```
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   MMMM  MMMM  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
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
```

FILEID**LABIOCIN

```
LL              AAAAAA    BBBBBBBB     IIIIII      OOOOOO      CCCCCCCC    IIIIII    NN      NN
LL              AAAAAA    BBBBBBBB     IIIIII      OOOOOO      CCCCCCCC    IIIIII    NN      NN
LL            AA      AA  BB      BB     II      OO      OO  CC              II      NN      NN
LL            AA      AA  BB      BB     II      OO      OO  CC              II      NN      NN
LL            AA      AA  BB      BB     II      OO      OO  CC              II      NNNN    NN
LL            AA      AA  BB      BB     II      OO      OO  CC              II      NNNN    NN
LL            AA      AA  BBBBBBBB       II      OO      OO  CC              II      NN  NN  NN
LL            AA      AA  BBBBBBBB       II      OO      OO  CC              II      NN  NN  NN
LL            AAAAAAAAAA  BB      BB     II      OO      OO  CC              II      NN    NNNN
LL            AAAAAAAAAA  BB      BB     II      OO      OO  CC              II      NN    NNNN
LL            AA      AA  BB      BB     II      OO      OO  CC              II      NN      NN      ....
LL            AA      AA  BB      BB     II      OO      OO  CC              II      NN      NN      ....
LLLLLLLLLL    AA      AA  BBBBBBBB     IIIIII      OOOOOO      CCCCCCCC    IIIIII    NN      NN      ....
LLLLLLLLLL    AA      AA  BBBBBBBB     IIIIII      OOOOOO      CCCCCCCC    IIIIII    NN      NN      ....

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

```
;       KW_HIST = 1
        .TITLE  LABIO_CIN - LABIO Connect-to-Interrupt Module
        .IDENT  'V04-000'
```



```
;++
; FACILITY:
;
;       LABIO demonstation system
;
; ABSTRACT:
;
;       This module contains the I/O code for handling
;       an AD11-K. It is an example of a connect-to interrupt
;       routine. This module contains code to perform the following
;
;               The start I/O routine
;               The interrupt service routine
;               The cancel I/O routine
;
; AUTHOR:
;
;       P. Programmer   15-Nov-79
;--
```

```
        .SBTTL  DATA STRUCTURES

        .PSECT  LABIO_SECTION   PIC,OVR,REL,GBL,SHR,NOEXE,RD,WRT,LONG


; The following data structures are also defined by a
; FORTRAN INCLUDE file. These definitions must agree.


; AD_BLOCK      A/D Control Block

MAX_AD_CHANNEL = 16             ;Number of A/D channels
AD_BLOCK_SLOTS  = 16            ;number of entries in one block
AD_BLOCK_SIZE  = MAX_AD_CHANNEL*AD_BLOCK_SLOTS

;AD_BLOCK offsets (long words)

AD_STATUS       = 0             ;STATUS (Unknown, inactive, or active )
        ACTIVE_L= 2             ; ACTIVE
        INACTIVE_L = 1          ; INACTIVE
PID             = 4             ; PID of connected process
TICS_SAMPLE     = 8             ; Rate in tics/sample
BUFFER_SIZE     = 12            ; User specified buffer size
BUFFER_COUNT    = 16            ; User specified buffer count
BUFFER_ACQ      = 20            ; Number of buffers acquired
VALID_BUF_IND   = 24            ; Index of current valid data buffer
VALID_BUF_COUNT = 28            ; Number of data points in last buffer
CUR_BUF_IND     = 32            ; Index to current acq. buffer
CUR_BUF_COUNT   = 36            ; Number of data points in last buffer
TICS_REMAINING  = 40            ; Tics remaining to next sample
CUR_ACQ_OFF     = 44            ; Offset to acq point
AD_BLOCK_END    = 64            ; Offset to end of a block

AD_BLOCK:       .BLKL   AD_BLOCK_SIZE

; DATA_BUFFER   Data buffers for LABIO

MAX_BUF_COUNT   = 2             ;Number buffers/channel
MAX_BUF_SIZE    = 512           ;Maximum buffer size (WORDS)

BUFFER_END      = MAX_BUF_COUNT*MAX_BUF_SIZE*2 ; Size of one set of buffers

DATA_BUF_SIZE   = MAX_AD_CHANNEL*MAX_BUF_SIZE*MAX_BUF_COUNT
DATA_BUFFER:    .BLKW   DATA_BUF_SIZE

DATA_BUFFER_OFF = DATA_BUFFER-AD_BLOCK  ;Offset to data buffer from
                                        ;beginning of data structure

; CONNECT_BLOCK         Process Connect control block

MAX_PID = 16            ;Max number of processes connected

CONNECT_SIZE = MAX_PID*2

CONNECT_BLOCK:  .BLKL   CONNECT_SIZE
```

```
        .SBTTL  I/O DEVICES

;This section defines the constants asocciated with the KW11-K clock
;and the AD11-K A/D converter

;KW11-K Clock
;CSR bit assignments
KW11$M_GO = ^01                 ;GO bit
KW11$M_RATE =  ^02              ;Rate = bits 2-4
KW11$M_INTENB = ^0100           ;Interrupt enable
KW11$M_READY =  ^0200           ;Ready bit
KW11$M_REPINT = ^0400           ;repeated interuupts

KW11_CSR_CONS = KW11$M_REPINT!KW11$M_INTENB!<1*KW11$M_RATE>
                                ;Repeated interrupts,interrupt enable
                                ;Rate = 1 MHz
KW11_PRESET = 1000.             ;Preset => Interrupt rate of 1 KHz

KW11_A_BUFFER = ^02             ;Offset to clock A preset buffer
KW11_A_COUNTER = ^024           ;Offset to clock A counter

;AD11-K A/D converter

AD11_OFFSET     = -4            ; Offset to the AD11 from thr KW11 clock CSR.
AD11_BUF        = 2             ; AD11 buffer offset fron AD11 CSR

AD11_GO         = 1             ; Go bit
AD11_MUX_INCR   = ^0400         ; Mux incr bit
AD11_CSR_CONS   = AD11_GO       ; Initial CSR value

;Limit for stopping ISR loop
AD11_LOOP_LIMIT = AD11_MUX_INCR*<MAX_AD_CHANNEL-1>!AD11_CSR_CONS

        $IDBDEF                 ; Definition for I/O drivers
        $UCBDEF                 ; Data structurs
        $IODEF                  ; I/O function codes
        $CINDEF                 ; Connect-to-interrupt
        $CRBDEF                 ; CRB stuff
        $VECDEF                 ; more
```

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

;++
; LABIO_CIN_START - Starts the KW11-K
;
; Functional description:
;
;       This routine starts the KW11-K
;               Rate = 1 Khz
;               Repeated interrupt
;
; Inputs:
;
;       0(R2)  - arg count of 4
;       4(R2)  - Address of the process buffer
;       8(R2)  - Address of the IRP (I/O request packet)
;       12(R2) - Address of the device's CSR
;       16(R2) - Address of the UCB (Unit control block)
;
; Outputs:
;       none
;
;       The routine must preserve all registers except R0-R2 and R4.
;
;--
        .PSECT LABIO_CIN

LABIO_CIN_START::
        MOVL    12(R2),R0               ; Get address of the KW11 CSR
        CLRW    (R0)                    ; Clear the Clock

        MNEGW   #KW11_PRESET,-          ; Preset count buffer
                KW11_A_BUFFER(R0)
        MOVW    #KW1T_CSR_CONS+KW11$M_GO,(R0) ; Set the bits for
                                        ;   Repeated interrupt
                                        ;   Interrupt Enable
                                        ;   GO!

        MOVW    #SS$_NORMAL,R0          ; Load a success code into R0.
        RSB                             ; Return
```

```
        .SBTTL  LABIO_CIN_INTERRUPT, Interrupt service routine

;++
; LABIO_CIN_INTERRUPT
; Functional description:
;
;
; Inputs:
;
;       0(R2)  - arg count of 5
;       4(R2)  - Address of the process buffer
;       8(R2)  - Address of the AST parameter
;       12(R2) - Address of the device's CSR
;       16(R2) - Address of the IDB (interrupt dispatch block)
;       20(R2) - Address of the UCB (Unit control block)
;
;
; Outputs:
;       Sets those bits in the AST parameter for those
;       channels who had a buffer filled
;
;       The routine must preserve all registers except R0-R4
;--
CIN_BUF_ADD = 4                         ;Address of CIN buffer
AST_PARM = 8                            ;Offset to AST parmeter address
CIN_CSR_ADD = 12                        ;Address of CSR

LABIO_CIN_INT::
        PUSHR   #^M<R5,R6>              ;Service device interrupt, save R5,R6
        MOVL    CIN_CSR_ADD(R2),R4      ;Address of the KW11 CSR
        MOVL    CIN_BUF_ADD(R2),R5      ;Address of AD_BLOCK, control block
                                        ;for each A/D Channel
        MOVAL   DATA_BUFFER_OFF(R5),R1  ;Data Buffers
        MOVAL   AD11_OFFSET(R4),R4      ;Address of the AD11 CSR

        MOVW    #AD11_CSR_CONS,R6       ;AD11 CSR bits, GO bit on
        CLRL    @AST_PARM(R2)           ;Zero the AST parameter
        CLRL    R3
AD_LOOP:
        CMPL    (R5),S^#ACTIVE_L        ;Is this channel active?
        BLSS    AD_LOOP_NEXT            ;No, try next channel

        SOBGTR  TICS_REMAINING(R5),AD_LOOP_NEXT
                                        ;Decr the timer for this channel
                                        ;Br if no conversion required

        MOVW    R6,(R4)                 ;Start conversion, while that's going on
        .IF DF KW_HIST                  ;Time histogram, stored in data buffer
        MOVZWL  KW11_A_COUNTER-AD11_OFFSET(R4),R0 ;Get current clock contents
        ADDW    #KW1T_PRESET,R0         ;Calc time from intgerrupt
        INCW    (R1)[R0]                ;Add one to that time bin
        .ENDC
; While the A/D is converting, the tic counter for this channel,
; get the offset to the data pointer, and update it. Take appropriate
; action if we have buffer overflow.
```

```
        MOVL    TICS_SAMPLE(R5),-       ;Reset timer for this channel
                TICS_REMAINING(R5)
        MOVL    CUR_ACQ_OFF(R5),R0      ;Get index to next data point
        INCL    CUR_ACQ_OFF(R5)         ;Advance it
        AOBLSS  BUFFER_SIZE(R5),-       ;Update current data count
                CUR_BUF_COUNT(R5),-     ;Br if no buffer overflow
                AD_LOOP_DATA
;Buffer overflowed, reset data pointer, reset buffer pointer
;increment acquired buffer count, termimate channel I/O if done

        MOVL    CUR_BUF_IND(R5),-       ;Valid data buf available for user
                VALID_BUF_IND(R5)
        MOVL    CUR_BUF_COUNT(R5),-     ;Number of points in buffer
                VALID_BUF_COUNT(R5)
        MULL3   CUR_BUF_IND(R5),-       ;Offset to next data point
                #MAX_BUF_SIZE,-
                CUR_ACQ_OFF(R5)
        CLRL    CUR_BUF_COUNT(R5)       ;Reset data count
        AOBLEQ  #MAX_BUF_COUNT,-        ;Next buffer index
                CUR_BUF_IND(R5),1$
        MOVL    #1,CUR_BUF_IND(R5)      ;Wrap-around, reset buffer index
        CLRL    CUR_ACQ_OFF(R5)         ;And buffer offset
1$:     INSV    #1,R3,#T,@AST_PARM(R2)  ;Set bit in AST parameter word
        AOBLSS  BUFFER_COUNT(R5),-      ;Incr buffer count
                BUFFER_ACQ(R5),2$       ;Done with all buffers?
        TSTL    BUFFER_COUNT(R5)        ;If original count was zero
        BEQL    2$                      ;Don't stop
        MOVL    #INACTIVE_L,(R5)        ;Deactivate channel
2$:

; Now, get the data point and store it in the buffer.
;
AD_LOOP_DATA:
1$:     TSTB    (R4)                    ;Wait for A/D conversion
        BGEQ    1$                      ;
        .IF NDF KW_HIST                 ;Time histogram don't store actual data
        MOVW    AD11_BUF(R4),(R1)[R0]   ;store data point in buffer.
        .ENDC

;All done with this channel, setup for the next
;
AD_LOOP_NEXT:
        ADDL    #AD_BLOCK_END,R5        ;Next channel block
        ADDL    #BUFFER_END,R1          ;Next buffer
        ADDW    #AD11_MUX_INCR,R6       ;Incr A/D MUX
        AOBLSS  S^#MAX_AD_CHANNEL,R3,-  ;Next channel
                AD_LOOP                 ;Br if not done

;Exit routine - If any buffer overflowed, queue an AST
        MOVL    @AST_PARM(R2),R0        ;If any bit in the AST parameter
        BEQL    1$                      ;is set we must queue an AST
        MOVL    #1,R0                   ; 1 means queue the AST, 0 means don't
1$:     POPR    #^M<R5,R6>              ; Restore R5,R6
        RSB                             ;
```

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

;++
; LABIO_CIN_CANCEL, Cancels an I/O operation in progress
;
; Functional description:
;
;       This routine turns off the KW11-K
;
; Inputs:
;       R5      -  Addr of the UCB
;
; Outputs:
;
;       The routine must preserve all registers except R0-R3.
;
;--

LABIO_CIN_CNCL::
        MOVL    UCB$L_CRB(R5),R0                ; Get Address of the CRB
        MOVL    CRB$L_INTD+VEC$L_IDB(R0),R0     ;Address of the IDB
        MOVL    IDB$L_CSR(R0),R0                ; Get addr of KW11
        CLRW    (R0)                            ; Turn of the KW11
        MOVW    #SS$_NORMAL,R0                  ; And return
        RSB
```

```
        .SBTTL  LABIO_CIN_END, End of Module

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

LABIO_CIN_END:                          ; Last location in module
```

SCR

```
;--
out
```

809
909

```
.SBTTL  AD_CIN_SETUP    Set-up routine for LABIO connect-to-interrupt

;+
; This routine issues the QIO to connect to the AD11/KW11 interrupts.
; It takes care of the internals associated with the connect-to-interrupt
; QIO. Input parameters  the VMS channel and the AST service routine address.
; The connect-to-interrupt QIO condition code is returned.

        .PSECT  AD_CIN_SETUP

AD_CIN_SETUP::
        .WORD   0
        MOVL    8(AP),USER_AST          ;Get the user AST routine addr
AD_CIN_QIO:
        $QIO_S  CHAN=@4(AP),-           ;Channel
                FUNC=#IO$_CONINTWRITE,- ;Allow writing to the data buffer
                IOSB=AD_CIN_IOSB,-      ;I/O status Block
                P1=AD_CIN_BUF_DESC,-    ;Buffer descriptor
                P2=#AD_CIN_ENTRY,-      ;Entry list
                P3=#AD_CIN_MASK,-       ;Status bits,etc
                P4=#AD_CIN_AST,-        ;AST service routine
                P6=#10                  ;preallocate some AST control blocks
        RET                             ;Return to caller

AD_CIN_BUF_DESC:                        ;Buffer descriptor for CIN
        .LONG   LABIO_CIN_END-AD_BLOCK  ;Size of buffer and CIN handler
        .LONG   AD_BLOCK                ;Address of buffer

AD_CIN_ENTRY:
        .LONG   0                       ;No init code
        .LONG   LABIO_CIN_START-AD_BLOCK;Start code
        .LONG   LABIO_CIN_INT-AD_BLOCK  ;Interrupt service routine
        .LONG   LABIO_CIN_CNCL-AD_BLOCK ;I/O cancel routine

AD_CIN_IOSB:
        .LONG   0,0                     ; I/O Status Block

; Control mask

AD_CIN_MASK = CIN$M_REPEAT!CIN$M_START!CIN$M_ISR!CIN$M_CANCEL

;
; AD_CIN_AST

; This AST routine calls the user AST routine. The user routine
; can not be called directly because the AST parameter itself
; not its address is returned via the connect-to-interrupt routine.
; This routine simply calls the  user routine with the ADDRESS of
; the AST parameter.

AD_CIN_AST::
        .WORD   0
        PUSHAL  4(AP)                   ;Get the AST parameter addr
        CALLS   #1,@USER_AST            ;Call the USER routine
        RET
```

```
USER_AST:
        .LONG   0                       ;Addr of the user AST routine
        .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