```
        .TITLE  HPEMS
        .IDENT  /001/
;
; EMS/EUN 1984-04-18
;
; This module contains all procedures of the HPEMS
; package.  They are:
; HPGET :  Get character if one has come.
; HPGETW:  Get character, wait if necessary.
; HPPUT :  Put out character string.
; HPPUTS:  Report number of strings in queue to be put out.
;
; Its purpose is to provide a fast and flexible
; interface for transfer of files between the VAX and HP
; computers.
;
; Success/error status is returned in the last argument
; according to EMS standard, i. e. a positive number for
; success and a negative number for error. Furthermore, a
; VAX standard condition value is returned in register R0.
;
; The difference between a loop that waits for HPGET to get a
; character and a single call to HPGETW is that HPGETW permits
; lower priority processes to execute during the wait.
;
; HPGET and HPGETW use event flag 21.
; HPPUT uses event flag 20.
;
;
;
        $SSDEF
        $IODEF
        $TTDEF
;
;
;
        .PSECT  HPEMS_DATA, PIC, USR, CON, REL, -
                LCL, NOSHR, NOEXE, RD, WRT, LONG
;
HPLINE_DESC:  .ASCID /_TTB4/                    ; DEVICE NAME DESCRIPTOR
;
TT_CHAN:  .WORD  0                              ; CHANNEL NUMBER
;
TT_PUT_STAT:  .LONG -1                          ; NUMBER OF STRINGS IN QUEUE
;
INIT_DONE:  .LONG 0                             ; INIT DONE FLAG
;
GET_BUF:  .BYTE  0                              ; INPUT CHARACTER
;
TYPEAHDCNT_BUF:  .BLKB 8                        ; TYPE AHEAD INFO
;
;
;
        .PSECT  HPEMS_CODE, PIC, USR, CON, REL, -
                LCL, SHR, EXE, RD, NOWRT, LONG
;
;
;
; Output completion AST routine.
;
PUT_AST:  .WORD  ^M<>                           ; ENTRY MASK
```

```
        DECL    TT_PUT_STAT                     ; ONE STRING LESS IN QUEUE
        RET                                     ; RETURN
;
;
;
;  Initialization routine.
;
INIT_SUB:  $ASSIGN_S -                          ; GET CHANNEL NUMBER
                DEVNAM = HPLINE_DESC, -
                CHAN = TT_CHAN
        BLBS    R0, 60$
        RSB                                     ; ERROR RETURN
;
60$:    MOVL    #-1, INIT_DONE                  ; SET INIT DONE FLAG
        RSB                                     ; NORMAL RETURN
;
;
;
;       CALL HPGET ( NCHAR, CHAR, STATUS )
;
;  NCHAR  number of characters, longword
;  CHAR   character, byte
;  STATUS EMS success/error code, longword
;
;  The subroutine HPGET checks if there is a character in the
;  type-ahead input buffer.  If there is, then NCHAR is set to 1
;  and the first character is moved to CHAR.  If no character
;  is in the buffer, NCHAR is set to 0 and CHAR is left
;  unchanged.
;
        .ENTRY  HPGET, ^M<>                     ; ENTRY MASK
;
        BLBS    INIT_DONE, 20$                  ; INIT DONE?
        BSBW    INIT_SUB                        ; NO, DO INIT
        BLBS    R0, 20$
        MOVL    #-1, @12(AP)                    ; INIT ERROR
        RET
;
20$:    $QIO_S -                                ; CHECK INPUT BUFFER
                CHAN = TT_CHAN, -
                FUNC = #IO$_SENSEMODE!IO$M_TYPEAHDCNT, -
                P1 = TYPEAHDCNT_BUF
        BLBS    R0, 30$
        MOVL    #-2, @12(AP)                    ; EVENT FLAG ERROR
        RET
;
30$:    TSTW    TYPEAHDCNT_BUF                  ; BUFFER EMPTY?
        BNEQ    40$                             ; BRANCH IF CHARACTER IN BUFFE
        CLRL    @4(AP)                          ; NO CHARACTER
        MOVL    #1, @12(AP)                     ; SUCCESS
        MOVL    #SS$_NORMAL, R0
        RET
;
40$:    $QIOW_S -                               ; READ ONE CHARACTER
                EFN = #21, -
                CHAN = TT_CHAN, -
                FUNC = #IO$_TTYREADALL!IO$M_NOECHO, -
                P1 = GET_BUF, -
                P2 = #1
        BLBS    R0, 50$
        MOVL    #-3, @12(AP)                    ; READ ERROR
```

```
        RET
;
50$:    MOVL    #1, @4(AP)              ; ONE CHARACTER
        MOVB    GET_BUF, @8(AP)         ; THE CHARACTER
        MOVL    #1, @12(AP)             ; SUCCESS
        MOVL    #SS$_NORMAL, R0
        RET
;
;
;
;       CALL HPGETW ( CHAR, STATUS )
;
; CHAR    character, byte
; STATUS EMS success/error code, longword
;
; The subroutine HPGETW gets the first character in the
; type-ahead input buffer and moves it into CHAR.  If no
; character is in the buffer, HPGETW waits until one comes.
;
        .ENTRY  HPGETW, ^M<>            ; ENTRY MASK
;
        BLBS    INIT_DONE, 40$          ; INIT DONE?
        BSBW    INIT_SUB                ; NO, DO INIT
        BLBS    R0, 40$
        MOVL    #-1, @8(AP)             ; INIT ERROR
        RET
;
40$:    $QIOW_S -                       ; READ ONE CHARACTER
                EFN = #21, -
                CHAN = TT_CHAN, -
                FUNC = #IO$_TTYREADALL!IO$M_NOECHO, -
                P1 = GET_BUF, -
                P2 = #1
        BLBS    R0, 50$
        MOVL    #-3, @8(AP)             ; READ ERROR
        RET
;
50$:    MOVB    GET_BUF, @4(AP)         ; TRANSFER THE CHARACTER
        MOVL    #1, @8(AP)              ; SUCCESS
        MOVL    #SS$_NORMAL, R0
        RET
;
;
;
;       CALL HPPUT ( NCHAR, CHAR, STATUS )
;
; NCHAR   number of characters, longword
; CHAR    character string, byte array
; STATUS EMS success/error code, longword
;
; The subroutine HPPUT puts a character string in an output
; queue.  If the queue is already full, then
; HPPUT waits until the current output has been completed,
; puts the string in the queue and returns to the calling
; program.
;
        .ENTRY  HPPUT, ^M<>             ; ENTRY MASK
;
        BLBS    INIT_DONE, 20$          ; INIT DONE?
        BSBW    INIT_SUB                ; NO, DO INIT
        BLBS    R0, 20$
```

```
        MOVL    #-1, @12(AP)            ; INIT ERROR
        RET
;
20$:    MOVL    @4(AP), R1              ; LENGTH OF STRING
        CMPL    R1, #0                  ; LESS THAN ZERO?
        BGEQ    30$
        MOVL    #-2, @12(AP)            ; YES, LENGTH ERROR
        RET
;
30$:    BEQL    40$                     ; ZERO LENGTH STRING?
        INCL    TT_PUT_STAT             ; NO, INCREMENT OUTPUT COUNT
        $QIO_S -                        ; STRING TO OUTPUT QUEUE
                EFN = #20, -
                CHAN = TT_CHAN, -
                FUNC = #IO$_WRITEVBLK!IO$M_NOFORMAT, -
                ASTADR = PUT_AST, -
                P1 = @8(AP), -
                P2 = @4(AP)
        BLBS    R0, 40$
        MOVL    #-3, @12(AP)            ; WRITE ERROR
        RET
;
40$:    MOVL    #1, @12(AP)             ; SUCCESS
        MOVL    #SS$_NORMAL, R0
        RET
;
;
;
;       CALL HPPUTS ( QUEUE, STATUS )
;
; QUEUE  number of strings in queue, longword
; STATUS EMS success/error code, longword
;
; The subroutine HPPUTS checks the number of strings in the
; queue for output. Its purpose is to make
; it possible to keep the queue at a suitable length when
; maximum output speed is required at the same time as no
; calculation time is lost in the same process.
;
; If no output is going on, then QUEUE is set to -1. If one
; string is being output but none is waiting, QUEUE is set to 0.
; If one string is being output, and one is waiting, QUEUE is
; set to 1, etc.
;
; N O T E !  Strings output by RMS (including FORTRAN WRITE)
; are not counted in HPPUTS.
;
        .ENTRY  HPPUTS, ^M<>            ; ENTRY MASK
;
        BLBS    INIT_DONE, 20$          ; INIT DONE?
        BSBW    INIT_SUB                ; NO, DO INIT
        BLBS    R0, 20$
        MOVL    #-1, @8(AP)             ; INIT ERROR
        RET
;
20$:    MOVL    TT_PUT_STAT, @4(AP)     ; NUMBER OF STRINGS IN QUEUE
        MOVL    #1, @8(AP)              ; SUCCESS
        MOVL    #SS$_NORMAL, R0
        RET
;
;
```

```
;
        .END
```

```
* CONMESS.FOR    /WSP    /EUN/MRH
*
* WRITES MESSAGE ABOUT CONNECTION POINTS ASSOCIATED WITH A GIVEN
* POSITION IN "ADATA"

      SUBROUTINE CONMESS (NUMBER1, NUMBER2, MESSP)

      INCLUDE '[WSP.DEF]MESSB.FOR'
      INTEGER CODE, NUMBER1, NUMBER2, N, MESSP, P1, P2, P3, EOM
*****************************************************************************
        P1 = MESSP
        P3 = P1
        EOM = MIN (P3+79, 512)
        DO 15 N = NUMBER1, NUMBER2
          CODE = 1
          DO WHILE (CODE .GT. 0)
            IF (P1 .GT. 512-9) GOTO 789
            CALL NAMECON (N, P1, EOM-2, P2, CODE)
            IF (P2 .LE. P1 .AND. CODE .EQ. 0) GOTO 15
            IF (P2 .GT. EOM-9) THEN
              MESSAGE (P3:P3) = ';'
              CALL TIDY (MESSAGE, P3, P2-1, P3)
              P1 = P3
              EOM = MIN (P3+79, 512)
            ELSE
              P1 = P2
            ENDIF
          END DO
15      CONTINUE

789     CONTINUE
        IF (P3 .NE. P1) THEN
          MESSAGE (P3:P3) = ';'
          CALL TIDY (MESSAGE, P3, P2-1, P1)
        ENDIF
        MESSP = P1
        RETURN
      END
```

```
* CONNEC.FOR /WSP    1983-12-12/EUN/MRH
*
* SUBROUTINE THAT MOVES AND AMPLIFIES DATA ACCORDING TO CONTROL
* INFORMATION IN /CONNECB/:
*
*                     1  2  3  4  5  6  7  8  9 10 11 .....MAXCON
*                   ..................................................
* 5.  CONTROL 2     :  :  :  :  :  :  :  :  :  :  :  :        :  :
*                   :..:..:..:..:..:..:..:..:..:..:..:.....:..:
* 4.  CONTROL 1     :  :  :  :  :  :  :  :  :  :  :  :        :  :
*                   :..:..:..:..:..:..:..:..:..:..:..:.....:..:
* 3.  DESTINATION   :  :  :  :  :  :  :  :  :  :  :  :        :  :
*                   :..:..:..:..:..:..:..:..:..:..:..:.....:..:
* 2.  SOURCE        :  :  :  :  :  :  :  :  :  :  :  :        :  :
*                   :..:..:..:..:..:..:..:..:..:..:..:.....:..:
* 1.  TRIGNR        :  :  :  :  :  :  :  :  :  :  :  :        :  :
*                   :..:..:..:..:..:..:..:..:..:..:..:.....:..:
*
* TRIGGER POINTS TO A LOGICAL VARIABLE IN "TRIGS"
* IF 0, THIS CONNECTION IS PERMANENTLY OPEN, I.E. NOT TRIGGERED
* SOURCE, CONTROLS, AND DESTINATION POINT TO ELEMENTS IN "ADATA"
*******************************************************************************
      SUBROUTINE CONNEC

      INTEGER N

      INCLUDE '[WSP.DEF]CONNECB.FOR'
      INCLUDE '[WSP.DEF]DATAB.FOR'
      INCLUDE '[WSP.DEF]TRIGB.FOR'
*******************************************************************************
         DO N = 1, NCONS

           IF (CTRIGNR (N) .LE. 0) THEN
             ADATA (CDESTIN (N)) = ADATA (CSOURCE (N)) *
     -            ADATA (CCONTROL1 (N)) + ADATA (CCONTROL2 (N))

           ELSE IF (TRIGS (CTRIGNR (N))) THEN
             ADATA (CDESTIN (N)) = ADATA (CSOURCE (N)) *
     -            ADATA (CCONTROL1 (N)) + ADATA (CCONTROL2 (N))
             TRIGS (CTRIGNR (N)) = .FALSE.
           ENDIF
         END DO

         RETURN
*******************************************************************************
      END
```

```
* DISP.FOR    /WSP    1983-12-19    /EUN/MRH
*
* SUBROUTINE THAT DISPLAYS GRAPHICALLY THE CONTENTS OF SPECIFIED
* POSITIONS IN "ADATA"
*
* "DISPS" CONTAINS:
*                                1   2   3   4   ....   MAXT
*                              .....................
*                              :   :   :   :   :          :   :
* 4.   SKIP (INTEGER VALUE)    :...:...:...:...:........:...:
*                              :   :   :   :   :          :   :
* 3.   SIZE (INTEGER VALUE)    :...:...:...:...:........:...:
*                              :   :   :   :   :          :   :
* 2.   SOURCE (ADATA)          :...:...:...:...:........:...:
*                              :   :   :   :   :          :   :
* 1.   SWITCHNR (SWITCHES)     :...:...:...:...:........:...:

      SUBROUTINE DISP

      INCLUDE '[WSP.DEF]DATAB.FOR'
      INCLUDE '[WSP.DEF]DISPB.FOR'
      INCLUDE '[WSP.DEF]SWITCHB.FOR'

      INTEGER N
      LOGICAL DOT
*******************************************************************************
         DOT = .FALSE.
         DO 15 N = 1, NDISPS
           IF (SWITCH (DSWITNR (N)) .NE. 0) THEN
             DISPTEMP (N) = DISPTEMP (N) - 1
             IF (DISPTEMP (N) .LE. 0) THEN
               DISPTEMP (N) = DSKIP (N)
               CALL TVDOT
     -           (DISPX, INT (ADATA (DSOURCE (N)) * FLOAT (DSIZE (N))))
               DOT = .TRUE.
             ENDIF
           ENDIF
15       CONTINUE

         IF (DOT) THEN
           DISPX = DISPX + 1
           IF (DISPX .GT. MAXX) DISPX = 0
         ENDIF

         RETURN
      END
```

```
* FUNCGEN.FOR   /WSP   1983-12-23   /EUN/MRH
*
* SUBROUTINE THAT CALCULATES OUTPUT OF FUNCTION GENERATOR BOXES.
* "FUNCS" CONTAINS:
*
*  FTIMING (SWITCH)  HOW OFTEN GENERATOR IS TO BE UPDATED:
*      0 & 1 = EVERY SAMPLE, 2 = EVERY OTHER SAMPLE, 3 = EVERY THIRD, ETC
*  FTRIGOUT (IDATA)  ADDRESS IN IDATA OF A LIST OF TRIGGERS TO BE SET
*                    WHEN NEW SEGMENTS ARE STARTED
*  FHOLD (SWITCH)     IF ON, NO CHANGE TO OUTPUT (DEFAULT = OFF)
*  FINVERT (SWITCH)  IF ON, OUTPUT IS INVERTED (n = 1 - n) (DEFAULT = OFF)
*  FRAMP (SWITCH)  0 = NO RAMP, 1 = LINEAR RAMP, -1 = CURVED RAMP
*                         (DEFAULT = 0)
*  FSUSTAIN (SWITCH) (DEFAULT = OFF)
*  FSUSSEGNR (VALUE)  SEGMENT TO BE SUSTAINED IF FSUSTAIN ON (DEFAULT = -1)
*  FBREAK (TRIGGER)   TRIGGER TO SKIP TO END OF ENVELOPE
*  FTRIGNR (TRIGGER)  EXTERNAL TRIGGER TO START ENVELOPE
*  FTRIGGING (IDATA)  1=INTERNAL, -1=EXTERNAL, 0=BOTH (DEFAULT = BOTH)
*  FSEGS (VALUE)    NUMBER OF SEGMENTS IN THIS GENERATOR
*  FDESTIN (ADATA)   FOR GENERATOR OUTPUT
*  FSPEED (ADATA)     SPEED RATIO FOR TOTAL FUNCTION (DEFAULT = 1.0)
*  FCURVECON (ADATA)   CURVE CONTROLS (NSEGMENTS)
*  FGAIN (ADATA)       GAIN CONTROLS (NSEGMENTS + 1)
*  FCONTROLS (ADATA)   SEGMENT DURATION CONTROLS
*  FCURVES (ADATA)     CURVE DATA
*  FDURATION (ADATA)   DURATION OF EACH SEGMENT (SECONDS)
*  FSOURCE (ADATA)   BREAKPOINT VALUES (NSEGMENTS + 1)
*
* IN ADDITION:
*  FSEGP (I) NUMBER OF SEGMENT CURRENTLY IN OPERATION
*      -1 = INACTIVE
*      +n = ACTIVE IN THIS SEGMENT
*  FVAL (R) CURRENT OUTPUT VALUE
*  FNOW (R) CURRENT TIME WITHIN SEGMENT
*  FCOUNTER (I) INTERNAL COUNTER FOR TIMING
*
* DATA IS STORED IN "ADATA" AS FOLLOWS:
*
* :..:..:..:..:..:..:..:..:..:..:..:..:..:..:..:..:..:..:..:..:..:..:..:..:
*        ^        ^         ^         ^           ^          ^     ^  ^
*        |        |         |         |           |          |     |  |
* |_nsegs+1_| |_nsegs_| |_nsegs_| |_nsegs_| |_nsegs+1_| |_nsegs_|  |  |
* breakpoints   curves  durations  duration     gain      curve  speed|
*                                 controls   controls   controls  output
*
* IN "IDATA":
* :..:..:..:..:..:..
*       ^
* |___nsegs___|
*   triggers
*
*                  DURATION   BREAKPOINT
*         SPEED    CONTROLS    CONTROLS
*           |         |           |
*           v         v           v
*   ...................................................... SIGNAL
*   :                                                    : ---->
*   :            FUNCTION GENERATOR                      : BREAKPOINT TRIGGERS
*   :              nsegs SEGMENTS                        : ---->
*   :....................................................: ---->
*      ^   ^    ^       ^   ^       ^        ^       ^
```

```
*      |    |     |        |    |         |         |          |
*    HOLD   |  RAMP       | SUST.    INVERT    TRIGGING  TIMING
*   ON/OFF |OFF/LIN/EXP |ON/OFF    ON/OFF INT/EXT/BOTH
*          |             |
*        START          BREAK
*         TRIG           TRIG
*
* CONTROLS ARE TAKEN INTO ACCOUNT EVERY SAMPLE
*
* *** THE EXTERNAL TRIGGER ASSOCIATED WITH THE FUNCTION GENERATOR IS
* TURNED OFF AT EVERY STUDIO SAMPLE, EVEN IF IT HAS NOT BEEN USED.
* *** THE GENERATOR OUTPUTS NOTHING AT ALL WHEN "HOLD" IS ON AND
* WHEN IN SEGMENT -1


      SUBROUTINE FUNCGEN

      INCLUDE '[WSP.DEF]DATAB.FOR'
      INCLUDE '[WSP.DEF]FUNCB.FOR'
      INCLUDE '[WSP.DEF]STATB.FOR'
      INCLUDE '[WSP.DEF]SWITCHB.FOR'
      INCLUDE '[WSP.DEF]TRIGB.FOR'

      INTEGER N, POSO
      REAL CURVE, DUR, NEXTVAL, OLDVAL
*************************************************************************
         DO 25 N = 1, NFUNCS

          IF (SWITCH (FHOLD (N)) .EQ. 0) THEN
           IF ((SWITCH (FTRIGGING (N)) .LE. 0 .AND.
     -         TRIGS (FTRIGNR (N))) .OR.
     -            (FSEGP (N) .LT. 0 .AND.
     -               SWITCH (FTRIGGING (N)) .GE. 0)) THEN
             FSEGP (N) = 1
             FNOW (N) = 0.
           ENDIF

           IF (FSEGP (N) .GT. 0) THEN

             IF (TRIGS (FBREAK (N))) GOTO 17
15           CONTINUE
             POSO = FSEGP (N) - 1
             DUR = ADATA (FDURATION (N) + POSO)
             IF (FNOW (N) .GE. DUR) THEN

               IF (FSEGP (N) .EQ. FSUSSEGNR (N) .AND.
     -             SWITCH (FSUSTAIN (N)) .NE. 0) GOTO 25

               TRIGS (IDATA (FTRIGOUT (N) + POSO)) = .TRUE.
               IF (FSEGP (N) .GE. FSEGS (N)) THEN
17               CONTINUE
                 IF (SWITCH (FTRIGGING (N)) .GE. 0) THEN
                   FSEGP (N) = 1
                 ELSE
                   IF (SWITCH (FINVERT (N)) .NE. 0) THEN
                     ADATA (FDESTIN (N)) =
     -                  1. - ADATA (FSOURCE (N) + FSEGS (N))
                   ELSE
                     ADATA (FDESTIN (N)) =
     -                  ADATA (FSOURCE (N) + FSEGS (N))
                   ENDIF
```

```
                FSEGP (N) = -1
                GOTO 25
              ENDIF
            ELSE
              FSEGP (N) = FSEGP (N) + 1
            ENDIF
            FNOW (N) = 0.
            GOTO 15
          ELSE
            FNOW (N) = FNOW (N) + ADATA (FSPEED (N)) *
     -         ADATA (FCONTROLS (N) + POS0) * INVSAMPSPERSEC
            IF (FCOUNTER (N) .GE. SWITCH (FTIMING (N))) THEN
              FCOUNTER (N) = 1
              OLDVAL = ADATA (FSOURCE (N) + POS0) *
     -           ADATA (FGAIN (N) + POS0)
              NEXTVAL = ADATA (FSOURCE (N) + FSEGP (N)) *
     -           ADATA (FGAIN (N) + FSEGP (N))

              IF (FNOW (N) .GE. DUR .OR. NEXTVAL .EQ. OLDVAL) THEN
                FVAL (N) = NEXTVAL
              ELSE IF (SWITCH (FRAMP (N)) .GT. 0) THEN
                FVAL (N) =
     -             OLDVAL + (NEXTVAL - OLDVAL) * FNOW (N) / DUR
              ELSE IF (SWITCH (FRAMP (N)) .LT. 0) THEN
                CURVE =  ADATA (FCURVES (N) + POS0) *
     -                ADATA (FCURVECON (N) + POS0)
                IF (NEXTVAL .LT. OLDVAL) CURVE = -CURVE
                IF (CURVE .GE. 10.) THEN
                  FVAL (N) = NEXTVAL
                ELSE IF (CURVE .LE. -10.) THEN
                  FVAL (N) = OLDVAL
                ELSE
                  CURVE = 40. * CURVE / (10. - CURVE) / (10. - CURVE)
                  FVAL (N) = (1. + CURVE) * (NEXTVAL - OLDVAL) *
     -              FNOW (N) / (CURVE * FNOW (N) + DUR) + OLDVAL
                ENDIF
              ENDIF

              IF (SWITCH (FINVERT (N)) .NE. 0) THEN
                ADATA (FDESTIN (N)) = 1. - FVAL (N)
              ELSE
                ADATA (FDESTIN (N)) = FVAL (N)
              ENDIF
            ELSE
              FCOUNTER (N) = FCOUNTER (N) + 1
            ENDIF
          ENDIF
        ENDIF
      ENDIF
      TRIGS (FTRIGNR (N)) = .FALSE.
      TRIGS (FBREAK (N)) = .FALSE.
25    CONTINUE

      RETURN
      END
```

```
* GEN.FOR    /WSP    1984-01-02     /EUN/MRH
*
* GENERATOR BOX
* "GENS" CONTAINS:
*  AMPLITUDE (ADATA POSITION)
*  FREQUENCY (ADATA POSITION)

      SUBROUTINE GEN

      INCLUDE '[WSP.DEF]GENB.FOR'
      INCLUDE '[WSP.DEF]DATAB.FOR'
      INCLUDE '[WSP.DEF]STATB.FOR'
      INCLUDE '[WSP.DEF]SYNTB.FOR'

      INTEGER N
      REAL NEXT
*****************************************************************************
          DO N = 1, NGENS
            IF (NSYS .GT. PSYNTS-8) THEN

*    RECORD FULL

            ELSE
              NEXT = MIN (MAXFRE, MAX (0., ADATA (GFREQ (N))))
              IF (NEXT .NE. GOLDFRE (N)) THEN
                SYNTS (NSYS+1) = FICODE * 256 + N
                SYNTS (NSYS+2) = ABS ((NEXT - GOLDFRE (N)) * FICONST /
     -                SAMPLENGTH)
                SYNTS (NSYS+3) = FRCODE * 256 + N
                SYNTS (NSYS+4) = INT (NEXT * FRCONST)
                GOLDFRE (N) = NEXT
                NSYS = NSYS + 4
              ENDIF

              NEXT = ADATA (GAMP (N))
              IF (NEXT .NE. GOLDAMP (N)) THEN
                SYNTS (NSYS+1) = AICODE * 256 + N
                SYNTS (NSYS+2) = ABS ((NEXT - GOLDAMP (N)) * AICONST /
     -                SAMPLENGTH)
                SYNTS (NSYS+3) = ACODE * 256 + N
                SYNTS (NSYS+4) = INT (NEXT * AMCONST)
                GOLDAMP (N) = NEXT
                NSYS = NSYS + 4
              ENDIF
            ENDIF
          END DO

          RETURN
        END
```

```
* MIX.FOR    /WSP    1984-01-04    /EUN/MRH
*
* SUBROUTINE BOX THAT ADDS AN ARBITRARY NUMBER OF SIGNALS, EACH WITH
* ITS OWN OPTIONAL GAIN CONTROL, TO A SINGLE OUTPUT POSITION
*
*                        n CONTROLS
*                         |   |   |
*                         V   V   V
*                      .............
*             ------> :             :
* n SIGNALS   ------> :       +     : ---->
*             ------> :             :
*                     :.............:
*
* SIGNALS, CONTROLS IN "ADATA" AS FOLLOWS:
*
*        :...:...:...:...:...:...:...:...:...:...:...:...:.......
*                 ^               ^           ^
*                 |               |           |
*        |_____ n _____| |_____ n _____| output
*              inputs          controls
```

```
      SUBROUTINE MIX

      INCLUDE '[WSP.DEF]DATAB.FOR'
      INCLUDE '[WSP.DEF]MIXB.FOR'

      INTEGER N, N1
      REAL SUM
*****************************************************************************
         DO N = 1, NMIX
           SUM = 0.
           DO N1 = 0, MINPUTS (N) - 1
             SUM = SUM +
     -           ADATA (MSOURCE (N) + N1) * ADATA (MCONTROLS (N) + N1)
           END DO
           ADATA (MDESTIN (N)) = SUM
         END DO

         RETURN
       END
```

```
* LIST.FOR    /WSP    1983-12-13/EUN/MRH
*
* USERNAME LISTING FOR WSP

      SUBROUTINE LIST (BOXNR, START, NDEVS, NAMES, MESSP, CODE)

      INTEGER P1, P2, P3, SAVEPOS /1/, SAVEPAR /1/,
     - BOXNR, NDEVS, MESSP, CODE, START
      CHARACTER*(*) NAMES (*), CR
      PARAMETER (CR=CHAR(13)//CHAR(10))

      INCLUDE '[WSP.DEF]BOXB.FOR'
      INCLUDE '[WSP.DEF]MESSB.FOR'
      SAVE SAVEPOS, SAVEPAR
*****************************************************************************
       IF (NDEVS .GE. START) THEN
         P1 = MESSP
         IF (SAVEPOS .LE. START) THEN
           MESSAGE (P1:P1+11) = '*** '//BOXTYPES (BOXNR)//CR
           P1 = P1 + 12
           SAVEPOS = START
         ENDIF
         P2 = 1
         DO WHILE (SAVEPOS .LE. NDEVS)
           MESSAGE (P1:P1+6) = ' '//NAMES (SAVEPOS)
           P1 = P1 + 7
           P2 = P2 + 7
           SAVEPOS = SAVEPOS + 1
           IF (P2 .GT. 80-6) THEN
             MESSAGE (P1:P1+1) = CR
             P1 = P1 + 2
             IF (P1 .GT. 512-83) GOTO 785
             P2 = 1
           ENDIF
         END DO

         IF (P2 .GT. 1) THEN
           MESSAGE (P1:P1+1) = CR
           P1 = P1 + 2
         ENDIF
         SAVEPOS = 1
         GOTO 789
*****************************************************************************
785      CONTINUE
         CODE = 1                      !still more to write out
         GOTO 791

789      CONTINUE
         CODE = 0                      !all complete
         GOTO 791

791      CONTINUE
         MESSP = P1
       ELSE
         CODE = -1                     !no devices, so nothing done
       ENDIF
       RETURN

      END
```

```
* QUANT.FOR    /WSP    /EUN/MRH
*
* QUANTIFIER BOX

       SUBROUTINE QUANT

       INCLUDE '[WSP.DEF]DATAB.FOR'
       INCLUDE '[WSP.DEF]QUANTB.FOR'
       INTEGER N, P
*******************************************************************************
          DO N = 1, NQUANTS
            P = MAX (0,
      -            MIN (INT (ADATA (QSOURCE (N)) * FLOAT (QSAVES (N))),
      -                 QSAVES (N) - 1))
            ADATA (QDESTIN (N)) =
      -       ADATA (QDATA (N) + P) * ADATA (QCONTROL (N) + P)
          END DO
          RETURN
        END
```

```
* RANDOM.FOR /WSP    1983-12-13/EUN/MRH
*
* RANDOM NUMBER BOX

      SUBROUTINE RANDOM

      INCLUDE '[WSP.DEF]RANB.FOR'
      INCLUDE '[WSP.DEF]DATAB.FOR'
      INCLUDE '[WSP.DEF]STATB.FOR'
      INCLUDE '[WSP.DEF]SWITCHB.FOR'
      INCLUDE '[WSP.DEF]TRIGB.FOR'

      INTEGER N
      REAL RANSEL
*****************************************************************************
         DO N = 1, NRANS
           IF (RTRIGNR (N) .LE. 0 .OR. TRIGS (RTRIGNR (N))) THEN
             IF (SWITCH (RSPEED (N)) .NE. 0) THEN
               ROLD (N) = ROLD (N) + RDIFF (N) * ADATA (RCONTROL1 (N))
               IF ((RUP (N) .AND. ROLD (N) .GE. RNEXT (N)) .OR.
     -              (.NOT. RUP (N) .AND. ROLD (N) .LE. RNEXT (N))) THEN
                 ADATA (RDESTIN (N)) = RNEXT (N)
                 ROLD (N) = RNEXT (N)
                 RNEXT (N) = RANSEL (SWITCH (RDIST (N)),
     -                RCONTROL2 (N), RCONSWITCH (N), RSEED (N))
                 RDIFF (N) = RNEXT (N) - ROLD (N)
                 RUP (N) = RNEXT (N) .GE. ROLD (N)
               ELSE
                 ADATA (RDESTIN (N)) = ROLD (N)
               ENDIF
             ELSE
               ROLD (N) = RNEXT (N)
               RDIFF (N) = 0.
               ADATA (RDESTIN (N)) = RANSEL (SWITCH (RDIST (N)),
     -                RCONTROL2 (N), RCONSWITCH (N), RSEED (N))
             ENDIF
           ENDIF
           TRIGS (RTRIGNR (N)) = .FALSE.
         END DO
         RETURN
       END
```

```
* RANSEL.FOR /WSP    /EUN/MRH
*
* RANDOM CALCULATIONS FOR RANDOM BOX

      REAL FUNCTION RANSEL (TYPE, RCONTROL, SCONTROL, SEED)

      INTEGER INORSW /0/, N1, TYPE, SCONTROL, SEED
      REAL A /0./, RCONTROL, Z1, Z2

      SAVE A, INORSW, Z2
*******************************************************************************
* EXPRAN:  X IS EXPONENTIALLY DISTRIBUTED WITH 1.0 AS MEAN

        IF (TYPE .EQ. -1) THEN
          A = -LOG (MAX (RAN (SEED), 1.0E-38))

* RECTANGULAR DISTRIBUTION

        ELSE IF (TYPE .EQ. 0) THEN
          A = RAN (SEED)

* ERLANG DISTRIBUTION: THE DISTRIBUTION OF THE SUM OF RCONSWITCH
* EXPONENTIAL VARIABLES

        ELSE IF (TYPE .EQ. 1) THEN
          A = 1.0
          IF (RCONTROL .NE. 0.) THEN
            DO N1 = 1, SCONTROL
              A = A * RAN (SEED)
            END DO
            A = -LOG (A) / RCONTROL
          ENDIF

* NORMAL DISTRIBUTION: GENERATES A RANDOM NUMBER THAT HAS A NORMAL
* DISTRIBUTION WITH ZERO MEAN AND STANDARD DEVIATION 1
* THE POLAR METHOD IS USED TO GENERATE NORMAL VARIATES

        ELSE IF (TYPE .EQ. 2) THEN
          IF(INORSW.EQ.0) THEN
            A = 1.
            DO WHILE (A .GE. 1.)
              Z1 = 2. * RAN (SEED) - 1.
              Z2 = 2. * RAN (SEED) - 1.
* Z1 AND Z2 NOW REC. DIST. BETWEEN -1. AND +1.
              A = Z1*Z1 + Z2*Z2
            END DO
            A = SQRT (-2. * LOG (A) / A)
            Z2 = Z2 * A
            A = Z1 * A
            INORSW = 1
          ELSE
            A = Z2
            INORSW = 0
          ENDIF

* POISSON DISTRIBUTED VARIATES N1 WITH MEAN RCONTROL ARE PRODUCED

        ELSE IF (TYPE .EQ. 3) THEN
          N1 = 0
          Z1 = EXP (-RCONTROL)
          A = 1.
```

```
            DO WHILE (A .GE. Z1)
              A = A * RAN (SEED)
              N1 = N1 + 1
            END DO
            A = FLOAT (N1)
          ELSE
            A = 0.
          ENDIF

          RANSEL = A
          RETURN
*****************************************************************************
        END
```

```
* TDELAY.FOR    /WSP    1984-01-03    /EUN/MRH
*
* SUBROUTINE THAT CALCULATES TRIGGER DELAY BOXES
*
*                 .................
*    ------------> :                 : ------------>
*      TRIGGER     :                 :   TRIGGER
*                  :.................:
*                           ^
*                           |
*                     DELAY (SECS)
*
*   TDELSAVES (VALUE): ACTUAL NUMBER OF TRIGS TO BE SAVED BEFORE THEY START
*                      OVERRIDING ONE ANOTHER
*   TDELSAVESX (VALUE): MAXIMUM NUMBER OF TRIGS TO BE SAVED BEFORE THEY START
*                      OVERRIDING ONE ANOTHER
*   TDELTIMES (ADATA): COUNTERS FOR SAVED TRIGS
*   TDELACTIVE (LDATA): INDICATORS FOR EACH TRIG SAVED - .TRUE. = ACTIVE,
*                      .FALSE. = INACTIVE
*   DTIME (ADATA): CONTROL TIME IN SECONDS BETWEEN RECEIVING TRIGGER &
*                     SETTING OUTPUT TRIGGER
*   DTRIGOUT (TRIGS): OUTPUT TRIGGER
*   DTRIGIN (TRIGS): INPUT TRIGGER

      SUBROUTINE TDELAY

      INCLUDE '[WSP.DEF]DATAB.FOR'
      INCLUDE '[WSP.DEF]TDELAYB.FOR'
      INCLUDE '[WSP.DEF]TRIGB.FOR'
      INCLUDE '[WSP.DEF]STATB.FOR'

      INTEGER N, N1, P1
*****************************************************************************
        DO N = 1, NTDELS
          IF (TRIGS (DTRIGIN (N))) THEN
            IF (TDELP (N) .GE. TDELSAVES (N)) TDELP (N) = 0
            ADATA (TDELTIMES (N) + TDELP (N)) = - INVSAMPSPERSEC
            LDATA (TDELACTIVE (N) + TDELP (N)) = .TRUE.
            TDELP (N) = TDELP (N) + 1
            TRIGS (DTRIGIN (N)) = .FALSE.
          ENDIF

          P1 = 0
          DO N1 = TDELTIMES (N), TDELTIMES (N) + TDELSAVES (N) - 1
            IF (LDATA (TDELACTIVE (N) + P1)) THEN
              ADATA (N1) = ADATA (N1) + INVSAMPSPERSEC
              IF (ADATA (N1) .GE. ADATA (DTIME (N))) THEN
                TRIGS (DTRIGOUT (N)) = .TRUE.
                LDATA (TDELACTIVE (N) + P1) = .FALSE.
              ENDIF
            ENDIF
            P1 = P1 + 1
          END DO
        END DO
        RETURN
      END
```

```
* TDIVIDE.FOR    /WSP    /EUN/MRH
*
* TRIG-DIVIDE BOX.  ONE TRIGGER INPUT IS OUTPUT SIMULTANEOUSLY TO
* SEVERAL SPECIFIED TRIG OUTPUTS.
*
* IN /TDIVIDEB/
*       NTDIVOUTS (PTDIVS)
*       NTDIVOUTSX (PTDIVS)
*       TDIVDESTIN (PTDIVS)
*       TDIVSOURCE (PTDIVS)
*       TDIVNAMES (PTDIVS)
*       NTDIVS

      SUBROUTINE TDIVIDE

      INCLUDE '[WSP.DEF]DATAB.FOR'
      INCLUDE '[WSP.DEF]TDIVIDEB.FOR'
      INCLUDE '[WSP.DEF]TRIGB.FOR'

      INTEGER N, N1
*****************************************************************************
         DO N = 1, NTDIVS
           IF (TRIGS (TDIVSOURCE (N))) THEN
             TRIGS (TDIVSOURCE (N)) = .FALSE.
             DO N1 = TDIVDESTIN (N), TDIVDESTIN (N) + NTDIVOUTS (N) - 1
               TRIGS (IDATA (N1)) = .TRUE.
             END DO
           ENDIF
         END DO

         RETURN
      END
```

```
* TIDY.FOR   /WSP   /EUN/MRH
*
* SUBROUTINE THAT TIDIES WSP OUTPUT LINES, BY REMOVING UNNECESSARY
* SPACES AND COMMAS, DELETING TRAILING ZEROES IN FLOATING-POINT
* NUMBERS, AND ADDING CARRIAGE RETURN AT END
* USE:
*         CALL TIDY (TEXT, FROM, TO, NEWTO)
*
* TEXT - CHARACTER STRING - UPDATED BY THIS CALL
* FROM - INTEGER - START POSITION IN 'TEXT'
* TO - INTEGER - END POSITION IN 'TEXT'
* NEWTO - INTEGER - RECEIVES NEW END POSITION + 1 (I.E. POSITION FOR
*           NEXT CHARACTER)

      SUBROUTINE TIDY (TEXT, FROM, TO, NEWTO)

      CHARACTER*(*) TEXT
      INTEGER FROM, TO, NEWTO, P1, P2, ZEROES
      LOGICAL POINT, SPACE
*************************************************************************
         P1 = TO
         DOWHILE (P1 .GE. FROM .AND.
     -        (TEXT (P1:P1) .EQ. ' ' .OR. TEXT (P1:P1) .EQ. ','))
           P1 = P1 -1
         END DO

         ZEROES = 0
         POINT = .FALSE.
         SPACE = .TRUE.
         P2 = FROM
         DO 15 P3 = FROM, P1
           IF (TEXT (P3:P3) .EQ. ' ') THEN
             IF (SPACE) GOTO 15
             SPACE = .TRUE.
             IF (ZEROES .GE. 2) P2 = P2 - ZEROES
             ZEROES = 0
             POINT = .FALSE.
           ELSE IF (TEXT (P3:P3) .EQ. ',') THEN
             SPACE = .TRUE.
             IF (ZEROES .GE. 2) P2 = P2 - ZEROES
             ZEROES = 0
             POINT = .FALSE.
           ELSE IF (TEXT (P3:P3) .EQ. '.') THEN
             SPACE = .FALSE.
             POINT = .TRUE.
           ELSE IF (POINT) THEN
             IF (TEXT (P3:P3) .EQ. '0') THEN
               ZEROES = ZEROES + 1
             ELSE
               ZEROES = 0
             ENDIF
             SPACE = .FALSE.
           ELSE
             POINT = .FALSE.
             SPACE = .FALSE.
             IF (ZEROES .GE. 2) P2 = P2 - ZEROES
             ZEROES = 0
           ENDIF
           TEXT (P2:P2) = TEXT (P3:P3)
           P2 = P2 + 1
15       CONTINUE
```

```
    IF (ZEROES .GE. 2) P2 = P2 - ZEROES
    TEXT (P2:P2+1) = CHAR (10) // CHAR (13)
    NEWTO = P2 + 2
    RETURN
END
```

```
* TSELEC.FOR    /WSP    /EUN/MRH
*
* TRIGGER SELECTOR BOX
* ____________________
*
* WHEN A TRIG PULSE IS INPUT, ONE OF AN ARBITRARY NUMBER OF SPECIFIED
* OUTPUT TRIGGERS IS SET.  THE NUMBERS OF THE OUTPUT TRIGGERS ARE STORED
* IN "IDATA", STARTING AT POSITION "TSELDESTIN ()" AND OCCUPYING
* "TSELNDEST()" POSITIONS.  THE OUTPUT TRIGGER IS DETERMINED BY ONE
* OF THREE METHODS (SPECIFIED WITH -1, 0 OR 1 IN SWITCH TSELSW):
*
*     1. CIRCULAR
*     ___________
*                              TRIGn      TRIG1
*                                   x     x   TRIG2
*                                 x           x  TRIG3
*                               x                x etc
*                     ^       x                  x
*                     ¦                                ¦
*                     ¦       x                  x     ¦
*                               x                x     v
*                                 x            x
*                                     x     x
*
*     2. "VOLTAGE" CONTROLLED
*     _______________________
*
*                                 ¦
*                          CONTROL ¦
*                          (0 - 1) V
*                    1.0 .........................      ----> TRIGn
*                        :                         :     ¦---->   .
*                        :                         :     ¦---->   .
*          TRIG  -------> :                         : -----¦---->   .
*          INPUT          :                         :     ¦----> TRIG3
*                         :                         :     ¦----> TRIG2
*                    0.0 :.........................:      ----> TRIG1
*
*     3. "SWITCH" CONTROLLED
*     ______________________
*
*                            SW    ¦
*                          CONTROL ¦
*                          (1 - n) V
*                      n .........................      ----> TRIGn
*                        :                         :     ¦---->   .
*                        :                         :     ¦---->   .
*          TRIG  -------> :                         : -----¦---->   .
*          INPUT          :                         :     ¦----> TRIG3
*                         :                         :     ¦----> TRIG2
*                      1 :.........................:      ----> TRIG1
*
```

```
      SUBROUTINE TSELEC

      INCLUDE '[WSP.DEF]DATAB.FOR'
      INCLUDE '[WSP.DEF]SWITCHB.FOR'
      INCLUDE '[WSP.DEF]TSELECB.FOR'
      INCLUDE '[WSP.DEF]TRIGB.FOR'

      INTEGER N, P1
*****************************************************************************
         DO N = 1, NTSELS
           IF (TRIGS (TSELSOURCE (N))) THEN
             TRIGS (TSELSOURCE (N)) = .FALSE.
             IF (SWITCH (TSELSW (N)) .LT. 0) THEN
               IF (TSELP (N) .GE. TSELNDEST (N)) TSELP (N) = 0
               P1 = TSELP (N)
               TSELP (N) = TSELP (N) + 1
             ELSE IF (SWITCH (TSELSW (N)) .EQ. 0) THEN
               P1 = MIN (MAX (INT (ADATA (TSELSIGCON (N)) *
     -              FLOAT (TSELNDEST (N))), 0), TSELNDEST (N) - 1)
             ELSE
               P1 =
     -           MIN (MAX (SWITCH (TSELSWCON (N)), 1), TSELNDEST(N)) - 1
             ENDIF
             TRIGS (IDATA (TSELDESTIN (N) + P1)) = .TRUE.
           ENDIF
         END DO

         RETURN
       END
```

```
* USER.FOR      /WSP    /EUN/MRH
*
* USER BOX, FOR USER'S TO DEFINE THEIR OWN BOX-TYPES & ALGORITHMS
* /USERB/ CONTAINS:
*  NUSERS: NUMBER OF 'USER' BOXES IN USE
*  NUDATA ()        NUMBER OF FLOATING-POINT DATA ITEMS IN "ADATA"
*  UDATA ()         POINTER TO DATA ITEMS IN "ADATA"
*  NUSIGIN ()       ACTUAL NUMBER OF 'VOLTAGE' INPUTS
*  NUSIGINX ()      NUMBER OF 'VOLTAGE' INPUTS
*  USIGIN ()        POINTER TO 'VOLTAGE' INPUT ADDRESSES
*  NUTRIGIN ()      ACTUAL NUMBER OF TRIG INPUTS
*  NUTRIGINX ()     NUMBER OF TRIG INPUTS
*  UTRIGIN ()       POINTER TO TRIG INPUT ADDRESSES
*  NUSWIN ()        ACTUAL NUMBER OF SWITCH INPUTS
*  NUSWINX ()       NUMBER OF SWITCH INPUTS
*  USWIN ()         POINTER TO SWITCH INPUT ADDRESSES
*  NUSIGOUT ()      ACTUAL NUMBER OF 'VOLTAGE' OUTPUTS
*  NUSIGOUTX ()     NUMBER OF 'VOLTAGE' OUTPUTS
*  USIGOUT ()       POINTER TO 'VOLTAGE' OUTPUT ADDRESSES
*  NUTRIGOUT ()     ACTUAL NUMBER OF TRIG OUTPUTS
*  NUTRIGOUTX ()    NUMBER OF TRIG OUTPUTS
*  UTRIGOUT ()      POINTER TO TRIG OUTPUT ADDRESSES
*  NUSWOUT ()       ACTUAL NUMBER OF SWITCH OUTPUTS
*  NUSWOUTX ()      NUMBER OF SWITCH OUTPUTS
*  USWOUT ()        POINTER TO SWITCH OUTPUT ADDRESSES

      SUBROUTINE USER

      INCLUDE '[WSP.DEF]DATAB.FOR'
      INCLUDE '[WSP.DEF]STATB.FOR'
      INCLUDE '[WSP.DEF]SWITCHB.FOR'
      INCLUDE '[WSP.DEF]TRIGB.FOR'
      INCLUDE '[WSP.DEF]USERB.FOR'
*****************************************************************************
         DO N = 1, NUSERS
         END DO
         RETURN
      END
```

```
* VALUE.FOR    /WSP    /EUN/MRH
*
* VALUE BOX

      SUBROUTINE VALUE

      INTEGER N
      INCLUDE '[WSP.DEF]DATAB.FOR'
      INCLUDE '[WSP.DEF]VALB.FOR'


         DO N = 1, NVALS
           ADATA (VALS (N) + 1) = ADATA (VALS (N))
         END DO
         RETURN
      END
```

```
* BOXB.FOR    /WSP    /EUN/MRH
*
       INTEGER PBOXTYPES
       PARAMETER (PBOXTYPES=20)
       CHARACTER*6 BOXTYPES (PBOXTYPES)
      - /'GENERA', 'TDELAY', 'RANDOM', '      ', 'USER  ', 'SDELAY',
      -  'SWITCH', 'PORTAM', 'QUANTI', '      ', 'CONNEC', 'TRIGGE',
      -  'DISPLA', 'VALUE ', 'FUNCTI', '      ', 'MIX   ', 'TSELEC',
      -  'TDIVID', '      '/

       COMMON /BOXB/ BOXTYPES
```

alter -

Switch

this

val

grant .

```
* COMMANDB.FOR

      INTEGER NCHARS, FIELD, ENDFIELD, NFIELDS
      LOGICAL T_COMMAND_RECEIVED, D_COMMAND_RECEIVED
      CHARACTER TERMINAL_COMMAND*80

      COMMON /COMMANDB/ T_COMMAND_RECEIVED, D_COMMAND_RECEIVED,
     - TERMINAL_COMMAND, NCHARS, FIELD (80), ENDFIELD (80), NFIELDS
```

```
* CONNECB.FOR

      INTEGER NCONS, PCONS
      INTEGER CTRIGNR, CSOURCE, CDESTIN, CCONTROL1, CCONTROL2
      PARAMETER (PCONS=100)
      CHARACTER*6 CONNAMES (PCONS)

      COMMON /CONNECB/ NCONS, CTRIGNR (PCONS),
     - CSOURCE (PCONS), CDESTIN (PCONS), CCONTROL1 (PCONS),
     - CCONTROL2 (PCONS), CONNAMES
```

```
* DATAB.FOR

      INTEGER IDATA, NDATA, NDATI, NDATL, PDATA, PDATI, PDATL
      LOGICAL LDATA
      REAL ADATA
      PARAMETER (PDATA=1000, PDATI=100, PDATL=100)
      COMMON /DATAB/ NDATA, ADATA (-1:PDATA), NDATL, LDATA (0:PDATL),
     - NDATI, IDATA (0:PDATI)
```

```
* DISPB.FOR

      INTEGER PDISPS, NDISPS, DISPX, DISPTEMP
      INTEGER DSWITNR, DSOURCE, DSIZE, DSKIP,  MAXX
      PARAMETER (PDISPS=10, MAXX=1023)
      CHARACTER*6 DISPNAMES (PDISPS)

      COMMON /DISPB/ NDISPS, DSWITNR (PDISPS),
     - DSOURCE (PDISPS), DSIZE (PDISPS), DSKIP (PDISPS), DISPNAMES,
     - DISPX, DISPTEMP (PDISPS)
```

```
* FUNCB.FOR

      INTEGER NFUNCS, PFUNCS
      INTEGER FSEGP, FSOURCE, FDURATION, FCONTROLS, FGAIN, FDESTIN,
     - FSEGS, FSEGSX, FTRIGGING, FTRIGNR, FBREAK, FSUSSEGNR, FSUSTAIN,
     - FRAMP, FINVERT, FHOLD, FSPEED, FCURVES, FTRIGOUT, FTIMING,
     - FCURVECON
      REAL FNOW, FVAL
      PARAMETER (PFUNCS=10)
      CHARACTER*6 FUNCNAMES (PFUNCS)

      COMMON /FUNCB/ NFUNCS, FSOURCE (PFUNCS), FDURATION (PFUNCS),
     - FCURVES (PFUNCS), FCONTROLS (PFUNCS), FGAIN (PFUNCS),
     - FCURVECON (PFUNCS), FDESTIN (PFUNCS), FSEGS (PFUNCS),
     - FSEGSX (PFUNCS), FTRIGGING (PFUNCS), FTRIGNR (PFUNCS),
     - FBREAK (PFUNCS), FSUSSEGNR (PFUNCS), FSUSTAIN (PFUNCS),
     - FRAMP (PFUNCS), FINVERT (PFUNCS), FHOLD (PFUNCS),
     - FTRIGOUT (PFUNCS), FTIMING (PFUNCS), FSPEED (PFUNCS),
     - FUNCNAMES, FNOW (PFUNCS), FVAL (PFUNCS), FSEGP (PFUNCS),
     - FCOUNTER (PFUNCS)
```

```
* GENB

      INTEGER PGENS, MAXG, NGENS
      INTEGER GFREQ, GAMP
      REAL GOLDAMP, GOLDFRE

      PARAMETER (PGENS=256)
      CHARACTER*6 GENNAMES (PGENS)

      COMMON /GENB/ MAXG, NGENS, GFREQ (PGENS), GAMP (PGENS), GENNAMES,
     - GOLDAMP (PGENS), GOLDFRE (PGENS)
```

```
* MESSB.FOR

      INTEGER MESSCHARS
      LOGICAL TTMESS
      CHARACTER*512 MESSAGE

      COMMON /MESSB/ TTMESS, MESSCHARS, MESSAGE
```

```
* MIXB.FOR

      INTEGER PMIX, MINPUTS, MINPUTSX, MSOURCE, MCONTROLS, MDESTIN, NMIX
      PARAMETER (PMIX=10)
      CHARACTER*6 MIXNAMES (PMIX)

      COMMON /MIXB/ NMIX, MINPUTS (PMIX), MINPUTSX (PMIX),
     - MSOURCE (PMIX), MCONTROLS (PMIX), MDESTIN (PMIX), MIXNAMES
```

```
* QUANTB.FOR

      INTEGER NQUANTS, PQUANTS, QSAVES, QSAVESX, QSOURCE, QDESTIN,
     - QDATA, QCONTROL

      PARAMETER (PQUANTS=20)
      CHARACTER*6 QUANTNAMES (PQUANTS)

      COMMON /QUANTB/ NQUANTS, QCONTROL (PQUANTS), QSOURCE (PQUANTS),
     - QDESTIN (PQUANTS), QSAVES (PQUANTS), QSAVESX (PQUANTS),
     - QDATA (PQUANTS), QUANTNAMES
```

```
* RANB.FOR

      INTEGER NRANS, PRANS, RSPEED, RDIST, RSEED
      INTEGER RTRIGNR, RCONTROL1, RCONTROL2, RCONSWITCH, RDESTIN
      LOGICAL RUP
      REAL RDIFF, RNEXT, ROLD

      PARAMETER (PRANS=20)
      CHARACTER*6 RANNAMES (PRANS)

      COMMON /RANB/ NRANS, RSPEED (PRANS), RDIST (PRANS),
     - RTRIGNR (PRANS), RCONTROL1 (PRANS), RCONTROL2 (PRANS),
     - RCONSWITCH (PRANS), RDESTIN (PRANS),
     - RSEED (PRANS), RANNAMES, RUP (PRANS), RDIFF (PRANS),
     - RNEXT (PRANS), ROLD (PRANS)
```

```
* SEQB.FOR

      INTEGER PSEQS, NSEQS
      INTEGER STYPE, STRIGNR, SCONTROL1, SDESTIN
      PARAMETER (PSEQS=1)
      CHARACTER*6 SEQNAMES (PSEQS)

      COMMON /SEQB/ NSEQS, STYPE (PSEQS), STRIGNR (PSEQS),
     - SCONTROL1 (PSEQS), SDESTIN (PSEQS), SEQNAMES
```

```
* STATB.FOR    /WSP    1983-12-14/EUN/MRH

      INTEGER STATUS, WSPUNIT
      LOGICAL DOCALC, DOSYNT, WSPFILEOPEN, WSPREAD
      REAL MAXFRE, SAMPLENGTH, SAMPRATE, SAMPSPERSEC, INVSAMPSPERSEC
      CHARACTER*20 WSPFILE, VERSION*6

      COMMON /STATB/ STATUS, DOCALC, DOSYNT, MAXFRE, SAMPLENGTH,
     - SAMPRATE, SAMPSPERSEC, INVSAMPSPERSEC, WSPUNIT, WSPFILEOPEN,
     - WSPREAD, WSPFILE, VERSION
```

```
* SWITCHB.FOR

      INTEGER NSWITCH, PSWITCH, SWITCH
      PARAMETER (PSWITCH=100)
      CHARACTER*6 SWITCHNAMES (PSWITCH)

      COMMON /SWITCHB/ NSWITCH, SWITCH (0:PSWITCH), SWITCHNAMES
```

```
* SYNTB.FOR    /WSP
*
       INTEGER PSYNTS, FRCODE, FICODE, PHCODE, ACODE, AICODE,
      - DESTCODE, AMPCODE, FMPCODE, GAINCODE, SRCODE,
      - O1CODE, OI1CODE, O2CODE, OI2CODE, O3CODE, OI3CODE,
      - O4CODE, OI4CODE, NSYS, SYNTS
       REAL FRCONST, FICONST, AMCONST, AICONST, PHCONST
       PARAMETER (PSYNTS=256*2*6, FRCODE=8, FICODE=4, PHCODE=12,
      - ACODE=20, AICODE=16, DESTCODE=24, AMPCODE=28, FMPCODE=32,
      - GAINCODE=36, SRCODE=255, O1CODE=44, OI1CODE=40, O2CODE=52,
      - OI2CODE=48, O3CODE=60, OI3CODE=56, O4CODE=68, OI4CODE=64)

       COMMON /SYNTB/ NSYS, SYNTS (PSYNTS),
      - FRCONST, FICONST, AMCONST, AICONST, PHCONST
```

```
* TDELAYB.FOR

      INTEGER PTDELS, NTDELS, TDELSAVES, TDELSAVESX, DTRIGIN, DTRIGOUT,
     -  DTIME, TDELTIMES, TDELACTIVE, TDELP
      PARAMETER (PTDELS=20)
      CHARACTER*6 TDELNAMES (PTDELS)

      COMMON /TDELAYB/ NTDELS, TDELSAVES (PTDELS), TDELSAVESX (PTDELS),
     - DTRIGIN (PTDELS), DTRIGOUT (PTDELS), DTIME (PTDELS),
     - TDELTIMES (PTDELS), TDELACTIVE (PTDELS), TDELP (PTDELS),
     - TDELNAMES
```

```
* TDIVIDEB.FOR    /WSP    /EUN/MRH

       INTEGER PTDIVS, NTDIVS, TDIVSOURCE, TDIVDESTIN, NTDIVOUTS,
      - NTDIVOUTSX
       PARAMETER (PTDIVS=10)
       CHARACTER*6 TDIVNAMES (PTDIVS)

       COMMON /TDIVIDEB/ NTDIVS, TDIVSOURCE (PTDIVS),
      - TDIVDESTIN (PTDIVS), NTDIVOUTS (PTDIVS), NTDIVOUTSX (PTDIVS),
      - TDIVNAMES
```

```
* TRIGB.FOR

      INTEGER NTRIGS, PTRIGS
      PARAMETER (PTRIGS=100)
      LOGICAL TRIGS
      CHARACTER*6 TRIGNAMES (PTRIGS)

      COMMON /TRIGB/ NTRIGS, TRIGS (-1:PTRIGS), TRIGNAMES
```

```
* TSELECB.FOR    /WSP

      INTEGER PTSEL, TSELSW, TSELSOURCE, TSELNDEST, TSELNDESTX,
     - TSELDESTIN, TSELSIGCON, TSELSWCON, TSELP, NTSELS
      PARAMETER (PTSEL=10)
      CHARACTER*6 TSELNAMES (PTSEL)

      COMMON /TSELECB/ NTSELS, TSELSW (PTSEL), TSELSOURCE (PTSEL),
     - TSELNDEST (PTSEL), TSELNDESTX (PTSEL), TSELDESTIN (PTSEL),
     - TSELSIGCON (PTSEL), TSELSWCON (PTSEL), TSELP (PTSEL), TSELNAMES
```

```
* USERB.FOR      /WSP    /EUN/MRH

       INTEGER PUSER, NUSERS, NUDATA, NUDATAX, UDATA, NUSIGIN,
      - NUSIGINX, USIGIN, NUTRIGIN, NUTRIGINX, UTRIGIN, NUSWIN,
      - NUSWINX, USWIN, NUSIGOUT, NUSIGOUTX, USIGOUT, NUTRIGOUT,
      - NUTRIGOUTX, UTRIGOUT, NUSWOUT, NUSWOUTX, USWOUT
       PARAMETER (PUSER=10)
       CHARACTER*6 UNAMES (PUSER)

       COMMON /USERB/ NUSERS, NUDATA (PUSER), NUDATAX (PUSER),
      - UDATA (PUSER), NUSIGIN (PUSER), NUSIGINX (PUSER),
      - USIGIN (PUSER), NUTRIGIN (PUSER), NUTRIGINX (PUSER),
      - UTRIGIN (PUSER), NUSWIN (PUSER), NUSWINX (PUSER), USWIN (PUSER),
      - NUSIGOUT (PUSER), NUSIGOUTX (PUSER), USIGOUT (PUSER),
      - NUTRIGOUT (PUSER), NUTRIGOUTX (PUSER), UTRIGOUT (PUSER),
      - NUSWOUT (PUSER), NUSWOUTX (PUSER), USWOUT (PUSER)
```

```
* VALB.FOR

      INTEGER NVALS, PVALS, VALS
      PARAMETER (PVALS=100)
      CHARACTER*6 VALNAMES (PVALS)

      COMMON /VALB/ NVALS, VALS (PVALS), VALNAMES
```