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

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

.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
; 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
$PLICDEF      ; Hardware IPL definitions
$IRPDEF       ; I/O request packet
$PRDEF        ; Internal processor registers
$PRIDEF       ; Scheduler priority increments
$SSDEF        ; System status codes
$UCBDEF       ; Unit control block
$VECDF        ; Interrupt vector block
$IO46DEF      ; Define pvmah I/O space.
$1RFDEF       ; 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
Sample Driver for a TURBOchannel Device

$DEFINE UCB
...=UCB$DPC+4
$DEF UCB$MAPREG_DESC ; The Mapping Register Descriptor.
$DEF UCB$START_MAPREG ; The Starting Map Register.
.BLK 1
$DEF UCB$NUMBER_MAPREG ; The number of Map Registers.
.BLK 1
$DEF UCB$CB_UNEXPECTED ; Counter for # of unexpected interrupts.
.BLK 1
$DEF UCB$CB_CSRTMP ; Temporary storage of Control Reg image.
.BLKL 1
$DEF UCB$CB_ADDRTMP ; Temporary storage of Address Reg.
.BLKL 1
$DEF UCB$CB_DATATMP ; Temporary storage of Data Reg.
.BLKL 1
$DEF UCB$CB_TESTTMP ; Temporary storage of Test Reg.
.BLKL 1
$DEF UCB$CB_VA ; Storage for Device VA.
.BLKL 1
$DEF UCB$CB_CSR ; Saved STATUS Reg on interrupt.
.BLKL 1
$DEF UCB$CB_ADDR ; Saved Address Reg on interrupt.
.BLKL 1
$DEF UCB$CB_DATA ; Saved Data Reg on interrupt.
.BLKL 1
$DEF UCB$CB_TEST ; Saved Test Reg on interrupt.
.BLKL 1
$DEF UCB$CB_BCNT ; Saved Byte Count.
.BLKL 1
UCB$K_SIZE=.
$DEFINE END UCB

6

; Device register offsets from CSR address

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

7

; Bit positions for device LED/control/status register

$DEFINE CB_CSR,0,<-
<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
>
$DEFINE CB ; End of CB definition

8

.SBTTL Device Driver Tables
; Driver prologue table
DPTAB -
    END=CB_END,-
    ADAPTER=TC,-
    FLAGS=DPTSM_SVP,-
    UCBSIZE=UCB$K_SIZE,-
    NAME=CBDRIVER
DPT_STORE INIT
    NAME=CBDRIVER

; Initialization table
DPT_STORE UCB, UCB$B_FLCK, B, SPL$C_IOLOCK8
DPT_STORE UCB, UCB$B_DIPL, B, 20
DPT_STORE UCB, UCB$L_DEVCHAR, L,<-
    DEVSM_AVL=<-
    DEVSM_RTM>
DPT_STORE UCB, UCB$B_DEVCLASS, B, DCS_REALTIME
DPT_STORE UCB, UCB$B_DEVTYPE, B, DT$XVIB
DPTSTORE REINIT
    START=CB_START,-
    FUNCTB=CB_FUNCTABLE,-
    CANCEL=CB_CANCEL
DPT_STORE END

; Driver dispatch table
DDTAB -
    DEVNAM=CB,-
    START=CB_START,-
    FUNCTB=CB_FUNCTABLE,-
    CANCEL=CB_CANCEL

; Function dispatch table

.SBTTL  CB_CONTROL_INIT, Controller initialization
Sample Driver for a TURBOchannel Device

; 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
ADDL #$X20000, R4

JSB G^LDR$ALLOC_PT

SUBL3 G^MMG$GL_SPTBASE,R1,R3
ASHL #7,R3,R3
BISL #<1@31>,R3

ASHL #-9, R4, R4
MOVL R3, R0

10$: INVALIDATE_TB R0,-

INST1=<BISL3 #<PTE$M_VALID!PTE$C_KW>,R4,(R1)+> ; Map a page
INCL R4
ADDL2 #512,R0
SOBTR R2,10$

MOVL IDBS$L_UCBLST(R5),R0
MOVL R3, UCBS$L_CB_VA(R0)
MOVL R0, IDBS$L_OWNER(R5)
BISW #UCBS$M_ONLINE,UCBS$W_STS(R0)

CLRW UCBSW_CB_UNEXPECTED(R0)

MOVL #$XFF00, UCBS$L_CB_CSRTMP(R0)

RSB; Done
+++; CB_FDT_PIO
; FDT for WRITEBLK
;
; 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:
   MOV L P1(AP),IRP$LOST1(R3) ; Get the TC address.
   JMP G'EXE$QIODRVPKT

+++; CB_FDT_DMA
; FDT for READBLK,WRITEBLK
;
; 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
   BBS #1,P1(AP),2$ ; a byte boundary.
   CMPL P2(AP),#4 ; or a word boundary
   BLEQU 20$ ; Branch if transfer less than 4
   MOVZWL #SS$BADPARAM,R0 ; Set error status code
   JMP G'EXE$ABORTIO ; Abort request

   TSTL P2(AP) ; Error if no transfer count.
   BEQL 2$
Sample Driver for a TURBOchannel Device

```
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$QIODRVPKT
14
; CB_FDT_INTR          FDT for INTERRUPT
; Functional description:
; Inputs:
; R3 = Address of IRP
; R4 = Address of PCB
; R5 = Address of UCB
; R6 = Address of CCB
; R8 = Address of FDT routine
; AP = Address of P1
; Outputs:
; Queued to start I/O routine
--
CB_FDT_INTR:
    JMP G$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 #IOS_WRITEPBLK,R1 ; Check for PIO function
    BEQL 40$ ; Branch if yes
```
Sample Driver for a TURBOchannel Device

```
CMPL #IOS_WRITEBLKLK,R1       ; Check for DMA Read function
BEQL 10$                        ; Branch if yes
CMPL #IOS_READBLKLK,R1          ; Check for DMA Write function
BEQL 20$                        ; Branch if yes
CMPL #IOS_READPBLKLK,R1         ; Check for Interrupt function
BEQL 30$                        ; Branch if yes

MOVZBL #SSS_IILLIOFUNC,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 #<PTE$M_PFN>,R1            ; Mask only PFN
MOVZBL #SSS_NORMAL,R0          ; Set success status
REQCOM ; Return to user
; Finish request in exec

50$: ; DMA
MOVL UCB$L_CRB(R5),R4          ; Address of CRB
MOVL @CRB$L_INT+VEC$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$L_INT+VEC$L_ADP(R2),R2 ; Get address of ADP
MOVL ADPS$CSR(R2), R0          ; TCA CSR address
BISL #PTASM$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
MOVAL IRP$W_BOFF(R3),R3         ; Calculate highest relative byte
MOVAL ^X3FF(R0)[R3],R3          ; and round
ASHL #9,R3,R3                   ; Calculate number of map registers
BSBW IOC$ALOTCMAP_DMAN          ; Allocate a set of VME map regs.
POP$ R3                         ; Restore R3
BLSB R0,60$                     ; Set to error and end QIO.

60$: MOVAL UCB$L_MAPREG_DESC(R5),R1       ; Set the Mapreg desc address
PUSH$R #M<R3,R4,R5>              ; Save R3-R5.
MOVL IRP$L_BCNT(R3),R4          ; Set R4 to the byte count.
MOVL 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$LOADTCLMAP_DMAN        ; Load the VME mapping registers
POP$R #M<R3,R4,R5>              ; Restore R3-R5.
```
Sample Driver for a TURBOchannel Device

; 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$L_CB_CSRTMP(R5), CB_ADDRESS(R2) ; Load address
MOVL #0, CB_TEST(R2) ; Set up DMA
MOVL #100, R0 ;
70$: 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$L_CRB(R5),R2 ; Get CRB address.
MOVL CB$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
Sample Driver for a TURBOchannel Device

WF1KPCH 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

RESULT CB_TIME_OUT, CB device Timeout Routine
;i+;
; 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
Sample Driver for a TURBOchannel Device

17

.SBTL  CB_INTERRUPT, Interrupt service routine.

; 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:

CB_INTERRUPT:

MOVL @(SP)+,R4 ; Address of IDB and pop SP
MOVQ (R4),R4 ; CSR and UCB address from IDB

DEVICELOCK -
LOCKADDR=UCBSL_DLCK(R5),- ; Lock device access
CONDITION=NOSETIPL,- ; already at DIPL
PREERVE=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$CB_CSR(R5) ; Store CSR data
MOVL  CB_TEST(R2), R0 ; Clear interrupt
MOVL  XFF00, CB_LED_CSR(R2) ; Clear CSR to clear interrupt
MOVQ  UCB$FR3(R5),R3 ; Restore drivers R3
JSB  @UCBSL_FPC(R5) ; Call driver back
BRB  25$

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

25$: DEVICEUNLOCK -
LOCKADDR=UCBSL_DLCK(R5),- ; Unlock device access
PRESERVE=NO
POPR #M<R0,R1,R2,R3,R4,R5> ; Restore registers
REI ; Return from interrupt
.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=UCBS_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$S_DUETIM(R5) ; clear timer
    BISW  #UCB$M_TIM,UCB$W_STS(R5) ; set timed
    BICW  #UCB$M_TIMOUT,-
      UCB$W_STS(R5) ; Clear timed out

30$:  DEVICEUNLOCK -
    LOCKADDR=UCBS_L_DLCK(R5),- ; Unlock device access
    NEWIPL=(SP)+,-                ; Enable interrupts
    PRESERVE=NO                  ; Return

RSB

CB_END:       ; End of driver label

.END