A

Sample Driver for a TURBOchannel Device

The following sample driver provides a TURBOchannel driver template to begin coding your specific driver program. Examples of programmed I/O, interrupt processing, and DMA reads and writes are provided. Table A–1 outlines the driver code by listing the sections and routines in order of their occurrence. The callouts in the driver code example refer to entries in Table A–1.

Table A–1 TURBOchannel Test Board Driver Code Contents

Driver Code Points	Function
1 External symbols	Defined
2 Local symbols	Defined
3 Argument list (AP)	Defined for device-dependent QIO parameters
4 Constants	Defined
5 Device specific UCB fields	Defined
6 Device register offsets from CSR	Defined
7 Bit positions of CSR	Defined
8 Driver prologue table (DPT)	Initialized with DPT_STORE
9 Driver dispatch table (DDT)	Initialized with DDTAB
10 Function decision table (FDT)	Loaded with FUNCTAB
11 CB_CONTROL_INIT routine	For controller initialization
12 CB_FDT_PIO FDT routine	For programmed I/O servicing WRITEPBLK
13 CB_FDT_DMA FDT routine	For data transfers servicing READLBLK and WRITELBLK
14 CB_FDT_INTR FDT routine	For interrupts servicing INTERRUPT
15 CB_START routine	Starting an I/O transfer
16 CB_TIME_OUT routine	Handling a device time-out
17 CB_INTERRUPT routine	Handling interrupts generated by the device
18 CB_CANCEL routine	Canceling an I/O operation

```
        .TITLE  CBDRIVER - VAX/VMS TURBOchannel Channel Board DRIVER
        .IDENT  'X-01'

;
;*****************************************************************************
;*                                                                           *
;*  COPYRIGHT (c) 1992 BY                                                    *
;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
;*  ALL RIGHTS RESERVED.                                                     *
;*                                                                           *
;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE    *
;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
;*  TRANSFERRED.                                                             *
;*                                                                           *
;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT    *
;*  CORPORATION.                                                             *
;*                                                                           *
;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
;*                                                                           *
;*                                                                           *
;*****************************************************************************
;
;++
;
; FACILITY:
;
;       VAX/VMS Executive, I/O Drivers
;
; ABSTRACT:
;
;       Example TURBOchannel Device Driver.
;
;       The TURBOchannel test board is a minimal implementation of
;       all TURBOchannel hardware functions: Programmed I/O, I/O
;       interrupt, DMA read, DMA write and I/O read/write conflict
;       testing.
;       The software view of the board consists of an EPROM address space,
;       a 32-bit ADDRESS register with bits scrambled for direct use as a TC
;       DMA address, a 32-bit DATA register used for programmed I/O and as the
;       holding register for DMA, a 16-bit LED/CSR and a 1-bit TEST register.
;       All registers MUST be accessed as 32-bit longwords, even when they are
;       not implemented as 32 bits.  The CSR contains bits to enable option
;       DMA read testing, conflict signal testing, I/O interrupt testing and
;       option DMA write testing.  It also contains a bit to indicate that
;       one or more of the tests are enabled, 4 byte mask flag bits and a
;       DMA done bit.
;
;       This example VAX/VMS driver provides a simple interface to the
;       TURBOchannel test board. (a) Programmed I/O access to test board.
;       (b) DMA to/from test board. (c) I/O interrupt fro test board.
;
; ENVIRONMENT:
;
;       Kernel Mode, Non-paged
;
;
;
;--

1
        .SBTTL  External and local symbol definitions
```

```
; External symbols

        $ACBDEF                         ; AST control block
        $ADPDEF                         ; Adapter control block
        $CRBDEF                         ; Channel request block
        $DCDEF                          ; Device types
        $DDBDEF                         ; Device data block
        $DEVDEF                         ; Device characteristics
        $DPTDEF                         ; Driver prolog table
        $DYNDEF                         ; Dynamic data structure types
        $EMBDEF                         ; EMB offsets
        $IDBDEF                         ; Interrupt data block
        $IODEF                          ; I/O function codes
        $IPLDEF                         ; Hardware IPL definitions
        $IRPDEF                         ; I/O request packet
        $PRDEF                          ; Intervnal processor registers
        $PRIDEF                         ; Scheduler priority increments
        $SSDEF                          ; System status codes
        $UCBDEF                         ; Unit control block
        $VECDEF                         ; Interrupt vector block
        $IO46DEF                        ; Define pvmariah I/O space.
        $IRPDEF                         ; I/O request packet
        $VADEF                          ; VA
        $PTEDEF                         ; PTE
        $PTADEF                         ; TCA

2
; Local symbols
; (your local symbols here)
;

3
; 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

4
; Constants

CB_DMA_DEF_TIMEOUT = 10                 ; 10 second DMA default timeout

5
; CB Board definitions that follow the standard UCB fields
```

```
        $DEFINI UCB
        .=UCB$L_DPC+4
$DEF    UCB$L_MAPREG_DESC               ; The Mapping Register Descriptor.
$DEF    UCB$W_START_MAPREG              ; The Starting Map Register.
                .BLKW   1
$DEF    UCB$W_NUMBER_MAPREG             ; The number of Map Registers.
                .BLKW   1
$DEF    UCB$W_CB_UNEXPECTED             ; Counter for # of unexpected interrupts.
                .BLKW   1
$DEF    UCB$L_CB_CSRTMP                 ; Temporary storage of Control Reg image
                .BLKL   1
$DEF    UCB$L_CB_ADDRTMP                ; Temporary storage of Address Reg
                .BLKL   1
$DEF    UCB$L_CB_DATATMP                ; Temporary storage of Data Reg
                .BLKL   1
$DEF    UCB$L_CB_TESTTMP                ; Temporary storage of Test Reg
                .BLKL   1
$DEF    UCB$L_CB_VA                     ; Storage for Device VA.
                .BLKL   1
$DEF    UCB$L_CB_CSR                    ; Saved STATUS Reg on interrupt
                .BLKL   1
$DEF    UCB$L_CB_ADDR                   ; Saved Address Reg on interrupt
                .BLKL   1
$DEF    UCB$L_CB_DATA                   ; Saved Data Reg on interrupt
                .BLKL   1
$DEF    UCB$L_CB_TEST                   ; Saved Test Reg on interrupt
                .BLKL   1
$DEF    UCB$L_CB_BCNT                   ; Saved Byte Count
                .BLKL   1

UCB$K_SIZE=.
        $DEFEND UCB
```

6

```
; Device register offsets from CSR address

        $DEFINI CB                      ; Start of Channel Board definitions
$DEF    CB_ADDRESS                      ; Address Register
                .BLKL   1               ;
$DEF    CB_DATA                         ; Data Register
                .BLKL   1               ;
$DEF    CB_LED_CSR                      ; LED/CSR Register
                .BLKL   1               ;
$DEF    CB_TEST                         ; TEST Register
                .BLKL   1               ;
```

7

```
; Bit positions for device LED/control/status register

        $VIELD  CB_CSR,0,<-               ; Control register
                <MASK,4,M>,-              ; byte-mask
                <DMADONE,,M>,-            ; 1 if DMA complete
                <ENTEST,,M>,-             ; 1 if int/conflict/DMA tests enabled
                <INT,,M>,-                ; 0 if interrupt asserted
                <TDONE,,M>,-              ; 1 if the enabled test is done
                <ENINT,,M>,-              ; 0 if interrupt enable
                <ENCONF,,M>,-             ; 0 if conflict enable
                <DMAR,,M>,-               ; 0 if DMA Read enable
                <DMAW,,M>,-               ; 0 if DMA Write enable
                <UNUSED,16,M>,-           ; Unused bits

        >

        $DEFEND CB                         ; End of CB definition
```

8

```
        .SBTTL  Device Driver Tables
```

```
; Driver prologue table

        DPTAB   -                               ; DPT-creation macro
                END=CB_END,-                    ; End of driver label
                ADAPTER=TC,-                    ; Adapter type
                FLAGS=DPT$M_SVP,-               ; Allocate system page table
                UCBSIZE=UCB$K_SIZE,-            ; UCB size
                NAME=CBDRIVER                   ; Driver name
        DPT_STORE INIT                          ; Start of load
                                                ; initialization table
        DPT_STORE UCB,UCB$B_FLCK,B,SPL$C_IOLOCK8 ; Device fork IPL
        DPT_STORE UCB,UCB$B_DIPL,B,20           ; Device interrupt IPL
        DPT_STORE UCB,UCB$L_DEVCHAR,L,<-        ; Device characteristics
                DEV$M_AVL!-                     ; Available
                DEV$M_RTM>                      ; Real Time device
        DPT_STORE UCB,UCB$B_DEVCLASS,B,DC$_REALTIME     ; Device class
        DPT_STORE UCB,UCB$B_DEVTYPE,B,DT$_XVIB  ; Device Type
        DPT_STORE REINIT                        ; Start of reload
                                                ; initialization table
        DPT_STORE DDB,DDB$L_DDT,D,CB$DDT        ; Address of DDT
        DPT_STORE CRB,CRB$L_INTD+4,D,-          ; Address of interrupt
                CB_INTERRUPT                    ; service routine
        DPT_STORE CRB,CRB$L_INTD+VEC$L_INITIAL,-; Address of controller
                D,CB_CONTROL_INIT               ; initialization routine
        DPT_STORE END                           ; End of initialization
                                                ; tables
9
; Driver dispatch table

        DDTAB   -                               ; DDT-creation macro
                DEVNAM=CB,-                     ; Name of device
                START=CB_START,-                ; Start I/O routine
                FUNCTB=CB_FUNCTABLE,-           ; FDT address
                CANCEL=CB_CANCEL                ; Cancel I/O routine

10
;
; Function dispatch table
;
CB_FUNCTABLE:                                   ; FDT for driver
        FUNCTAB ,-                              ; Valid I/O functions
                <READPBLK,READLBLK,READVBLK,WRITEPBLK,WRITELBLK,WRITEVBLK>
        FUNCTAB ,                               ; No buffered functions
        FUNCTAB CB_FDT_PIO,<WRITEPBLK>          ; PIO FDT
        FUNCTAB CB_FDT_DMA,<READLBLK,WRITELBLK> ; DMA FDT
        FUNCTAB CB_FDT_INTR,<READPBLK>          ; INTR FDT


11
        .SBTTL  CB_CONTROL_INIT, Controller initialization
```

```
;++
; CB_CONTROL_INIT, Called when driver is loaded, system is booted
;
; Functional Description:
;
;       1) Allocates the direct data path permanently
;       2) Assigns the controller data channel permanently
;       3) Clears the Control and Status Register
;       4) Map a page of device physical address space
;          to virtual address space
;
; Inputs:
;
;       R4 = base address of TC slot
;       R5 = address of IDB
;       R6 = address of DDB
;       R8 = address of CRB
;
; Outputs:
;
;       None
;--
CB_CONTROL_INIT:
        MOVL    #1, R2                  ; 1 page
        ADDL    #^X20000, R4            ; Base Physical Address for
                                        ; Device CSRs
;
;       Map phyical address space to virtual address space
;
        JSB     G^LDR$ALLOC_PT          ; Request the pages
                                        ; R1 = Virtual Address of
                                        ; 1st SPTE
        SUBL3   G^MMG$GL_SPTBASE,R1,R3  ; Byte offset of first SPTE
        ASHL    #7,R3,R3                ; Get a system VA
        BISL    #<1@31>,R3              ; R3 = Virtual Address represented
                                        ; by 1st SPTE

        ASHL    #-9, R4, R4             ; Get PFN

        MOVL    R3, R0                  ; Place VA in R0

; Set up each page

10$:    INVALIDATE_TB   R0,-            ; Clear TB of temporary mapping
                INST1=<BISL3 #<PTE$M_VALID!PTE$C_KW>,R4,(R1)+> ; Map a page
        INCL    R4                      ; Next PFN.
        ADDL2   #512,R0                 ; Advance to next page
        SOBGTR  R2,10$                  ; Map another page

        MOVL    IDB$L_UCBLST(R5),R0     ; Address of UCB
        MOVL    R3, UCB$L_CB_VA(R0)     ; Device VA address
        MOVL    R0,IDB$L_OWNER(R5)      ; Make permanent controller owner
        BISW    #UCB$M_ONLINE,UCB$W_STS(R0)
                                        ; Set device status "on-line"

        CLRW    UCB$W_CB_UNEXPECTED(R0) ; Init Unexpected Interrupt counter

                                        ; Enable device interrupts
        MOVL    #^XFF00, UCB$L_CB_CSRTMP(R0)
                                        ; Initialize device CSR
        RSB                             ; Done
```

```
12
   ;++
   ; CB_FDT_PIO                    FDT for WRITEPBLK
   ;
   ; Functional description:
   ;
   ; Inputs:
   ;
   ;
   ;               R3 = Address of IRP
   ;               R4 = Address of PCB
   ;               R5 = Address of UCB
   ;               R6 = Address of CCB
   ;               R8 = Address of FDT routine
   ;               AP = Address of P1
   ;                       P1 = TC address
   ;
   ; Outputs:
   ;
   ;       Queued to start IO routine
   ;--
   CB_FDT_PIO:
           MOVL    P1(AP),IRP$L_IOST1(R3)          ; Get the TC address.
           JMP     G^EXE$QIODRVPKT

13
   ;++
   ; CB_FDT_DMA            FDT for READLBLK,WRITELBLK
   ;
   ; Functional description:
   ;
   ;       1) Rejects QUEUE I/O's with transfer count greater than 4
   ;
   ; Inputs:
   ;
   ;       R3 = Address of IRP
   ;       R4 = Address of PCB
   ;       R5 = Address of UCB
   ;       R6 = Address of CCB
   ;       R8 = Address of FDT routine
   ;       AP = Address of P1
   ;               P1 = Buffer Address
   ;               P2 = Buffer size in bytes
   ;               P3 = DMA Time Out Time in seconds
   ;
   ; Outputs:
   ;
   ;       R0 = Error status if odd transfer count
   ;
   ;--
   CB_FDT_DMA:
           BLBS    P1(AP),2$               ; The Buffer address must not be on
                                           ; a byte boundary.
           BBS     #1,P1(AP),2$            ; or a word boundary

           CMPL    P2(AP),#4               ; Branch if transfer less than 4
           BLEQU   20$

   2$:     MOVZWL  #SS$_BADPARAM,R0        ; Set error status code
   5$:     JMP     G^EXE$ABORTIO           ; Abort request

   20$:    TSTL    P2(AP)                  ; Error if no transfer count.
           BEQL    2$
```

```
                MOVL    P3(AP),IRP$L_MEDIA(R3)  ; Save the Time Out time.
                BNEQ    30$                     ; Branch if there is a time out time
                MOVL    #CB_DMA_DEF_TIMEOUT,-   ; Set Time Out time to the default
                        IRP$L_MEDIA(R3)

        30$:
                MOVL    P1(AP),R0               ; Get the buffer address
                MOVL    P2(AP),R1               ; Get the byte count
                JSB     G^EXE$MODIFYLOCKR       ; Check buffer for access and lock down
                BLBC    R0,5$
                JMP     G^EXE$QIODRVPKT


14
        ;++
        ; CB_FDT_INTR                   FDT for INTERRUPT
        ;
        ; Functional description:
        ;
        ; Inputs:
        ;
        ;
        ;               R3 = Address of IRP
        ;               R4 = Address of PCB
        ;               R5 = Address of UCB
        ;               R6 = Address of CCB
        ;               R8 = Address of FDT routine
        ;               AP = Address of P1
        ;
        ; Outputs:
        ;
        ;       Queued to start IO routine
        ;--
        CB_FDT_INTR:
                JMP     G^EXE$QIODRVPKT


15
                .SBTTL  CB_START, Start I/O routines
        ;++
        ; CB_START - Start PIO or DMA or INTERRUPT
        ;
        ; Functional Description:
        ;
        ; Inputs:
        ;
        ;       R3 = Address of the I/O request packet (IRP)
        ;       R5 = Address of the UCB
        ;
        ; Outputs:
        ;
        ;
        ;--
                .ENABL  LSB

        CB_START:
        ;+
        ; Do final set up and dispatch to the routine which performs the function.
        ;-
                EXTZV   #IRP$V_FCODE,#IRP$S_FCODE,-   ; Extract I/O function code...
                        IRP$W_FUNC(R3),R1             ; ...without function modifiers
                CMPL    #IO$_WRITEPBLK,R1             ; Check for PIO function
                BEQL    40$                           ; Branch if yes
```

```
        CMPL    #IO$_WRITELBLK,R1            ; Check for DMA Read function
        BEQL    10$                          ; Branch if yes
        CMPL    #IO$_READLBLK,R1             ; Check for DMA Write function
        BEQL    20$                          ; Branch if yes
        CMPL    #IO$_READPBLK,R1             ; Check for Interrupt function
        BEQL    30$                          ; Branch if yes

                                             ; Illegal function code
        MOVZBL  #SS$_ILLIOFUNC,R0            ; Specify the error type
        REQCOM

10$:    BICL    #CB_CSR$M_DMAR, UCB$L_CB_CSRTMP(R5)
        BRW     50$                          ; DMA read
20$:    BICL    #CB_CSR$M_DMAW, UCB$L_CB_CSRTMP(R5)
        BRW     50$                          ; DMA write

30$:    BRW     100$                        ; Interrupt

40$:                                         ; PIO
        MOVL    IRP$L_IOST1(R3), R1          ; Get the TC address

        ASHL    #-9,R1,R1                    ; Determine PFN of this node.
        BICL    #^C<PTE$M_PFN>,R1            ; Mask only PFN
        MOVZBL  #SS$_NORMAL,R0               ; Set succus status

        REQCOM                               ; Return to user
                                             ; Finish request in exec

50$:                                         ; DMA
        MOVL    UCB$L_CRB(R5),R4             ; Address of CRB
        MOVL    @CRB$L_INTD+VEC$L_IDB(R4),R4 ; Get the CSR address.

        MOVAL   UCB$L_MAPREG_DESC(R5),R1    ; Set R1 to address of mapreg desc.
        MOVL    UCB$L_CRB(R5),R2            ; Get CRB address.
        MOVL    CRB$L_INTD+VEC$L_ADP(R2),R2 ; Get address of ADP

        MOVL    ADP$L_CSR(R2), R0         ; TCA CSR address
        BISL    #PTA$M_ENBMAP, (R0)       ; Enable mapping
        PUSHL   R3                        ; Save R3.
        MOVL    IRP$L_BCNT(R3),R0         ; Get the byte count.
        MOVL    R0,UCB$L_CB_BCNT(R5)      ; Store byte count in UCB
        MOVZWL  IRP$W_BOFF(R3),R3
        MOVAB   ^X3FF(R0)[R3],R3          ; Calculate highest relative byte
                                          ; and round
        ASHL    #-9,R3,R3                 ; Calculate number of map registers
                                          ; required

        BSBW    IOC$ALOTCMAP_DMAN         ; Allocate a set of VME map regs.
        POPL    R3                        ; Restore R3
        BLBS    R0,60$

        MOVZWL  #SS$_INSFMAPREG,R0        ; Set to error and end QIO.
        CLRL    R1
        JMP     110$

60$:
        MOVAL   UCB$L_MAPREG_DESC(R5),R1  ; Set the Mapreg desc address
        PUSHR   #^M<R3,R4,R5>             ; Save R3-R5.
        MOVL    IRP$L_BCNT(R3),R4         ; Set R4 to the byte count.
        MOVZWL  IRP$W_BOFF(R3),R5         ; Set R5 to the byte offset into
                                          ; 1st page.
        MOVL    IRP$L_SVAPTE(R3),R3       ; Set R3 to the SVAPTE of first page
        BSBW    IOC$LOADTCMAP_DMAN        ; Load the VME mapping registers
        POPR    #^M<R3,R4,R5>             ; Restore R3-R5.
```

```
;
; Set up CSR's
;
        MOVZWL  IRP$W_BOFF(R3),R1         ; Byte offset in first page of xfer
        INSV    UCB$W_START_MAPREG(R5),#9,#13,R1
                                          ; Insert the Starting Map Register
                                          ; number
        ASHL    #-2, R1, R1               ;
        CLRL    R0                        ; reset R0
        INSV    R1,#5,#27,R0              ; Set up address for TC bus bit<31:5>
                                          ; bit<4:0> <-- DMA addr<33:28>
        MOVL    R0, UCB$L_CB_ADDRTMP(R5)  ; Save TC Address

        MOVL    UCB$L_CB_VA(R5), R2       ; Get the device VA address

        MOVL    UCB$L_CB_ADDRTMP(R5), CB_ADDRESS(R2)
                                          ; Load address
        MOVL    UCB$L_CB_CSRTMP(R5), CB_LED_CSR(R2)
                                          ; Set up DMA
        MOVL    #0, CB_TEST(R2)           ; Set GO bit

        MOVL    #100, R0                        ;
70$:    BBS     #CB_CSR$V_DMADONE, CB_LED_CSR(R2),80$
                                          ; Is DMA complete
        SOBGTR  R0, 70$

        MOVZWL  #SS$_TIMEOUT, R0          ; Flag timeout error

        BRW     90$

; Handle request completion, release TC resources, check for errors

80$:
        MOVL    CB_LED_CSR(R2), UCB$L_CB_CSR(R5)
                                          ; Store CSR data

        MOVL    CB_TEST(R2), R0           ; Clear DMA

        MOVL    #^XFF00, UCB$L_CB_CSRTMP(R5)
                                          ; Clear CSR
        BRW     93$

90$:    MOVL    CB_LED_CSR(R2), UCB$L_CB_CSR(R5)
                                           ; Store CSR data
93$:    MOVAL   UCB$L_MAPREG_DESC(R5),R1   ; Get address of mapreg desc.
        MOVL    UCB$L_CRB(R5),R2           ; Get CRB address.
        MOVL    CRB$L_INTD+VEC$L_ADP(R2),R2  ; Get address of ADP
        BSBW    IOC$RELTCMAP_DMAN          ; Release the mapping registers

        BBC     #CB_CSR$V_DMADONE, UCB$L_CB_CSR(R5),95$
        MOVZWL  #SS$_NORMAL, R0           ; Set Success


95$:    MOVL     UCB$L_CB_CSR(R5), R1     ; Return CSR status in IOSB
        REQCOM                            ; Finish request in exec

100$:   BICL    #CB_CSR$M_ENINT, UCB$L_CB_CSRTMP(R5)


        DEVICELOCK -
                LOCKADDR=UCB$L_DLCK(R5),- ; Lock device access
                SAVIPL=-(SP),-            ; Save current IPL
                PRESERVE=NO               ; DON'T Preserve R0

        MOVL    UCB$L_CB_VA(R5), R2       ; Get the device VA address
        MOVL    UCB$L_CB_CSRTMP(R5), CB_LED_CSR(R2)
                                          ; Set up Interrupt
        MOVL    #0, CB_TEST(R2)           ; Set GO bit

        MOVL    UCB$L_CB_ADDRTMP(R5), CB_ADDRESS(R2)
                                          ; Load address reg to initiate
                                          ; interrupt
```

```
        WFIKPCH CB_TIME_OUT,IRP$L_MEDIA(R3)
                                        ; Wait for interrupt

                                        ; Device has interrupted, FORK
        IOFORK                          ; FORK to lower IPL

        MOVZWL  #SS$_NORMAL, R0         ; Success status in IOSB
        MOVL     UCB$L_CB_CSR(R5), R1   ; Return CSR status in IOSB
        MOVL    #^XFF00, UCB$L_CB_CSRTMP(R5)
                                        ; Initialize device csr
110$:   REQCOM                          ; Finish request in exec


16
        .SBTTL  CB_TIME_OUT, CB device Timeout Routine
;++
; CB_TIME_OUT - CB device Timeout Routine
;
; Functional Description:
;
; Inputs:
;
;
; Outputs:
;
;
;--
        .ENABL  LSB
CB_TIME_OUT:
        IOFORK                            ; Fork to complete request

        MOVAL   UCB$L_MAPREG_DESC(R5), R1
                                          ; Get address of mapreg desc.
        MOVL    UCB$L_CRB(R5), R2         ; Get CRB address
        MOVL    CRB$L_INTD+VEC$L_ADP(R2), R2
                                          ; Get address of ADP
        BSBW    IOC$RELTCMAP_DMAN         ; Release the mapping registers

        MOVZWL  #SS$_TIMEOUT, R0          ; Flag timeout error

        REQCOM                            ; Complete I/O in exec
```

```
17
        .SBTTL  CB_INTERRUPT, Interrupt service routine.
;++
; CB_TIME_OUT - CB device interrupt service routine
;
; Functional Description:
;
; Inputs:
;
;       00(SP) = Pointer to address of the device IDB
;       04(SP) = saved R0
;       08(SP) = saved R1
;       12(SP) = saved R2
;       16(SP) = saved R3
;       20(SP) = saved R4
;       24(SP) = saved R5
;       28(SP) = saved PC
;       32(SP) = saved PSL
;
; Outputs:
;
;       The driver is called at its Wait For Interrupt point if an
;       interrupt was expected.
;       The current value of the CB CSR's are stored in the UCB.
;
;--
CB_INTERRUPT:
        MOVL    @(SP)+,R4                   ; Address of IDB and pop SP
        MOVQ    (R4),R4                     ; CSR and UCB address from IDB

        DEVICELOCK -
                LOCKADDR=UCB$L_DLCK(R5),-   ; Lock device access
                CONDITION=NOSETIPL,-        ; already at DIPL
                PRESERVE=NO                 ; Don't preserve R0

; Check to see if device transfer request active or not
; If so, call driver back at Wait for Interrupt point and
; Clear unexpected interrupt flag.

        BBCC    #UCB$V_INT,UCB$W_STS(R5),24$
                                            ; If clear, no interrupt expected

; Read the CB device registers  and store into UCB.
        MOVL    CB_LED_CSR(R2), UCB$L_CB_CSR(R5)
                                          ; Store CSR data
        MOVL    CB_TEST(R2), R0           ; Clear interrupt
        MOVL    #^XFF00, CB_LED_CSR(R2)
                                          ; Clear CSR to clear interrupt

        MOVQ    UCB$L_FR3(R5),R3          ; Restore drivers R3
        JSB     @UCB$L_FPC(R5)            ; Call driver back
        BRB     25$

24$:                                      ; Read device registers
        INCW    UCB$W_CB_UNEXPECTED(R5)  ; Increment Unexpected Interrupt count

25$:    DEVICEUNLOCK -
                LOCKADDR=UCB$L_DLCK(R5),- ; Unlock device access
                PRESERVE=NO
        POPR    #^M<R0,R1,R2,R3,R4,R5>    ; Restore registers
        REI                               ; Return from interrupt
```

```
18
        .SBTTL  CB_CANCEL, Cancel I/O routine
;++
; CB_CANCEL, Cancels an I/O operation in progress
;
; Functional description:
;
;       Flushes  AST queue for the user.
;       Clear interrupt expected flag.
;
; Inputs:
;
;       R2 = negated value of channel index
;       R3 = address of current IRP
;       R4 = address of the PCB requesting the cancel
;       R5 = address of the device's UCB
;
; Outputs:
;
;--
CB_CANCEL:                                      ; Cancel I/O

        DEVICELOCK -
                LOCKADDR=UCB$L_DLCK(R5),- ; Lock device access
                SAVIPL=-(SP),-            ; Save current IPL
                PRESERVE=NO               ; Don't preserve R0

; Check to see if a data transfer request is in progress
; for this process on this channel

20$:    BBC     #UCB$V_INT,-              ; br if I/O not in progress
                UCB$W_STS(R5),30$
        JSB     G^IOC$CANCELIO            ; Check if transfer going
        BBC     #UCB$V_CANCEL,-
                UCB$W_STS(R5),30$         ; Branch if not for this guy
;
; Force timeout
;
        CLRL    UCB$L_DUETIM(R5)          ; clear timer
        BISW    #UCB$M_TIM,UCB$W_STS(R5)  ; set timed
        BICW    #UCB$M_TIMOUT,-
                UCB$W_STS(R5)             ; Clear timed out
30$:
        DEVICEUNLOCK -
                LOCKADDR=UCB$L_DLCK(R5),- ; Unlock device access
                NEWIPL=(SP)+,-            ; Enable interrupts
                PRESERVE=NO
        RSB                               ; Return

CB_END:                                   ; End of driver label
        .END
```