EDUCATIONAL SERVICES

VAX/VMS
Training

VAX/VMS
Device Driver
Driver Tables

digital

EY-2278E-ME-0001

DRIVER TABLES

Prepared by Educational Services
of
Digital Equipment Corporation

First Edition, October 1984

Printed in U.S.A.

The manuscript for this book was created using DIGITAL Standard Runoff. Book production was done by Educational Services Development and Publishing in Nashua, NH.

CONTENTS

FIGURES

EXAMPLES

INTRODUCTION

Every device driver declares three tables that describe the device and the driver:

- Driver Prologue Table (DPT)
- Driver Dispatch Table (DDT)
- Function Decision Table (FDT)

In addition, most drivers define their own extensions to the standard UCB (in which device-specific information, describing the current I/O operation, is stored). This module describes the fields in which data structures need to be initialized by the driver tables, how extensions are made to the UCB, and which system macros should be used to create the tables listed above.

OBJECTIVES

On completion of this module, you will be able to:

1. Identify the data structures that need to be defined in a driver (and their contents).

2. Identify the fields in other data structures that need to be defined in a driver (and their function).

3. Use the DPT, DDTAB, and FUNCTAB macros to define the driver prologue table, driver dispatch table, and function decision table.

RESOURCE

1. Guide to Writing a Device Driver for VAX/VMS

TOPICS

- Creating the following driver tables:
  - Driver Prologue Table (DPT)

    Used by SYSGEN to load driver
  - Driver Dispatch Table (DDT)

    Used to list driver routines addresses
  - Function Decision Table (FDT)

    List of functions vs. preprocessing routine addresses
- Fields within the driver tables that need to be defined
- Usage of macros to create the driver tables:
  - DPT macro (Driver Prologue Table)
  - DDTAB macro (Driver Dispatch Table)
  - FUNCTAB macro (Function Decision Table)

DRIVER DATA STRUCTURES REQUIRING INITIALIZATION

The following I/O data structures (or certain fields within the data structures) may need to be initialized by the driver:

- Device Data Block (DDB)
- Channel Request Block (CRB)
- Unit Control Block (UCB)
- Driver Dispatch Table (DDT)
- Function Decision Table (FDT)

Some tables are produced in the driver image file as a result of driver source code and/or macro invocations. These include the Driver Prologue Table (DPT), DDT, and the FDT. These tables are loaded with the driver code into nonpaged pool by the driver loading utility SYSGEN.

Other data structures (UCB, CRB, DDB) are created in nonpaged pool by SYSGEN. Only **offsets** into these data structures are defined by driver source code. Some fields in these data structures are initialized by SYSGEN when they are created. The initial values for these fields are found in the DPT.

There are macros to define these data structures and their offsets. First, a detailed description of the important fields within each structure (that may need to be initialized) is presented. Next, the macros that define the structures are discussed.

Consult the Guide to Writing a Device Driver for VAX/VMS for a complete description of all of the fields in each I/O data structure.

Device Data Block (DDB) Fields

DDB$L_DDT*	Address of the driver dispatch table; the driver prologue table must specify a value for this field.
DDB$L_ACPD	Name of the default ACP for the controller. For file-structured devices, this field contains the first four letters of the name of an ACP that controls access to the devices (the last three letters of the name are ACP).

* These fields must be initialized by the driver. Other fields are optional.

Channel Request Block (CRB) Fields

CRB$L_INTD+4 * CRB$L_INTD2+4	Address(es) of interrupt service routine(s); more than one needs to be specified if the device can interrupt at more than one vector location. Initialized by driver prologue table.**
CRB$L_INTD+ VEC$L_INITIAL	The address of the controller initialization routine. Initialized by the driver prologue table.
CRB$L_INTD+ VEC$L_UNITINIT	The address of the unit initialization routine. Initialized by the driver prologue table.***

* This field must be initialized by the driver. Other fields are optional.

** The following formula may be used to initialize vector number n (for a device with more than 2 vectors):

CRB$L_INTD+4+<n-1>*VEC$K_LENGTH

*** Note that there is a field in the DDTAB macro (explained later) for the unit initialization routine. If a value is specified there, that routine is called regardless of what is placed in CRB$L_INTD+VEC$L_UNITINIT. If no address is specified in DDTAB for the unit initialization routine, then the routine specified in CRB$L_INTD+ VEC$L_UNITINIT is called. There is only one controller and one unit initialization routine for the driver (regardless of the number of interrupt vectors).

Unit Control Block (UCB) Fields

UCB$B_FIPL *	Fork IPL at which the driver code executes (usually 8).
UCB$L_DEVCHAR *	The driver prologue table must specify all device characteristics which the device possesses.
UCB$B_DEVCLASS	May be used to define a device class for the device in the driver prologue table (used by error-logging routines). For user-written drivers, use a value in the range 250-255.
UCB$B_DEVTYPE	May be used to define a device type in the driver prologue (also used in error-logging). For user-written drivers, use a value in the range 250-255.
UCB$W_DEVBUFSIZ	Default buffer size for device. Drivers with set mode and device characteristics functions rewrite the value in this field with data supplied in a $QIO request. This field is initialized in the driver prologue table, and is device-dependent.
UCB$L_DEVDEPEND	The driver prologue table may be used to initialize this field, which contains any device-dependent data that the driver-writer chooses to place in this field.
UCB$B_DIPL *	Device IPL at which the device requests hardware interrupts. This field is initialized by the driver prologue table (value in range 20-23).
UCB$W_DEVSTS	Device-dependent status; read and written only by the driver.

* These fields must be initialized by the driver. Other fields are optional.

Driver Dispatch Table (DDT) Fields

The following fields are initialized by the DDTAB macro.

DDT$L_START *	Address of the driver's start I/O routine.
DDT$L_UNSOLINT	Address of the driver's unsolicited interrupt routine (used by MASSBUS drivers).
DDT$L_FDT *	Address of the driver's function decision table (FDT).
DDT$L_CANCEL *	Address of the driver's cancel I/O routine.
DDT$L_REGDUMP	Address of a register dump routine called when performing error logging and/or diagnostics.
DDT$W_DIAGBUF	Size of diagnostic buffer (in bytes).
DDT$W_ERRORBUF	Size of the error log buffer (in bytes).
DDT$L_UNITINIT	Address of the driver's unit initialization routine.
DDT$_ALTSTART	Address of the driver's alternate start I/O entry point.
DDT$L_MNTVER	Address of VAX/VMS routine IOC$MNTVER called at start and end of MOUNT verification operation. Use of other routines reserved to DIGITAL.
DDT$W_FDTSIZE	Number of bytes in the function decision table. Used to relocate FDT addresses to system addresses.

* These fields must be initialized by the driver. Other fields are optional.

Function Decision Table (FDT) Fields

The FDT is built using repeated invocations of the FUNCTAB macro. No symbolic names are given to the fields in the FDT. All entries are either a quadword mask of function codes, or the address of an FDT routine to service the function codes.

DRIVER PROLOGUE TABLE (DPT)

- Must be defined at the beginning of a driver, ahead of any code that takes up memory.
- Is loaded with driver into nonpaged system memory when the driver is loaded (i.e., SYSGEN).
- Contains initialization information about the DDB, UCB and CRB.
- Contains information about the driver (size, name, etc.).
- Information in the DPT is used to construct the DDB, UCB, and CRB when units are connected.
- All DPTs are linked together (with listhead at IOC$GL_DPTLIST).

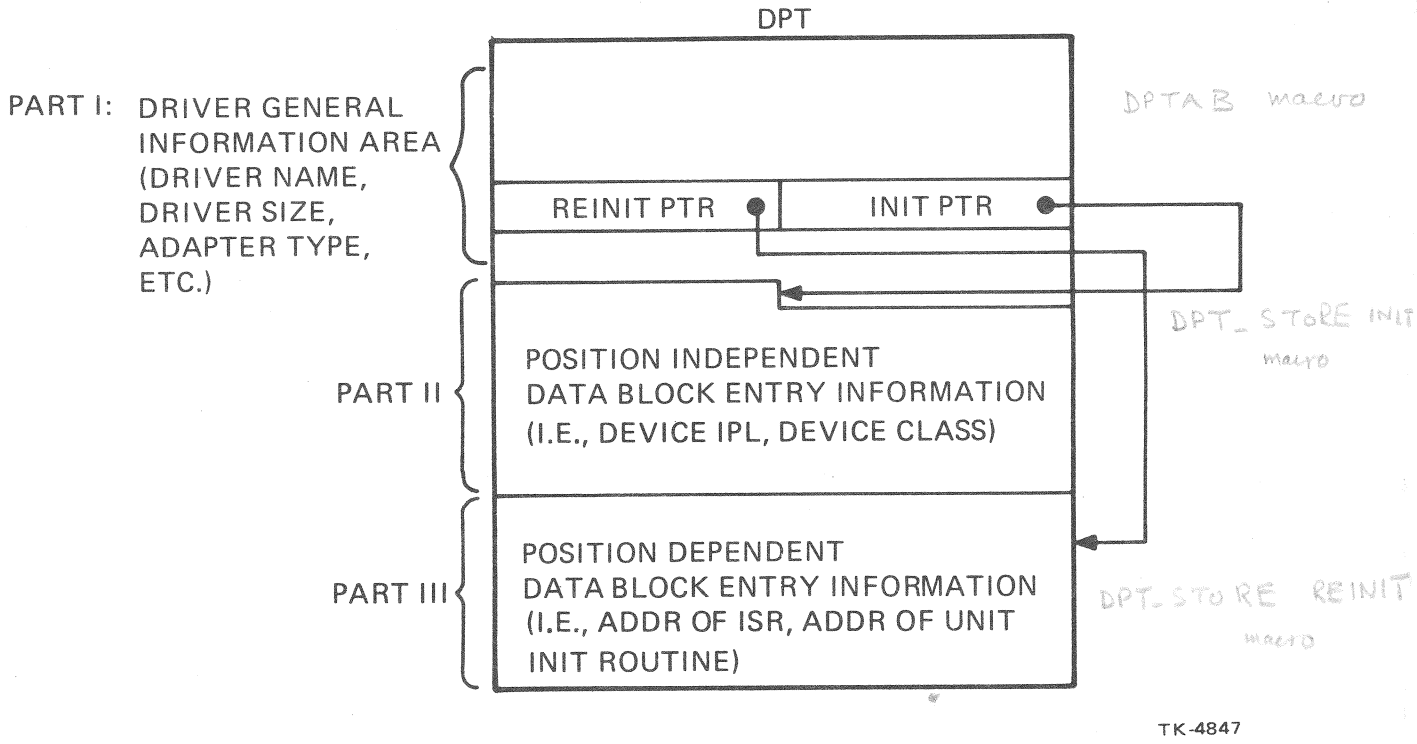

Figure 5-1 Driver Prologue Table

Macros Used to Create the DPT

The first macro invoked in creating the DPT is the DPTAB macro. It has a number of parameters used to specify general driver information.

```
DPTAB -                 ;DEFINE DPT (Builds Part I; see Figure 5-1)

        END=aaa,-       ;aaa=LABEL OF DRIVER .END STATEMENT

        ADAPTER=bbb,-   ;bbb=ADAPTER TYPE (UBA, MBA, DR or NULL)

        FLAGS=ccc,-     ;ccc=DPT$M_SVP      ;SYSTEM PAGE TABLE
                                            ;REQUIRED BY DRIVER
                        ;=DPT$M_SUBCNTRL    ;DEVICE IS UNDER
                                            ;A SUBCONTROLLER (MASSBUS)
                        ;=DPT$M_NOUNLOAD    ;DRIVER CANNOT BE RELOADED

        UCBSIZE=ddd,-   ;ddd=UCB SIZE IN BYTES

        UNLOAD=eee,-    ;eee=ADDR OF SUBROUTINE TO BE INVOKED WHEN
                        ;DRIVER IS RELOADED

        MAXUNITS=fff,-  ;fff=MAXIMUM NUMBER OF UNITS (WILL DETERMINE
                        ;SIZE OF IDB).  DEFAULT IS 8.

        VERSION=ggg,-   ;ggg=VERSION NUMBER TO IDENTIFY FORMAT OF
                        DPT (DEFAULT FROM DPTAB MACRO).  VALUE
                        CHECKED BY SYSGEN.  ERROR GENERATED, IF
                        VALUE NOT "CURRENT" VALUE.

        DEFUNITS=hhh,-  ;hhh=NUMBER OF UCBs TO CONFIGURE.  ALSO THE
                        NUMBER OF TIMES DELIVER ROUTINE CALLED.

        DELIVER=iii,-   ;iii=ADDRESS  OF  ROUTINE  CALLED  AT
                        AUTOCONFIGURE TIME.

        VECTOR=jjj,-    ;jjj=ADDRESS OF DRIVER-SPECIFIC VECTOR,
                        RESERVED TO DIGITAL.

        NAME=lll        ;lll=DRIVER NAME (xyDRIVER by convention)
```

```
141 ;
142 ; DRIVER PROLOGUE TABLE
143 ;
144
145         DPTAB   -                           ;DEFINE DRIVER PROLOGUE TABLE
146                 END=LP_END,-                ;END OF DRIVER
147                 ADAPTER=UBA,-               ;ADAPTER TYPE
148                 UCBSIZE=UCB$K_LPLENGTH      ;UCB SIZE
149                 NAME = LPDRIVER             ;DRIVER NAME
```

Example 5-1 The DPTAB of the LPDRIVER

General Structure of the DPT

Following the DPTAB macro, a series of DPT_STORE macros are invoked to specify position independent and position dependent data block entries (i.e., initial values to be placed in various data structures after they are created by SYSGEN).

The INIT section indicates the start of fields to initialize when the driver is LOADed (by SYSGEN). The REINIT section indicates the start of additional fields to initialize when the driver is LOADed or RELOADed.

```
(Builds Part II; see Figure 5-1)

DPT_STORE       INIT        ; This macro signifies the beginning
                              of position independent data block
                              entry information.
    •
    •           Position independent DPT_STORE macros.
    •           (Initialize values in data structures.)

(Builds Part III; see Figure 5-1)

DPT_STORE       REINIT      ; This macro signifies the beginning
                              of position dependent data block
                              entry information.
    •
    •
    •           Position dependent DPT_STORE macros.
                (Initialize addresses in data structures.)


DPT_STORE       END         ; This macro signifies the end of all
                              DPT macros.
```

Both position dependent and position independent DPT_STORE macros use the following general format (refer to Example 5-2).

```
DPT_STORE  aaa,bbb,c,ddd,[eee],[fff]
```

where

aaa = Data block type (DDB,UCB,CRB,IDB).

bbb = Symbolic offset of data block entry (e.g., UCB$B_FIPL).

c = B for byte entry, W for word entry, L for long word entry, D for driver address entry, V for bit field.

ddd = Parameter to be stored.

eee = Starting bit position within specified field (specified only if entry c = V).

fff = Number of bits in field (specified only when c = V).

NOTE: The DPT_STORE macro can only be used to initialize the first 256 bytes of a data structure.

```
150         DPT_STORE INIT                      ;CONTROL BLOCK INIT VALUES
151         DPT_STORE UCB,UCB$B_FIPL,B,8        ;FORK IPL
152         DPT_STORE UCB,UCB$L_DEVCHAR,L,-     ;DEVICE CHARACTERISTICS
153                 <DEV$M_REC-                 ; RECORD ORIENTED
154                 !DEV$M_AVL-                 ; AVAILABLE
155                 !DEV$M_CCL-                 ; CARRIAGE CONTROL DEVICE
156                 !DEV$M_ODV>                 ; OUTPUT DEVICE
157         DPT_STORE UCB,UCB$B_DEVCLASS,B,DC$_LP ;DEVICE CLASS
158         DPT_STORE UCB,UCB$B_DEVTYPE,B,LP$_LP11 ;DEVICE TYPE
159         DPT_STORE UCB,UCB$W_DEVBUFSIZ,W,132 ;DEFAULT BUFFER SIZE
160         DPT_STORE UCB,UCB$L_DEVDEPEND,L,<64@24+LP$M_MECHFORM> ;PRINTER PARAMETERS
161         DPT_STORE UCB,UCB$B_DIPL,B,20       ;DEVICE IPL
162         DPT_STORE UCB,UCB$L_LP_MUTEX,W,-1   ;INITIALIZE MUTEX
163         DPT_STORE REINIT                    ;CONTROL BLOCK RE-INIT VALUES
164         DPT_STORE CRB,CRB$L_INTD+4,D,LP$INT ;INTERRUPT SERVICE ROUTINE ADDRESS
165         DPT_STORE CRB,CRB$L_INTD+VEC$L_INITIAL,D,LP_LX11_CINIT ;CONTROLLER INIT
166         DPT_STORE CRB,CRB$L_INTD+VEC$L_UNITINIT,D,LP_LX1$_INIT ;UNIT INIT
167         DPT_STORE DDB,DDB$L_DDT,D,LP$DDT    ;DDT ADDRESS
168         DPT_STORE END                       ;
```

Example 5-2 The Lineprinter Driver DPT_STORE Macros

DDTAB MACRO

The DDTAB macro generates the complete Driver Dispatch Table (DDT) according to the parameters specified (see Example 5-3).

```
DDTAB  DEVNAM   = aa,-      ;aa   = DEVICE MNEMONIC

       START    = bbb,-     ;bbb  = DRIVER START I/O ENTRY
                                    ADDRESS

       UNSOLIC  = ccc,-     ;ccc  = DRIVER UNSOLICITED INTERRUPT
                                    ENTRY ADDRESS (MASSBUS ONLY)

       FUNCTB   = ddd,-     ;ddd  = FUNCTION DECISION TABLE
                                    BEGINNING ADDRESS

       CANCEL   = eee,-     ;eee  = DRIVER CANCEL I/O ENTRY
                                    ADDRESS PRECEDED BY "+" SIGN
                                    IF ADDRESS IS WITHIN THE
                                    EXECUTIVE

       REGDMP   = fff,-     ;fff  = DRIVER ERR. LOG./DIAG-
                                    NOSTICS REGISTER DUMP ROUTINE
                                    ADDRESS

       DIAGBF   = ggg,-     ;ggg  = SIZE OF DIAGNOSTIC BUFFER

       ERLGBF   = hhh,-     ;hhh  = SIZE OF ERR. LOG. BUFFER

       UNITINIT = iii,-     ;iii  = ADDRESS OF UNIT INITIAL-
                                    IZATION ROUTINE

       ALTSTART = jjj,-     ;jjj  = ADDRESS OF ALTERNATE START
                                    I/O ENTRY POINT

       MNTVER   = kkk,-     ;kkk  = IOC$MNTVER
```

NOTE: The device mnemonic "aa" is used to generate the symbolic address "aa$DDT" for the DDT. This address has to be stored in the DDB.

```
170 ;
171 ; DRIVER DISPATCH TABLE
172 ;
173
174         DDTAB   LP,-                        ;DRIVER DISPATCH TABLE
175                 STARTIO,-                   ;START I/O OPERATION
176                 0,-                         ;UNSOLICITED INTERRUPT
177                 FUNCTABLE,-                 ;FUNCTION TABLE
178                 +IOC$CANCELIO,-             ;CANCEL I/O
179                 0,-                         ;REGISTER DUMP ROUTINE
180                 0,-                         ;SIZE OF DIAGNOSTIC BUFFER
181                 0                           ;SIZE OF ERROR LOG BUFFER
```

external routine

Example 5-3 The DDTAB Macro in LPDRIVER

```
258 ; Driver dispatch table
259
260         DDTAB   -                                   ; DDT-creation macro
261                 DEVNAM=XA,-                         ; Name of device
262                 START=XA_START,-                    ; Start I/O routine
263                 FUNCTB=XA_FUNCTABLE,-               ; FDT address
264                 CANCEL=XA_CANCEL,-                  ; Cancel I/O routine
265                 REGDMP=XA_REGDUMP,-                 ; Register dump routine
266                 DIAGBF=<<13*4>+<<3+5+1>*4>>,-       ; Diagnostic buffer size
267                 ERLGBF=<<13*4>+<1*4>+<EMB$L_DV_REGSAV>> ; Error log buffer size
268
```

Example 5-4 The DDTAB Macro in XADRIVER

FUNCTAB MACRO

The FUNCTAB macro generates an FDT entry.

```
FUNCTAB  aaa,<bbb,.....,nnn>
```

where

aaa = FDT routine address (optional)

bbb,.....,nnn = Bit values to be set in the function mask corresponding to aaa. Refer to Chapter 7 of the Guide to Writing a Device Driver for VAX/VMS manual for system defined symbolic bit values.

When setting up the legal function mask and buffered I/O function mask, the FUNCTAB macro must be invoked without the FDT routine address (see Example 5-4). Note that when specifying function codes, the IO$_ prefix is omitted. When debugging your driver, it is a good idea to add a last FDT mask of all legal function codes, and an error handling routine to catch FDT dispatching errors.

```
184 ;
185 ; LP11/LS11/LV11 FUNCTION DECISION TABLE
186 ;
187
188 FUNCTABLE:                                      ;FUNCTION DECISION TABLE
189         FUNCTAB ,-                              ;LEGAL FUNCTIONS
190                 <SENSECHAR,-                    ;SENSE CHARACTERISTICS
191                  SETCHAR,-                      ;SET CHARACTERISTICS
192                  SENSEMODE,-                    ;SENSE MODE
193                  SETMODE,-                      ;SET MODE
194                  WRITELBLK,-                    ;WRITE LOGICAL BLOCK
195                  WRITEPBLK,-                    ;WRITE PHYSICAL BLOCK
196                  WRITEVBLK>                     ;WRITE VIRTUAL BLOCK
197         FUNCTAB ,-                              ;BUFFERED FUNCTIONS
198                 <SENSECHAR,-                    ;SENSE CHARACTERISTICS
199                  SETCHAR,-                      ;SET CHARACTERISTICS
200                  SENSEMODE,-                    ;SENSE MODE
201                  SETMODE,-                      ;SET MODE
202                  WRITELBLK,-                    ;WRITE LOGICAL BLOCK
203                  WRITEPBLK,-                    ;WRITE PHYSICAL BLOCK
204                  WRITEVBLK>                     ;WRITE VIRTUAL BLOCK
205         FUNCTAB LP_WRITE,<WRITELBLK,WRITEPBLK,WRITEVBLK> ;WRITE FUNCTIONS
206         FUNCTAB LP_SETMODE,<SETCHAR,SETMODE> ;SET CHARACTERISTICS FUNCTIONS
207         FUNCTAB +EXE$SENSEMODE,-                ;
208                 <SENSECHAR,-                    ;SENSE CHARACTERISTICS
209                  SENSEMODE>                     ;SENSE MODE
```

Example 5-5 The Function Decision Table for LPDRIVER

digital

EY-2278E-ME-0001
Printed in U.S.A.