```
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
```

```
LL              AAAAAA    BBBBBBBB      IIIIII      OOOOOO      AAAAAA      CCCCCCCC    QQQQQQ
LL              AAAAAA    BBBBBBBB      IIIIII      OOOOOO      AAAAAA      CCCCCCCC    QQQQQQ
LL            AA      AA  BB      BB      II      OO      OO  AA      AA  CC          QQ      QQ
LL            AA      AA  BB      BB      II      OO      OO  AA      AA  CC          QQ      QQ
LL            AA      AA  BB      BB      II      OO      OO  AA      AA  CC          QQ      QQ
LL            AA      AA  BB      BB      II      OO      OO  AA      AA  CC          QQ      QQ
LL            AA      AA  BBBBBBBB        II      OO      OO  AA      AA  CC          QQ      QQ
LL            AA      AA  BBBBBBBB        II      OO      OO  AA      AA  CC          QQ      QQ
LL            AAAAAAAAAA  BB      BB      II      OO      OO  AAAAAAAAAA  CC          QQ  QQ  QQ
LL            AAAAAAAAAA  BB      BB      II      OO      OO  AAAAAAAAAA  CC          QQ  QQ  QQ
LL            AA      AA  BB      BB      II      OO      OO  AA      AA  CC          QQ    QQ
LL            AA      AA  BB      BB      II      OO      OO  AA      AA  CC          QQ    QQ      ....
LLLLLLLLLL    AA      AA  BBBBBBBB      IIIIII      OOOOOO      AA      AA    CCCCCCCC    QQQQ  QQ    ....
LLLLLLLLLL    AA      AA  BBBBBBBB      IIIIII      OOOOOO      AA      AA    CCCCCCCC    QQQQ  QQ    ....

FFFFFFFFFF      OOOOOO    RRRRRRRR
FFFFFFFFFF      OOOOOO    RRRRRRRR
FF            OO      OO  RR      RR
FF            OO      OO  RR      RR
FF            OO      OO  RR      RR
FF            OO      OO  RR      RR
FFFFFFFF      OO      OO  RRRRRRRR
FFFFFFFF      OO      OO  RRRRRRRR
FF            OO      OO  RR  RR
FF            OO      OO  RR  RR
FF            OO      OO  RR    RR
FF            OO      OO  RR    RR
FF              OOOOOO    RR      RR
FF              OOOOOO    RR      RR
```

N 9

```
File:  LABIOACQ.FOR
       Version 'V04-000'
```



```
        Program LABIO_DATA_ACQ

! This is the program that acquires data for the LABIO system
! It uses the connect-to-interrupt feature of VMS to acquire
! via a user written I/O routine. The actual I/O routine is
! written in MACRO. The main program monitors the event flags
! and enables and disables data acquisition for each channel.
! It also notifies users via event flags when a buffer is full.


! Define the LABIO data base

        Include 'LABCHNDEF.FOR'

! Local Variables
        Logical*4 SECTION_FLAGS, SECTION_PROT

! System Services
        Logical*4 SYS$ASCEFC,SYS$MGBLSC,SYS$ASSIGN,SYS$QIO
        Logical*4 SYS$CLREF

! External constants
        External SEC$M_GBL,SEC$M_WRT,SS$_CREATED,SS$_WASSET
        External SET_EF_AST

! Misc.
        Logical*4 AD_CIN_UP,SUCCESS

!
! Create the Global Section for the data buffer
```

```
! This data buffer will be READ/WRITE for the owner, READ only  for the GROUP.
!
! First see if the global section already exists, if it
! does just map to it. and set the restart flag.
!
! If not, Open the Data File. This can not be openned
! via FORTRAN since we need the VMS channel number.
!
!
      SECTION(1) = %Loc( LABIO_BUFFER_S)      !Start address of section
      SECTION(2) = %Loc( LABIO_BUFFER_E) - 1  !End address
! Page count for the section
      SECTION_SIZE = ( SECTION(2) - SECTION(1) )/512 + 1

! FLAGS for Section are GLOABAL,SHARED,NON_ZEROED,READ/WRITE,TEMP,GLOBAL
!
        SECTION_FLAGS = %Loc( SEC$M_GBL ) + %Loc( SEC$M_WRT )

! Try just mapping to the global section
      SUCCESS = SYS$MGBLSC( SECTION,,,%Val(SECTION_FLAGS),'LABIOCOMMON',,)
      If( SUCCESS ) Then
        RESTART = .TRUE.      !Succes, this is a restart
      Else
        SUCCESS = GBL_SECTION_UFO( SECTION_SIZE, 'LABIO_SEC_FILE',
     1                             SECTION_CHANNEL )
         f( .not. SUCCESS )
     1   Call FATAL_ERROR(SUCCESS,'Opening Global Section File')

! PROTECTION is OWNER = READ/WRITE, GROUP = READ, SYSTEM/WORLD = none
!
        SECTION_PROT = 'F E 0 F'X !Protection for section

! Create and Map the Section
!
        SUCCESS = SYS$CRMPSC( SECTION,,,%Val(SECTION_FLAGS),'LABIOCOMMON',
     1            ,,%Val(SECTION_CHANNEL),%Val(SECTION_SIZE),,
     1           %Val(SECTION_PROT),%Val(SECTION_SIZE))
        If( .not. SUCCESS )
     1 Call FATAL_ERROR(SUCCESS,'Creating Global Section')
        RESTART = .FALSE.                       !We are not restarting
      End If
!
! If this is not a restart, clear the data structures
!
      If( .not. RESTART ) Then
       Do 32 I = 1, MAX_AD_CHANNEL              !Clear AD_BLOCK
        Do 30 J = 1, 16
30       AD_BLOCK(J,I) = 0
        Do 31 K = 1, BUFFER_COUNT               !Clear Data buffers
         Do 31 J = 1, MAX_BUF_SIZE
31         DATA_BUFFER(J,K,I) = 0
32     Continue
       Do 33 I = 1, MAX_PID
        Do 33 J = 1,2
33       CONNECT_BLOCK(I,J) = 0                 !Clear Process connect block
      End If
```

```
!
! Create event flag cluster EF_NOTIFY and associate with event flags 64-95
! These are used to notify the Data Acquisition process.

        SUCCESS = SYS$ASCEFC( %VAL(EF_NOTIFY_1),EF_NOTIFY_CLSTR,,)
        If ( .not. SUCCESS)
        1       Call FATAL_ERROR( SUCCESS, 'CREATING EVENT FLAG CLUSTER')

! Create event flag cluster EF_STATUS and associate with event flags 96-127
! These are used to notify and report the status of the user buffers
!
        SUCCESS = SYS$ASCEFC( %VAL(EF_STATUS_1),EF_STATUS_CLSTR,,)
        If ( .not. SUCCESS)
        1       Call FATAL_ERROR( SUCCESS, 'CREATING EVENT FLAG CLUSTER')


! Make sure that we can't be swapped
!
        Call SYS$SETSWM(%Val(1))


! Set-up the Connect-to-Interrupt
! First assign a VMS channel for the device
! Then call the connect-to-interrupt setup routine.
!
        SUCCESS = SYS$ASSIGN( 'LABIO_AD',CIN_CHANNEL,, )
        If ( .not. SUCCESS )
        1 Call FATAL_ERROR( SUCCESS, 'assigning A/D device' )

        SUCCESS = AD_CIN_SETUP( CIN_CHANNEL,SET_EF_AST )
        If( .not. SUCCESS )
        1 Call FATAL_ERROR( SUCCESS, 'connecting-to-interrupt')
!
! End Of Initialization, Notify other processes by setting EF_DATA_ACQ

        Call SYS$SETEF( %Val( EF_DATA_ACQ) )
```

```
!
! Wait for an event flag in the EF_NOTIFY cluster
! Then read the EF_NOTIFY CLUSTER and EF_STATUS_CLUSTER

10      Call SYS$WFLOR( %Val(EF_NOTIFY_1) , %Val('FFFF'X) )
!
! Look for the flag(s) set in EF_NOTIFY
! If the corresponding activity flag is set, activate the channel,
! otherwise deactivate it. Also check the buffer status flag, if clear
! clear the buffer index.
!
        Do 20 I = 1,16
        If( SYS$CLREF( %Val(EF_NOTIFY_OFF + I)) .eq. %Loc(SS$_WASSET)) Then
          If( AD_BLOCK(1,I) .ne. 0 ) Then
            If( SYS$READEF( %Val(EF_ACTIVITY_OFF + I),EF_STATE )
     1                          .eq. %Loc(SS$_WASSET ) ) Then
                AD_BLOCK(1,I) = ACTIVE
             Else
                AD_BLOCK(1,I) = INACTIVE
             End if
            If( SYS$READEF( %Val(EF_STATUS_OFF + I),EF_STATE )
     1                          .eq. %Loc(SS$_WASCLR))  AD_BLOCK(7,I) = 0
          End If
        End If
20      Continue
        Go To 10

        End
```

```
        Subroutine SET_EF_AST( EVENT_FLAGS )

! This is a AST routine which is invoked by the
! Interrupt service routine. This routine sets
! the event flags indicated by the ISR.

        Include 'LABCHNDEF.FOR'
        Integer EVENT_FLAGS

!
! The Event flags are set in cluster EF_STATUS_CLSTR
!

        Do 10 I = 1,16
        If( (EVENT_FLAGS .and. BIT(I)) .ne. 0 )
        ;  Call SYS$SETEF( %Val(EF_STATUS_OFF + I) )
10      Continue
        Return

        End
![End of File]
```

XALINK
MAR

DRMASTER
FOR

LABIOLINK
COM

LPATEST
FOR

LABIOPEAK
FOR

LABIOSTRT
COM

XAMESSAGE
MAR

XATEST
FOR

LABIOCOM
FOR

LBRDEMO
COM

LABMBXDEF
FOR

LABIOSAMP
FOR

MAILCOMPRESS
COM

LABCHNDEF
FOR

CONNECT
COM

LABIOCON
FOR

LBRDEMO
FOR

PEAK
FOR

DRCOPYBLD
COM

XIDRIVER
MAR

LABIOSEC
FOR

DRSLAVE
FOR

LABIOACQ
FOR

LABIOCOMP
COM

LABIOSTAT
FOR

TESTLABIO
FOR