AMT

Active Memory Technology

DAP Series

System Management under VAX/VMS

(man020.01)

Table of Contents

2.4.2.6 Customizing the user environment

The environment that is established by including the execution of **DAPSTARTUP.COM** as part of system startup, may be altered depending on the requirements of the DAP site. Such changes could include the establishment of:

- Default target machine type
- Default directory specification for text files to be included in FORTRAN-PLUS compilation
- Default text library to be included in FORTRAN-PLUS compilation
- Default object libraries to be searched for symbol resolution during program linkage

You can use the AMT manual *DAP Series: Program Development under VAX/VMS* as a guide to the facilities described in the following sections.

Default target machine type

Many of the DAP utilities such as the FORTRAN-PLUS compiler, the DAP linker and DAP librarian default to assuming that the target machine is a DAP 500. To direct the software to assume that a DAP 600 is the target machine, the **DAPSIZE** qualifier is used. Thus:

```
$ DLINK SMALL_DAP
```

links the DAP object file **SMALL_DAP.DOB** to produce the DAP executable file **LARGE_DAP.DEX** for a DAP 500. However, the command:

```
$ DLINK/DAPSIZE=64 LARGE_DAP
```

links the DAP object file **LARGE_DAP.DOB** to produce the DAP executable file **LARGE_DAP.DEX** for a DAP 600.

The logical name **DAP_SIZE** can be used to establish the target DAP (for information on this see the manual *DAP Series: Program Development under VAX/VMS*). If the target machine is a DAP 600, it is sensible to establish a system-wide logical name. A system-wide logical name is defined for any process on the system and therefore removes the necessity for the individual user to establish a private (process-wide) logical name or explicitly specify the **DAPSIZE** qualifier.

To establish **DAP_SIZE** as a system wide logical name, such that the target DAP is assumed to be a DAP 600, include the following command in the startup command file:

```
$ DEFINE/SYSTEM DAP_SIZE 64
```

The command:

```
$ DLINK LARGE_DAP
```

will now link the DAP object file **LARGE_DAP.DOB** to produce the DAP executable file **LARGE_DAP.DEX** for a DAP 600.

Default text files to be included in FORTRAN-PLUS compilation

To establish a default search path for FORTRAN-PLUS source files that have been included in a FORTRAN-PLUS program, the logical name **DAPF_INCLUDE** can be defined (see chapter 2 in the manual *DAP Series: Program Development under VAX/VMS*). If you wish to establish such a search path on a system-wide basis, then this definition can be established in the system startup file, using the DCL command:

```
$ DEFINE/SYSTEM DAPF_INCLUDE USERDISK1:[DAP.INCLUDE]
```

where **USERDISK1:[DAP.INCLUDE]** is a directory set up to hold the relevant FORTRAN-PLUS source files.

Default text library to be included in FORTRAN-PLUS compilation

FORTRAN-PLUS source files may be put into a text library, and included in a FORTRAN-PLUS program. The situation is similar to that given above, except that the default is a text library rather than a directory specification. You can use the logical name **DAPF_LIBRARY** to establish the default library.

For example, suppose there are a number of FORTRAN-PLUS **PARAMETER** definitions that will be included in programs written by many users. If these files are put into a text library in **SYS$LIBRARY** called **PARAMETERS.TLB**; instead of using the **DFORTRAN** command:

```
$ DFORTRAN MYPROG+SYS$LIBRARY:PARAMETERS/LIBRARY
```

the logical name **DAPF_LIBRARY** can be established on a system-wide basis:

```
$ DEFINE/SYSTEM DAPF_LIBRARY SYS$LIBRARY:PARAMETERS
```

thus simplifying the **DFORTRAN** command to:

```
$ DFORTRAN MYPROG
```

Chapter 3

Powering up the DAP

3.1 Introduction

The control panel of a DAP (see figure 3.1 below) includes:

- A 20 character by 2 row vacuum fluorescent display panel
- Two indicator lights:

  POWER FAULT
- Three control buttons:

  SELECT ANSWER EXIT
- A four-position keyswitch, with positions:

  PWR OFF RESET RUN MENU

3.2 Indicator lights

The green indicator **POWER** is a 'power-on' light.

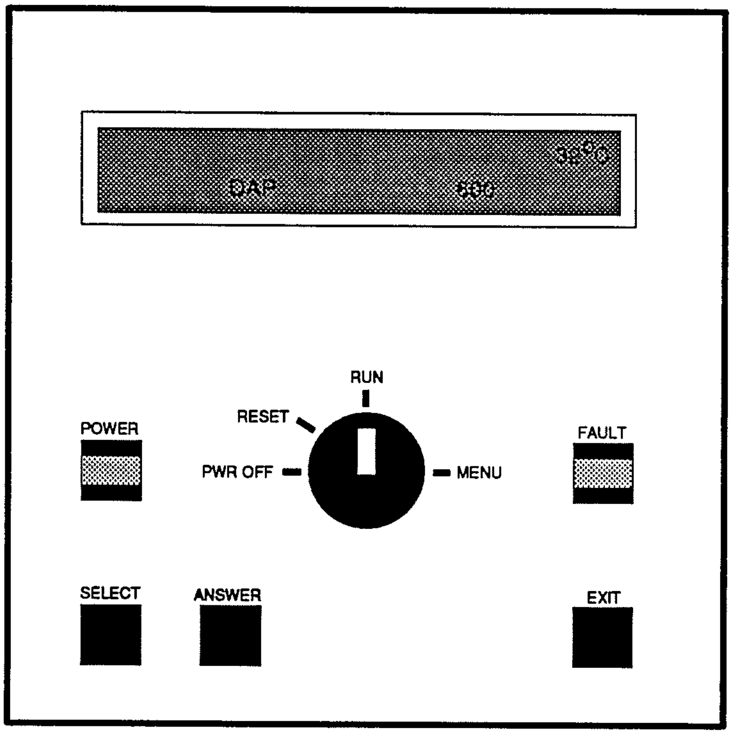


Figure 3.1
DAP control panel

STOP or RUN

Accepting one of these options changes the displayed menu once again, and you are now offered the menu:

STOP

RUN

These options control whether or not the item already selected at the menu level above – for example DAPTITLE – is displayed (RUN) or is not displayed (STOP). Note that the option highlighted when the STOP/RUN menu first comes up is the one that is currently active.

As a result of your selection one or more of the options listed below will be displayed on the control panel when the key is returned to the **RUN** position.

- The time
- A moving logo, along with the average temperature of the boards inside the DAP.

  Only this option is being RUN on the DAP 600 in figure 3.1 on page 27
- The temperatures of the individual boards inside the DAP

DAP TESTS

Accepting the **DAP TESTS** option from the top level menu allows you to run manually the self-tests which are normally run by the DAP at power-up. You are first reminded that **DAPBOOT STOP** must be executed before you begin these tests by the message:

```
CONFIRM DAPBOOT REMOVED
```

on the DAP display panel. When you confirm that **DAPMONITOR** has been removed (by pressing **ANSWER**), a sub-menu is displayed.

DAP 500 tests

On a DAP 500 the following tests are available:

- DAP SELF TESTS
- MCU TESTS PART 1
- ARRAY STORE
- CODE STORE
- PE TESTS
- MCU TESTS PART 2
- MCU TESTS PART 3
- PE M/S TESTS

DAP 600 tests

On a DAP 600 the following tests are available:

- DAP SELF TESTS
- MCU TESTS PART 1

- ARRAY STORE
- CODE STORE
- PE TESTS PART 1
- PE TESTS PART2
- MCU TESTS PART2
- MCU TESTS PART3
- PE M/S TESTS

The first test in the menu, **DAP SELF TESTS**, is effectively an amalgam of the other options, and runs each of them in turn.

When you turn the key back to **RUN** from **MENU,** the message:

```
READY FOR DAPBOOT
```

will appear on the DAP display panel for a short time. If the reason for selecting **MENU** was to change the display on the panel, and **`DAPBOOT STOP`** has not been executed, the message `READY FOR DAPBOOT` may be ignored.

Chapter 4

Using DAPBOOT

Once the DAP has been turned on and has completed its self tests (see chapter 3), various control programs and possibly device drivers must be downloaded to the DAP before it can run user programs. This bootstrap procedure is carried out by the **DAPBOOT** and **DAPMONITOR** programs as described below:

4.1 DAPBOOT and DAPMONITOR programs

The process of booting the DAP is divided into two parts, performed by two programs, **DAPBOOT** and **DAPMONITOR**. The **DAPBOOT** program is run either interactively or from the system startup file and performs the following tasks:

- It runs the **DAPMONITOR** program as a detached process
- It sends boot information to **DAPMONITOR**
- It reports the progress of the boot process on **SYS$OUTPUT**

Note that operator (OPER) privilege is required to run **DAPBOOT**.

When the DAP bootstrap process has been completed, the **DAPBOOT** program will terminate.

The **DAPMONITOR** program – which runs as a detached process **AMTDAP_ 1** – carries out three tasks:

- It boots the DAP according to the information it receives from **DAPBOOT**
- It records the accounting information requested (see Chapter 6) in the accounting files:

  SYS$MANAGER:DAPSYS_1.LOG

  SYS$MANAGER:ACCOUNTNG.DAT
- It records DAP hardware errors in the engineer's log file:

  SYS$MANAGER:AMTDAP_1.LOG

The **DAPMONITOR** program must be running before user programs can be executed on the DAP.

If **DAPMONITOR** terminates due to a DAP error it sends the following message to the operator's console via the **OPCOM** process:

```
DAPMONITOR has terminated
```

The process of booting the DAP consists of loading the following programs:

- DAP message server
- DAP MCU control program
- DAP HCU control program
- User written HCU-based device drivers

The names of the files containing the above programs are held in the DAP configuration file which is created by the **DAPCONFIG.COM** command procedure, (see Chapter 2, pages 16-17, for further details on running **DAPCONFIG**). The configuration file is read by **DAPBOOT** and the necessary information is then passed to **DAPMONITOR** which performs the load. **DAPMONITOR** reports any errors in the engineer's log file **SYS$MANAGER:AMTDAP_1.LOG** and sends an error status report to **DAPBOOT** which displays an error message on **SYS$OUTPUT**. If the boot process is successful, **DAPBOOT** terminates with the status message:

```
DAP booted successfully
```

4.2 DAPBOOT command

The **DAPBOOT** command has the form:

DAPBOOT [/*qualifiers*] [*action*]

where:

/*qualifiers* specify various options to the boot process

action specifies whether the **DAPMONITOR** program is to be started, restarted or stopped

The **DAPBOOT** parameter *action* takes one of the following values:

- **START**

  This parameter signals a request to boot the DAP, provided the **DAPMONITOR** program is not already running. If **DAPMONITOR** is running **DAPBOOT** reports an error and terminates

- **RESTART**

  This parameter signals a request to boot the DAP, after stopping the **DAPMONITOR** program if it is already running

- **STOP**

  This parameter signals a request to stop the **DAPMONITOR** program

The default value for *action* is **START**. The parameter can be abbreviated provided it remains long enough to be identified uniquely.

Note that the **RESTART** and **STOP** options both terminate **DAPMONITOR** (if it is running) and this will result in any DAP jobs currently executing being abandoned.

The **DAPBOOT** command can take several qualifiers which can be used to control the way the boot process is carried out and the subsequent operation of the **DAPMONITOR** program.

4.2.1 /ACCOUNTING qualifier

The **/ACCOUNTING** qualifier controls the accounting information written out by **DAPMONITOR** whenever a DAP process terminates. There are two types of accounting output (see chapter 6) and both are optional. The two types are:

- ASCII accounting in the DAP system log file:

  SYS$MANAGER:DAPSYS_1.LOG

- Binary accounting in the VMS accounting file:

  SYS$MANAGER:ACCOUNTNG.DAT

If you specify **/ACCOUNTING**, both accounting files are written – this is the default. You can turn off the output to one or both files by specifying **/ACCOUNTING=***type* , where *type* is one (or more) of the following:

NONE specifies no accounting information is to be output

FULL specifies both accounting files are to be written (the default)

ASCII specifies that the **DAPSYS_1.LOG** file is to be written

BINARY specifies that the **ACCOUNTNG.DAT** file is to be written

If you give more than one *type*, then you should separate the values by commas and enclose them in parentheses.

If you do use the accounting files, you should inspect or analyse them regularly, and delete them when necessary as they may become very large.

Note that you must stop **DAPMONITOR** before you can delete **DAPSYS_1.LOG**. You can use the **DCL** command:

```
$ SET ACCOUNTING/NEWFILE
```

to close the existing version of **ACCOUNTNG.DAT** and open a new one, without stopping **DAPMONITOR**.

4.2.2 /CONFIGURATION_FILE qualifier

You can use the **/CONFIGURATION_FILE** qualifier to specify the DAP configuration file which is created by **DAPCONFIG.COM** (see section 2.4.2.1 pages 16-17). The default configuration file assumed by **DAPBOOT** is **SYS$MANAGER:DAP_1.CFG** and this is the default file created by **DAPCONFIG.COM**. An alternative file can be specified by using the **/CONFIGURATION_FILE** qualifier in the form:

```
$ DAPBOOT/CONFIGURATION_FILE=file-spec
```

The file type of *file-spec* defaults to **.CFG**, thus the command:

```
$ DAPBOOT/CONFIGURATION_FILE=TEST
```

would use the file **TEST.CFG** as the configuration file.

4.2.3 /FORCELOAD qualifier

The **/FORCELOAD** qualifier forces the **DAPMONITOR** program to ignore any errors encountered during the DAP self tests which are performed before any control programs are loaded. Failure of these self tests indicates a fault on the DAP and should be reported to AMT. However, under certain circumstances, it may be possible to continue using the DAP successfully despite the failure of the confidence tests, until the fault has been corrected. The **/FORCELOAD** qualifier should only be used on the advice of AMT.

4.2.4 DAPBOOT messages

When DAPBOOT is invoked it sends a message via OPCOM to the operator's console. The form of the message is:

```
DAPBOOT started from process id id by user name
```

As the DAP boots up, each successfully loaded program is reported via **DAPBOOT** to **SYS$OUTPUT** and to the engineer's log file **SYS$MANAGER:AMTDAP_1.LOG**. If the DAP boots successfully, the information message:

```
%DAPBOOT-T-BOOTSUCC, DAP booted successfully
```

is displayed by **DAPBOOT** and the message:

```
DAP booted successfully
```

is also sent to the operator's console via the opcom process. DAPBOOT then terminates. At this point, user programs can be executed on the DAP as described in *DAP Series: Program Development under VAX/VMS* (chapter 4).

If the DAP bootstrap fails, **DAPBOOT** will display a failure messge on **SYS$OUTPUT**. Depending on the cause of the failure extra information may be sent to the engineer's log file **SYS$MANAGER:AMTDAP_1.LOG**. The information message:

```
%DAPBOOT-F-BOOTFAIL, DAPBOOT failed to perform the required operation
```

is displayed by **DAPBOOT** if the DAP bootstrap fails. In this case **DAPBOOT** sends the message:

```
DAPBOOT failed
```

to the operator's console via the OPCOM process. Other messages will give more detailed information about the cause of the error and these are listed in Appendix M of *DAP Series: Program Development Under VAX/VMS* (editions 04 and upwards of man004).

4.2.5 Files used by DAPMONITOR

The **DAPMONITOR** program creates (if necessary) and writes information to the following files:

- **SYS$MANAGER:ACCOUNTNG.DAT**
  (if binary accounting is selected)
- **SYS$SYSTEM:DAPPARAMS.DAT**
- **SYS$MANAGER:DAPSYS_1.LOG**
  (if ASCII accounting is selected)
- **SYS$MANAGER:AMTDAP_1.LOG**

Note that the accounting and log files are created if they do not exist, but otherwise information is appended to them. Also, if **DAPMONITOR** cannot create or write to **DAPPARAMS.DAT** it will terminate and **DAPBOOT** will fail.

The engineer's log file (**AMTDAP_1.LOG**) will record error information in the case of a DAP fault. If the DAP fails at any time, please report the information in **AMTDAP_1.LOG** to AMT.

More information on **AMTDAP_1.LOG** is given in section 5.1 on pages 39-41.

Chapter 5

Maintaining DAP system files

There are a variety of files used by the DAP system software to record DAP specific information (for example DAP resource limits, error logging and accounting). This chapter describes the purpose of these files and how they are used.

5.1 Engineer's log file AMTDAP_1.LOG

The DAP engineer's log file **AMTDAP_1.LOG** is stored in the directory **SYS$MANAGER** and records the following information:

- The progress of the DAP bootstrap. For each attempt to bootstrap the DAP the **DAPMONITOR** program outputs the following information to **AMTDAP_1.LOG**:
  - The date, time and host-DAP interface used
  - The name of the configuration file
  - Whether one or both PE banks are selected
  - The times at which the DAP was last powered up and down
  - The files being down loaded to the DAP
  - The DAP clock speed.

  Figure 5.1 on the next page shows an example of a successful bootstrap of the DAP 500 recorded in **AMTDAP_1.LOG**

  In addition, if the **DAPMONITOR** program is stopped using either the **RESTART** or **STOP** parameter to **DAPBOOT** (see section 4.2 pages 34-35), a message of the form:

```
date time - DAPBOOT action used to stop DAPMONITOR
```

is recorded in **AMTDAP_1.LOG**, where *action* is either **RESTART** or **STOP**.

If the DAP bootstrap fails then, depending on the cause of the error, the following diagnostic information may be recorded in the engineer's log file.

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

Dapboot invoked on 24-FEB-1989 14:28:09.54 via DR11 parallel line

Configuration filename: SYS$MANAGER:DAP_1.CFG

Both PE banks selected

Dap last powered up:        Thu Feb 23 11:45:04 1989
Dap last powered down:      Thu Feb 23 08:01:05 1989

DAP confidence tests successfully completed
Message Server successfully loaded from file 'SYS$SYSTEM:MSGS.HCU'
MCUCP successfully loaded from file 'SYS$SYSTEM:MCUCP_DPIO5.DEX'
HCUCP successfully loaded from file 'SYS$SYSTEM:HCUCP_DPIO5.HCU'

Clock speed: 100 ns
```

Fig 5.1: An example of a successful bootstrap of the DAP 500

- A summary of any non-fatal errors and exceptions recorded when the DAP was running, and stored in the NVRAM of the DAP. These summaries are preceded by the messages `Error dump` and `Exceptions dump` respectively
- DAP failures. If a DAP hardware fault or fatal software error occurs when the DAP is running, a summary of the machine registers and other locations is dumped to the engineer's log file.

  Such a dump is preceded by the message:

```
HCUCP dump following error on date time
```

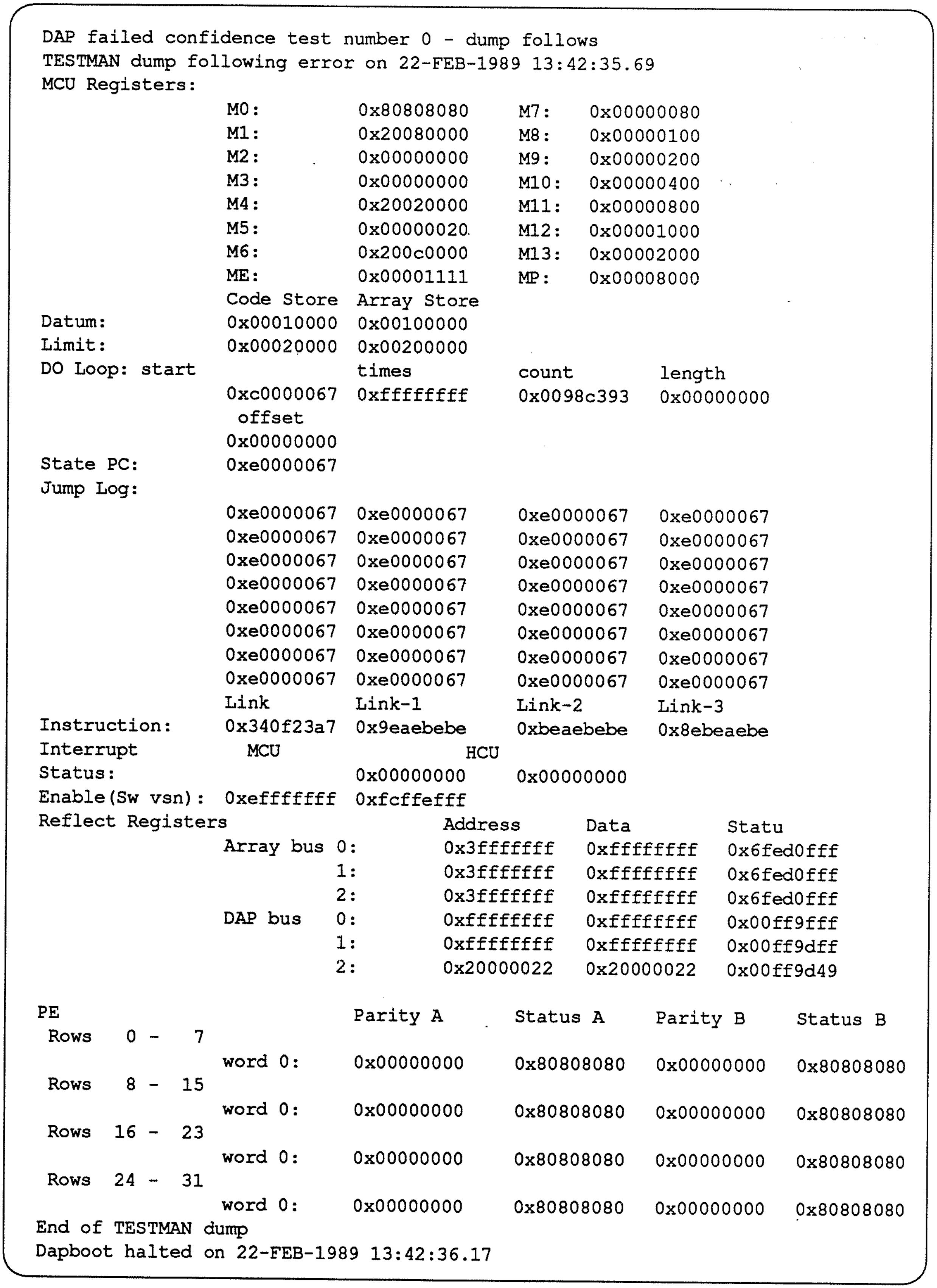
A similar dump is generated if the DAP fails one of its confidence tests during the bootstrap procedure and is preceded by the message:

```
TESTMAN dump following error on date time
```

If either type of dump occurs it should be reported to AMT.

An example of a dump produced during the bootstrap procedure is given in Figure 5.2 on the page opposite.

The engineer's log file is created (if it does not already exist) by the **DAPMONITOR** program; if the file does exist, **DAPMONITOR** appends to it.



```
DAP failed confidence test number 0 - dump follows
TESTMAN dump following error on 22-FEB-1989 13:42:35.69
MCU Registers:
                M0:         0x80808080    M7:    0x00000080
                M1:         0x20080000    M8:    0x00000100
                M2:         0x00000000    M9:    0x00000200
                M3:         0x00000000    M10:   0x00000400
                M4:         0x20020000    M11:   0x00000800
                M5:         0x00000020    M12:   0x00001000
                M6:         0x200c0000    M13:   0x00002000
                ME:         0x00001111    MP:    0x00008000
                Code Store  Array Store
Datum:          0x00010000  0x00100000
Limit:          0x00020000  0x00200000
DO Loop: start              times         count        length
                0xc0000067  0xffffffff    0x0098c393   0x00000000
                 offset
                0x00000000
State PC:       0xe0000067
Jump Log:
                0xe0000067  0xe0000067    0xe0000067   0xe0000067
                0xe0000067  0xe0000067    0xe0000067   0xe0000067
                0xe0000067  0xe0000067    0xe0000067   0xe0000067
                0xe0000067  0xe0000067    0xe0000067   0xe0000067
                0xe0000067  0xe0000067    0xe0000067   0xe0000067
                0xe0000067  0xe0000067    0xe0000067   0xe0000067
                0xe0000067  0xe0000067    0xe0000067   0xe0000067
                0xe0000067  0xe0000067    0xe0000067   0xe0000067
                Link        Link-1        Link-2       Link-3
Instruction:    0x340f23a7  0x9eaebebe    0xbeaebebe   0x8ebeaebe
Interrupt         MCU                 HCU
Status:                     0x00000000    0x00000000
Enable(Sw vsn): 0xefffffff  0xfcffefff
Reflect Registers                   Address     Data        Statu
                Array bus 0:        0x3fffffff  0xffffffff  0x6fed0fff
                          1:        0x3fffffff  0xffffffff  0x6fed0fff
                          2:        0x3fffffff  0xffffffff  0x6fed0fff
                DAP bus   0:        0xffffffff  0xffffffff  0x00ff9fff
                          1:        0xffffffff  0xffffffff  0x00ff9dff
                          2:        0x20000022  0x20000022  0x00ff9d49

PE                          Parity A      Status A     Parity B     Status B
 Rows    0 -   7
                word 0:     0x00000000    0x80808080   0x00000000   0x80808080
 Rows    8 -  15
                word 0:     0x00000000    0x80808080   0x00000000   0x80808080
 Rows   16 -  23
                word 0:     0x00000000    0x80808080   0x00000000   0x80808080
 Rows   24 -  31
                word 0:     0x00000000    0x80808080   0x00000000   0x80808080
End of TESTMAN dump
Dapboot halted on 22-FEB-1989 13:42:36.17
```

Fig 5.2 An example of a dump produced during the bootstrap procedure for a DAP 500

5.5 AMT Text Libraries

The DAP basic software contains several VMS text libraries which are stored in the directory **SYS$LIBRARY** and are available for general use.

5.5.1 DAPFDEF.TLB

The text library **DAPFDEF.TLB** contains one module, **PATTERNS**, which can be included in a FORTRAN-PLUS source file and makes several useful bit patterns available to the program (see *DAP Series: FORTRAN PLUS Language*, Appendix A, for more details). The **PATTERNS** module does not have to be extracted from the library since the FORTRAN-PLUS compiler automatically searches **DAPFDEF.TLB** for a text module included using the **#include** directive in the form:

```
#include module-name
```

5.5.2 APALDEF.TLB

The text library **APALDEF.TLB** contains one module, **USRMACS**, which can be included in an **APAL** source file. **USRMACS** makes several useful bit patterns available to the program and also provides a set of macros designed to simplify the task of interfacing between different code sections (see *DAP Series: APAL Language*, chapters 9 and 10). The APAL assembler automatically searches **APALDEF.TLB** for a text module included using the **#include** directive as shown above.

5.5.3 AMTFORDEFS.TLB and AMTCDEFS.TLB

The text libraries **AMTFORDEFS.TLB** and **AMTCDEFS.TLB** each contain a module, **AMTDEF**, which can be included in **VAX FORTRAN** and **VAX C** source files respectively to provide symbolic names for the DAP accounting packet structure given in table 6.3 on page 48.

Chapter 6

Accounting

6.1 DAP system ACCOUNTING facilities

AMT software generates accounting information about DAP processes in two forms:

- An ASCII text file which contains a summary of each DAP process
- User data entries in the standard VMS accounting file

You can use the data recorded in these files to learn how the system performs under different circumstances and how different groups of users use the DAP; you can also use the data for billing purposes if necessary.

6.2 The DAP system log

When a DAP process terminates – that is, **DAPREL** is executed or the host process terminates – the **DAPMONITOR** program adds an ASCII text record to the file **SYS$MANAGER:DAPSYS_1.LOG** giving the following information in tabular form:

username	The VAX username of the process owner
time loaded	The absolute time at which the process completed loading
time unloaded	The absolute time at which the process completed unloading
array size	The amount of array store requested by the program (in planes)
code size	The amount of code store requested by the program (in bytes)
CPU time	The amount of DAP CPU time used (in milliseconds)
timeslice	The final timeslice used by the process
program name	The name of the DAP executable program which was run (truncated to at most 32 characters by the VMS library routine **LIB$TRIM_FILESPEC**)

If the program fails to finish loading (for example, where the load is interrupted) then the message:

```
***Load Abandoned***
```

is written across the columns normally used for **time loaded**, **array size**, **code size**, **CPU time** and **timeslice**.

The **DAPSYS_1.LOG** file also contains records indicating every time **DAPBOOT** is invoked and every time a change is made to the system parameter **Quantum**.

These **DAPSYS_1.LOG** records have the form:

```
DAPBOOT invoked on date time
date time Quantum set to n
```

You can suppress the logging of accounting information to the **DAPSYS_1.LOG** file by invoking **DAPBOOT** with the qualifier:

/ACCOUNTING=NOASCII

Note that AMT reserves the right to change the format and content of the data sent to **DAPSYS_1.LOG** in future releases.

6.3 VMS ACCOUNTING

When a process terminates – that is, **DAPREL** is executed or the host process terminates – the **DAPMONITOR** program appends a record to the VMS system accounting file **SYS$MANAGER:ACCOUNTNG.DAT**. Each record consists of a record header and an identifier packet, followed by a binary accounting packet of type **User data** (refer to *VAX/VMS Accounting Utility Reference Manual* (VMS V4 users) or *VMS Accounting Utility Manual* (VMS V5 users) for details of accounting packets).

Figure 6.1 below shows a block diagram of the layout of fields in a user data packet. The layout conforms to the VMS convention in which data is represented as a series of 4-byte strips, with the lowest address byte being at the top right of the representation. The **Count** field holds the number of bytes in **User data**, and occupies the first byte of the user data.

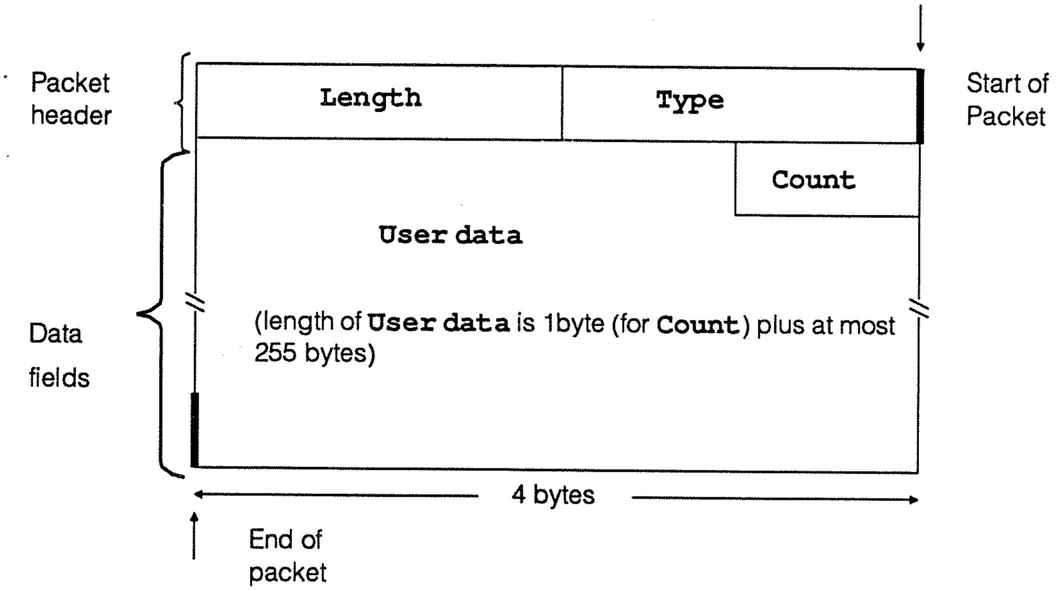


Figure 6.1 Block diagram of a user data packet

Table 6.1 below describes the fields contained in the packet; their symbolic offsets from start-of-packet, and their contents.

Field	*Symbolic offset from start of packet*	*Contents (length)*
Type	**ACR$W_TYPE**	See table 6.2 (1 word)
Length	**ACR$W_LENGTH**	Total length of the packet (1 word)
User data (including **Count**)	**ACR$T_USER_DATA**	AMT data (counted string)

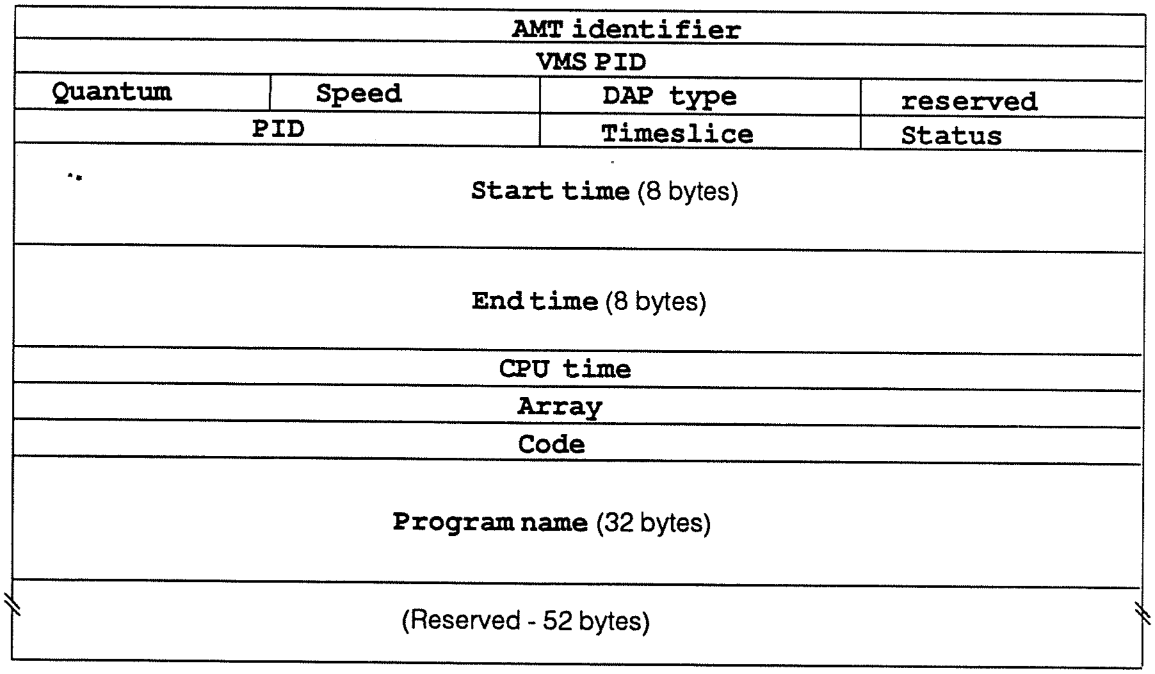
Table 6.1 Field descriptions of a user data packet

The layout of the **Type** field in the user data packet is set out in table 6.2 below:

Field	*Symbolic offset from start-of-packet*	*Contents (length)*
Header	**ACR$V_PACKET**	Identifies the item as a packet if the field is set to 1 (1 bit)
Type	**ACR$V_TYPE**	The user packet type code (7 bits) **ACR$K_USER_DATA**

Table 6.2 Sub-field descriptions for field **Type**

The organisation of the sub-fields in the **User data** field is shown in figure 6.2 below.



AMT identifier

VMS PID

Quantum | **Speed** | **DAP type** | **reserved**

PID | **Timeslice** | **Status**

Start time (8 bytes)

End time (8 bytes)

CPU time

Array

Code

Program name (32 bytes)

(Reserved - 52 bytes)

Figure 6.2 Block diagram of the subfields in the **User data** *field in a user data packet*

Chapter 7

Trouble Shooting

This chapter attempts to cover some of the more common problems that you may experience in installing a DAP and attempting to use **DAPBOOT** to start up the DAP.

- The software installation procedure fails:

  Action:

  - Check that you have sufficient disk space to perform the installation (see section 2.3 pages 14-15)
  - Ensure that you are logged on as **SYSTEM**. Software installation may require certain privileges that you do not have if you are not using the **SYSTEM** account

- Execution of **DAPSTARTUP.COM** either manually or during system startup produces errors. This will occur if:

  The adapter name is wrong – DR11-W/DRV11-WA only

  The CSR is invalid or already in use – DR11-W/DRV11-WA only (see section 1.4.1.2 pages 8-10)

  The interrupt vector is invalid or already in use – DR11-W/DRV11-WA only (see section 1.4.1.2 pages 8-10)

  The VAXBI node number is invalid or already in use – DRB32 only (see section 1.3.1.2 pages 4-5)

  Action:

  - Confirm the hardware settings with the DEC engineer. For DR11-W/DRV11-WA you need to know the Unibus/Qbus to which the device is attached, the CSR address and interrupt vector address of the device. For DRB32 devices, confirm the VAXBI node number with the Digital engineer
  - Ensure that **SYS$MANAGER:DAPSTARTUP.COM** has been modified to reflect these values (see chapter 1, page 18)

- Running **DAPBOOT** returns the message:

```
Cannot open configuration file file-name
```

This occurs if **DAPCONFIG** has not been run

Action:

- To execute **DAPCONFIG** use the command:

```
$ @SYS$MANAGER:DAPCONFIG
```

generally means that the device (physical) settings and the VMS **(DAPSTARTUP)** settings match. If the address does not respond with anything similar to FFE0 there may well be a mismatch.

- ▫ Check the cables between the VAX and the DAP. Ensure that they are crossed J1 <--> J2 (DR11-W/DRV11-WA only)

- **DAPBOOT** terminates immediately with the message:

```
You need operator (OPER) privilege to run DAPBOOT
```

Action:

- ▫ Your VAX system manager must authorize the granting of OPER privilege to you.

Index

Reader comment form

AMT
Reading, UK

Any comments you care to make, whether reporting bugs in the manual or making more general comment, about this or any AMT publications will help us improve their quality and usefulness. To report bugs, if you have the time, the ideal way from our point of view is to send us a photo-copy of the relevant page, with the bug marked on it. If you are in the UK, please use our FREEPOST address to send us the copy.

If you also can spare the time to fill in the mini-questionnaire below that would be doubly useful to us. To send us this form, please fold it as indicated, and post it – postage is pre-paid for the UK.

Comments

Title of publication: **System Management under VAX/VMS** (man020.01) / other – please specify:

. . . .

My name and job title: . . . .

My department: . . . .

My company: . . . .

My company address: . . . .

. . . .

My telephone number – country: . . . . number: . . . .

I used the publication:

□ As an introduction to the subject

□ To teach myself

□ To teach others

□ As a reference manual

□ Other – please specify

I found the contents:

	True	Partly true	Not true
Helpful	□	□	□
Accurate	□	□	□
Written clearly	□	□	□
Well illustrated	□	□	□
Well indexed	□	□	□
Other – please specify	□	□	□

. . . .

Thank you for your help.

23 May 89

Second fold

First fold

Third fold

Third fold

No postage needed for posting in the UK.
If posting outside UK, please stick stamps to normal value.

Publications Manager
Active Memory Technology Ltd
FREEPOST (RG 1436)
Reading
Berkshire RG6 1BR
United Kingdom

Fourth fold

Fourth fold

Tuck into third fold

Second fold

First fold