```
EEEEEEEEEE  XX      XX    AAAAAA    MM      MM  PPPPPPPP    LL          EEEEEEEEEE    SSSSSSSS
EEEEEEEEEE  XX      XX    AAAAAA    MM      MM  PPPPPPPP    LL          EEEEEEEEEE    SSSSSSSS
EEEEEEEEEE  XX      XX    AAAAAA    MM      MM  PPPPPPPP    LL          EEEEEEEEEE    SSSSSSSS
EE          XX      XX   AA    AA   MMMM  MMMM  PP      PP  LL          EE          SS
EE          XX      XX   AA    AA   MMMM  MMMM  PP      PP  LL          EE          SS
EE          XX      XX   AA    AA   MMMMMMMMMM  PP      PP  LL          EE          SS
EE            XX  XX     AA    AA   MM  MM  MM  PP      PP  LL          EE          SS
EE            XX  XX     AA    AA   MM  MM  MM  PP      PP  LL          EE          SS
EE            XX  XX     AA    AA   MM  MM  MM  PP      PP  LL          EE          SS
EEEEEEEE        XX       AA    AA   MM      MM  PPPPPPPP    LL          EEEEEEEE      SSSSSS
EEEEEEEE        XX       AA    AA   MM      MM  PPPPPPPP    LL          EEEEEEEE      SSSSSS
EEEEEEEE        XX       AA    AA   MM      MM  PPPPPPPP    LL          EEEEEEEE      SSSSSS
EE            XX  XX     AAAAAAAAAA   MM      MM  PP          LL          EE                  SS
EE            XX  XX     AAAAAAAAAA   MM      MM  PP          LL          EE                  SS
EE            XX  XX     AAAAAAAAAA   MM      MM  PP          LL          EE                  SS
EE          XX      XX   AA    AA   MM      MM  PP          LL          EE                  SS
EE          XX      XX   AA    AA   MM      MM  PP          LL          EE                  SS
EE          XX      XX   AA    AA   MM      MM  PP          LL          EE                  SS
EEEEEEEEEE  XX      XX   AA    AA   MM      MM  PP          LLLLLLLLLL  EEEEEEEEEE  SSSSSSSS
EEEEEEEEEE  XX      XX   AA    AA   MM      MM  PP          LLLLLLLLLL  EEEEEEEEEE  SSSSSSSS
EEEEEEEEEE  XX      XX   AA    AA   MM      MM  PP          LLLLLLLLLL  EEEEEEEEEE  SSSSSSSS
```

```
TTTTTTTTTT  DDDDDDDD    RRRRRRRR      IIIIII    VV      VV  EEEEEEEEEE  RRRRRRRR
TTTTTTTTTT  DDDDDDDD    RRRRRRRR      IIIIII    VV      VV  EEEEEEEEEE  RRRRRRRR
    TT      DD      DD  RR      RR      II      VV      VV  EE          RR      RR
    TT      DD      DD  RR      RR      II      VV      VV  EE          RR      RR
    TT      DD      DD  RR      RR      II      VV      VV  EE          RR      RR
    TT      DD      DD  RR      RR      II      VV      VV  EE          RR      RR
    TT      DD      DD  RRRRRRRR        II      VV      VV  EEEEEEEE    RRRRRRRR
    TT      DD      DD  RRRRRRRR        II      VV      VV  EEEEEEEE    RRRRRRRR
    TT      DD      DD  RR  RR          II      VV      VV  EE          RR  RR
    TT      DD      DD  RR  RR          II      VV      VV  EE          RR  RR
    TT      DD      DD  RR    RR        II        VV  VV    EE          RR    RR        ....
    TT      DD      DD  RR    RR        II        VV  VV    EE          RR    RR        ....
    TT      DDDDDDDD    RR      RR    IIIIII        VV      EEEEEEEEEE  RR      RR      ....
    TT      DDDDDDDD    RR      RR    IIIIII        VV      EEEEEEEEEE  RR      RR      ....

MM      MM    AAAAAA    RRRRRRRR
MM      MM    AAAAAA    RRRRRRRR
MMMM  MMMM  AA      AA  RR      RR
MMMM  MMMM  AA      AA  RR      RR
MM  MM  MM  AA      AA  RR      RR
MM  MM  MM  AA      AA  RR      RR
MM      MM  AA      AA  RRRRRRRR
MM      MM  AA      AA  RRRRRRRR
MM      MM  AAAAAAAAAA  RR  RR
MM      MM  AAAAAAAAAA  RR  RR
MM      MM  AA      AA  RR    RR
MM      MM  AA      AA  RR    RR
MM      MM  AA      AA  RR      RR
MM      MM  AA      AA  RR      RR
```

```
        .TITLE  TDRIVER - VAX/VMS TEMPLATE DRIVER
        .IDENT  'V04-000'
```



```
;++
; FACILITY:
;
;       VAX/VMS Template driver
;
; ABSTRACT:
;
;       This module contains the outline of a driver:
;
;               Models of driver tables
;               Controller and unit initialization routines
;               An FDT routine
;               The start I/O routine
;               The interrupt service routine
;               The cancel I/O routine
;               The device register dump routine
;
; AUTHOR:
;
;       S. Programmer   11-NOV-1979
;
; REVISION HISTORY:
;
;       V02     JHP001  J. Programmer   2-Aug-1979      11:27
;               Remove BLBC instruction from CANCEL routine.
;
;       V02-001 ROW0067 Ralph O. Weber  11-FEB-1981     13:10
```

```
;               Add description of reason argument to CANCEL routine.
;               Correct references to channel index number.
;
;--
```



```
;++
;  TI
;  Fu
;
;
;  In
;
;
;
;  Ou
;
;
;
;
;
;--
TD_F
```

```
        .SBTTL  External and local symbol definitions

;
; External symbols
;

        $CANDEF                         ; Cancel reason codes
        $CRBDEF                         ; Channel request block
        $DCDEF                          ; Device classes and types
        $DDBDEF                         ; Device data block
        $DEVDEF                         ; Device characteristics
        $IDBDEF                         ; Interrupt data block
        $IODEF                          ; I/O function codes
        $IPLDEF                         ; Hardware IPL definitions
        $IRPDEF                         ; I/O request packet
        $SSDEF                          ; System status codes
        $UCBDEF                         ; Unit control block
        $VECDEF                         ; Interrupt vector block

;
; Local symbols
;

;
; Argument list (AP) offsets for device-dependent QIO parameters
;

P1      = 0                             ; First QIO parameter
P2      = 4                             ; Second QIO parameter
P3      = 8                             ; Third QIO parameter
P4      = 12                            ; Fourth QIO parameter
P5      = 16                            ; Fifth QIO parameter
P6      = 20                            ; Sixth QIO parameter

;
; Other constants
;

TD_DEF_BUFSIZ   = 1024                  ; Default buffer size
TD_TIMEOUT_SEC  = 10                    ; 10 second device timeout
TD_NUM_REGS     = 4                     ; Device has 4 registers

;
; Definitions that follow the standard UCB fields
;

        $DEFINI UCB                     ; Start of UCB definitions

        .=UCB$K_LENGTH                  ; Position at end of UCB

$DEF    UCB$W_TD_WORD                   ; A sample word
                        .BLKW   1
$DEF    UCB$W_TD_STATUS                 ; Device's CSR register
                        .BLKW   1
$DEF    UCB$W_TD_WRDCNT                 ; Device's word count register
                        .BLKW   1
```

```
$DEF    UCB$W_TD_BUFADR                 ; Device's ouffer address
                        .BLKW   1       ; register
$DEF    UCB$W_TD_DATBUF                 ; Device's data buffer register
                        .BLKW   1
$DEF    UCB$K_TD_UCBLEN                 ; Length of extended UCB

;
; Bit positions for device-dependent status field in UCB
;

        $VIELD  UCB,0,<-                ; Device status
                <BIT_ZERO,,M>,-         ; First bit
                <BIT_ONE,,M>,-          ; Second bit
                >

        $DEFEND UCB                     ; End of UCB definitions

;
; Device register offsets from CSR address
;

        $DEFINI TD                      ; Start of status definitions

$DEF    TD_STATUS                       ; Control/status
                        .BLKW   1

;
; Bit positions for device control/status register
;

        _VIELD  TD_STS,0,<-             ; Control/status register
                <GO,,M>,-               ; Start device
                <BIT1,,M>,-             ; Bit one
                <BIT2,,M>,-             ; Bit two
                <BIT3,,M>,-             ; Bit three
                <XBA,2,M>,-             ; Extended address bits
                <INTEN,,M>,-            ; Enable interrupts
                <READY,,M>,-            ; Device ready for command
                <BIT8,,M>,-             ; Bit eight
                <BIT9,,M>,-             ; Bit nine
                <BIT10,,M>,-            ; Bit ten
                <BIT11,,M>,-            ; Bit eleven
                <,1>,-                  ; Disregarded bit
                <ATTN,,M>,-             ; Attention bit
                <NEX,,M>,-              ; Nonexistent memory flag
                <ERROR,,M>,-            ; Error or external interrupt
        >

$DEF    TD_WRDCNT                       ; Word count
                        .BLKW   1
$DEF    TD_BUFADR                       ; Buffer address
                        .BLKW   1
$DEF    TD_DATBUF                       ; Data buffer
                        .BLKW   1

        $DEFEND TD                      ; End of device register
```

; definitions.

```
        .SBTTL  Standard tables

;
; Driver prologue table
;

        DPTAB   -                               ; DPT-creation macro
                END=TD_END,-                    ; End of driver label
                ADAPTER=UBA,-                   ; Adapter type
                UCBSIZE=<UCB$K_TD_UCBLEN>,-     ; Length of UCB
                NAME=TDDRIVER                   ; Driver name
        DPT_STORE INIT                          ; Start of load
                                                ; initialization table
        DPT_STORE UCB,UCB$B_FIPL,B,8            ; Device fork IPL
        DPT_STORE UCB,UCB$B_DIPL,B,22           ; Device interrupt IPL
        DPT_STORE UCB,UCB$L_DEVCHAR,L,<-        ; Device characteristics
                DEV$M_IDV!-                     ;   input device
                DEV$M_ODV>                      ;   output device
        DPT_STORE UCB,UCB$B_DEVCLASS,B,DC$_SCOM ; Sample device class
        DPT_STORE UCB,UCB$W_DEVBUFSIZ,W,-       ; Default buffer size
                TD_DEF_BUFSIZ

        DPT_STORE REINIT                        ; Start of reload
                                                ; initialization table
        DPT_STORE DDB,DDB$L_DDT,D,TD$DDT        ; Address of DDT
        DPT_STORE CRB,CRB$L_INTD+4,D,-          ; Address of interrupt
                TD_INTERRUPT                    ; service routine
        DPT_STORE CRB,-                         ; Address of controller
                CRB$L_INTD+VEC$L_INITIAL,-      ; initialization routine
                D,TD_CONTROL_INIT
        DPT_STORE CRB,-                         ; Address of device
                CRB$L_INTD+VEC$L_UNITINIT,-     ; unit initialization
                D,TD_UNIT_INIT                  ; routine

        DPT_STORE END                           ; End of initialization
                                                ; tables

;
; Driver dispatch table
;

        DDTAB   -                               ; DDT-creation macro
                DEVNAM=TD,-                     ; Name of device
                START=TD_START,-                ; Start I/O routine
                FUNCTB=TD_FUNCTABLE,-           ; FDT address
                CANCEL=TD_CANCEL,-              ; Cancel I/O routine
                REGDMP=TD_REG_DUMP              ; Register dump routine

;
; Function decision table
;

TD_FUNCTABLE:                                   ; FDT for driver
        FUNCTAB ,-                              ; Valid I/O functions
                <READVBLK,-                     ; Read virtual
                READLBLK,-                      ; Read logical
```

```
                READPBLK,-                      ; Read physical
                WRITEVBLK,-                     ; Write virtual
                WRITELBLK,-                     ; Write logical
                WRITEPBLK,-                     ; Write physical
                SETMODE,-                       ; Set device mode
                SETCHAR>                        ; Set device chars.
        FUNCTAB ,                               ; No buffered functions
        FUNCTAB +EXE$READ,-                     ; FDT read routine for
                <READVBLK,-                     ; read virtual,
                READLBLK,-                      ; read logical,
                READPBLK>                       ; and read physical.
        FUNCTAB +EXE$WRITE,-                    ; FDT write routine for
                <WRITEVBLK,-                    ; write virtual,
                WRITELBLK,-                     ; write logical,
                WRITEPBLK>                      ; and write physical.
        FUNCTAB +EXE$SETMODE,-                  ; FDT set mode routine
                <SETCHAR,-                      ; for set chars. and
                SETMODE>                        ; set mode.
```

```
        .SBTTL  TD_CONTROL_INIT, Controller initialization routine

;++
; TD_CONTROL_INIT, Readies controller for I/O operations
;
; Functional description:
;
;       The operating system calls this routine in 3 places:
;
;               at system startup
;               during driver loading and reloading
;               during recovery from a power failure
;
; Inputs:
;
;       R4      - address of the CSR (controller status register)
;       R5      - address of the IDB (interrupt data block)
;       R6      - address of the DDB (device data block)
;       R8      - address of the CRB (channel request block)
;
; Outputs:
;
;       The routine must preserve all registers except R0-R3.
;--

TD_CONTROL_INIT:                        ; Initialize controller
        RSB                             ; Return
```

```
        .SBTTL  TD_UNIT_INIT, Unit initialization routine

;++
; TD_UNIT_INIT, Readies unit for I/O operations
;
; Functional description:
;
;       The operating system calls this routine after calling the
;       controller initialization routine:
;
;               at system startup
;               during driver loading
;               during recovery from a power failure
;
; Inputs:
;
;       R4      - address of the CSR (controller status register)
;       R5      - address of the UCB (unit control block)
;
; Outputs:
;
;       The routine must preserve all registers except R0-R3.
;--

TD_UNIT_INIT:                           ; Initialize unit
        BISW    #UCB$M_ONLINE, -
                UCB$W_STS(R5)           ; Set unit online
        RSB                             ; Return
```

```
        .SBTTL  TD_FDT_ROUTINE, Sample FDT routine

;++
; TD_FDT_ROUTINE, Sample FDT routine
;
; Functional description:
;
;       T.B.S.
;
; Inputs:
;
;       R0-R2   - scratch registers
;       R3      - address of the IRP (I/O request packet)
;       R4      - address of the PCB (process control block)
;       R5      - address of the UCB (unit control block)
;       R6      - address of the CCB (channel control block)
;       R7      - bit number of the I/O function code
;       R8      - address of the FDT table entry for this routine
;       R9-R11  - scratch registers
;       AP      - address of th  1st function dependent QIO parameter
;
; Outputs:
;
;       The routine must preserve all registers except R0-R2, and
;       R9-R11.
;
;--

TD_FDT_ROUTINE:                         ; Sample FDT routine
        RSB                             ; Return
```

```
        .SBTTL  TD_START, Start I/O routine

;++
; TD_START - Start a transmit, receive, or set mode operation
;
; Functional description:
;
;       T.B  .
;
; Inputs:
;
;       R3      - address of the IRP (I/O request packet)
;       R5      - address of the UCB (unit control block)
;
; Outputs:
;
;       R0      - 1st longword of I/O status: contains status code and
;                 number of bytes transferred
;       R1      - 2nd longword of I/O status: device-dependent
;
;       The routine must preserve all registers except R0-R2 and R4.
;
;--

TD_START:                               ; Process an I/O packet

        WFIKPCH TD_TIMEOUT,#TD_TIMEOUT_SEC
;
; After a transfer completes successfully, return the number of bytes
; transferred and a success status code.
;
        IOFORK
        INSV    UCB$W_BCNT(R5),#16,-    ; Load number of bytes trans-
                #16,R0                  ; ferred into high word of R0.
        MOVW    #SS$_NORMAL,R0          ; Load a success code into R0.

;
; Call I/O postprocessing.
;

COMPLETE_IO:                            ; Driver processing is finished.
        REQCOM                          ; Complete I/O.

;
; Device timeout handling. Return an error status code.
;

TD_TIMEOUT:                             ; Timeout handling
        SETIPL  UCB$B_FIPL(R5)          ; Lower to driver fork IPL
        MOVZWL  #SS$_TIMEOUT,R0         ; Return error status.
        BRB     COMPLETE_IO             ; Call I/O postprocessing.
```

```
        .SBTTL  TD_INTERRUPT, Interrupt service routine

;++
; TD_INTERRUPT, Analyzes interrupts, processes solicited interrupts
;
; Functional description:
;
;       The sample code assumes either
;
;               that the driver is for a single-unit controller, and
;               that the unit initialization code has stored the
;               address of the UCB in the IDB; or
;
;               that the driver's start I/O routine acquired the
;               controller's channel with a REQPCHANL macro call, and
;               then invoked the WFIKPCH macro to keep the channel
;               while waiting for an interrupt.
;
; Inputs:
;
;        0(SP)  - pointer to the address of the IDB (interrupt data
;                 block)
;        4(SP)  - saved R0
;        8(SP)  - saved R1
;       12(SP)  - saved R2
;       16(SP)  - saved R3
;       20(SP)  - saved R4
;       24(SP)  - saved R5
;       28(SP)  - saved PSL (program status longword)
;       32(SP)  - saved PC
;
;       The IDB contains the CSR address and the UCB address.
;
; Outputs:
;
;       The routine must preserve all registers except R0-R5.
;--

TD_INTERRUPT:                           ; Service device interrupt
        MOVL    @(SP)+,R4               ; Get address of IDB and remove
                                        ; pointer from stack.
        MOVL    IDB$L_OWNER(R4),R5      ; Get address of device owner's
                                        ; UCB.
        MOVL    IDB$L_CSR(R4),R4        ; Get address of device's CSR.
        BBCC    #UCB$V_INT,-            ; If device does not expect
                UCB$W_STS(R5),-         ; interrupt, dismiss it.
                UNSOL_INTERRUPT
;
; This is a solicited interrupt. Save
; the contents of the device registers in the UCB.
;
        MOVW    TD_STATUS(R4),-         ; Otherwise, save all device
                UCB$W_TD_STATUS(R5)     ; registers. First the CSR.
```

```
        MOVW    TD_WRDCNT(R4),-         ; Save the word count register.
                UCB$W_TD_WRDCNT(R5)
        MOVW    TD_BUFADR(R4),-         ; Save the buffer address
                UCB$W_TD_BUFADR(R5)     ; register.
        MOVW    TD_DATBUF(R4),-         ; Save the data buffer register.
                UCB$W_TD_DATBUF(R5)

;
; Restore control to the main driver.
;

RESTORE_DRIVER:                         ; Jump to main driver code.
        MOVL    UCB$L_FR3(R5),R3        ; Restore driver's R3 (use a
                                        ; MOVQ to restore R3-R4).
        JSB     @UCB$L_FPC(R5)          ; Call driver at interrupt
                                        ; wait address.

;
; Dismiss the interrupt.
;

UNSOL_INTERRUPT:                        ; Dismiss unsolicited interrupt.
        POPR    #^M<R0,R1,R2,R3,R4,R5>  ; Restore R0-R5
        REI                             ; Return from interrupt.
```

```
        .SBTTL  TD_CANCEL, Cancel I/O routine

;++
; TD_CANCEL, Cancels an I/O operation in progress
;
; Functional description:
;
;       This routine calls IOC$CANCELIO to set the cancel bit in the
;       UCB status word if:
;
;               the device is busy,
;               the IRP's process ID matches the cancel process ID,
;               the IRP channel matches the cancel channel.
;
;       If IOC$CANCELIO sets the cancel bit, then this driver routine
;       does device-dependent cancel I/O fixups.
;
; Inputs:
;
;       R2      - channel index number
;       R3      - address of the current IRP (I/O request packet)
;       R4      - address of the PCB (process control block) for the
;                 process canceling I/O
;       R5      - address of the UCB (unit control block)
;       R8      - cancel reason code, one of:
;                       CAN$C_CANCEL    if called through $CANCEL or
;                                       $DALLOC system service
;                       CAN$C_DASSGN    if called through $DASSGN system
;                                       service
;                 These reason codes are defined by the $CANDEF macro.
;
; Outputs:
;
;       The routine must preserve all registers except R0-R3.
;
;       The routine may set the UCB$M_CANCEL bit in UCB$W_STS.
;
;--

TD_CANCEL:                              ; Cancel an I/O operation
        JSB     G^IOC$CANCELIO          ; Set cancel bit if appropriate.
        BBC     #UCB$V_CANCEL,-         ; If the cancel bit is not set,
                UCB$W_STS(R5),10$       ; just return.

;
; Device-dependent cancel operations go next.
;

;
; Finally, the return.
;

10$:
        RSB                             ; Return
```

```
        .SBTTL  TD_REG_DUMP, Device register dump routine

;++
; TD_REG_DUMP, Dumps the contents of device registers to a buffer
;
; Functional description:
;
;       Writes the number of device registers, and their current
;       contents into a diagnostic or error buffer.
;
; Inputs:
;
;       R0      - address of the output buffer
;       R4      - address of the CSR (controller status register)
;       R5      - address of the UCB (unit control block)
;
; Outputs:
;
;       The routine must preserve all registers except R1-R3.
;
;       The output buffer contains the current contents of the device
;       registers. R0 contains the address of the next empty longword in
;       the output buffer.
;--

TD_REG_DUMP:                            ; Dump device registers
        MOVZBL  #TD_NUM_REGS,(R0)+      ; Store device register count.
        MOVZWL  UCB$W_TD_STATUS(R5),-   ; Store device status register.
                (R0)+
        MOVZWL  UCB$W_TD_WRDCNT(R5),-   ; Store word count register.
                (R0)+
        MOVZWL  UCB$W_TD_BUFADR(R5),-   ; Store buffer address register.
                (R0)+
        MOVZWL  UCB$W_TD_DATBUF(R5),-   ; Store data buffer register.
                (R0)+
        RSB                             ; Return
```

```
        .SBTTL  TD_END, End of driver

;++
; Label that marks the end of the driver
;--

TD_END:                                 ; Last location in driver
        .END
```

LPMULT
B32

DRMAST
MAR

ADDRIVER
MAR

TDRIVER
MAR

USSTEST
MAR

GB_SECUFO
MAR

USSDISP
MAR

DOD_ERAPAT
MAR

LBRMAC
MAR

XADRIVER
MAR

LABIOCIN
MAR

DRSLV
MAR

DTE_DF03
MAR

WORKQ
LIS

SCRFT
MAR

EXAMPLES