

CHAPTER 4

CONSOLE DIAGNOSTIC OPERATION

The VAX 8600 Console diagnostic is named ED0BA and is a stand-alone program that can be loaded directly by the PROM "T" command. Its tests begin where the PROM self-tests leave off, but has loop controls superior to those in the PROM.

The purpose of this diagnostic is to detect problems in consoles installed in either a VAX 8600 system or in one of the special console test stations used in manufacturing. The 8600 CPU must be brought down before the diagnostic can be run; it cannot be run on-line.

Note that the diagnostic cannot be run by the RTY user since it tests the RTY interface.

4.1 HOW TO RUN THE DIAGNOSTIC

The console diagnostic can be loaded directly from the PROM command prompt, using the T command, or down-line loaded from a remote host using the X command and then started with the S command. The normal start address is 200, which is also the restart address. The entry at location 204 is used by the T command as an auto-run entry where ED0BA automatically runs 2 complete passes and then returns control to the PROM code.

When the PROM "T" command is used without argument the console diagnostic is automatically loaded and started at location 204 after the PROM self-tests complete. If ED0BA detects an error or is interrupted by the operator (via ^G) then the return to the PROM is inhibited and ED0BA runs indefinately.

While ED0BA is running the operator can enter any of the following control characters at any time:

```
^S - SUSPEND PROGRAM OUTPUT, PROGRAM WAITS FOR ^Q
^Q - RESUME PROGRAM OUTPUT
^G - ENTER SWREG-CHANGE MODE
```

While in SWREG-CHANGE mode the following control characters may
be used:
>   ^U - IGNORE INPUT LINE AND PREPARE TO ACCEPT ANOTHER
>   ^C - RESTART THE PROGRAM
>   ^P - EXIT TO THE PROM COMMAND PROMPT

4.2 PROGRAM REQUIREMENTS

The diagnostic program requires that the local terminal (CTY)
interface be set to the following characteristics:

>   Transmit/Receive Baud rate = 9600/9600
>   8 data bits, no parity, 1 stop bit

The RTY interface is tested and cannot be used as a remote
terminal during execution of ED0BA.

Also required is an RL02 disk pack loaded and ready in the RL
drive.   No disk-write operations are done by the diagnostic, only
register and DMA reads are done.

4.2.1 Diagnostic SWREG Settings

The ^G character causes the diagnostic to enter SWREG-CHANGE
mode.   This begins by displaying the current setting of the SWREG, in
octal, and then prompts the operator for a new value or a <CR> to
leave it unchanged.   The list below shows the different SWREG bit
positions.

SWITCH	BIT	DESCRIPTION
15	100000	EXIT SUBTEST LOOP AND PROCEED
14	040000	LOOP ON (NEXT) FAILING SUBTEST
13	020000	INHIBIT ERROR TYPEOUTS
12	010000	ENABLE TEST-TRACE MODE
11	004000	INHIBIT INTERNAL TEST ITERATIONS
9	001000	LOOP ON TEST IN SWR<7:0>
7-0	000XXX	TEST NUMBER TO LOOP ON

4.2.2 Test Looping Characteristics

When it is desired to force a specific test into a loop the SWREG
is used. The number of the test to loop on is loaded into SWREG bit 0
thru 7 and bit 9 is set. For example, a SWREG value of 1120 will
cause ED0BA to loop on test number 120 the next time that test is
encountered. In the test loop all test intialization and subtests are
performed on each interation.   To exit the test loop bit 9 of the
SWREG must be cleared, or another test number can be placed in bits 7

thru 0.

When SWREG bit 14 is set the diagnostic will loop on the next failing subtest encountered. Only the failing subtest is executed in the loop, regardless of whether or not the subtest continues to fail. To exit this loop SWREG bit 14 can be cleared, or bit 15 can be set. Note that setting both bits 14 and 15 causes the looping test to exit but looping resumes on the next failing test encountered. Note that a subtest loop due to a detected failure has precedence over a test loop controlled by SWREG bit 9.

When a diagnostic test fails and begins looping the console signal "CL09 DIAG TRIGGER H" is fired once on each iteration of the loop. This helps to synchronize a 'scope to the test and thus isolate the failure. This is not identical to the way this signal is used by the PROM self-tests.

4.2.3 Interpreting Failure Reports

The general format of an error report generated by the diagnostic is shown below. The "expected and received data" field is included only where pertinent data is available.

    ERROR DETECTED BY TEST #xxx, SUBTEST #yyy
    "descriptive error message"
    "expected and received data"

The "descriptive error message" will, in most cases, consist of a console module signal name and the schematic page number (in parenthesis) where the error was actually perceived by the diagnostic (point of visibility). The diagnostic listing must be referred to for details of the actual failure, and from there the cause of the failure. In most cases a subtest failure involves only a few logic components and thus provides the isolation necessary for quick manufacturing repairs.

The contents of the "expected and received data" field is a function of the test being performed. Each data item displayed has its own header which indicates what the data is.

Some examples of typical error reports are:

    ERROR DETECTED BY TEST #4, SUBTEST #1
    "CL18 STOP CLOCK" FAILURE (CL18)

    ERROR DETECTED BY TEST #40, SUBTEST #2
    MCSR0 REGISTER BIT FAILURE (CL08)
    EXPD   RCVD
    000242 000252

In the first error report, the signal "CL18 STOP CLOCK" failed to respond correctly to the test stimulus applied. The point of visibility is given to the right of the descriptive message.

In the second error report the console register MCSR0 failed one of its data pattern tests. The expected data (test pattern) is shown under EXPD and the received data (test result) is shown under RCVD.
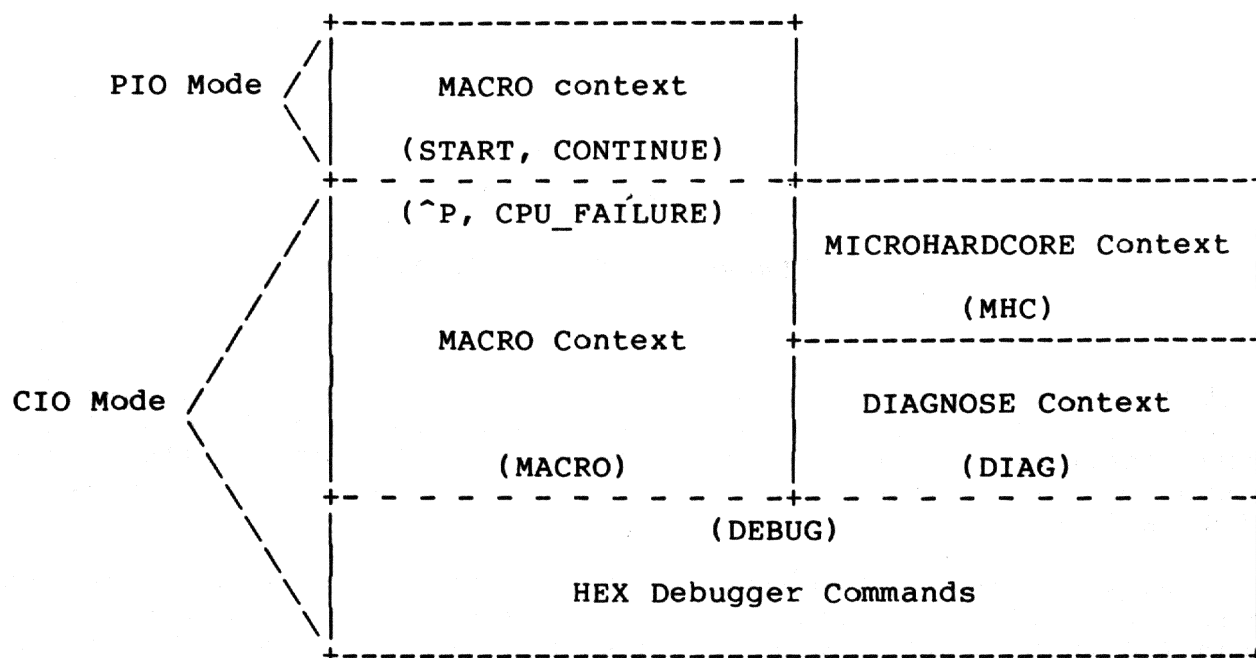
CHAPTER 5

CONSOLE SOFTWARE EXTERNAL OPERATION

This chapter discusses general console operations that are, in most cases, visible to the user. Such items include the handling of external interrupts from the VAX processor, front panel switch input and LED output, remote and local terminal handling, and system initialization.

Briefly, there are two modes of operation in the console program: Console I/O (CIO) mode, where operator command input is accepted, and Program I/O (PIO) mode, where the console acts as a slave to the VAX processor servicing its requests. These two modes are discussed in more detail in other chapters. For now, it suffices to know that there are three separate CIO mode contexts that can be run: MACRO context, DIAGNOSTIC context, and MICROHARDCORE context. Each context has its own command set and specific purpose, and are discussed in their own sections within this document. Unless otherwise indicated, the term CIO mode will be used to refer to the generic CIO mode, where any of the three contexts could be running.

The following diagram illustrates how transitions can be made between the major states of the console program using operator commands and special character input. The event (or command) required to cause a transition to a specific state is shown in parenthesis.

```
                    +----------------------+
         PIO Mode  /|       MACRO context  |
                   \|                       |
                    |   (START, CONTINUE)   |
                   +- - - - - - - - - - - -+----------------------+
                  /| (^P, CPU_FAILURE)     |                      |
                 / |                       |  MICROHARDCORE Context|
                /  |                       |                      |
               /   |                       |       (MHC)          |
              /    |   MACRO Context       +----------------------+
   CIO Mode  /     |                       |                      |
             \     |                       |   DIAGNOSE Context   |
              \    |                       |                      |
               \   |      (MACRO)          |       (DIAG)         |
                \ +- - - - - - - - - - - -+- - - - - - - - - - - -+
                 \ |            (DEBUG)                           |
                  \|      HEX Debugger Commands                   |
                   \                                              |
                    +---------------------------------------------+
```

5.1 FRONT PANEL SWITCHES

The setting of the two front panel switches, combined with the
current console mode, control the action of the console program to a
variety of external events. The console polls the state of the front
panel switches using a dedicated SDB (Serial Diagnostic Bus) control
channel.

The orientation of the switches on the VAX 8600 front panel is:

```
                              LOCAL
                                |
                                ^
   RESTART    / \     RESTART     LOCAL    / \     REMOTE
   BOOT -----/   \----- HALT      DISABLE --/   \-- DISABLE
          \       /                      \       /
   BOOT --\     /-- HALT           OFF --\     /-- REMOTE

   (Restart Control switch)        (Terminal Control switch)
```
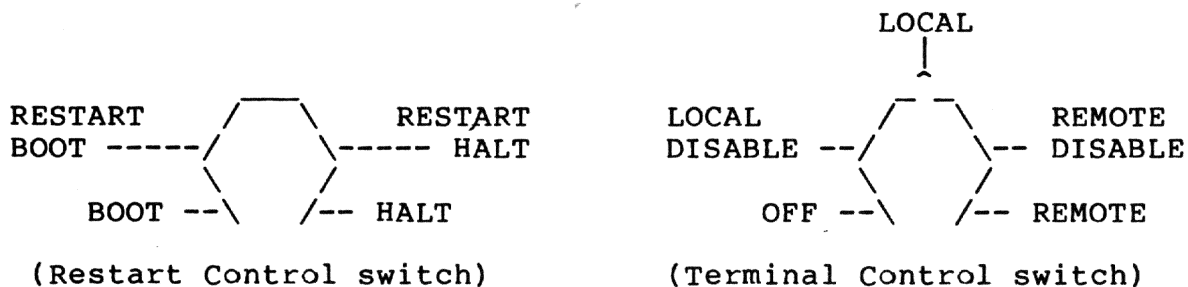
5.1.1 Terminal Control Switch

The terminal control switch (located as the right-most switch)
controls remote (RD) and local (CTY) terminal port access to Console
I/O (CIO) mode, and access by the RTY port to software running on the
VAX processor. When in the LOCAL DISABLE position (console locked)
the console program ignores the setting of the restart control switch

(left-most switch).

There are 5 positions to the terminal control switch, each described in a separate section below.

5.1.1.1 OFF Position

In this position the power system is inoperable and the battery backup unit is disabled to the CPU memory arrays (but continues to power the console's Time Of Year logic. When moved from this position the MPS provides +/-12 volt power to the EMM and the module keying circuit. If the keying circuit is satisfied (all modules in their correct slot) then regulator A is enabled, which powers the console and EMM TTL logic.

At this point, the console PROM code begins execution and performs the steps of the PROM CODE INITIALIZATION procedure.

5.1.1.2 LOCAL DISABLE Position

Interpret this position as saying "REMOTE terminal inactivated, ^P DISABLED".

With the terminal control switch in this position during console program initialization the setting of the restart control switch is ignored and the console performs an automatic RESTART BOOT attempt. An automatic restart/boot attempt will also be done in the event of a system power failure or CPU KEEP ALIVE failure.

In CIO mode the console accepts command input from the CTY device only. Remote port access is not allowed and any remote connection established will be disconnected if the terminal control switch is moved to this position.

In PIO mode the console program passes ^P characters entered at the CTY through the the CPU. Remote port access is not allowed and any remote connection established is disconnected if the terminal control switch is moved to this position.

5.1.1.3 LOCAL Position

Interpret this position as saying "REMOTE terminal inactivated, ^P ENABLED".

With the terminal control switch in this position during console program initialization the setting of the restart control switch is used to determine console action to power-up initialization and to CPU KEEP ALIVE failures.

In CIO mode the console accepts command input from the CTY device only. Remote port access is not allowed and any remote connection established will be disconnected if the terminal control switch is moved to this position.

In PIO mode the console program intercepts ^P characters entered at the CTY and exits PIO mode. Remote port access is not allowed and any remote connection established is disconnected if the terminal control switch is moved to this position.

5.1.1.4 REMOTE DISABLE Position

Interpret this position as saying "REMOTE terminal activated, but ^P DISABLED".

With the terminal control switch in this position during console program initialization the setting of the restart control switch is used to determine console action to power-up initialization and to CPU KEEP ALIVE failures.

In CIO mode the console accepts command input from the CTY device only. Remote port access is not allowed and any remote connection established will be disconnected if the terminal control switch is moved to this position, or if the console mode changes from PIO to CIO for any reason.

In PIO mode the console program passes ^P characters entered at either the CTY or RTY through to the CPU. Remote port access is allowed as a user terminal to the CPU, independent of the local terminal.

The "Local-Copy" feature, which allows all Rty dialogue to be copied to the Cty, is not enabled automatically by this switch position. See the SET LOCAL-COPY command for a description of how to enable local copy.

5.1.1.5 REMOTE Position

Interpret this position as saying "REMOTE terminal activated, ^P ENABLED".

With the terminal control switch in this position during console program initialization the setting of the restart control switch is used to determine console action to power-up initialization and to CPU KEEP ALIVE failures.

In CIO mode RTY access is allowed and the console accepts simultaneous command input from both the CTY and RTY (on a line by line basis). Refer to the REMOTE TERMINAL SUPPORT section for remote port access and operation information.

In PIO mode remote port access is allowed as a user terminal to the CPU, independent of the local terminal. The console intercepts ^P characters and exits PIO mode when encountered from either the CTY or RTY.

The "Local-Copy" feature, which allows all Rty dialogue to be copied to the Cty, is not enabled automatically by this switch position. See the SET LOCAL-COPY command for a description of how to enable local copy.

5.1.2 Restart Control Switch

The restart control switch is provided to give the user control over the console program action during console program initialization and in response to power failures and CPU KEEP ALIVE failures. When the terminal control switch is in the LOCAL DISABLE position the setting of this switch is ignored and assumed to be in the RESTART BOOT position.

The restart control switch has 4 positions, each described in a separate section below.

5.1.2.1 BOOT Position

With the restart control switch in this position during console program initialization, and the terminal control switch is NOT in the LOCAL DISABLE position, the console attempts to boot the operating system using the defaults for the BOOT command at the termination of the console software initialization.

If the boot attempt fails the console program enters CIO mode and "halts" (waits for command input). A CPU BOOT operation is also known as a COLD START attempt.

In PIO mode (CPU running), and the terminal control switch is not in the LOCAL DISABLE position, then any CPU KEEP ALIVE failures will cause the cold start action to occur.

5.1.2.2 RESTART BOOT

With the restart control switch is in this position during the console program initialization, and the terminal control switch is NOT in the LOCAL DISABLE position, the console will attempt to restart the operating system at the completion of console program initialization. If the restart attempt fails the console tries to cold start (BOOT) the operating system. If the boot attempt fails the console program enters CIO mode and "halts" (waits for command input). A CPU restart operation is also known as a WARM START attempt.

In PIO mode (CPU running), and the terminal control switch is not
in the LOCAL DISABLE position, then CPU KEEP ALIVE failure recovery
follows this same procedure of restart-attempt/boot-attempt.

5.1.2.3 RESTART HALT

With the restart control switch is in this position during the
console program initialization, and the terminal control switch is NOT
in the LOCAL DISABLE position, the console will attempt to restart the
operating system at the termination of console program initialization.
If the restart attempt fails the console program enters CIO mode and
"halts" (waits for command input).

In PIO mode (CPU running), and the terminal control switch is not
in the LOCAL DISABLE position, then CPU KEEP ALIVE failure recovery
follows this same procedure of restart-attempt/halt (note that CPU
restart attempts are made only for those KAF's defined in DEC STD 032
as ERROR HALTS).

5.1.2.4 HALT

With the restart control switch is in this position during the
console prorgam initialization, and the terminal control switch is NOT
in the LOCAL DISABLE position, the console program enters CIO mode and
"halts" (waits for command input) at the termination of console
prorgam initialization.

In PIO mode (CPU running), and the terminal control switch is not
in the LOCAL DISABLE position, then CPU KEEP ALIVE failure recovery is
not attempted.

5.1.2.5 SWITCH TABLE

The following table summarizes actions taken by the console for
all combinations of the terminal control switch (TCS) and restart
control switch (RCS). The INIT/KAF column is the action taken during
initialization and CPU KEEP ALIVE FAILURE recovery phases.

| | | ------ Action ------ | | |
RCS	TCS	INIT /KAF	RD Enbl	^P Enbl
ANY	LOC/DIS	R/B	NO	NO
BOO	LOC	B	NO	YES
RES/BOO	LOC	R/B	NO	YES
RES/HLT	LOC	R/H	NO	YES
HLT	LOC	H	NO	YES

```
BOO      RMT/DIS      B       YES  *    NO
RES/BOO  RMT/DIS      R/B     YES  *    NO
RES/HLT  RMT/DIS      R/H     YES  *    NO
HLT      RMT/DIS      H       YES  *    NO

BOO      RMT          B       YES       YES
RES/BOO  RMT          R/B     YES       YES
RES/HLT  RMT          R/H     YES       YES
HLT      RMT          H       YES       YES
```

             * Remote access allowed in PIO mode only.

5.2   FRONT PANEL LED INDICATORS

     There are four LEDs on the front panel that provide some visual
status of the machine.   Three of these LEDs are green and indicate
state information, while the fourth LED is red and   indicates
over-temp, air-flow, or EMM problems.

     During the first steps of console prorgam initialization the
front panel LEDs are used by the console PROM code to display a series
of test numbers as each test is run. See the chapter on the PROM code
for more information.

     The LEDs are controlled totally by the console program through a
dedicated SDB (Serial Diagnostic Bus) control channel. The only
exception to this is that the console RESET instruction causes all
LEDs to turn on (and remain on until software sets them to another
state). The console program is also able to read the state of the
front panel LEDs using the same SDB control channel.

     The orientation of the LEDs on the front panel is:

         (GREEN) (GREEN) (GREEN)   (RED)
            ^   ^   ^   ^
            | | | +--- ALERT
            | | +----------- REMOTE ACTIVE
            | +------------------- REMOTE ENABLE
            +--------------------------- VAX STATE
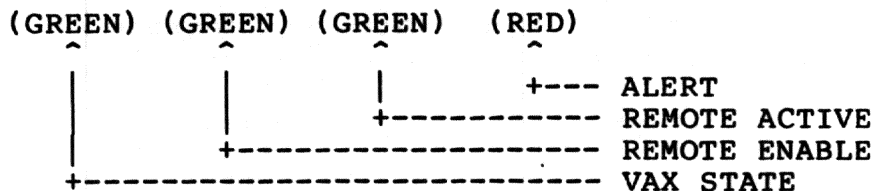
                         NOTE

     The state of the front panel LEDs is affected by a
     CONSOLE REBOOT.   When a console reboot occurs the
     console executes a RESET instruction which causes all
     LEDs   to   light.   It is not until the console
     re-synchronizes with the running CPU (enters PIO mode)
     that the LEDs will again become meaningful.

5.2.1 ALERT Indicator

     This LED is the only RED indicator on the front panel and has
FOUR states.   It is set to the ON state when the EMM notifies the
console of any "yellow-zone" temperature conditions within the CPU
cabinet.   A temperature in the yellow-zone should be considered, by
operators and users, as a warning that should be investigated
immediately.   The   ALERT LED is turned OFF when all reported
yellow-zone violations have been eliminated, as detected by the EMM.

     When a temperature measurement exceeds the upper limit of the
yellow-zone the ALERT LED is flashed at an interval of 2 seconds (1
on, 1 off) to indicate a "red-zone" condition.   This is a panic
situation that will result in total system power shutdown in one
minute if the condition is not eliminated.   Note that a condition
bordering between the yellow- and normal-zone could cause the ALERT
LED to appear to be flashing.

     The ALERT LED is also flashed at a 2 second cycle when an AIR
FLOW fault is pending.   The automatic shutdown feature is armed to
trip in 3 minutes for a single air flow violation and is shortened to
1 minute for a double air flow fault (there are two sensors that
measure air flow entering the CPU cabinet).

     The last state the ALERT LED can be in is a fast flash (1/4
second on, 1/4 second off) which indicates problems with the EMM or
EMM communication link. The LED continues to flash at this rate until
normal communication with the EMM is regained. The attempt to regain
communication is done every 30 seconds.

     There are four temperature sensors in the CPU cabinet, one which
measures inlet air and three others for outlet air temperatures. The
EMM checks for individual temperature violations at each sensor and
also for excessive temperature differentials between the inlet sensor
and each of the outlet sensors.

          The yellow and red zone limits (in degrees C)
          for each measurement is listed below:

                         yellow-zone   red-zone

          T1 (inlet)   37   39
          T2 (outlet)   45   47
          T3 (outlet)   45   47
          T4 (outlet)   45   47
          delta T1, T2   10   15
          delta T1, T3   10   15
          delta T1, T4   10   15

5.2.2   REMOTE ACTIVE Indicator

This LED is set to the on state when the carrier signal from the MODEM, or null MODEM, is asserted. When carrier is lost this LED is turned off.

5.2.3   REMOTE ENABLE Indicator

This LED is set to the on state when RTY port access is enabled by the combination of two things: the terminal control switch setting and the major mode of the console program (PIO or CIO). For instance, when the terminal control switch is in the REMOTE DISABLE position and the console is in PIO mode (i.e., VMS running) then the REMOTE ENABLE indicator is turned ON. If the console exits PIO mode (and enters CIO mode), or the terminal control switch is changed to a non-REMOTE position, the REMOTE ENABLE indicator is turned off.

When REMOTE ENABLE is ON it indicates, to the operator, that the machine is not secure from remote access.

5.2.4   VAX STATE Indicator

This LED has two states. When off, it indicates that the CPU is not running, or that the console program is not in PIO mode. When the VAX CPU is running the LED is flashed at 1.2 second intervals (.6 on, .6 off).

5.3   TERMINAL SUPPORT

There are two terminal port USART's (Signetics 2661's) available on the console for providing user access. These are known as the CTY and RTY ports. The CTY port is intended to be wired to a terminal in the vicinity of the 8600 CPU cabinet. The RTY port has MODEM capabilities and can be used to access the machine from a remote site. Input from these terminal ports is handled by the console program and either passed through to the VAX CPU, or treated as command input to the console, and this is determined by the setting of a front panel switch and the current mode the console program is running in (i.e., Program IO mode or Console IO mode).

5.3.1   In Program IO Mode

When the console is running in Program IO (PIO) mode it passes all input from the CTY port, with the possible exception of ^P, through to the VAX CPU (via the RX registers). The passing of the ^P character is controlled by the setting of the front panel Terminal

Control switch. CTY output requests from the VAX CPU (via the TX
registers) are also handled by the console while in PIO mode. The
same is also true for the RTY port if the port is enabled by the
REMOTE or REMOTE DISABLE front panel switch positions and a remote
connection has been established.

        While in PIO mode the console generates no terminal output of its
own. Terminal output is generated only by requests from the VAX CPU.

        A feature exists in the console program that allows the customer
to force a hardcopy of all RTY activity to also go to the CTY. This
feature is described in a subsection in the RTY TERMINAL SUPPORT
chapter.

5.3.2 In Console IO Mode

        When the console is running in Console IO (CIO) mode it treats
all input from the CTY port as command input. If the terminal control
switch is in the REMOTE position, and a remote connection has been
established, then RTY port input is also treated as command input
(command lines from the two ports may become intermixed).   Switching
the front panel switch to any other position will break the remote
connection.

        It is assumed that when both ports are enabled the users of the
two ports will be cooperating with each other. Each user can type on
their respective keyboard and have characters echoed back to them as
usual. When either user enters a carriage return that line just typed
is taken as the next command line, and is echoed as a line to the
other port.   Either user can ^C the command. Users can communicate
with each other using comment lines (which begin with a !).

        Refer to the REMOTE TERMINAL SUPPORT chapter for a detailed
description of the remote port handling.

5.4 SYSTEM (AC) POWER FAILURE HANDLING

5.4.1 Detection

        The console is alerted to an AC LO condition through a priority 7
interrupt that vectors the console to its power fail service routine.
The same signal source that alerts the console is also seen
simultaneously by the CPU and SBIA (signals originate on the EMM). At
the assertion of the AC LO condition software is guaranteed a minimum
of four milliseconds before DC LO asserts. At the assertion of DC LO
software operation would become unpredictable.

        If the console program is running in CIO mode the AC LO interrupt
is treated as an unexpected interrupt and vectored to the PROM code.

In PIO mode the console attempts to treat all AC LO conditions as if they where potential "brown out" conditions. A brown out condition, as far as the console is concerned, occurs when AC power drops enough to cause the AC LO interrupt but not enough to cause the console to loose DC power. Thus, the console performs a timeout after an AC LO interrupt and then, if it's still running, deasserts the MEM BUS ENABLE and ABUS ENABLE signals and forces the DC LO signal to assert. The console then jumps to the PROM code where the power up sequence begins. In this way all power failures, no matter how small, are converted to "black outs".

Also, if the AC LO condition occurs while in PIO mode the console forces an immediate message to the CTY indicating the event. Because most terminals perform character buffering internally there is a good chance that this message will make it to the terminal within the 4 millisecond window and be displayed before the terminal power is lost. No attempt is made to send a similar message to the RTY since the transfer time is well beyond the 4 millisecond window.

5.4.2 Recovery

When console power is restored after a power failure a normal CONSOLE PROGRAM INITIALIZATION sequence is performed according to the setting of the front panel switches.

5.5 POWER REGULATOR FAILURES

When the power system fails to provide stable DC power to the system the EMM alerts the system by asserting "EMM CPU DC LO" and "EMM SBIA DC LO", and then sending an exception message to the console. This is an abnormal power fail situation, since "EMM CPU AC LO" never asserted. It indicates that a DC regulator has failed.

When the console receives the priority message from the EMM the CPU has long since been dead (since the ARRAYs lock up at the assertion of DC LO). The console then requests the Ebox to enter CSM microcode, and treats the event as a non-recoverable KEEP ALIVE FAILURE. A snapshot of the system is taken (if enabled) and the CPU is forced into a cold boot (also, if enabled). Note that the cold-start attempt will fail if the DC regulator problem still exists.

5.6 CPU PARITY ERROR HANDLING

Dynamic single-bit parity error detection is provided for the following CPU components:

ECS, ICS, IDRAM, FBACS, FBMCS, FDRAM, and MCS.

The console software provides an ECC table for each of the above
rams. ECC codes are included with the udata in the corresponding .BPN
files and are stored in the ECC tables as each ram is loaded. In the
case of the FDRAM, the actual udata is stored instead of its ECC.

Control store parity errors are detected in the CPU and cause an
external interrupt to the console processor on inputs EBE CPU ERR
<2:0>. The interrupt includes a 1 of 7 code which indicates the ram
in which the error occurred.

Except for the special cases noted below, the CSPE error handler
determines the bad uaddress and reads the bad data word. An ECC code
is calculated on the bad word and processed (XOR'ed) with the original
(good) ECC to determine the bad bit position. If the error was caused
by a single bit failure, the correction algorithm will yield the
failing bit number. That bit is then complimented and rewritten back
to the ram. An error syndrome, including the bad bit number and
uaddress, is returned to the CPU and error recovery (roll back and
re-execute current instruction) is attempted.

Errors in the MCS are processed as above except that correction
and recovery are not attempted.   An additional check is done to
determine if the MCS error was caused by an ADDRESS PARITY error or a
STACK ERROR (the appropriate message is printed).

Errors in the FDRAM are not corrected via ECC, since the failing
microaddress is not available at interrupt time. Instead, the entire
ram is reloaded using the original microdata stored in the ECC table.

The correction routines include a mechanism for catching
triple-bit errors assuming that such errors are attributable to a
single bad ram chip and not distributed across 2 or more.   Triple-bit
errors are not correctable.

A calculation resulting in a zero-syndrome indicates the presense
of a transient error.   No correction is done but the same recovery
procedure is followed to allow the CPU to continue.

Note that while in CIO mode, control store parity errors are
reported but correction or syndrome calculation is not attempted.

Proper handling of control store parity errors is impossible when
failing microword data cannot be examined correctly, such as when
faults exist in visibility logic associated with control store reads.
Results can certainly be misleading if such faults exist, so be sure
that VTERM problems identified by MHC are corrected.

Whenever a non-correctable (or zero-syndrome) parity error is
encountered (in PIO mode) the good, bad, and XOR data is displayed
along with the calculated syndrome information.   The   syndrome
identifies the failing bit in the bad data word and is displayed as
part of a 32-bit register called CSES.

The error display is formatted as follows:

        Bad_nnnn....nnnn
        Good_nnnn....nnnn   Syndrome_aaaaeecc
      W.xor.R_nnnn....nnnn

    where: Bad_   is the observed (bad) udata.
            Good_   is the corrected (good) udata (zero if synd was bad).
            W.xor.R_is the XOR of the written udata and a following
                        verification read-back.
            Syndrome_is the 32 bit value returned to the EBOX for CSES
                        as per 8600 register spec.

    Note that all udata is in .BPN (physical) format.

5.7 CI780 MAINTENANCE RESTART PACKETS (CI REBOOT HANDLING)

     When a restart packet is received by the SBIA, SBI FAIL is
asserted on that SBI, followed by the assertion of SBI DEAD. The
CI780 subsystem then deasserts SBI DEAD, leaving SBI FAIL asserted.
The assertion of SBI FAIL causes the CPU (on the 11/780) to vector to
a power fail service routine and asserts one of four ABUS DEAD inputs
to the console module. The console responds to the ABUS DEAD
interrupt in the same way that it handles an actual power failure,
which includes performing CPU initialization and either restarting,
rebooting, or "halting" the CPU depending on the position of the front
panel switches. If the ABUS DEAD condition occurs while the console
is in CIO mode then it is treated as an unexpected interrupt and
control is passed to the console PROM code.

5.8 CPU REBOOT REQUESTS

     Occasionally the operating system running on the CPU will request
the console to reboot the CPU. This is known as a CPU reboot request
and is communicated to the console through the "logical console" port
of the TX register. Upon reception of this request the console
asserts "CL09 CSM REQ H" to stop the ISP and cause the Ebox to enter
the CSM console-loop. The console reports the halted PSL and PC to
the operator, as usual, and appends a "CPU reboot requested" notice to
the right of the csm.status field.

     The console then tests the state of the front panel switches to
see if the CPU reboot request is to be honored. If the switch
combination does not allow this (such as HALT or RESTART HALT
positions) then the console program does nothing but wait at the
command prompt for input. If the switches do allow the reboot (such
as BOOT or RESTART BOOT or LOCAL DISABLE positions) the console
executes an INIT/CPU command (to reset the machine to a known state)
followed by the default BOOT command.

5.9   CPU KEEP ALIVE FAILURE HANDLING

     During PIO mode execution the console program monitors a console
input signal called "EBE CPU KEEP ALIVE". This signal is asserted by
CPU logic each time it begins executing a new macro instruction.   The
console tests the state of this flag periodically (approximately once
every 300 milliseconds) and deasserts the flag if the CPU is found to
be alive.   If, for any reason, the CPU fails to continue executing
macro instructions this signal becomes inactive and the console
program detects a CPU KEEP ALIVE FAILURE (KAF). The ^P break
character, entered from the operator, is an exception to this.
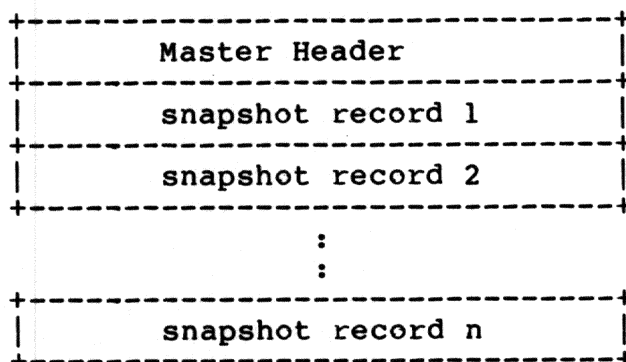
     When a CPU KEEP ALIVE FAILURE is detected the console takes a
snapshot (if the SET SNAP ON command has been done) of the VAX
processor and saves the information on the console pack. The steps to
handling a KAF are:

     1.   The CPU clock is stopped and the state of all visibility
          signals accessible through the SDB are collected.

     2.   The status of the EMM is requested and saved.

     3.   The status of the console module and RL subsystem is
          collected and saved.

     4.   The Ebox is forced to begin execution at the CSM start
          address (100D).   If CSM does not respond then the following
          step is skipped.

     5.   The contents of all IPR's, Miscellaneous registers, Internal
          registers,   and the relevant portion of the PAMM is saved,
          along with the contents of the Ebox scratchpad RAM.   The
          CSM.STATUS byte is also read and saved in the snapshot
          buffer.

     6.   A message is sent to the CTY (and RTY if enabled) indicating
          the presence of the KAF condition and a one-line statement
          describing the "reason" for the KAF failure. See description
          of the various REASON CODES in the next section.

     7.   All control stores are verified by comparing them to the
          contents of the files used to load them last (or the defaults
          for that control store if the console program has lost the
          information due to a console reboot). Note that this step
          (and the following one) takes a few minutes and because of
          this it can be disabled by the /NOVERIFY switch on the SET
          SNAP ON command.

     8.   The contents of the PAMM is compared with its expected
          contents if, and only if, the console still has the original
          PAMM data saved in T-11 memory. This data may have been lost
          during a console reboot.

9. The state of the SBIAs and NEXUSES is collected and saved.

10. Finally, the top 64 longwords on the interrupt stack are saved.

11. The collected "snapshot" information is written to the console disk file in the following manner:

    o If both SNAP1.DAT and SNAP2.DAT are invalid or don't exist, then a new SNAP1.DAT file is created and the snapshot information is written to it.

    o If the file SNAP1.DAT exists and is valid, and SNAP2.DAT doesn't exist or is invalid, then a new SNAP2.DAT file is created and the snapshot information is written to it.

    o If both SNAP1.DAT and SNAP2.DAT are valid then no data is written to the disk.

    o If the file SNAP2.DAT is valid, and SNAP1.DAT isn't, then SNAP2.DAT is renamed to SNAP1.DAT and a new SNAP2.DAT is created for information storage.

12. If the reason for the KAF, as determined by the CSM status word, indicates that one of the CPU ERROR HALT conditions occurred, as defined in DEC STD 032, then a flag is set that will direct the recovery procedure to attempt a WARM START.

13. A full CONSOLE PROGRAM INITIALIZATION is performed as described in a later section.

5.9.1 Snapshot File Format

    The general format of information in the snapshot file (e.g., SNAP1.DAT), created when a KAF occurs, is illustrated below.

```
+----------------------------------+
|           Master Header          |
+----------------------------------+
|         snapshot record 1        |
+----------------------------------+
|         snapshot record 2        |
+----------------------------------+
                 :
                 :
+----------------------------------+
|         snapshot record n        |
+----------------------------------+
     (Total size of snapshot file = 4280. bytes)
```

5.9.1.1 Master Header Record Format

The master header record is the first record of every snapshot file and contains information pertaining to the cause of the KAF and generally supportive information (such as microcode revision levels). The individual fields of the header record, and their sizes, is shown below. All byte counts are in decimal; codes and data values are hexadecimal.

Field name:	Bytes:	Description:
Header ID	1	contains "20" to indicate presence of the master header
Status Code	1	contains one of the following values FF = invalid file (stale information) 01 = valid file information
Record Length in bytes	2	total number of bytes in the master header record.
CPU program loaded	6	Name of the last file (in ASCII format) loaded by the console into CPU memory. This will be "VMB" to indicate that VMS was running at the time of the KAF, as opposed to a diagnostic supervisor, etc. The name is padded with NULLs if it is less than 6 characters long. Note: if this field is empty (all null) it indicates that the console was rebooted since the last LOAD and has lost the information.
File Length in bytes	2	total number of bytes in the entire SNAPx.DAT file.
CSM Status Code	1	contains one of the following CPU ERROR HALT codes as defined in DEC STD 032 0 = CSM could not be forced to run by the console program after the KAF. 4 = Interrupt Stack not valid 5 = Non-Ebox double error 6 = Kernel mode HALT 7 = SCB vector with <1:0> = 3 8 = SCB vector with <1:0> = 2 but no WCS 9 = pending error on HALT A = CHMx with IS = 1 B = CHMx vector <1:0> not 0
KAF Reason Code	1	contains one of the following values

18 = Parity Error in both ESC A and B.
     Ebox is looping at address 20.

19 = EHM was in the process of handling
     an error when a second error trap
     occurred. Ebox is looping at
     address 21.

1A = Ebox detected a WBUS parity error.
     The Ebox micro is looping at
     address 24.

1B = CPU ERROR HALT encountered

1C = Non-correctable parity error

1D = Power system failure. Indicates
     that the EMM sensed a DC LO
     condition without a preceding AC LO.

1E = Unidentified KAF occurred

1F = MBOX/SBIA DMA COMMAND ERROR or NXM.
     The mbox is stalled at upc FF.

Time Stamp Integer	4	VAX formatted 32-bit TOY integer
Console PROM revision	1	Prom revision number.
EMM PROM revision	1	Prom revision number.
Minimum HW revision	2	. . . All 2-byte revs have following format:
Ebox Ucode revision	2	
Ibox Ucode revision	2	byte 0 - major rev
Mbox Ucode revision	2	byte 1 - minor rev
FboxA Ucode revision	2	
FboxM Ucode revision	2	
ACCESS Ucode revision	2	
CPR Ucode revision	2	
MCF Ucode revision	2	
CONTEXT Ucode revision	2	
Ucode Release revision	2	
Ebox Ucode Verification	1	. . .All verification bytes will contain
MCF Ucode verification	1	the total number of verification errors
CONTEXT Ucode verification	1	detected (0 = none, max is 255)
FBOXA Ucode verification	1	
FBOXM Ucode verification	1	
FDRAM Ucode verification	1	
IBOX Ucode verification	1	
IDRAM Ucode verification	1	
MBOX Ucode verification	1	
CPR Ucode verification	1	
ACCESS Ucode verification	1	
PAMM verification	1	

MCC shift channel	11	saved state of the 88 bit mbox shift path.
Spare	1	
Console software version	2	Console software rev (major,minor).
CPU clock status	2	CPU_clock state (word).

$<15>$ rate 1/5 (1) or full (0).
$<14{:}08>$ frequency in mhz (1 if ext).
$<07{:}06>$ mbz
$<05>$ mark_stop (1) or not (0).
$<04>$ clk_running (1) or not (0).
$<03{:}00>$ if stopped, bit<n> = phase n.

Calculated CSPE syndrome   4   CS parity error syndrome (longword).
                                    FFFF if this crash NOT due to CSPE.

(Total size = 74. bytes)

5.9.1.2   Snapshot Record Header Format

After building the master header the console program needs to collect and save data in the form of separate "snapshot records". The data contained in a particular snapshot record is indicated by the header id code located in the first byte of the record (in the snapshot record header). The format of all snapshot record headers is shown below. All byte counts are in decimal; codes and data values are hexadecimal.

Field name:	Bytes:	Description:
Header ID	1	contains one of the following codes

21 = FBA		22 = FBM	
23 = MCD		24 = IBD	
25 = IDP		26 = ICA	
27 = ICB		28 = CLK	
29 = EDP		2A = EBE	
2B = MCC		2C = MAP	
2D = EBD		2E = EBC	
2F = CSB		30 = CSA	
31 = MTM		32 = CSL	
33 = EMM		34 = IPRs	
35 = ESC		36 = PAMM	
37 = ISP		38 = SBI0	
39 = SBI1			

Field name:	Bytes:	Description:
Status Code	1	contains one of the following values

FF = invalid record data
01 = valid record data

a record could be invalid if, for instance, CSM is not alive to provide

the console program with the data.

Record Length in bytes	2	total number of bytes in this record (including header)
CDF filename	6	name of the .CDF file used to extract the SDB data contained in this record CDF filenames will always be 6 characters. (for non-SDB records, this field will contain an ascii record name in the form 'NAMrec')

(Total size = 10. bytes)

5.9.1.3   Snapshot Record Data Formats

Associated snapshot data follows the record header and is variable length depending on the header ID of the record. For SDB visibility data, contained in record IDs ranging from 21 to 31 (hex). For all these records the data is formatted the same, but the first two (FBA and FBM) records are 74. bytes long, whereas all other SDB snapshot records are 42. bytes long. Here is a sample of the format of data in a typical SDB snapshot record. All byte counts are in decimal; codes and data values are hexadecimal.

Field name:	Bytes:	Description:
record header	10	see description above
SDB data byte 0	1	bits <7:0> of MUX SEL 0, ENA 0
SDB data byte 1	1	bits <7:0> of MUX SEL 1, ENA 0
SDB data byte 2	1	bits <7:0> of MUX SEL 2, ENA 0
SDB data byte 3	1	bits <7:0> of MUX SEL 3, ENA 0
SDB data byte 4	1	bits <7:0> of MUX SEL 4, ENA 0
SDB data byte 5	1	bits <7:0> of MUX SEL 5, ENA 0
SDB data byte 6	1	bits <7:0> of MUX SEL 6, ENA 0
SDB data byte 7	1	bits <7:0> of MUX SEL 7, ENA 0
SDB data byte 8	1	bits <7:0> of MUX SEL 0, ENA 1
SDB data byte 9	1	bits <7:0> of MUX SEL 1, ENA 1
:		:
:		:
SDB data byte 30	1	bits <7:0> of MUX SEL 6, ENA 3
SDB data byte 31	1	bits <7:0> of MUX SEL 7, ENA 3

The above example must be modified slightly for the FBA and FBM data records, which have 2 bytes of information per MUX SEL and ENA. Here's an example of a single element in the FBA/FBM record:

NOTE

Only bits <11:00> of each 2-byte set in the FBA and FBM records contain valid data. Bits <15:12> are zero and should be ignored.

SDB data byte 0	2	bits <11:0> of MUX SEL 0, ENA 0
SDB data byte 1	2	bits <11:0> of MUX SEL 1, ENA 0
:		:
SDB data byte 31	2	bits <11:0> of MUX SEL 7, ENA 3

The snapshot record format for CSL data (data collected regarding the state of the console module) is:

Field name:	Bytes:	Description:
record header	10	see description above
MCSR0	1	state of console register
MCSR1	1	state of console register
MCSR2	1	state of console register
MCSR3	1	state of console register
ERSR	1	state of console register
LRSR	1	state of console register
RRSR	1	state of console register
spare	1	
QCSR0	1	state of console register
QCSR1	1	state of console register
QCSR2	1	state of console register
QCSR3	1	state of console register
SID0	1	state of console register
SID1	1	state of console register
SID2	1	state of console register
SID3	1	state of console register
RL CTLR STATUS	2	state of console load device
RL DRIVE STATUS	2	state of console load device

(Total size = 30. bytes)

The snapshot record format for EMM data is:

Field name:	Bytes:	Description:
record header	10	see description above
POWREG	1	regulator on/off state
MARGEN	1	margen enable register
MARHILO	1	margen high/low register
MODOK	2	module OK register
MISREG	1	misc. hardware status
SWREG	1	misc. software status
PROM revision	1	Prom revision number
REGULATOR_A VOLTAGE	2	voltage measurement of +5V
REGULATOR_B VOLTAGE	2	voltage measurement of +5V
REGULATOR_C VOLTAGE	2	voltage measurement of +5V
REGULATOR_D VOLTAGE	2	voltage measurement of -2V
REGULATOR_E VOLTAGE	2	voltage measurement of -2V
REGULATOR_F VOLTAGE	2	voltage measurement of -5.2V
REGULATOR_H VOLTAGE	2	voltage measurement of -5.2V
GND CURRENT VALUE	2	voltage measurement of Ground current circuit
REGULATOR_L + VOLTAGE	2	voltage measurement of +12V
REGULATOR_L - VOLTAGE	2	voltage measurement of -12V

```
REGULATOR_K + VOLTAGE    2        voltage measurement of +15V
REGULATOR_K - VOLTAGE    2        voltage measurement of -15V
T1 TEMPERATURE VOLTAGE   2        voltage measurement of input thermistor
T2 TEMPERATURE VOLTAGE   2        voltage measurement of output thermistor
T3 TEMPERATURE VOLTAGE   2        voltage measurement of output thermistor
T4 TEMPERATURE VOLTAGE   2        voltage measurement of output thermistor
                                  (Total size = 50. bytes)
```

The snapshot record format for Ebox Scratchpad (ESC) data is:

Field name:	Bytes:	Description:
record header	10	see description above
Escratch location 0	4	32-bit Ebox scratchpad ram data
Escratch location 1	4	32-bit Ebox scratchpad ram data
Escratch location 2	4	32-bit Ebox scratchpad ram data
:		:
:		:
Escratch location 254	4	32-bit Ebox scratchpad ram data
Escratch location 255	4	32-bit Ebox scratchpad ram data
		(Total size = 1034. bytes)

The snapshot record format for IPRs and miscellaneous registers included under the header id of "IPR" is:

Field name:	Bytes:	Description:
record header	10	see description above
KSP	4	Internal Processor Registers
ESP	4	
SSP	4	
USP	4	
ISP	4	
P0BR	4	
P0LR	4	
P1BR	4	
P1LR	4	
SBR	4	
SLR	4	
PCBB	4	
SCBB	4	
IPL	4	
ASTLVL	4	
SISR	4	
ICCS	4	
ICR	4	
TODR	4	
RXCS	4	
RXDB	4	
TXCS	4	
ACCS	4	
MAPEN	4	
PME	4	
SID	4	

PAMACC	4	
PAMLOC	4	
CSWP	4	
MDECC	4	
MENA	4	
MDCTL	4	
MCCTL	4	
MERG	4	
EHSR	4	
STXCS	4	
STXDB	4	
VPCBITS	4	Internal Registers
EDPSR	4	
EBCS	4	
CSLINT	4	
EDMC	4	
IBESR	4	
EMD	4	
IVASAV	4	
VIBASAV	4	
ESASAV	4	
ISASAV	4	
CPC	4	
MSTAT1	4	
MSTAT2	4	
MEDR	4	
MEAR	4	
CSHCTL	4	
CSES	4	Miscellaneous Registers
IBGPR	4	
PSL	4	
SPADR	4	
STATE	4	
EVMQSAV	4	

(Total size = 250. bytes)

The snapshot record format for PAMM data is shown below. Note that only the low-order byte of the PAMM data is included, as all other bits are irrelevant.

Field name:	Bytes:	Description:
record header	10	see description above
PAMM location 0	1	contents of PAMM array
PAMM location 1	1	
PAMM location 2	1	
:		
PAMM location 1022	1	
PAMM location 1023	1	

(Total size = 1034. bytes)

The ISP record contains the top 64. longwords on the interrupt stack.

The snapshot record format for the ISP record is shown below.

Field name:	Bytes:	Description:
Record header	10	see description above.
(ISP)	4	1st long word on interrupt stack.
4(ISP)	4	2nd long word
:		:
256(ISP)	4	64th long word on interrupt stack.

(total size = 266. bytes)

The snapshot record format for the two SBIA records (SB0/SB1) is shown below.

Field name:	Bytes:	Description:	
Record header	10	see description above.	
Configuration	4	Configuration.	2x08 0000
Control/status	4	Control/status.	2x08 0004
Error summary	4	Error summary.	2x08 0008
Diag control	4	Diag control.	2x08 000C
DMAI cmd/addr/id	8	DMAI cmd/addr and id.	2x08 0010
DMAA cmd/addr/id	8	DMAA cmd/addr and id.	2x08 0018
DMAB cmd/addr/id	8	DMAB cmd/addr and id.	2x08 0020
DMAC cmd/addr/id	8	DMAC cmd/addr and id.	2x08 0028
SBI silo	64.	SBI silo (4x16.)	2x08 0030
SBI error	4	SBI error.	2x08 0034
SBI time-out addr	4	SBI time-out addr.	2x08 0038
SBI fault status	4	SBI fault status.	2x08 003C
SBI silo comparator	4	SBI silo comparator.	2x08 0040
SBI maintanence	4	SBI maintenance.	2x08 0044
TR1 NEXUS	4	TR1 NEXUS csr	2x0n n000
error summary	4	error summary.	2x08 0008
error	4	error.	2x08 0034
fault status	4	fault status.	2x08 003C
TR2 (thru TR15) NEXUS	16.x14.	TR2 thru TR15 NEXUS (as above).	

(total size = 382. bytes each)

5.10   CONSOLE SUBSYSTEM REBOOT

The console can be rebooted on request from the VAX CPU (thru the
CRBT IPR register).   The console reboot request causes the console
program to vector directly to a fixed PROM location where the reboot
procedure begins.   The action of the PROM code is described in the
chapter on the PROM.

When the console program begins running again it determines if
the CPU is already running. This decision is based on the state of
two indicators: if the CPU clock is running, and if the MEM BUS ENA
signal (in console status register MCSR1) is asserted. If the CPU is
running the console restores the state of several software flags and
parameters that where saved prior to the reboot and enters PIO mode to
allow normal system operation to continue. If the CPU is not running
the console assumes a normal power up is occurring and looks at the
front panel switches to determine its actions.

When PIO mode is first entered after a console reboot the console
sends a message to the running CPU to confirm the successful
completion of a console reboot. A description of the message sent is
described in another section of this document (in the section
describing the RX register).

                              NOTE

        During the console reboot cycle the state of all front
        panel LEDs is affected. It is not until the console
        re-synchronizes with the running VAX CPU (enters PIO
        mode) that the LEDs will again become meaningful.

5.11   CONSOLE PACK HANDLING CONSIDERATIONS

The console program monitors the state of the console RL02, in
both CIO and PIO modes, to detect pack changes made by the operator.
When a pack change is detected a software flag is set in the console.
The next CIO mode disk access attempt is preceded with a checksum
verification of the first block of the console software image.   If
this pack verification passes the pack is assumed valid and the
intended I/O is performed.   If the verification fails then the
operator is requested to replace the original console pack by the
following message:

        ?WRONG PACK -- remount original console pack and hit <CR>, or
                  type ^C to reboot the pack presently mounted

The pack verification check is repeated after the <CR> response
is entered, and if the verification again fails the message is
repeated. The only escape from this message is for the correct pack
to be re-installed in the drive or a ^C typed as a reponse to the

above message. The ^C essentially causes a REBOOT command to be
executed, which may attempt to return to PIO mode when
re-initialization completes.

The recommened technique of swapping the console pack is to:

1. have the console running in CIO mode (the CPU may be running
    too)

2. swap the console pack

3. enter the REBOOT command at the console prompt

The console will reboot the RL, reload tables and overlays and,
if the CPU is running, attempt to re-enter program mode.

5.11.1   Standalone Backup Procedure

In order to use the Standalone Backup utility to build a VMS
system disk it is necessary that pack changes be done while the
console is in both CIO and PIO mode. The pack being swapped-in is a
files-11 structure containing the Standalone Backup image and
associated files. The procedure for running Standalone Backup is as
follows:

o After successful completion of the FULL console prorgam
   initialization enter the command "BOOT CS1" to load VMB from
   the console pack with the intent of loading the VMS operating
   system from the same device.

o VMB will prompt the operator to replace the console pack with
   the VMS structured pack. The console pack can be removed and
   the new pack inserted in the drive. When the pack is ready,
   respond to the VMB prompt.

o VMB will load and start Standalone Backup, which will perform
   its normal pack build function.

o When Standalone Backup completes the original console pack
   should be returned to the console drive.

CHAPTER 6

CONSOLE SOFTWARE (MACHINE) INITIALIZATION

In the event of console power being turned on or restored after a power failure the console software performs a FULL machine initialization. This initialization begins with the execution of code in the console PROM, passes to a once-only initialization (once per console program boot) of the console program, and ends with the MACRO context initialization. The only other time the console performs the FULL initialization is in response to either the CONSOLE REBOOT request from the CPU, or the REBOOT command itself.

There are two ways to interrupt the flow of the FULL initialization sequence, which may be necessary if a console hardware problem or CPU hardware problem is suspected. By interrupting the sequence while the PROM code is running (by hitting any keyboard character within 5 seconds after the sound of the terminal BELL) the operator is given access to the limited PROM command set where the T(est) command may be used to load and run either the console diagnostic or console disk subsystem diagnostic. By interrupting the sequence during MACRO context initialization (using the ^C character) the operator is given immediate access to the MACRO context command set and may enter the DIAG or MHC commands to run microdiagnostics or microhardcore tests, respectively.

6.1 PROM CODE INITIALIZATION

See the "PROM code overview" in the PROM code operation chapter for a description of prom initialization.

6.2 CONSOLE PROGRAM INITIALIZATION

Console program initialization is performed whenever control is passed from the PROM code (following the PROM code initialization), or in response to the RESTART command. Once it begins it cannot be interrupted by the operator. The steps taken in this portion of the machine initialization include the following.

1. Start a 260-millisecond marktimer for REMOTE PORT and front panel LED (REMOTE ENABLE, REMOTE ACTIVE and ALERT) handling.

2. Display the console program banner and revision number.

3. Initialize interrupt vectors and various internal data structures.

4. Read and checksum the first block of the console program disk image and save for use in detecting console pack changes.

5. Initialize and enable the console RAM mapping logic.

6. Print "Initializing" message to indicate standard console initialization in progress.

7. Load the HEXBOX overlay to construct the SDB name tables. This procedure opens CDF86n.DAT and extracts .CDF filenames.

8. Buffer several CSM microcode overlays to eliminate most disk I/O dependencies while in PIO mode.

9. Start a 260-millisecond marktimer for KAF detection and front panel LED (VAX STATE) handling.

10. If the MEM BUS ENA bit in MCSR1 is set, and the CPU clock are running, then this is a console reboot recovery: restore the state of console control flags and the remote terminal parameters.

11. Print "Initializing power system" message and init the MPS (see the section describing "POWER SYSTEM INITIALIZATION").

12. Turn off all front panel LEDs.

13. Initialize the system clock by executing commands in CLOCK.COM.

Note that many of the above steps are skipped when the RESTART command is used.

Following the steps above is the power-system initialization, described below. This is done whether or not the console is recovering from a console reboot.

6.2.1 Power System (MPS) Initialization

Initialization of the power system involves the following steps. Note that this procedure is followed, for the most part, by the INIT/POWER command.

1. Initialize the EMM protocol control block. The console
    assumes the EMM unit number is 0.

2. Initialize the console's PCI used for EMM communication. The
    line characteristics used here are 9600 baud, 8 data bits, 1
    1/2 stop bits, and no parity generation/checking.

3. Start a 20-millisecond marktimer to control EMM protocol
    state timeouts and message transmissions.

4. Check that the EMM is alive by requesting status information.

5. The regulator-voltage and temperature parameter limits are
    down-line-loaded into the EMM. Note the following:

    o The parameter limits loaded into the EMM allow a 3% to 4%
       deviation in regulator voltage measurements before the
       being reported by the EMM.

    o The yellow-zone/red-zone limit for T1 is 37/39 degrees C.

    o The yellow-zone/red-zone limit for T2, T3, and T4 is
       45/47 degrees C.

    o The yellow-zone/red-zone limit for all temperature
       difference measurements is 10/15 degrees C.

    o Automatic system shutdown due to a red-zone temperature
       is 1 minute.

    o Automatic system shutdown due to a single bad air flow is
       3 minutes.

    o Automatic system shutdown due to a double bad air flow is
       1 minute.

6. The DEFAULT MODE MASK register is loaded to select which EMM
    functions will be enabled. Note that the GROUND CURRENT
    MONITOR voltage checking is not enabled because of
    inconsistencies between systems at different sites.

7. Using a request to the EMM, check that regulator A, K, and L
    are OK.

8. Using a request to the EMM, check that all regulators that
    are presently off have deasserted their MODULE OK signals.

9. Using a request to the EMM, read the state of regulator B OK
    and save it in T-11 memory as the "BBU valid on power up"
    flag.

10. Using a request to the EMM, disable all regulator output voltage marginning.

11. Using requests to the EMM, turn on regulators B, F/H, D/E, and C (in that order) and check that each regulator is OK as it is turned on.

12. Delay 100 milliseconds and then command the EMM to deassert its 'EMM LAT DC LO' latched output and check that it does deassert.

13. Command the EMM to enable its 5.5 interrupt and check that the interrupt does not occur (is the DC LO interrupt).

14. Command the EMM to deassert its 'EMM LAT AC LO' latched output and check that it does deassert. Then clear the CPU AC LO latch on the console module.

15. Delay 100 milliseconds to allow the power system to stabilize.

16. Command the EMM to assert and deassert its 'AIR FLOW RESET' output to clear the magnetic disk display on the EMM's front panel and to reset any possible latched air flow faults. Check that there is no air flow fault.

17. Command the EMM to enable its default mode operation.

The power-system initialization is followed immediately by MACRO context intialization, described below, which are the final steps involved in the full machine initialization. During console reboot recovery much of this phase is bypassed, as indicated in the text.

6.3 MACRO CONTEXT INITIALIZATION

MACRO context initialization is performed automatically during the console program initialization since MACRO context is the default context after a power-on or console reboot. MACRO initialization will also be performed whenever the MACRO command is entered, or when a CPU KEEP ALIVE FAILURE recovery is being attempted.

This portion of the initialization may be aborted at any time by the ^C or ^P characters. Aborting MACRO context intialization is useful if the users wants to switch immediately to DIAGNOSTIC or MICROHARDCORE context, where context-specific initialization is done automatically.

1. If this is a console reboot recovery (MEM BUS ENA and CPU CLOCK true) then print the message "Reloading control store ECC data" and submit the file ULOAD.COM for special execution (only ECC data is taken from the microcode files and saved in

T11 memory). When complete, print the message "Console Reboot complete, entering PIO mode" and enter PIO mode immediately.

2. Print "Initializing CPU" message to indicate the start of this section.

3. Submit for execution the command file LOAD.COM, which performs a complete initialization of the CPU for use as a MACROcode processor. The steps taken by the LOAD.COM file is described in the "CPU INITIALIZATION" section.

4. Unconditonally clear the system cache (without affecting the state of main memory) followed by an UNJAM operation to the adapters found on the system.

5. If this is the first initialization after a system power on and the battery backup unit was not active on entry, or if this is a context switch coming from DIAG or MHC context, then perform a CLEAR MEMORY operation.

6. If the terminal control switch is in the LOCAL DISABLE position, or if the restart control switch is in the RESTART BOOT or RESTART HALT position then attempt a warm-start. Otherwise, go to the next step. The steps involved in a warm-start attempt are:

   o Print "Attempting System Restart" message.

   o Test that the BBU was valid at the time power was restored (flag saved during console program initialization). If not, the warm-start attempt has failed (go to the next step).

   o Test if the WARM flag is set, which indicates that a previous warm-start attempt was not successful. If set, the warm-start attempt has failed (go to the next step).

   o Command CSM to search for a valid Restart Parameter Block. If not found, the warm-start attempt has failed (go to the next step).

   o Load VAX processor GPR's with applicable data, as per DEC STD 032, and set the WARM flag.

   o Start the VAX processor running at its restart address (located in the RPB).

   o Enter PIO mode (see section on "PIO MODE INITIALIZATION").

7. If the restart attempt failed and the terminal control switch
    is in the LOCAL DISABLE position, or if the restart control
    switch is in the RESTART BOOT or BOOT position then attempt a
    cold-start. Otherwise, go to the next step.

    o Print "Attempting System Bootstrap" message.

    o Test if the COLD flag is set, which indicates that a
       previous cold-start attempt has failed. If set, the
       cold-start attempt has failed (go to the next step).

    o Load VAX processor GPR's with applicable data, as per DEC
       STD 032, and set the COLD flag.

    o Submit the BOOT command, which in turn submits for
       execution the file DEFBOO.COM. A sample of the
       DEFBOO.COM file is shown in an appendix.

8. If the boot attempt failed or if the restart control switch
    is in the HALT position then enter the CIO null loop, MACRO
    context, and prompt for operator commands.

6.3.1 CPU Initialization

     CPU initialization is performed by the console as part of the
MACRO context initialization or whenever the command file LOAD.COM is
invoked. The steps performed by the LOAD.COM file are described
below. A sample of the LOAD.COM file and ULOAD.COM file is shown in
an appendix.

1. The RESET command is entered to perform a Master Reset of the
    machine.

2. The INIT/POWER command is used to initialize and/or ensure
    that the power system is fully operational.

3. The INIT/SDB command is used to force all SDB control
    channels to a known state to allow microcode loading.

4. The INIT/MICRO command is used to load all control stores
    with the default system microcode. INIT/MICRO submits the
    file ULOAD.COM for execution (see sample of ULOAD.COM in
    appendix).

5. The INIT/SDB command is again used to force all SDB control
    channels to a NOP or RUN state.

6. The INIT/CPU command is used to initialize CSM and many of
    the CPU's registers and IPR's. See the description of the
    INIT/CPU command in the MACRO context command section.

7. Lastly, the PAMM is initialized using the INIT/PAMM command.

6.3.2 PIO Mode Initialization

On entry to PIO mode, which occurs as result of the START or CONTINUE commands, or automatically after a console reboot or KAF recovery, special initialization is performed, as follows:

1. All opened command files are closed.

2. CTY and RTY drivers are set to single-character mode, 8-bit, passall.

3. The RX register is setup to notify the CPU of a new remote terminal connection if the CPU doesn't already know of it.

On entry to PIO mode some additional checks are done, such as for a need to acknowledge the completion of a console reboot to the CPU, or the acknowledgement of a Console Command String. Console Command Strings (CCS's) are used solely by diagnostic programs, such as EDKAX, to perform special console-cpu interactions. /CCS's are described slightly in the 8600 REGISTER SPEC./

6.4 DIAGNOSTIC CONTEXT INITIALIZATION

DIAGNOSTIC context initialization is performed whenever the DIAGNOSE command is entered, whether of not DIAGNOSTIC context is already running. This initialization brings the machine to a state where microdiagnostics can be loaded, executed, and monitored. The steps taken by this procedure are:

1. The DC initialization file DCLOAD.COM is submitted internally tc perform the following steps:

    o Set default diagnostic switch settings.

    o Stop the CPU clock and perform a Master Reset.

    o Load control stores as necessary to initialize parity logic, etc.

    o Load the Diagnostic Support Microcode (DSM) into the Ebox control store.

    o Force the Ebox to begin executing at the DSM start address and start the CPU clocks.

2. Start the mark time service to check TXCS RDY flag.

3. Display the DIAGNOSTIC context command prompt (DC>) and await
    commands.

6.5 MHC CONTEXT INITIALIZATION

The MICROHARDCORE context performs its own general initialization
upon entry, and each MHC test executes a more local initialization
when started.

CHAPTER 7

REMOTE TERMINAL (RTY) SUPPORT

The descriptions of RTY handling in this chapter apply to the console program (ED0AA) only -- the PROM code has a simplified connect/disconnect procedure discussed in another subsection.

The MODEM connect and disconnect protocols described in DEC STD 052 have been used in the design of the remote terminal support. This implementation allows auto-answer MODEM support to work properly in all areas of the world. Auto-dial is not supported.

Once the MODEM connection has been established the remote user has the option of running the Rty in "Transparent mode" (non-protocol), which is the default mode for new connections, or in protocol mode. Protocol mode can only be turned on with a special protocol packet transmission to the console and is therefore restricted to inter-computer communication.

7.1 ESTABLISHING AN RTY CONNECTION

The following conditions must exist before a remote port connection can be made by the console program.

   o If the console program is in CIO mode the terminal control switch must be in the REMOTE position.

   o If the console program is in PIO mode the terminal control switch must be in either the REMOTE or REMOTE DISABLE position.

   o A MODEM device must be connected to the rear async-panel port for the remote terminal, or a nearby terminal may be connected using a NULL MODEM cable (the NULL MODEM must assert DSR and CD to the console).

With either of the first two items satisfied the console program asserts the DTR (Data Terminal Ready) and RTS (Ready To Send) signals to the MODEM, turns on the REMOTE ENABLE light, and waits indefinitely for DSR (Data Set Ready) from the MODEM. Note that the console

deasserts the DSRS (Data Set Rate Select) signal during its
initialization.

Once DSR asserts the console program begins a 30-second timeout
waiting for CD (Carrier Detect) from the MODEM to assert. The console
program does not look at the state of CD until approximately 520
milliseconds after the assertion of DSR to avoid data transfers during
this turn-on period.

If the 30-second timer expires before CD asserts then the console
initiates a disconnect sequence (see next section). Otherwise, the
assertion of CD triggers the console program to turn the front panel
REMOTE ACTIVE light on, and to send the message "<CR><LF>Console
Password? " to the remote terminal if the password feature is enabled
(see SET TERMINAL command). If the password prompt is sent the
console begins a 2-minute timeout waiting for the correct password.
If the timeout occurs a disconnect sequence is initiated.

At this point the remote connection has been made (remote
active).

Once the password is successfully entered the console program
sends an RX register request to the CPU (asserts logical CARRIER in
RXCS). This is done only if the console is in PIO mode and only if
the logical DTR bit in RXDB has been previously set by the CPU. In
either case, the console then enters a state where it processes CTY
and RTY characters and monitors different disconnect indicators.

7.2 RTY DISCONNECTION

A disconnect sequence occurs at any stage of the remote port
connection and steady-state operation when any of the following events
occurs:

    o CD (carrier detect) is not asserted within 30 seconds after
       the assertion of DSR (Data Set Ready)

    o the password is not entered within 2 minutes

    o DSR (data set ready) from the MODEM deasserts

    o CD (carrier detect) from the MODEM deasserts for more than 2
       seconds

    o the Terminal Control Switch is changed to a non-remote
       position

    o console enters CIO mode (for any reason) with the TCS in the
       REMOTE DISABLE position.

The disconnect sequence involves a few steps. First of all, the DTR signal to the MODEM is deasserted and a 2-second timer is started. The remote port transmitter and receiver are turned off and the LOGICAL CARRIER flag to the CPU is deasserted. The disconnect sequence ends when either the DSR signal from the MODEM deasserts or the 2-second time expires.

7.3 OPERATION

7.3.1 Transparent Mode

Once a remote port connection has been established, and the console password prompt has been satisfied, the console processes input and output characters with the remote terminal. Special internal flags are provided which allow the RTY user to remain connected while running a non-operating system type program in the CPU (see Provisions for running SA below).

The ^P break character is honored from the RTY user if the front panel switch is in the REMOTE position.

In PIO mode, RTY port input characters are sent to the VAX CPU through the RX register interface with no control character checking or masking (providing of course that the CPU has accepted the remote connection). RTY characters are echoed by the VAX operating system through the TX console While in PIO mode all input from the RTY and CTY is sent to the VAX CPU independent of each other. In this way the two terminal ports behave as separate devices, even though the console is controlling character flow to and from both of them.

In CIO mode, input characters from the RTY are echoed (to RTY port only) and placed in a command input buffer. The standard command line control characters (^U, ^R, ^O, and rubout) can be used by the RTY user. When the command line terminator is entered the contents of the RTY command buffer is displayed on the CTY device and the command is parsed and executed. Any output resulting from the command is sent to both terminal ports.

The following operational characteristics of the RTY handler should be noted.

o The console password is not effective until the SET TERMINAL/PASSWORD command is encountered (optionally) in the LOAD.COM file during MACRO context initialization. If the front panel terminal control switch is in the REMOTE position and a remote connection occurs before this command is encountered then the remote connection will be made without a password. To safegard against this the terminal control switch can be placed in any non-REMOTE position during system initialization.

o When prompted, the password must be entered within 2 minutes.
   Upper and lower case characters are treated equally for the
   password. A <CR> is needed to terminate input.

o There are no "talk mode" commands between the Rty and Cty.
   However, the local operator may place the front panel switch
   in the REMOTE position allowing both Cty and Rty users to
   enter CIO mode where inter-terminal communication can be
   performed using the exclamation prefix character. This can
   be done without affecting the operation of the CPU. If the
   remote protocol is running the remote host system may
   "broadcast" a message to the local Cty.

o The remote user's input characters are echoed only while the
   console program is executing in its null loop. If the
   console program is executing a command (whether it was
   entered by the remote user or local user) then input
   character echoing will be suspended until the executing
   command completes. Because of this, the RTY input buffer in
   the console program can easily be filled with type-ahead
   while the console program is executing a command.

o The SET TERMINAL command cannot be used from the RTY, with
   the exception of the SET TERM/SCOPE and SET TERM/NOSCOPE
   switches. SET LOCAL-COPY OFF also cannot be issued from the
   Rty.

o In both CIO and PIO mode the console program will send the
   XOFF control character to the RTY when the console's input
   ring buffer is 3/4 full. This in indicated by the appearance
   of the KBD LOCKED led on a VT100 device. The XON control
   character is sent once when the input buffer is 1/4 full.
   This does not occur if the remote protocol is running.

o When in CIO mode, the ^S control character (XOFF) entered
   from either terminal will suspend console program operation
   for both terminals. A ^Q (XON) entered at the same terminal
   will resume console program output. A ^C or ^P entered from
   either terminal forces an XON sequence which frees terminal
   output if locked from a previous XOFF (note: this works only
   if the terminal does not have "AUTO XOFF" enabled in its
   setup). This does not occur if the remote protocol is
   running.

o The PROM "T" command cannot be used by the remote user since
   it tests the RTY interface, as is also the case with the
   console diagnostic (ED0BA).

o Entry to RT11 fom the remote terminal, via the PROM/RT
   command, is not supported and results in a warning message.

7.3.2 Protocol Mode

As soon as the MODEM connection has been established the remote host may send the RCP (remote console protocol) startup packet. This can be done before answering the console password prompt. With RCP running the remote host must enter the console password, if enabled, or send a broadcast message to the Cty to request assistance. Once the connection is complete the operation of the console and VMS is identical to that of transparent mode.

Operation of RCP is described in the RCP DESIGN SPECIFICATION available for internal use from the FSE group in Stow, Mass.

7.3.3 Provisions For Local Copy Of RTY Activity

To comply with security requirements of some European customers a feature for using the CTY as a hardcopy output device for logging RTY activities has been added to the console program. A software control flag, LOCAL-COPY, has been defined to enable or disable this feature, along with the setting of the front panel switches. Only the CTY user can SET LOCAL-COPY OFF; either user can turn the feature on.

To enable local-copy the user SETs the LOCAL-COPY flag ON. From that point all further RTY log-ins and activity are logged on the CTY. Note that this has an affect of making the CTY fairly useless as a user terminal while the RTY connection is active. Operating system messages sent to the CTY will still go there but may be mixed with RTY dialogue.

While LOCAL-COPY is enabled checks are made to block control-type (non-printing) characters from being echoed on the CTY. These characters pass directly between the CPU and the RTY.

Since the affects of setting the LOCAL-COPY flag only last until the next console re-initialization it is recommended that the command (SET LOCAL ON) be placed in the standard console initialization file (LOAD.COM). The state of the LOCAL-COPY flag is preserved through console reboots. Also, the setting of the terminal control switch (REMOTE versus REMOTE DISABLE) has no affect on the enabling or disabling of the LOCAL-COPY feature.

7.3.4 Provisions For Running Stand-alone Programs

Because of autonomy between the CTY and RTY terminal ports, as seen by the CPU, a limitation arises for the remote user when trying to run stand-alone CPU programs that don't specifically handle the RTY device. This restricts the RTY user from running programs like EVKAA (a basic macro-instruction test) or even booting VMS in conversational mode where sysgen parameters need to be changed well before VMS would normally begin to service the RTY device.

To eliminate these restrictions the console program has been
enhanced to duplicate CTY output on the RTY, and to pass RTY input to
the CPU through the CTY port, if the following conditions exist:

   1. the transition from CIO mode to PIO mode was invoked by the
      RTY user, and

   2. the CPU program has not yet asserted the "logical DTR" flag
      in the RXCS register

With these conditions met the remote user continues to have
control of the CPU, in parallel with the CTY, until the remote
connection is broken or the CPU asserts "logical DTR", at which point
the normal RTY controls will take over.

CHAPTER 8

EMM HANDLING

The EMM serves as an intellegent controller for the VAX 8600 power system. It provides power system state information to the console upon request, and reports good-to-bad and bad-to-good transitions of temperature sensors, regulator output levels, and other various power system events. On command from the console it also turns on power to the system, margins regulator outputs, and resets error latches.

Several console commands request services from the EMM. These commands are INIT/POWER, SHOW POWER, SHOW VERSION, SET MARGIN, and SET BBU. The CPU can also request status from the EMM (through the console) and select voltage margin levels.

Details of the EMM interface registers can be found in the EMM INTERFACE SPECIFICATION.

8.1 EMM/CONSOLE COMMUNICATIONS

Communication between the EMM and console is done using a specially designed protocol functionally similar to the ETHERNET protocol. The console/EMM protocol is called XXNET and is technically a multiaccess (multidrop) protocol with collision detect. It operates over a single twisted wire pair at 9600 baud and provides adequate data integrity of messages. A complete description of this protocol is available in the "XXNET - A PROTOCOL FOR EMM TO HOST COMMUNICATIONS" specification.

Two types of protocol messages can be sent over the communication channel: normal data messages, and priority messages. The console requests status and sets power system state with normal data messages, which are also used by the EMM to respond to these requests. Priority messages are sent only by the EMM to alert the console of some change in the monitored system status.

In PIO mode, the CPU may request status from the EMM through the console and can also select voltage margins controlled by the EMM. The console receives such requests through the TX registers and directs the request to the EMM. EMM responses are then sent back to

the CPU to complete the handshake. /The CPU requests honored are
described in the 8600 REGISTER SPEC./

8.2 POLLING FOR EMM STATUS

Under normal operating conditions the console polls the EMM every
5 seconds to request status of its RTDREG. This register returns a
code used to determine if the EMM has recently encountered an internal
failure, and specifically what that failure was. A code of zero
indicates that no failures have occured since the last one detected by
the console using this polling technique. If the EMM doesn't respond
to the request, or the code returned indicates trouble, then the
console re-initializes the EMM parameter table and re-arms "default
mode" operation of the EMM. If the re-initialization fails it is
repeated 3 times, and if the final attempt fails the EMM is reported
as being "temporarily out of service" and the ALERT led begins to
flash. The console begins polling the EMM every 30 seconds to attempt
re-initialization. If unsuccessful, no error messages are reported
but the ALERT led continues to flash. If the EMM comes back to life
the ALERT led is turned off and the console resumes its normal
5-second polling interval. Note that this same situation could occur
due to excessive transmission errors between the console and EMM.

When the RTDREG contains a failure code it means that the EMM has
encountered the indicated fault and has restarted its own internal ROM
program in attempt to recover. Successful recovery is implied when
the READ RTDREG request is answered properly, meaning the EMM is back
on its feet.

The types of EMM failures reported in the RTDREG are:

RTDREG code	Failure
0	No failure encountered
1	EMM RAM parity error encountered (and recovered)
2	EMM encountered a RESTART 1 instruction
3	EMM encountered an unexpected trap to PC 0
4	EMM encountered an unexpected TRAP interrupt
5	EMM encountered an unexpected 6.5 interrupt

8.3 PRIORITY MESSAGE HANDLING

The console program polls for received EMM priority messages once
during each pass through its control (null) loop. This is done in
both CIO and PIO modes. When a priority message is received and
detected it is broken down and reported directly on the system console
terminal(s) ONLY WHEN DETECTED IN CIO MODE. In PIO mode, the EMM
message is reformatted and sent to the CPU through the RX registers,
where it is then entered into the system error log. It is up to the
macro program running on the CPU whether or not to print anything on

the console terminal.

After sending a priority message concerning a bad air flow or red-zone temperature the EMM starts a 1 to 3 minute timer. If this timer expires before the error condition is corrected (or goes away) the power to the system is turned off. The timer handling and shutdown are done by the EMM prom code and requires no interactions from the console.

/Refer to the 8600 REGISTER SPEC for a complete description of the passing of EMM priority messages (exceptions) to the CPU through the RX registers./

the console terminal.

After sending a priority message indicating a bad air flow or pod-zone temperature, the EMR starts a 1 to 5 minute timer. If this timer expires before the error condition is corrected (or some event the power to the system is turned off). The timer handling and shutdown are done by the EMR prom code and requires no interaction from the console.

Refer to the 8600 REGISTER SPEC for a complete description of the passing of EMR priority messages (exceptions) to the CPU through the RX registers.

CHAPTER 9

CONSOLE I/O MODE

Separate CIO mode contexts are provided to facilitate the handling of separate, unrelated console tasks. The MACRO context (default on power up) is used for CPU initialization and running VAX MACRO code; MICROHARDCORE context is used to test the hardcore of the machine prior to running any microdiagnostics; DIAGNOSTIC context is used to run microdiagnostic test programs and to isolate machine faults. Switching between contexts affects the state of the machine significantly as local initialization is performed by each context on entry.

In CIO mode operator commands entered from the CTY are always processed, while command input from the RTY is honored only if the front panel "terminal control" switch is in the REMOTE position.

In addition to performing terminal I/O and command processing during CIO mode the console also handles the following items. Detailed descriptions of each item can be found in other sections of this document.

   o handles the reporting of unsolicited warning messages from the EMM.

   o handles the reporting of CPU control store parity errors (no correction).

   o handles front panel switches and LEDs.

When the system is powered up, and after all types of initialization have been performed, the console enters CIO mode (if the front panel switches are the HALT and LOCAL positions). CIO mode can also be entered from PIO mode using the ^P break character, provided the front panel switches allow. CIO mode is also entered whenever attempts to boot or restart the VAX processor fail.

When running in the MACRO context CIO mode can be exited, and control passed to PIO mode operation, with the START or CONTINUE commands. Other than turning off the power or causing a console reboot this is the only way to exit from CIO mode.

9.1   CONSOLE COMMAND SYNTAX

The information in this section applies to all command sets in the console. Command syntax is illustrated using the following conventions:

[ ]   brackets are used to indicate an OPTIONAL switch or keyword

{ }   braces are used to show a list of choices from which one item MUST BE SELECTED

The console parses commands made up of one or more of the following parts.

o   command -- the first item on any command line input

o   switch -- switches begin with the slash "/" character and select some option associated with the command. Switches must appear in proper sequence with other parts of a command line, as shown by the syntax rules describing a command. Switches may be separated from commands, keywords, or other switches by space or tab characters.

o   switch argument -- some switches require associated numerical arguments which are specified as "/switch:number"

o   keyword -- keywords appear to the right of the command part and are arguments to the command. Keywords must be separated from commands, switches, or other keywords with one or more spaces or tabs. Keywords can be numeric values or symbolic names

o   keyword argument -- some keywords require associated numeric or symbolic arguments, such as "keyword:argument"

The following notes describe the main characteristics of the console with respect to command input.

o   A single console command line cannot exceed 80 characters, counting the terminating CR and LF characters. Truncation of excessive input is done and the command line is parsed normally.

o   The DELETE character will rubout the previously typed character.

o   The ^U character will flush the console input buffer and begin a fresh line for command input.

o   The ^R character is ignored from the Local terminal, but from
    the remote terminal it causes the current command line to be
    retyped on the following line.

o   The ^O character acts as a toggle switch that, when set,
    suppresses all output to the issuing terminal (the code
    continues to run). The state of the ^O flag is reset at the
    completion of any command.

o   Upper and lower case input is treated interchangeably.

o   A parser error message results when an unrecognized command
    is entered, or when an invalid switch or keyword is used with
    a command, or when an invalid switch argument or keyword
    argument is used. A parser error also occurs when invalid
    hex numbers are entered, or, in some cases, when the
    allowable range of the numeric input is exceeded. All
    numeric input is taken as hexadecimal, except for the SET
    CLOCK FREQUENCY command (decimal), and numeric output is also
    in hex unless indicated as decimal by either a decimal point
    "." or unit name (e.g., Megacycles, Volts, etc.).

o   An optional radix specifier prefix (%O, %D, %X) may be used
    to override the default radix specified above. i.e. %D100
    implies decimal 100.

o   Abbreviations are formed by dropping characters from the end
    of a command, switch, or keyword. In general, commands may
    be abbreviated to the point where they become no longer
    unique among all other commands of the current context. The
    exception to this is that some commands, like EXAMINE and
    DEPOSIT, are defined architectually as having single (or
    double) character "equivalents". Thus, although the command
    input "E" may not be unique in the current context it is
    treated as an equivalent to "EXAMINE". The complete list of
    such command equivalences is shown below.

o   Keywords and switches may also be abbreviated to the point
    where they become no longer unique amoung other keywords or
    switches for that command.

Table of architectually-defined command and switch equivalences:

Short form	Long form
B	BOOT
C	CONTINUE
D	DEPOSIT
E	EXAMINE
F	FIND
H	HALT
I	INITIALIZE

```
L         LOAD
N         NEXT
S         START
SE        SET
SH        SHOW
U         UNJAM
V         VERIFY
W         WAIT

/P        /PHYSICAL
```

9.1.1   General Command Set

Commands in the "general command set" are available to all CIO mode contexts (MACRO, DIAGNOSE, MHC). It provides commands necessary for changing the current context as well as others for controlling basic console functions.

Also considered part of the general command set are the control characters ^C and ^P. While in CIO mode, these control characters are treated as equivalent and will abort most command lines and all levels of command files.

The following sections describe the console commands, available to all contexts, in full detail.

9.1.2   DEBUG

Command Syntax:

   DEBUG

Description:

This command allows the command set of the HEX debugger to be concatenated to the command set of the MACRO or DIAGNOSTIC contexts. This command is not available under the Microhardcore context. All trace breakpoints presently set are cleared by this command.

Commands available under the HEX command set are described in another section of this chapter.

9.1.3   DIAGNOSE

Command Syntax:

   DIAGNOSE

Description:

This command switches to CIO mode DIAGNOSTIC context. Upon
entry, the diagnostic context performs its own initialization
of the machine, prints its prompt, and waits for operator
input. The commands available in this context are described
in another section.

9.1.4   HELP

Command Syntax:

    HELP
    HELP category
    HELP category topic [subtopic]

Description:

This command provides on-line help on the various console software
commands (and other topics) and their proper usage.

For the novice user, and/or if the category and topic(s) are
unknown, HELP<cr> will display all available help categories and
wait for you to select one. Having done that, all available help
topics in the selected category are displayed and once again you
are asked to select one, whereupon the help text on that topic is
displayed.

The 'subtopic' argument is used to differentiate certain generic
command topics such as SHOW, SET, EXAM, etc, which utilize optional
switch </> or keyword <sp> arguments.

i.e.
HELP HEX EXAM SDB<cr> shows help on the HEX EXAMINE/SDB command.
HELP HEX DEP CSPE<cr> shows help on the HEX DEPOSIT/CSPE command.
HELP DIAG SHO SWI<cr> shows help on DC's SHOW SWITCHES command.
HELP GEN SHO UCOD<cr> shows help on the general SHOW UCODE command.

9.1.5   IF

Command Syntax:

        IF cond [cond ...] cmd_string

        where 'cond' is one or more of 8600, 8650, FPA, NOFPA,
        and 'cmd_string' is any command valid in the current context.

Description:

If condition(s) is(are) TRUE, then execute the command, else don't.

Note that complimentary conditions while valid syntax-wise,
are totally useless otherwise.

The IF command is invalid in the following cases:

       no condition specified.
       no command string specified.
       CPU running (Logical_CCS from VMS).

9.1.6   INITIALIZE (CLOCK, POWER, SDB)

Command Syntax:

    INITIALIZE { /CLOCK, /POWER, /SDB }

Description:

This command, with the switches listed, performs the initialization
of three fundamental system components.

INIT/CLOCK -- Initializes the system clock and clock distribution
logic (10141 reset) and sets the clock parameters to those
saved with the last SET CLOCK DEFAULT command. After a console
reboot, however, the saved parameters are lost and this command
resorts to using the global defaults of NORMAL frequency, FULL
speed. The state of the SOMM enables is also cleared by this
command.

INIT/POWER -- This command performs the initialization of the EMM
and then of the power system. After the completion of this all
voltage regulators are on (normal marginning) and the EMM is busy
monitoring regulator outputs, air flow, and temperature levels.

INIT/POWER/ELEV:n -- An optional form of INIT/POWER provides for
adjusting the EMM yellow and red zone temperature limits to
account for systems installed at sites considerably above sea level.

When /ELEV:n is used, the argument n, is the site elevation in
feet (acceptable range 0 to 10000). Console software will adjust
the T1 thru T4 limits (preserved across console_reboots), and
init the EMM as above. Subsequent INIT/POWER commands will use
the adjusted limits, which are retained as long as the system has
power applied. The EMM control logic may be reset to the default
(non-adjusted) case by INIT/POWER/ELEV:0.

SHOW POWER commands will include any specified
non-zero elevation in the 'miscellaneous' area.

NOTE -- console software rounds /ELEV:n at 500 ft increments, thus 4700 rounds up to 5000, etc.

INIT/SDB -- This command forces all SDB control channels to the state necessary for normal system operation. The following control channels are loaded with data shown:

- o CSB = 0008 -- -FLIP USTK PAR H
- o EDP = 0060 -- -FLIP GPRA H, -FLIP GPRB H
- o EBC = A000 -- set DIAG register to 0
- o EBC = C000 -- set EIS register to 0
- o EBC = E000 -- set control to NOP
- o FBA = 0000 -- normal operation
- o FBM = 0000 -- normal operation
- o IBD = 1000 -- normal operation (Optimize)
- o ICA = 0000 -- normal operation
- o MCC = 0000 -- normal operation
- o VBA = 0801 -- normal SBIA operation

9.1.7 LOAD (control Store)

Command syntax:

LOAD { /switch } [filename[.BPN]]

Where '/switch' can be any one of the following:

/ACCESS	/CONTEXT	/CYCLE
/ECS	/FBACS	/FBMCS
/FDRAM	/ICS	/IDRAM
/MCF	/MCS	

and 'filename' is the name of the .BPN file to load. If not specified the default system microcode file for that control store will be loaded.

Description:

This command loads the specified file (.BPN) from the system disk to the specified control store or ram. Not all control stores can be loaded directly without prior initialization of some kind. The INITIALIZE/MICRO command will perform the necessary initialization and load all control stores with system microcode in the proper sequence.

If the INIT/MICRO command has been issued already, and that command successfully found the KA86n.REV file on the console pack, then system-microcode revisions are checked as they are loaded. A version number mismatch is reported as an -I- (information) message and processing continues.

If the specified control store has already been loaded with data
from the specified (or default) .BPN file, and has not been
modified by a DEPOSIT (control store), DEPOSIT/MARK, or
DEPOSIT/CSPE command, then the load operation terminates without
actually reloading the control store.

Note that this command will complain if attempted while the CPU
clock is running.

The default system microcode filenames are:

```
                        8600            8650 (if different)
                        ------          ------
         ACCESS   -  ACCESS.BPN
         CONTEXT  -     CTX.BPN
         CYCLE    -   CYCLE.BPN
         ECS      -  KA8600.BPN       KA8650.BPN
         FBACS    -   FADD0.BPN        FADD5.BPN
         FBMCS    -    FMUL.BPN
         FDRAM    -   FADD0.BPN        FADD5.BPN
         ICS      -    IBOX.BPN
         IDRAM    -  KA8600.BPN       KA8650.BPN
         MCF      -     MCF.BPN
         MCS      -  UCODE0.BPN       UCODE5.BPN
```

Example:

>>>LOAD/ECS KA8600

9.1.8   LUPC

Command Syntax:

LUPC /switch hex_address

where '/switch' is one of the following

```
         /ECS       load EBOX control store upc
         /ICS       load IBOX control store upc
         /MCS       load MBOX control store upc
         /FBACS     load FBOXA control store upc
         /FBMCS     load FBOXM control store upc
```

Description:

This command forces the specified microsequencer to the address
specified as "hex_address". The command does not perform range
checking on the hex_address.

9.1.9   MACRO

Command Syntax:

MACRO

Description:

This command switches to the CIO mode MACRO context. Upon entry,
the MACRO context performs its own initialization, prints its
prompt, and waits for operator input. The commands available in
this context are described in another section.

At the completion of macro context initialization the console
prompts for command input. No attempt is made to read the front
panel switches and automatically restart or boot the system.

9.1.10   MHC

Command Syntax:

MHC

Description:

This command switches to the CIO mode microhardcore context. Upon
entry, the MHC context performs its own initialization, prints its
prompt, and waits for operator input. The commands available in
this context are described in another section.

9.1.11   ODT

Command Syntax:

ODT

Description:

This command invokes the octal debugging tool. The ODT prompt is a
star (*) character. The command set of this ODT is similar to that
of the standard DEC ODT products. This command cannot be
abbreviated to anything less than "ODT".

9.1.12   PROM

Command Syntax:

PROM [/RT [filename]]

Description:

The PROm command, without the /RT switch, causes console control
to be passed to the console PROM prompt. Before the transition
is done the user is asked to confirm the request. This command
is valid form the Rty and the PROM will continue to service the
Rty device.

The /RT switch is provided for INTERNAL use only and offers an
entry path to the RT monitor. The use of the /RT switch in the
field is NOT recommended and will NOT be supported. The PROM/RT
command is valid from the RTY only if an optional "filename" is
included on the command line. The "filename" specified is the
name of the executable (.sav) file to chain to (possibly another
version of the console program). The Rty is disconnected when RT
is entered but the connection can be re-established provided the
command invokes a usable version of console software.

9.1.13   REBOOT

Command Syntax:

    REBOOT

Description:

This command reboots the console software. The console follows
its normal power up procedure, as outlined in another section, and
attempts to re-enter PIO mode if the REBOOT command was issued
while in the MACRO context, and if the CPU was running at the
time. See also the description of the CONSOLE REBOOT function.

When issued from the DIAG or MH contexts this command will not
attempt to re-enter PIO mode when MACRO context initialization
completes.

Also note that the state of many of the console control flags are
preserved (or unaffected) by the reboot action, while others are
forced to a known state. See the decription of the SET FLAGS
command.

9.1.14   REPEAT

Command Syntax:

    REPEAT [dec_num] command

Where 'dec_num' specifies the number of times the command is to be
repeated. If not specified the command repeats indefinitely, or
until a ^C or ^P is entered.

Description:

This command allows console commands to be repeated. Each iteration causes the prompt and command to be re-displayed. The ^O character can be used to bypass the output, causing the repeat function to run even faster. Repeated commands obey the ABORT (on error) flag.

Commands requiring user input, confirmation, or not considered as 'repeatable' are not allowed as arguments to the REPEAT command. This includes the following commands:

DEPOSIT/MARK	DEPOSIT/CSPE
REPEAT	NEXT
START	CONTINUE
MICROSTEP	STATESTEP
TMICRO	TSTATE
TRACE DEFINE	TRACE DELETE

9.1.15   RESET

Command Syntax:

RESET

Description:

This command performs a CPU logic initialization sequence consisting of the following steps.

1. The CPU clock is stopped and state-stepped to the T3 state.
2. The signal "CL09 CPU RESET H" is asserted.
3. The signal "CL09 HOLD STATE RESET H" is asserted.
4. The CPU clock is burst at a frequency of 60 Mhz for 1024 steps.
5. The CPU clock is forced to the T3 state.
6. The signals "CL09 CPU RESET H" and "CL09 HOLD STATE RESET" are deasserted.

Note that this command does nothing to the state of the SBIA clock, regardless of the presence of the SBIA Visibility Module.

9.1.16   RESTART

Command Syntax:

RESTART

Description:

This command forces a restart of the console program where a CONSOLE PROGRAM INITIALIZATION is performed. When initialization

completes, control is passed to the CIO mode MACRO context where
MACRO CONTEXT INITIALIZATION occurs. Note that this command DOES
NOT cause a new copy of the console program to be loaded from disk
(only the REBOOT command will do that).

9.1.17   SET BASE

Command Syntax:

        SET BASE hex_number

Where 'hex_number' is a 32-bit value to be added to the array
address specified in all subsequent /VIRTUAL and /PHYSICAL commands.

Description:

The BASE value is added to both virtual and physical memory accesses
done using the E/V, D/V, E/P and D/P commands (and also E and D with
no switches).

Use the SHOW FLAGS command to display the current setting of the
BASE value. The base is set to 0 initially and is not reset by
any commands (except REBOOT).

9.1.18   SET Flag ON/OFF

Command Syntax:

        SET flag { ON, OFF } { INVALID, /NOVERIFY, NOW }

Where 'flag' can be any one of the following keywords:

ABORT	ABUS	BBU
COLD	EXTI	FBOX
IOSAFE	LOCAL-COPY	MEMENA
SNAP	STXALT	WARM
QUIET		

Description:

These commands control the setting of software and hardware flags
in the console subsystem. Setting the COLD, WARM, FBOX, and STXALT
flags are applicable to the MACRO context only and result in no
immediate action by the console program. The ABORT and QUIET flags
control the handling of errors and command-line echoing during a
command file run, respectively. The EXTI (external interrupt) flag
and the ABUS flag directly control hardware signals on the console
module. Use the SHOW FLAG command to examine the state of all flags.
The INVALID, /NOVERIFY, and NOW options apply to the SNAP flag only

(see SET SNAP below).

The state of many of these flags is restored after a console reboot,
or are simply unaffected by the reboot (those that are hardware
register bits). The following table summarizes the affect of a
console reboot on the control flags.

Flag	State forced by console reboot
ABORT	ON
COLD	OFF
FBOX	OFF
WARM	OFF
ABUS	Unaffected
BBU	Unaffected
EXTI	Unaffected
MEMENA	Unaffected
IOSAFE	Unaffected (Restored to original state)
LOCAL-COPY	Unaffected (Restored to original state)
SNAP	Unaffected (Restored to original state)
STXALT	Unaffected (Restored to original state)
QUIET	Unaffected (Restored to original state)

ABORT -- This flag is set ON during console program initialization
or whenever the console is rebooted.

The ABORT flag provides a limited control over the error handling
within a command file. When ABORT is set ON command file errors
result in an abort of all command file levels. When OFF command
file errors are reported and processing continues.

Note that REPEATed commands will also obey this flag.

ABUS -- This flag is set OFF during console program initialization
and is unaffected by console reboots.

The ABUS flag controls the console signal "CL09 ABUS ENABLE".
Setting this flag ON enables the ABUS. The transition from the
OFF state to the ON state causes an ABUS INIT pulse to be
generated in the SBI's. The INIT/PAMM command automatically
generates the ABUS INIT sequence and leaves the ABUS enabled.

BBU -- This flag is set ON during console program initialization
and by the INIT/POWER command. It is unaffected by console
reboots.

When BBU is ON, the Battery Backup Unit is enabled (but not
draining) and will provide backup power to the system in the event
of an AC power failure. When BBU is OFF the Backup Unit is
disabled and will not provide backup power during the next AC
power failure.

COLD -- This flag is set ON internally by the console program when

a CPU bootstrap is being attempted. The CPU sets this console-resident flag OFF when a bootstrap completes with success.

The purpose of the COLD flag is to inhibit repeated attempts at unsuccessful CPU bootstrapping. This is most useful when a KAF triggers a CPU bootstrap while a bootstrap was already in progress.

EXTI -- This flag is set ON by the INIT/CPU and remains in this state during PIO mode operation of the console. In CIO mode this flag is turned off automatically if any of three "unexpected" CBUS interrupts occur. The state of this flag is unaffected by console reboots.

This flag controls the enabling of external interrupts coming onto the console module. When ON, the console will receive interrupts from the EBE CPU ERR inputs (CS PARITY ERRORS), the ABUS DEAD inputs, and from the EMM CPU AC LO input.

FBOX -- This flag is set OFF during console program initialization or by a console reboot.

This flag does not directly control the enabling or disabling of the Floating Point Accelerator module. Its purpose is to control the action of the INIT command regarding the enabling or disabling of the Fbox. If set ON, the Fbox (if present) will be enabled by the INIT, INIT/CPU, or INIT/PAMM commands.

IOSAFE -- This flag is set OFF during console program initialization, but its state is preserved through console reboots.

IOSAFE is a flag that controls a software write-protect feature for STX (console storage device) transfers from the CPU. When OFF, the CPU can modify blocks on the console pack; when ON, the STX WRITE operations are essentially NOP'ed. All handshaking occurs when IOSAFE is ON, but the pack is not modified.

LOCAL-COPY -- This flag is set OFF during console program initialization, but its state is preserved through console reboots. This flag cannot be set OFF by the remote user (through the RTY).

When ON, it enables the use of the CTY as a hardcopy output device to which all RTY activity is logged (regardless of the position of the front panel "REMOTE" switch).

MEMENA -- This flag is set ON by the console software during CPU initialization and normally remains on until a power failure or system shutdown occurs. Microdiagnostics may manipulate this bit also.

The MEMENA flag controls the state of the console signal "CL09 MEM BUS ENABLE", which when set OFF disables write

access to the arrays and switches all array modules to their
internal refresh state.

SNAP -- This flag is set OFF during console program initialization,
but its state is preserved through console reboots.

When SNAP is ON, the console will perform a CPU snapshot operation
whenever the CPU stops executing macro instructions. DEC STD 032,
chapter 11, lists the architectually defined conditions for which
the console will perform the snapshot (e.g., Invalid Interrupt
Stack). By default, SNAP is OFF and snapshots are inhibited.

SET SNAP ON/NOVERIFY -- may be used to disable control store and
pamm verification during the snap-shot thereby providing a 'quick'
variation of the snap-shot procedure.

SET SNAP INVALID -- may used to invalidate both snap files on
the system disk. Use of this command does not change the
ON/OFF state of the snap flag.

SET SNAP NOW -- may be used to force the creation of a snap file
(/NOVERIFY) based on current machine state. If two valid files
already exist, this command will flush the oldest (SNAP1),
rename SNAP2 to SNAP1, and create a new SNAP2.

QUIET -- This flag is set ON during console program initialization,
but its state is preserved through console reboots.

The QUIET flag provides control over the command-echoing feature
of the command file processor (see "@" command), and also control
over output generated by other commands (see "VERIFY", "TMICRO",
"TSTATE" commands). When QUIET is set ON all input taken from a
command file is suppressed; only output generated by the commands
is displayed. When set OFF commands taken from the command file
are echoed before being executed.

STXALT -- This flag is set OFF during console program
initialization, but its state is preserved through console reboots.

The purpose of the STXALT flag is to indicate that disk IO at the
request of the CPU (via the STX registers) is to be on the
alternate console disk (unit #1). This flag is intended for use by
manufacturing and NOT FOR CUSTOMER USE.

When STXALT is set ON all console disk transfers continue to
access RL unit 0; all STX transfers access unit 1.

WARM -- This flag is set ON internally by the console program when a
CPU restart is being attempted. The CPU sets this console-resident
flag OFF when the restart completes with success.

The purpose of the WARM flag is to inhibit repeated attempts at
unsuccessful CPU restarts. This flag would only be used if, during
a restart attempt, a power "brown-out" occurred. If a full power
black-out occurs the console looses the state of the WARM flag.

9.1.19   SET CLOCK

***
***   Command Syntax for rev E01 and above clock modules:
***

     SET CLOCK Xn dec_num [{ /NORMAL, /HIGH }]

     SET CLOCK FREQUENCY { NORMAL, HIGH, X1, X2, X3, X4, X5, X6, EXTERNAL }

     SET CLOCK { FULL, ONE-FIFTH, DEFAULT }

     Where 'dec_num' is the clock frequency in Mega-cycles (Mhz) and
     is in the range of 40. to 90.
     Note that frequencies above 65. are invalid on 8600 processors.

       Description:

       The command "SET CLOCK Xn dec_num" assigns the value of "dec_num"
       to the specified crystal mnemonic (X1, X2, ...) where the mnemonic
       can then be used to SET the CLOCK FREQUENCY. This isolates the
       console software from the actual values assigned to the crystals
       on the clock module. Up to 6 crystal values can exist, and any
       one of them can be assigned to be the "NORMAL" or "HIGH" clock
       values. These assignments should normally be done once in the
       CLOCK.COM file. Note that Xn assignment information is preserved
       through console reboots.

       The command "SET CLOCK FREQUENCY" can be used to select one of the
       previously assigned crystals (X1, X2, ...) as the base frequency of
       the system. Alternately, the keywords NORMAL or HIGH can be used to
       select the nominal and high margin clock values. EXTERNAL can be
       used to specify that cpu clock source should be taken from the
       external source applied to the appropriate clock module backplane pin.

       The remaining SET CLOCK command has three keyword options to select
       FULL CPU execution speed, or ONE-FIFTH CPU execution speed (1/5th of
       the base frequency). The DEFAULT option forces the console to save
       the current base frequency and FULL/FIFTH information where it will
       be used by subsequent INIT/CLOCK commands as the default setting to
       which the clock should be set.

***
***   Command Syntax for rev C05 clock modules:

***

SET CLOCK FREQUENCY { dec_num, NORMAL, QUIET }

SET CLOCK { FULL, ONE-FIFTH, DEFAULT }

Where 'dec_num' is the clock frequency in Mega-cycles (Mhz) and
is in the range of 40 to 64.

Description:

The SET CLOCK FREQUENCY command(s) provides selection of the CPU
clock frequency source, which can be a fixed-output crystal value
or the variable control oscillator (VCO) output. The NORMAL
argument selects the stable crystal output, while a numeric
argument selects a VCO output. The frequency setting is affected
by the INIT/CLOCK command. The QUIET arguement can be used to
silence the warning messages that may occur at some VCO settings.

The SET CLOCK command recognizes 3 arguements to control the gross
output speed of the CPU clock and also forces the DEFAULT clock
settings to become the present values (used by INIT/CLOCK).

9.1.20   SET SOMM

Command Syntax:

    SET SOMM [/switches] { ON OFF }

Where 'switches' can be any combination of the following items. If
no switch is specified the default of /ECS is assumed when turning
SOMM ON. If no switch is specified when turning SOMM OFF, then
ALL SOMM ENABLES ARE TURNED OFF.

            /ECS   /ICS   /MCS   /FIELD

SOMM is an acronym for Stop On Micro Mark.

Description:

This command controls the enabling and disabling of micro-mark
breakpoint detection in the CPU clock module. When enabled for one
or more control stores the CPU clocks are stopped in the T0 state
when a mark breakpoint is encountered. This condition is detected
by the console program, which in turn forces the clocks to the T3
state while collecting micro-upc data for the Ebox, Mbox, and Ibox.
The CPU clock is stopped. See the DEPOSIT/MARK command for a
description of how mark breakpoints can be set or cleared.

When the micro breakpoint occurs the current UPC of the Ebox, Mbox
and Ibox microsequencers is displayed. When running with the clock

in 1/5 speed the UPC data will match the micro address of the
location containing the micro breakpoint and the microsequencer for
the stopped box will have already executed that micro instruction.
Under FULL speed operation the UPC data displayed will be 2 full
microcycles past the breakpointed instruction. This is due to
propagation delays in various signals between the clock and CPU
modules. Also, in FULL speed micro breakpoints set on adjacent
micro instructions will overrun the second in the sequence (it will
never be "encountered").

All SOMM flags are initialized by the console program to the OFF
state during CONSOLE PROGRAM INITIALIZATION and by the INIT/CLOCK
command.

The /FIELD switch enables detection of an externally driven
clock-module input.

Example:

>>>SET SOMM/ECS/MCS ON   !Enable ECS and MCS bpts

>>>SET SOMM/ICS ON   !Included ICS bpts now

>>>SET SOMM/MCS OFF   !Disable only MCS bpts

>>>SET SOMM OFF   !Disable ALL bpts

9.1.21   SET TERMINAL

Command Syntax:

    SET TERMINAL /switches

where '/switches' can be any of the following switches, combined
in any way. If the /PASSWORD switch is used it must be the last
switch on the line.

The defaults shown are the ones set during console software
initialization.

	Initially set to:
/BAUD:nnnn	As set in STARTF.COM file
/RECEIVE:nnnn	1200 baud
/TRANSMIT:nnnn	1200 baud
/PASSWORD [password]	no password defined
/[NO]SCOPE	SCOPE
/[NO]PARITY	NOPARITY
/[NO]DSRS	NODSRS
/ODD	Default when PARITY is enabled
/EVEN	

where 'nnnn' can be 50, 75, 110, 134, 150, 300, 600, 1200,
1800, 2000, 2400, 3600, 4800, 7200, 9600, or 19200

Description:

This command sets terminal port (PCI) characteristics for the CTY
or RTY interfaces. Note that while in CIO mode the most
significant bit of characters received from the CTY or RTY ports
is stripped off (except for the X command). Full 8-bit character
support is handled while in PIO mode. The default characteristics
for the remote PCI are 1200/ 1200 baud, 1 stop bit, no parity, no
DSRS (data set rate select), SCOPE, and no password.

The /BAUD switch always applies to the CTY port, while all others
apply to the RTY port.

Only the /[NO]SCOPE switch is allowed to be entered by the RTY user.
All others switches must be issued from the CTY.

CTY control of the [NO]SCOPE flag can only be done by modifying the
STARTF.COM command file on the console pack and rebooting the
console so that RT executes the command file. By default the CTY
will be treated as a NOSCOPE (hardcopy) device.

All parameters set by this command are strored by the console and
recalled automatically after a console reboot.

/BAUD:nnnn -- sets PCI transmitter and receiver baud rate for the
CTY port

/RECEIVE:nnnn -- sets PCI receive baud rate for the RTY port

/TRANSMIT:nnnn -- sets PCI transmit baud rate for the RTY port

/[NO]PARITY -- enables or disables parity generation on characters
transmitted from the RTY port

/EVEN
/ODD -- selects parity type for RTY port transmissions and
automatically turns parity generation/detection on

/[NO]DSRS -- DATA SET RATE SELECT is an output signal from the
console located at the RTY port connect, PIN 23. It may be used
with some MODEMs for selection of low speed/high speed operation.

/[NO]SCOPE -- specifies the type of terminal device connected to the
RD port so that rubout characters are handled correctly

/PASSWORD [password] -- sets the login password for RTY port
access. The password, if specified, must be 1 to 6 characters
in length and must contain only alpha-numeric characters (0-9, A-Z,
Lower case characters are converted to upper case). If no
password is specified the RTY port password feature is disabled
and RTY access to the CPU or console is controlled entirely from the

setting of the front panel Terminal Control switch.

9.1.22   SHOW CLOCK

Command Syntax:

    SHOW CLOCK

Description:

This command displays the current state of the CPU (and system)
clock. The information in the display consists of the following
components.

o   whether or not the system clock is running
o   whether or not the system clock is locked on the current
    frequency
o   whether or not the CPU clock is running
o   the currently selected frequency of the CPU clock source
o   if the CPU clock is stopped, the quarter phase it is stopped at
o   whether or not a MARK STOP condition is presently detected by
    the clock module

If the rev E01 or higher clock module is present the following
information is displayed:

o   assignment values of Xn mnemonics
o   assignment of NORMAL and HIGH keywords

9.1.23   SHOW File

Command Syntax:

    SHOW filename[.DAT] [{ /ASCII, /BINARY }]

Description:

This command locates the specified file on the console RL02 and
displays the contents of that file on the console terminal in
a binary dump format (by default) or in a straight ASCII format
if the /ASCII switch is used. The /BINARY switch can only be
issued from the Rty with the remote protocol running. It is meant
as a means for transferring binary files to a remote site.

The primary purpose of this command is to allow the operator to
display the contents of binary-formatted CPU snapshot files
(SNAP1.DAT and SNAP2.DAT). It can be used, however, to display
the contents of any file on the RL02.

9.1.24   SHOW FLAGS

Command Syntax:

SHOW FLAGS

Description:

This command displays the current state of the console program
control flags as controlled by the SET flag ON!OFF command.

9.1.25   SHOW PANEL

Command Syntax:

SHOW PANEL [/TEST]

Description:

This command displays the setting of the two front panel switches
and the four indicator LEDs. The /TEST switch causes the command
to loop while shifting a floating 1 pattern through the LEDs, and
reading each pattern back to verify the LED-read logic. The
terminal display is updated to show the current state of the front
panel during the test. Changes in the setting of either front panel
switch is also included in the updated display.

The /TEST loop will only execute once when invoked within a command
file.

9.1.26   SHOW POWER

Command Syntax:

SHOW POWER

Description:

This command displays the current status of the EMM, of the power
system, and of the CPU cabinet environment. The information in the
report includes:

o   all DC regulator voltage measurements (volts)
o   status of regulator margin levels
o   all four thermistor temperature measurements (degrees Celsius)
o   ground current monitor level (milliamps)
o   status of air flow sensors

o site elevation (if set non-zero via INIT/POWER/ELEV:n)
o status of BBU availability
o status of latch AC LO and DC LO signals
o status of automatic shutdown timer
o status of default mode enable bit
o status of EMM's 5.5 interrupt disable bit

The accuracy of voltage and temperature values displayed by this
command are accurate to within +/- one least significant digit
of the actual measurement made by the EMM.

9.1.27   SHOW TERMINAL

Command Syntax:

    SHOW TERMINAL

Description:

This command displays various state information concerning the CTY
and RTY ports. The information displayed includes the
following:

o state of the RTY port RTS/DTR, DSR and CD signals
o remote port status (connected, disconnected, etc.)
o state of RTY port [NO]SCOPE flag
o state of RTY port [NO]PARITY flag and ODD or EVEN parity
o Rty port transmit and receive baud rates
o password enabled/disabled status (not the password itself)
o indication of remote protocol/transparent mode active
o remote protocol transmission statistics

9.1.28   SHOW UCODE

Command Syntax:

    SHOW UCODE

Description:

This command displays the state of all control_stores and rams in
terms of:

o the current (last loaded) .BPN file name.
o the default .BPN file name for that control store.
o the ucode revision of the currently loaded microcode.
o the ram status: bit 15 -- 1 = modified
                     bit 14 -- 1 = failed verification
                     bit 13:00 -- parity error count

The "modified" bit, when set, indicates that one or more locations have been deposited or otherwise altered after the last load of that control store. Note that this bit is always set for the E_SCratch ram.

The "failed verification" bit, when set, indicates that the VERIFY command detected one or more discrepancies in the control store data. This could be the result of a multiple (undetected) control store parity error.

The "Parity error count" field contains the total number of detected parity errors for the control store, whether they were corrected or not. MCF, CTX, CYC, and ACC rams always show 0's in this field.

9.1.29   SHOW VERSION

Command Syntax:

    SHOW VERSION

Description:

This command displays revision information for the following items:

o   the console program major revision
o   the CPU hardware revision
o   the major revision of the system microcode
o   the revision of the console PROM code
o   the revision of the EMM prom code
o   various addresses of interest to developers and debuggers

9.1.30   START/STOP CPU

Command Syntax:

    START CPU-CLOCK

    STOP CPU-CLOCK

Description:

These two commands provide on/off control of the CPU clock. If the SYSTEM clock is off, then the START CPU-CLOCK command will start it before starting the CPU clock. The STOP CPU-CLOCK command stops the CPU clock but does not affect the system clock.

These commands have no affect on the state of the SBIA Visibility module and its internal clock.

9.1.31   START/STOP SYSTEM

    Command Syntax:

        START SYSTEM-CLOCK

        STOP SYSTEM-CLOCK

    Description:

    These two commands provide on/off control of the SYSTEM clock. If
    the CPU clock is on, then the STOP SYSTEM-CLOCK command will stop
    it before stopping the system clock. The START SYSTEM-CLOCK
    command starts the system clock running at the frequency last set
    by the SET CLOCK FREQUENCY command.

    These commands have no affect on the state of the SBIA Visibility
    module and its internal clock.

9.1.32   UNHANG

    Command Syntax:

        UNHANG

    Description:

    This command performs an Unhang Reset sequence to the CPU by first
    stopping the CPU clocks, asserting the console signal CL09 HOLD
    STATE RESET, bursting the CPU clock 1024 ticks, deasserting HOLD
    STATE RESET, loading the Ebox micro-pc to 100D(16) and starting
    the CPU clocks. This reset sequence does not clear latched error
    status in the machine as master RESET does, nor does it affect the
    system cache.

9.1.33   VTERM

    Command Syntax:

    VTERM { Symbol_name, Hex_id }

    Where 'Symbol_name' is the V$ symbol assigned to the particular
            visibility signal by the SDB CAD (.CDF) files.

        'Hex_id' is a 16-bit SDB ID. SDB_ID's are hardware
            visibility-bit addresses and are defined in the .CDF files.

    Description:

    This command provides a fundamental aid for isolating faults in
    Vterm (visibility terminator) and visibility logic. The command

accepts a V$ symbol or SDB hex_id and sets up the visibility
control logic to select the specified bit. The visibility bit is
left selected so that static measurements can be made. Only a
console RESTART or REBOOT will reset the state of the SDB control
logic, or another VTERM command.

9.1.34 WAIT

Command Syntax:

WAIT [hex_count]

Where 'hex_count' is a number between 0 and FF. If not specified
        the default of 1 is assumed.

Description:

This command will cause the console to stop processing command, and
most other chores, for the specified number of 20 millisecond
intervals. The count of 0 returns immediately, while the count of
FF pauses for approximately 5.12 seconds before control is returned
to the terminal.

This command does not have the same function as the same 11/780
console command does. On the 8600 console this command is used
in command files to introduce delays.

9.1.35 X

Command Syntax:

X hex_adr hex_count

Where 'hex_adr' is the address of main memory where data is to be
        transferred to/from.

And where 'hex_count' is the number of bytes to be transferred.

Note that 'hex_count' is interpreted as a 32 bit integer where
bit31 implies the direction for the data transfer.
i.e. X 100 FF will 'load' FF bytes into main memory at loc 100.
    X 100 800000FF will 'unload' from main memory at loc 100.

Description:

This command is for automatic communication between the console
and other systems. It is NOT INTENDED FOR USE BY OPERATORS. It
is used to load (write) and unload (read) the contents of main
memory.

Data following this command must be the checksum byte for the command line itself, in binary. Up to 4kb of binary data follows the checksum, according to the number specified as "hex_count". The data is buffered by the console in T11 ram until the transfer is complete. Then the buffer is transferred to physical CPU memory beginning at the addresses specified as "hex_adr".

Refer to DEC STD 032, chapter 11, for a complete description of the X command protocol.

9.1.36 @

Command Syntax:

      @filename[.COM]

Description:

The @ (at) command causes command input to be taken from the file indicated by 'filename'. As commands are read from the file they may or may not be echoed on the command terminal device depending on the state of the QUIET flag. Output generated from commands in the command file are always displayed.

Command files may be nested up to four levels deep, but at that level no command can be executed that specifies access to a file on the disk (such as "LOAD filename").

The setting of the ABORT flag controls whether or not a command file is aborted when an error (or warning) is detected. A command file will normally run until one of the following events occur:

    o end of file is reached (at level 0)

    o a control_C is entered

    o execution of a START command

    o execution of a CONTINUE command

    o a warning or error is detected and the ABORT flag is set

The default filename extension is .COM, but may be overridden by specifying another.

Comments can be included in a command file by preceding all data on the line with a ! (exclamation mark).

Commands that require confirmation or "separate line" input can be used in command files if the required input is included following the command line.

9.2 MACRO CONTEXT

The MACRO context provides console commands to initialize the machine as a VAX processor. It also allows access to PIO mode operation of the console, which allows VAX macrocode to be run while the console acts as a interface for both console terminals and the EMM.

9.2.1 Context Initialization

This is the context which is entered initially by the console program on power up or whenever the MACRO command is entered. Whenever this context is entered a MACRO CONTEXT INITIALIZATION is performed, as detailed in the "CONSOLE PROGRAM INITIALIZATION" section.

9.2.2 Console Support Microcode (CSM)

Many of the functions of the MACRO context depend on Console Support Microcode (CSM). CSM is a microcode package which allows the console program to communicate with the CPU by means of data packets passed through the CBUS. Refer to the CSM FUNCTIONAL SPECIFICATION for details regarding the operation of CSM.

CSM is divided into microcode overlays which are managed by the console program. Depending on the type of packet function being used the console loads and starts the correct microcode overlay. To improve performance of the overlay loading all CSM overlays are buffered in T-11 memory after the first load from the console pack.

9.2.3 MACRO Context Command Set

9.2.4 BOOT

Command Syntax:

BOOT [/switches] [device]

Where 'switches' can be any combination of the following:

        /R5:hex_num - specify value to place in R5 (0 = default)
        /NOSTART   - specify that the boot command file not start
                      the operating system at its completion.

and where 'device' can is a one-to-three character device mnemonic,
        such as DU0 or CS1, that is appended automatically with
        a "BOO.COM" suffix. The resulting filename is used to
        locate the xxxBOO.COM file to boot the operating system.

If 'device' is more than 3 characters then it is taken as
a filename with the default extention of .COM. This file
is then located and used to boot the operating system. If
'device' is not specified the file DEFBOO.COM is used.

Description:

This command handles the booting of timesharing code into the
VAX processor, as outlined in DEC STD 032. The boot operation is
attempted regardless of the present setting of the COLD flag.

. Before the specified command file is invoked the console loads
the CPU's R5 register with the data specified on the /R5 switch,
or with zero if no switch was used. It also sets the COLD flag
ON, to indicate that a bootstrap attempt is being made. The last
known PC, PSL and HALT code are loaded into the CPU's R10, R11,
and R12 (AP) respectively.

A typical boot command file contains instructions to INIT the
CPU, turn off the cache, search for a good section of main memory,
initialize gpr's R0 through R4, load a bootstrap program (e.g.,
VMB) into main memory, and start the bootstrap program (if the
/nostart switch is not used).

Note that BOOT command files, such as DEFBOO, should not contain
commands that unintentionally modify the state of R5, R10, R11, AP,
or SP. Commands that initialize the Ebox Scratch pad (INIT/ESC,
INIT/CPU) will alter the state of all the above mentioned GPR's.

Example:

>>>BOOT   - loads R5 with 0 and invokes DEFBOO.COM

>>>BOOT/R5:7 DU0   - loads R5 with 7 and invokes DU0BOO.COM

>>>BOOT/R5:1/NOSTART   - loads R5, runs DEFBOO.COM, waits for input

>>>BOOT CIGEN   - invokes CIGEN.COM

9.2.5 CONTINUE

Command Syntax:

    CONTINUE

Description:

This command has two different effects depending on the state of
the CPU. If the CPU is running (executing macro instructions) it
causes a transition from CIO mode to PIO mode. This situation
occurs when a running system is interrupted from the console
terminal by the ^P break character.

If the CPU is not running then the CONTINUE command performs an IBUFF FLUSH and forces instruction execution to begin from the current PC.

9.2.6   CLEAR MEMORY

Command Syntax:

    CLEAR [MEMORY]

Description:

This command clears the first block of contiguous array memory, as determined by the contents of the PAMM. The INIT/PAMM command can be used to initialize the PAMM to the correct state before issuing this command.

This command performs an implicit INITIALIZE operation, to get the CPU to a known state, before attempting to clear memory. It assumes the system microcode has already been loaded.

9.2.7   DEPOSIT (CPU Or T11 Address Spaces)

Command Syntax:

DEPOSIT [/space] [/NEXT:hex_num] [/data_type]
                              { hex_addr, reg_name } hex_data

Where '/space' can be any ONE of the following items. If none specified then the default of /PHYSICAL is assumed.

         /ESCRATCH - access Ebox scratch pad RAM space
         /GENERAL   - access VAX processor GPR register space
         /INTERNAL - access VAX processor IPR register space
         /PAMM   - access PAMM RAM
         /PHYSICAL - access VAX physical memory space
         /U   - access T-11 RAM address space
         /VIRTUAL   - access VAX virtual memory space if the map
                      is enabled, otherwise same as /PHYSICAL but
                      bits 30 and 31 of address are ignored

   and where '/data_type' can be any ONE of the following items if
         the /space access is for /PHYSICAL, /VIRTUAL, or /U.
         The default here is /LONG.

         /BYTE   - access a single byte of data
         /LONG   - access a longword (4 bytes) of data
         /WORD   - access a word (2 bytes) of data

   and where 'hex_addr' is the numeric address of the register or

address being examined. Optionally, any of the following
positional operators can be used to represent a 'hex_addr'
relative to the last address accessed. Note that the
default address is set to 0 by the INIT command.

*   (Access last specified address)
+   (Access address following last (*) address)
-           (Access address preceding last (*) address)
@   (Use contents of last specified address as the
            address to be accessed)

and where 'reg_name' is a valid register name for an IPR, GPR,
       Internal Register (IR), or Miscellaneous register. See
       the sections following this command description for a
       complete list of supported register names.

and where 'hex_data' is the hexadecimal data to be deposited into
       the specified address space.

Description:

This command allows data to be deposited into various address
spaces in the CPU and console, but not into any control stores.
Data can be anywhere from 1 to 4 bytes in length, and using the
/NEXT switch successive locations can be deposited with the
same data.

Deposits to all address spaces, except for T11 memory, require
that the Ebox be loaded with system microcode (KA86n0.BPN), that
CSM be running in the Ebox, and that the Ebox scratch pad RAM be
initialized. In addition, the PAMM must be initialized by the
INIT/PAMM command before access to main memory can be made.

Address and data size checking is performed by this command.

Example:

>>>>D/G/N:3 0 0   ; Clear R0, R1, R2, R3

>>>>D/P/B 1F FF   ; Write FF to byte location 1F

>>>>D/PAMM/n:FF 20 1F   ; Write to PAMM locations

9.2.8 EXAMINE (CPU And T11 Address Spaces)

Command Syntax:

EXAMINE [/space] [/NEXT:hex_num] [/data_type]
                                    { [hex_addr, reg_name] }

Where '/space' can be any ONE of the following items. If none
       specified then the default of /PHYSICAL is assumed.

```
/ESCRATCH  - access Ebox scratch pad RAM space
/GENERAL   - access VAX processor GPR register space
/INTERNAL  - access VAX processor IPR register space
/PAMM      - access PAMM RAM
/PHYSICAL  - access VAX physical memory space
/U         - access T-11 RAM address space
/VIRTUAL   - access VAX virtual memory space if the map
             is enabled, otherwise same as /PHYSICAL but
             bits 30 and 31 of address are ignored
```

and where '/data_type' can be any ONE of the following items if
the /space access is for /PHYSICAL, /VIRTUAL, or /U.
The default here is /LONG.

```
/BYTE   - access a single byte of data
/LONG   - access a longword (4 bytes) of data
/WORD   - access a word (2 bytes) of data
```

and where 'hex_addr' is the numeric address of the register or
address being examined. Optionally, any of the following
positional operators can be used to represent a 'hex_addr'
relative to the last address accessed. Note that the
default address is set to 0 by the INIT command.

```
*       (Access last specified address)
+       (Access address following last (*) address)
-       (Access address preceding last (*) address)
@       (Use contents of last specified address as the
         address to be accessed)
```

If 'hex_addr' is unspecified, + is assumed.

and where 'reg_name' is a valid register name for an IPR, GPR,
Internal Register (IR), or Miscellaneous register. See
the sections following this command description for a
complete list of supported register names.

Description:

This command allows several address spaces to be examined.
Access can be made either using a numeric address or, if
available, a symbolic register name. Where /BYTE or /WORD switches
are used the value is displayed as a zero extended long word,
but the access in the machine is as specified in the switch.

Examines to all address spaces, except for Tll memory, require
that the Ebox be loaded with system microcode (KA86n0.BPN), that
CSM be running in the Ebox, and that the Ebox scratch pad RAM be
initialized. In addition, the PAMM must be initialized by the
INIT/PAMM before access to main memory can be made.

Information is displayed in the following formats:

/ESC

```
              E HH HHHHHHHH

   /GENERAL and named GPR's

              G HH HHHHHHHH

   /INTERNAL and named IPR's

              I HH HHHHHHHH

   /PAMM

          PAMM HHHHHHHH HHHHHHHH

   /PHYSICAL and /VIRTUAL

          P HHHHHHHH HHHHHHHH

   /U
          U OOOOOO OOOOOO          (numbers are OCTAL)

Named but not addressed internal registers

          MICRO REG: HHHHHHHH

Named but not addressed miscellaneous registers

          U reg_name HHHHHHHH
```

Example:

```
   >>>>E/I 0                      ; Examine IPR 0 (KSP)

   >>>>E/NEXT:10 KSP              ; Examine IPR KSP through SISR

   >>>>E/BYTE/U/NEXT:5 12         ; Examine T11 memory bytes
```

9.2.8.1 Supported GPR Names

    The following table lists the supported General Purpose Register
(GPR) names known by the MACRO context EXAMINE and DEPOSIT commands.
When any of these names are used in an examine or deposit command no
'/datatype' or '/space' switches are allowed, as the defaults are
assumed to be /LONG/GENERAL.

Name	Access	/G Address	Purpose
R0	E/D	00	Processor Register 0
R1	E/D	01	Processor Register 1
R2	E/D	02	Processor Register 2
R3	E/D	03	Processor Register 3
R4	E/D	04	Processor Register 4

```
R5      E/D      05              Processor Register 5
R6      E/D      06              Processor Register 6
R7      E/D      07              Processor Register 7
R8      E/D      08              Processor Register 8
R9      E/D      09              Processor Register 9
R10     E/D      0A              Processor Register 10
R11     E/D      0B              Processor Register 11
AP      E/D      0C              Processor Argument Pointer
FP      E/D      0D              Processor Frame Pointer
SP      E/D      0E              Processor Stack Pointer
PC      E/D      0F              Processor PC
```

9.2.8.2 Supported IPR Names

The following table lists the supported Internal Privileged Register (IPR) names known by the MACRO context EXAMINE and DEPOSIT commands. When any of these names are used in an examine or deposit command no '/datatype' or '/space' switches are allowed, as the defaults are assumed to be /LONG/INTERNAL.

Those addresses which are new for THE 8600 have a (*) beside them. Those registers which have the same function, but a different design from other VAX's are tagged with a (~).

```
Name    Access   /I Address     Purpose
----    ------   ----------     -------
KSP     E/D      00             Kernel Stack Pointer
ESP     E/D      01             Exec Stack Pointer
SSP     E/D      02             Supervisor Stack Pointer
USP     E/D      03             User Stack Pointer
ISP     E/D      04             Interrupt Stack Pointer
P0BR    E/D      08             P0 Base register
P0LR    E/D      09             P0 Length Register
P1BR    E/D      0A             P1 Base Register
P1LR    E/D      0B             P1 Length Register
SBR     E/D      0C             System Base Register
SLR     E/D      0D             System limit Register
PCBB    E/D      10             Process Control Block Base
SCBB    E/D      11             System Control Block Base
IPL     E/D      12             Interrupt Priority Level
ASTLVL  E/D      13             AST Level
SIRR     D       14             Software Interrupt Request Register
SISR    E/D      15             Software Interrupt Summary Register
ICCS    E/D      18             Interval Clock Control
NICR     D       19             Next Interval Count Register
ICR      E       1A             Interval Count Register
TODR    E/D      1B             Time of Day Register
~RXCS   E/D      20             Receive transfer control status
~RXDB    E       21             Receive transfer Data Buffer
~TXCS   E/D      22             Transmit transfer control status
~TXDB    D       23             Transmit transfer Data Buffer
ACCS    E/D      28             Accelerator Status Register
```

```
     MAPEN   E/D      38                Memory Management enable
     TBIA    D        39                Translation Buffer Invalid All
     TBIS    D        3A                Translation Buffer Invalid Single
     PME     E/D      3D                Performance Monitor Enable
     PMR     E/D      3D                Performance Monitor Register
                                          (same as PME)
     SID     E        3E                System ID
    *PAMACC  E/D      40                Physical Array Map access
    *PAMLOC  E/D      41                Physical Array Map Location
    *CSWP    E/D      42                Cache Sweep
    *MDECC   E/D      43                Mbox data ECC
    *MENA    E/D      44                Mbox Error Enable
    *MDCTL   E/D      45                Mbox Data Control
    *MCCTL   E/D      46                Mbox MCC Ctrl
    *MERG    E/D      47                Mbox Error generation
    *CRBT    D        48                Console Reboot
    *DFI     D        49                Diagnostic Fault Insertion Register
    *EHSR    E/D      4A                Error Handling Status Register
    *STXCS   E/D      4C                Storage Transfer Exchange Control Status
    *STXDB   E/D      4D                Storage Transfer Exchange Data Buffer
```

9.2.8.3 Supported IR Names

The following table lists the supported Internal Register (IR) names known by the MACRO context EXAMINE and DEPOSIT commands. When any of these names are used in an examine or deposit command no '/datatype' switch is allowed, as the default data type is assumed to be /LONG.

Internal Registers are not numerically addressable and can be accessed only by using the specific register names given below.

```
    Name     Access   Purpose
    ----     ------   -------
    CPC      E        Ibox Current PC register
    CSHCTL   E/D      Mbox Cache Control register
    CSES     E        Control Store Error Status register
    CSLINT   E/D      Console Interrupt status register (Ebox)
    EBCS     E/D      Ebox Control Status register
    EDMC     D        Ebox Diagnostic Maintenance Control register
    EDPSR    E        Ebox Data path status register
    EMD      E        Ibox EMD register
    ESASAV   E        Ibox ESASAV register
    IBESR    E        Ibox error status register (from Ebox)
    ISASAV   E        Ibox ISASAV register
    IVASAV   E        Ibox IVASAV register
    MEDR     E        Mbox Error Data Register
    MEAR     E        Mbox Error address register
    MSTAT1   E        Mbox status 1 register
    MSTAT2   E        Mbox Status 2 register
    VIBASAV  E        Ibox VIBASAV register
    VPCBITS  E        Valid PC Bits
```

9.2.8.4 Supported Miscellaneous Register Names

     The following table lists the supported Miscellaneous Register
(Misc) names known by the MACRO context EXAMINE and DEPOSIT commands.
When any of these names are used in an examine or deposit command no
'/datatype' switch is allowed, as the default is assumed to be /LONG.

     Miscellaneous Registers are not numerically addressable and can
be accessed only by using the specific register names given below.

Name	Access	Purpose
IBGPR	E	Ibox GPR
PSL	E/D	Program Status Longword
SPADR	E/D	Scratch Pad Address
STATE	E/D	STATE register
EVMQSAV	E	VMQ save

9.2.9   FIND

    Command Syntax:

        FIND [{ /RPB, /MEMORY }]

        Where the default for no switches is FIND/RPB.

    Description:

    This command searches main memory starting at location zero for a
    page-aligned 64KB block of good physical, or a Restart Parameter
    Block (RPB). If the item is found its address plus 200(hex) is
    loaded into the CPU's SP register.

    The searching algorithm is implemented as a pair of CSM overlays.
    The algorithm assumes that the INIT/PAMM command has already been
    used. INIT/PAMM will enable access to all of physical memory and
    save the top of main memory in the console RAM space.

    The FIND/MEM command is used mainly in boot command files to
    locate good memory in with to load the operating system's bootstrap
    code. Note that this command destroys that contents of main memory
    as it searches.

    The FIND/RPB command, as it is implemented in CSM, will only work
    if issued immediately after CSM initialization (INIT/CPU command).
    A FIND/RPB command is done internally by the console on power up
    when the front panel switches indicate that a system restart should
    be attempted. This command does not affect the contents of main
    memory.

9.2.10 HALT

Command Syntax:

    HALT

Description:

This command is used to halt the CPU by forcing it to enter the CSM
console service loop. The value of the macro PC is displayed and
the CSM status code should be 11(16). If the CPU is already halted
a message indicating this wil be printed, followed by the last
recorded macro PC and CSM status.

9.2.11 INITIALIZE (CPU, ESCRATCH, MICRO, PAMM)

Command Syntax:

    INITIALIZE [/switch]

    where '/switch' can be any one of the following:

        /CPU   /ESCRATCH   /MICRO   /PAMM

Description:

This command is used to initialize various facets of the MACRO
machine. When used with no switches, it executes the VAX processor
initialization described in DEC STD 032. See also the INIT command
described in the "general command set" section of this chapter.

INIT   - This command meets the requirements of DEC STD 032
                  for VAX processor initialization. Steps performed by
                  this command include:

                      o ACCS register (enabled if FBOX flag is ON,
                         else FBOX is disabled)
                      o RX TX and STX registers inited
                      o ASTLVL is set to 4
                      o SISR, ICCS, MAPEN, PME set to 0
                      o MCCTL is set to 0 to enable Mbox overlaps
                         if the CPU clock is running full speed.
                         Otherwise overlaps are disabled (MCCTL = 1).
                      o Turns off the VAX STATE lamp
                      o the IBUFF and system cache are not affected

INIT/CPU   - Provided that the Ebox has been loaded with system
                  microcode this command performs a full CSM
                  initialization. It combines the CSM initialization
                  with the steps for the INIT/ESC and INIT commands.
                  Specifically, this command does the following:

    o   Stop CPU clocks
    o   Master RESET
    o   Loads and runs CSM040 and CSM041 overlays
    o   SETs EXTI flags ON (see SET EXTI ON)
    o   Performs INIT/ESCRATCH operation
    o   Performs INIT operation

INIT/ESCRATCH   - This command loads the EBOX scratch pad registers
with the values needed to start the CPU.   It
uses CSM to do this, and assumes that the Ebox is
loaded and running.   Other CSM communications
cannot occur nor can CPU operations occur until
this command has been executed. Data used to load
the Escratch RAM is normally taken from the file
ECODE.BPN. An optional .bpn file name may be
specified if desired. i.e. INIT/ESC FOO[.BPN].
Note that this command will initialize GPR's (R0-SP).

INIT/MICRO   - This command performs microcode initialization.
It checks for the file KA86n.REV and, if found,
loads microcode revision information from this
file where it will be used by the LOAD/xxx command.
Next, the command file ULOAD.COM is submitted.
Commands in the ULOAD file will perform the actual
microcode initialization (see sample of ULOAD.COM
in appendix).

This command also checks that the hardware
revision of the machine, as seen at the SID
jumpers, meet the minimum hardware revision
requirement of the microcode.

INIT/PAMM   - This command determines the amount of physical
memory and the number and type of IO adapters on
the system and configures the Physical Address
Memory Map (PAMM) accordingly. An ABUS INIT
sequence is performed and the CONFIGURATION
register of all SBIA's found are initialized with
the number of megabytes of memory present,
and SBI CYCLES IN and OUT are enabled in the
CONTROL/STATUS register (of each SBIA). A synopsis
of the memory and IO configuration is assembled
and displayed. On entry, the MAP is turned off.
On exit, the INIT command sequence is performed.

9.2.12 LOAD (main Memory)

Command Syntax:

    LOAD [/START:hex_adr] filename[.exe]

Description:

This command is used to load main memory with binary data taken
from the specified file. If the /START switch is used then the
data is loaded starting at the specified 'hex_adr', otherwise
the data is loaded starting at location zero.

9.2.13 START

Command Syntax:

    START [hex_address]

Description:

This command is essentially the same as the CONTINUE command except
that if a 'hex_address' is specified it is used as the current PC
at which to start the VAX processor. If the CPU is already running
then this command simply re-enters PIO mode without affecting the
running CPU, even if a 'hex_address' is included.

9.2.14 START/STEP

Command Syntax:

    START/STEP [hex_address]

Description:

This is a special debug tool (for INTERNAL use only) which provides
a convenient means of preparing the CPU for micro-stepping through
macro code.

This command loads a special CSM overlay into the ebox, sets the PC
if specified, and starts the CSM idle loop. The operator may then
enter DEBUG mode (if not already there) and micro-step through the
macro code using MIC and/or TMIC commands.

Note that this command forces the Ebox to quit execution of the CSM
control loop so that macro instructions can be stepped. When
stepping is done the UNHANG command can be used to reset the Ebox
to the start of the CSM control loop without affecting the state of
the machine. Otherwise, the CPU clocks can be started and the macro
instruction microcode will continue to run at the point stepping
was stopped.

9.2.15 NEXT

Command Syntax:

    NEXT [hex_count]

Description:

This command single steps the VAX processor the specified number of
macro-instructions, or the default of 1 if no count is given. At
the completion of the full step count the PC of the next
macroinstruction is displayed and the command enters space-bar-
step-mode, indicated by the >>> prompt appearing at the end of the
current line (not at the left margin).

Single stepping is very inefficient as it involves loading several
separate CSM overlays. Also, the "next PC" will not be executed
if an interrupt condition is pending and enabled. In this case,
the first instruction of the interrupt service routine is executed
instead.

9.2.16   UNJAM

Command Syntax:

UNJAM [{ IOA0, IOA1, IOA2, IOA3 }]

The keywords SBI0 and SBI1 are equivalent to IOA0 and IOA1.
If no keyword is specified then all SBIA's present in the system,
as determined by the current contents of the PAMM, are UNJAMed.

Description:

This command enables the cycles in and out, asserts the master
interrupt enable bit, and issues an UNJAM to the specified SBI, or
to all SBI's present in the system (as determined by the current
contents of the PAMM). If no available adapters are present in the
system no warning is given by this command.

The UNJAM occurs when the UNJAM register of a specific adapter is
written to. The UNJAM register address is 2x080048, where x is
the I/O adapter number times 2. Performing an UNJAM operation to
an adapter type other than an SBIA has unpredictable results.

9.2.17   VERIFY

Command Syntax:

VERIFY [/switch]

Where '/switch' can be any one of the following values. If no
        switch is specified the default of /ALL is assumed.

        /ACCESS   /CONTEXT   /CYCLE
        /ECS   /FBACS   /FBMCS

                /FDRAM   /ICS   /IDRAM
                /MCF   /MCS   /PAMM

Description:

This command verifies the contents of any one or all control stores
by comparing the data read from the control store with the contents
of the .BPN file used to load it. If no switch is used, then all
control stores and the PAMM are verified.

Verification of a control store involves a word-by-word comparison
of data taken from the .BPN file with data read from the machine.
For a PAMM verify, the present memory configuration is determined
and the contents of the PAMM are checked against these findings.

On entry, this command performs a RESET command followed by a CPU
clock burst of 3 cycles. This guarantees access to all of the
control stores. This same procedure is used when an error is
detected to clear any possible parity error that may have been the
cause of the discrepancy.

If the HEX debugger is not enabled this command displays the total
number of discrepancies found. If HEX is enabled each failure is
reported with good/bad data (in .BPN format).

9.3 DIAGNOSTIC CONTEXT

The DIAGNOSTIC context (DC) provides commands necessary to support the loading and running of microdiagnostics. This context, along with the actual microdiagnostic programs, is designed to be used as a machine verification and trouble-shooting aid. It assumes that the MICROHARDCORE context has already been used to verify the hardcore of the machine. Refer to the 8600 DIAGNOSTIC USERS GUIDE for information on how to run and interpret MHC.

9.3.1 Context Initialization

The DIAGNOSTIC context is entered in response to the DIAGNOSE command. Upon entry, the context initialization procedure is performed as described in the DIAGNOSTIC CONTEXT INITIALIZATION section.

9.3.2 Diagnostic Support Microcode (DSM)

The monitoring and controlling of microdiagnostic programs is made possible by the DSM program running in the Ebox. This microcode provides DC with access to internal machine registers, such as the Ebox scratchpad, and provides microdiagnostic test dispatching. Refer to the DSM FUNCTIONAL SPECIFICATION for a description of DSM.

9.3.3 Console/DSM Communication

The console (DC) and DSM communicate together through CBUS command packet passing. DSM to DC packets consist of 6 bytes (1 function, 4 data, 1 checksum) while DC to DSM packets are 7 bytes in length (1 function, 1 extended-function, 4 data, 1 checksum). Bytes in a packets are transmitted in parallel (each byte has its own separate spot in the CBUS RAM) and both DSM and DC packets can be assembled and transmitted simultaneously since each packet uses its own section of the CBUS RAM.

When DC is ready to send a packet to DSM it places a non-zero function code into the DC$CONTROL byte of the packet (see symbol name definitions below). DSM senses (by polling) the non-zero function code and verifys the checksum of the packet. If the checksum of the packet, which includes the function and extended-function bytes, is bad then DSM returns an error status in the DC$CONTROL byte and ignores the packet; otherwise, the packet is processed and the DC$CONTROL byte is cleared (by DSM) to indicate successful completion. A "DC/DSM CHECKSUM FAILURE" is reported if an excessive number of packet checksum errors occurs.

Functions codes that can be sent to DSM by DC include the
following:

   o Examine or deposit Ebox Scratchpad location

   o Examine or deposit system Cache location

   o Examine or deposit Wbus register location

   o Start a microdiagnostic program

   o Continue a microdiagnostic program

   o Pause at end of current test

When DSM is ready to send a packet to DC it places a non-zero
function code into the DSM$CONTROL byte of the packet (see symbol name
definitions below). The writing of the DSM$CONTROL byte causes the
ready bit (RDY) to set in the TXCS register. TXCS RDY is detected by
DC in the mark time service routine which polls for TX interrupts
periodically. The checksum of the packet is then verified. If the
checksum of the packet is bad then DC returns an error status in the
DSM$CONTROL byte and ignores the packet; otherwise, the packet is
processed and the DSM$CONTROL byte is cleared (by the console) to
indicate successful completion. A "DC/DSM CHECKSUM FAILURE" is
reported if an excessive number of packet checksum errors occur.

There are two categories of function codes that DSM sends to DC,
as follows:

   o Status Change function codes are

     1. PAUSE_ON_FAULT_DETECTED -- a microdiagnostic has detected
         a fault and DSM has stopped executing tests in order to
         report the fault.

     2. PAUSE_WITH_FAULT_ON_LAST_PASS -- a microdiagnostic has
         detected a fault on the last iteration of the test. DSM
         has stopped executing tests in order to report the fault.
         This function code is equivalent to a FAULT_DETECTED
         function code followed by a PAUSE_AT_END_OF_PASSES
         function code.

     3. PAUSE_ON_UNEXPECTED_UTRAP -- an unexpected micro-trap
         caused DSM to enter one of the micro-trap vectors. This
         micro-trap occurred while DSM was running and indicates a
         failure in the CPU hardware. This is a fatal error.

     4. PAUSE_AT_END_OF_TEST -- DSM has completed a single test,
         and has entered the "Paused" state.

5. PAUSE_AT_END_OF_PASSES -- DSM has completed all tests up through "END_TEST" and has now stopped executing tests.

6. "PAUSE_AT_END_OF_DISPATCH -- DSM has executed all tests and has encountered the end of the diagnostics test dispatch table.

o Command function codes are

1. Requests to write data into a specific SDB control channel

2. Requests to have the EMM assert and deassert AC LO or DC LO

3. Requests to have DC manipulate bits in the console's miscellaneous status registers

The following CBUS addresses and symbol names are used internally for DC/DSM communication. The usage of the DSM$ALIVE byte is described in a later section.

Symbol	CBUS	Usage	
DSM$CONTROL	174000	DSM to DC:	function code
DSM$CHECK	174003		packet checksum
DSM$D0	174004		data byte 0 (LSB)
DSM$D1	174005		data byte 1
DSM$D2	174006		data byte 2
DSM$D3	174007		data byte 3 (MSB)
DC$CONTROL	174010	DC to DSM:	function code
DC$TARGET	174012		extended function code
DC$CHECK	174013		packet checksum
DC$D0	174014		data byte 0 (LSB)
DC$D1	174015		data byte 1
DC$D2	174016		data byte 2
DC$D3	174017		data byte 3 (MSB)
DSM$ALIVE	174020	"DSM is Alive" status	

9.3.4 Microdiagnostic Operation

While DC is busy monitoring a running microdiagnostic (via the RUN, STEP, START, or CONTINUE commands), it polls for TXCS RDY which indicates that a DSM packet needs to be processed. In addition to this, DC polls the CBUS location DSM$ALIVE to determine if the diagnostic is still running. If any test takes an excessively long time to complete (in the order of milliseconds) a 5-second timer is

started in the console. If the timer expires before the test
completes (and before DSM updates the DSM$ALIVE byte) DC reports an
error. After a diagnostic timeout the current command is aborted.
The DSM$ALIVE byte is incremented by DSM each time it is called by a
microdiagnostic to perform some system service. It is when the
diagnostic fails to call DSM that the alive byte timeout will occur.

9.3.5 Pausing/Continuing

Once a microdiagnostic has been started, DC can get back to its
prompt in one of four (4) ways:

1. User has selected the /FAULT:PAUSE switch. DC will report
   the first occurance of an error, print a "pausing" message
   and return to its command prompt.

2. User has selected /FAULT:ISOLATE in which case DC attempts to
   isolate, prints its "pausing" message and returns to its
   command prompt.

3. User has typed ^P in which case DC will print its "pausing"
   message and return to its command prompt when the current
   test has completed.

4. User has typed ^C in which case DC will abort the current
   test and return to its command prompt (without a "pausing"
   message).

If the user responds with the CONTINUE command, DC is expected to
continue running the tests from where it left off (only when the pause
was proceeded with a "pausing" message) .DC will complete pending
passes of a test before advancing to the next test. A diagnostic that
has been aborted by a ^C can not be continued.

In general, when DC is processing a command file (@filename) and
ends up back at the prompt due to a pause condition, it is impossible
to continue the command file. The concepts of pausing, /SINGLE_STEP,
CONTINUE, STEP are not supported in conjunction with command files.

If the user gives CONTINUE command and /SINGLE_STEP was in
effect, /SINGLE_STEP will be cancelled and DC will return to its
normal mode of operation.

9.3.6 Control Character Handling

There are three control characters handled by the console while
in DIAGNOSE context (DC).

^P (PAUSE microdiagnostic)

This control character causes a running microdiagnostic to be interrupted on completion of the current pass and control of the keyboard returned to the operator. Console commands can then be entered to modify the diagnostic environment followed by a CONTINUE command to resume execution of the microdiagnostic. If no microdiagnostic is running when the ^P is entered it is treated the same as a ^C.

^C (ABORT)

This control character causes an abort of the current console command which may include the need to abort a running microdiagnostic. Control of the console terminal is returned to the operator.

^T (Report status)

This control character causes DC to display information about the currently running diagnostic program without interfering with it.

9.3.7 Error Report Format

The general format of the error report displayed by DC is shown below. The amount of information displayed depends on the setting of the /PRINTMODE switch (see SET SWITCH command). The following example is the brief version.

FAULT DETECTED IN TEST #<number>

The following example is the verbose version of error report:

Diagnostic test name: <diagnostic's test name>
BPN version: <BPN test name, rev #, date>
Syndrome #: <number>
Start test: <number>
End test: <number>
Passes: <number>
Current Pass: <number>
Ebox scratchpad data: <all data setup with SET DATA command>
Fault detected in test #<number> - Diasgnostic name

9.3.8 Diagnostic Fault Isolation

9.3.8.1 Solid Faults

When a test detects a fault, DC will continue to run that test /PASSES number of times. Each time the test detects an additional fault, its fault symptom data (syndrome) is compared with the fault symptom data (syndromes) from previous failures. This is repeated until all /PASSES complete to insure that the failure mode is solid. A solid fault exists when the test fails 100% of the time and all syndromes are identical. When a failure is determined to be solid DC will execute the appropriate fault isolation algorithm with a high level of confidence (provided the /FAULT:ISOLATE or /FAULT:NOABORT option is in effect).

When isolation is performed, Isolation data in the form of ASCII messages and chip callouts are appended to the final error report.

9.3.8.2 Non-Solid Faults

If a test fails less than 100% of the time or all failing test syndromes are not identical, DC concludes that the fault is not solid and isolation will not be attempted. Although isolation is not attempted for non-solid fault situations, the standard error report is still produced for some maximum number of new syndromes as they occur. This limit on the number of error reports is controlled by the SET ISOLATION/ERROR_DUMPS command. The final error report contains pass/fail ratio data along with a message indicating that isolation could not be performed. This final report also shows a count of how many unique syndromes occured during the running of all the passes. If the number of unique syndromes is greater than 10, a message to that effect is printed (since only the first 10 syndromes are recorded).

9.3.9 DIAGNOSTIC Context Command Set

9.3.9.1 CLEAR DATA

Command Syntax:

CLEAR DATA

Description:

This command clears the table of pointers defined by the SET DATA command and should be used prior to re-defining a new set of Ebox scratchpad locations for purposes of error reporting.

9.3.9.2   CONTINUE (a Microdiagnostic)

Command Syntax:

    CONTINUE

Description:

    This command allows the currently loaded microdiagnostic to
    resume test execution after being paused by either a switch
    setting (/FAULT:PAUSE, /FAULT:ISOLATE) or ^P. The
    microdiagnostic resumes at the test following the one being
    executed when the pause occurred. If /SINGLE_STEP is in
    effect when the CONTINUE command is issued, single stepping is
    stopped and the remaining tests are run. A CONTINUE command
    issued prior to giving either a RUN or START command will
    result in an error message.

9.3.9.3   DEPOSIT (Cache, Escratch, Wbus)

Command Syntax:

    DEPOSIT { /switch } hex_addr hex_data

    Where '/switch' can be any one of the following items:

        /CACHE   /ESCRATCH   /WBUS

Description:

    This command allows the user to modify the Ebox scratchpad
    RAM, the system CACHE, or the WBUS RAM. The 32-bit hexdata is
    deposited into the hexaddr location of the specified RAM. The
    CPU Clock must be running.

9.3.9.4   EXAMINE (Cache, Escratch, Wbus)

Command Syntax:

    EXAMINE { /switch } hex_addr

    Where '/switch' can be any one of the following items:

        /CACHE   /ESCRATCH   /WBUS

Description:

    This command allows the user to examine the contents of
    specific Ebox Scratchpad, CACHE, or WBUS locations. The CPU
    Clock must be running.

9.3.9.5 FAULT_FREE

Command Syntax:

FAULT_FREE [/switch] [filename[.COM]]

Where '/switch' can be any combination of the following switches,
       provided that the command line does not exceed 80
       characters. Switches appended to the RUN command remain in
       effect until the command completes. If a RUN command is
       given without switches, their default values are used.

       /BELL:[ON, OFF]
       /DEBUG:{ 0, 1, 2, 3, 4 }
       /NUMBER:FIRST_TEST [{ <space>,<comma> } LAST_TEST]
       /PRINT_MODE:{ BRIEF, VERBOSE }
       /SINGLE_STEP

Description:

       This command initiates the execution of a command file, if
       specified, whose purpose is to load and start a
       microdiagnostic program. Execution begins with the first
       number specified in the /NUMBER switch and ends with either
       the last test in the group of microdiagnostic tests or with
       the last test specified in the /NUMBER switch. Note that the
       CPU clock must be running for the RUN command to work.

       Refer to the description of either the SET SWITCH, or START
       command for information on the switches for this command.

9.3.9.6 GENERATE

Command Syntax:

       GENERATE [filename]

Description:

       The GENERATE command is used to invoke the DC generate/verify
       process for validation and generation of some of the isolation
       data. This command is not for field use.   It is used to
       support the in-house development of isolation algorithms.
       There are no switches associated with this command. The only
       option is the name of the command file responsible for loading
       the microdiagnostic to be generated.

9.3.9.7   RUN

Command Syntax:

RUN [/switch] [filename[.COM]]

Where '/switch' can be any combination of the following
        switches, provided that the command line does not
        exceed 80 characters. Switches appended to the RUN
        command remain in effect until the command completes.
        If a RUN command is given without switches, their
        default values are used.

        /BELL:{ ON, OFF }
        /FAULT:{ ISOLATE, NOABORT, LOOP, PAUSE, CONTINUE, IGNORE }
        /DEBUG:{ 0, 1, 2, 3, 4 }
        /NUMBER:FIRST_TEST [{<space>,<comma>} LAST_TEST]
        /PASSES:dec_num
        /PRINT_MODE:{ BRIEF, VERBOSE }
        /SINGLE_STEP

Description:

        This command initiates the execution of a command file whose
        purpose is to load and start a microdiagnostic program.
        Execution begins with the first number specified in the
        /NUMBER switch and ends with either the last test in the group
        of microdiagnostic tests or with the last test specified in
        the /NUMBER switch. Note that the CPU clock must be running
        for the RUN command to work.

        Refer to the description of the SET SWITCH command for
        information on switches not described here.

        /NUMBER:FIRST_TEST [{<space>,<comma>} LAST_TEST]
                                                        default = 01,FF

                This switch is used to specify the starting and ending
                test numbers in hex.   If the switch is omitted,
                default values are used.

        /PASSES:'dec_num'   default = 100

                'dec_num' is the number of passes of a test to execute
                before continuing to the next test or attempting
                isolation. If the /PASSES switch is omitted, the
                default value will be used.

                A special case of /PASSES:0 modifies the sequencing of
                the tests so that a particular group of tests can be
                run indefinitely.   When /PASSES:0,   DC   will
                indefinetely run each of the tests (from FIRST to
                LAST) until interrupted by ^C, ^P, or a fault is
                detected.   When a fault is detected, the action will

be governed by the current setting of the /FAULT
switch.

/SINGLE_STEP

This switch enables the single_step feature. When
specified, diagnostic execution is interrupted after
completing the proper number of passes of an
individual test. A message, along with the test
number just completed, is displayed and the operator
is prompted for input. The STEP command can be used
to run the next test, followed by another pause at
completion. The CONTINUE command can be used to clear
this mode and resume full speed test execution.

9.3.9.8   SET DATA

Command Syntax:

SET DATA escratch_address data_name

Where 'escratch_address' is a hexadecimal address within the
        Ebox Scratchpad RAM (0 to FF).

and where 'data_name' is a 1-30 character string to be
        associated with the specified Escratch data

Description:

This command allows a symbolic name to be assigned to a
location in the Ebox Scratchpad RAM. When a microdiagnostic
test detects a fault, and DC is in verbose printing mode, the
'dataname' and data taken from the associated
'escratch_address' are included in the error report.

SET DATA commands are normally used in command files
responsible for loading microdiagnostics and setting
diagnostic control switches. A maximum of sixteen SET DATA
commands can be used at one time. The CLEAR DATA command
should be used to initialize the SET DATA memory prior to
defining set data information.

The SHOW DATA command can be used to display the current
settings of SET DATA, along with the current state of the
Escratch variables.

9.3.9.9   SET ISOLATION

Command Syntax:

        SET ISOLATION [/switch]

    Where '/switch' can be any combination of the following
            switches provided that the line does not exceed 80
            characters.

            /PASSES:'dec_num'
            /ERROR_DUMPS:'dec_num'

Description:

    This command provides a way to modify default parameters
    associated with isolation.   Once the defaults are modified,
    they remain in effect until DC is re-loaded (ie.   DIAG
    command).

    /PASSES:'dec_num'   default = 100

        'dec_num' is the number of times each test will be
        executed   before   continuing   to   the   next   test   or
        attempting isolation. It is not recommended that this
        number be altered in the field as it effects the
        confidence level of any isolation data that may
        follow.   Factory settings are prefered. This number
        should always be greater than zero (0).

    /ERROR_DUMPS:'dec_num'   default = 10

        'dec_num' is the maximum number of error printouts
        that will occur as faults with different syndromes are
        detected within a given test.   Lowering this number
        has the effect of reducing the amount of error
        printouts at the expense of throwing away what may be
        useful data in tracking down an intermittant. This
        number has a maximum value of 10.

9.3.9.10   SET NAME

    Command Syntax:

        SET NAME microdiagnostic_filename

    Description:

        This command specifies the name of the microdiagnostic
        currently being loaded. This information and is included in
        the fault report and also to locate other associated files
        with the same name (different extensions).

9.3.9.11 SET SWITCH

Command Syntax:

SET SWITCH [/switch]

Where '/switch' can be any combination of the following
     switches, provided that the line does not exceed 80
     characters. Switches altered with this command retain
     the new setting until changed again with the SET
     SWITCH command, or DC is reloaded.

     /BELL:{ ON, OFF }
     /FAULT:{ ISOLATE, NOABORT, LOOP, PAUSE, CONTINUE, IGNORE }
     /DEBUG:{ 0, 1, 2, 3, 4 }
     /PRINT_MODE: { BRIEF, VERBOSE }

Description:

     This command redefines the default switch settings that govern
     the behavior of DC and DSM in the control and running of
     microdiagnostics. A description of each switch is provided
     below, including the default setting of the switch when
     DIAGNOSTIC context is entered.

     /BELL:{ ON, OFF }   default = OFF

          When ON, this switch causes the terminal bell to ring
          each time a new fault is detected and reported.

     /FAULT:{ ISOLATE, NOABORT, LOOP, PAUSE, CONTINUE, IGNORE }
                                          default = ISOLATE

          This switch controls the action taken by DC after a
          microdiagnostic test detects a fault. Only one of the
          above arguments may be specified; they are mutually
          exclusive.

          /FAULT:ISOLATE

          This setting, which is also the default, directs DC to
          attempt isolation on all solid faults encountered.
          After the isolation attempt is completed, DC returns
          to its command prompt. The CONTINUE or STEP commands
          can be used to resume test execution.

          /FAULT:NOABORT

          This setting, directs DC to attempt isolation on all
          solid faults encountered. After the isolation attempt
          is completed, DC will continue onto the next test,
          thus providing a method of continuing with the
          diagnostic execution after isolating.

/FAULT:LOOP

Tests within a diagnostic are targeted to run /PASSES times. When an error is detected, the standard error report is displayed and then DSM is instructed to loop forever on the failing test. Only ^C or ^P can terminate the loop.

/FAULT:PAUSE

This setting causes DC to return to its command prompt after reporting a test failure. The CONTINUE or STEP command can be used to resume test execution.

/FAULT:CONTINUE

With this switch in effect, DC will perform all of its normal error reporting. After each error report, DC will continue its normal test sequencing as if no error had occured. Isolation will not take place.

/FAULT:IGNORE

This is a special switch intended only for microcoder use. It has no useful application in the field. It directs DSM to never signal DC that an error has occured. No hardware errors can ever be detected or reported.

/DEBUG:{ 0, 1, 2, 3, 4 }   default = OFF

This switch is used by microcoders during debug of isolation code. This switch controls the print of isolation line numbers as follows:

        0 = none
        1 = Fault statements only
        2 = Fault and Print statements only
        3 = Fault, Print, and Conditional statements
        4 = All conditional statements

/PRINT_MODE:{ BRIEF, VERBOSE }   default = VERBOSE

This switch controls the level of detail included in an error report.

9.3.9.12   SHOW DATA

Command Syntax:

SHOW DATA

Description:

This command displays all symbolic names and associated data currently defined by the SET DATA command. The range of diagnostic TEST NUMBERS along with the number of PASSES is also displayed under this command for the users information needs.

9.3.9.13   SHOW SWITCHES

Command Syntax:

SHOW SWITCHES

Description:

This command displays the current switch settings and the default switch setting. The default setting are either those settings that were in place when DC was loaded or have been modified via the SET SWITCH command.

The current switch settings are normally the same as the default settings since local settings (ie appending switches to RUN or START) return to defaults when the command completes.

The current switches will differ from the defaults when they are examined from within another command. This could occur by examining the switches from the paused state of a RUN or START command which uses the /SINGLE_STEP switch. Proceeding from that point with the CONTINUE or STEP command will cause the current switch settings to remain in effect until the initial RUN or START command has gone to completion.

9.3.9.14   START

Command Syntax:

START [/switch]

Where '/switch' can be any combination of the following switches, provided that the command line does not exceed 80 characters. Switches appended to the START command remain in effect until the command completes. If a START command is given without switches, their default values are used.

    /BELL:{ ON, OFF }
    /FAULT:{ ISOLATE, NOABORT, LOOP, PAUSE, CONTINUE, IGNORE }
    /DEBUG:{ 0, 1, 2, 3, 4 }

```
/NUMBER:FIRST_TEST [ {<space>,<comma>} LAST_TEST ]
/PASSES:dec_num
/PRINT_MODE:{ BRIEF, VERBOSE }
/SINGLE_STEP
```

Description:

This command initiates the execution of a test or group of tests within an already loaded microdiagnostic program. Execution begins with the first number specified in the /NUMBER switch and ends with either the last test in the group of microdiagnostic tests or with the last test specified in the /NUMBER switch. Note that the CPU clock must be running for the START command to work.

Refer to the description of the SET SWITCH command for information on the switches not described here.

/NUMBER:FIRST_TEST [{ <space>,<comma> }LAST_TEST]

default = 01,FF

This switch is used to specify the starting and ending test numbers in hex. If the switch is omitted, the default values are used.

/PASSES:'dec_num'

default = 100

'dec_num' is the number of test passes to execute before continuing to the next test or attempting isolation. If the /PASSES switch is not specified, the default value will be used.

A special case of /PASSES:0 modifies the sequencing of the tests so that a particular group of tests can be run indefinitely. When /PASSES:0, DC will indefinetely run each of the tests (from FIRST to LAST) until interrupted by ^C, ^P, or a fault is detected. When a fault is detected, the action will be governed by the current setting of the /FAULT switch.

/SINGLE_STEP

This switch enables the single_step feature. When specified, diagnostic execution is interrupted after completing the proper number of passes of an individual test. A message, along with the test number just completed, is displayed and the operator is prompted for input. The STEP command can be used to run the next test, followed by another pause at completion. The CONTINUE command can be used to clear this mode and resume full speed test execution.

9.3.9.15   STEP

Command Syntax:

STEP

Description:

This command causes the next test in the currently loaded
micro-diagnostic to be executed. The STEP command is invalid
unless the microdiagnostics have been started. Once they have
been started and there is a pause in the dianostic execution
(/FAULT:PAUSE, /FAULT:ISOLATE, or ^P), the STEP command may be
used.   A STEP command issued prior to giving either a RUN or
START command will result in an error message.

9.4 MICROHARDCORE CONTEXT

The MHC context provides the tests necessary to verify the proper operation of each micro-sequencer of the 8600 CPU. The tests are divided into individual CPU box-related groups. The operator may select to execute one or all of these test groups, or a series of tests in any specified group. Refer to the 8600 DIAGNOSTIC USERS GUIDE and EDKAA.DOC for information on how to run and interpret MHC.

MHC testing requires that the console module, console terminal and the console load device (RL02) be fully functioning.

MHC testing requires that the correct information be contained in the file CDF86n.DAT on the load media. This file provides pointers to the correct CAD files for visibility bus translations.

Enter the MHC context by typing "MHC" at the MACRO prompt (>>>) or at the DIAG prompt (DC>). MHC performs its initialization at then prompts for operator commands (MH>).

9.4.1 MICROHARDCORE Context Command Set

The following commands are present in the MHC set, including those described in the "general command set" section.

9.4.2 START

Command Syntax:

    S[TART][{Q, C, E, F, I, L, M, N, R, S, U}] [/switches]

    where '/switches' can be any combination of the following:

            /FAULT:{ ISOLATE, CONTINUE, LOOP, PAUSE }
            /NUMBER:START_TEST,END_TEST
            /PASS:dec_num
            /PRINT_MODE:{ VERBOSE, BRIEF, QUIET}
            /BELL:{ OFF, ON }
            /ERROR_DUMPS:dec_num
            /QUICKVERIFY

Description:

The start command causes the selected test group to run for one pass, or for the number of passes specified by /PASS. By default the entire test group is run, unless the /NUMBER switch is used to specify a range of tests within that group. The test group is selected by appending a single letter character to the START command or to its abbreviated form of "S".

A brief description of what each START command does is listed below:

    S   Run all tests in all groups (approx. 20 minutes)
    SQ   Run most tests in all groups (quicker, approx. 10 min.)
    SC   Run all Clock tests
    SE   Run all Ebox tests (group R, S, and U)
    SF   Run all Fbox tests
    SI   Run all Ibox tests
    SL   Run all Last multi box access tests
    SM   Run all Mbox logic tests
    SN   Run all Mbox ucode tests
    SR   Run only the Ebox MCF-CTX RAM and basic ucode tests
    SS   Run only the Ebox SDB and control store tests
    SU   Run only the Ebox expanded ucode tests

The switch options available on this command are listed below.

/FAULT:{ ISOLATE, CONTINUE, LOOP, PAUSE }   default = ISOLATE

    /FAULT:ISOLATE
    Continue after reporting a test-detected error.

    /FAULT:CONTINUE
    Continue after reporting a test-detected error.

    /FAULT:LOOP
    Loop indefinitely on the next failing test. Only ^C will
    stop this loop.

    /FAULT:PAUSE
    Pause at MH> prompt after reporting an error. The
    CONTINUE command can be used to resume testing at the next test.

/NUMBER:START_TEST,END_TEST

    Specifies the series of test numbers to be run. This range
    applies to the group or groups of tests being run. Test
    numbers in MHC are in HEXadecimal format.

/PASS:dec_num   default = 1

    where 'dec_num' specifies the number of iterations to make
    through the entire test series. The default of
    one pass is made if this switch is not used.

    This switch overrides the default pass count of 1. If a range
    of tests is selected with /NUMBER then those tests will be
    repeated, in sequence, /PASS number of times.

 /PRINT_MODE:{ VERBOSE, BRIEF, QUIET}   default = VERBOSE

    /PRINT_MODE:VERBOSE
    Reports errors in long form, with RAM chip callouts.
    This cancels QUIET mode.

/PRINT_MODE:BRIEF
Reports errors in short form, with NO RAM chip callouts.
This cancels QUIET mode.

/PRINT_MODE:QUIET
Suppresses all output except for error reports, which are
displayed in VERBOSE form.

/BELL:{ ON, OFF }   default = OFF

When ON, this switch causes the terminal bell to ring each
time a new fault is detected.

/ERROR_DUMPS:dec_num   default = 5

where 'dec_num" is the maximum number of error printouts that
will occur as faults are detected within a given test.
Lowering this number has the effect of reducing the amount of
error printouts at the expense of throwing away what may be
useful data in tracking down an failure.

This switch overrides the default error dump count of 5.
Once this switch is used, the new value will become the
default value of error printouts that will occur as faults
are detected within a given test.

/QUICKVERIFY

Allows the operator to over-ride normal program execution
and run the selected test with shorter execution time.

9.4.3   LOOP

Command Syntax:

L[OOP] [{Q, C, E, F, I, L, M, N, R, S, U}] [/switches]

where '/switches' can be any combination of the following:

        /FAULT:{ ISOLATE, CONTINUE, LOOP, PAUSE }
        /NUMBER:START_TEST,END_TEST
        /PASS:dec_num
        /PRINT_MODE:{ VERBOSE, BRIEF, QUIET }
        /BELL:{ OFF, ON }
        /ERROR_DUMPS:dec_num
        /QUICKVERIFY

Description:

The loop command causes the selected test group to run indefinitely,
or for the number of passes specified by /PASS. By default the

entire test group is run, unless the /NUMBER switch is used to
specify a range of tests within that group. The test group is
selected by appending a single letter character to the LOOP command
or to its abbreviated form of "L".

A brief description of what each LOOP command does is listed below:

       L   Repeat all tests in all groups (approx. 20 minutes)
       LQ   Repeat most tests in all groups (quicker, approx. 10 min.)
       LC   Repeat all Clock tests
       LE   Repeat all Ebox tests (group R, S, and U)
       LF   Repeat all Fbox tests
       LI   Repeat all Ibox tests
       LL   Repeat all Last multi box access tests
       LM   Repeat all Mbox logic tests
       LN   Repeat all Mbox ucode tests
       LR   Repeat only the Ebox MCF-CTX RAM and basic ucode tests
       LS   Repeat only the Ebox SDB and control store tests
       LU   Repeat only the Ebox expanded ucode tests

The switch options available on this command are listed below.

/FAULT:{ ISOLATE, CONTINUE, LOOP, PAUSE }   default = ISOLATE

    /FAULT:ISOLATE
    Continue after reporting a test-detected error.

    /FAULT:CONTINUE
    Continue after reporting a test-detected error.

    /FAULT:LOOP
    Loop indefinitely on the next failing test. Only ^C will
    stop this loop.

    /FAULT:PAUSE
    Pause at MH> prompt after reporting an error. The
    CONTINUE command can be used to resume testing at the next test.

/NUMBER:START_TEST,END_TEST

    Specifies the series of test numbers to be run. This range
    applies to the group or groups of tests being run. Test
    numbers in MHC are in HEXadecimal format.

/PASS:dec_num   default = Infinity

    where 'dec_num' specifies the number of iterations to make
    through the entire test series. The default of
    infinity is made if this switch is not used.

    This switch overrides the default pass count of infinity. If
    a range of tests is selected with /NUMBER then those tests
    will be repeated, in sequence, /PASS number of times.

/PRINT_MODE:{ VERBOSE, BRIEF, QUIET}   default = VERBOSE

/PRINT_MODE:VERBOSE
Reports errors in long form, with RAM chip callouts.
This cancels QUIET mode.

/PRINT_MODE:BRIEF
Reports errors in short form, with NO RAM chip callouts.
This cancels QUIET mode.

/PRINT_MODE:QUIET
Suppresses all output except for error reports, which are
displayed in VERBOSE form.

/BELL:{ ON, OFF }   default = OFF

When ON, this switch causes the terminal bell to ring each
time a new fault is detected.

/ERROR_DUMPS:dec_num   default = 5

where 'dec_num" is the maximum number of error printouts that
will occur as faults are detected within a given test.
Lowering this number has the effect of reducing the amount of
error printouts at the expense of throwing away what may be
useful data in tracking down an failure.

This switch overrides the default error dump count of 5.
Once this switch is used, the new value will become the
default value of error printouts that will occur as faults
are detected within a given test.

/QUICKVERIFY

Allows the operator to over-ride normal program execution
and run the selected test with shorter execution time.

9.4.4   CONTINUE

Command Syntax:

CONTINUE

Description:

This command can be used to continue testing after execution was
paused due to an error detected and the /FAULT:PAUSE switch selected.

9.4.5 REPORT

Command Syntax:

REPORT

Description:

This command causes the end of pass report to be displayed. An example of what the report looks like is shown below.

```
              SECTION, TOTAL # OF ERRORS
              ---------------------------
              CLOCK   (SC) = 00000
              E SDB/CS(SS) = 00000
              E RAMS   (SR) = 00000
              E UCODE  (SU) = 00009
              I BOX    (SI) = 00000
              M LOGIC  (SM) = 00000
              M UCODE  (SN) = 00000
              F BOX    (SF) = 00000
              VAX-8600(SL) = 00000
PASS COUNTER = 00001, # OF ERRORS = 00009, TOTAL # OF ERRORS = 00009
```

9.5 THE HEX DEBUGGER

This chapter discusses the hardware debug and trace facility, referred to as HEX, which can be optionally enabled from either the DIAGNOSTIC or MACRO contexts. Debug commands are not supported under the MICROHARDCORE context.

In general, the debug command set allows the operator to modify and/or interrogate the state of the CPU. The primary purpose of the DEBUGGER then, is to serve as a debugging tool for manufacturing, engineering, and field service.

The HEX command set is enabled with the DEBUG command. The presence of the HEX command set is indicated by the addition of another angle bracket ">" to the end of the current command prompt. For example, the "DC>" prompt becomes "DC>>" when the HEX command set is enabled.

9.5.1 The HEX Command Set

9.5.2 CLEAR BREAK

Command syntax:

CLEAR { ABREAK, OBREAK } { Symbol, Reg, Hex_id, ALL }

Where 'Symbol' is a V$XXXX visibility symbol name,
    'Reg' is a visibility register name,
    'Hex_id' is a 16-bit hexadecimal id number as described
    in the EXAMINE/SDB section
    'ALL' will remove all breakpoints from the specified table

Description:

This command removes the specified breakpoint from the appropriate table. If 'ALL' is specified, all of the breakpoints in that table are cleared.

Examples:

>>>>CLEAR ABREAK V$A123

>>>>CLEAR OBREAK ALL

9.5.3 CLEAR COUNT

Command syntax:

CLEAR COUNT

Description:

This command explicitly initializes the step-counter to zero.

The counter keeps track of the number of machine steps (micro or state) since last time it was cleared. The REPORT command will display the state of this counter, along with a normal trace-display operation.

The step-counter is automatically initialized whenever "trace-step-mode" is changed using the TMICRO or TSTATE commands. For instance, if you are TMIC stepping and enter the TSTATE command the step-counter is automatically cleared.

9.5.4 DEPOSIT (control Store Address Spaces)

Command Syntax:

    DEPOSIT [/NEXT:hex_num] { /switch } hex_addr hex_data

Where '/switch' can be any one of the following:

/ACCESS	/CONTEXT	/CYCLE
/ECS	/FBACS	/FBMCS
/FDRAM	/ICS	/IDRAM
/MCF	/MCS	

And where 'hex_addr' can be a hexadecimal number or one of the following relative-to-last-location specifiers:

    *   (Access last specified address)
    +   (Access address following last (*) address)
    -        (Access address preceding last (*) address)

And where 'hex_data' can be any hexadecimal number within the range limits of the specified control store or RAM.

Description:

This command allows the operator to modify the contents of any control store or RAM location in the 8600 CPU. Note that the MCS, ACCESS, and CYCLE rams can be deposited only if the system is out of the RESET state. The command "MICROstep 3" can be used to ensure that this is true.

The /NEXT switch can be used in combination with any other switch to perform successive deposits of the same data into consecutive locations.

This command will complain if attempted while the CPU clock is running.

Examples:

>>>>DEPOSIT/ACCESS 0 0

>>>>DEPOSIT/ECS/NEXT:10 0 0

>>>>DEPOSIT/CYCLE * 0FA

9.5.5   DEPOSIT/CHANNEL

Command Syntax:

DEPOSIT/CHANNEL hex_chnl hex_data

Where 'hex_chnl' is one of the following SDB channel numbers:

```
00 = FBA
01 = FBM
03 = IBD
05 = ICA
08 = EDP
09 = EBE
0A = MCC
0D = EBC
0E = CSB
10 = VBA              (SBIA Visibility Module)
```

And where 'hex_data' is a 16-bit hex value to be deposited into
    the control channel.

Description:

This command deposits the specified 'hex_data' into the specified
SDB control channel. Control channels vary in length, so if the
specified data exceeds the length of the channel then the high
order bits in the data will overflow the channel and be ignored.

Obviously, a strong understanding of SDB channel logic is
recommended before attempting to use this command.

Note that this command will complain if attempted while the CPU
clock is running.

A list of control channel bit positions is included here.

Channel	Bit	Signal name
00 (FBA)	00	MSQ4 SBDADR 0 L
	01	MSQ4 SBDADR 1 L
	02	MSQ4 SBDADR 2 L
	03	MSQ4 SBDADR 3 L
	04	MSQ4 SBDADR 4 L

```
                     05      MSQ4 SBDADR 5 L
                     06      MSQ4 SBDADR 6 L
                     07      MSQ4 SBDADR 7 L
                     08      MSQ4 SBDADR 8 L
                     09      MSQ4 SDBCTL 0 L
                     10      MSQ4 SDBCTL 1 L
                     11      MSQ4 SDBCTL 2 L

         01 (FBM)    00      FM02 SDB FDR A4 H
                     01      FM02 SDB FDR A5 H
                     02      FM02 SDB FDR A6 H
                     03      FM02 SDB FDR A7 H
                     04      FM02 SDB FDR A8 H
                     05      FM02 SDB FDR A9 H

         03 (IBD)    00      IBDH SDB DA00 H
                     01      IBDH SDB DA01 H
                     02      IBDH SDB DA02 H
                     03      IBDH SDB DA03 H
                     04      IBDH SDB DA04 H
                     05      IBDH SDB DA05 H
                     06      IBDH SDB DA06 H
                     07      IBDH SDB DA07 H
                     08      IBDH SDB DA08 H
                     09      IBDH SDB DA09 H
                     10      IBDH SDB DA10 H
                     11      IBDH SDB DA11 H
                     12      IBDH SDB DA12 H
                     13      IBDH SDB DA13 H
                     14      IBDH SDB DA14 H
                     15      IBDH SDB DA15 H

         05 (ICA)    00      ICA1 ICS CNTRL CHNL 0
                     01      ICA1 ICS CNTRL CHNL 1
                     02      ICA1 ICS CNTRL CHNL 2
                     03      ICA1 ICS CNTRL CHNL 3

         08 (EDP)    00      EDPI DISA BYTE 32 PAR H
                     01      EDPI DISA BYTE 10 PAR H
                     02      EDPI FLIP WREG PAR H
                     03      EDPI DISA SCE AR BUS H
                     04      EDPI DISA ALU AR BUS H
                     05     -EDPI FLIP GPRA H
                     06     -EDPI FLIP GPRB H
                     07      EDPI SPARE 0 H

         09 (EBE)    00      EBEH SDB CTL D00 H
                     01      EBEH SDB CTL D01 H
                     02      EBEH SDB CTL D02 H
                     03      EBEH SDB CTL D03 H
                     04      EBEH SDB CTL D04 H
                     05      EBEH SDB CTL D05 H
                     06      EBEH SDB CTL D06 H
                     07      EBEH SDB CTL D07 H
```

```
                    08        EBEH SDB CTL D08 H
                    09        EBEH SDB CTL D09 H

        0A (MCC)    00        MCCA LOAD ENA H
                    01        MCCA WRITE MICRO A L
                    02        Unused
                    03        MCCA SHIFT ENA H
                    04        MCCA WRITE MICRO B L
                    05        Unused
                    06        MCCB DIAG INTR H
                    07        MCCB WRITE VIOL L
                    08        Unused
                    09        MCCB DIAG RAM WRITE H
                    10        MCCB CPR WRITE L
                    11        Unused

        0D (EBC)    00        EBC1 SDB CTL D00 H
                    01        EBC1 SDB CTL D01 H
                    02        EBC1 SDB CTL D02 H
                    03        EBC1 SDB CTL D03 H
                    04        EBC1 SDB CTL D04 H
                    05        EBC1 SDB CTL D05 H
                    06        EBC1 SDB CTL D06 H
                    07        EBC1 SDB CTL D07 H
                    08        EBC1 SDB CTL D08 H
                    09        EBC1 SDB CTL D09 H
                    10        EBC1 SDB CTL D10 H
                    11        EBC1 SDB CTL D11 H
                    12        EBC1 SDB CTL D12 H
                    13        EBC1 SDB CTL D13 H
                    14        EBC1 SDB CTL D14 H
                    15        EBC1 SDB CTL D15 H

        0E (CSB)    00        CSBR CNSL OP1 FLAG H
                    01        CSBR CNSL OP2 FLAG H
                    02        CSBR LOAD DIAG CNTR H
                    03        -CSBR FLIP USTK PAR H

        10 (VBA)    00        VB06 CLK CNTRL DATA 0 H
                    01        VB06 CLK CNTRL DATA 1 H
                    02        VB06 CLK CNTRL DATA 2 H
                    03        VB06 CLK CNTRL DATA 3 H
                    04        VB06 CLK CNTRL DATA 4 H
                    05        VB06 CLK CNTRL DATA 5 H
                    06        VB06 CLK CNTRL DATA 6 H
                    07        VB06 CLK CNTRL DATA 7 H
                    08        VB06 CLK CNTRL 8 H
                    09        VB06 CLK CNTRL 9 H
                    10        VB06 CLK CNTRL 10 H
                    11        VB06 CLK REG LD ENA H
```

9.5.6 DEPOSIT/CSPE

Command Syntax:

DEPOSIT/CSPE { /switch } [/NOFILE] hex_addr

Where '/switch' can be any one of the following:

        /ECS   /FBACS   /FBMCS
        /ICS   /MCS   /IDRAM
        /FDRAM

And where 'hex_addr' is a hexadecimal number.

NEXT:n and relative address (*, +, -) specifiers are not supported.

Description:

This command allows the operator to re-deposit a control store micro-word with bad parity. The data deposited is taken from the .BPN file currently loaded into the specified control store and modified to contain bad parity (parity bit(s) are flipped).

The /NOFILE switch may be used to force the source data to be taken from the ram itself (read-modify-write) rather than the .BPN file.

Correct parity can be restored by either re-loading the entire control store (using the LOAD command), or by using the DEPOSIT command and specifying the original control store data. Also, the DEPOSIT/MARK command could be used to restore data in three of the control stores.

Note that this command will complain if attempted while the CPU clock is running.

Examples:

    >>>>DEPOSIT/CSPE/ECS 3B

9.5.7 DEPOSIT/MARK

Command Syntax:

DEPOSIT/MARK { /switch } [/NOFILE] hex_addr { ON, OFF }

Where '/switch' can be any ONE of the following items:

        /ECS   /ICS   /MCS

And where 'hex_addr' is a hexadecimal number.

NEXT:n and relative address (*, +, -) specifiers are not supported.

Description:

This command allows the operator to set or clear a control store
mark-stop bit. The ON/OFF keyword in the command determines which
operation will be performed. The data deposited is taken from the
.BPN file currently loaded into the specified control store and
modified to contain the proper mark-bit setting and correct parity.

The /NOFILE switch may be used to force the source data to be taken
from the ram itself (read-modify-write) rather than the .BPN file.

Setting a MARK bit in a control store location causes the CPU clock
to stop when the microword is encountered only if the SOMM flag
(stop on micro-mark) has been set with the appropriate switch.
When running at full system clock speed the SOMM flag need not be
cleared before the MICROSTEP and STATESTEP commands can be used,
but in 1/5th speed the SOMM flag must be cleared since the processor
will stop directly on the instruction containing the markstop bit.

Note that this command will complain if attempted while the CPU
clock is running.

Examples:

>>>>DEPOSIT/MARK/ICS 2A ON

9.5.8 EXAMINE (control Store Address Spaces)

Command Syntax:

    EXAMINE [/NEXT:hex_num] { /switch } hex_addr

    Where '/switch' can be any one of the following:

            /ACCESS   /CONTEXT   /CYCLE
            /ECS   /FBACS   /FBMCS
            /FDRAM   /ICS   /IDRAM
            /MCF   /MCS

    And where 'hex_addr' can be a hexadecimal number or one of the
    following relative-to-last-location specifiers:

            *   (Access last specified address)
            +   (Access address following last (*) address)
            -        (Access address preceding last (*) address)

    If 'hex_addr' is unspecified, + is assumed.

Description:

This command allows the operator to examine the contents of any
control store or RAM location in the 8600 CPU. The /NEXT switch

can be used to examine any number of successive locations. Note
that the MCS, ACCESS, and CYCLE rams can be examined only if the
system is out of the RESET state. The command "MICROstep 3" can
be used to ensure that this is true.

Note that this command will complain if attempted while the CPU
clock is running. Furthermore, examining any control store
location (ECS, MCS, ICS, FBACS, or FBMCS) will alter the state
of the CPU in different ways, depending on which control store
is examined.

                              Note

The EXAM/MCS command performs a special function when the specified
address is outside the range of the Mbox control store (i.e.,
greater than FF). When an invalid address is used the command read
the state of the MCC shift path and converts it to .ULD format
(UCODE.ULD) and displays the result. This allows the operator to
inspect data in the MCC shift path that has caused an Mbox Parity
error.

Examples:

>>>>EXAMINE/ECS/NEXT:5 0

>>>>EXAMINE/CYCLE *

9.5.9   EXAMINE/CHANNEL

Command Syntax:

EXAMINE/CHANNEL hex_chnl

Where 'hex_chnl' is the SDB channel number.

SDB channel numbers are assigned as follows:

00 = FBA	01 = FBM	02 = MCD
03 = IBD	04 = IDP	05 = ICA
06 = ICB	07 = CLK	08 = EDP
09 = EBE	0A = MCC	0B = MAP
0C = EBD	0D = EBC	0E = CSB
0F = CSA	10 = IOA0	11 = IOA1
12 = IOA2	13 = IOA3	14 = MTM
15 = Reserved	16 = Reserved	17 = Reserved

Description:

This command will display all visibility bits in the specified
SDB channel. The data is presented as a string of hex characters
followed by a three character checksum. The number of characters
displayed (4 bits per) varies as follows:

                    FBA/FBM ..... 72 characters (288 bits -- 24. x 12.)
                    IDP/IBD/ICB . 64 characters (256 bits -- 32. x 8.)
                    ALL OTHERS .. 48 characters (192 bits -- 24. x 8.)

     Note that this command will complain if attempted while the CPU
     clock is running.

Example:

          >>>>EXAMINE/CHAN 0

9.5.10 EXAMINE/SDB

     Command Syntax:

     EXAMINE/SDB { Symbol_name, Register_name, Hex_id, "Signal_name"}

     Where 'Symbol_name' is the V$ symbol assigned to the particular
          visibility signal by the SDB CAD (.CDF) files.

          'Register_name' is the name of a visibility register defined
               within HEX or by the TRACE DEFINE command. A visibility
               register is a group of vterm bits joined together in some
               logical arrangement. See appendix for a list of pre-
               defined visibility register names.

          'Hex_id' is a 16-bit SDB or REGister ID. SDB_ID's are
               hardware visibility-bit addresses and are defined in the
               .CDF files. REG_ID's are software-defined numerical tags
               for visibility registers.

          "Signal_name" is the full visibility signal name, including
               H/L specifier, enclosed in double quotes.

     Description:

     This command displays the name and state of a single visibility
     signal, or the name and state of a visibility register.

     Note that this command will complain if attempted while the CPU
     clock is running.

Examples:

          >>>>EXAMINE/SDB V$A123   ; Use symbol name

          >>>>EXAMINE/SDB WBUS   ; Use register name

          >>>>EXAMINE/SDB 1   ; Use SDB id

          >>>>EXAMINE/SDB 8000   ; Use register id

9.5.11   EXIT

Command Syntax:

    EXIT

Description:

This command disables the HEX command set. This is indicated by
the disappearance of the right-most angle bracket on the current
command prompt.

9.5.12   MICROSTEP

Command syntax:

MICROSTEP [hex_num]

Where 'hex_num' is a hexadecimal step count in the range of 1 to
100. If unspecified, 1 is assumed.

Description:

The MICROSTEP command causes 'hex_num' microinstructions to be
executed at full system clock speed. Before stepping begins the
CPU clock is forced to the T3 state. CPU clocks must be stopped
for this command to function.

If the SBIA Visibility Module is present this command attempts to
step the SBIA clock as well. If the MICROSTEP count is greater
than 1 then a single SBIA clock cycle is done at the completion of
the CPU clock cycles. For a single cpu cycle no SBIA cycle is done,
but additional cpu cycles invoked in space-bar-step mode will cause
the SBIA clock to be cycled once for every other cpu cycle.

The execution of a MICROSTEP command invokes Space-bar-step mode
(indicated by the SBSM>> or SBSMDC>> prompts). In Space-bar-step
Mode, the user can cause one additional step to be executed by
hitting the space-bar (one microstep per space character). Space-
bar-step mode remains in effect until something other than a space
is input. The SBIA clock is stepped every other time through the
space-bar-step cycle (only if the SBIA Visibility Module is present).

All box UPC's are displayed at the end of each microstep command.
The UPC information for this display is collected after all
stepping has finished, while the clock is in the T3 state.

9.5.13   REPORT

Command syntax:

REPORT

Description:

The REPORT command will read all visibility data and display this information in accordance with the current trace list options (set by the TRACE command).

REPORT is implicitly invoked on completion of any trace function (either TMICRO or TSTATE).

The trace reporter assumes all visibility registers are 48. bits or less in length. Certain pre-defined registers are greater than 48 bits and are not intended to be 'traceable'. If one of these large registers is placed in the trace list (whether intentional or not), its displayed value will be truncated and its name prefixed with 'T' to denote that truncation has occurred.

Any register or signal whose value is different from that observed on the previous report will be prefixed with '*' to denote that a change has occurred.

9.5.14   SET BREAK

Command syntax:

SET { ABREAK, OBREAK } { Symbol, Reg, Hex_id } [SPAN, VALUE:val]

Where 'Symbol' is a V$XXXX visibility symbol name assigned by the appropriate CAD (.CDF) data file.

'Reg' is the name of a visibility register defined within HEX or by the TRACE DEFINE command. A visibility register is a group of visibility bits joined together in some logical arrangement. See appendix for a list of pre-defined visibility register names.

'Hex_id' is a 16-bit SDB or REGister ID. SDB_ID's are hardware visibility-bit addresses and are defined in the .CDF files. REG_ID's are software-defined numerical tags for visibility registers.

SPAN and VALUE:val are optional breakpoint parameters.

Description:

SET ABREAK sets a break condition in the 'and' break table.
SET OBREAK sets a break condition in the 'or' break table.

Up to 4 breakpoints can be set in each table.

If the signal or register specified is already in the table, the user will be asked if he/she wants to alter the parameters of that table entry. If so, the new parameters are accepted.

When tracing with breakpoints set, the TMICRO and TSTATE commands will stop and report the current machine state only when a break condition is met. This occurs when all breakpoints in the 'and' table are true, or when any condition in the 'or' table is true.

The 'SPAN' option causes a break condition to occur when that signal or register is in a state other than what it was when the trace was started.

The 'VALUE:val' option causes a break condition to occur when that signal or register equals (=) the value specified.

If neither option is selected, 'CHANGE' is implied, and a break will occur every time that signal or register changes.

Examples:

>>>>SET ABREAK V$A123   ; Break when V$A123 changes...

>>>>SET ABREAK V$E456 VAL:1   ; ...and V$E456 = 1.

>>>>SET OBREAK WBUS SPAN   ; Break if WBUS is different than it
                               ; is now

9.5.15 SET HISTORY

Command Syntax:

SET HISTORY [argument]

Where 'argument' can be any ONE of the following keywords and, where applicable, switches. If no keyword is specified the default of CONTINUOUS is assumed.

    CONTINUOUS
    EVENT-STOP [/LATE] [/POSTMORTEM]
    FULL-STOP
    UPC-STOP hex_upc
    UTRAP-STOP
    STALL
    NOSTALL

Description:

This command provides control of the optional HIStory module, which is used mainly by engineering in the development of microcode. It is not expected that the HIStory module will be available for use

outside of Large Vax Engineering.

Any time the SET HISTORY command is issued, whether it includes an argument or not, the HISTORY module is reset and armed to begin recording as soon as the CPU clocks are started. It also stops the CPU clock while the command is executing, and restores the clock to its original state (i.e., running or not running) upon completion of the command. Also, when the command completes the current setting of the HISTORY module (operation mode) is displayed (implicit SHOW HISTORY).

It is not possible to combine operating modes; only one mode can be enabled at a time, with the exception of the STALL/NOSTALL mode which can be combined with any other command mode.

SET HISTORY

> This command, with no arguments, selects the default HISTORY module operator mode (CONTINUOUS, STALL). These default operating parameters are also set by the DEBUG command (when DEBUG mode is entered).

SET HISTORY CONTINUOUS

> This command selects the mode where the HISTORY module continuously records Ebox UPC data.

SET HISTORY EVENT-STOP [/LATE] [/POSTMORTEM]

> This command selects the mode where the HISTORY module stops recording Ebox UPC's after detecting an external event sensed at one of the modules two external (backplane jumper) inputs (one for active high signals, the other for active low signals). If the /POSTMORTEM switch is specified then Ebox UPC tracing will continue for another 512. cycles.

> The sampling of the external inputs occurs, by default, at time T0A and again at time T1A. The /LATE switch specifies that the sampling of the external inputs to occur at time T2A and time T3A.

SET HISTORY FULL-STOP

> This command selects the mode where the HISTORY module stops recording Ebox UPC's after filling the history module ram (1024. entries).

SET HISTORY UPC-STOP hex_upc

> This command selects the mode where the HISTORY module stops recording Ebox UPC's after recording the UPC value specified.

SET HISTORY UTRAP-STOP

This command selects the mode where the HISTORY module stops recording Ebox UPC's when an Ebox UTrap is detected. Ebox Utraps are sensed by the HISTORY module directly via a back-panel connection.

SET HISTORY STALL

This command selects the mode where the HISTORY module stops (stalls) recording Ebox UPC's while the Ebox is stalled. The setting of this "flag" is maintained and unaffected by other SET HISTORY commands, except for the SET HISTORY NOSTALL command.

SET HISTORY NOSTALL

This command selects the mode where the HISTORY module does not stop recording Ebox UPC's while the Ebox is stalled. The setting of this "flag" is maintained and unaffected by other SET HISTORY commands, except for the SET HISTORY STALL command.

9.5.16   SET MARGINS

Command Syntax:

SET MARGIN { HIGH, LOW, NORMAL } [{ A, B, C, DE, FH, ALL }]

where the regulator outputs to margin are specified by the second argument. If the second argument is not specified then the default of ALL is assumed (all regulator outputs set to the desired margin level).

Description:

This command allows the operator to adjust the output level of the power supplies for purposes of detecting marginally active faults. Power supply margins can be set to their nominal values (NORMAL), or 5% below normal (LOW), or 5% above normal (HIGH). The D and E regulators, and the F and H regulators, are tied together since independent marginning of these is not possible.

This command requires that the EMM is functioning.

9.5.17   SHOW BREAK

Command syntax:

SHOW BREAK

Description:

This command displays the contents of the ABREAK and OBREAK tables.
The signal or register id, name, and optional break condition are
included for each table entry.

See the SET xBREAK command for more information about trace
breakpoints.

9.5.18   SHOW DEFINE

Command syntax:

SHOW DEFINE { Reg_name, Hex_id }

Where 'Reg_name' is the name of a visibility register defined within
        HEX or by the TRACE DEFINE command. A visibility register
        is a group of visibility bits joined together in some
        logical arrangement. See appendix for a list of
        pre-defined visibility register names.

    'Hex_id' is a 16-bit SDB or REGister ID. SDB_ID's are
        hardware visibility-bit addresses and are defined in the
        .CDF files. REG_ID's are software-defined numerical tags
        for visibility registers.

Description:

This command displays a list of all visibility symbol names
(V$NNNN) used to construct the visibility register. The display
is oriented in descending order (i.e. high order to low order, left
to right) with 12. terms per line, and shows the last 4 characters
of each V$NNNN symbol.

Example:

    >>>>SHOW DEFINE REGNAM
        NNNN   REGNAM   13.   ! Id, name, size
        1001 2002 3003 4004 5005 ... 1212
        1313

9.5.19   SHOW HISTORY

Command Syntax:

        SHOW HISTORY [/OUTPUT:filename[.DAT]] [hex_num] [hex_num]

Description:

This behavior of this command is unpredictable if the optional
HIStory module is not in place in the system.

This command, whether an argument is specified or not, stops the CPU
clocks and restores them to the original state upon completion.

The "/OUTPUT:filename" switch can be applied to any of the
following command syntaxes. When used, it causes all output
generated by the command to go to the specified filename. Each
time the switch is used a new file by the specified name is
created; no file concatenation is performed.

SHOW HISTORY

    This command displays the current operation mode of the
    HISTORY module, as set by the last SET HISTORY command.

SHOW HISTORY hex_num

    This command displays the last "hex_num" entries added to
    the HISTORY module file. In most cases, the last recorded
    entry will be that of the "event" which caused HISTORY module
    recording to stop. If the operation mode was "EVENT-STOP
    /POSTMORTEM" then the data displayed begins at the event
    (512. entries from the last recorded).

    If "hex_num" is greater than the range of valid data
    currently contained in the HISTORY module ram file then the
    display is truncated where data becomes invalid.

SHOW HISTORY hex_num, hex_num

    This command treats the HISTORY module ram file as a ram
    device with an address range of 0 to 3FF. The data in the
    ram file between the two specified hex_numbers is displayed,
    inclusively. No checking is done to determine if the data is
    within the range of valid data in the HISTORY module file.
    This is essentially a way to read absolute-addressed data
    from the ram file.

9.5.20   SHOW NAME

Command syntax:

SHOW NAME { Symbol_name, Reg_name, Hex_id, "Signal_name" }

Where 'Symbol_name' is a V$XXXX visibility symbol name assigned by
        the appropriate CAD (.CDF) data file.

    'Reg_name' is the name of a visibility register defined
        within HEX or by the TRACE DEFINE command. A visibility
        register is a group of visibility bits joined together in
        some logical arrangement. See appendix for a list of
        pre-defined visibility register names.

'Hex_id' is a 16-bit SDB or REGister ID. SDB_ID's are hardware visibility-bit addresses and are defined in the .CDF files. REG_ID's are software-defined numerical tags for visibility registers.

"Signal_name" is the full visibility signal name, including H/L specifier, enclosed in double quotes.

Description:

This command does a look-up of the symbol_name, register_name, id or signal name and prints the corresponding V$ symbol, signal/register name and id. If showing a register, the size (number of V$ terms defined) is also displayed.

Examples:

```
>>>>SHOW NAME V$5163          ; Use a symbol name
   5145  V$5163 IBD DRAM LAST H

>>>>SHOW NAME STALL           ; Use a register name
   802C  STALL  29.

>>>>SHOW NAME 8000            ; Use a register id
   8000  EUPC  13.

>>>>SHOW NAME "IBD DRAM LAST H" ; Use a signal name
   5145  V$5163 IBD DRAM LAST H
```

9.5.21   SHOW REGISTER

Command syntax:

SHOW REGISTER

Description:

This command displays a list of all visibility registers currently defined, whether they are defined internally or by the TRACE DEFINE command. The display includes each register's id_number, name, and size.

Note: A validity check is done on each register before it is displayed. If a bad entry in a register is found the name "- badreg -" is assigned to the register. Because of the bad register definition there is no way for the code to display the true name of the register.

9.5.22   SHOW TRACE

Command syntax:

    SHOW TRACE

Description:

This command displays a list of all registers and signals currently
selected for tracing. The trace list may be altered by the user
using the TRACE command. The current breakpoint options (if any)
are also included in this display.

9.5.23   STATESTEP

Command syntax:

STATESTEP [hex_num]

Where 'hex_num' is a hexadecimal step count in the range of 1 to
400(hex). If unspecified, 1 step is assumed.

Description:

The STATESTEP command causes 'hex_num' clock ticks to be executed.
A STATESTEP of 4 is equivalent to a MICROSTEP of 1, but the UPC of
the boxes is not displayed. Statesteps are executed at full
machine speed.

If the SBIA Visibility Module is present this command attempts to
tick the SBIA clock as well. If the STATESTEP count is greater
than 1 then a single SBIA clock tick is done at the completion of
the CPU clock ticks. For a single cpu tick no SBIA cycle is done,
but additional cpu ticks invoked in space-bar-step mode will cause
the SBIA clock to be ticked once for every other cpu clock tick.

The execution of the STATESTEP command invokes Space-bar-step mode
(indicated by the SBSM>> or SBSMDC>> prompts). In Space-bar-step
Mode the user can cause additional steps to be executed by hitting
the space-bar (one statestep per space character). Space-bar-step
mode remains in effect until something other than a space is input.
The SBIA clock is stepped every other time through the space-bar-
step cycle (only if the SBIA Visibility Module is present).

CPU clock status information is displayed at the end of each step
command. The format of the data is as follows:

    bit <15:15> - state of CLK RATE FULL (0 = 1/5th rate)
    bit <14:08> - clock frequency setting in Mhz

    bit <07:06> - reserved for future use
    bit <05:05> - state of MARK STOP FLAG bit (1 = stopped)

bit <04:04> - state of CLK RUN bit (if set, phase info invalid)

bit <03:03> - T3 phase status bit (1 = active)
bit <02:02> - T2 phase status bit (1 = active)
bit <01:01> - T1 phase status bit (1 = active)
bit <00:00> - T0 phase status bit (1 = active)

9.5.24 TMICRO, TSTATE

Command syntax:

TMICRO [hex_num]

TSTATE [hex_num]

Where 'hex_num' is a hexadecimal step-count. If unspecified, 1
is assumed (if no trace breakpoints are set); else, stepping
continues indefinitely until a break condition or ^C is entered.

Description:

These commands initialize the trace facility and execute CPU clock
steps (1 clock cycle for TMICRO, 1 clock phase for TSTATE) while
watching for a trace breakpoint condition, or until the step count
expires. The state of the machine is then captured and reported in
accordance with the current trace settings. Before MICRO stepping
the CPU clock is always forced to the T3 phase.

If the SBIA Visibility Module is present this command attempts to
step the SBIA clock as well. If the step count is greater than 1
then a single SBIA clock cycle (or tick for TSTATE) is done at the
completion of the CPU clocks. For a single cpu cycle no SBIA cycle
is done, but additional cpu cycles invoked in space-bar-step mode
will cause the SBIA clock to be cycled once for every other cpu cycle.

If any trace breakpoints are set (see SET xBREAK command) trace
stepping stops, and the trace report generated, as soon as a break
condition is met. Otherwise, stepping continues until the step
count expires (if specified). If no step count is given, and trace
breakpoint conditions are defined, then stepping continues
indefinitely until either a breakpoint or ^C input character is
detected.

When a breakpoint is detected, or the step count expires, HEX
enters Space-bar-step mode (indicated by the SBSM>> or SBSMDC>>
prompts). In Space-bar-step Mode, the user can cause additional
steps to be executed and traced by hitting the space-bar (one step
per space-bar character). Space-bar-step mode remains in effect
until something other than a space is input to the SBSM prompt.

The current setting of the QUIET flag affects the output of this
command. If QUIET is ON, no trace information is displayed until
the final trace step has completed; if QUIET is OFF then trace

information is displayed for each step.

Note that the combination of the MICROSTEP and REPORT commands is equivalent to the TMICRO command. A similar relationship is true for TSTATE.

9.5.25 TRACE ADD

Command syntax:

>   TRACE ADD item
>
>   Where 'item' is any combination of the following identifiers:
>
>   Reg_name -- add named registers to the trace list
>   V$nnnn -- add signal corresponding to symbol name to list

Description:

This command is used to customize the register and signal trace lists. The list of items to add is included in the command line and may specify as many items (separated with spaces) as will fit in the remainder of that line. Reg_names and V$ symbols may be intermixed at will.

Note that the V$ prefix IS REQUIRED on all symbol name specifications.

Available register names can be seen using the SHOW REGISTER command.

Example:

>   >>>>TRACE ADD EUPC USRREG PARITY V$1001 V$2345 UPCSAV

9.5.26   TRACE DEFINE

Command syntax:

TRACE DEFINE Reg_name

Where 'Reg_name' is a unique name to be assigned to the new user-defined register. Names strings must be 6 characters or less in length (any excess will be truncated).

Description:

This command is used to define visibility registers in addition to those already in the pre-defined (canned) set. The command will check to see if the new name is unique (not already defined) and if so prompt the user to enter a list of V$ symbol names comprising the definition. Only the last 4 characters of the 6 character V$NNNN

symbol are accepted (do not include the V$ prefix). Symbols must
be input beginning with the most-significant in the register.
Input of the symbol names can be continued on another line by
entering a CR when the present line is full. A CR on a null input
line terminates the definition.

The symbol names used in the new register are then verified, and if
ok the register name and size is displayed ala SHOW NAME.

User-defined registers are automatically added to the register
trace list (i.e., automatic TRACE ADD reg_name is done).

A user-defined visibility register should not exceed 48 (decimal)
symbol names if it is to be traced correctly (without truncation).
The maximum number of user-defined visibility registers is only
limited by the amount of space required to define them, and this
limit varies between releases of console software.

Example:

```
 >>>>TRACE DEFINE newregister
    Enter V$ terms separated with <sp> or <,>    ! command prompt.
    1001 2002 3003 ...                           ! Users symbol list...
    <cr>                                         !...end of definition.
    NNNN   NEWREG  N.                            ! Verification display.
 >>>>
```

9.5.27   TRACE DELETE

Command syntax:

TRACE DELETE Reg_name

Where 'Reg_name' is the name of a previously defined user
        visibility register.

Description:

This command may be used to delete only user-defined visibility
registers previously established with TRACE DEFINE.

The command will ask for confirmation before taking any action.
If the delete request is confirmed, the named register is deleted
from the user-defined register list and also removed from the
register trace list (i.e., automatic TRACE REMOVE is done).

Registers in the pre-defined (canned) set may not be deleted.

Example:

```
 >>>>TRACE DELETE newreg
    Delete register NEWREG -- comfirm (Y/N) y    ! comfirm response.
```

>>>>

9.5.28   TRACE REMOVE

Command syntax:

TRACE REMOVE item

Where 'item' can be any combination of the following identifiers:

    R* -- remove all registers from the list
    S* -- remove all signals from the list
    Reg_name -- remove named registers from the list
    V$nnnn -- remove signal corresponding to this symbol from list

Description:

This command is used to customize the register and signal trace
lists by allowing unwanted information to be removed. The list of
items to remove is included in the command line and may specify as
many items (separated with spaces) as will fit in the remainder of
that line. Reg_names and V$ symbols may be intermixed at will.

Note that the V$ prefix IS REQUIRED on all symbol name specifications.

Example:

>>>>TRACE REMOVE REG1 REG2 S* REG3

9.5.29   TRACE RESTORE

Command syntax:

TRACE RESTORE

Description:

This command is used to restore the default trace list (including
all canned visibility registers and signals) and zero all counters
and variables.

CHAPTER 10

PROGRAM I/O MODE

In PIO mode the console program services the requests of the VAX CPU and provides special monitoring functions to the overall system. PIO mode can only be entered from the MACRO context prompt using the START or CONTINUE commands. PIO mode is entered automatically at the completion of a console reboot, or after console prorgam initialization if an automatic warm-start or cold-start attempt is to be made.

PIO mode is exited by a ^P typed on a terminal enabled for ^P, or if the CPU HALTS (such as for a KAF). PIO mode always exits to CIO mode, MACRO context. When CIO mode is entered an information message is printed on the terminal to alert the operator that the CPU is still running. Leaving a system in this state is not recommended since it could hang VMS processes trying to use the RX, TX, or STX registers. Also, the console will not perform control store parity error correction until PIO mode is re-entered.

While running in PIO mode the console program does not accept any console commands from the operator. Anything typed on the CTY or RTY is passed to the CPU.

The following functions are performed by the console program while in PIO mode.

- o independent character transfers between the CPU and the two console terminal ports.

- o handles the passing of requests to the EMM from the CPU, and passes responses and unsolicited warning messages from the EMM to the CPU

- o handles "logical console" requests from the CPU

- o handles CPU control store parity error correction (where possible)

- o handles detection and recovery of CPU ERROR HALTS and CPU KEEP ALIVE FAILURES

o   handles CI reboot operations (abus dead interrupt)

o   handles detection of and recovery from power failures.

o   provides CPU access to the console RL pack

o   provides time-of-year (TOY) information to the CPU with 10ms
    resolution

o   handles front panel LED indicator updating

10.1   IPR'S HANDLED BY CONSOLE DURING PIO MODE

     Some of the IPR's have more than a passing relationship with
console software in that they are implemented by the console program
in whole or in part.   In general, IPR's handled by the console program
are done so only while PIO mode is running.   This is one of the
reasons why it is not advised to leave the CPU running while not in
PIO mode.   All such IPR's, with the exception of the SID register, are
implemented in the CBUS RAM space.

     The IPR's handled by the console program are described in the
following sub-sections.

10.1.1   SID

     Location DF in the EBOX scratch pad is loaded with values
supplied to the console via back plane jumpers on the console slot.
These values indicate the serial number and plant of origin of the
machine.   The high order byte of the register is loaded with a value
of four (4) indicating the CPU type as an 8600.   Accessing the SID
with a CPU instruction (MFPR) or console EXAMINE command causes this
location in EBOX scratch pad to be accessed.

10.1.2   CRBT

     This IPR is write-only by the CPU and causes a high-priority
interrupt to the console whenever it is accessed.   The T-11 senses
this interrupt and vectors to the REBOOT entry point of the PROM code.
The TSTRT interrupt, as it is refered to by the console prints, enters
at the PR8 level, which means it is not maskable by the T-11's
priority level logic in the PSW.   The TSTRT interrupt can be masked
only be disabling all T-11 interrupts, which is useful to the console
diagnostic.

10.1.3 TODR

This register provides the CPU with time-of-day/year information at a resolution of 10ms. The handling of TODR is done entirely by the RT-11 kernel. Examines and deposits to this IPR work correctly while in CIO mode.

10.1.4 TXCS And TXDB

These registers can be used by CPU macrocode while using the NEXT command in CIO mode. DEPOSIT's to these registers will have UNPREDICTABLE affects while in CIO mode.

Refer to the 8600 REGISTER SPECIFICATION for a complete description of how the TX register operates. In general, the function of the TXCS register is to allow the CPU with a means of selecting a set of zero to four transmitter lines that are to be enabled. The console will multiplex data through the TXDB register in accordance with the enabled lines. Whenever the TXDB register or the mask field of the TXCS register is written the console program is interrupted.

The four lines available to the CPU through the TX register allow it to send data to one of four console devices, without the need to wait for a slow device to complete a transfer before a fast device can get another byte of data. The CPU may send data bytes to the console's CTY, the console's RD, the EMM, or to the "logical" console device.

10.1.5 RXCS And RXDB

These registers can be used by CPU macrocode while using the NEXT command in CIO mode. DEPOSIT's to these registers will have UNPREDICTABLE affects while in CIO mode.

Refer to the 8600 REGISTER SPECIFICATION for a complete description of how the RX register operates. In general, the function of the RXCS register is to allow the console to interrupt the CPU when it has placed a data byte into the RXDB register. RXCS also serves the CPU by allowing it to enable or disable MODEM access through the RTY port. The RXDB register is setup by the console program when a data byte is to be passed to the CPU. The ID field in RXDB indicates the origin/destination of the data byte (i.e., RTY, EMM, etc.). The CARRIER field of the RXDB register is also controlled by the console program.

10.1.6 STXCS And STXDB

These registers can NOT be used by CPU macrocode while using the
NEXT command in CIO mode.   DEPOSIT's to these registers will have
UNPREDICTABLE affects while in CIO mode.

Refer to the 8600 REGISTER SPECIFICATION for   a   complete
description of how the STX register operates. In general, the purpose
of the STXCS register is to allow the CPU to interrupt the console
with a function code and logical block number at which to perform a
transfer between the console pack and the STXDB register. Status of
the transfer is also included in the STXCS register by the console
program.

APPENDIX A

SAMPLE OF DEFBOO.COM FILE

DEFBOO.COM is the command file submitted for execution by the BOOT command when no argument is specified. The purpose of DEFBOO is to boot the operating system's primary bootstrap code from the console pack. The primary bootstrap code will then load the operating system kernel from the default disk device.

```
!
! Load VMB.EXE from the console pack.
!
! Operating system Disk:          RK06/RK07
!
!
SET SNAP ON              ! Enable CPU snapshots on error
SET FBOX OFF             ! Operating system will enable Fbox when ready
INIT                     ! DEC STD 032 processor init
UNJA                     ! UNJAM all I/O adapters found
DEPOSIT CSWP 8           ! Turn off the cache

DEPOSIT R0 1             ! Device Type is RK06/7
DEPOSIT R1 4             ! Abus adapter #0; TR number of adapter is 4
DEPOSIT R2 3FF20         ! Unibus address of the device's CSR
DEPOSIT R3 0             ! Unit number to boot from
DEPOSIT R4 0             ! Logical block number to boot from (if R5 bit 3 = 1)

FIND/MEMORY             ! Locate a 64KB chunk of good memory
EXAMINE SP              ! Display load address
LOAD/START:@ VMB        ! Load VMB 200 bytes above the start of the good block
START @                 ! Start VMB at the load address
```
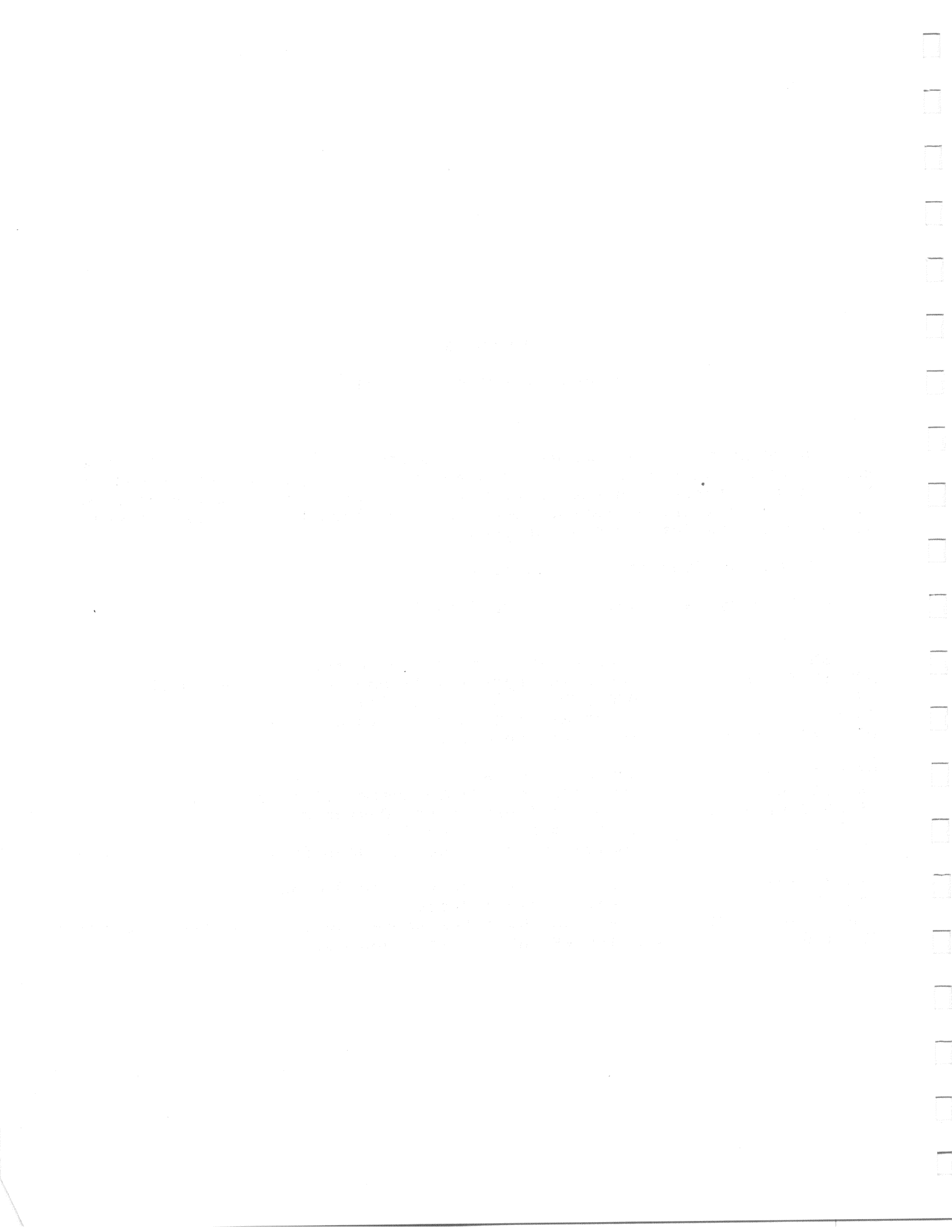
APPENDIX B

SAMPLE CLOCK.COM, LOAD.COM AND ULOAD.COM FILES

The CLOCK.COM file is executed immediately after the power system is initialized whenever macro context is entered. Unlike other command files, this one cannot be aborted by ^C.

```
!
! CLOCK.COM - init clock and define clock parameters
!
SET CLOCK X1 40          ! Assign mnemonics for rev E01/F01 clock modules
SET CLOCK X2 50/NORMAL
SET CLOCK X3 53/HIGH
SET CLOCK X5 56
SET CLOCK FREQ NORMAL     ! Force normal clock operation
SET CLOCK FULL            ! Full speed
SET CLOCK DEFAULT         !...and make it stick
INIT/CLOCK                ! Initialize clock logic and do 141 reset
```

The file LOAD.COM is submitted as part of the MACRO context initialization immediately after the "Initializing CPU" message. Its purpose is to init the CPU so that macrocode can be run.

```
!
! LOAD.COM file
!
RESET                   ! Stop the CPU clock
INIT/POWER              ! Check for solid power system
INIT/SDB                ! Init control channels for first time
INIT/MICRO              ! Load system microcode (submits ULOAD.COM)
INIT/SDB                ! Init control channels after loading control
                        ! stores.
INIT/CPU                ! Init CPU/CSM/E Scratch
INIT/PAMM               ! Init PAMM and I/O adapters
```

The command INIT/MICRO submits the file ULOAD.COM for execution. ULOAD performs the commands necessary for loading default system microcode into all control stores of the machine.

```
!
! ULOAD.COM file
!
SET EXTI OFF            ! Turn off CPU-to-console interrupts
LOAD/ECS               ! Load the Ebox control stores (KA86n0.BPN)
LOAD/MCF               !   (MCF.BPN)
LOAD/CONTEXT           !   (CTX.BPN)
```

```
LOAD/ICS          ! Load the Ibox control stores (IBOX.BPN)
LOAD/IDRAM        !   (KA86n0.BPN)
LOAD/MCS          ! Load the Mbox control stores (UCODEn.BPN)
LOAD/ACCESS       !   (ACCESS.BPN)
LOAD/CYCLE        !   (CYCLE.BPN)
LOAD/FBA          ! Load the Fbox control stores (FADDn.BPN)
LOAD/FBM          !   (FMUL.BPN)
LOAD/FDRAM        !   (FADDn.BPN)
```

APPENDIX C

CONSOLE MODULE INTERRUPT SYSTEM

The interrupt system of the console module is summarized in the following table. Note that the signal "CL09 ENA T11 INTR H" must be asserted before ANY of these interrupts can be seen by the T11 processor.

L0201 PROGRAMMED INTERRUPT VECTORS AND PRIORITIES:

Vector	Priority	Device Name
24	8 (UNMASKABLE)	MANUAL RESTART (external switch input)
140	7	CONSOLE RAM PARITY ERROR
144	7	unused
150	7	SYSTEM AC LOW
154	7	ABUS DEAD
100	6	unused
104	6	TOY 1ms INTERVAL
110	6	CPU CONTROL STORE PARITY ERROR
114	6	STOR RDY
120	5	TXCS RDY
124	5	RXCS DNE
130	5	QBUS ADAPTER
134	5	EMM PCI TRANSMIT/RECIEVE
60	4	REMOTE PCI TRANSMIT/RECIEVE/MODEM
64	4	LOCAL PCI TRANSMIT/RECIEVE
70	4	QBUS REPLY TIMEOUT

The CL15 TSTRT interrupt can be generated by the CPU or by the console command 'REBOOT' and causes the T11 to jump directly to prom address 172004. The prom code will then reinitialize the console logic (without affecting the state of the CPU) and reboot the console software.
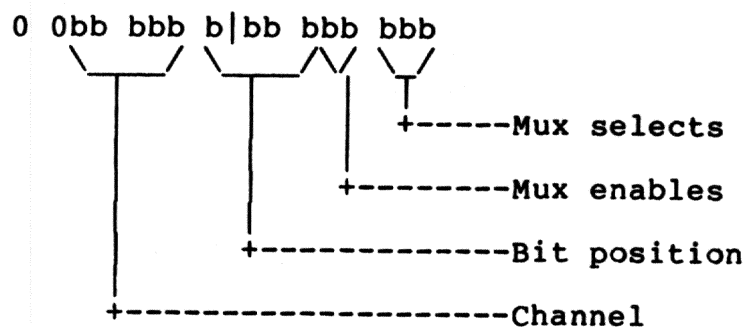
The CL02 MAN RESTART interrupt is an external input on the console back plane. The T11 treats this as a TRANSITION interrupt (as it does with TSTRT) through vector 24, which console software has initialized to point to prom address 172010 (interrupt handler). Prom code then displays the state of the T11 processor and its prompt (ROM>), and waits for further input.

All other interrupts are LEVEL sensitive inputs to the T11.

APPENDIX D

SDB ID FORMAT

The format of an SDB id is shown in the following diagram. These
id's are used internally in the console software for locating
visibility data but can also be used as arguments to such commands as
SHOW NAME and EXAM/SDB.

```
    0 0bb bbb b|bb bbb bbb
       \___/ \_____/\_/ \_/
         |       |    |   |
         |       |    |   +-----Mux selects
         |       |    +--------Mux enables
         |       +------------Bit position
         +--------------------Channel
```

DEFAULT VISIBILITY REGISTERS

The HEX debugger command set provides access to a number of pre-defined visibility registers useful for tracing the machine state. Most of these registers are part of the default TRACE list used by the TMICRO, TSTATE, and REPORT commands. The TRACE ADD command can be used to add a register to the TRACE list that is not already there. The TRACE DEFINE command can be used to define a new visibility register. The following list summarizes the various visibility registers present in the console program.

ABUS	ARADR	ARBUS	ARYBUS	DBUS
EBFLSH	EDPPE	EFORK	EMCF	ESTALL
EUPC	EVABUS	FABUS	FAUPC	FMUPC
IBDBUF	IBUF	IBXERR	IDIAG	INCR
IOPSEL	IUPC	IVABUS	MDBUSI	MDBUSM
MEMREQ	MUPC	NATRAM	OPAR	OPBUS
OPCODE	OPMCF	OPPORT	PAACK	PAMD
PAMM	PARITY	PSL	REGBUS	STALL
UPCSAV	WBUS			

The above information is not complete without a breakdown of what each register contains. So, the following is a complete list of visibility registers and the signal names that make them up. Note that bits are listed in order of the MOST-SIGNIFICANT BIT FIRST. The special filler symbol 'XXXX' is used to 0-pad a visibility register in places where signals are no longer defined or where nibble aligment is desired. If the accuracy of a visibility register definition is in question the actual definition can be found in the source listing for the HEXBOX module (HEXBOXT11.LST).

E.1 ABUS

Name:	V$ symbol:	Bit:	Nibble:	Signal name:
ABUS	V$A223	43	11	SB ABUS IOA REQUEST 3 H
	V$A225	42		SB ABUS IOA REQUEST 2 H
	V$A215	41		SB ABUS IOA REQUEST 1 H
	V$A217	40		SB ABUS IOA REQUEST 0 H

V$A130	39	10	ABUS CPU BUF DONE H
V$A226	38		ABUS CPU BUF ERROR H
V$A220	37		ABUS LEN STAT 1 H
V$A222	36		ABUS LEN STAT 0 H
V$A211	35	9	ABUS CMD MASK 3 H
V$A214	34		ABUS CMD MASK 2 H
V$A219	33		ABUS CMD MASK 1 H
V$A210	32		ABUS CMD MASK 0 H
V$B195	31	8	ABUS DATA ADDRS 31 H
V$B203	30		ABUS DATA ADDRS 30 H
V$B139	29		ABUS DATA ADDRS 29 H
V$B128	28		ABUS DATA ADDRS 28 H
V$B199	27	7	ABUS DATA ADDRS 27 H
V$B197	26		ABUS DATA ADDRS 26 H
V$B198	25		ABUS DATA ADDRS 25 H
V$B183	24		ABUS DATA ADDRS 24 H
V$B200	23	6	ABUS DATA ADDRS 23 H
V$B130	22		ABUS DATA ADDRS 22 H
V$B204	21		ABUS DATA ADDRS 21 H
V$B131	20		ABUS DATA ADDRS 20 H
V$B189	19	5	ABUS DATA ADDRS 19 H
V$B135	18		ABUS DATA ADDRS 18 H
V$B138	17		ABUS DATA ADDRS 17 H
V$B136	16		ABUS DATA ADDRS 16 H
V$B140	15	4	ABUS DATA ADDRS 15 H
V$B190	14		ABUS DATA ADDRS 14 H
V$B141	13		ABUS DATA ADDRS 13 H
V$B191	12		ABUS DATA ADDRS 12 H
V$B127	11	3	ABUS DATA ADDRS 11 H
V$B196	10		ABUS DATA ADDRS 10 H
V$B208	09		ABUS DATA ADDRS 09 H
V$B193	08		ABUS DATA ADDRS 08 H
V$B194	07	2	ABUS DATA ADDRS 07 H
V$B169	06		ABUS DATA ADDRS 06 H
V$B209	05		ABUS DATA ADDRS 05 H
V$B206	04		ABUS DATA ADDRS 04 H
V$B205	03	1	ABUS DATA ADDRS 03 H
V$B168	02		ABUS DATA ADDRS 02 H
V$B176	01		ABUS DATA ADDRS 01 H
V$B177	00		ABUS DATA ADDRS 00 H

E.2   ARADR

Name:	V$ symbol:	Bit:	Nibble:	Signal name:
ARADR	V$K103	28	8	MAP1 ARRAY ADR 28 H
	V$K105	27	7	MAP1 ARRAY ADR 27 H
	V$K101	26		MAP1 ARRAY ADR 26 H
	V$K179	25		MAP1 ARRAY ADR 25 H
	V$K174	24		MAP1 ARRAY ADR 24 H
	V$K177	23	6	MAP1 ARRAY ADR 23 H
	V$K173	22		MAP1 ARRAY ADR 22 H
	V$K171	21		MAP1 ARRAY ADR 21 H
	V$K159	20		MAP1 ARRAY ADR 20 H
	V$K162	19	5	MAP1 ARRAY ADR 19 H
	V$K161	18		MAP1 ARRAY ADR 18 H
	V$K157	17		MAP1 ARRAY ADR 17 H
	V$K156	16		MAP2 ARRAY ADR 16 H
	V$K155	15	4	MAP2 ARRAY ADR 15 H
	V$K151	14		MAP2 ARRAY ADR 14 H
	V$K154	13		MAP2 ARRAY ADR 13 H
	V$K148	12		MAP2 ARRAY ADR 12 H
	V$K147	11	3	MAP2 ARRAY ADR 11 H
	V$K143	10		MAP2 ARRAY ADR 10 H
	V$K146	09		MAP2 ARRAY ADR 09 H
	V$K142	08		MAP2 ARRAY ADR 08 H
	V$K145	07	2	MAP2 ARRAY ADR 07 H
	V$K141	06		MAP2 ARRAY ADR 06 H
	V$K140	05		MAP2 ARRAY ADR 05 H
	V$K139	04		MAP2 ARRAY ADR 04 H
	V$K135	03	1	MAP2 ARRAY ADR 03 H
	V$XXXX	02		Zero-Filler
	V$XXXX	01		Zero-Filler
	V$XXXX	00		Zero-Filler

E.3   ARBUS

Name:	V$ symbol:	Bit:	Nibble:	Signal name:
ARBUS	V$K113	31	8	ARRAY BUS D31 H
	V$K112	30		ARRAY BUS D30 H
	V$K106	29		ARRAY BUS D29 H
	V$K107	28		ARRAY BUS D28 H
	V$K108	27	7	ARRAY BUS D27 H
	V$K102	26		ARRAY BUS D26 H

V$K175	25		ARRAY BUS D25 H
V$K178	24		ARRAY BUS D24 H
V$K104	23	6	ARRAY BUS D23 H
V$K100	22		ARRAY BUS D22 H
V$K176	21		ARRAY BUS D21 H
V$K172	20		ARRAY BUS D20 H
V$K167	19	5	ARRAY BUS D19 H
V$K169	18		ARRAY BUS D18 H
V$K166	17		ARRAY BUS D17 H
V$K170	16		ARRAY BUS D16 H
V$K131	15	4	ARRAY BUS D15 H
V$K127	14		ARRAY BUS D14 H
V$K129	13		ARRAY BUS D13 H
V$K130	12		ARRAY BUS D12 H
V$K126	11	3	ARRAY BUS D11 H
V$K125	10		ARRAY BUS D10 H
V$K128	09		ARRAY BUS D09 H
V$K119	08		ARRAY BUS D08 H
V$K123	07	2	ARRAY BUS D07 H
V$K124	06		ARRAY BUS D06 H
V$K122	05		ARRAY BUS D05 H
V$K118	04		ARRAY BUS D04 H
V$K121	03	1	ARRAY BUS D03 H
V$K117	02		ARRAY BUS D02 H
V$K120	01		ARRAY BUS D01 H
V$K116	00		ARRAY BUS D00 H

E.4   DBUS

Name:	V$ symbol:	Bit:	Nibble:	Signal name:
DBUS	V$3271	31	8	-DBUS D31 H
	V$3262	30		-DBUS D30 H
	V$3272	29		-DBUS D29 H
	V$3266	28		-DBUS D28 H
	V$3170	27	7	-DBUS D27 H
	V$3225	26		-DBUS D26 H
	V$3241	25		-DBUS D25 H
	V$3240	24		-DBUS D24 H
	V$3270	23	6	-DBUS D23 H
	V$3263	22		-DBUS D22 H
	V$3273	21		-DBUS D21 H
	V$3275	20		-DBUS D20 H

V$ symbol	Bit	Nibble	Signal name
V$3171	19	5	-DBUS D19 H
V$3187	18		-DBUS D18 H
V$3237	17		-DBUS D17 H
V$3236	16		-DBUS D16 H
V$3269	15	4	-DBUS D15 H
V$3268	14		-DBUS D14 H
V$3274	13		-DBUS D13 H
V$3267	12		-DBUS D12 H
V$3220	11	3	-DBUS D11 H
V$3222	10		-DBUS D10 H
V$3239	09		-DBUS D09 H
V$3242	08		-DBUS D08 H
V$3176	07	2	-DBUS D07 H
V$3178	06		-DBUS D06 H
V$3105	05		-DBUS D05 H
V$3104	04		-DBUS D04 H
V$3145	03	1	-DBUS D03 H
V$3144	02		-DBUS D02 H
V$3243	01		-DBUS D01 H
V$3126	00		-DBUS D00 H

E.5   EBFLSH

Name:	V$ symbol:	Bit:	Nibble:	Signal name:
EBFLSH	V$3198	04	2	-ICA BUF FLUSH MRES3 H
	V$5230	03	1	-ICAB EFLSH FR CPC LAT H
	V$5218	02		-CSB UMCF 2 A H
	V$5242	01		ICA6 IFORK CTL 2 H
	V$5194	00		-CSB UMCF 0 A H

E.6   EDPPE

Name:	V$ symbol:	Bit:	Nibble:	Signal name:
EDPPE	V$9104	43	11	EDP EDPE D3 H
	V$D157	42		EDP EDPE D2 H
	V$9102	41		EDP EDPE D1 H
	V$D156	40		EDP EDPE D0 H
	V$E138	39	10	EDP STATE 7 H
	V$E109	38		EDP STATE 6 H
	V$E177	37		EDP STATE 5 H
	V$E120	36		EDP STATE 4 H

V$E184	35	9	EDP STATE 3 H
V$E136	34		EDP STATE 2 H
V$E110	33		EDP STATE 1 H
V$E178	32		EDP STATE 0 H
V$A180	31	8	ICA ISTALL A H
V$9103	30		EBD RSV MODE H
V$9100	29		ICB RLOG PE H
V$9181	28		ICB IBUF PE H
V$5123	27	7	ICA7 ICS PAR ERR H
V$9101	26		ICB IDRAM PE H
V$9176	25		IDP IAMUX PE H
V$9182	24		IDP IBMUX PE H
V$XXXX	23	6	Zero-Filler
V$6127	22		MCC OP PA ACK B H
V$5167	21		MCC IBF PA ACK H
V$9162	20		MCC MBOX CS PE H
V$5221	19	5	EBD ESTALL TO ICA H
V$5186	18		EBE IBOX ERR LTH A H
V$9170	17		EBD ECS PE FLAG H
V$9110	16		EBD ECS PE LST CYC H
V$D166	15	4	EBD EBOX ERR LST CYC H
V$4167	14		EBD EBOX ERR TO IDP H
V$9170	13		EBD ECS PE FLAG H
V$9110	12		EBD ECS PE LST CYC H
V$D158	11	3	EBD EDP PE FLAG A H
V$9169	10		EBD EDP PE FLAG H
V$9173	09		EBD EMCR PE FLAG H
V$9161	08		EBD EN ETRAP H
V$9174	07	2	EBD USTK PE FLAG H
V$9143	06		EBD WBUS PE FLAG H
V$C159	05		-EBC MCF RAM PAR ERR H
V$C161	04		EBD6 EVC D09 H
V$4139	03	1	-ICA BMUX CHK WITH CTX H
V$0131	02		EBE WBUS OPAR B2 C H
V$0129	01		EBE WBUS OPAR B1 C H
V$4281	00		-DBUS D08 H

E.7   EFORK

Name:	V$ symbol:	Bit:	Nibble:	Signal name:
EFORK	V$6146	08	3	-CSB EBOX FORK E H
	V$0100	07	2	-CSB EBOX FORK D H

V$ symbol	Bit	Nibble	Signal name
V$8124	06		-CSB EBOX FORK C H
V$5159	05		-CSB EBOX FORK B H
V$C226	04		-CSB EBOX FORK A H
V$XXXX	03	1	Zero-Filler
V$C102	02		-CSB EBOX IRD A H
V$6101	01		-CSB EBOX IRD B H
V$1231	00		-CSB EBOX IRD C H

E.8   EMCF

Name:	V$ symbol:	Bit:	Nibble:	Signal name:
EMCF	V$D184	37	10	CSA UMCF 5 A H
	V$D186	36		CSA UMCF 4 A H
	V$D181	35	9	CSA UMCF 3 A H
	V$D183	34		CSB UMCF 2 A H
	V$D180	33		CSB UMCF 1 A H
	V$D182	32		CSB UMCF 0 A H
	V$XXXX	31	8	Zero-Filler
	V$XXXX	30		Zero-Filler
	V$A153	29		EBC EBOX MCF 5 H
	V$A182	28		EBC EBOX MCF 4 H
	V$A174	27	7	EBC EBOX MCF 3 H
	V$A165	26		EBC EBOX MCF 2 H
	V$A164	25		EBC EBOX MCF 1 H
	V$A154	24		EBC EBOX MCF 0 H
	V$XXXX	23	6	Zero-Filler
	V$D154	22		-EBC9 MCF-LOAD N H
	V$D153	21		-EBC9 MCF-LOAD T H
	V$D155	20		-EBC9 MCF-REQUE N H
	V$D151	19	5	-EBC9 MCF-REQUE T H
	V$D146	18		EBCA MCF-EN MBOX H
	V$D140	17		EBCA MCF-WCHK H
	V$D150	16		EBC8 MCF-E DISP I H
	V$C179	15	4	-EBC MCF-ACK REQ LTH H
	V$C187	14		-EBC MCF-CLR EBPS LTH H
	V$C184	13		-EBC MCF-CLR IBPS LTH H
	V$C186	12		-EBC MCF-CLR OPPS LTH H
	V$C180	11	3	-EBC MCF-EN OWSTL LTH H
	V$C178	10		-EBC MCF-MEM REQ LTH H
	V$C127	09		-EBC MCF-MEM WRT LTH H
	V$C199	08		-EBC MCF-OP WRT LTH H
	V$D143	07	2	-EBCA MCF-ACK REQ H

V$D138	06		-EBC8 MCF-CLR EBPS H
V$D149	05		-EBC8 MCF-CLR IBPS H
V$D137	04		-EBC8 MCF-CLR OPPS H
V$D134	03	1	-EBC7 MCF-EN OWSTL H
V$D139	02		-EBC7 MCF-MEM REQ H
V$D104	01		-EBC7 MCF-MEM WRT H
V$D105	00		-EBC7 MCF-OP WRT H

E.9   ESTALL

Name:	V$ symbol:	Bit:	Nibble:	Signal name:
ESTALL	V$E183	11	3	-EBD ESTALL TO CSB H
	V$D171	10		EBD ESTALL TO EBC H
	V$9166	09		EBD ESTALL TO EBE H
	V$8160	08		EBD ESTALL TO EDP H
	V$0190	07	2	EBD ESTALL TO FBA H
	V$1123	06		EBD ESTALL TO FBM H
	V$3157	05		-EBD ESTALL TO IBD H
	V$5221	04		EBD ESTALL TO ICA H
	V$6138	03	1	EBD ESTALL TO ICB H
	V$4126	02		EBD ESTALL TO IDP H
	V$A184	01		EBD ESTALL TO MCC H
	V$2114	00		EBD ESTALL TO MCD H

E.10   EUPC

Name:	V$ symbol:	Bit:	Nibble:	Signal name:
EUPC	V$F122	12	4	CSB UPC 12 H
	V$F123	11	3	CSB UPC 11 H
	V$F111	10		CSB UPC 10 H
	V$F113	09		CSB UPC 09 H
	V$F107	08		CSB UPC 08 H
	V$F112	07	2	CSB UPC 07 H
	V$F115	06		CSB UPC 06 H
	V$F109	05		CSB UPC 05 H
	V$F110	04		CSB UPC 04 H
	V$F114	03	1	CSB UPC 03 H
	V$F126	02		CSB UPC 02 A H
	V$F118	01		CSB UPC 01 A H
	V$F127	00		CSB UPC 00 A H

E.11 EVABUS

Name:	V$ symbol:	Bit:	Nibble:	Signal name:
EVABUS	V$B222	31	8	EDP EVA A31 H
	V$B112	30		EDP EVA A30 H
	V$B187	29		EDP EVA A29 H
	V$B182	28		EDP EVA A28 H
	V$B188	27	7	EDP EVA A27 H
	V$B132	26		EDP EVA A26 H
	V$B201	25		EDP EVA A25 H
	V$B119	24		EDP EVA A24 H
	V$B118	23	6	EDP EVA A23 H
	V$B133	22		EDP EVA A22 H
	V$B134	21		EDP EVA A21 H
	V$B123	20		EDP EVA A20 H
	V$B126	19	5	EDP EVA A19 H
	V$B192	18		EDP EVA A18 H
	V$B122	17		EDP EVA A17 H
	V$B207	16		EDP EVA A16 H
	V$B215	15	4	EDP EVA A15 H
	V$B143	14		EDP EVA A14 H
	V$B146	13		EDP EVA A13 H
	V$B145	12		EDP EVA A12 H
	V$B144	11	3	EDP EVA A11 H
	V$B216	10		EDP EVA A10 H
	V$B213	09		EDP EVA A09 H
	V$B175	08		EDP EVA A08 H
	V$B171	07	2	EDP EVA A07 H
	V$B179	06		EDP EVA A06 H
	V$B173	05		EDP EVA A05 H
	V$B174	04		EDP EVA A04 H
	V$B167	03	1	EDP EVA A03 H
	V$B170	02		EDP EVA A02 H
	V$B159	01		EDP EVA A01 A H
	V$B162	00		EDP EVA A00 A H

E.12 FABUS

Name:	V$ symbol:	Bit:	Nibble:	Signal name:
FABUS				
DRGFAB:	V$1197	31	8	-FBA FA BUS D31 H
	V$1269	30		-FBA FA BUS D30 H
	V$1138	29		-FBA FA BUS D29 H

V$1180	28		-FBA FA BUS D28 H
V$1171	27	7	-FBA FA BUS D27 H
V$1167	26		-FBA FA BUS D26 H
V$1250	25		-FBA FA BUS D25 H
V$1170	24		-FBA FA BUS D24 H
V$1198	23	6	-FBA FA BUS D23 H
V$1260	22		-FBA FA BUS D22 H
V$1139	21		-FBA FA BUS D21 H
V$1165	20		-FBA FA BUS D20 H
V$1261	19	5	-FBA FA BUS D19 H
V$1248	18		-FBA FA BUS D18 H
V$1137	17		-FBA FA BUS D17 H
V$1169	16		-FBA FA BUS D16 H
V$1199	15	4	-FBA FA BUS D15 H
V$1262	14		-FBA FA BUS D14 H
V$1134	13		-FBA FA BUS D13 H
V$1183	12		-FBA FA BUS D12 H
V$1263	11	3	-FBA FA BUS D11 H
V$1244	10		-FBA FA BUS D10 H
V$1249	09		-FBA FA BUS D09 H
V$1164	08		-FBA FA BUS D08 H
V$1200	07	2	-FBA FA BUS D07 H
V$1265	06		-FBA FA BUS D06 H
V$1133	05		-FBA FA BUS D05 H
V$1196	04		-FBA FA BUS D04 H
V$1264	03	1	-FBA FA BUS D03 H
V$1245	02		-FBA FA BUS D02 H
V$1251	01		-FBA FA BUS D01 H
V$1168	00		-FBA FA BUS D00 H

E.13 FAUPC

Name:	V$ symbol:	Bit:	Nibble:	Signal name:
FAUPC	V$0243	08	3	FA11 UPCA A8 H
	V$0321	07	2	FA11 UPCA A7 H
	V$0320	06		FA11 UPCA A6 H
	V$0319	05		FA11 UPCA A5 H
	V$0325	04		FA11 UPCA A4 H
	V$0324	03	1	FA11 UPCA A3 H
	V$0322	02		FA11 UPCA A2 H
	V$0242	01		FA11 UPCA A1 H
	V$0335	00		FA11 UPCA A0 H

E.14 FMUPC

Name:	V$ symbol:	Bit:	Nibble:	Signal name:
FMUPC	V$1152	08	3	FM11 UPC A8 H
	V$1149	07	2	FM11 UPC A7 H
	V$1179	06		FM11 UPC A6 H
	V$1178	05		FM11 UPC A5 H
	V$1155	04		FM11 UPC A4 H
	V$1154	03	1	FM11 UPC A3 H
	V$1148	02		FM11 UPC A2 H
	V$1153	01		FM11 UPC A1 H
	V$1150	00		FM11 UPC A0 H

E.15 IBDBUF

Name:	V$ symbol:	Bit:	Nibble:	Signal name:
IBDBUF	V$4289	42	11	IBD DRAM CTX 2 H
	V$4288	41		IBD DRAM CTX 1 H
	V$4287	40		IBD DRAM CTX 0 H
	V$5117	39	10	IBD DRAM TYPE 1 H
	V$5211	38		IBD DRAM TYPE 0 H
	V$5116	37		IBD DRAM REF 1 H
	V$5214	36		IBD DRAM REF 0 H
	V$5192	35	9	IBD BDEST NEXT H
	V$5163	34		IBD DRAM LAST H
	V$5171	33		IBD DRAM SUSP H
	V$5236	32		IBD DISABLE SB HIT H
	V$5245	31	8	IBD BUF FD INST H
	V$5249	30		IBD EXT OPC H
	V$6129	29		IBD BUF IBUF PE H
	V$6128	28		IBD BUF DRAM PE H
	V$5270	27	7	IBD IFORK VALID H
	V$5175	26		IBD DMUX VALID H
	V$5172	25		IBD EXC ONLY H
	V$5213	24		IBD EPC 0 OR 7 H
	V$5111	23	6	IBD ISEL B2 H
	V$3193	22		ICA LD IFORK DISP H
	V$5101	21		IBD BUF LD CTL 1 H
	V$5103	20		IBD BUF LD CTL 0 H
	V$4286	19	5	-IBD OPTIMIZED H
	V$4192	18		IBD BUF DELTA PC 2 H
	V$4187	17		IBD BUF DELTA PC 1 H

V$ symbol	Bit	Nibble	Signal name
V$4193	16		IBD BUF DELTA PC 0 H
V$6132	15	4	IBD IBGPR 3 H
V$6179	14		IBD IBGPR 2 H
V$6135	13		IBD IBGPR 1 H
V$6133	12		IBD IBGPR 0 H
V$4285	11	3	IBD IGPR 3 H
V$4284	10		IBD IGPR 2 H
V$4283	09		IBD IGPR 1 H
V$4282	08		IBD IGPR 0 H
V$5161	07	2	IBD BUF FA 7 H
V$5136	06		IBD BUF FA 6 H
V$5253	05		IBD BUF FA 5 H
V$5158	04		IBD BUF FA 4 H
V$5179	03	1	IBD BUF FA 3 H
V$6123	02		IBD BUF FA 2 H
V$6119	01		IBD BUF FA 1 H
V$6121	00		IBD BUF FA 0 H

E.16   IBUF

Name:	V$ symbol:	Bit:	Nibble:	Signal name:
IBUF	V$5114	15	4	IBD IBUF DATA B1 7 H
	V$5208	14		IBD IBUF DATA B1 6 H
	V$5209	13		IBD IBUF DATA B1 5 H
	V$5210	12		IBD IBUF DATA B1 4 H
	V$5237	11	3	IBD IBUF DATA B1 3 H
	V$5233	10		IBD IBUF DATA B1 2 H
	V$5246	09		IBD IBUF DATA B1 1 H
	V$5243	08		IBD IBUF DATA B1 0 H
	V$5104	07	2	IBD IBUF DATA B0 7 H
	V$5106	06		IBD IBUF DATA B0 6 H
	V$5108	05		IBD IBUF DATA B0 5 H
	V$5105	04		IBD IBUF DATA B0 4 H
	V$5250	03	1	IBD IBUF DATA B0 3 H
	V$5248	02		IBD IBUF DATA B0 2 H
	V$5247	01		IBD IBUF DATA B0 1 H
	V$5252	00		IBD IBUF DATA B0 0 H

E.17   IBXERR

Name:	V$ symbol:	Bit:	Nibble:	Signal name:
IBXERR	V$C197	04	2	-EBE IBOX ERR LTH E H
	V$8163	03	1	EBE IBOX ERR LTH D H
	V$4166	02		EBE IBOX ERR LTH C H
	V$6204	01		EBE IBOX ERR LTH B H
	V$5186	00		EBE IBOX ERR LTH A H

E.18   IDIAG

Name:	V$ symbol:	Bit:	Nibble:	Signal name:
IDIAG	V$D102	05	2	ICA IBOX DIAG DONE H
	V$5244	04		EBC E DISP I DLY H
	V$D150	03	1	EBC8 MCF-E DISP I H
	V$5196	02		-ICB ID FULL STALL H
	V$6116	01		ICA PC ISTALL H
	V$4130	00		IDP5 ISTALL A H

E.19   INCR

Name:	V$ symbol:	Bit:	Nibble:	Signal name:
INCR	V$4175	07	2	-IBD INCR VIBA BY 4 H
	V$3131	06		IBD9 SHIFT COUNT 2 B H
	V$3150	05		IBDD LAT CTX 1 H
	V$3253	04		IBD9 SHIFT COUNT 0 B H
	V$6171	03	1	IBD UNLAT CUR VAL 3 H
	V$6170	02		IBD UNLAT CUR VAL 2 H
	V$6174	01		IBD UNLAT CUR VAL 1 H
	V$6206	00		IBD UNLAT CUR VAL 0 H

E.20   IOPSEL

Name:	V$ symbol:	Bit:	Nibble:	Signal name:
IOPSEL	V$3163	05	2	CSB UOPSEL 1 B H
	V$3206	04		CSB UOPSEL 0 B H
	V$8146	03	1	CSA UMISC 3 H
	V$8168	02		CSA UMISC 2 H
	V$8143	01		CSA UMISC 1 H
	V$8150	00		CSA UMISC 0 H

E.21   IUPC

Name:	V$ symbol:	Bit:	Nibble:	Signal name:
IUPC	V$5120	07	2	ICA8 LD OR SAV 7 H
	V$5119	06		ICA8 LD OR SAV 6 H
	V$5157	05		ICA8 LD OR SAV 5 H
	V$5121	04		ICA8 LD OR SAV 4 H
	V$5155	03	1	ICA8 LD OR SAV 3 H
	V$5154	02		ICA8 LD OR SAV 2 H
	V$5150	01		ICA8 LD OR SAV 1 H
	V$5156	00		ICA8 LD OR SAV 0 H

E.22   IVABUS

Name:	V$ symbol:	Bit:	Nibble:	Signal name:
IVABUS	V$B160	31	8	IVA BUS 31 H
	V$B223	30		IVA BUS 30 H
	V$B202	29		IVA BUS 29 H
	V$B185	28		IVA BUS 28 H
	V$B117	27	7	IVA BUS 27 H
	V$B186	26		IVA BUS 26 H
	V$B184	25		IVA BUS 25 H
	V$B115	24		IVA BUS 24 H
	V$B114	23	6	IVA BUS 23 H
	V$B129	22		IVA BUS 22 H
	V$B116	21		IVA BUS 21 H
	V$B125	20		IVA BUS 20 H
	V$B121	19	5	IVA BUS 19 H
	V$B120	18		IVA BUS 18 H
	V$B124	17		IVA BUS 17 H
	V$B217	16		IVA BUS 16 H
	V$B214	15	4	IVA BUS 15 H
	V$B147	14		IVA BUS 14 H
	V$B149	13		IVA BUS 13 H
	V$B148	12		IVA BUS 12 H
	V$B218	11	3	IVA BUS 11 H
	V$B219	10		IVA BUS 10 H
	V$B220	09		IVA BUS 09 H
	V$B211	08		IVA BUS 08 H
	V$B172	07	2	IVA BUS 07 H
	V$B210	06		IVA BUS 06 H
	V$B180	05		IVA BUS 05 H
	V$B166	04		IVA BUS 04 H

	V$B178	03	1	IVA BUS 03 H
	V$B181	02		IVA BUS 02 H
	V$B161	01		IVA BUS 01 H
	V$B158	00		IVA BUS 00 H

E.23 MDBUSI

Name:	V$ symbol:	Bit:	Nibble:	Signal name:
MDBUSI	V$3281	31	8	MD BUS D31 H
	V$3232	30		MD BUS D30 H
	V$3244	29		MD BUS D29 H
	V$3251	28		MD BUS D28 H
	V$3252	27	7	MD BUS D27 H
	V$3282	26		MD BUS D26 H
	V$3116	25		MD BUS D25 H
	V$3165	24		MD BUS D24 H
	V$3280	23	6	MD BUS D23 H
	V$3234	22		MD BUS D22 H
	V$3229	21		MD BUS D21 H
	V$3235	20		MD BUS D20 H
	V$3120	19	5	MD BUS D19 H
	V$3279	18		MD BUS D18 H
	V$3118	17		MD BUS D17 H
	V$3283	16		MD BUS D16 H
	V$3277	15	4	MD BUS D15 H
	V$3230	14		MD BUS D14 H
	V$3248	13		MD BUS D13 H
	V$3233	12		MD BUS D12 H
	V$3121	11	3	MD BUS D11 H
	V$3256	10		MD BUS D10 H
	V$3119	09		MD BUS D09 H
	V$3168	08		MD BUS D08 H
	V$3276	07	2	MD BUS D07 H
	V$3228	06		MD BUS D06 H
	V$3249	05		MD BUS D05 H
	V$3250	04		MD BUS D04 H
	V$3254	03	1	MD BUS D03 H
	V$3257	02		MD BUS D02 H
	V$3135	01		MD BUS D01 H
	V$3164	00		MD BUS D00 H

E.24 MDBUSM

Name:	V$ symbol:	Bit:	Nibble:	Signal name:
MDBUSM	V$2130	31	8	MD BUS D31 H
	V$2131	30		MD BUS D30 H
	V$2129	29		MD BUS D29 H
	V$2132	28		MD BUS D28 H
	V$2140	27	7	MD BUS D27 H
	V$2139	26		MD BUS D26 H
	V$2135	25		MD BUS D25 H
	V$2136	24		MD BUS D24 H
	V$2144	23	6	MD BUS D23 H
	V$2134	22		MD BUS D22 H
	V$2155	21		MD BUS D21 H
	V$2161	20		MD BUS D20 H
	V$2154	19	5	MD BUS D19 H
	V$2153	18		MD BUS D18 H
	V$2162	17		MD BUS D17 H
	V$2160	16		MD BUS D16 H
	V$2166	15	4	MD BUS D15 H
	V$2156	14		MD BUS D14 H
	V$2167	13		MD BUS D13 H
	V$2157	12		MD BUS D12 H
	V$2158	11	3	MD BUS D11 H
	V$2101	10		MD BUS D10 H
	V$2195	09		MD BUS D09 H
	V$2100	08		MD BUS D08 H
	V$2103	07	2	MD BUS D07 H
	V$2102	06		MD BUS D06 H
	V$2106	05		MD BUS D05 H
	V$2198	04		MD BUS D04 H
	V$2107	03	1	MD BUS D03 H
	V$2199	02		MD BUS D02 H
	V$2200	01		MD BUS D01 H
	V$2105	00		MD BUS D00 H

E.25 MEMREQ

Name:	V$ symbol:	Bit:	Nibble:	Signal name:
MEMREQ	V$C171	25	7	MCC EBOX PA ACK A H
	V$C172	24		MCC OP PA ACK A H
	V$5167	23	6	MCC IBF PA ACK H

V$ symbol	Bit	Nibble	Signal name
V$C215	22		EBC EBD MAST RST DLY H
V$C173	21		MCC DEST CODE 1 H
V$C174	20		-EBD9 MEM REQ LST CYC H
V$C193	19	5	MCC PORT STAT CODE 3 H
V$C191	18		MCC PORT STAT CODE 2 H
V$C190	17		MCC PORT STAT CODE 1 H
V$C192	16		MCC PORT STAT CODE 0 H
V$A180	15	4	ICA ISTALL A H
V$A167	14		ICA IBF REQUEST H
V$C231	13		MCC STAT CODE OUT 1 H
V$C226	12		-CSB EBOX FORK A H
V$A168	11	3	ICB OP MCF 3 H
V$A179	10		ICB OP MCF 2 H
V$A190	09		ICB OP MCF 1 H
V$A186	08		ICB OP MCF 0 H
V$A221	07	2	ICA OP ABORT A H
V$A184	06		EBD ESTALL TO MCC H
V$A153	05		EBC EBOX MCF 5 H
V$A182	04		EBC EBOX MCF 4 H
V$A174	03	1	EBC EBOX MCF 3 H
V$A165	02		EBC EBOX MCF 2 H
V$A164	01		EBC EBOX MCF 1 H
V$A154	00		EBC EBOX MCF 0 H

E.26   MUPC

Name:	V$ symbol:	Bit:	Nibble:	Signal name:
MUPC	V$A192	07	2	MCC1 UADR B 7 H
	V$A231	06		MCC1 UADR B 6 H
	V$A273	05		MCC1 UADR B 5 H
	V$A274	04		MCC1 UADR B 4 H
	V$A109	03	1	MCC1 UADR A 3 H
	V$A110	02		MCC1 UADR A 2 H
	V$A288	01		MCC1 UADR A 1 H
	V$A279	00		MCC1 UADR A 0 H

E.27   NATRAM

Name:	V$ symbol:	Bit:	Nibble:	Signal name:
NATRAM	V$3213	19	5	IBDC NAT CTX 2 H
	V$3217	18		IBDC NAT CTX 1 H
	V$3209	17		IBDC NAT CTX 0 H

	V$ symbol	Bit	Nibble	Signal name
	V$3112	16		IBDC NAT TYPE 1 H
	V$3183	15	4	IBDC NAT TYPE 0 H
	V$3167	14		IBDC NAT REF 1 H
	V$3182	13		IBDC NAT REF 0 H
	V$3210	12		IBDC NAT CTL 1 H
	V$3177	11	3	IBDC NAT CTL 0 H
	V$3208	10		IBDC NAT SUSPEND H
	V$3109	09		IBDC NAT BDEST NXT H
	V$3211	08		IBDC NAT LAST H
	V$3184	07	2	IBDB NAT OPAR H
	V$3101	06		IBDB NAT FPA H
	V$3180	05		IBDB NAT ADRS 05 H
	V$3103	04		IBDB NAT ADRS 04 H
	V$3278	03	1	IBDB NAT ADRS 03 H
	V$3102	02		IBDB NAT ADRS 02 H
	V$3108	01		IBDB NAT ADRS 01 H
	V$3100	00		IBDB NAT ADRS 00 H

E.28   OPAR

Name:	V$ symbol:	Bit:	Nibble:	Signal name:
OPAR	V$4278	37	10	ICA FORCE AMUX OPAR H
	V$4102	36		ICA FORCE BMUX OPAR H
	V$C144	35	9	EDP OPR PAR ERR H
	V$3184	34		IBDB NAT OPAR H
	V$3221	33		IBDD LAT OPAR H
	V$5170	32		ICA5 ICS OPAR H
	V$XXXX	31	8	Zero-Filler
	V$4206	30		-ICA EN OP OPAR VAL H
	V$8184	29		-IDP OP OPAR VALID H
	V$8191	28		IDP OP LWD OPAR H
	V$8154	27	7	EBE WBUS OPAR B3 A H
	V$8155	26		EBE WBUS OPAR B2 A H
	V$8153	25		EBE WBUS OPAR B1 A H
	V$8156	24		EBE WBUS OPAR B0 A H
	V$0186	23	6	EBE WBUS OPAR B3 C H
	V$0131	22		EBE WBUS OPAR B2 C H
	V$0129	21		EBE WBUS OPAR B1 C H
	V$0180	20		EBE WBUS OPAR B0 C H
	V$4205	19	5	EBE WBUS OPAR B3 B H
	V$4123	18		EBE WBUS OPAR B2 B H
	V$4210	17		EBE WBUS OPAR B1 B H

V$ symbol	Bit	Nibble	Signal name
V$4209	16		EBE WBUS OPAR B0 B H
V$2126	15	4	-EBE WBUS OPAR B3 H
V$2127	14		-EBE WBUS OPAR B2 H
V$2125	13		-EBE WBUS OPAR B1 H
V$2113	12		-EBE WBUS OPAR B0 H
V$3153	11	3	MCD MD BUS OPAR B3 H
V$3149	10		MCD MD BUS OPAR B2 H
V$3152	09		MCD MD BUS OPAR B1 H
V$3148	08		MCD MD BUS OPAR B0 H
V$XXXX	07	2	formally MCD MD BUS OPAR B3 H
V$XXXX	06		formally MCD MD BUS OPAR B2 H
V$XXXX	05		formally MCD MD BUS OPAR B1 H
V$XXXX	04		formally MCD MD BUS OPAR B0 H
V$4101	03	1	IBD DBUS OPAR B3 H
V$4104	02		IBD DBUS OPAR B2 H
V$4140	01		IBD DBUS OPAR B1 H
V$4100	00		IBD DBUS OPAR B0 H

E.29   OPBUS

Name:	V$ symbol:	Bit:	Nibble:	Signal name:
OPBUS	V$0178	31	8	OP BUS D31 H
	V$0174	30		OP BUS D30 H
	V$0148	29		OP BUS D29 H
	V$0106	28		OP BUS D28 H
	V$0203	27	7	OP BUS D27 H
	V$0260	26		OP BUS D26 H
	V$0153	25		OP BUS D25 H
	V$0232	24		OP BUS D24 H
	V$0202	23	6	OP BUS D23 H
	V$0261	22		OP BUS D22 H
	V$0149	21		OP BUS D21 H
	V$0228	20		OP BUS D20 H
	V$0285	19	5	OP BUS D19 H
	V$0172	18		OP BUS D18 H
	V$0154	17		OP BUS D17 H
	V$0229	16		OP BUS D16 H
	V$0295	15	4	OP BUS D15 H
	V$0198	14		OP BUS D14 H
	V$0344	13		OP BUS D13 H
	V$0289	12		OP BUS D12 H
	V$0301	11	3	OP BUS D11 H

V$ symbol	Bit	Nibble	Signal name
V$0300	10		OP BUS D10 H
V$0293	09		OP BUS D09 H
V$0271	08		OP BUS D08 H
V$0286	07	2	OP BUS D07 H
V$0274	06		OP BUS D06 H
V$0103	05		OP BUS D05 H
V$0275	04		OP BUS D04 H
V$0179	03	1	OP BUS D03 H
V$0264	02		OP BUS D02 H
V$0152	01		OP BUS D01 H
V$0233	00		OP BUS D00 H

E.30   OPCODE

Name:	V$ symbol:	Bit:	Nibble:	Signal name:
OPCODE	V$1101	07	2	ICA OPC BIT 7 A H
	V$1103	06		ICA OPC BIT 6 A H
	V$1238	05		ICA OPC BIT 5 A H
	V$1125	04		ICA OPC BIT 4 A H
	V$1240	03	1	ICA OPC BIT 3 A H
	V$1237	02		ICA OPC BIT 2 A H
	V$1141	01		ICA OPC BIT 1 A H
	V$1241	00		ICA OPC BIT 0 A H

E.31   OPMCF

Name:	V$ symbol:	Bit:	Nibble:	Signal name:
OPMCF	V$A168	03	1	ICB OP MCF 3 H
	V$A179	02		ICB OP MCF 2 H
	V$A190	01		ICB OP MCF 1 H
	V$A186	00		ICB OP MCF 0 H

E.32   OPPORT

Name:	V$ symbol:	Bit:	Nibble:	Signal name:
OPPORT	V$6186	63	16	ICA OPV CTL 0 H
	V$6190	62		ICA OPV CTL 1 H
	V$5149	61		-ICAG SET OPV FB H
	V$5141	60	15	-ICB SET OPV H
	V$6158	59		-ICB9 IMD OPV LAT H
	V$6154	58		-ICB9 PEND IMD OPV H

V$XXXX	57		Zero-Filler
V$XXXX	56	14	Zero-Filler
V$C172	55		MCC OP PA ACK A H
V$6127	54		MCC OP PA ACK B H
V$XXXX	53		Zero-Filler
V$XXXX	52	13	Zero-Filler
V$0191	51		EBD OPBUS VALID H
V$1206	50		-FBA OPBU VAL TO FBM H
V$XXXX	49		Zero-Filler
V$C212	48	12	-ICB ALWAYS OP VALID H
V$C211	47		-ICB ALMOST OP VALID H
V$6155	46		-ICB8 ALMOST SET OPV H
V$XXXX	45		Zero-Filler
V$A221	44	11	ICA OP ABORT A H
V$C183	43		ICA OP ABORT B LAT H
V$1120	42		ICA IBF OR OP FLUSH H
V$A264	41		ICA MBOX FLUSH A H
V$C188	40	10	-ICA OP FLUSH B H
V$4206	39		-ICA EN OP OPAR VAL H
V$8184	38		-IDP OP OPAR VALID H
V$XXXX	37		Zero-Filler
V$XXXX	36	9	Zero-Filler
V$C201	35		ICB KILL OP WRT H
V$E102	34		MCC NEXT OP WRT H
V$E186	32		MCC TRAP OP WCHK H
V$E135	31	8	MCC TRAP OP WRT H
V$C176	30		-EBD9 EN OP WRT STALL H
V$5254	29		-ICAG OP REQ CIP LAT H
V$5148	28		ICB SET OP WRT CIP H
V$6109	27	7	-ICBA OP WRT CIP H
V$C213	26		-EBD9 OP WRT IN PROG H
V$C148	25		-EBD9 OP WRT LST CYC H
V$6134	24		-ICA OP WRT LAT H
V$C170	23	6	-ICA EN OP WRT ACK H
V$6165	22		-ICA ENA OP MD RESP H
V$XXXX	21		Zero-Filler
V$4147	20		-ICA IVA SEL OP VA H
V$4176	19	5	-ICA ENA OP VA LD H
V$5131	18		IDP OP VA BIT 00 H
V$5241	17		IDP OP VA BIT 01 H
V$A186	16		ICB OP MCF 0 H
V$A190	15	4	ICB OP MCF 1 H
V$A179	14		ICB OP MCF 2 H

V$A168	13		ICB OP MCF 3 H
V$XXXX	12		Zero-Filler
V$A185	11	3	ICB OP MEM CTX 0 H
V$A170	10		ICB OP MEM CTX 1 H
V$A183	09		ICB OP MEM CTX 2 H
V$3143	08		ICB OP MEM REQ H
V$4148	07	2	ICB OP MEM REQ A H
V$XXXX	06		Zero-Filler
V$6182	05		-ICA SET FA OPMEM REQ H
V$C117	04		ICB FA OP MEM REQ H
V$XXXX	03	1	Zero-Filler
V$4172	02		IDP5 IVA SEL OP VA A H
V$4221	01		IDP5 IVA SEL OP VA B H
V$4108	00		IDP5 IVA SEL OP VA C H

E.33 PAACK

Name:	V$ symbol:	Bit:	Nibble:	Signal name:
PAACK	V$D167	06	2	-MCC EBOX PA ACK B H
	V$C171	05		MCC EBOX PA ACK A H
	V$6142	04		EBD EB PA ACK LTH H
	V$XXXX	03	1	Zero-Filler
	V$5167	02		MCC IBF PA ACK H
	V$C172	01		MCC OP PA ACK A H
	V$6127	00		MCC OP PA ACK B H

E.34 PAMD

Name:	V$ symbol:	Bit:	Nibble:	Signal name:
PAMD	V$D167	23	6	-MCC EBOX PA ACK B H
	V$A184	22		EBD ESTALL TO MCC H
	V$C215	21		EBC EBD MAST RST DLY H
	V$A167	20		ICA IBF REQUEST H
	V$A264	19	5	ICA MBOX FLUSH A H
	V$A180	18		ICA ISTALL A H
	V$A221	17		ICA OP ABORT A H
	V$A277	16		ICA MBOX FLUSH B H
	V$A168	15	4	ICB OP MCF 3 H
	V$A179	14		ICB OP MCF 2 H
	V$A190	13		ICB OP MCF 1 H
	V$A186	12		ICB OP MCF 0 H

V$C193	11	3	MCC PORT STAT CODE 3 H
V$C191	10		MCC PORT STAT CODE 2 H
V$C190	09		MCC PORT STAT CODE 1 H
V$C192	08		MCC PORT STAT CODE 0 H
V$C172	07	2	MCC OP PA ACK A H
V$5167	06		MCC IBF PA ACK H
V$A153	05		EBC EBOX MCF 5 H
V$A182	04		EBC EBOX MCF 4 H
V$A174	03	1	EBC EBOX MCF 3 H
V$A165	02		EBC EBOX MCF 2 H
V$A164	01		EBC EBOX MCF 1 H
V$A154	00		EBC EBOX MCF 0 H

E.35 PAMM

Name:	V$ symbol:	Bit:	Nibble:	Signal name:
PAMM	V$A212	04	2	MAP9 PAMM CONF A H
	V$A173	03	1	MAP9 PAMM CONF 8 H
	V$A172	02		MAP9 PAMM CONF 4 H
	V$A224	01		MAP9 PAMM CONF 2 H
	V$A216	00		MAP9 PAMM CONF 1 H

E.36 PARITY

Name:	V$ symbol:	Bit:	Nibble:	Signal name:
PARITY	V$9170	65	17	EBD ECS PE FLAG H
	V$D158	64		EBD EDP PE FLAG A H
	V$9169	63	16	EBD EDP PE FLAG H
	V$9173	62		EBD EMCR PE FLAG H
	V$9174	61		EBD USTK PE FLAG H
	V$9143	60		EBD WBUS PE FLAG H
	V$XXXX	59	15	Zero-Filler
	V$XXXX	58		Zero-Filler
	V$XXXX	57		Zero-Filler
	V$9110	56		EBD ECS PE LST CYC H
	V$C219	55	14	-CSB ECS PAR ERR H
	V$E164	54		CSA PAR ERR H
	V$F104	53		CSAS CSA PAR H
	V$F124	52		-CSB CS PAR OK A H
	V$C142	51	13	CSB USTK PAR ERR H
	V$E140	50		-CSBR FLIP USTK PAR H

V$E151	49		CSBS DATA PAR H
V$C144	48		EDP OPR PAR ERR H
V$C160	47	12	-EDP RESULT PAR ERR H
V$8134	46		DISA BYTE 10 PAR H
V$8165	45		EDPI DISA BYTE 32 PAR H
V$8126	44		EDPI FLIP WREG PAR H
V$9150	43	11	EBC FLIP WBUS PAR B0 H
V$9149	42		EBC FLIP WBUS PAR B1 H
V$9145	41		EBC FLIP WBUS PAR B2 H
V$9144	40		EBC FLIP WBUS PAR B3 H
V$XXXX	39	10	Zero-Filler
V$C159	38		-EBC MCF RAM PAR ERR H
V$D142	37		EBCA MCF PAR H
V$D133	36		EBCH FLIP MCF RAM PAR H
V$XXXX	35	9	Zero-Filler
V$0240	34		FA17 GPR PPAR 00 H
V$0241	33		FA17 GPR PPAR 01 H
V$0122	32		FA17 GPR PPAR 02 H
V$0121	31	8	FA19 UWD PARITY H
V$0169	30		FBM CS PAR ERROR H
V$0165	29		FBM FDRAM PAR ERROR H
V$1275	28		FM16 UWD PARITY H
V$XXXX	27	7	Zero-Filler
V$6128	26		IBD BUF DRAM PE H
V$6129	25		IBD BUF IBUF PE H
V$4201	24		ICA FORCE GPR PE H
V$6115	23	6	ICA FORCE RLOG PE H
V$9179	22		ICA ICS PE H
V$9181	21		ICB IBUF PE H
V$9101	20		ICB IDRAM PE H
V$9100	19	5	ICB RLOG PE H
V$9176	18		IDP IAMUX PE H
V$9182	17		IDP IBMUX PE H
V$XXXX	16		Zero-Filler
V$XXXX	15	4	Zero-Filler
V$XXXX	14		Zero-Filler
V$2165	13		MAP2 <29:4> PAR H
V$XXXX	12		formally MCC1 MMS PAR ERR H
V$XXXX	11	3	formally MCC2 ACCESS PAR H
V$A118	10		MCCC U CPR PAR A H
V$A126	09		MCCC U CPR PAR B H
V$2109	08		MCCM INV CACH BYT PAR H
V$A201	07	2	MCD3 ABUS DAT PERR H

```
                V$A251          06                  -MCDU CACH DAT PERR H
                V$A250          05                  -MCDU WR DAT PERR H
                V$A200          04                  MAP2 ABUS ADR PERR H

                V$A204          03          1       MAPL TAG PERR H
                V$A202          02                  MAPL TAG W PERR H
                V$A148          01                  -MAPR TB PERR H
                V$9162          00                  MCC MBOX CS PE H
```

E.37   PSL

Name:	V$ symbol:	Bit:	Nibble:	Signal name:
PSL	V$E117	20	6	EBE PSL CM TO CSB H
	V$C113	19	5	-EBE PSL TP H
	V$E112	18		EBD UTRAP VECTOR 4 H
	V$E152	17		EBE PSL IS TO CSB H
	V$A144	16		EBE CURMOD 1 TO MCC H
	V$A162	15	4	EBE CURMOD 0 TO MCC H
	V$XXXX	14		Zero-Filler
	V$XXXX	13		Zero-Filler
	V$D125	12		EBE PSL IPL 4 H
	V$D127	11	3	EBE PSL IPL 3 H
	V$D126	10		EBE PSL IPL 2 H
	V$D131	09		EBE PSL IPL 1 H
	V$D130	08		EBE PSL IPL 0 H
	V$XXXX	07	2	Zero-Filler
	V$XXXX	06		Zero-Filler
	V$C109	05		-EBE PSL IV TO EBD H
	V$XXXX	04		Zero-Filler
	V$9171	03	1	EDP PSL N BIT A H
	V$9172	02		EDP PSL Z BIT A H
	V$9168	01		EDP PSL V BIT A H
	V$9167	00		EDP PSL C BIT A H

E.38   REGBUS

Name:	V$ symbol:	Bit:	Nibble:	Signal name:
REGBUS	V$2205	07	2	REG BUS 7 H
	V$2203	06		REG BUS 6 H
	V$2227	05		REG BUS 5 H
	V$2226	04		REG BUS 4 H
	V$2224	03	1	REG BUS 3 H

	V$2225	02		REG BUS 2 H
	V$2204	01		REG BUS 1 H
	V$2223	00		REG BUS 0 H

E.39 STALL

Name:	V$ symbol:	Bit:	Nibble:	Signal name:
STALL	V$6202	28	8	ICA IFORK NOP H
	V$3189	27	7	ICA IFORK CYCLE H
	V$6185	26		ICA CTX UNALIGNED H
	V$5133	25		ICA6 UTRAP CTL 1 H
	V$5132	24		ICA6 UTRAP CTL 0 H
	V$A180	23	6	ICA ISTALL A H
	V$9107	22		ICA ISTALL B H
	V$C205	21		ICA ISTALL BUF A H
	V$6137	20		ICA ISTALL C H
	V$3197	19	5	ICA ISTALL D H
	V$4156	18		ICA ISTALL E H
	V$5196	17		-ICB ID FULL STALL H
	V$3190	16		ICA PC ISTALL A H
	V$9103	15	4	EBD RSV MODE H
	V$9100	14		ICB RLOG PE H
	V$5123	13		ICA7 ICS PAR ERR H
	V$9181	12		ICB IBUF PE H
	V$9101	11	3	ICB IDRAM PE H
	V$9176	10		IDP IAMUX PE H
	V$9182	09		IDP IBMUX PE H
	V$XXXX	08		Zero-Filler
	V$C172	07	2	MCC OP PA ACK A H
	V$6127	06		MCC OP PA ACK B H
	V$5167	05		MCC IBF PA ACK H
	V$9162	04		MCC MBOX CS PE H
	V$5221	03	1	EBD ESTALL TO ICA H
	V$5186	02		EBE IBOX ERR LTH A H
	V$9170	01		EBD ECS PE FLAG H
	V$9110	00		EBD ECS PE LST CYC H

E.40   UPCSAV

Name:	V$ symbol:	Bit:	Nibble:	Signal name:
UPCSAV	V$E173	12	4	CSBQ UPCSAVE 12 H
	V$E172	11	3	CSBQ UPCSAVE 11 H
	V$E143	10		CSBQ UPCSAVE 10 H
	V$E157	09		CSBQ UPCSAVE 09 H
	V$E156	08		CSBQ UPCSAVE 08 H
	V$E150	07	2	CSBQ UPCSAVE 07 H
	V$E171	06		CSBQ UPCSAVE 06 H
	V$E174	05		CSBQ UPCSAVE 05 H
	V$E170	04		CSBQ UPCSAVE 04 H
	V$E149	03	1	CSBQ UPCSAVE 03 H
	V$E141	02		CSBP UPCSAVE 02 H
	V$E142	01		CSBP UPCSAVE 01 H
	V$E148	00		CSBP UPCSAVE 00 H

E.41   WBUS

Name:	V$ symbol:	Bit:	Nibble:	Signal name:
WBUS	V$4232	31	8	WBUS D31 H
	V$4231	30		WBUS D30 H
	V$4227	29		WBUS D29 H
	V$4223	28		WBUS D28 H
	V$4107	27	7	WBUS D27 H
	V$4257	26		WBUS D26 H
	V$4271	25		WBUS D25 H
	V$4256	24		WBUS D24 H
	V$4251	23	6	WBUS D23 H
	V$4252	22		WBUS D22 H
	V$4266	21		WBUS D21 H
	V$4253	20		WBUS D20 H
	V$4157	19	5	WBUS D19 H
	V$4131	18		WBUS D18 H
	V$4127	17		WBUS D17 H
	V$4128	16		WBUS D16 H
	V$4247	15	4	WBUS D15 H
	V$4243	14		WBUS D14 H
	V$4244	13		WBUS D13 H
	V$4245	12		WBUS D12 H
	V$4122	11	3	WBUS D11 H
	V$4274	10		WBUS D10 H

V$4276	09		WBUS D09 H
V$4202	08		WBUS D08 H
V$4184	07	2	WBUS D07 H
V$4182	06		WBUS D06 H
V$4158	05		WBUS D05 H
V$4177	04		WBUS D04 H
V$4194	03	1	WBUS D03 H
V$4191	02		WBUS D02 H
V$4178	01		WBUS D01 H
V$4188	00		WBUS D00 H