```
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
```

```
  AAAAAA    DDDDDDDD    DDDDDDDD    RRRRRRRR      IIIIII    VV      VV  EEEEEEEEEE  RRRRRRRR
  AAAAAA    DDDDDDDD    DDDDDDDD    RRRRRRRR      IIIIII    VV      VV  EEEEEEEEEE  RRRRRRRR
AA      AA  DD      DD  DD      DD  RR      RR      II      VV      VV  EE          RR      RR
AA      AA  DD      DD  DD      DD  RR      RR      II      VV      VV  EE          RR      RR
AA      AA  DD      DD  DD      DD  RR      RR      II      VV      VV  EE          RR      RR
AA      AA  DD      DD  DD      DD  RR      RR      II      VV      VV  EE          RR      RR
AA      AA  DD      DD  DD      DD  RRRRRRRR        II      VV      VV  EEEEEEEE    RRRRRRRR
AA      AA  DD      DD  DD      DD  RRRRRRRR        II      VV      VV  EEEEEEEE    RRRRRRRR
AAAAAAAAAA  DD      DD  DD      DD  RR  RR          II      VV      VV  EE          RR  RR
AAAAAAAAAA  DD      DD  DD      DD  RR  RR          II      VV      VV  EE          RR  RR
AA      AA  DD      DD  DD      DD  RR    RR        II        VV  VV    EE          RR    RR        ....
AA      AA  DD      DD  DD      DD  RR    RR        II        VV  VV    EE          RR    RR        ....
AA      AA  DDDDDDDD    DDDDDDDD    RR      RR    IIIIII        VV      EEEEEEEEEE  RR      RR      ....
AA      AA  DDDDDDDD    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
```

```
ADDR

: CO
: US
:

AD_B

: CS
:
AD_C

: DE
: PC
:
AD_T

: PC
: LC
: DL
:
AD_F
10$

: NC
:
AD_(
```

```
        .TITLE  ADDRIVER - VAX/VMS AD11-K DRIVER
        .IDENT  'V04-000'
```



```
;++
; FACILITY:
;
;       VAX/VMS AD11-K I/O DRIVER
;
; ABSTRACT:
;
; DEVICE TABLES AND DRIVER CODE FOR THE AD11-K ANALOGUE
; TO DIGITAL CONVERTER WITH OPTIONAL AM11-K MULTIPLEXER.
;
; AUTHOR:
;
;       S. PROGRAMMER, SEPTEMBER 1978.
;
; MODIFIED BY:
;
;--
```

```
        .SBTTL  FUNCTIONAL DESCRIPTION OF DRIVER
;+
; THE DRIVER SUPPORTS A/D SAMPLING ON GROUPS OF CHANNELS VIA QIO
; READ REQUESTS.  NO EXTERNALLY TRIGGERED SAMPLING (I.E., CLOCK
; OVERFLOW OR SCHMITT TRIGGER) IS SUPPORTED.  THE AM11-K MULTIPLEXER
; MAY BE PRESENT, BUT NO AUTOMATIC RANGING AMPLIFICATION IS
; DONE AT DRIVER LEVEL.  THE BUILT-IN DAC MAY BE USED FOR TESTING VIA
; A LOOPBACK QIO FUNCTION DEFINED ESPECIALLY FOR THIS DEVICE.
;
; THE QIO FUNCTIONS AVAILABLE ARE:
;
; IO$_READVBLK          -READ VIRTUAL BLOCK
; IO$_READLBLK          -READ LOGICAL BLOCK
; IO$_READPBLK          -READ PHYSICAL BLOCK=IO$_LOOPBACK
; IO$_LOOPBACK          -WRITE DAC, READ RESULTS; REQUIRES
;                        PHYSICAL I/O PRIVILEGE
;
; THE STANDARD QIO PARAMETERS ARE:
;
; P1=BUFFER ADDRESS
; P2=BUFFER BYTE COUNT
; P3=SPECIFIER OF CHANNELS TO SAMPLE:
;       BIT 0-7/INITIAL CHANNEL # (0-63)
;       BIT 8-15/TOTAL # OF CHANNELS TO SAMPLE (1-64)
;       BIT 16-23/CHANNEL INCREMENT (0-63)
;       BIT 24-31/IGNORED
; P4=DAC VALUE, USED FOR LOOPBACK ONLY:
;       BIT 0-7/8 BIT DAC VALUE
;       BIT 8-31/IGNORED
; P5,P6 ARE NOT USED
;
; IN ADDITION TO THE STANDARD STATUS CODES THAT CAN BE RETURNED FOR
; A QIO, THE FOLLOWING DEVICE-SPECIFIC I/O STATUS VALUES ARE DEFINED:
;
; SS$_DATAOVERUN        -ERROR BIT SET IN CSR; SAMPLING ABORTED
;                        WITH LAST GOOD SAMPLE IN BUFFER
; SS$_BADPARAM          -INVALID CHANNEL SPECIFIER; NO SAMPLES TAKEN
; SS$_BUFFEROVF         -USER BUFFER OVERRUN; AS MANY CHANNELS AS WILL
;                        FIT ARE SAMPLED
;
; THE SAMPLES ARE RETURNED IN THE CALLER'S BUFFER PACKED ONE SAMPLE
; PER WORD, BITS 0-11. THE BYTE COUNT RETURNED IN THE SECOND WORD OF
; THE I/O STATUS BLOCK ALWAYS REFLECTS THE # OF BYTES ACTUALLY FILLED
; WITH SAMPLE DATA.  THE NUMBER OF SAMPLES IS ONE HALF THE RETURNED
; BYTE COUNT.
;
; EXAMPLE: SWEEP THROUGH 32 INPUTS CONNECTED IN DIFFERENTIAL MODE
;          (AD11-K AND AM11-K):
;
; SWEEPBUF:     .BLKW   32
; NUMINPUT:     .LONG   32
; CHANSPEC:     .BYTE   0,32,2          ;START WITH CHANNEL 0;
;                                       ; SAMPLE CHANNELS 0,2,4,...,62
;
;               $QIO_S  CHAN=X,FUNC=IO$_READVBLK,-
;                       P1=SWEEPBUF,P2=NUMINPUT,P3=CHANSPEC
```

```
;-
```

```
        .SBTTL  MACRO LIBRARY CALLS

;
; EXTERNAL SYMBOLS (LIB/LIB):
;

        $CRBDEF                         ;CHANNEL REQUEST BLOCK
        $DDBDEF                         ;DEVICE DATA BLOCK
        $IDBDEF                         ;INTERRUPT DATA BLOCK
        $IODEF                          ;I/O FUNCTION CODES
        $IPLDEF                         ;HARDWARE IP DEFINITIONS
        $IRPDEF                         ;I/O REQUEST PACKET
        $UCBDEF                         ;UNIT CONTROL BLOCK
        $VECDEF                         ;INTERRUPT VECTOR BLOCK
        $JIBDEF                         ;JOB INFORMATION BLOCK

;
; USER DEFINED EXTERNAL SYMBOLS ARE CONTAINED IN A USER LIBRARY .
; THE CONTENTS OF THIS LIBRARY CAN BE MERGED WITH THE SYSTEM LIBRARY
; TO ALLOW USER PROGRAMS TO USE EXTENDED FUNCTION CODES WITHOUT HAVING
; TO DEFINE THEM LOCALLY.
; THIS DRIVER MUST BE ASSEMBLED WITH A USER LIBRARY TO DEFINE $XIODEF.
;
        $XIODEF                         ;EXTENDED QIO FUNCTIONS.THIS MACRO
                                        ;CONTAINS THE DEFINITIONS FOR
                                        ;IO$_LOOPBACK
```

```
        .SBTTL  LOCAL DEFINITIONS

;
; LOCAL DEFINITIONS:
;
; QIO ARGUMENT LIST OFFSETS:
;

P1=0                                    ;FIRST,
P2=4                                    ; SECOND,
P3=8                                    ; THIRD,
P4=12                                   ; FOURTH,
P5=16                                   ; FIFTH,
P6=20                                   ; AND SIXTH PARAMETERS

;
; DEVICE PARAMETERS:
;

DAC_TIMER=20                            ;20 USEC TIMER FOR DAC SETTLE
MAX_INLCHN=63                           ;MAXIMUM INITIAL CHANNEL #,
MAX_NUMCHN=64                           ; NUMBER OF CHANNELS,
MAX_INCCHN=63                           ; AND CHANNEL INCREMENT
ADC_TIMER=2                             ;A/D CONVERSION TIMEOUT=2 SEC

;
; DEVICE REGISTER DEFINITIONS:
;

        $DEFINI AD

$DEF    AD_CSR  .BLKW   1               ;CONTROL/STATUS REGISTER

        _VIELD  AD_CSR,0,<-             ;DEFINE CSR FIELDS: AD_CSR_M_XXX
        <GO,,M>,-                       ; START A/D CONVERSION
        <,3>,-                          ; 3 UNUSED BITS
        <EXT,,M>,-                      ; EXTERNAL START ENABLE
        <COV,,M>,-                      ; CLOCK OVERFLOW ENABLE
        <IE,,M>,-                       ; INTERRUPT ENABLE
        <DON,,M>,-                      ; CONVERSION DONE FLAG
        <MUX,6,M>,-                     ; 6 BIT MUX CHANNEL #
        <,1>,-                          ; BIT 14 IS UNUSED
        <ERR,,M>,-                      ; ERROR FLAG
        >                               ;END OF CSR FIELDS
$DEF    AD_DBR  .BLKW   1               ;A/D DATA BUFFER REGISTER
.=.-2                                   ;DATA BUFF REG=DAC BUFF REG
$DEF    AD_DAC  .BLKW   1               ;DAC DATA BUFFER REF

        $DEFEND AD                      ;END OF A/D REGISTER DEFNS

;
; DEVICE DEPENDENT UCB EXTENSIONS:
;

        $DEFINI UCB
```

```
.=UCB$K_LENGTH                          ;STEP TO END OF STANDARD UCB
                                        ;NOTE:  NEXT 4 BYTES ASSUMED
                                        ; ADJACENT
$DEF    UCB$B_AD_CURCHN .BLKB   1       ;CURRENT MUX CHANNEL #
$DEF    UCB$B_AD_NUMCHN .BLKB   1       ;# CHANNELS LEFT TO SAMPLE
$DEF    UCB$B_AD_INCCHN .BLKB   1       ;CHANNEL INCREMENT
                        .BLKB   1       ;SPARE BYTE
$DEF    UCB$W_AD_CSR    .BLKW   1       ;SAVED CSR
        $FIELD  UCB$W_CSR,1,<-          ;BORROW UNUSED CSR BIT
        <BFO,,M>,-                      ; FOR USER BUFFER OVERRUN
        >
UCB$K_ADLENGTH=.                        ;LENGTH OF A/D UCB

        $DEFEND UCB                     ;END OF UCB EXTENSIONS
;
; A/D DRIVER USE OF TEMPORARY IRP STORAGE:
;
IRP$L_CHSPEC=IRP$L_MEDIA                ;CHANNEL SPECIFIER(P3)
IRP$L_DACVAL=IRP$L_MEDIA+4              ;OPTIONAL DAC VALUE(P4)
```

```
        .SBTTL  DRIVER PROLOGUE AND DISPATCH TABLES

;
; DRIVER PROLOGUE TABLE:
;

        DPTAB   -                       ;DEFINE DRIVER PROLOGUE TABLE:
                END=AD_END,-            ; END OF DRIVER,
                ADAPTER=UBA,-           ; UNIBUS ADAPTER,
                UCBSIZE=UCB$K_AD_LENGTH,-; SIZE OF A/D UCB,
                NAME=ADDRIVER           ; DRIVER NAME
                                        ;
        DPT_STORE       INIT            ;VALUES TO BE SET ON LOAD
        DPT_STORE       UCB,UCB$B_FIPL,B,8      ;DEVICE FORK IPL
        DPT_STORE       UCB,UCB$B_DIPL,B,22     ;AD11 HARDWARE IPL
        DPT_STORE       UCB,UCB$L_DEVCHAR,L,-   ;AD11 DEVICE CHARACTERISTICS:
                        <DEV$M_AVL-             ; AVAILABLE,
                        !DEV$M_IDV-             ; INPUT DEVICE,
                        !DEV$M_RTM>             ; REALTIME DEVICE
                                                ;
        DPT_STORE       REINIT                  ;VALUES TO SET ON RELOAD
        DPT_STORE       CRB,CRB$L_INTD+4,D,-    ;INTERRUPT SERVICE ADDR
                        AD_INTERRUPT
        DPT_STORE       CRB,-                   ;ADDR OF CONTROLLER
                        CRB$L_INTD+VEC$L_INITIAL,- ; INITIALIZATION
                        D,AD_CTLINIT
        DPT_STORE       CRB,-                   ;ADDR OF UNIT
                        CRB$L_INTD+VEC$L_UNITINIT,- ; INITIALIZATION
                        D,AD_UNITINIT
        DPT_STORE       DDB,DDB$L_DDT,D,-       ;ADDR OF DRIVER
                        AD$DDT                  ; DISPATCH TABLE
                                                ;
        DPT_STORE       END                     ;END DRIVER PROLOGUE

;
; DRIVER DISPATCH TABLE:
;

        DDTAB   -                       ;DDT CREATION MACRO
                DEVNAM=AD,-             ;NAME OF DEVICE
                START=AD_STARTIO,-      ;ADDR OF START I/O ROUTINE
                FUNCTB=AD_FUNCTABLE     ;ADDR OF FDT
```

```
        .SBTTL  AD11-K FUNCTION DECISION TABLE

;
; AD11 FUNCTION DECISION TABLE:
;

AD_FUNCTABLE:                           ;FUNCTION DECISION TABLE START
        FUNCTAB ,-                      ;LEGAL FUNCTIONS:
                <LOOPBACK,-             ; LOOPBACK READ FROM DAC
                READPBLK,-              ; READ PHYSICAL BLOCK
                READLBLK,-              ; READ LOGICAL BLOCK
                READVBLK>               ; READ VIRTUAL BLOCK
        FUNCTAB ,-                      ;BUFFERED I/O FUNCTIONS:
                <LOOPBACK,-             ; LOOPBACK READ FROM DAC
                READPBLK,-              ; READ PHYSICAL BLOCK
                READLBLK,-              ; READ LOGICAL BLOCK
                READVBLK>               ; READ VIRTUAL BLOCK
        FUNCTAB -                       ;PREPROCESSING ROUTINES:
                AD_READ,-               ;CALL SINGLE PREPROCESSOR FOR:
                <LOOPBACK,-             ; LOOPBACK READ FROM DAC
                READPBLK,-              ; READ PHYSICAL BLOCK
                READLBLK,-              ; READ LOGICAL BLOCK
                READVBLK>               ; AND READ VIRTUAL BLOCK
```

```
        .SBTTL  AD_READ:        READ FUNCTION PROCESSING
;+
; AD_READ - READ FUNCTION PREPROCESSING
;
; THIS ROUTINE IS CALLED FROM THE FUNCTION DECISION TABLE DISPATCHER
; TO PROCESS A READ PHYSICAL, READ LOGICAL, READ VIRTUAL, OR LOOPBACK
; I/O FUNCTION.
;
; AD_READ FIRST VERIFIES THE CALLER'S PARAMETERS, TERMINATING THE
; REQUEST WITH IMMEDIATE SUCCESS OR ERROR IF NECESSARY.  P3 AND
; P4 ARE STORED IN THE IRP.  A SYSTEM BUFFER IS ALLOCATED AND
; ITS ADDRESS IS SAVED IN THE IRP.  THE CALLER'S QUOTA IS UPDATED,
; AND THE READ REQUEST IS QUEUED TO THE DRIVER FOR STARTUP.
;
; INPUTS:
;
;       R0,R1,R2 = SCRATCH
;       R3 = IRP ADDRESS
;       R4 = ADDR OF PCB FOR CURRENT PROCESS
;       R5 = DEVICE UCB ADDRESS
;       R6 = ADDRESS OF CCB
;       R7 = I/O FUNCTION CODE
;       R8 = FDT DISPATCH ADDR
;       R9-R11 = SCRATCH
;       AP = ADDR OF FUNCTION PARAMETER LIST
;
; OUTPUTS:
;
;       R0,R1,R2 = DESTROYED
;       R3-R11,AP = PRESERVED
;       IRP$L_CHSPEC(R3) = CHANNEL SPECIFIER (P3)
;       IRP$L_DACVAL(R3) = OPTIONAL DAC VALUE (P4)
;       IRP$L_SVAPTE(R3) = ADDR OF ALLOCATED SYSTEM BUFFER
;       IRP$W_BOFF(R3) = REQUESTED BYTE COUNT
;
;       SYSTEM BUFFER:
;               LONGWD 0/ADDR OF START OF DATA=BUFF ADDR+12
;               LONGWD 1/ADDR OF USER BUFFER
;               LONGWD 2/DATA STRUCTURE BOOKKEEPING
;-
;
        .ENABL  LSB
;
AD_READ:                                ;READ FUNCTION PREPROCESSING
        MOVZWL  P2(AP),R1               ;GET USER BYTE COUNT
        BEQL    10$                     ;BRANCH IF READ OF 0 BYTES
                                        ; (=INSTANT SUCCESS)
        MOVZWL  #SS$_BADPARAM,R0        ;ASSUME CHANNEL SPEC ERROR
        MOVAL   P3(AP),R2               ;GET ADDR OF CHANNEL SPEC
        CMPB    (R2)+,#MAX_INLCHN       ;INITIAL CHAN # TOO LARGE?
        BGTRU   20$                     ;BRANCH IF SO
        TSTB    (R2)                    ;# CHANNELS = 0?
        BEQL    10$                     ;BRANCH IF SO (SUCCESS)
        CMPB    (R2)+,#MAX_NUMCHN       ;# CHANNELS TO SAMPLE TOO LARGE?
        BGTRU   20$                     ;BRANCH IF SO
        CMPB    (R2),#MAX_INCCHN        ;CHANNEL INCREMENT TOO LARGE?
```

```
        BGTRU   20$                     ;BRANCH IF SO
        MOVQ    P3(AP),IRP$L_CHSPEC(R3) ;STORE P3 AND P4 (OPTIONAL DAC)
                                        ; IN IRP UNTIL REQUEST EXECUTION
        MOVL    P1(AP),R0               ;GET ADDR OF USER BUFFER
        JSB     G^EXE$READCHK           ;VERIFY THAT CALLER HAS
                                        ; WRITE ACCESS TO BUFFER
        PUSHR   #^M<R0,R3>              ;SAVE USER BUFF ADDR, IRP ADDR
        ADDL    #12,R1                  ;ADD 12 BYTES TO REQUESTED BUFF
                                        ; SIZE FOR BUFF HEADER
        JSB     G^EXE$BUFFRQUOTA        ;VERIFY BUFFER SPACE LEFT
                                        ; IN CALLER'S QUOTA
        BLBC    R0,30$                  ;BRANCH IF INSUFFICIENT QUOTA
        JSB     G^EXE$ALLOCBUF          ;ALLOCATE A SYSTEM BUFFER
        BLBC    R0,30$                  ;BRANCH IF NONE AVAILABLE
        POPR    #^M<R0,R3>              ;RESTORE USER BUFFER, IRP ADDR
        MOVL    R2,IRP$L_SVAPTE(R3)     ;SAVE ADDR OF SYSTEM BUFFER
        MOVW    R1,IRP$W_BOFF(R3)       ; AND REQUESTED BYTE COUNT
        MOVZWL  R1,R1                   ;CONVERT TO LONGWORD
        MOVL    PCB$L_JIB(R4),R4        ;GET JOB INFORMATION BLOCK ADDRESS
        SUBL    R1,JIB$L_BYTCNT(R4)     ;DEDUCT REQUESTED BYTE COUNT
                                        ; FROM PROCESS' QUOTA
        MOVAB   12(R2),(R2)+            ;SAVE ADDR OF START OF USER DATA
                                        ; IN 1ST LONGWD OF SYSTEM BUFFER
        MOVL    R0,(R2)                 ;SAVE USER BUFFER ADDR IN
                                        ; 2ND LONGWD
        JMP     G^EXE$QIODRVPKT         ;QUEUE I/O PKT TO DRIVER

;
; COME HERE IF USER REQUESTED READ OF 0 BYTES OR 0 CHANNELS.
; THIS IS ALWAYS SUCCESSFUL AND DOES NO DEVICE I/O:
;

10$:    MOVZWL  #SS$_NORMAL,R0          ;SET NORMAL COMPLETION STATUS
20$:    JMP     G^EXE$FINISHIOC         ;COMPLETE I/O REQUEST

;
; COME HERE TO ABORT I/O REQUEST WITH EXCEPTION STATUS IN R0:
;

30$:    POPR    #^M<R2,R3>              ;CLEAR BUFFER ADDR; RESTORE IRP
                                        ; ADDR
        JMP     G^EXE$ABORTIO           ;COMPLETE I/O REQUEST

        .DSABL  LSB
```

```
        .SBTTL  AD_STARTIO:     PERFORM A/D CONVERSIONS
;+
; AD_STARTIO - START I/O OPERATION ON AD11-K A/D CONVERTER.
;
; THIS ROUTINE IS ENTERED WHEN THE ASSOCIATED UNIT IS IDLE AND A
; PACKET IS AVAILABLE FOR PROCESSING.
;
; TO PREPARE FOR SAMPLING, AD_STARTIO PERFORMS THESE STEPS:
;
; 1. SET UP UCB WITH CHANNEL SPECIFIER AND ADDRESS IN SYSTEM
;    BUFFER TO HOLD FIRST SAMPLE.
; 2. IF LOOPBACK WAS SPECIFIED, THE DAC IS SET WITH THE CALLER-
;    SPECIFIED VALUE.
;
; THE DRIVER THEN LOOPS FROM AD_NXTSAMPLE TO AD_ENDSAMPLE
; COLLECTING SAMPLES UNTIL ALL SAMPLES HAVE BEEN COLLECTED,
; OR AN ERROR OCCURS.  AN INTERRUPT IS RECEIVED FOR EACH SAMPLE,
; BUT, TO SAVE TIME, THE DRIVER NEVER FORKS UNTIL TIME TO
; COMPLETE THE I/O REQUEST.
;
; INPUTS:
;
;       R3 = ADDR OF IRP
;       R5 = ADDR OF DEVICE UNIT UCB
;
; OUTPUTS:
;
;       R0,R1,R2 = DESTROYED
;       OTHER REGISTERS ARE PRESERVED
;-

        .ENABL  LSB

AD_STARTIO:                             ;START NEXT QIO
        MOVL    IRP$L_CHSPEC(R3),-      ;COPY CHANNEL SPEC FROM
                UCB$B_AD_CURCHN(R5)     ; IRP TO UCB
        MOVL    @IRP$L_SVAPTE(R3),-     ;SET ADDR OF START DATA
                UCB$L_SVAPTE(R5)        ; IN UCB
        MOVL    UCB$L_CRB(R5),R4        ;GET CRB ADDRESS,
        MOVL    @CRB$L_INTD+VEC$L_IDB(R4),R4 ; THEN CSR ADDRESS
        BICB3   #^C<IO$M_FCODE>,-       ;GET THE I/O
                IRP$W_FUNC(R3),R0       ; FUNCTION CODE
        CMPB    R0,#IO$_LOOPBACK        ;LOOPBACK?
        BNEQ    AD_NXTSAMPLE            ;BRANCH IF NOT
        MOVZBW  IRP$L_DACVAL(R3),-      ;SET DAC VALUE IN
                AD_DAC(R4)              ; DAC BUFFER REGISTER
        MFPR    S^#PR$_ICR,R1           ;GET CURRENT INTERVAL COUNTER (USEC)
        ADDL    #DAC_TIMER,R1           ; +DAC SETTLE TIME IN USEC
        BLSS    10$                     ;BRANCH IF COUNTER DOESN'T
                                        ; OVERFLOW
        MOVAW   -10000(R1),R1           ;ELSE CALCULATE COUNTER
                                        ; FOR NEXT INTERVAL
10$:    MFPR    S^#PR$_ICR,R0           ;READ INTERVAL COUNTER NOW
        CMPL    R0,R1                   ;REACHED SETTLE TIME YET?
        BLSS    10$                     ;BRANCH IF NOT
```

```
AD_NXTSAMPLE:                           ;START NEXT SAMPLE
        MOVZBW  #AD_CSR_M_IE!AD_CSR_M_GO,R0 ;SET INTERRUPT ENABLE AND
                                        ; START A/D CONVERSION
        INSV    UCB$B_AD_CURCHN(R5),-   ;SET MUX CHAN #
                #8,#6,R0                ; FOR CSR
        DSBINT                          ;DISABLE INTERRUPTS (IPL=IPL$POWER)
        BBSC    #UCB$V_POWER,-          ;BRANCH IF POWER FAILURE
                UCB$W_STS(R5),AD_POWERFAIL ; AND CLEAR POWER FAIL SIGNAL
        MOVW    R0,AD_CSR(R4)           ;SET CSR
        WFIKPCH AD_TIMEOUT,#ADC_TIMER   ;WAIT FOR INTERRUPT, OR TIMEOUT
        MOVW    AD_CSR(R4),UCB$W_AD_CSR(R5) ;SAVE CSR IN UCB
        BLSS    AD_CSRERROR             ;BRANCH IF ERROR
        MOVW    AD_DBR(R4),@UCB$L_SVAPTE(R5) ;COPY A/D VALUE INTO
                                        ; SYSTEM BUFFER
        ADDL    #2,UCB$L_SVAPTE(R5)     ;STEP BUFFER POINTER
        SUBL    #2,UCB$W_BCNT(R5)       ;DECREASE # BYTES LEFT IN REQUEST
        DECB    UCB$B_AD_NUMCHN(R5)     ;DECR # CHANNELS LEFT TO SAMPLE
        BEQL    AD_DONE                 ;BRANCH IF NONE
        CMPW    UCB$W_BCNT(R5),#2       ;AT LEAST 2 BYTES LEFT IN BUFFER?
        BLSSU   AD_BUFFEROVF            ;BRANCH IF NOT
        BICW    #UCB$W_CSR_M_BFO,-      ;ELSE CLEAR BUFFER OVERRUN
                UCB$W_AD_CSR(R5)        ; BIT IN CSR COPY
        ADDB    UCB$B_AD_INCCHN(R5),-   ;NEXT CHANNEL # =
                UCB$B_AD_CURCHN(R5)     ; CURRENT CHANNEL+INCREMENT
        BICB    #^C<MAX_NUMCHN-1>,-     ; MODULO MAXIMUM
                UCB$B_AD_CURCHN(R5)     ; CHANNEL #

AD_ENDSAMPLE:                           ;THIS SAMPLE COMPLETE
        BRB     AD_NXTSAMPLE            ;GO START NEXT SAMPLE

        .DSABL  LSB
```

```
        .SBTTL  -               I/O REQUEST COMPLETION

;
; COME HERE TO COMPLETE I/O REQUEST WITH NORMAL OR ERROR STATUS.
;
; USER BUFFER OVERRUN, I.E., NO MORE SAMPLES CAN BE COLLECTED:
;
        .ENABL  LSB

AD_BUFFEROVF:                           ;
        BISW    #UCB$W_CSR_M_BFO,-      ;SET BUFFER OVERRUN BIT
                UCB$W_AD_CSR(R5)        ; IN CSR COPY

;
; CSR ERROR BIT WAS SET:
;
AD_CSRERROR:                            ;
        TSTW    AD_DBR(R4)              ;CLEAR ERROR
        BRB     AD_DONE                 ;JOIN COMMON I/O COMPLETION

;
; DEVICE TIMED OUT DUE TO EITHER A REAL TIMEOUT OR TO A
; POWER FAILURE.  BOTH CAUSES ARE HANDLED THE SAME.
;
AD_TIMEOUT:                             ;
        CLRW    AD_CSR(R4)              ;CLEAR INTERRUPT ENABLE,
        TSTW    AD_DBR(R4)              ; PENDING CONVERSION, INT, OR ERROR
        SETIPL  UCB$B_FIPL(R5)          ;LOWER PRIORITY TO DEVICE LEVEL
        BRB     10$                     ;JOIN COMMON CODE TO
                                        ; TERMINATE REQUEST

;
; POWER FAILURE DETECTED WHILE ATTEMPTING TO INITIATE A READ OR
; LOOPBACK REQUEST.  TERMINATE REQUEST THE SAME AS IF IT OCCURRED
; DURING THE QIO.
;
AD_POWERFAIL:                           ;
        ENBINT                          ;LOWER IPL BACK TO FORK IPL
10$:    MOVZWL  #SS$_TIMEOUT,R0         ;SET STATUS TO TIMED OUT
        BRB     20$                     ;JOIN COMMON CODE TO TERMINATE
                                        ; REQUEST

;
; NORMAL STATUS, CANCEL I/O, AND GENERAL I/O REQUEST COMPLETION:
;
AD_DONE:                                ;
        CLRW    AD_CSR(R4)              ;CLEAR INTERRUPT ENABLE
        IOFORK                          ;REQUEST RESUMPTION AS FORK PROCESS
        MOVZWL  #SS$_DATAOVERUN,R0      ;ASSUME CSR ERROR
        BBS     #AD_CSR_V_ERR,-         ;BRANCH IF SO
                UCB$W_AD_CSR(R5),20$    ;
```

```
        MOVZWL  #SS$_BUFFEROVF,R0       ;ASSUME BUFFER OVERRUN
        BBS     #UCB$V_CSR_V_BFO,-      ;BRANCH IF SO
                UCB$W_AD_CSR(R5),20$    ;
        MOVZWL  #SS$_NORMAL,R0          ;ELSE, STATUS IS NORMAL
                                        ;
20$:    SUBW3   UCB$W_BCNT(R5),-        ;GET # BYTES REQUESTED
                IRP$W_BCNT(R3),R1       ; -# BYTES NOT XFERRED
        INSV    R1,#16,#16,R0           ; =# BYTES XFERRED
        CLRL    R1                      ;CLEAR SECOND I/O STATUS LONGWD
        REQCOM                          ;REQUEST I/O COMPLETION

        .DSABL  LSB
```

```
        .SBTTL  AD_INTERRUPT:   AD11-K A/D CONVERTER INTERRUPT SERVICE

;+
; AD_INTERRUPT - A/D CONVERTER INTERRUPT SERVICE
;
; THIS ROUTINE IS ENTERED VIA A JSB INSTRUCTION WHEN AN
; INTERRUPT OCCURS ON AN AD11 A/D CONVERTER.  INTERRUPT SERVICE
; GETS THE ADDRESS OF THE UCB OF THE INTERRUPTING DEVICE, RESTORES
; THE REMAINING CONTEXT OF THE DRIVER FORK PROCESS WHICH INITIATED
; THE DEVICE ACTIVITY, AND CALLS THE DRIVER FORK PROCESS.
;
; INPUTS:
;
;       ALL GENERAL REGISTERS = RANDOM
;       SP/ INTERRUPT STACK
;       0(SP) = ADDR OF IDB ADDR
;       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 PC
;       32(SP) = SAVED PSL
;       IPL/ HARDWARE DEVICE LEVEL
;
; OUTPUTS AT CALL TO DRIVER FORK:
;
;       R3 = RESTORED FROM DRIVER FORK PROCESS (IRP ADDR)
;       R4 = RESTORED FROM DRIVER FORK PROCESS (CSR ADDR)
;       R5 = UCB ADDR
;       STACK IS SAME AS ABOVE, BUT IDB POINTER POPPED
;       IPL/ HARDWARE DEVICE LEVEL
;-

        .ENABL  LSB

AD_INTERRUPT:                           ;A/D CONVERTER INTERRUPT SERVICE
        MOVL    @(SP)+,R3               ;GET IDB ADDR
        MOVQ    IDB$L_CSR(R3),R4        ;GET DEVICE CSR AND UCB ADDR
        BBCC    #UCB$V_INT,-            ;BRANCH IF INT UNEXPECTED,
                UCB$W_STS(R5),AD_UNSOL  ; AND CLEAR EXPECTED BIT
        MOVL    UCB$L_FR3(R5),R3        ;RESTORE REMAINING DRIVER
                                        ; CONTEXT: R3; (R4 ALREADY SET)
        JSB     @UCB$L_FPC(R5)          ;CALL DRIVER FORK PROCESS
                                        ;
10$:    MOVQ    (SP)+,R0                ;RESTORE REGISTERS
        MOVQ    (SP)+,R2                ;
        MOVQ    (SP)+,R4                ;
        REI                             ;
AD_UNSOL:                               ;HANDLE UNSOLICITED INTERRUPT
        CLRW    AD_CSR(R4)              ;DISMISS SPURIOUS INTERRUPT
        TSTW    AD_DBR(R4)              ;READ DATA BUFFER TO CLEAR ERROR
        BRB     10$                     ;JOIN INTERRUPT RESTORE

        .DSABL  LSB
```

```
        .SBTTL  AD_CTLINIT:     AD11-K CONTROLLER INITIALIZATION
;+
; AD_CTLINIT - AD11-K CONTROLLER INITIALIZATION
;
; THIS ROUTINE IS CALLED AT SYSTEM STARTUP AND AFTER A POWER
; FAILURE.
;
; THE CSR IS CLEARED TO DISABLE INTERRUPTS.  THIS WILL FORCE THE
; LAST SAMPLE (IF ONE IS IN PROGRESS) TO TIME OUT IN CASE INITIALIZATION
; IS THE RESULT OF A POWER FAILURE.  THE TIMEOUT WILL OCCUR IN 0-1
; SECONDS.
;
; THE DATA BUFFER REGISTER IS READ TO CLEAR A PENDING CONVERSION,
; INTERRUPT, OR ERROR FOR DEVICE INITIALIZATION.
;
; INPUTS:
;
;       R4 = AD11 CSR ADDRESS
;       R5 = IDB ADDRESS OF DEVICE UNIT
;       R6 = ADDR OF DDB
;       R8 = ADDR OF CRB
;
; OUTPUTS:
;
;       ALL REGISTERS PRESERVED
;-

AD_CTLINIT:                             ;
        CLRW    AD_CSR(R4)              ;CLEAR CSR (IE IN PARTICULAR)
        TSTW    AD_DBR(R4)              ;CLEAR ANY PENDING CONVERSION,
                                        ; INTERRUPT, OR ERROR
        RSB                             ;
```

```
        .SBTTL  AD_UNITINIT:    AD11-K UNIT INITIALIZATION                      ; S
;+
; AD_UNITINIT - AD11-K UNIT INITIALIZATION                                      90$
;
; THIS ROUTINE IS CALLED AT SYSTEM STARTUP AND AFTER A POWER
; FAILURE.  THE UCB AND IDB ARE INITIALIZED.
;
; INPUTS:
;
;       R5 = ADDRESS OF DEVICE UCB
;
; OUTPUTS:
;
;       R0 = IDB ADDRESS
;       OTHER REGISTERS ARE PRESERVED
;       UCB$W_STS(R5), ONLINE BIT IS SET
;       IDB$L_OWNER(R0) = ADDRESS OF OWNING UCB
;-

AD_UNITINIT:                            ;
        BISW    #UCB$M_ONLINE,-         ;SET UNIT ONLINE
                UCB$W_STS(R5)           ;
        MOVL    UCB$L_CRB(R5),R0        ;GET CRB ADDRESS
        MOVL    CRB$L_INTD+VEC$L_IDB(R0),R0 ;GET IDB ADDR
        MOVL    R5,IDB$L_OWNER(R0)      ;SET UCB ADDR OF OWNING UNIT
        RSB                             ;

AD_END:                                 ;END OF DRIVER LABEL



        .END
.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