

*
* GLOBALS FOR APL RUNTIME SUPERVISOR
*

MACRO BOOL _ SCALAR
MACRO PROC _ SCALAR
MACRO REPEAT _ WHILE TRUE DO
MACRO ENDREPEAT _ ENDWHILE
MACRO UNLESS _ IF NOT
MACRO ENDUNLESS _ ENDIF
MACRO UNTIL _ WHILE NOT
MACRO ENDUNTIL _ ENDWHILE
MACRO MOD _ REM

SCALAR CTRL\TERM; * PHYSICAL LINE NO OF CONTROLLING TERMINAL
BOOL PROMPTED; * HAS USER BEEN GIVEN A PROMPT?

*
* AIPU STATE-AREA
*

CONSTANT SA\SIZE _ 8
FIELD SA\SBASE (0: 0,15), SA\LTOP (0: 16,31)
FIELD SA\FLAGS (1: 0,15), SA\LBASE (1: 16,31)
FIELD SA\TRAPWORD (2)
 FIELD SA\TRAPCLASS (2: 0,15), SA\TRAPNUMBER (2: 16,31)
FIELD SA\ROVER (6: 0,15)
FIELD SA\BLOCKPTR (7: 0,15)

MACRO FIND\STATE (DTSEG) _ (SETREFSIZE (@(DTSEG)[0], SA\SIZE))

*
* AIPU STACK-FRAME
*

CONSTANT SF\DSIZE _ 2
FIELD SF\LASTLBASE (0: 0,15)
FIELD SF\FNDESC (1: 0,15), SF\PCTR (1: 16,31)

*
* AIPU FUNCTION-HEADER FIELDS
*

FIELD FH\CALLTRACE (0: 1,1), FH\RTNTRACE (0: 2,2), ..
 FH\RESULT (0: 3,3), FH\NLines (0: 4,15), ..
 FH\NARGS (0: 16,19), FH\NLOCALS (0: 20,31)
FIELD FH\NCODE (1: 0,15)

*
FIELD APL\TYPE (0: 1,7), APL\VALUE (0: 8,31)
MACRO APL\UNDEFINED _ (2 @ APL\TYPE)

*
CONSTANT TYPE\ARRAY _ 10B, TYPE\INTEGER _ 40B, TYPE\CHAR _ 20B
*

* AIPU INSTRUCTIONS OF INTEREST

*
CONSTANT BRKOP _ 323B

*

*
* SHARED FIELDS IN MULTIPLE TABLES
*

*
* IN ALL DESCRIPTORS
*

FIELD D\FREE (0); * IS THIS DESCRIPTOR FREE ?

*
* TERMINAL TABLE
*

CONSTANT MAXTERMS _ 16; * MAX TERMINALS
CONSTANT TD\SIZE _ 6; * SIZE OF TERMINAL DESCRIPTOR
*

VECTOR TERM\TABLE [MAXTERMS TD\SIZE WORD]
*

FIELD TD\PORT (1), ..PHYSICAL LINE NUMBER OF TERMINAL
TD\IN\BOX (2), ..INPUT ATTACHED MAILBOX
TD\OUT\BOX (3), ..OUTPUT ATTACHED MAILBOX
TD\MVEC (4 THRU 5 TO VECTOR), ..INPUT MESSAGE VECTOR
TD\CURSOR (6: 0,15), .. PHYSICAL CURSOR POSITION
TD\WIDTH (6: 16,31) ;* LINE WIDTH OF TERMINAL

*

```

*
* PROCESS TABLE
*
CONSTANT MAXPROCS _ 16;    * MAX PROCESSES
CONSTANT PD\SIZE _ 8;     * SIZE OF PROCESS DESCRIPTOR
*
VECTOR PROC\TABLE [ MAXPROCS PD\SIZE WORD ]
*
FIELD PD\BREAK ( 1: 0,0 ),      ..WAS LAST TRAP BKPT (2,4)?
PD\STOPPED ( 1: 1,1 ),          ..IS PROCESS STOPED?
PD\PRCAP ( 1: 2,15 ),           ..CLIST INDEX OF PROCESS CAP
PD\DCNT ( 1: 16,31 ),           ..MESSAGE DELIVERY COUNT
PD\DTSEG ( 2 THRU 3 TO VECTOR ), .. VECTOR DESCR OF DATA SEG
    PD\DTCAP ( 3: 16,31 ), ..
PD\TRAPWORD ( 4 ),              ..INFO ON LAST ERROR TRAP
    PD\TRAPCLASS ( 4: 0,15 ), ..
    PD\TRAPNUMBER ( 4: 16,31 ), ..
PD\MBOXLIST ( 5 THRU 7 )        ;*LIST OF RECEIVER QUEUES

MACRO CLEAR\TRAP ( PD ) _ PD . PD\TRAPWORD _ 0

*
* INFO ON CURRENT DEBUGGEE PROCESS
*
SCALAR DB\PN;                  * PROCESS NUMBER
MACRO DB\CHECK _ IF DB\PN < 0 DO FRETURN
MACRO NO\DB _ DB\PN _ -1
REFERENCE DB\PD;               * PD OF DEBUGGEE
REFERENCE DB\ENV;              * ENVIRONMENT STACK FRAME FOR PV COMMAND
*
* INFO ON MOST RECENTLY TRAPPED PROCESS
*
SCALAR TRAP\PN
REFERENCE TRAP\PD

```

*

```

*
* MAILBOX TABLE
*
  CONSTANT MAXBOXES _ 64;*LIMIT ON NBOXES
  CONSTANT MBD\SIZE _ 10;* SIZE OF MAILBOX DESCRIPTOR
  *
  VECTOR MAILBOX\TABLE [ MAXBOXES MBD\SIZE WORD ]
  *
  FIELD MBD\MQHD ( 1 THRU 2 TO VECTOR ),    ..MSG QUEUE HEAD
        MBD\MQTL ( 3 THRU 4 TO VECTOR ),    ..MSG QUEUE TAIL
        MBD\RCVQ ( 5 THRU 7 ),              ..RECEIVER QUEUE
        MBD\PUTM ( 8: 0,15 ),               ..PUT-MSG FUNCTION
        MBD\GETM ( 8: 16,31 ),              ..GET-MSG FUNCTION
        MBD\ACTION ( 9: 0,7 ),              ..ARE ALL INPUT CHARS ACTION CHARS?
        MBD\ASTAT ( 9: 8,15 ),              ..ATTACHMENT STATUS
        MBD\AOBJ ( 9: 16,31 )              ;*ATTACHED OBJECT ( IF ANY )

  CONSTANT MBD\UNATTACHED _ 0,              .. NOT ATTACHED
        MBD\TERM _ 1,                      .. TERMINAL
        MBD\FILE _ 2,                      ..
        MBD\TIMER _ 3

  MACRO RCVQ\EMPTY(B) _ (LENGTH(@(MAILBOX\TABLE[B]$MBD\RCVQ))=0)

```

```

*
* MESSAGES
*
  VECTOR MSG\SPACE [ 2000 ]
  *
  CONSTANT MSG\DISP _ 3;                    * SIZE OF MESSAGE-HEADER
  MACRO MSG\SIZE ( L ) _ ((L) + MSG\DISP)
  MACRO MSG\HDR ( MV ) _ (SETREFSIZE ( @ (MV) [ 0 ], MSG\DISP ))
  *
  FIELD MSG\NEXT ( 0 THRU 1 TO VECTOR ),    ..MSG QUEUE LINK
        MSG\DELIVER ( 2: 0,0 ),             ..IS MSG TO BE DELIVERED ?
        MSG\SOURCE ( 2: 16,31 )             ;*WHO SENT MSG ?

  MACRO MSG\NULL ( M ) _ ((M) $ VDBASE = 0)
  VECTOR NULL\MSG

```

```

*
* FILE TABLE
*
CONSTANT MAXFILES _ 16;    * MAX FILES
CONSTANT FD\SIZE _ 9;      * SIZE OF FILE-TABLE-ENTRY
*
VECTOR FILE\TABLE [ MAXFILES FD\SIZE WORD ]
*
FIELD FD\CAPX ( 1 ),      ..C-LIST INDEX OF FILE CAP
FD\RING ( 2 THRU 4 ),    ..RING FOR SEQ FILE I/O
FD\VEC(2 THRU 3 TO VECTOR), ..VEC PART OF RING
FD\VEC\CAPX(3:16,31),    ..CAPX FOR FILE IN RING-DES
FD\RING\IN(4:0,15),      ..FILE READ-POINTER
FD\RING\OUT(4:16,31),    ..FILE WRITE-POINTER
FD\IN\BOX ( 5 ),         ..INPUT ATTACHED MAILBOX
FD\OUT\BOX ( 6 ),        ..OUTPUT ATTACHED MAILBOX
FD\IN\MSG(7 THRU 8 TO VECTOR) ;*PHONY INPUT MSG VECTOR
*
VECTOR FD\MSG\VECS[MAXFILES*MSG\DISP] ;*PHONY-MSG HEADERS

CONSTANT SEGNAME\LEN _ 8
CONSTANT PAGE\SIZE _ 512
*
* CREATING APL FILES
*
CONSTANT FILE\CAPX _ ALLOCATE\CAP
VECTOR FILE\VEC [ 1 EXTERNAL FILE\CAPX ]

```

★

*

* DELAY-TIMER

*

```
FIELD TL\LTIME(1:16,31),  ..TIMER LIST LEFT PART OF TIME
      TL\RTIME(0),         ..TIMER LIST RIGHT PART OF TIME
      TL\TBOX(1:0,15)      ;*TIMER LIST MAIL BOX NUMBER
STRING TIMER\LIST[MAXBOXES 2 WORD] ;*TIMER LIST
MACRO RTCLOCK _ CODE(4160B)
```

*

*

* NUMBER FORMATS

*

SCALAR INT\WIDTH, ..
FRAC\WIDTH, ..
EXP\WIDTH, ..
NUM\SIGN, ..
MAX\CONV, ..
POINT, ..
SIGDIG

*

SCALAR MAX\INOT, ..
MAX\FNOT, ..
MIN\FNOT

*

SCALAR SAVE\FRAC\WIDTH, ..
SAVE\EXP\WIDTH

*

*

* CONVERSION MODES

*

CONSTANT CFORMAT _ 0, ..
IFORMAT _ 1, ..
FFORMAT _ 2, ..
EFORMAT _ 3

*

*

*

```

CONSTANT IDLENGTH _ 30
MACRO MESSAGE(M)_SOUT(M)&COUT('&J')
MACRO ERRMSG(M)_MESSAGE(M) & ERRFLAG _ TRUE
BOOL ERRFLAG
*
CONSTANT TAILSZ _ 2
STRING TEXTTAIL _ "T"
STRING CODETAIL _ "C"
STRING DEBUGTAIL _ "D"
*
CONSTANT PNAMEMAX _ 6
CONSTANT FNAMEMAX _ PNAMEMAX + TAILSZ
*
BOOL LOADED
STRING PROGNAME [ PNAMEMAX ]
    STRING TEXTNAME [ FNAMEMAX ]
    STRING CODENAME [ FNAMEMAX ]
    STRING DEBUGNAME [ FNAMEMAX ]
*
BOOL RTLOADED
STRING RUNTNAME [ PNAMEMAX ]
    STRING RTEXTNAME [ FNAMEMAX ]
    STRING RCODENAME [ FNAMEMAX ]
    STRING RDEBUGNAME [ FNAMEMAX ]
*
MACRO NBLOCKS _ DIRSIZE(TXSEG) - 1
MACRO INVALID(B) _ NOGOOD(TXSEG,B)
MACRO RTINVALID(B) _ NOGOOD(RTXSEG,B)
*
*
CONSTANT LOCALLY _ 0, GLOBALLY _ 1
CONSTANT GLOBLK _ 1
STRING GLOBNAME _ "GLOBAL"

SCALAR INITPFD; * FN DESCR OF @INIT
SCALAR INITOPFD; * AND OF @INIT0
*
MACRO PCHECK _ UNLESS LOADED DO FRETURN
MACRO RCHECK_UNLESS(RUNFLAG=1 OR RTLOADED)DO FRETURN
MACRO PROCSEG(R)_(IF(R)=0 THEN CDSEG ELSE RCDSEG)
MACRO DBUGSEG(R)_(IF(R)=0 THEN DBSEG ELSE RDBSEG)
*

```

```

*
* SIZES OF VARIOUS THINGS IN USER PROGRAM
*
SCALAR DTSIZE,      ..TOTAL SIZE OF DATA SEGMENT, MADE UP OF:
      GVSIZE, ..GLOBAL-VARIABLE AREA
      ABSIZE, ..ARRAY-BLOCK-STORAGE AREA
      STSIZE  ;*STACK AREA

```

```

*
* STRING STUFF
*
MACRO CLEAR ( S ) _ SETS ( (S), 0, 0 )
CONSTANT BLANK _ ' ', BLANKS _ '&167', CRESC _ '&U', ..
      BLOT _ '$', BELL _ '&G', RUBOUT _ '&137', ..
      CARRET _ '&M', NUL _ '&140'
CONSTANT NONCHAR _ -1
MACRO REAL ( CHAR ) _ ( (CHAR) # NONCHAR )

```

```

*
* GLOBALS FOR NUMBER CONVERSION ROUTINES
*
CONSTANT EBIAS _ -128
MACRO FRET(N)_FRETURN N
MACRO LMUL(A,B)_(A&CODE(4114B)&B&CODE(11040B))
MACRO DOUBLE_VECTOR
FIELD SIGNF(0:0,0), IN(0:2,2)
*

```

```

*
* MACHINE-DEPENDENT STUFF
*
MACRO \PUSH(A)_((A)&CODE(4114B))
MACRO \PUSHD(A)_((A)&CODE(4116B));*WILL SCREW UP IF NOT OF 2-WORD TYPE
MACRO \TRAP(A)_((A)&CODE(4164B,44106B,1,4100B))

CONSTANT CPW_4,    ..CHARS/WORD
          WPW_1,    ..1 FOR SIMPLE, 2 FOR QSPL
          INST1_8,  .. ADDR OF 1ST INST IN FUNCTION
          BPW_4,    .. INST BYTES PER WORD
          LSZW_20   ;*LINE SIZE IN WORDS
CONSTANT LSZ_LSZW*CPW  ;*LINE SIZE IN CHARS

STRING CSTRING[LSZ], TARGET[LSZ]
*
FIELD LASTCHAR(0:24,31)
MACRO MORELINE (W)_ ((W)$LASTCHAR#CARRET)

*
* OBJECT RETURNED FROM SERVICE CALL
*
MACRO PUSH\RESUME ( OBJ , PD ) _ ..
          ( PUSHWORD ( OBJ , PD ) & ATTN\RESUME ( PD ) )

*
* INFO ON ERRORS CAUSED BY SERVICE CALLS
*
CONSTANT CALL\ERR _ 11;      * ERROR-CLASS
*
CONSTANT NO\ERR _ -1,        ..NO ERROR
          PARAM\ERR _ 0,      ..BAD PARAMETER
          FULL\ERR _ 1,       ..TABLE FULL
          NO\OBJ\ERR _ 2      ;*NO SUCH OBJECT
*
CONSTANT FILEOPEN\ERR _ 3,   ..TRIED TO DELETE OPEN FILE
          DUPNAME\ERR _ 4,   ..TRIED TO CREATE FILE WITH SAME NAME
          NAMELEN\ERR _ 5,   ..NAME TOO LONG
          NOT\FINBOX\ERR _ 6, ..NOT FILE-INPUT BOX
          FILE\OPENING\ERR _ 7, .. CAN NOT OPEN FILE
          BOX\ATTACH\ERR _ 8, ..BOX ATTACHMENT ERR
          SAME\BOX\ERR _ 9,   ..IN AND OUT BOXES ARE SAME
          ALARM\SET\ERR _ 10, ..ALARM ALREADY SET
          NOT\TINBOX\ERR _ 11, ..NOT TERMINAL-INPUT BOX
          NO\TERM\ERR _ 12    ;* NO SUCH TERMINAL
          * ( NOTE: ADD ANY NEW ERRORS TO FUNCTION "RSCALL\ERR" AS WELL. )

*
* PARAMETER CHECKING STUFF
*

```

FIELD WRD1 (0)
MACRO LEGAL (N) _ ((N) \$ WRD1 # ILLEGAL)
CONSTANT ILLEGAL _ 0
VECTOR ILLEGAL\VEC

*

```

*
* LINE POINTERS AND WINDOWS
*
FIELD SEGMENT ( 0 THRU 1 TO VECTOR ), BLOCK ( 2 )
*
*LINE POINTERS
*
FIELD LINE ( 3 ), STRNG ( 4 THRU 6 ), STARTX ( 7 ), ENDX ( 8 )
CONSTANT NPTRS _ 8, PTRSIZE _ 9
VECTOR PTRVEC [ NPTRS PTRSIZE WORD ]
REFERENCE CURRENT,SEARCH,IPTR1,IPTR2
CONSTANT NIL _ -1
VECTOR PTRNAME [ NPTRS ] _ ..
    NIL, NIL, NIL, NIL, 'A', 'B', 'C', 'D'
MACRO PTRADDR(I) _ (@PTRVEC[I])
MACRO MAKENULL(P) _ P.BLOCK _ 0
MACRO NULL(P) _ ((P).BLOCK = 0)
MACRO SETPTRADDR(P,Q) _ BCOPY(P,Q,PTRSIZE)
*
* WINDOWS
*
FIELD BARRAY ( 3 THRU 4 TO VECTOR ), BUFFER ( 5 )
MACRO BLENGTH ( W ) _ ( ( W ).BARRAY ) $ VDSIZE )
CONSTANT NWINS _ 4, WINSIZE _ 6
VECTOR WINVEC [ NWINS WINSIZE WORD ]
REFERENCE TXWINDOW,GDBWINDOW,CDBWINDOW,CCDWINDOW
MACRO WINADDR(I) _ (@WINVEC[I])
*

```

```
*
* TEXT BUFFER
*
SCALAR TSZ
VECTOR TARRAY
SCALAR TLENGTH, TLINEs
*
* GLOBAL STATE VARIABLES
*
BOOL WRAP
SCALAR SCOPE
*
* SYMBOL TABLE STUFF
*
MACRO ISFUNCTION(STE)_(STE.STYPE=SFUNC)
*
```

```
*
* EX-SIMULATED EXTERNAL-SEGMENT-FIELD MACHINERY
*
MACRO XFIELD ( NAME, XWORD, XLBIT, XRBIT ) _ ..
    FIELD NAME ( XWORD: XLBIT, XRBIT )
*
*
*
MACRO GETFIELD(A,F)_A.F
MACRO PUTFIELD(A,F,X)_A.F_X
*
```

```

REFERENCE TOP
REFERENCE ENV
*
*
*
* DEBUGGER BLOCK HEADER
*
CONSTANT DBHDR _ 0
CONSTANT DBHDRSZ _ 3
*
* DEBUGGER BLOCK AND LINE-TABLE FIELDS
*
FIELD LTEISAVE ( 0: 5,12 ),...
    LTEFLAGS ( 0: 13,15 ), LTEBREAK ( 0: 13,13 ),...
        LTESTEP ( 0: 14,14 ), LTETSTEP ( 0: 15,15 ),...
        LTECODEPTR ( 0: 16,31 )
CONSTANT LTENULL _ 0
*
* CURRENT LINE TABLE ENTRY
*
SCALAR LTE
*
* CURRENT SYMBOL-TABLE ENTRY
*
REFERENCE STE

VECTOR STEVEC [ 2 ]
*
```

```
CONSTANT FCMARKER _ 177777B;      * SPECIAL LASTLBASE ON FORCED CALLS
*
* FIELDS IN FUNCTION DESCRIPTORS
*
FIELD FNRUNT ( 0: 20,20 ), FNBLK ( 0: 21,31 )
MACRO FNDESC(RUNT,BLK) _ OR ( RUNT @ FNRUNT, BLK @ FNBLK )
MACRO STEFUNCD ( E ) _ (IF(E).SFTYPE>1 THEN (E).SVAL1 ELSE (E).SVAL2 )
*
```

```
MACRO DM_MACRO
DM DI_SCALAR
DM DS_RING
DM DV_VECTOR
DM DC_CONSTANT
DM DF_FIELD
DM DR_REFERENCE
DM DEQV_EQUIVALENCE
DM LSH_SHIFT-
DM RSH_SHIFT
```

```
DF LEFT16(0:0,15)
DF RIGHT16(0:16,31)
DF RIGHT1(0:31,31)
```

★

DC NCHWD_4;*NUMBER OF CHAR PER WORD
DC NWDCON_1;*NUMBER OF WORDS PER CONSTANT

*DEBUGGER BLOCK DECLARATIONS

DF VRGAPTR(0:16,31)
DF STFREE(2:16,31)
DF HCSIZE(1:16,31)
DC HCBASE_3
DF LTBASE(2:16,31)
DF LTSIZE(2:0,15)

*SYMBOL TABLE ENTRY DECLARATIONS

DC SDISP_2
DF SCHAIN(0:16,31)
DF STYPE(0:0,1)
DC SVAR_1,SLABEL_2,SFUNC_3
DF SFTYPE(0:2,4)
DC SNULLADIC_4,SMONADIC_2,SDYADIC_1
DF SPRIMATIVE(0:6,10)
DF SLG(0:5,5)
DF SVAL(1:0,31)
DF SVAL1(1:0,15)
DF SVAL2(1:16,31)
DF SSIZE(0:11,15)
DF SNAME0(2)
DF SWD0(0)
DF SPWD0(0:0,15)

*

*BUFFER ROUTINE DECLARATIONS

```
DC NBUF_7
VECTOR BUFBASE[NBUF VECTORS]
DM BUFSIZE(XX)_BUFBASE[XX]$VDSIZE
VECTOR BUFSEG[NBUF VECTORS]
VECTOR BUFBLK[NBUF]
VECTOR BUFREF[NBUF REFERENCES]
DC GDBBUF_0,CDBBUF_1,CCDBUF_2,CTXBUF_3,TTBUF_4,TLBUF_5,TNBUF_6
DEQV GDBBASE=BUFBASE[GDBBUF]
DM GDBSIZE_GDBBASE$VDSIZE
DEQV GDBREF=BUFREF[GDBBUF]
DEQV CDBBASE=BUFBASE[CDBBUF]
DM CDBSIZE_CDBBASE$VDSIZE
DEQV CDBREF=BUFREF[CDBBUF]
DEQV CCDBASE=BUFBASE[CCDBUF]
DM CCDSIZE_CCDBASE$VDSIZE
DEQV CTXBASE=BUFBASE[CTXBUF]
DM CTXSIZE_CTXBASE$VDSIZE
DEQV TT=BUFBASE[TTBUF]
DM TTSIZE_TT$VDSIZE
DEQV TLBASE=BUFBASE[TLBUF]
DM TLSIZE_TLBASE$VDSIZE
DEQV TNBASE=BUFBASE[TNBUF]
DM TNBSIZE_TNBASE$VDSIZE
```

*RUNTIME FLAG

```
DI RUNFLAG
```

*

* CAPABILITY SEGMENT SLOT ASSIGNMENTS *

```
    CONSTANT BIG _ 177777B; * MAX SIZE VECTORS *  
MACRO EXT\VECTOR(V) _ ..
```

```
    CONSTANT EVTEMP _ ALLOCATE\CAPS(1) ! ..  
    VECTOR V [ BIG EXTERNAL EVTEMP ]
```

```
    EXT\VECTOR ( RDBSEG );*RUNTIME DEBUGGER SEGMENT  
    EXT\VECTOR ( RTXSEG );*RUNTIME TEXT SEGMENT  
    EXT\VECTOR ( RCDSEG );*RUNTIME CODE SEGMENT  
    EXT\VECTOR ( DBSEG );*DEBUGGER SEGMENT  
    EXT\VECTOR ( TXSEG );*TEXT SEGMENT  
    EXT\VECTOR ( CDSEG );*CODE SEGMENT
```

```
    CONSTANT DTCAPS _ ALLOCATE\CAPS ( MAXPROCS )  
    CONSTANT TRAP\CAP _ ALLOCATE\CAPS ( 1 )  
    CONSTANT PRCAPS _ ALLOCATE\CAPS ( MAXPROCS )  
    CONSTANT FTCAPS _ ALLOCATE\CAPS ( MAXFILES )  
    CONSTANT TCAP _ ALLOCATE\CAPS ( 1 )
```

*

*SEGMENT INPUT-OUTPUT ROUTINES

```
DM DBI(WW,XX,YY,ZZ)_VCOPY(ZZ,SUBVEC(WW,YY,XX),XX);*READ BLOCK
DM DBO(WW,XX,YY,ZZ)_VCOPY(SUBVEC(WW,YY,XX),ZZ,XX);*WRITE BLOCK
```

*SEGMENT ACCESS ROUTINES

```
DC DIRESIZE_2;*DIRECTORY ELEMENT SIZE
DM SEGSIZE(XX)_XX[0]$RIGHT16
DM DIRSIZE(XX)_XX[1]$RIGHT16
DM BLKBASE(XX,YY)_XX[(YY) LSH 1]$LEFT16
DM BLKHEAD(XX,YY)_XX[(YY) LSH 1]$RIGHT16
DM BLKSIZE(XX,YY)_XX[((YY) LSH 1)+1]$LEFT16
DM BLKINFO(XX,YY)_XX[((YY) LSH 1)+1]$RIGHT16
DF BLKINVALID(0:31,31);*CODE FOR BLOCK INVALID
```

*

```
*  
* FREE-STORAGE POOL  
*  
  VECTOR FREEPool [ 500 ]  
  *  
END
```

```
FUNCTION ARS()  
*  
* APL RUNTIME SUPERVISOR  
*  
  RSINIT()  
  *  
  REPEAT  
    COMMANDS()  
    SERVICES()  
    ERRORS()  
  ENDREPEAT  
END
```

```
FUNCTION ACTION\CHAR ( C, MBOX )
*
*
* ARE ALL CHARACTERS ACTION CHARACTERS?
*
  IF MAILBOX\TABLE [ MBOX ] $ MBD\ACTION DO
    RETURN TRUE
  ELSE DO
    RETURN C = CARRET
  ENDIF
END
```

```

FUNCTION ALL\CHAR(OBJ) RETURNING VECTOR
*
*CHECK THAT 'OBJ' IS ALL CHAR TYPE AND THEN RETURN
* VECTOR DESCRIPTOR TO IT, ELSE RETURN ILLEGAL\VEC
*
SCALAR L
VECTOR VEC[4], AVEC
FIELD RANK(0:8,15), REFCNT(0:0,15), NELTS(0:16,31)
*
  IF OBJ$APL\TYPE=TYPE\ARRAY THEN
    AVEC_ARRAY\BLOCK(OBJ,TRAP\PD)
    IF AVEC[0]$RANK=1 THEN
      L_AVEC[1]$NELTS+2
      FOR I_3 TO L DO
        IF AVEC[I]$APL\TYPE#TYPE\CHAR THEN
          RETURN ILLEGAL\VEC
        ENDIF
      ENDFOR
    ELSE
      RETURN ILLEGAL\VEC
    ENDIF
  ELSEIF OBJ$APL\TYPE=TYPE\CHAR THEN
    AVEC_VEC
    AVEC[0]$RANK_1
    AVEC[1]$REFCNT_1
    AVEC[1]$NELTS_1
    AVEC[2]_APL\INTEGER(1)
    AVEC[3]_OBJ
  ELSE
    RETURN ILLEGAL\VEC
  ENDIF
  RETURN AVEC
END

```

```
FUNCTION ALPHABETIC ( SCALAR C )  
  IF 'A' <= C AND C <= 'Z' DO  
    RETURN TRUE  
  ELSE DO  
    RETURN FALSE  
  ENDIF  
END
```

```
FUNCTION ALPHANUMERIC ( SCALAR C )  
  IF ALPHABETIC ( C ) OR NUMERIC ( C ) DO  
    RETURN TRUE  
  ELSE DO  
    RETURN FALSE  
  ENDIF  
END
```

```

FUNCTION APLCSN(STRING STR) RETURNING SCALAR
*APL VERSION OF "CONVERT STRING TO NUMBER".
CONSTANT NST_10
VECTOR C1[NST 16 BIT]_A3,A3,A2,A3,A9,A8,A9,A11,A11,A11
VECTOR C2[NST 16 BIT]_A2,A2,A2,A2,A8,A8,A8,A11,A11,A11
VECTOR C3[NST 16 BIT]_A4,A13,A13,A1,A1,A13,A13,A10,A1,A13
VECTOR C4[NST 16 BIT]_A5,A14,A7,A5,A1,A13,A13,A1,A1,A13
VECTOR C5[NST 16 BIT]_A1,A13,A6,A1,A1,A6,A6,A1,A1,A13
VECTOR C6[NST 16 BIT]_A1,A13,A13,A1,A1,A13,A13,A1,A1,A13
VECTOR ACT[6 VECTORS]_C1,C2,C3,C4,C5,C6
SCALAR NUM,CS,EXP,DFLAG,POW,LAB,NSIGN,ESIGN,CH,NCOUNT,B
MACRO MFSMT(X,L)_LAB_L! CS_X! GOTO FSMT
MACRO MFSM(X)_CS_X! GOTO FSM
DOUBLE FSMC
SCALAR VTENBASE
FIELD NUMF(0)
FIELD LABF(1:16,31)
FIELD UP(1)
MACRO LOADFSM_FSMC$NUMF_NUM! FSMC$FRSTR_STR.SDFPTR! FSMC$LABF_LAB
MACRO UNLOADFSM_NUM_FSMC$NUMF! STR.SDFPTR_FSMC$FRSTR! LAB_FSMC$LABF
MACRO SIGN(N)_(N$SIGNF*-2+1)
DOUBLE DPROD
FIELD FRSTR(1:0,15)
CONSTANT MAX_177777777777B/10
*INITIALIZATION
    NUM_0
    EXP_0
    POW_0
    NCOUNT_FALSE
    NSIGN_FALSE
    ESIGN_FALSE
    CS_0
    DFLAG_FALSE
    LAB_A0
*LOAD INITIAL FSM CONFIGURATION
    LOADFSM
*REMOVE PRECEEDING BLANKS
    WHILE GCNI(STR:(CH_'X'&GOTO FSM2))=' ' DO
        GCI(STR:(CH_'X'&GOTO FSM2))
    ENDWHILE
    GOTO FSM1
*LOAD FSM CONFIGURATION ON LEAVING TERMINAL STATE AND ..
*    ENTERING NON-TERMINAL STATE
FSMT:  LOADFSM
*FINITE STATE MACHINE
FSM:   CH_GCI(STR:(CH_'X'&GOTO FSM2))
FSM1:  CH_GCNI(STR:(CH_'X'&GOTO FSM2))
FSM2:  GOTO ACT[CLASS(CH)][CS]
* FSM STATE-ACTION
*FAIL TO FIND NUMBER
A0:    FRET(11)
*ILLEGAL CHARACTER; BACK UP FSM TO LAST TERMINAL STATE(A0 OR A13)
A1:    UNLOADFSM
        GOTO LAB

```

```

*COLLECT SIGNIFICANT DIGITS BEFORE DECIMAL POINT
A2:   IF NUM<MAX THEN
        NUM_NUM*10+CH-'0'
    ELSE
        POW_POW+1
    ENDIF
    MFSM(2)
*IGNOR LEADING ZEROS
A3:   MFSM(1)
*RECORD SIGN OF NUMBER
A4:   NSIGN_TRUE
        MFSM(3)
*DECIMAL POINT: REAL NUMBER
A5:   DFLAG_TRUE
        MFSM(4)
*EXPONENT
A6:   MFSMT(7,A13)
*DECIMAL POINT: REAL NUMBER
A7:   DFLAG_TRUE
        MFSM(5)
*COLLECT SINIFICANT DIGITS AFTER DECIMAL POINT
A8:   IF NUM<MAX THEN
        NUM_NUM*10+CH-'0'
        POW_POW-1
    ENDIF
    MFSM(5)
*COLLECT LEADING ZEROS OF FRACTION
A9:   POW_POW-1
        MFSM(6)
*RECORD SIGN OF EXPONENT
A10:  ESIGN_TRUE
        MFSM(8)
*COLLECT EXPONENT
A11:  EXP_EXP*10+CH-'0'
        MFSM(9)
*DECIMAL POINT: REAL NUMBER
A14:  DFLAG_TRUE
        MFSM(6)
*EXIT WITH INTEGER OR REAL
A13:  IF NUM=0 THEN RETURN 1@IN
        IF ESIGN THEN EXP_-EXP
        POW_POW+EXP
        IF NUM<0 THEN FRET(16)      ; * SHOULD NEVER HAPPEN
        WHILE NUM REM 10 = 0 DO
            NUM_NUM/10
            POW_POW+1
        ENDWHILE
        IF DFLAG=FALSE AND POW<8 THEN
            WHILE NUM<4B7 AND POW>0 DO
                NUM_NUM*10
                POW_POW-1
            ENDWHILE
            IF NUM<4B7 AND POW=0 THEN
                NUM$SIGNF_NSIGN

```

```

        NUM$IN_1
        RETURN NUM
    ENDIF
ENDIF
IF POW<-47 OR POW>47 THEN FRET(12)
NCOUNT_1
WHILE NUM>0 DO
    NUM_NUM SHIFT -1
    NCOUNT_NCOUNT-1
ENDWHILE
NUM_NUM SHIFT 1
VTENBASE_0
IF POW<0 THEN VTENBASE_POW+POW-48
DPROD_LMUL(NUM,FRAC TEN(POW-VTENBASE))
NUM_DPROD$UP SHIFT -1
POW_EXPTEN(POW-VTENBASE)*SIGN(POW)
POW_POW+NCOUNT
RETURN NORM(NUM,POW,NSIGN:FRET(102))
END

```

```

FUNCTION APLMAKE(VECTOR SPACE, SCALAR MAKESIZE) RETURNING VECTORS
*
*RETURN A VECTOR OF SIZE 'MAKESIZE' FROM VECTOR 'SPACE'.
*VECTOR INCLUDES HEADER, BUT EXCLUDES SLOP.
*
    CONSTANT MAXSLOP_4
    SCALAR I
    REFERENCE P
    FIELD SIZE(0:16,31)
    FIELD THISFREE(0:0,0)
    FIELD PREFREE(0:1,1)
    FIELD SLOP(0:2,7)
    FIELD NEXT(1:0,15)
    FIELD LAST ( 1: 16,31 )
    FIELD XNEXT(0:0,15)
    FIELD XLAST(0:16,31)
    MACRO ROVER(SPACE) _ FIND\STATE ( SPACE ) . SA\ROVER
    I_ROVER(SPACE)      ;* START SEARCH AT ROVER
    WHILE SPACE[I]$SIZE<MAKESIZE DO      ;* SEARCH FOR LARGE ENOUGH BLK.
        I_SPACE[I+1]$XNEXT
        IF I=ROVER(SPACE) THEN FRETURN      ;* SEARCHED COMPLETE FREE-LIST?
    ENDWHILE
    P_SETREFSIZE(@SPACE[I])      ;* FORM POINTER TO BLK
    SPACE[P.SIZE+I]$PREFREE_FALSE ;* FLAG BLK-RESERVED ON NEXT BLK.
    ROVER(SPACE)_P.NEXT          ;* UPDATE ROVER
    IF P.SIZE-MAKESIZE<MAXSLOP THEN ;* PREVENT CREATION OF SMALL BLK.
        SPACE[P.LAST+1]$XNEXT_P.NEXT ;* UNCHAIN BLK FROM FREE-LIST
        SPACE[P.NEXT+1]$XLAST_P.LAST
        P.SLOP_P.SIZE-MAKESIZE      ;* RECORD SLOP SIZE
        P.PREFREE_FALSE            ;* SET FLAG: PREVIOUS BLK RESERVED
    ELSE ;* DIVIDE FREE-BLK AND MAKE EXACT SIZE
        P.SIZE_P.SIZE-MAKESIZE      ;* REDUCE SIZE OF FREE BLK.
        I_P.SIZE+I                  ;* I BECOMES INDEX TO RESERVED BLK
        SPACE[I-1]$SIZE_P.SIZE      ;* RECORD NEW SIZE AT TAIL OF FREE BLK
        P_SETREFSIZE(@SPACE[I])    ;* FORM POINTER TO RESERVED BLK
        P.SLOP_0                    ;* RECORD SLOP SIZE
        P.PREFREE_TRUE              ;* SET FLAG FOR PREVIOUS BLK FREE
    ENDIF
    P.THISFREE_FALSE              ;* FLAG BLK RESERVED
    P.SIZE_MAKESIZE               ;* RECORD RESERVED BLK SIZE
    SPACE$VDSIZE_MAKESIZE         ;* PACK VECTOR DESCRIPTOR
    SPACE$VDBASE_SPACE$VDBASE+I
    RETURN SPACE
END

```

```
-----  
FUNCTION APL\ARRAY ( VECTOR A )  
*  
* RETURNS APL ARRAY DESCRIPTOR WORD FOR  
* EXTERNAL VECTOR 'A'  
*  
    SCALAR B  
    *  
    B _ A $ VDBASE  
    RETURN OR ( TYPE\ARRAY @ APL\TYPE, B @ APL\VALUE )  
END
```

```
FUNCTION APL\CHAR ( C )
*
* RETURNS APL CHARACTER DESCRIPTOR WORD FOR
* CHARACTER 'C'
*
    RETURN OR ( TYPE\CHAR @ APL\TYPE, C @ APL\VALUE )
END
```

```
FUNCTION APL\CREATE(CINDEX,STRING DNAME,STRING UNAME,STRING RNAME)
*CREATES AN APL PROCESS WITH THESE SEGMENTS (THE SEGMENTS
*   MUST NOT BE OPEN WINDOWED AT THE TIME OF THE CALL) AND
*   RETURNS A CAPABILITY FOR IT AT CAP-SEG ENTRY CINDEX

    \PUSHD(\NAME\CONV(RNAME))
    \PUSHD(\NAME\CONV(UNAME))
    \PUSHD(\NAME\CONV(DNAME))
    \PUSH(CINDEX)
    \TRAP(18)
    RETURN
END
```

```
FUNCTION APL\DELETE(CINDEX)
```

```
    \PUSH(CINDEX)
```

```
    \TRAP(21)
```

```
    RETURN
```

```
END
```

FUNCTION APL\INIT()

*INITIALIZE THE APL MANAGER

 \TRAP(17)

 RETURN

END

```

FUNCTION APL\INTEGER ( SCALAR N )
*
* FORMAT N AS AN APL INTEGER
*
  FIELD OVERFLOW ( 0: 0,8 ), SIGNED\TYPE ( 0: 0,7 )
  CONSTANT POSINT _ 040B, NEGINT _ 240B
  SCALAR TYPE, VALUE
  *
  IF N < 0 DO
    TYPE _ NEGINT
    VALUE _ -N
  ELSE DO
    TYPE _ POSINT
    VALUE _ N
  ENDIF
  IF VALUE $ OVERFLOW = 0 DO
    RETURN ( TYPE @ SIGNED\TYPE + VALUE @ APL\VALUE )
  ELSE DO
    FRETURN
  ENDIF
END

```

FUNCTION APL\OFF(CINDEX)

 \PUSH(CINDEX)

 \TRAP(20)

 RETURN

END

```
FUNCTION APL\ON(CINDEX)
```

```
    \PUSH(CINDEX)
```

```
    \TRAP(19)
```

```
    RETURN
```

```
END
```

FUNCTION APL\TRAP(CINDEX)

*THIS FUNCTION EITHER RETURNS 0 OR ELSE RETURNS 1 AND PUTS A CAPABILITY
* FOR A NEWLY TRAPPED APL PROCESS AT THE SPECIFIED CAPABILITY INDEX

 \PUSH(CINDEX)
 RETURN \TRAP(22)

END

```
FUNCTION APPENDDD(REFERENCE D, REFERENCE S)
*
*APPEND DOUBLE-WORD STRING S TO D, UPDATING DESC OF D.
*
STRING T
    SDCOPY(T, S)
    REPEAT
        WDI(GDI(T:SRETURN),D:FRETURN)
    ENDREPEAT
END
```

```

-----
FUNCTION ARRAY\BLOCK ( OBJ, REFERENCE PD ) RETURNING VECTOR
*
* CONSTRUCT A VECTOR DESCRIPTOR FOR ARRAY-BLOCK OF
* APL-OBJECT 'OBJ' IN PROCESS 'PD'.
*
  SCALAR ADDR, ASIZE
  VECTOR AVEC, DTSEG
  FIELD ADDRESS ( 0: 16, 31 ), SIZE ( 0: 16,31 )
  *
  DTSEG _ PD . PD\DTSEG
  ADDR _ OBJ $ ADDRESS
  ASIZE _ DTSEG [ ADDR ] $ SIZE
  AVEC _ SUBVEC ( DTSEG, ADDR, ASIZE )
  RETURN AVEC
END

```

```

FUNCTION ATTACH\FILE()
*
*SERVICE CALL:
*  PARAM: APLCHARS OF FILE NAME
*          VECTOR OF 2 MAILBOXES(IN\BOX,OUT\BOX)
*  ACTION: GET FILENAME,IN\BOX AND OUT\BOX FROM PARAM
*          CHECK IN\BOX AND OUT\BOX :EXIST,UNATTACHED,DIFF
*          GET NEW FILE ENTRY NUMBER
*          OPEN THE FILE
*          SET READ/WRITE POINTERS
*          RECORD IN\BOX AND OUT\BOX IN FILE ENTRY
*          ENQUEUE PHONEY MESSAGE ON IN\BOX QUEUE
*          FOR BOTH MAILBOXES:SET ASTAT,AOBJ,GETM,PUTM
*  FAILIF: ILLEGAL NAME; NO BOXES; BOXES ATTACHED; BOXES SAME;
*          FILE TABLE FULL; NO FILE.
*
STRING APL\FILE[SEGMAME\LEN]
VECTOR PAIR
FIELD NELTS(0:16,31)
SCALAR IN\BOX, OUT\BOX, FN, FILE\SIZE
REFERENCE INMBD, OUTMBD, FD
BOOL OK
*
  GET\FILENAME(APL\FILE,PARAM(2):RETURN)
  IF LEGAL(PAIR_INTEGER\VECTOR(PARAM(1))) THEN
    IF PAIR[1]$NELTS=2 THEN
      IN\BOX_PAIR[3]$APL\VALUE
      OUT\BOX_PAIR[4]$APL\VALUE
      IF BOX\EXISTS(IN\BOX) AND BOX\EXISTS(OUT\BOX) THEN
        IF IN\BOX#OUT\BOX THEN
          INMBD_@MAILBOX\TABLE[IN\BOX]
          OUTMBD_@MAILBOX\TABLE[OUT\BOX]
          IF (INMBD.MBD\ASTAT=MBD\UNATTACHED) AND ..
            (OUTMBD.MBD\ASTAT=MBD\UNATTACHED) THEN
            FN_GET\DESCR(FILE\TABLE:-1)
            IF FN>=0 THEN
              FD_@FILE\TABLE[FN]
              OK_TRUE
              OPEN\WINDOWED(FD.FD\CAPX,APL\FILE:OK_FALSE)
              IF OK THEN
                *
                INMBD.MBD\ASTAT_MBD\FILE
                INMBD.MBD\AOBJ_FN
                INMBD.MBD\PUTM_PUTM\FILE
                INMBD.MBD\GETM_-1
                *
                OUTMBD.MBD\ASTAT_MBD\FILE
                OUTMBD.MBD\AOBJ_FN
                OUTMBD.MBD\PUTM_-1
                OUTMBD.MBD\GETM_GETM\FILE
                *
                FILE\SIZE __ READ\LENGTH ( APL\FILE )
                SETS(@FD.FD\RING,1,(FD.FD\VEC)[0] )
                FD.FD\RINGSVDSIZE_FILE\SIZE

```

```

        FD.FD\IN\BOX_IN\BOX
        FD.FD\OUT\BOX_OUT\BOX
        ENQUEUE\MESSAGE(FD.FD\IN\MSG,IN\BOX)
        PAIR\OFF(IN\BOX)
        *
        EAT(3)
        ATTN\RESUME(TRAP\PD)
    ELSE
        BADNEWS(FILE\OPENING\ERR)
        FREE\DESCR(FD)
    ENDIF
ELSE
    BADNEWS(FULL\ERR)
ENDIF
ELSE
    BADNEWS(BOX\ATTACH\ERR)
ENDIF
ELSE
    BADNEWS(SAME\BOX\ERR)
ENDIF
ELSE
    BADNEWS(NO\OBJ\ERR)
ENDIF
ELSE
    BADNEWS(PARAM\ERR)
ENDIF
ELSE
    BADNEWS(PARAM\ERR)
ENDIF
RETURN
END

```

FUNCTION ATTACH\TERMINAL()

*

* ATTACH SPECIFIED TERMINAL TO SPECIFIED PAIR OF MAILBOXES

*

SCALAR TERM, TN, IN\BOX, OUT\BOX

VECTOR PAIR

REFERENCE TD, MBD

FIELD NELTS (0: 16,31)

*

IF LEGAL (PAIR _ INTEGER\VECTOR (PARAM (1))) DO

IF PAIR [1] \$ NELTS = 2 DO

IN\BOX _ PAIR [3] \$ APL\VALUE

OUT\BOX _ PAIR [4] \$ APL\VALUE

IF BOX\EXISTS (IN\BOX) AND BOX\EXISTS (OUT\BOX) DO

IF LEGAL (TERM _ SINGLE\INTEGER (PARAM (2))) DO

TERM _ TERM \$ APL\VALUE

IF TERM < 0 OR TERM > 31 DO

BADNEWS (NO\TERM\ERR)

RETURN

ENDIF

TN _ GET\DESCR (TERM\TABLE :-1)

IF TN < 0 DO

BADNEWS (FULL\ERR)

ELSE DO

EAT (3)

INITTERM (TERM)

*

TD _ @ TERM\TABLE [TN]

TD . TD\PORT _ TERM

TD . TD\IN\BOX _ IN\BOX

TD . TD\OUT\BOX _ OUT\BOX

TD . TD\MVEC _ NULL\MSG

*

INIT\BOX (IN\BOX)

MBD _ @ MAILBOX\TABLE [IN\BOX]

MBD . MBD\ACTION _ FALSE

MBD . MBD\ASTAT _ MBD\TERM

MBD . MBD\AOBJ _ TN

MBD . MBD\PUTM _ PUTM\QUAD

MBD . MBD\GETM _ -1

*

INIT\BOX (OUT\BOX)

MBD _ @ MAILBOX\TABLE [OUT\BOX]

MBD . MBD\ASTAT _ MBD\TERM

MBD . MBD\AOBJ _ TN

MBD . MBD\PUTM _ -1

MBD . MBD\GETM _ GETM\CONV

ATTN\RESUME (TRAP\PD)

ENDIF

ELSE DO

BADNEWS (PARAM\ERR)

ENDIF

ELSE DO

BADNEWS (NO\OBJ\ERR)

```
ENDIF
ELSE DO
    BADNEWS ( PARAM\ERR )
ENDIF
ELSE DO
    BADNEWS ( PARAM\ERR )
ENDIF
RETURN
END
```

```

FUNCTION ATTACH\TIMER ( )
*
*  SERVICE CALL :
*    PARAM: SINGLE INTEGER OF MAILBOX NUMBER
*    ACTION: SET ATTACH STATUS TO MBD\TIMER
*
SCALAR MBOX
REFERENCE MBD
*
  IF LEGAL(MBOX _ SINGLE\INTEGER(PARAM(1))) THEN
    MBOX _ MBOX$APL\VALUE
    IF BOX\EXISTS(MBOX) THEN
      IF BOX\UNATTACHED(MBOX) AND BOX\EMPTY(MBOX) THEN
        MAILBOX\TABLE[MBOX]$MBD\ASTAT _ MBD\TIMER
        EAT(2)
        ATTN\RESUME(TRAP\PD)
      ELSE
        BADNEWS(BOX\ATTACH\ERR)
      ENDIF
    ELSE
      BADNEWS(NO\OBJ\ERR)
    ENDIF
  ELSE
    BADNEWS(PARAM\ERR)
  ENDIF
  RETURN
END

```

```
FUNCTION ATTN\ERR ( N )  
*MISSED ATTN TRAP ( SHOULD NEVER HAPPEN  
  MESSAGE ( "SYSTEM ERROR = 4,0&M" )  
  RETURN  
END
```

```

FUNCTION ATTN\RESUME ( REFERENCE PD )
*
* RESTART APL PROCESS AFTER AN ATTN TRAP
*
    VECTOR DTSEG
    SCALAR LBASE, SBASE
    REFERENCE STATE\AREA, STACK\FRAME
    *
    DTSEG _ PD . PD\DTSEG
    STATE\AREA _ FIND\STATE ( DTSEG )
    LBASE _ STATE\AREA . SA\LBASE
    SBASE _ STATE\AREA . SA\SBASE
    STACK\FRAME _ SETREFSIZE ( @ DTSEG [ SBASE + LBASE ], SF\DSIZE )
    STACK\FRAME . SF\PCTR _ STACK\FRAME . SF\PCTR + 1
    *
    APL\ON ( PD . PD\PRCAP )
    *
    RETURN
END

```

```

FUNCTION BADBLKNO ( SCALAR FD )
*
* CHECKS VALIDITY OF FUNCTION-DESCRIPTOR FD
*
  VECTOR SEG
  SCALAR B, MAXB
  *
  IF FD $ FNRUNT = 1 DO
    SEG _ TXSEG
  ELSE DO
    SEG _ RTXSEG
  ENDIF
  MAXB _ DIRSIZE ( SEG ) - 1
  B _ FD $ FNBLK
  RETURN ( B < 2 OR MAXB < B )
END

```

```
FUNCTION BADINFO ( VECTOR S, SCALAR F )
*
* TEST VALIDITY OF DEBUGGER BLOCK INFO FOR FUNCTION F
* OF DEBUG SEGMENT S
*
  IF S $ VDCAP = DBSEG $ VDCAP DO
    IF INVALID ( GLOBLK ) OR INVALID ( F ) DO
      RETURN TRUE
    ENDIF
  ENDIF
  RETURN FALSE
END
```

```
FUNCTION BADNEWS ( ERRNUM )
*
* SETS ERROR STATUS OF TRAPPED PROCESS
* ( MAY LATER BE REPLACED WITH VERSION WHICH GENERATES
*   A "REAL" TRAP, WHICH COULD BE CAUGHT BY OFFENDING PROCESS. )
*
  TRAP\PD . PD\TRAPCLASS _ CALL\ERR
  TRAP\PD . PD\TRAPNUMBER _ ERRNUM
  RETURN
END
```

```

FUNCTION BLKEXPAND(VECTOR SEG,BLK,INCR,MV)

*EXPAND BY INCR WORDS THE SPACE ALLOCATED TO BLOCK BLK OF
*   SEGMENT SEG
*IF MV = 1 THEN MOVE CONTENTS TO END OF EXPANDED BLOCK
* IN ALL CASES, LEAVE BLKBASE AND BLKSIZE OF BLK UNCHANGED

    DI TEMPBASE,OLDBASE,NEWBASE

    SEGSIZE(SEG) _ SEGSIZE(SEG) + INCR
    TEMPBASE _ BLKBASE(SEG,BLK)
    FOR I _ DIRSIZE(SEG)-1 BY -1 TO BLK+1-MV DO
        OLDBASE_BLKBASE(SEG,I)
        NEWBASE_OLDBASE+INCR
        VCOPY(SUBVEC(SEG,NEWBASE),SUBVEC(SEG,OLDBASE), ..
            BLKSIZE(SEG,I))
        BLKBASE(SEG,I)_NEWBASE
    ENDFOR
    BLKBASE(SEG,BLK) _ TEMPBASE
    RETURN

END

```

FUNCTION BLKSPACE(VECTOR SEG,I)

*RETURNS SIZE OF SPACE ALLOCATED FOR BLOCK I OF SEGMENT SEG

```
    RETURN (IF I+1 # DIRSIZE(SEG) ..  
            THEN BLKBASE(SEG,(I+1)) ..  
            ELSE SEGSIZE(SEG)) - BLKBASE(SEG,I)  
END
```

```
FUNCTION BOX\EMPTY ( MBOX )
*
* TEST MAILBOX NUMBER 'MBOX' FOR EMPTY MESSAGE QUEUE
*
    RETURN MSG\NULL ( MAILBOX\TABLE [ MBOX ] $ MBD\MQHD )
END
```

```
FUNCTION BOX\EXISTS ( MBOX )
*
* CHECK FOR EXISTENCE OF MAILBOX NUMBER 'MBOX'
*
  IF MBOX < 0 OR MAXBOXES-1 < MBOX DO
    RETURN FALSE
  ELSEIF MAILBOX\TABLE [ MBOX ] $ D\FREE DO
    RETURN FALSE
  ELSE DO
    RETURN TRUE
  ENDIF
END
```

```
FUNCTION BOX\UNATTACHED(MBOX)
*
*RETURN TRUE IF BOX IS UNATTACHED
*
  IF (@MAILBOX\TABLE[MBOX]).MBD\ASTAT=MBD\UNATTACHED THEN
    RETURN TRUE
  ELSE
    RETURN FALSE
  ENDIF
END
```

FUNCTION BUFINIT()

*INITIALIZE ALL BUFFERS

FOR I _ 0 TO NBUF-1 DO

BUFSIZE(I) _ 0

ENDFOR

RETURN

END

```
FUNCTION BUFSET ( VECTOR SEG, BLK, BUF )
```

```
* SETUP BUFFER 'BUF' TO HOLD BLOCK 'BLK' OF SEGMENT 'SEG'
```

```
    BUFBASE [ BUF ] _ SUBVEC ( SEG, BLKBASE ( SEG, BLK ), BLKSIZE ( SEG, BI
```

```
    BUFSEG [ BUF ] _ SEG
```

```
    BUFBLK [ BUF ] _ 0
```

```
    BUFREF [ BUF ] _ REFVEC ( BUFBASE [ BUF ] )
```

```
    RETURN
```

```
END
```

```

FUNCTION CATCH\TRAP()
*
* CATCH APL TRAP, IF ANY
*
  SCALAR T
  VECTOR TC
  REFERENCE STATE\AREA
  VECTOR DTSEG
  *
  IF APL\TRAP ( TRAP\CAP ) DO
    TC _ DISPLAY\CAP ( TRAP\CAP )
    TRAP\PN _ PFIND ( TC $ CAP\UNIQUE\NAME )
    TRAP\PD _ @ PROC\TABLE [ TRAP\PN ]
    DTSEG _ TRAP\PD.PD\DTSEG
    STATE\AREA _ FIND\STATE ( DTSEG )
    T _ STATE\AREA.SA\TRAPWORD
    RETURN T
  ELSE DO
    RETURN 0
  ENDIF
END

```

```

-----
FUNCTION CBPROC()
*
* CLEAR-BREAKPOINT COMMAND
*
*     NOTE: DOES NOT FIX SOURCE
*           DOES NOT HANDLE MULT LINES
*
  CMDMOVE ( IPTR1 :FRETURN )
  *
  UNLESS LTEGET ( DBSEG, IPTR1.BLOCK, IPTR1.LINE :FRETURN ) DO FRETURN
  IF LTE $ LTEBREAK = 0 DO
    ERRMSG ( "NO BREAKPOINT SET&M" )
  ELSE DO
    LTE $ LTEBREAK _ 0
    OUTBREAK ( IPTR1.BLOCK, LTE )
    LTEPUT ( DBSEG, IPTR1.BLOCK, IPTR1.LINE :CRASH() )
  ENDIF
  RETURN
END

```

```
FUNCTION CLASS(SCALAR CH) RETURNING SCALAR
***** THIS IS THE BEGINNING OF *****
***** CONVERT STRING TO NUMBER *****
    IF CH='0' THEN RETURN 0
    IF CH>='1' AND CH<='9' THEN RETURN 1
    IF CH='-' THEN RETURN 2
    IF CH='.' THEN RETURN 3
    IF CH='E' THEN RETURN 4
    RETURN 5
END
```

```
FUNCTION CMAT ( STRING MSTR, CHAR )
*
* APPEND CHARACTER 'CHAR' TO STRING 'MSTR'.
*
FIELD BYTE4 ( 0: 24,31 )
    IF CHAR $ APL\TYPE # TYPE\CHAR THEN FRETURN
    WCI ( CHAR $ BYTE4, MSTR )
    RETURN
END
```

```

FUNCTION CMDARG ( STRING S )
*
* CMDARG GETS THE NEXT ARGUMENT IN THE CURRENT COMMAND
* AND RETURNS IT IN S.
*
  BOOL INSTRING
  SCALAR CHAR
  *
  INSTRING _ FALSE
  CLEAR ( S )
  CHAR _ GNBCI ( CSTRING :RETURN )
  *
  UNTIL (( CHAR = BLANK ) AND NOT INSTRING ) DO
    IF CHAR = '"' OR CHAR = ':' DO
      INSTRING _ NOT INSTRING
    ENDIF
    WCI ( CHAR, S : FRETURN )
    CHAR _ GCI ( CSTRING :RETURN )
  ENDUNTIL
  RETURN
  *
END

```

```

FUNCTION CMDFN( )
*
* FINDS USER PROGRAM FUNCTION NAMED BY NEXT PARAMETER
*
  SCALAR FNUM
  STRING FNAME [ IDLENGTH ]
  *
  CMDARG ( FNAME )
  FNUM _ FNFIND ( FNAME :FRETURN )
  IF FNUM < 0 DO
    FRETURN
  ELSE DO
    RETURN FNUM
  ENDIF
END
END

```

```

FUNCTION CMDMOVE ( REFERENCE P )
*
* MOVE POINTER 'P' TO LINE SPECIFIED IN COMMAND
* (NOTE: KLUDGE VERSION OF EDITOR ROUTINE OF SMAE NAME )
*
  SCALAR C, FD
  STRING S [ IDLENGTH ]
  *
  CLEAR ( S )
  GCI ( CSTRING :FRETURN )
  UNTIL ( ( C _ GCI ( CSTRING :FRETURN ) ) = '[' ) DO
    WCI ( C, S )
  ENDUNTIL
  FD _ FNFIND ( S: FRETURN )
  IF FD < 0 DO FRETURN
  P . BLOCK _ FD $ FNBLK
  P . LINE _ CSN ( CSTRING :FRETURN ) + 1
  RETURN TRUE
END

```

```

FUNCTION COMMANDS()
*
* COMMANDS COLLECTS A TERMINAL LINE AND DECODES IT AS A COMMAND.
* IT THEN CALLS THE APPROPRIATE ROUTINE IN THE
* EDITOR, COMPILER, OR DEBUGGER.
*
*     VERSION 1.0
*
SCALAR I, CMD
BOOL SUCCESS
STRING CMDLINE
STRING CMDBUF [ LSZ ]
*
* COMMAND TABLES
*
CONSTANT NCMDS _ 12
VECTOR CMDS [ NCMDS ] _ ..
    'SP', 'SR', 'SB', 'CB', 'DO', ..
    'GO', 'PC', 'ST', 'EX', 'PV',..
    'DB', 'SZ'
*
VECTOR CMDPROC [ NCMDS ] _ ..
    SPPROC, SRPROC, SBPROC, CBPROC, DOPROC, ..
    GOPROC, PCPROC, STPROC, EXPROC, PVPROC,..
    DBPROC, SZPROC
*

```

```

*
* COMMAND LOOP
*
ERRFLAG _ FALSE
SUCCESS _ FALSE
PROMPT()
IF TEST\INPUT\EMPTY() DO RETURN
EDITLINE ( CMDBUF :GOTO BADCOM )
PROMPTED _ FALSE
SDCOPY ( CMDLINE, CMDBUF )
GCD ( CMDLINE )
WHILE LENGTH ( CMDLINE ) > 0 DO
    SUCCESS _ FALSE
    PEEL ( CMDLINE )
    *
    CMD _ 0
    FOR I _ 1 TO 2 DO
        CMD _ OR ( CMD LSH 8, GCI ( CSTRING :GOTO BADCOM ) )
    ENDFOR
    *
    FOR I _ 0 TO NCMDS - 1 DO
        IF CMD = CMDS [ I ] DO
            CMDPROC [ I ] ( :GOTO BADCOM )
            SUCCESS _ TRUE
        ENDIF
    ENDFOR
BADCOM:
    IF NOT SUCCESS DO
        ERRMSG ( "INVALID COMMAND&M" )
    ENDIF
    IF ERRFLAG DO CLEAR ( CMDLINE )
ENDWHILE
RETURN
END

```

```
FUNCTION CRASH()  
*  
* FATAL ERROR  
*  
    MESSAGE ( "CRASH!" ); BREAK  
END
```

```

FUNCTION CREATE\FILE()
*
*SERVICE CALL:
*  PARAM: APL CHARS OF FILE NAME
*  ACTION: GET NAME FROM PARAM, CREATE FILE WITH NAME.
*  FAILIF: ILLEGAL NAME; EXISTING FILE WITH SAME NAME.
*
STRING APL\FILE[SEGNAME\LEN]
BOOL OK
*
  GET\FILENAME(APL\FILE,PARAM(PAGE\SIZE):RETURN)
  OK_TRUE
  CREATE\SEGMENT(APL\FILE, 1:OK_FALSE)
  IF OK THEN
    OPEN\WINDOWED ( FILE\CAPX, APL\FILE : OK _ FALSE )
    IF OK THEN
      FILE\VEC [ 0 ] _ 1
      EAT(2)
      ATTN\RESUME(TRAP\PD)
    ELSE
      BADNEWS ( FILE\OPENING\ERR )
    ENDIF
  ELSE
    BADNEWS(DUPNAME\ERR)
  ENDIF
  RETURN
END

```

```

FUNCTION CREATE\MAILBOX( )
*
* SERVICE CALL: CREATE A MAILBOX, CAPABLE OF SERVICING
* A SPECIFIED MAXIMUM NUMBER OF RECEIVERS
*
  SCALAR MBOX, RMAX
  BOOL FULL
  VECTOR RQVEC
  REFERENCE MBRQ
  *
  IF LEGAL ( RMAX _ SINGLE\INTEGER ( PARAM ( 1 ) ) ) DO
    FULL _ FALSE
    MBOX _ GET\DESCR ( MAILBOX\TABLE :FULL _ TRUE )
    IF FULL DO
      BADNEWS ( FULL\ERR )
    ELSE DO
      EAT ( 2 )
      MBRQ _ @ ( MAILBOX\TABLE [ MBOX ] $ MBD\RCVQ )
      SMAKE\STRING ( MBRQ, RMAX $ APL\VALUE )
      INIT\BOX ( MBOX )
      MBOX _ APL\INTEGER ( MBOX )
      PUSHWORD ( MBOX, TRAP\PD )
      ATTN\RESUME ( TRAP\PD )
    ENDIF
  ELSE DO
    BADNEWS ( PARAM\ERR )
  ENDIF
  RETURN
END

```

FUNCTION CREATE\PROCESS()

*

* CREATE APL PROCESS, STARTING SPECIFIED FUNCTION AND PASSING IT:
* THE EXTRA ARGUMENT 'PARM' SPECIFIED BY THE CALLER
* AFTER FIRST CALLING THE APL 'INIT' FUNCTION, PASSING IT THE
* PROCESS ID AND THE VECTOR 'STATE', ALSO SPECIFIED BY THE CALLER.
*

SCALAR FD, MAXB, NEW\PN, APL\PN, PARM, STATE

BOOL BADMAX

REFERENCE NEW\PD

*

IF LEGAL (FD _ SINGLE\INTEGER (PARAM (4))) DO

FD _ FD \$ APL\VALUE

IF BADBLKNO (FD) OR (FNHEADER(FD)) \$ FH\NARGS # 1 DO

BADNEWS (PARAM\ERR)

ELSE DO

IF LEGAL (MAXB _ SINGLE\INTEGER (PARAM (2))) DO

NEW\PN _ GET\DESCR (PROC\TABLE :-1)

IF NEW\PN < 0 DO

BADNEWS (FULL\ERR)

ELSE DO

NEW\PD _ @ PROC\TABLE [NEW\PN]

APL\PN _ APL\INTEGER (NEW\PN)

BADMAX _ FALSE

PROCSETUP (NEW\PN, MAXB \$ APL\VALUE :BADMAX _ TRUE)

IF BADMAX DO

FREE\DESCR (NEW\PD)

BADNEWS (PARAM\ERR)

ELSE DO

*

* MESSAGE NEW PROCESS

*

DTINIT (NEW\PD)

PARM _ PARAM (1)

OBJPUSH (PARM, NEW\PD, TRAP\PD)

FACPUSH (FD, NEW\PD)

STATE _ PARAM (3)

OBJPUSH (STATE, NEW\PD, TRAP\PD)

FACPUSH (INITPFD, NEW\PD)

APL\ON (NEW\PD . PD\PRCAP)

*

* MESSAGE CALLING PROCESS

*

EAT (5)

PUSHWORD (APL\PN, TRAP\PD)

ATTN\RESUME (TRAP\PD)

ENDIF

ENDIF

ELSE DO

BADNEWS (PARAM\ERR)

ENDIF

ENDIF

ELSE DO

BADNEWS (PARAM\ERR)

```
FUNCTION CRECEIVE()  
    CRASH()  
END
```

```
FUNCTION DBPROC()  
*  
* DB-COMMAND: DEBUG SPECIFIED PROCESS  
*  
  SCALAR P  
  *  
  P _ CSN ( CSTRING :FRETURN )  
  DB\SET ( P: FRETURN )  
  *  
  RETURN  
END
```

```

FUNCTION DB\SET ( P )
*
* SET DEBUGGEE POINTER
*
    VECTOR DTSEG
    REFERENCE SA
    *
    UNLESS ( PROC\TABLE [ P ] $ PD\STOPPED ) DO FRETURN
    IF P < 0 OR MAXPROCS-1 < P DO
        FRETURN
    ELSEIF PROC\TABLE [ P ] $ D\FREE DO
        FRETURN
    ELSE DO
        DB\PN _ P
        DB\PD _ @ PROC\TABLE [ P ]
        DTSEG _ DB\PD . PD\DTSEG
        SA _ FIND\STATE ( DTSEG )
        DB\ENV _ @ DTSEG [ SA . SA\SBASE + SA . SA\LBASE ]
        DB\ENV _ SETREFSIZE ( DB\ENV, SF\DSIZE )
    ENDIF
    RETURN
END

```

```
FUNCTION DEBUG\TRAP ( N )
```

```
* DEBUGGING TRAP
```

```
    IF N = 4 DO
```

```
        MESSAGE ( "BREAKPOINT&M" )
```

```
    ELSE DO
```

```
        SOUT ( "SYSTEM ERROR = 2," )
```

```
        IOUT ( N ); CRLF()
```

```
    ENDIF
```

```
    RETURN
```

```
END
```

```

FUNCTION DECIM(STRING STR, SCALAR NUM, SCALAR SIGN, SCALAR W,  ..
                SCALAR TRIM : TRUE )
*32 BIT UNSIGNED SIMPLE NUMBER OUTPUT AS DECIMAL
SCALAR N
STRING STR1[10]
  N_0
  SETS(STR1,0,0)
  WHILE BTU NUM>0 DO
    N_N+1
    WCD(NUM-NUM/10*10+'0',STR1 :FRET(2))
    NUM_NUM/10
  ENDWHILE
  IF INT\WIDTH < N THEN INT\WIDTH _ N
  OUTSTR(STR,STR1,SIGN,W,N,TRIM :FRET(2))
  RETURN
END

```

```
FUNCTION DELIVER()  
*  
* SERVICE CALL: DELIVER MESSAGE TO SPECIFIED MAILBOX(ES)  
*  
    SEND\DELIVER ( TRUE )  
    RETURN  
END
```

```

FUNCTION DELIVERY\CHECK ( VECTOR MVEC )
*
* IF MESSAGE 'MVEC' IS BEING DELIVERED, WAKE UP THE
* SOURCE PROCESS, IF APPROPRIATE
*
    SCALAR SPN
    REFERENCE MHDR, SPD
    *
    MHDR _ MSG\HDR ( MVEC )
    IF MHDR . MSG\DELIVER DO
        SPN _ MHDR . MSG\SOURCE
        SPD _ @ PROC\TABLE [ SPN ]
        IF ( SPD . PD\DCNT _ SPD . PD\DCNT - 1 ) = 0 DO
            ATTN\RESUME ( SPD )
            IF DB\PN = SPN DO NO\DB
                SPD . PD\STOPPED _ FALSE
            ENDIF
        ENDIF
    ENDIF
    RETURN
END

```

```

FUNCTION DEQUEUE\MESSAGE ( MBOX ) RETURNING VECTOR
*
* DEQUEUE 1ST MESSAGE-VECTOR FROM HEAD OF MESSAGE-QUEUE
* OF MAILBOX NUMBER 'MBOX' AND RETURN IT AS VALUE.
* FAILS IF MESSAGE-QUEUE IS EMPTY
*
  REFERENCE MBD, MHDR
  VECTOR MVEC, NEWHD
  *
  MBD _ @ MAILBOX\TABLE [ MBOX ]
  MVEC _ MBD . MBD\MQHD
  IF MSG\NULL ( MVEC ) DO FRETURN
  *
  MHDR _ MSG\HDR ( MVEC )
  NEWHD _ MHDR . MSG\NEXT
  MBD . MBD\MQHD _ NEWHD
  IF MSG\NULL ( NEWHD ) DO; * IF QUEUE NOW EMPTY DO *
    MBD . MBD\MQTL _ NULL\MSG
  ENDIF
  RETURN MVEC
END

```

```

FUNCTION DEQUEUE\RECEIVER ( MBOX )
*
* REMOVE PROCESS FROM HEAD OF RECEIVER QUEUE
* OF MAILBOX NUMBER 'MBOX' AND RETURN ITS PROCESS
* NUMBER.
* FAILS IF RECEIVER QUEUE EMPTY
*
  SCALAR P, M
  REFERENCE MBRQ, PMBL
  *
  MBRQ _ @ ( MAILBOX\TABLE [ MBOX ] $ MBD\RCVQ )
  P _ GCNI ( MBRQ :FRETURN )
  PMBL _ @ ( PROC\TABLE [ P ] $ PD\MBOXLIST )
  *
  * REMOVE PROCESS 'P' FROM ALL RECEIVER QUEUES
  *
  WHILE ( M _ GCI ( PMBL :-1 ) ) >= 0 DO
    REMOVE ( P, M )
  ENDWHILE
  RETURN P
END

```

```

FUNCTION DESTROY\FILE()
*
*SERVICE CALL:
*  PARAM: APL CHARS OF FILE NAME
*  ACTION: GET NAME FROM PARAM, DESTROY NAMED FILE.
*  FAILIF: ILLEGAL NAME; FILE IS OPEN(I.E. BOX ATTACHED TO IT)
*
STRING APL\FILE[SEGNAME\LEN]
BOOL OK
*
  GET\FILENAME(APL\FILE,PARAM(1):RETURN)
  OK_TRUE
  DELETE\SEGMENT(APL\FILE:OK_FALSE)
  IF OK THEN
    EAT(2)
    ATTN\RESUME(TRAP\PD)
  ELSE
    BADNEWS(FILEOPEN\ERR)
  ENDIF
  RETURN
END

```

```
FUNCTION DESTROY\MAILBOX()  
*  
* DUMMY  
*  
    CRASH()  
END
```

```
FUNCTION DESTROY\PROCESS()  
*  
* DUMMY  
*  
    CRASH()  
END
```

```
FUNCTION DETACH()  
*  
* DUMMY  
*  
    CRASH()  
END
```

```
FUNCTION DETACH\FILE(FILENUM)
*
*CLOSE FILE USING CAPABILITY IN FILE ENTRY NUMBER 'FILENUM'
*DEQUEUE PHONEY INPUT MESSAGE
*RELEASE FILE TABLE ENTRY
*
REFERENCE FD
*
  FD_@FILE\TABLE[FILENUM]
  CLOSE(FD.FD\CAPX)
  DEQUEUE\MESSAGE(FD.FD\IN\BOX)
  FREE\DESCR(FD)
  RETURN
END
```

```

FUNCTION DETACH\TIMER ( TBOX )
*
* FRETURN IF : NOT TIMER BOX, ALARM PENDING, MESSAGE PENDING
*
REFERENCE MBD
SCALAR OLDLIST
STRING T\LIST [MAXBOXES 2 WORD]
*
  MBD _ @MAILBOX\TABLE[TBOX]
  IF MBD.MBD\ASTAT = MBD\TIMER AND BOX\EMPTY(TBOX) THEN
    OLDLIST _ TIMER\LIST.SDRS
    IF FIND\TBOX( TBOX , T\LIST) THEN
      TIMER\LIST.SDRS _ OLDLIST
      FRETURN
    ELSE
      TIMER\LIST.SDRS _ OLDLIST
    ENDIF
  ELSE
    FRETURN
  ENDIF
  RETURN
END

```

```
FUNCTION DOMAIN\ERR ( N )
```

```
* DOMAIN ERROR
```

```
    SOUT ( "DOMAIN ERROR" )
```

```
    IF N = 1 OR N = 3 DO
```

```
        MESSAGE ( ": OVERFLOW&M" )
```

```
    ELSEIF N = 2 DO
```

```
        MESSAGE ( ": UNDERFLOW&M" )
```

```
    ELSEIF N = 4 DO
```

```
        MESSAGE ( ": DIVISION BY ZERO&M" )
```

```
    ELSE DO
```

```
        CRLF()
```

```
    ENDIF
```

```
    RETURN
```

```
END
```

```

FUNCTION DOPROC()
*
* DO-FUNCTION COMMAND: START EXECUTION OF DEBUGGEE PROCESS
*
    SCALAR FN, PRCAP, TN, PN
    REFERENCE PD
    STRING TERM [ 3 ]
    *
    FN _ CMDFN ( :FRETURN )
    IF ( FNHEADER ( FN ) ) $ FH\NARGS # 0 DO
        ERRMSG ( "WRONG NUMBER OF ARGUMENTS&M" )
        RETURN
    ENDIF
    *
    CMDARG ( TERM : FRETURN )
    IF LENGTH ( TERM ) = 0 DO
        TN _ 17
    ELSE
        TN _ CSN ( TERM : FRETURN )
    ENDIF
    *
    OBJINIT()
    PN _ GET\DESCR ( PROC\TABLE )
    PD _ @PROC\TABLE [ PN ]
    PROCSETUP ( PN, MAXBOXES :FRETURN )
    DTINIT ( PD )
    * PUSH USER FUNCTION ON STACK
    FACPUSH ( FN, PD )
    * PUSH "@INIT0 TN" ONTO STACK
    PUSHWORD ( APL\INTEGER ( TN ), PD )
    FACPUSH ( INITOPFD, PD )
    *
    * RUN THE FUNCTION
    *
    PRCAP _ PD . PD\PRCAP
    APL\ON ( PRCAP )
    *
    RETURN
END

```

```

FUNCTION DROP()
*
* DROP CLOSSES ALL SEGMENTS OF THE CURRENT PROGRAM (IF ANY)
* AND SETS ALL POINTERS TO NULL.
*
  SCALAR I
  REFERENCE P
  *
  CLEAR ( PROGNAME )
  *
  IF LOADED DO
    CLOSE ( TXSEG $ VDCAP )
    CLOSE ( CDSEG $ VDCAP )
    CLOSE ( DBSEG $ VDCAP )
  ENDIF
  *
  FOR I _ 0 TO NPTRS-1 DO
    MAKENULL ( PTRADDR (I) )
  ENDFOR
  *
  LOADED _ FALSE
  RETURN
END

```

```

FUNCTION DTINIT ( REFERENCE PD )
*
* INITIALIZE DATA SEGMENT
*
    REFERENCE STATE\AREA, BLKHDR
    VECTOR DTSEG
    SCALAR BIGBLOCK, DUMMYBLOCK
    MACRO BLKREF ( B ) _ SETREFSIZE ( @ DTSEG [ (B) ], 2 )
    FIELD ABLK\FREE ( 0:0,0 ), ABLK\SIZE ( 0: 16,31 ), ..
        ABLK\FPTR ( 1: 0,15 ), ABLK\BPTR ( 1: 16,31 )
    *
    DTSEG _ PD . PD\DTSEG
    STATE\AREA _ FIND\STATE ( DTSEG )
    STATE\AREA . SA\SBASE _ GVSIZE + ABSIZE
    STATE\AREA . SA\LTOP _ 0
    STATE\AREA . SA\LBASE _ 0
    STATE\AREA . SA\FLAGS _ 1;      * INDEX ORIGIN = 1 *
    *
    * SET UP ARRAY BLOCK STORAGE WITH TWO BLOCKS ON FREE CHAIN
    *
    BIGBLOCK _ GVSIZE
    DUMMYBLOCK _ BIGBLOCK + ( ABSIZE - 2 )
    *
    BLKHDR _ BLKREF ( BIGBLOCK )
    BLKHDR . ABLK\FREE _ TRUE
    BLKHDR . ABLK\SIZE _ DTSEG [ DUMMYBLOCK - 1 ] _ ABSIZE - 2
    BLKHDR . ABLK\FPTR _ BLKHDR . ABLK\BPTR _ DUMMYBLOCK
    *
    BLKHDR _ BLKREF ( DUMMYBLOCK )
    BLKHDR . ABLK\FREE _ FALSE
    BLKHDR . ABLK\SIZE _ 0
    BLKHDR . ABLK\FPTR _ BLKHDR . ABLK\BPTR _ BIGBLOCK
    *
    STATE\AREA . SA\ROVER _ BIGBLOCK
    *
    * INITIALIZE GLOBAL VARIABLES
    *
    FOR I _ SA\SIZE TO GVSIZE-1 DO
        DTSEG [ I ] _ APL\UNDEFINED
    ENDFOR
    *
    RETURN
END

```

```
FUNCTION DW(String STR, SCALAR NUM, SCALAR W:11)
*32 BIT SIGNED SIMPLE NUMBER OUTPUT AS DECIMAL
  DECIM(STR,ABS(NUM),NUM$SIGNF,W,FALSE:FRET(3))
  RETURN
END
```

```

FUNCTION EAT ( N )
*
* EAT N WORDS FROM TOP OF STACK OF TRAPPED PROCESS. IF A
* WORD IS AN ARRAY DESCRIPTOR, DECREMENT REFERENCE
* COUNT OF ARRAY BLOCK, WHICH MUST NOT BECOME ZERO.
* FAILS: IF ANY REFERENCE COUNT BECOMES ZERO
*
  SCALAR LTOP, SBASE, W, A, I
  VECTOR DTSEG
  REFERENCE STATE\AREA
  FIELD ADDRESS ( 0:16,31 ), REFCNT ( 0: 0,15 )
  *
  DTSEG _ TRAP\PD . PD\DTSEG
  STATE\AREA _ FIND\STATE ( DTSEG )
  LTOP _ STATE\AREA . SA\LTOP
  SBASE _ STATE\AREA . SA\SBASE
  FOR I _ 0 TO N-1 DO
    W _ DTSEG [ LTOP + SBASE - I ]
    IF W $ APL\TYPE = TYPE\ARRAY DO
      A _ W $ ADDRESS + 1
      IF ( DTSEG[A] $ REFCNT _ DTSEG[A] $ REFCNT - 1 ) = 0 DO
        FRETURN
      ENDIF
    ENDIF
  ENDFOR
  STATE\AREA . SA\LTOP _ LTOP - N
  RETURN
END

```

```
FUNCTION ECHO ( C, L )
*
* ECHO CHAR 'C' ON LINE L, AS APPROPRIATE
* ( CURRENTLY, TTY_DRIVER ECHO-MODE = ON )
*
    IF C = CARRET DO
        CRLF ( 1, L )
    ENDIF
    RETURN
END
```

```

FUNCTION EDITLINE ( STRING L )
*
* READ LINE L FROM TERMINAL
*
  SCALAR CHAR
  *
  CLEAR ( L )
  *
  REPEAT
    CHAR _ CIN()
    IF CHAR = '&M' DO
      COUT ( '&J' )
      WCI ( CHAR, L :FRETURN )
      RETURN
    ELSEIF CHAR = '&Q' DO
      GCD ( L :0 )
      COUT ( ' _ ' )
    ELSEIF CHAR = '&Y' DO
      CLEAR ( L )
      COUT ( '^' )
      CRLF ( 1 )
      SOUT ( "      " )
    ELSE DO
      WCI ( CHAR, L :FRETURN )
    ENDIF
  ENDREPEAT
END

```

```

FUNCTION EMAT(STRING STR, SCALAR NUM, SCALAR W:15, SCALAR D:6, ..
              SCALAR EX:3, SCALAR TRIM :TRUE)
*APL NUMBER OUTPUT IN E OR TRIMED-E FORMAT
SCALAR EXP
SCALAR POW
SCALAR DIV
SCALAR DIG
SCALAR M, X
SCALAR SAVEDIV
FIELD E(0:24,31), L31(0:31,31)
FIELD F(0:1,23)
FIELD UP(1)
DOUBLE DPROD
  IF D<0 THEN D_0
  IF D>6 OR W<D+6 THEN FRET(60)
  NUM_FLOAT(NUM:FRET(61))
  IF NOT TRIM THEN
    FOR I_W-D-7 BY -1 TO 1 DO
      WCI(' ',STR :FRET(62))
    ENDFOR
  ENDIF
  IF NUM$SIGNF=0 THEN
    IF NOT TRIM THEN WCI(' ',STR :FRET(62))
  ELSE
    WCI('- ',STR :FRET(62))
  ENDIF
  EXP_NUM$E+EBIAS
  NUM _ NUM$F @ F
  IF EXP<-128 OR EXP>128 THEN FRET(64)
  IF EXP<0 THEN EXP _ 128-EXP
  DPROD_LMUL(NUM ,FRACTWO(EXP))
  NUM _ (DPROD$UP/10) SHIFT -2
  POW_EXPTWO(EXP) +1
  IF D+1<7 THEN
    M_1
    FOR I_D+1 BY -1 TO 1 DO
      M_10*M
    ENDFOR
    M_M-1
  ELSE
    M_9999999
  ENDIF
  DIG_0
  WHILE NUM>M DO
    DIG_NUM REM 10
    NUM_NUM/10
    POW_POW+1
  ENDWHILE
  IF DIG>5 OR (DIG=5 AND ((NUM$L31)=0)) THEN NUM_NUM+1
  DIV_1000000
  DIG_0
  IF NUM>0 THEN
    WHILE DIG=0 DO
      DIG_NUM/DIV

```

```

        DIV_DIV/10
    ENDWHILE
    SAVEDIV_DIV
    NUM_NUM-DIV*DIG*10
ELSE
    POW_0
    SAVEDIV_0
ENDIF
WCI(DIG+'0',STR:FRET(62))
WCI('.',STR:FRET(62))
FOR I_1 TO D DO
    IF DIV<0 THEN FRET(63)          ; * NEEDED IF D>6
    DIG_NUM/DIV
    WCI(DIG+'0',STR:FRET(62))
    NUM_NUM-DIV*DIG
    DIV_DIV/10
ENDFOR
IF TRIM THEN
    WHILE GCND(STR:FRET(62))='0' DO
        GCD(STR:FRET(62))
    ENDWHILE
    IF GCND(STR:FRET(62))='.' THEN GCD(STR:FRET(62))
ENDIF
WHILE SAVEDIV>0 DO
    POW_POW+1
    SAVEDIV_SAVEDIV/10
ENDWHILE
WCI('E',STR:FRET(62))
X _ LENGTH(STR)
IF POW>=0 THEN
    IF NOT TRIM AND EX=3 THEN
        WCI('0',STR:FRETURN)
        EX_EX-1
    ENDIF
ELSE
    WCI('-',STR:FRET(62))
    POW_-POW
    EX_EX-1
ENDIF
DIG_POW/10
IF DIG#0 OR( NOT TRIM AND EX >1 ) THEN WCI(DIG+'0',STR:FRET(62))
WCI(POW-DIG*10+'0',STR:FRET(62))
X _ LENGTH(STR) - X
IF EXP\WIDTH < X THEN EXP\WIDTH _ X
RETURN
END

```

```

FUNCTION ENQUEUE\MESSAGE ( VECTOR MVEC, MBOX )
*
* ENQUEUE MESSAGE-VECTOR 'MVEC' ON TAIL OF
* MESSAGE-QUEUE OF MAILBOX NUMBER 'MBOX'
*
REFERENCE MBD, MHDR, THDR
VECTOR TVEC
*
MBD _ @ MAILBOX\TABLE [ MBOX ]
TVEC _ MBD . MBD\MQTL
MBD . MBD\MQTL _ MVEC
*
MHDR _ MSG\HDR ( MVEC )
MHDR . MSG\NEXT _ NULL\MSG
*
IF MSG\NULL ( TVEC ) DO; * IF QUEUE WAS EMPTY DO *
    MBD . MBD\MQHD _ MVEC
ELSE DO
    THDR _ MSG\HDR ( TVEC )
    THDR . MSG\NEXT _ MVEC
ENDIF
RETURN
END

```

```

FUNCTION ENQUEUE\RECEIVER ( P, MBOX )
*
* ENQUEUE PROCESS NUMBER 'P' IN RECEIVER QUEUE
* OF MAILBOX NUMBER 'MBOX'
* FAILS IF QUEUES OVERFLOW
*
  REFERENCE PMBL, MBRQ
  *
  PMBL _ @ ( PROC\TABLE [ P ] $ PD\MBOXLIST )
  MBRQ _ @ ( MAILBOX\TABLE [ MBOX ] $ MBD\RCVQ )
  *
  WCI ( P, MBRQ :FRETURN )
  WCI ( MBOX, PMBL :GCD ( MBRQ ) & FRETURN )
  *
  PROC\TABLE [ P ] $ PD\STOPPED _ TRUE
  RETURN
END

```

```

FUNCTION ERRORS()
*
* CHECK FOR AN ERROR ( I.E. NON-ATTN TRAP ) BY ANY
* APL PROCESS AND PRINT MESSAGE IF ERROR HAS OCCURRED
*

REFERENCE PD, SA, SF
VECTOR DTSEG
SCALAR C, PC, FD, LN
CONSTANT NTRAPS _ 11
VECTOR ERR\PROC [ NTRAPS ONE\BASED ] _ STOVF\ERR, DEBUG\TRAP, ..
SYSTEM\ERR, ATTN\ERR, RANK\ERR, LENGTH\ERR, TYPE\ERR, ..
DOMAIN\ERR, INDEX\ERR, VALUE\ERR, RSCALL\ERR

FOR I _ 0 TO MAXPROCS-1 DO
  PD _ @ PROC\TABLE [ I ]
  UNLESS PD . D\FREE DO
    C _ PD . PD\TRAPCLASS
    IF C > 0 DO
      PD . PD\STOPPED _ TRUE
      DB\SET ( I )
      UNPROMPT()
      *
      DTSEG _ PD . PD\DTSEG
      SA _ FIND\STATE ( DTSEG )
      SF _ @ DTSEG [ SA . SA\LBASE + SA . SA\SBASE ]
      SF _ SETREFSIZE ( SF, SF\DSIZE )
      PC _ SF . SF\PCTR
      FD _ SF . SF\FNDESC
      LN _ LTEFIND ( FD $ FNRUNT, FD $ FNBLK, PC )

      PLADDR ( FD, LN )

      PD . PD\BREAK _ ( C = 2 AND PD . PD\TRAPNUMBER = 4 ..
                        AND LTE $ LTEFLAGS # 0 )

      *
      SOUT ( " IN PROCESS #" )
      IOUT ( I )
      SOUT ( ":" )
      ERR\PROC [ C ] ( PD . PD\TRAPNUMBER )
      CLEAR\TRAP ( PD )
    ENDIF
  ENDUNLESS
ENDFOR
RETURN
END

```

```
FUNCTION EWD(String STR, SCALAR NUM, SCALAR W:13, SCALAR D:6)
*APL NUMBER OUTPUT IN E FORMAT
    EMAT(STR, NUM, W, D, FALSE: FRET(62))
    RETURN
END
```

```
FUNCTION EXPAND\FILE(REFERENCE FD, WORDS)
```

```
*
```

```
*EXPAND SEGMENT SIZE IF TOO SMALL.
```

```
*
```

```
SCALAR SIZE
```

```
IF FD.FD\RINGSVDSIZE <= FD.FD\RING\OUT + WORDS THEN
```

```
FD.FD\RINGSVDSIZE_FD.FD\RINGSVDSIZE + PAGE\SIZE
```

```
SET\LENGTH(FD.FD\CAPX, FD.FD\RINGSVDSIZE )
```

```
ENDIF
```

```
RETURN
```

```
END
```

```
FUNCTION EXPROC()  
*  
* EXIT FROM LANGUAGE SYSTEM COMMAND  
*  
    DROP()  
    RTDROP()  
    MESSAGE ( "GOOD DAY&M" ); BREAK  
END
```

```

FUNCTION EXPDEN(INDEX)
*RETURN ETEN[INDEX]
VECTOR ETEN1[32 8 BIT] _ ..
    1,    4,    7,   10,   14,   17,   20,   24, ..
    27,   30,   34,   37,   40,   44,   47,   50, ..
    54,   57,   60,   64,   67,   70,   74,   77, ..
    80,   84,   87,   90,   94,   97,  100,  103
VECTOR ETEN4[16 8 BIT] _ ..
    107,  110,  113,  117,  120,  123,  127,  130, ..
    133,  137,  140,  143,  147,  150,  153,  157
VECTOR ETEN2[24 8 BIT] _ ..
    0,    3,    6,    9,   13,   16,   19,   23, ..
    26,   29,   33,   36,   39,   43,   46,   49, ..
    53,   56,   59,   63,   66,   69,   73,   76
VECTOR ETEN3[24 8 BIT] _ ..
    79,   83,   86,   89,   93,   96,   99,  102, ..
    106,  109,  112,  116,  119,  122,  126,  129, ..
    132,  136,  139,  142,  146,  149,  152,  156
VECTOR ETEN
CONSTANT VTENSIZE_96
    ETEN_ETEN1
    ETEN$VDSIZE_VTENSIZE
    RETURN ETEN[INDEX]
END

```

```

FUNCTION EXPTWO(INDEX)
*RETURN ETWO[INDEX]
VECTOR ETWO1[24 SIGNED 8 BIT] _ ..
    -9, -9, -8, -8, -8, -7, -7, -7, ..
    -6, -6, -6, -6, -5, -5, -5, -4, ..
    -4, -4, -3, -3, -3, -3, -2, -2
VECTOR ETWO2[40 8 BIT] _ ..
    -2, -1, -1, -1, 0, 0, 0, 1, ..
    1, 1, 1, 2, 2, 2, 3, 3, ..
    3, 4, 4, 4, 4, 5, 5, 5, ..
    6, 6, 6, 7, 7, 7, 7, 8, ..
    8, 8, 9, 9, 9, 10, 10, 10
VECTOR ETWO3[40 8 BIT] _ ..
    10, 11, 11, 11, 12, 12, 12, 13, ..
    13, 13, 13, 14, 14, 14, 15, 15, ..
    15, 16, 16, 16, 16, 17, 17, 17, ..
    18, 18, 18, 19, 19, 19, 19, 20, ..
    20, 20, 21, 21, 21, 22, 22, 22
VECTOR ETWO4[24 8 BIT] _ ..
    22, 23, 23, 23, 24, 24, 24, 25, ..
    25, 25, 25, 26, 26, 26, 27, 27, ..
    27, 28, 28, 28, 28, 29, 29, 29
VECTOR ETWO5[24 8 BIT] _ ..
    30, -9, -9, -10, -10, -10, -11, -11, ..
    -11, -12, -12, -12, -12, -13, -13, -13, ..
    -14, -14, -14, -15, -15, -15, -15, -16
VECTOR ETWO6[24 8 BIT] _ ..
    -16, -16, -17, -17, -17, -18, -18, -18, ..
    -18, -19, -19, -19, -20, -20, -20, -21, ..
    -21, -21, -21, -22, -22, -22, -23, -23
VECTOR ETWO7[24 8 BIT] _ ..
    -23, -24, -24, -24, -24, -25, -25, -25, ..
    -26, -26, -26, -27, -27, -27, -27, -28, ..
    -28, -28, -29, -29, -29, -30, -30, -30
VECTOR ETWO8[24 8 BIT] _ ..
    -31, -31, -31, -31, -32, -32, -32, -33, ..
    -33, -33, -34, -34, -34, -34, -35, -35, ..
    -35, -36, -36, -36, -37, -37, -37, -37
VECTOR ETWO9[24 8 BIT] _ ..
    -38, -38, -38, -39, -39, -39, -40, -40, ..
    -40, -40, -41, -41, -41, -42, -42, -42, ..
    -43, -43, -43, -43, -44, -44, -44, -45
VECTOR ETWO10[9 8 BIT] _ ..
    -45, -45, -46, -46, -46, -46, -47, -47, ..
    -47
VECTOR ETWO
CONSTANT VTWOSIZE_257
    ETWO_ETWO1
    ETWOSVDSIZE_VTWOSIZE
    RETURN ETWO[INDEX]
END

```

```

FUNCTION FACPUSH ( SCALAR F, REFERENCE PD )
*
* PUSHES ACTIVATION OF FUNCTION F ONTO STACK OF INDICATED PROCESS.
* EXPECTS ARGUMENTS TO BE ON STACK ALREADY.
*
  SCALAR NL, I, LASTLBASE
  REFERENCE STATE\AREA
  *
  STATE\AREA _ FIND\STATE ( PD . PD\DTSEG )
  IF STATE\AREA . SA\LBASE = 0 DO
    LASTLBASE _ FCMARKER
  ELSE DO
    LASTLBASE _ STATE\AREA . SA\LBASE
  ENDIF
  *
  * PUSH UNDEFINED LOCAL VARIABLES
  *
  NL _ ( FNHEADER ( F ) ) $ FH\NLOCALS
  FOR I _ 1 TO NL DO
    PUSHWORD ( APL\UNDEFINED, PD )
  ENDFOR
  *
  * PUSH FRAME-DESCRIPTOR
  *
  PUSHWORD ( LASTLBASE @ SF\LASTLBASE, PD )
  STATE\AREA . SA\LBASE _ STATE\AREA . SA\LTOP
  PUSHWORD ( F @ SF\FNDESC + INST1 @ SF\PCTR, PD )
  *
  RETURN
END

```

```

FUNCTION FILE\EXIST()
*
*  SERVICE CALL:
*    PARAM:    CHAR VECTOR OF NAME
*    ACTION:   RETURN TRUE(1) IF FILE EXIST, FALSE(0) OTHERWISE.
*
CONSTANT SEGNAME\LEN _ 8
STRING APL\FILE [ SEGNAME\LEN ]
SCALAR EXIST
*
  GET\FILENAME ( APL\FILE, PARAM ( 1 ) : RETURN )
  EXIST _ TRUE
  READ\LENGTH ( APL\FILE : EXIST _ FALSE )
  EAT ( 2 )
  PUSH\RESUME ( APL\INTEGER ( EXIST ) , TRAP\PD )
  RETURN
END

```

```

FUNCTION FIND\TBOX(TBOX,STRING T\LIST)
*
*SCAN FOR 'TBOX' IN 'TIMER\LIST' AND PUT ALL SCANNED ENTRIES IN 'T\LIST'
*RETURN TRUE IF FOUND ELSE FALSE
*
  CLEAR(T\LIST)
  WHILE( GDND(TIMER\LIST:RETURN FALSE))$TL\TBOX#TBOX DO
    WDD(GDD(TIMER\LIST), T\LIST)
  ENDWHILE
  RETURN TRUE
END

```

```

FUNCTION FIX( SCALAR NUM ) RETURNING SCALAR
* RETURN APL INT NUM FROM APL NUM
* FRETURN IF NOT APL NUM OR CANT BE CONVERTED
*
SCALAR SIGN, EXP
FIELD EXPF(0:24,31), REAL\VALUE(0:1,23), L31(0:31,31)
*
  IF NOT INTNUM(NUM:FRETURN) THEN
    EXP _ NUM$EXPF-150
    IF EXP > 0 OR EXP < -22 THEN FRETURN
    SIGN _ NUM$SIGNF
    NUM_NUM$REAL\VALUE
    WHILE EXP < 0 AND NUM$L31 = 0 DO
      NUM _ NUM SHIFT 1
      EXP _ EXP + 1
    ENDWHILE
    IF EXP # 0 THEN FRETURN
    NUM _ APL\INTEGER(NUM)
    NUM$SIGNF _ SIGN
  ENDIF
  RETURN NUM
END

```

```

FUNCTION FLOAT(SCALAR NUM) RETURNING SCALAR
*RETURN APL REAL NUMBER FROM APL INTEGER OR REAL
*FRETURN IF NOT APL NUMBER
SCALAR SIGN
SCALAR POW
FIELD E(0:24,31)
  IF INTNUM(NUM:FRET(41)) THEN
    NUM$IN_0
    IF NUM#0 THEN
      SIGN_NUM$SIGNF
      POW_158
      NUM_NUM SHIFT -1
      WHILE NUM>0 DO
        POW_POW-1
        NUM_NUM SHIFT -1
      ENDWHILE
      NUM_NUM SHIFT 1
      NUM$SIGNF_SIGN
      NUM$E_POW
    ELSE
      NUM$E_128
    ENDIF
  ENDIF
  RETURN NUM
END

```

```

FUNCTION FMAT(STRING STR, SCALAR NUM, SCALAR W:15, SCALAR D:7, ..
              SCALAR TRIM : TRUE)
*APL NUM OUTPUT IN F OR T FORMAT
SCALAR SIGN
SCALAR EXP
SCALAR POW
SCALAR N
SCALAR M
SCALAR NONE
SCALAR NUM1
SCALAR CH
STRING STR1[40]
STRING STR2[40]
FIELD E(0:24,31)
FIELD F(0:1,23)
FIELD L31(0:31,31)
FIELD UP(1)
DOUBLE DPROD
  IF D<0 THEN D_0
  IF W<2 OR D>=W THEN FRET(50)
  SETS(STR1,0,0)
  SETS(STR2,0,0)
  NUM_FLOAT(NUM:FRET(51))
  SIGN_NUM$SIGNF
  EXP_NUM$E+EBIAS
  NUM_NUM$F @ F
  IF EXP<-128 OR EXP>128 THEN FRET(54)
  IF EXP<0 THEN EXP_128-EXP
  DPROD_LMUL(NUM ,FRACTWO(EXP))
  NUM_ (DPROD$UP / 10) SHIFT -2
  POW_EXPTWO(EXP) +1
  M_9999999
  N_0
  WHILE NUM>M DO
    N_NUM REM 10
    NUM_NUM/10
    POW_POW+1
  ENDWHILE
  IF N>5 OR (N=5 AND ((NUM$L31)=0)) THEN NUM_NUM+1
  N_FORMSTR(STR1,NUM,POW,D ,TRIM:FRET(54))
  IF INT\WIDTH < N THEN INT\WIDTH _N
  OUTSTR(STR2,STR1,SIGN,W,N+D+1,TRIM:FRET(53))
  CH_GCNI(STR2:FRET(54))
  IF CH='>' OR CH='<' THEN RETURN APPEND(STR,STR2:FRET(54))
  CH_GCI(STR1:CH_'X')
  IF CH>='5' AND CH<='9' THEN
    N_1
    L: WHILE CH>='0' AND CH<='9' DO
      N_N*10
      CH_GCI(STR1:EXIT L)
    ENDWHILE
    NUM_NUM+N
    SETS(STR1,0,0)
    N_FORMSTR(STR1,NUM,POW,D,TRIM:FRET(54))

```

```
        OUTSTR(STR,STR1,SIGN,W,N+D+1,TRIM:FRET(53))
ELSE
        APPEND(STR,STR2:FRET(54))
ENDIF
RETURN
END
```

```

FUNCTION FNAME ( SCALAR FNUM, STRING FSTR )
*
* FNAME CONVERTS A FUNCTION-NUMBER INTO A
* SYMBOLIC NAME VIA THE GLOBAL SYMBOL TABLE
*
  IF INVALID ( GLOBLK ) AND FNUM $ FNRUNT = 0 DO
    CNS ( FSTR, FNUM $ FNBLK )
  ELSE DO
    FUNCNAME ( FNUM, FSTR, DEBUGSEG ( FNUM $ FNRUNT ) )
  ENDIF
  *
  RETURN
END

```

```

FUNCTION FNFind ( STRING FName )
*
* LOOK UP A USER FUNCTION BY NAME OR NUMBER
*
    SCALAR FNUM, FD
    REFERENCE ENTRY
    *
    IF STREQL ( FName, GLOBNAME ) DO
        RETURN GLOBLK
    ELSE DO
        IF NUMERIC ( GCNI ( FName :RETURN -1 ) ) DO
            FNUM _ CSN ( FName )
            IF BADBLKNO ( FNUM ) DO
                RETURN -1
            ELSE DO
                RETURN FNUM
            ENDIF
        ELSE DO
            IF INVALID ( GLOBLK ) DO
                FRETURN
            ENDIF
            ENTRY _ STEFIND ( FName, GLOBLK :RETURN -1 )
            FD _ STEFUNCD ( ENTRY )
            IF ISFUNCTION ( ENTRY ) DO
                RETURN FD
            ELSE DO
                RETURN -1
            ENDIF
        ENDIF
    ENDIF
ENDIF
END

```

```
FUNCTION FNHEADER ( SCALAR FD ) RETURNING DESCRIPTOR
*
* RETURNS TWO-WORD FUNCTION HEADER OF APL FUNCTION
*
  VECTOR SEG
  SCALAR I
  REFERENCE R TO DESCRIPTOR
  *
  SEG _ PROCSEG ( FD $ FNRUNT )
  I _ BLKBASE ( SEG, FD $ FNBLK )
  R _ SETREFSIZE ( @ SEG [ I ], 2 )
  RETURN $R
END
```

```

FUNCTION FORMSTR(STRING STR,SCALAR NUM,SCALAR POW,SCALAR DP, ..
                BOOL TRIM:TRUE ) RETURNING SCALAR
*FORM OUTPUT STRING FOR FWD
SCALAR M,N,NONE,D
  N_0
  IF NOT TRIM THEN
    FOR I_1 TO DP DO
      WCD(' ',STR:FRET(52))
    ENDFOR
  ENDIF
  NONE_TRUE
  FOR I_POW TO -1 DO
    D _ NUM REM 10
    IF (NOT NONE) OR (NOT TRIM) OR (D#0) THEN
      WCD( D+'0',STR:FRETURN)
      NONE _ FALSE
    ENDIF
    NUM _ NUM/10
  ENDFOR
  IF NONE AND NOT TRIM THEN WCD('0', STR: FRETURN)
  IF FRAC\WIDTH< LENGTH(STR) THEN FRAC\WIDTH _ LENGTH(STR)
  IF NOT (NONE AND TRIM ) THEN
    WCD('.', STR:FRETURN)
    POINT _ TRUE
  ENDIF
  NONE _ TRUE
  FOR I_1 TO POW DO
    WCD('0',STR:FRET(52))
    N_N+1
    NONE_FALSE
  ENDFOR
  WHILE NUM>0 DO
    N_N+1
    WCD((NUM REM 10) +'0',STR:FRET(52))
    NUM_NUM/10
    NONE_FALSE
  ENDWHILE
  IF NONE THEN
    WCD('0',STR:FRET(52))
    N_N+1
  ENDIF
  RETURN N
END

```

FUNCTION FRACTEN(INDEX)

*

*RETURN FTEN[INDEX]

*TABLE OF 10^N EXPRESSED IN BINARY. (E.G. $10^1=.120000000000B*2^4$

* $10^{-1}=.14631463146B*2^3$)

*RANGE OF N IS -47 TO 47. (N GOES FROM 0 TO 47; 0 TO -47)

*RANGE OF EXPONENT IS -157 TO 157.

*FRACTEN IS THE FRACTIONAL PART. (OCTAL-POINT AT LEFT)

*EXPTEN IS THE ABSOLUTE VALUE OF THE EXPONENTIAL PART.

*SIGN OF EXPTEN IS SAME AS SIGN OF N.

VECTOR FTEN1[32] _ ..

10000000000B, 12000000000B, 14400000000B, 17500000000B, ..
11610000000B, 14152000000B, 17204400000B, 11422640000B, ..
13727410000B, 16715312000B, 11240276200B, 13510355640B, ..
16432451210B, 11060471625B, 13274610172B, 16153752231B, ..
10703362340B, 13064257027B, 15701332635B, 10530710602B, ..
12657072743B, 15432711534B, 10360636031B, 12455005440B, ..
15170206747B, 10213124261B, 12255751335B, 14731343624B, ..
10047716274B, 12061701754B, 14476262347B, 17615737041B

VECTOR FTEN4[16] _ ..

11670553324B, 14246706211B, 17320467653B, 11502302713B, ..
14022763476B, 17027560415B, 11316646250B, 13602417722B, ..
16543123707B, 11135764334B, 13365361424B, 16262655730B, ..
10757614547B, 13153557701B, 16006513661B, 10604117317B

VECTOR FTEN2[24] _ ..

0B, 14631463146B, 12172702437B, 10142233514B, ..
15066705654B, 12370553044B, 10306757203B, 15327745153B, ..
12571435611B, 10456027641B, 15574677547B, 12775377606B, ..
10627463005B, 16045604641B, 13204467033B, 11003537174B, ..
16322545137B, 13416752431B, 11162273507B, 16603622477B, ..
13634502062B, 11343550050B, 17071100100B, 14055546400B

VECTOR FTEN3[24] _ ..

11527605147B, 17362641727B, 14302201423B, 11716464334B, ..
17660755371B, 14532275773B, 12110227774B, 10071654626B, ..
14766107527B, 12304722737B, 10235417114B, 15225513340B, ..
12504411115B, 10403472412B, 15470767167B, 12707137305B, ..
10554114236B, 15740172060B, 13114773215B, 10727142475B, ..
16213404142B, 13326003265B, 11104634221B, 16472706664B

VECTOR FTEN

CONSTANT VTENSIZE_96

FTEN_FTEN1

FTEN\$VDSIZE_VTENSIZE

RETURN FTEN[INDEX]

END

FUNCTION FRACTWO(INDEX)

*

*RETURN FTWO[INDEX]

*TABLE OF 2^N EXPRESSED IN DECIMAL. (E.G. $2^1=2000000000.*10^{-9}$

*

$2^{-1}=0500000000.*10^{-9}$)

*RANGE OF N IS -128 TO 128. (N GOES FROM 0 TO 128; -1 TO -128)

*RANGE OF EXPONENT IS -47 TO 30.

*FRACTWO IS THE FRACTIONAL PART. (DECIMAL POINT AT RIGHT)

*EXPTWO IS THE EXPONENTIAL PART.

VECTOR FTWO1[44] _ ..

1000000000,	2000000000,	0400000000,	0800000000,	..
1600000000,	0320000000,	0640000000,	1280000000,	..
0256000000,	0512000000,	1024000000,	2048000000,	..
0409600000,	0819200000,	1638400000,	0327680000,	..
0655360000,	1310720000,	0262144000,	0524288000,	..
1048576000,	2097152000,	0419430400,	0838860800,	..
1677721600,	0335544320,	0671088640,	1342177280,	..
0268435456,	0536870912,	1073741824,	0214748365,	..
0429496730,	0858993459,	1717986918,	0343597384,	..
0687194767,	1374389535,	0274877907,	0549755814,	..
1099511628,	0219902326,	0439804651,	0879609302	

VECTOR FTWO2[44] _ ..

1759218604,	0351843721,	0703687442,	1407374884,	..
0281474977,	0562949953,	1125899907,	0225179981,	..
0450359963,	0900719926,	1801439851,	0360287970,	..
0720575940,	1441151881,	0288230376,	0576460752,	..
1152921505,	0230584301,	0461168602,	0922337204,	..
1844674407,	0368934882,	0737869763,	1475739526,	..
0295147905,	0590295810,	1180591621,	0236118324,	..
0472236648,	0944473297,	1888946593,	0377789319,	..
0755578637,	1511157275,	0302231455,	0604462910,	..
1208925820,	0241785164,	0483570328,	0967140656,	..
1934281311,	0386856262,	0773712525,	1547425049	

VECTOR FTWO3[41] _ ..

0309485010,	0618970020,	1237940039,	0247588008,	..
0495176016,	0990352031,	1980704063,	0396140813,	..
0792281625,	1584563250,	0316912650,	0633825300,	..
1267650600,	0253530120,	0507060240,	1014120480,	..
2028240960,	0405648192,	0811296384,	1622592768,	..
0324518554,	0649037107,	1298074215,	0259614843,	..
0519229686,	1038459372,	2076918743,	0415383749,	..
0830767497,	1661534995,	0332306999,	0664613998,	..
1329227996,	0265845599,	0531691198,	1063382397,	..
2126764793,	0425352959,	0850705917,	1701411835,	..
0340282367				

VECTOR FTWO4[43] _ ..

0500000000,	0250000000,	1250000000,	..	
0625000000,	0312500000,	1562500000,	0781250000,	..
0390625000,	1953125000,	0976562500,	0488281250,	..
0244140625,	1220703125,	0610351563,	0305175781,	..
1525878906,	0762939453,	0381469727,	1907348633,	..
0953674316,	0476837158,	0238418579,	1192092896,	..
0596046448,	0298023224,	1490116119,	0745058060,	..
0372529030,	1862645149,	0931322575,	0465661287,	..

```

0232830644, 1164153218, 0582076609, 0291038305, ..
1455191523, 0727595761, 0363797881, 1818989404, ..
0909494702, 0454747351, 0227373675, 1136868377
VECTOR FTWO5[44] _ ..
0568434189, 0284217094, 1421085472, 0710542736, ..
0355271368, 1776356839, 0888178420, 0444089210, ..
0222044605, 1110223025, 0555111512, 0277555756, ..
1387778781, 0693889390, 0346944695, 1734723476, ..
0867361738, 0433680869, 0216840435, 1084202172, ..
0542101086, 0271050543, 1355252716, 0677626358, ..
0338813179, 1694065895, 0847032947, 0423516474, ..
2117582368, 1058791184, 0529395592, 0264697796, ..
1323488980, 0661744490, 0330872245, 1654361225, ..
0827180613, 0413590306, 2067951531, 1033975766, ..
0516987883, 0258493941, 1292469707, 0646234854
VECTOR FTWO6[41] _ ..
0323117427, 1615587134, 0807793567, 0403896784, ..
2019483917, 1009741959, 0504870979, 0252435490, ..
1262177448, 0631088724, 0315544362, 1577721810, ..
0788860905, 0394430453, 1972152263, 0986076132, ..
0493038066, 0246519033, 1232595164, 0616297582, ..
0308148791, 1540743956, 0770371978, 0385185989, ..
1925929944, 0962964972, 0481482486, 0240741243, ..
1203706215, 0601853108, 0300926554, 1504632769, ..
0752316385, 0376158192, 1880790961, 0940395481, ..
0470197740, 0235098870, 1175494351, 0587747175, ..
0293873588
VECTOR FTWO
CONSTANT VTWOSIZE_257
      FTWO_FTWO1
      FTWOSVDSIZE_VTWOSIZE
      RETURN FTWO[INDEX]
END

```

```
FUNCTION FREE\DESCR ( REFERENCE D )  
*  
* RELEASE DESCRIPTOR SLOT  
*  
  D . D\FREE _ TRUE  
  RETURN  
END
```

FUNCTION FUNCNAME (SCALAR FNUM, STRING FSTR, VECTOR SEG)

* CONVERT FUNCTION NUMBER INTO SYMBOLIC NAME

SCALAR BLK, I, J, K, L, C
REFERENCE ENTRY
VECTOR A
STRING ASTR

BLK _ FNUM \$ FNBLK
SELWINDOW (GDBWINDOW, SEG, GLOBLK)

A _ GDBWINDOW . BARRAY
I _ BLKINFO (SEG, BLK)
ENTRY _ @A[I]
L _ ENTRY . SSIZE
J _ I + SDISP
K _ J + L - 1

MAKESTRING (ASTR, A, J, K)
SCOPY (FSTR, ASTR)
WHILE (C _ GCD(FSTR)) = BLANK DO; ENDWHILE
WCI (C, FSTR)

RETURN
END

00000
FUNCTION FWD(String STR, SCALAR NUM, SCALAR W:8, SCALAR D:3)
*APL NUMBER OUTPUT IN F FORMAT
 FMAT(STR,NUM,W,D,TRUE:FRET(71))
 RETURN
END

```

FUNCTION GCI\MSG ( REFERENCE TD )
*
* GET NEXT CHARACTER FROM CURRENT OUTPUT MESSAGE
* OF TERMINAL 'TD' AND RETURN IT
*
    SCALAR MBOX, C
    REFERENCE MBD, MSTRING
    VECTOR MVEC
    BOOL EMPTY
    *
    MBOX _ TD . TD\OUT\BOX
    MBD _ @ MAILBOX\TABLE [ MBOX ]
    MVEC _ MBD . MBD\MQHD
    MSTRING _ SETREFSIZE ( @ MVEC [ MSG\DISP ], 3 )
    *
    EMPTY _ FALSE
    C _ GCI ( MSTRING : EMPTY _ TRUE )
    *
    IF LENGTH ( MSTRING ) = 0 OR EMPTY DO
        DEQUEUE\MESSAGE ( MBOX )
        DELIVERY\CHECK ( MVEC )
        FREE ( MSG\SPACE, MVEC )
        IF EMPTY THEN FRETURN
    ENDIF
    *
    RETURN C
END

```

```
-----  
FUNCTION GDD(REFERENCE S) RETURNING DOUBLE  
*  
*GET DOUBLE WORD AND DECREMENT  
*  
SCALAR RS  
DOUBLE T  
*  
    RS_S.SDRS  
    T_GETR(S.SDVD,RS:FRETURN)  
    S.SDRS_RS  
    SRETURN T  
END
```

```
FUNCTION GDI(REFERENCE S)RETURNING DOUBLE
*
*GET DOUBLE WORD AND INCREMENT
*
SCALAR RS
DOUBLE T
  RS_S.SDRS
  T_GETF(S.SDVD,RS:FRETURN)
  S.SDRS_RS
  SRETURN T
END
```

10000

```
FUNCTION GDND(REFERENCE S) RETURNING DOUBLE
*
*GET DOUBLE WORD FROM REAR OF STRING AND DON'T DECREMENT
*
    SRETURN REAR(S.SDVD,S.SDRS:FRETURN)
END
```

```

-----
FUNCTION GETBYTE ( SCALAR FN, SCALAR ADDR )
*
* GET CODE BYTE FROM USER CODE SEGMENT
* F-BASE REL ADDRESS 'ADDR' IN FUNCTION 'FN'.
*
  VECTOR A
  SCALAR VAL
  *
  SELWINDOW ( CCDWINDOW, CDSEG, FN $ FNBLK )
  A _ BYTEVEC ( CCDWINDOW.BARRAY )
  VAL _ A [ ADDR ]
  RETURN VAL
END

```

```
FUNCTION GETM\CONV ( P, OBJ, MBOX ) RETURNING VECTOR
*
    RETURN GETM\QUAD( P, OBJ, MBOX)
END
```

```

FUNCTION GETM\FILE(P,OBJ,BOX) RETURNING VECTOR
*
*GET APL-OBJECT 'OBJ' FROM ADDRESS SPACE OF SENDER (PROCESS P)
* AND COPY IT INTO THE FILE GIVEN BY 'BOX'.
*THE NULL MESSAGE 'NULL\MSG' IS RETURNED AS VALUE.
*
VECTOR ABODY
FIELD SIZE(0:16,31)
REFERENCE FD,PD,MBD,FILE
STRING MSTR
*
    PD_@PROC\TABLE[P]
    MBD_@MAILBOX\TABLE[BOX]
    FD_@FILE\TABLE[MBD.MBD\AOBJ]
    FILE _ @ ( FD . FD\RING )
    *
    IF OBJ$APL\TYPE=TYPE\ARRAY THEN
        ABODY_ARRAY\BLOCK(OBJ,PD)
        ABODY$VDSIZE_ABODY[0]$SIZE + 1
        SETSTRING(MSTR,ABODY)
        EXPAND\FILE(FD, ABODY$VDSIZE+ 1 )
        *
        WCI(OBJ,FILE)
        APPEND(FILE,MSTR)
    ELSE
        EXPAND\FILE(FD, 1)
        WCI(OBJ,FILE)
    ENDIF
    (FD . FD\VEC) [ 0 ] _ FD . FD\RING\OUT
    RETURN NULL\MSG
END

```

```

FUNCTION GETM\NORM ( P, OBJ, MBOX ) RETURNING VECTOR
*
* GET APL-OBJECT 'OBJ' FROM ADDRESS SPACE OF
* SENDER ( PROCESS NUMBER 'P' ) AND COPY IT INTO A
* NEW MESSAGE VECTOR, WHICH IS RETURNED AS VALUE
*
  SCALAR ASIZE
  VECTOR MVEC, MBODY, ABODY
  REFERENCE PD
  *
  PD _ @ PROC\TABLE [ P ]
  IF OBJ $ APL\TYPE = TYPE\ARRAY DO
    ABODY _ ARRAY\BLOCK ( OBJ, PD )
    ASIZE _ ABODY $ VDSIZE
    MVEC _ MAKE ( MSG\SPACE, MSG\SIZE ( 1 + ASIZE ) )
    MBODY _ SUBVEC ( MVEC, MSG\DISP + 1, ASIZE )
    VCOPY ( MBODY, ABODY )
  ELSE DO
    MVEC _ MAKE ( MSG\SPACE, MSG\SIZE ( 1 ) )
  ENDIF
  MVEC [ MSG\DISP ] _ OBJ
  RETURN MVEC
END

```

```

FUNCTION GETM\QUAD ( P, OBJ, MBOX ) RETURNING VECTOR
*
* GET APL OBJECT 'OBJ' FROM ADDRESS SPACE OF SENDER
* ( PROCESS NUMBER 'P' ), CALLING THE APPROPRIATE
* FUNCTION TO CONVERT IT INTO STRING FORMAT AND
* PACK IT INTO A NEW MESSAGE VECTOR
*
REFERENCE MSTR
FIELD NELTS(0:16,31), RANK(0:8,15)
VECTOR MVEC,ABODY
STRING CRLF _ "&M&J"
*
  IF OBJ$APL\TYPE = TYPE\ARRAY THEN
    ABODY _ ARRAY\BLOCK( OBJ, @PROC\TABLE[P])
    IF ABODY[1]$NELTS = 0 THEN
      MVEC _ MAKE(MSG\SPACE, MSG\SIZE (5) )
      MSTR _ MSG\SETSTRING(MVEC)
      IF MAILBOX\TABLE[MBOX]$MBD\GETM = GETM\QUAD THEN ..
        APPEND(MSTR,CRLF)
      ELSEIF ABODY[0]$RANK = 1 THEN
        MVEC _ VECTOR\CONV(ABODY, MBOX)
      ELSE
        MVEC _ MULTI\DIH\CONV(ABODY, MBOX)
      ENDIF
    ELSE
      MVEC _ SCALAR\CONV( OBJ, MBOX)
    ENDIF
  RETURN MVEC
END

```

```
FUNCTION GETM\QUOTEQUAD ( P, OBJ, MBOX ) RETURNING VECTOR
*
* QUOTEQUAD TERMINATE OUTPUT WITH CR LF
*
  RETURN GETM\QUAD(P, OBJ, MBOX)
END
```

```

FUNCTION GET\DESCR ( VECTOR TABLE )
*
* GET FREE DESCRIPTOR IN SPECIFIED TABLE AND
* RETURN ITS INDEX. FAILS IF TABLE FULL
*
  SCALAR I, N
  *
  N _ TABLE $ VDSIZE
  FOR I _ 0 TO N-1 DO
    IF TABLE [ I ] $ D\FREE DO
      TABLE [ I ] $ D\FREE _ FALSE
      RETURN I
    ENDIF
  ENDFOR
  FRETURN
END

```

```

FUNCTION GET\FILENAME(String APL\FILE, OBJ)
*
*COLLECT CHARS FROM 'OBJ'
*  FAILIF: 'OBJ' NOT ALL CHARS; LENGTH OF NAME > SEGNAME\LEN
*
CONSTANT SEGNAME\LEN _ 8
VECTOR AVEC
SCALAR L
FIELD NELTS(0:16,31)
*
  IF LEGAL(AVEC_ALL\CHAR(OBJ)) THEN
    L_AVEC[1]$NELTS
    IF (L>0) AND (L<=SEGNAME\LEN) THEN
      CLEAR(APL\FILE)
      L_L+2
      FOR I_3 TO L DO
        WCI(AVEC[I]$APL\VALUE, APL\FILE)
      ENDFOR
    ELSE
      BADNEWS(NAMELEN\ERR)
      FRETURN
    ENDIF
  ELSE
    BADNEWS(PARAM\ERR)
    FRETURN
  ENDIF
  RETURN
END

```

```
FUNCTION GNBCI ( STRING S )
*
* GET NON-BLANK CHARACTER AND INCREMENT
*
  SCALAR CHAR
  *
  WHILE ( CHAR _ GCI ( S :FRETURN )) = BLANK DO; ENDWHILE
  RETURN CHAR
END
```

```

FUNCTION GOPROC()
*
* GO-COMMAND:
* RESUME EXECUTION OF DEBUGGEE PROCESS
*
VECTOR DTSEG
REFERENCE STATE\AREA, TOP
SCALAR FLAGS, PRCAP, TRAP, FD, FADDR
CONSTANT FLSTEP _ 000002B, FLOTHER _ 177775B
MACRO STEPTRAP _ ( 2 @ SA\TRAPCLASS + 1 @ SA\TRAPNUMBER )
*
DB\CHECK
DTSEG _ DB\PD . PD\DTSEG
STATE\AREA _ FIND\STATE ( DTSEG )
PRCAP _ DB\PD . PD\PRCAP
*
IF DB\PD . PD\BREAK DO
    FADDR _ STATE\AREA . SA\SBASE + STATE\AREA . SA\LBASE
    TOP _ SETREFSIZE ( @ DTSEG [ FADDR ], SF\DSIZE )
    FD _ TOP . SF\FNDESC
    *
    * RE-FIND LTE CORRESP TO PCTR
    *
    LTEFIND ( FD $ FNRUNT, FD $ FNBLK, TOP . SF\PCTR :CRASH() )
    IF LTE $ LTEFLAGS # 0 DO
        *
        * RESTORE SAVED CODE FROM LINE-TABLE
        *
        PUTBYTE ( FD $ FNBLK, LTE $ LTECODEPTR, LTE $ LTEISAVE )
        *
        * SET SINGLE-INSTRUCTION-STEP MODE
        *
        STATE\AREA . SA\FLAGS _ OR ( STATE\AREA . SA\FLAGS, FLSTEP )
        *
        * EXECUTE SAVED CODE
        *
        APL\ON ( PRCAP )
        TRAP _ TRAP\WAIT ( DB\PD )
        *
        * TURN OFF STEP MODE
        *
        STATE\AREA . SA\FLAGS _ AND ( STATE\AREA . SA\FLAGS, FLOTHER )
        *
        * RESTORE BKPT-TRAP INSTRUCTION
        *
        PUTBYTE ( FD $ FNBLK, LTE $ LTECODEPTR, BRKOP )
        *
        * IF SAVED CODE RAN OK, KEEP GOING UNTIL NEXT TRAP
        *
        IF TRAP = STEPTRAP DO
            CLEAR\TRAP ( DB\PD )
        ELSE DO
            NO\DB
            RETURN
    
```

```
        ENDIF
    ENDIF
ENDIF
APL\ON ( PRCAP )
DB\PD . PD\STOPPED _ FALSE
NO\DB
RETURN
END
```

```

FUNCTION GWD(STRING STR,SCALAR NUM)
*
FIELD FRAC\PART(0:1,23), EXP\PART(0:24,31)
SCALAR CH,N
BOOL OK
*
  IF MAX\CONV = EFORMAT THEN
    EMAT(STR,NUM)
  ELSE
    IF NUM$APL\TYPE = TYPE\INTEGER THEN
      N _ NUM$APL\VALUE
      IF N # 0 AND N >= MAX\INOT THEN
        EMAT(STR,NUM)
        MAX\CONV _ EFORMAT
      ELSE
        GWD2(STR,NUM)
      ENDIF
    ELSE
      N _ OR(NUM$FRAC\PART SHIFT 8, NUM$EXP\PART SHIFT -23)
      IF N >= MAX\FNOT OR N <= MIN\FNOT THEN
        EMAT(STR,NUM)
        MAX\CONV _ EFORMAT
      ELSE
        GWD2(STR,NUM)
      ENDIF
    ENDIF
  ENDIF
  RETURN
END

```

```

FUNCTION GWD2(STRING STR, SCALAR NUM)
*APL NUM OUTPUT IN MINIMAL I FORMAT(MAX:W) IF POSSIBLE
*      ELSE IN MINIMAL F FORMAT(MAX:W,D) IF POSSIBLE ELSE
*      IN MINIMAL E FORMAT
STRING STR1[20]
SCALAR CH
BOOL OK
  CLEAR(STR1)
  OK _ TRUE
  IMAT(STR,NUM : OK _ FALSE)
  IF NOT OK THEN
    FMAT( STR1, NUM : WCI( '>', STR1:FRETURN))
    CH _ GCNI(STR1)
    IF (CH='<') OR (CH='>') THEN
      EMAT( STR,NUM :FRETURN)
      MAX\CONV _ EFORMAT
    ELSE
      APPEND(STR,STR1:FRETURN)
      IF MAX\CONV < FFORMAT THEN MAX\CONV _ FFORMAT
    ENDIF
  ELSE
    IF MAX\CONV < IFORMAT THEN MAX\CONV _ IFORMAT
  ENDIF
  RETURN
END

```

```
FUNCTION IMAT(STRING STR, SCALAR NUM, SCALAR W:8, SCALAR TRIM:TRUE)
*23 BIT SIGNED APL INTEGER OUTPUT AS DECIMAL
FIELD VAL(0:8,31)
  NUM_FIX( NUM:FRETURN)
    DECIM(STR,NUM$VAL,NUM$SIGNF,W,TRIM :FRET(4))
  RETURN
END
```

```
FUNCTION INBREAK ( SCALAR BLK, SCALAR LTE )
*
* INSERT BREAKPOINT-TRAP PATCH AND RETURN UPDATED LTE AS VALUE
*
  IF LTE $ LTEFLAGS = 0 DO
    LTE $ LTEISAVE _ GETBYTE ( BLK, LTE $ LTECODEPTR )
    PUTBYTE ( BLK, LTE $ LTECODEPTR, BRKOP )
  ENDIF
  RETURN LTE
END
```

```
FUNCTION INDEX\ERR ( N )
```

```
* INDEX ERROR
```

```
  IF N = 1 DO
```

```
    MESSAGE ( "WRONG NUMBER OF SUBSCRIPTS&M" )
```

```
  ELSE DO
```

```
    MESSAGE ( "INDEX ERROR&M" )
```

```
  ENDIF
```

```
  RETURN
```

```
END
```

```

FUNCTION INIT\BOX ( MBOX )
*
* INITIALIZE MAILBOX NUMBER 'MBOX'
*
  REFERENCE MBD, MBRQ
  *
  MBD _ @ MAILBOX\TABLE [ MBOX ]
  *
  * SET QUEUES EMPTY
  *
  MBD . MBD\MQHD _ NULL\MSG
  MBD . MBD\MQTL _ NULL\MSG
  *
  MBRQ _ @ ( MBD . MBD\RCVQ )
  CLEAR ( MBRQ )
  *
  * SET STATUS TO UNATTACHED
  *
  MBD . MBD\ASTAT _ MBD\UNATTACHED
  MBD . MBD\AOBJ _ 0
  MBD . MBD\PUTM _ PUTM\NORM
  MBD . MBD\GETM _ GETM\NORM
  *
  RETURN
END

```

```

FUNCTION INSERT\TIME(DOUBLE TIME)
*
*INSERT 'TIME' ON 'TIMER\LIST'
*
STRING T\LIST[MAXBOXES 2 WORD]
BOOL AGAIN
DOUBLE T, T1
FIELD WD1(1)
    CLEAR(T\LIST)
L: WHILE (GDND(TIMER\LIST:EXIT L))$TL\LTIME<TIME$TL\LTIME DO
    WDD(GDD(TIMER\LIST),T\LIST)
    ENDWHILE
    AGAIN_TRUE
    T1$WD1_EOR(TIME$TL\LTIME,-1)
    T1$TL\RTIME_EOR(TIME$TL\RTIME,-1)
L1:WHILE AGAIN DO
    T_GDND(TIMER\LIST:EXIT L1)
    T_LADD(T1,T)
    IF T$WD1>=0 THEN
        AGAIN_FALSE
    ELSE
        WDD(GDD(TIMER\LIST), T\LIST)
    LNDIF
    ENDWHILE
    WDI(TIME,TIMER\LIST)
    APPENDDD(TIMER\LIST,T\LIST)
    RETURN
END

```

```

FUNCTION INTEGER\VECTOR ( OBJ ) RETURNING VECTOR
*
* CHECK PARAMETER: IS IT AN INTEGER VECTOR ?
*
    BOOL OK
    SCALAR R, L, I, N
    VECTOR AVEC
    FIELD RANK ( 0: 8,15 ), NELTS ( 0: 16,31 )
    *
    IF OBJ $ APL\TYPE = TYPE\ARRAY DO
        AVEC _ ARRAY\BLOCK ( OBJ, TRAP\PD )
        R _ AVEC [ 0 ] $ RANK
        IF R = 1 DO
            L _ AVEC [ 1 ] $ NELTS
            OK _ TRUE
            FOR I _ 0 TO L-1 DO
                N _ AVEC [ I+3 ]
                IF N $ APL\TYPE # TYPE\INTEGER DO
                    OK _ FALSE
                ENDIF
            ENDFOR
            IF OK DO RETURN AVEC
        ENDIF
    ENDIF
    RETURN ILLEGAL\VEC
END

```

```
FUNCTION INTNUM(SCALAR NUM) RETURNING SCALAR
*IF NUM IS APL INTEGER THEN RETURN TRUE
*IF NUM IS APL REAL THEN RETURN FALSE
*OTHERWISE FRETURN
FIELD INT(0:1,8)
FIELD REALNUM(0:1,1)
  IF NUM$INT=100B THEN
    RETURN TRUE
  ELSEIF NUM$REALNUM=1 THEN
    RETURN FALSE
  ELSE
    FRET(11)
  ENDIF
END
```

```
FUNCTION IW(String STR, NUM, W:8)
*23 BIT SIGNED APL INTEGER OUTPUT AS DECIMAL
  IMAT(STR,NUM,W,TRUE:FRET(4))
  RETURN
END
```

```

-----
FUNCTION LADD(DOUBLE A , DOUBLE B) RETURNING DOUBLE
*
*DO DOUBLE PRECISION ADD, FRETURN IF OVERFLOW
*
FIELD LL3(0:0,2), LR29(0:3,31)
FIELD LBIT28(0:28,28),WD(0),WD1(1)
FIELD UL3(1:0,2), UR29(1:3,31)
FIELD L1BIT(1:0,0)
FIELD L3BITS(0:0,2), R29BITS(0:3,31), L29BITS(0:0,28)
SCALAR T1,T2
DOUBLE R
*
  T1_A$LR29+B$LR29
  T2_T1$LL3+A$LL3+B$LL3
  R$WD_(T2 SHIFT -29) + T1$LR29
  T1_A$UR29+B$UR29+T2$LBIT28
  T2_T1$L3BITS+A$UL3+B$UL3
  R$WD1_(T2 SHIFT -29) + T1$R29BITS
  IF A$L1BIT = B$L1BIT AND A$L1BIT # R$L1BIT THEN FRETURN
  RETURN R
END

```

FUNCTION LENGTH\ERR (N)

* APL LENGTH ERROR

MESSAGE ("LENGTH ERROR&M")

RETURN

END

```

FUNCTION LOOKUP ( STRING PNAME )
*
* LOOKUP REFORMATS NAME AND CALLS "STLK" TO LOOK IN
* SYMBOL TABLE IN BUFFER.
*
  SCALAR WLENGTH
  VECTOR WVEC
  STRING NAME [ IDLENGTH + 3 8 BIT ]
  *
  SCOPY ( NAME , PNAME )
  *
  UNTIL ( LENGTH ( NAME ) MOD CPW = 0 ) DO
    WCI(BLANK,NAME)
  ENDUNTIL
  WVEC _ FULLVEC ( NAME.SDVD )
  WLENGTH _ LENGTH ( NAME ) / CPW
  *
  RETURN STLK ( GDBWINDOW.BUFFER, WVEC, WLENGTH :FRETURN)
  *
END

```

FUNCTION LSUB (DOUBLE A , DOUBLE B) RETURNING DOUBLE

*

* DOUBLE PRECISION SUBTRACT, FRETURN IS OVERFLOW... A - B

*

FIELD LOB (0), HOB (1)

DOUBLE D1

B\$LOB _ EOR(B\$LOB ,-1); * COMPLIMENT BITS OF LOW ORDER.

B\$HOB _ EOR(B\$HOB,- 1); * COMPLIMENT BITS OFHIGH ORDER.

D1\$LOB _ 1; * SET A DOUBLE PREC. CONSTANT O F 1.

D1\$HOB _ 0; * SET A DOUBLE PREC. CONSTANT OS 1.

B _ LADD (B,D1); * 2'S COMPLIMENT NOW IN B.

RETURN LADD(A,B); * ADD 2'S COMPLIMENT.

END

```

FUNCTION LTEFIND ( SCALAR R, SCALAR F, SCALAR P )
*
* FIND LINE-TABLE-ENTRY CORRESPONDING TO FUNCTION-RELATIVE
* P-COUNTER P IN FUNCTION F. R IS RUNTIME FLAG. LOADS THE
* GLOBAL LTE BUFFER AND RETURNS THE LINE NUMBER OF THE
* STATEMENT-LINE. RETURNS -1 IF P IS OUTSIDE OF F.
* FAILS IF DEBUGGER BLOCK OF F IS INVALID.
*
    SCALAR START, FINISH, FIRST, NLines, I, CP
    REFERENCE HDR
    VECTOR A
    *
    *CHECK FOR INVALID DEBUGGER BLOCK
    *
    IF BADINFO ( DEBUGSEG(R), F ) DO
        FRETURN
    ENDIF
    *
    * CHECK FOR BAD P-COUNTER
    *
    START _ INST1
    FINISH _ BLKSIZE(PROCSEG(R),F)*BPW - 1
    IF P<START OR FINISH<P DO
        RETURN -1
    ENDIF
    *
    * GET LINE TABLE
    *
    SELWINDOW ( CDBWINDOW, DEBUGSEG(R), F )
    A _ CDBWINDOW.BARRAY
    HDR _ SETREFSIZE ( @A[DBHDR], DBHDRSZ )
    FIRST _ HDR.LTBASE
    NLines _ HDR.LTSIZE
    *
    * SEARCH FOR LTE CONTAINING P-COUNTER VALUE
    *
    FOR I _ NLines-1 BY -1 TO 0 DO
        LTE _ A[FIRST + I*WPW]
        CP _ LTE $ LTECODEPTR
        IF CP # LTENULL AND P >= CP DO
            RETURN I+1
        ENDIF
    ENDFOR
    *
    * FALL THRU IF BAD LINE TABLE
    *
    CRASH()
END

```

```

FUNCTION LTEGET ( VECTOR S, SCALAR F, SCALAR L )
*
* GET LINE TABLE ENTRY FOR LINE L OF FUNCTION F OF
* DEBUG SEGMENT S. LOADS GLOBAL LTE BUFFER, AND
* RETURNS TRUE. RETURNS FALSE IF NO SUCH LINE IN F.
* FAILS IF DEBUGGER BLOCK OF F IS INVALID.
*
    SCALAR FIRST, NLINES
    REFERENCE HDR
    VECTOR A
    *
    * CHECK FOR INVALID DEBUGGER BLOCK
    *
    IF BADINFO ( S, F ) DO
        FRETURN
    ENDIF
    *
    * GET LINE TABLE
    *
    SELWINDOW ( CDBWINDOW, S, F )
    A _ CDBWINDOW.BARRAY
    HDR _ SETREFSIZE ( @A[DBHDR], DBHDRSZ )
    FIRST _ HDR.LTBASE
    NLINES _ HDR.LTSIZE
    *
    * CHECK FOR BAD LINE NUMBER
    *
    IF L > NLINES DO
        RETURN FALSE
    ENDIF
    *
    * FETCH LTE
    *
    LTE _ A [ FIRST + L - 1 ]
    RETURN TRUE
    *
END

```

```

FUNCTION LTEPUT ( VECTOR S, SCALAR F, SCALAR L )
*
* PUT MODIFIED LTE FOR LINE L OF FUNCTION F OF DEBUG SEGMENT
* S BACK INTO LINE-TABLE.
*
    SCALAR FIRST, NLINES
    REFERENCE HDR
    VECTOR A
    *
    IF BADINFO ( S, F ) DO
        FRETURN
    ENDIF
    *
    SELWINDOW ( CDBWINDOW, S, F )
    A _ CDBWINDOW.BARRAY
    HDR _ SETREFSIZE ( @ A [ DBHDR ], DBHDRSZ )
    FIRST _ HDR.LTBASE
    NLINES _ HDR.LTSIZE
    *
    IF L > NLINES DO
        RETURN FALSE
    ENDIF
    *
    A [ FIRST + L - 1 ] _ LTE
    RETURN TRUE
END

```

```
FUNCTION LVOUT ( STRING L, SCALAR V )  
*  
* PRINT LABELED OCTAL VALUE  
*  
    SOUT ( L )  
    IOUT ( V,, 8 )  
    RETURN  
END
```

```

FUNCTION MAKESTRING ( STRING S, VECTOR V, I, J )
*
* MAKES STRING 'S' HOLD CHARACTERS PACKED IN
* WORD-VECTOR 'V', ADDING THE "EXTRA" CHAR AT
* THE END TO MAKE A KOSHER STRING.
*
  V _ SUBVEC ( V, I, J-I+1 )
  V _ BYTEVEC ( V )
  V $ VDSIZE _ V $ VDSIZE + 1
  SETSTRING ( S, V )
  RETURN
END

```

```

FUNCTION MD\CONV ( VECTOR ABODY, MBOX ) RETURNING VECTOR
*
* CONVERT APL ARRAY 'ABODY' INTO A CHARACTER STRING
* USING 'CONV'. THE STRING ( PREFIXED BY ITS DESCRIPTOR ) IS
* STORED INTO A NEW MESSAGE VECTOR, WHICH IS RETURNED AS VALUE
*

SCALAR ASIZE, WSIZE, R, S, P, WIDTH
VECTOR MVEC
REFERENCE MSTR
STRING CRLF _ "&M&J"
FIELD NELTS ( 0: 16,31 ), RANK ( 0: 8,15 )
*
S _ 1
P _ 1
R _ ABODY [ 0 ] $ RANK
FOR I _ 2 TO R DO
    P _ ABODY [ I ] $ APL\VALUE * P
    S _ P + S
ENDFOR
ASIZE _ ABODY [ 1 ] $ NELTS + S * 2
IF MAX\CONV = CFORMAT THEN
    WIDTH _ 1
ELSEIF MAX\CONV = IFORMAT THEN
    WIDTH _ INT\WIDTH + 1 + NUM\SIGN
ELSEIF MAX\CONV = FFORMAT THEN
    WIDTH _ INT\WIDTH + POINT + FRAC\WIDTH + 1 + NUM\SIGN
ELSE
    WIDTH _ SIGDIG + 2 + EXP\WIDTH + NUM\SIGN + 1
ENDIF
WSIZE _ ( ASIZE * WIDTH + 5 ) / 4
MVEC _ MAKE ( MSG\SPACE, MSG\SIZE ( WSIZE + 3 ) )
MSTR _ MSG\SETSTRING ( MVEC )
SAVE\FRAC\WIDTH _ FRAC\WIDTH
SAVE\EXP\WIDTH _ EXP\WIDTH
*
    RANKCURSE ( MSTR, ABODY, 1, R+2, MAX\CONV, WIDTH )
    FOR I _ 2 TO R DO
        GCD(MSTR);GCD(MSTR)
    ENDFOR
    IF MAILBOX\TABLE[MBOX]$MBD\GETM = GETM\QUAD THEN ..
        APPEND( MSTR, CRLF)
RETURN CONTRACT ( MSG\SPACE, MVEC, ..
    MSG\SIZE ( ( LENGTH ( MSTR ) + 3 ) / 4 + 3 ) )
END

```

```

FUNCTION MSG\SETSTRING ( VECTOR MVEC ) RETURNING STRING
*
* SETS UP STRING DESCRIPTOR OF "CONVERTED" FORMAT MESSAGE
* VECTOR AND RETURNS STRING POINTER AS VALUE
*
    SCALAR WORDS
    REFERENCE MSTRING
    VECTOR SVEC
    *
    MSTRING _ SETREFSIZE ( @ MVEC [ MSG\DISP ], 3 )
    WORDS _ MVEC $ VDSIZE - ( MSG\DISP + 3 )
    SVEC _ SUBVEC ( MVEC, MSG\DISP + 3, WORDS )
    SVEC _ BYTEVEC ( SVEC )
    SVEC $ VDSIZE _ SVEC $ VDSIZE + 1
    SETSTRING ( MSTRING, SVEC )
    CLEAR ( MSTRING )
    RETURN MSTRING
END

```

```

FUNCTION MULTI\DIM\CONV(VECTOR ABODY, MBOX ) RETURNING VECTOR
*
* CONVERT ARRAY WITH RANK > 1
*
STRING STR[20]
SCALAR INDEX, ARANK, CH
FIELD NELTS(0:16,31), RANK(0:8,15)
*
    ARANK _ ABODY[0]$RANK
    INDEX _ ABODY[1]$NELTS + ARANK +1
*
    NUM\SIGN _ FALSE
    POINT _ FALSE
    INT\WIDTH _ 0
    FRAC\WIDTH _ 0
    EXP\WIDTH _ 0
    MAX\CONV _ CFORMAT
*
    FOR I _ ARANK + 2 TO INDEX DO
        CLEAR( STR)
        IF ABODY[I]$APL\TYPE # TYPE\CHAR THEN
            GWD(STR, ABODY[I] )
            NUM\SIGN _ NUM\SIGN OR ABODY[I]$SIGNF
        ENDIF
    ENDFOR
    IF SIGDIG < INT\WIDTH + FRAC\WIDTH THEN MAX\CONV _ EFORMAT
    RETURN MD\CONV(ABODY, MBOX)
END

```

```

FUNCTION NEW\SEGMENT ( STRING SEGNAME, LENGTH )
*
* CREATES A NEW SEGMENT, WHETHER ONE EXISTED OR NOT
*
    SCALAR L
    BOOL EXISTS

    EXISTS _ TRUE
    L _ READ\LENGTH ( SEGNAME :EXISTS _ FALSE )
    IF EXISTS DO
        OPEN\WINDOWED ( TCAP, SEGNAME :OPEN\LOCKED(TCAP,SEGNAME:CRASH()))
        REPEAT
            CLOSE ( TCAP :EXIT )
        ENDREPEAT
        IF L # LENGTH DO
            OPEN\WINDOWED ( TCAP, SEGNAME )
            SET\LENGTH ( TCAP, LENGTH )
            CLOSE ( TCAP )
        ENDIF
    ELSE DO
        CREATE\SEGMENT ( SEGNAME, LENGTH :CRASH() )
    ENDIF
    RETURN
END

```

```
FUNCTION NOGOOD ( VECTOR SEG, SCALAR BLK )
*
* TESTS INVALID BIT OF BLOCK BLK IN SEGMENT SEG
*
  SCALAR INFO
  *
  INFO _ BLKINFO ( SEG, BLK )
  RETURN ( INFO $ BLKINVALID = 1 )
END
```

```

FUNCTION NORM(SCALAR NUM, SCALAR POW, SCALAR NSIGN) RETURNING SCALAR
*CONVERT POSITIVE SIMPLE NUMBER INTO NORMALIZED APL NUMBER
FIELD EF(0:24,31)
FIELD RF(0:24,24)
FIELD FF(0:0,23)
  IF NUM=0 THEN RETURN 128
  WHILE NUM>0 DO
    NUM_NUM SHIFT -1
    POW_POW-1
  ENDWHILE
  NUM_NUM SHIFT 1
  POW_POW+159
  IF NUM$RF=1 THEN
    NUM$FF_NUM$FF+1
    IF NUM<0 THEN
      NUM_NUM SHIFT 1
      POW_POW+1
    ENDIF
  ENDIF
  NUM$SIGNF_NSIGN
  IF POW>255 THEN FRET(21)
  NUM$EF_POW
  RETURN NUM
END

```

```
FUNCTION NUMERIC ( SCALAR C )  
  IF '0' <= C AND C <= '9' DO  
    RETURN TRUE  
  ELSE DO  
    RETURN FALSE  
  ENDIF  
END
```

FUNCTION NUMFORM\INIT (SIGDIG)

*

*

MAX\INOT _ 1

FOR I _ 1 TO SIGDIG DO

MAX\INOT _ MAX\INOT * 10

ENDFOR

MAX\FNOT _ OR ((EXPTEN(SIGDIG) - EBIAS) SHIFT -23, ..
FRACTEN(SIGDIG) SHIFT 8)

*

MIN\FNOT _ OR ((-EXPTEN(48+4) - EBIAS) SHIFT -23, ..
FRACTEN(SIGDIG) SHIFT 8)

RETURN

END

```

FUNCTION OBJINIT()
*
* INITIALIZE UNIVERSE OF APL INTERPROCESS OBJECTS
*
  VECTOR SEGVEC [ BIG EXTERNAL 0 ]; * USED IN PD INITIALIZATION
  REFERENCE PD, FD, MHDR
  SCALAR I
  *
  INIT ( FREEPOOL )
  INIT ( MSG\SPACE )
  APL\INIT()
  *
  * INITIALIZE TERMINAL TABLE
  *
  FOR I _ 0 TO MAXTERMS-1 DO
    TERM\TABLE [ I ] $ D\FREE _ TRUE
  ENDFOR
  *
  * INITIALIZE PROCESS TABLE
  *
  FOR I _ 0 TO MAXPROCS-1 DO
    PD _ @ PROC\TABLE [ I ]
    PD.D\FREE _ TRUE
    PD.PD\PRCAP _ PRCAPS + I
    PD . PD\DTSEG _ SEGVEC
    PD . PD\DTCAP _ DTCAPS + I
    CLEAR\TRAP ( PD )
  ENDFOR
  NO\DB
  *
  * INITIALIZE MAILBOX TABLE
  *
  FOR I _ 0 TO MAXBOXES-1 DO
    MAILBOX\TABLE [ I ] $ D\FREE _ TRUE
  ENDFOR
  *
  * INITIALIZE FILE TABLE
  *
  FOR I _ 0 TO MAXFILES-1 DO
    FD _ @FILE\TABLE [ I ]
    FD . D\FREE _ TRUE
    FD.FD\CAPX _ FTCAPS + I
    FD.FD\VEC _ SEGVEC
    FD.FD\VEC\CAPX _ FTCAPS + I
    FD . FD\IN\MSG _ SUBVEC ( FD\MSG\VECS, I*MSG\DISP, MSG\DISP )
    MHDR _ SETREFSIZE ( @(FD.FD\IN\MSG)[0], MSG\DISP )
    MHDR . MSG\DELIVER _ FALSE
  ENDFOR
  *
  * INITIALIZE NUMBER OUTPUT FORMAT
  *
  SIGDIG _ 7
  NUMFORM\INIT(SIGDIG)
  *

```

```
*  
*  INIT TIMER LIST  
*  
CLEAR ( TIMER\LIST )  
*  
RETURN  
END
```

```

FUNCTION OBJPUSH ( OBJ, REFERENCE TO\PD, REFERENCE FROM\PD )
*
* PUSH APL OBJECT 'OBJ' FROM PROCESS 'FROM\PD' ONTO STACK OF
* PROCESS 'TO\PD'
*
    VECTOR TO\VEC, FROM\VEC
    SCALAR TO\HDR, ASIZE
    FIELD RANK ( 0: 8,15 ), REFCNT ( 0: 0,15 ), ADDR ( 0: 16,31 )
    *
    IF OBJ $ APL\TYPE = TYPE\ARRAY DO
        FROM\VEC _ ARRAY\BLOCK ( OBJ, FROM\PD )
        ASIZE _ FROM\VEC $ VDSIZE
        TO\VEC _ APLMAKE ( TO\PD . PD\DTSEG, ASIZE )
        *
        TO\HDR _ TO\VEC [ 0 ]
        VCOPY ( TO\VEC, FROM\VEC )
        TO\VEC [ 0 ] _ TO\HDR
        TO\VEC [ 0 ] $ RANK _ FROM\VEC [ 0 ] $ RANK
        TO\VEC [ 1 ] $ REFCNT _ 1
        OBJ $ ADDR _ TO\VEC $ VDBASE
    ENDIF
    PUSHWORD ( OBJ, TO\PD )
    RETURN
END

```

```

FUNCTION OCT(STRING STR, SCALAR NUM, SCALAR SIGN, SCALAR W,    ..
              SCALAR TRIM)
*32 BIT UNSIGNED SIMPLE NUMBER OUTPUT AS OCTAL
SCALAR N
FIELD F3(0:29,31)
STRING STR1[11]
  N_0
  SETS(STR1,0,0)
  WHILE BTU NUM>0 DO
    N_N+1
    WCD(NUM$F3+'0',STR1 :FRET(5))
    NUM_NUM SHIFT 3
  ENDWHILE
  OUTSTR(STR,STR1,SIGN,W,N,TRIM :FRET(5))
  RETURN
END

```

```
FUNCTION OUTBREAK ( SCALAR BLK, SCALAR LTE )
*
* REMOVE BREAKPOINT-TRAP PATCH
*
  IF LTE $ LTEFLAGS = 0 DO
    PUTBYTE ( BLK, LTE $ LTECODEPTR, LTE $ LTEISAVE )
  ENDIF
  RETURN
END
```

```

FUNCTION OUTSTR(STRING STR, STRING STR1, SCALAR SIGN, SCALAR W, ..
                SCALAR W1, BOOL TRIM : TRUE)
***** THIS THE BEGINNING OF *****
***** CONVERT FROM NUMBER TO STRING *****
*OUTPUT STR1 ONTO STR WITH PROPER SIGN AND SPACING
*
SCALAR BLK
  IF W<=0 THEN FRET(2)
  BLK_W=W1-SIGN
  IF NOT TRIM THEN
    FOR I_1 TO BLK DO
      WCI(' ',STR :FRET(1))
    ENDFOR
  ENDIF
  IF BLK<0 AND NOT TRIM THEN
    IF SIGN THEN
      WCI('<',STR :FRET(1))
    ELSE
      WCI('>',STR :FRET(1))
    ENDIF
    W1_W-1
  ELSE
    IF SIGN THEN WCI('-',STR :FRET(1))
  ENDIF
  IF NOT TRIM THEN
    FOR I_1 TO W1 DO
      WCI(GCI(STR1 :FRET(1)),STR :FRET(1))
    ENDFOR
  ELSE
    REPEAT
      WCI(GCI(STR1:RETURN), STR :FRETURN)
    ENDREPEAT
  ENDIF
  RETURN
END

```

```
FUNCTION OW(STRING STR, SCALAR NUM, SCALAR W:9)
*23 BIT APL INT OUTPUT AS OCTAL
FIELD VAL(0:8,31)
  NUM _FIX (NUM : FRETURN)
    OCT(STR,NUM$VAL,NUM$SIGNF,W,FALSE:FRET(33))
  RETURN
END
```

```

FUNCTION PAIR\OFF ( MBOX )
*
* EXAMINES QUEUES OF MAILBOX NUMBER 'MBOX' FOR A
* MESSAGE-RECEIVER PAIR. IF FOUND, MESSAGE IS
* HANDED TO RECEIVER, WHO IS TURNED ON, AND TRUE IS
* RETURNED. OTHERWISE, NOTHING IS DONE AND
* FALSE IS RETURNED.
*
VECTOR MQHD, HVEC
REFERENCE MBRQ, RPD, MBD, MHDR
SCALAR RPN, OBJ, BOX
PROC PUTMSG
*
MBD _ @ MAILBOX\TABLE [ MBOX ]
MQHD _ MBD . MBD\MQHD
MBRQ _ @ ( MBD . MBD\RCVQ )
*
IF LENGTH ( MBRQ ) = 0 OR MSG\NULL ( MQHD ) DO
RETURN FALSE
ELSE DO
MVEC _ DEQUEUE\MESSAGE ( MBOX )
MHDR _ MSG\HDR ( MVEC )
RPN _ DEQUEUE\RECEIVER ( MBOX )
RPD _ @ PROC\TABLE [ RPN ]
DELIVERY\CHECK ( MVEC )
PUTMSG _ MBD . MBD\PUTM
OBJ _ PUTMSG ( RPN, MVEC, MBOX )
*
BOX _ APL\INTEGER ( MBOX )
PUSHWORD ( BOX, RPD )
PUSHWORD ( OBJ, RPD )
ATTN\RESUME ( RPD )
IF DB\PN = RPN DO NO\DB
RPD . PD\STOPPED _ FALSE
RETURN TRUE
ENDIF
END

```

```

FUNCTION PARAM ( N )
*
* RETURNS NTH ITEM FROM TOP OF STACK OF APL PROCESS
* WITH GIVEN PD . GIVEN THAT P HAS JUST DONE AN
* ATTENTION-TRAP, THIS IS THE NTH PARAMETER OF THE
* SERVICE CALL
*
    SCALAR INDEX
    VECTOR DTSEG
    REFERENCE STATE\AREA
    *
    DTSEG _ TRAP\PD . PD\DTSEG
    STATE\AREA _ FIND\STATE ( DTSEG )
    INDEX _ STATE\AREA.SA\LTOP + STATE\AREA . SA\SBASE - N
    RETURN DTSEG [ INDEX ]
END

```

```

FUNCTION PCPROC()
*
* 'PC' COMMAND: PRINT CODE AND LTE CORRESPONDING TO LINE ADDRESS
*
VECTOR A, S
SCALAR FIRST, LAST, I, B, L
*
IF CMDMOVE ( IPTR1 :FRETURN ) DO
  S _ DBSEG
  B _ IPTR1.BLOCK
  L _ IPTR1.LINE
  IF LTEGET ( S, B, L :FRETURN ) DO
    *
    * PRINT LTE
    *
    SOUT ( "LTE = " )
    IOUT ( LTE $ LTEISAVE,, 8 ); COUT ( ' ' )
    IOUT ( LTE $ LTEFLAGS,, 8 ); COUT ( ' ' )
    IOUT ( LTE $ LTECODEPTR,, 8 ); CRLF ( 1 )
    *
    * LOCATE CODE ( IF ANY )
    *
    FIRST _ LTE $ LTECODEPTR
    IF FIRST # LTENULL DO
      LAST _ LTENULL
      WHILE LAST = LTENULL DO
        L _ L + 1
        IF LTEGET ( S, B, L :FRETURN ) DO
          LAST _ LTE $ LTECODEPTR
        ELSE DO
          LAST _ BLKSIZE ( CDSEG, B ) * BPW
        ENDIF
      ENDWHILE
      *
      * PRINT CODE
      *
      SELWINDOW ( CCDWINDOW, CDSEG, B )
      A _ BYTEVEC ( CCDWINDOW.BARRAY )
      FOR I _ FIRST TO LAST-1 DO
        IOUT ( A [ I ],, 8 )
        IF I REM 4 = 3 DO
          CRLF ( 1 )
        ELSE DO
          COUT ( ' ' )
        ENDIF
      ENDFOR
      CRLF ( 1 )
    ENDIF
    RETURN
  ENDIF
ENDIF
FRETURN
END

```

```

FUNCTION PEEL ( STRING S )
*
* PEEL NEXT COMMAND FROM S, AND PUT IT IN CSTRING
*
  SCALAR CHAR
  *
  CLEAR ( CSTRING )
  UNTIL ( ( CHAR _ GCI(S :RETURN) ) = ';' ) DO
    WCI ( CHAR, CSTRING )
  ENDUNTIL
  RETURN
END

```

```

FUNCTION PFIND ( P )
*
* FIND THE PROCESS DESCRIPTOR FOR APL PROCESS
* WITH UNIQUE-NAME P
*
REFERENCE PD
VECTOR C
*
FOR I _ 0 TO MAXPROCS-1 DO
  PD _ @ PROC\TABLE [ I ]
  C _ DISPLAY\CAP ( PD . PD\PRCAP )
  IF C $ CAP\UNIQUE\NAME = P DO
    RETURN I
  ENDIF
ENDFOR
FRETURN
END

```

```

FUNCTION PLACE\MESSAGE ( OBJ, MBOX, BOOL DELIVER )
*
* PLACE MESSAGE OBJECT 'OBJ' IN MAILBOX 'MBOX' AND
* SET DELIVERY-STATUS INFO AS INDICATED BY 'DELIVER'.
*
REFERENCE MBD, MHDR
VECTOR MVEC
PROC GETMSG
*
MBD _ @ MAILBOX\TABLE [ MBOX ]
GETMSG _ MBD . MBD\GETM
MVEC _ GETMSG ( TRAP\PN, OBJ, MBOX )
UNLESS MSG\NULL ( MVEC ) DO
    MHDR _ MSG\HDR ( MVEC )
    .MHDR . MSG\DELIVER _ DELIVER
    MHDR . MSG\SOURCE _ TRAP\PN
    *
    IF DELIVER DO
        TRAP\PD . PD\DCNT _ TRAP\PD . PD\DCNT + 1
        TRAP\PD . PD\STOPPED _ TRUE
    ENDIF
    *
    ENQUEUE\MESSAGE ( MVEC, MBOX )
ENDUNLESS
RETURN
END

```

```
FUNCTION PLADDR ( SCALAR FN, SCALAR LN )
*
* PRINTS LINE ADDR OF LINE 'LN' IN FUNCTION 'FN'
*
  STRING FUNC [ IDLENGTH ]
  *
  FNAME ( FN, FUNC )
  SOUT ( FUNC )
  COUT ( '[' )
  IOUT ( LN - 1 )
  COUT ( ']' )
  RETURN
END
```

```

FUNCTION PROCSETUP ( SCALAR PN, MAXB )
*
* SETS UP PROCESS NUMBER PN
*
    REFERENCE PD, PMBL
    STRING DATANAME [ FNAMEMAX ]
    STRING DATATAIL [ TAILSZ ]
    REFERENCE HDR
    *
    *
    SELWINDOW ( GDBWINDOW, DBSEG, GLOBLK )
    HDR _ @ ( GDBWINDOW.BARRAY ) [ 0 ]
    GVSIZE _ HDR.VRGAPTR
    *
    DTSIZE _ GVSIZE + ABSIZE + STSIZE
    *
    PD _ @ PROC\TABLE [ PN ]
    PD . PD\STOPPED _ FALSE
    PD . PD\DCNT _ 0
    CLEAR\TRAP ( PD )
    IF MAXB < 0 OR MAXBOXES < MAXB DO
        FRETURN
    ELSE DO
        PMBL _ @ ( PD . PD\MBOXLIST )
        MAKE\STRING ( PMBL, MAXB )
    ENDIF
    *
    CLEAR ( DATATAIL )
    CNS ( DATATAIL, PN )
*
    SEGNAME ( DATANAME, DATATAIL )
    SCOPY ( DATANAME, DATATAIL )
    APPEND ( DATANAME, "ARS" )
    *
    NEW\SEGMENT ( DATANAME, DTSIZE )
    OPEN\LOCKED ( PD . PD\DTCAP, DATANAME )
    *
    APL\CREATE ( PD . PD\PRCAP, DATANAME, CODENAME, RCODENAME )
    *
    RETURN
END

```

```
FUNCTION PROMPT()  
*  
* PROMPTS THE USER AT THE CONTROLLING TERMINAL, IF THIS  
* HAS NOT ALREADY BEEN DONE.  
*  
  STRING P _ "      "  
  *  
  IF NOT PROMPTED DO  
    SOUT ( P )  
    PROMPTED _ TRUE  
  ENDIF  
  RETURN  
END
```

```

FUNCTION PUSHWORD ( SCALAR W, REFERENCE PD )
*
* PUSH WORD W ONTO STACK OF INDICATED PROCESS
*
    SCALAR LTOP, SBASE
    VECTOR DTSEG
    REFERENCE STATE\AREA
    *
    DTSEG _ PD . PD\DTSEG
    STATE\AREA _ FIND\STATE ( DTSEG )
    LTOP _ STATE\AREA . SA\LTOP + 1
    SBASE _ STATE\AREA . SA\SBASE
    DTSEG [ LTOP + SBASE ] _ W
    STATE\AREA . SA\LTOP _ LTOP
    RETURN
END

```

```
-----  
FUNCTION PUTBYTE ( SCALAR FN, SCALAR ADDR, SCALAR BYTE )  
*  
* PUT CODE BYTE INTO USER CODE SEGMENT AT  
* F-BASE REL ADDRESS IN FUNCTION FN  
*  
  VECTOR A  
  *  
  SELWINDOW ( CCDWINDOW, CDSEG, FN $ FNBLK )  
  A _ BYTEVEC ( CCDWINDOW.BARRAY )  
  A [ ADDR ] _ BYTE  
  RETURN  
END
```

```

-----
FUNCTION PUTH\FILE(P, VECTOR MVEC, BOX)
*
*PUT APL-OBJECT FROM FILE GIVEN BY BOX INTO ADDRESS SPACE OF
* RECIEVER (PROCESS P); ENQUEUE DUMMY MESSAGE VECTOR 'MVEC'
* AND RETURN APL DESCRIPTOR WORD FOR OBJECT.
*RETURN PSEUDO APL-CHARACTER '377B' ON EOF.
*
CONSTANT APLEOF_2000000377B
SCALAR OBJ, HD1, ASIZE
VECTOR DTSEG, AVEC
FIELD RANK(0:8,15), SIZE(0:16,31), REFCNT(0:0,15), ..
      ADDRESS(0:16,31), NELTS(0:16,31)
REFERENCE FD, PD, MBD, FILE
STRING ASTR
*
  ENQUEUE\MESSAGE(MVEC, BOX)
  MBD_@MAILBOX\TABLE[BOX]
  FD_@FILE\TABLE[MBD.MBD\AOBJ]
  FILE_@FD.FD\RING
  OBJ_GCI(FILE:APLEOF)
  ASIZE_0
  *
  IF OBJ$APL\TYPE=TYPE\ARRAY THEN
    PD_@PROC\TABLE[P]
    DTSEG_PD.PD\DTSEG
    HD1_GCI(FILE)
    AVEC_APLMAKE(DTSEG,HD1$SIZE)
    *
    ASIZE_HD1$SIZE
    AVEC$VDSIZE_ASIZE+1
    SETSTRING(ASTR,AVEC)
    SETS(ASTR,0,1)
    FOR I_2 TO ASIZE DO
      WCI(GCI(FILE),ASTR)
    ENDFOR
    AVEC[0]$RANK_HD1$RANK
    AVEC[1]$REFCNT_1
    *
    OBJ$ADDRESS_AVEC$VDBASE
  ENDIF
  RETURN OBJ
END

```

```

-----
FUNCTION PUTH\NORM ( P, VECTOR MVEC, MBOX )
*
* PUT APL OBJECT IN 'MVEC' INTO ADDRESS SPACE OF
* RECEIVER ( PROCESS NUMBER 'P' ), RELEASE 'MVEC', AND
* RETURN APL DESCRIPTOR WORD FOR OBJECT.
*
    SCALAR OBJ, ASIZE
    VECTOR DTSEG, AVEC, ABODY, MBODY
    FIELD ADDRESS ( 0: 16,31 ), SIZE ( 0: 16,31 ), ..
        RANK ( 0: 8,15 ), REFCNT ( 0: 0,15 )
    REFERENCE PD
    *
    PD _ @ PROC\TABLE [ P ]
    OBJ _ MVEC [ MSG\DISP ]
    IF OBJ $ APL\TYPE = TYPE\ARRAY DO
        DTSEG _ PD . PD\DTSEG
        ASIZE _ MVEC [ MSG\DISP + 1 ] $ SIZE
        AVEC _ APLMAKE ( DTSEG, ASIZE )
        *
        ABODY _ SUBVEC ( AVEC, 1, ASIZE-1 )
        MBODY _ SUBVEC ( MVEC, MSG\DISP+2, ASIZE-1 )
        *
        VCOPY ( ABODY, MBODY )
        *
        *
        AVEC [ 0 ] $ RANK _ MVEC [ MSG\DISP + 1 ] $ RANK
        AVEC [ 1 ] $ REFCNT _ 1
        *
        OBJ $ ADDRESS _ AVEC $ VDBASE
    ENDIF
    FREE ( MSG\SPACE, MVEC )
    RETURN OBJ
END

```

```

FUNCTION PUTM\QUAD ( P, VECTOR MVEC, MBOX )
*
* CONVERT CHARACTER STRING IN VECTOR 'MVEC' INTO APL FORMAT
* NUMBERS AND CHARACTERS AND PUT IT INTO APL VECTOR IN
* ADDRESS SPACE OF RECEIVER ( PROCESS NUMBER 'P' ). RELEASE
* MVEC AND RETURN APL DESCRIPTOR WORD FOR OBJECT
*

    SCALAR OBJ, ASIZE, MSG\APLSIZE
    VECTOR DTSEG, AVEC, MSG\BUF
    FIELD SLOP ( 0: 2,7 ), SIZE ( 0: 16,31 ), RANK ( 0: 8,15 ), ..
        REFCNT ( 0: 0,15 ), NELTS ( 0: 16,31 )
    REFERENCE PD, MSTR
    *
    PD _ @ PROC\TABLE [ P ]
    MSTR _ SETREFSIZE ( @ MVEC [ MSG\DISP ], 3 )
    ASIZE _ LENGTH ( MSTR )
    MSG\APLSIZE _ 0
    MSG\BUF _ MAKE (MSG\SPACE, ASIZE+3 )
L: FOR I _ 1 TO ASIZE DO
    MSG\BUF[I] _ APLCSN( MSTR :APL\CHAR(GCI (MSTR: EXIT L )))
    MSG\APLSIZE _ I
ENDFOR
IF MSG\APLSIZE = 1 THEN
    OBJ _ MSG\BUF[1]
ELSE
    DTSEG _ PD.PD\DTSEG
    AVEC _ APLMAKE ( DTSEG, MSG\APLSIZE+3 )
    AVEC[0]$RANK _ 1
    AVEC[1]$REFCNT _ 1
    AVEC[1]$NELTS _ MSG\APLSIZE
    AVEC[2] _ APL\INTEGER ( MSG\APLSIZE )
    FOR I _ 1 TO MSG\APLSIZE DO
        AVEC[I+2] _ MSG\BUF[I]
    ENDFOR
    OBJ _ APL\ARRAY ( AVEC )
ENDIF
FREE (MSG\SPACE, MSG\BUF)
FREE (MSG\SPACE, MVEC)
RETURN OBJ
END

```

```

FUNCTION PUTM\QUOTEQUAD ( P, VECTOR MVEC, MBOX )
*
* CONVERT CHARACTER STRING IN VECTOR 'MVEC' INTO AN APL
* CHARACTER ARRAY OR SCALAR AND PUT IT INTO THE
* ADDRESS SPACE OF RECEIVER ( PROCESS NUMBER 'P' ).
* RELEASE 'MVEC' AND RETURN APL DESCRIPTOR WORD FOR OBJECT.
*

SCALAR ASIZE, OBJ
VECTOR DTSEG, AVEC
REFERENCE PD, MSTR
FIELD RANK ( 0: 8,15 ), REFCNT ( 0: 0,15 ), NELTS ( 0: 16,31 )
*
PD _ @ PROC\TABLE [ P ]
MSTR _ SETREFSIZE ( @ MVEC [ MSG\DISP ], 3 )
ASIZE _ LENGTH ( MSTR )
IF ASIZE = 1 THEN
    OBJ _ APL\CHAR ( GCI ( MSTR ) )
ELSE
    DTSEG _ PD . PD\DTSEG
    AVEC _ APLMAKE ( DTSEG, ASIZE + 3 )
    AVEC [ 0 ] $ RANK _ 1
    AVEC [ 1 ] $ REFCNT _ 1
    AVEC [ 1 ] $ NELTS _ ASIZE
    AVEC [ 2 ] _ APL\INTEGER ( ASIZE )
    FOR I _ 1 TO ASIZE DO
        AVEC [ I + 2 ] _ APL\CHAR ( GCI ( MSTR ) )
    ENDFOR
    OBJ _ APL\ARRAY ( AVEC )
ENDIF
FREE ( MSG\SPACE, MVEC )
RETURN OBJ
END

```

FUNCTION PVPROC()

* PRINT VARIABLE COMMAND

```
REFERENCE STE, STATE\AREA, MST
CONSTANT TYPE\IPW _ 4
FIELD IPW\SEG ( 0: 15,15 ), IPW\ADDR ( 0: 16,31 )
SCALAR FD, TYPE, ADDR, VAL
BOOL FOUND
STRING VNAME [ IDLENGTH ], FNAME [ IDLENGTH ]
VECTOR DTSEG, M
```

```
DB\CHECK
CMDARG ( VNAME )
CMDARG ( FNAME )
FOUND _ FALSE
IF LENGTH ( FNAME ) > 0 DO
    STE _ STEFIND ( FNAME, GLOBLK :FRETURN )
    IF STE . STYPE = SFUNC DO
        FD _ STEFUNCD ( STE )
    ELSE DO
        FRETURN
    ENDIF
    FOUND _ TRUE
    STE _ STEFIND ( VNAME, FD :FOUND _ FALSE & LEAVE )
ENDIF
```

```
IF NOT FOUND DO
    FOUND _ TRUE
    STE _ STEFIND ( VNAME, GLOBLK :FOUND _ FALSE & LEAVE )
ENDIF
```

```
DTSEG _ DB\PD . PD\DTSEG
STATE\AREA _ FIND\STATE ( DTSEG )
```

```
IF FOUND DO
    TYPE _ STE . STYPE
    IF TYPE = SVAR DO
        ADDR _ STE . SVAL
        IF STE . SLG DO
            ADDR _ ADDR + STATE\AREA . SA\SBASE + STATE\AREA . SA\LBASE
        ENDIF
        VAL _ DTSEG [ ADDR ]
        IF VAL $ APL\TYPE = TYPE\IPW DO
            ADDR _ VAL $ IPW\ADDR
            IF VAL $ IPW\SEG = 0 DO
                ADDR _ ADDR + STATE\AREA . SA\SBASE
            ENDIF
            VAL _ DTSEG [ ADDR ]
        ENDIF
        IF VAL = APL\UNDEFINED DO
            MESSAGE ( "(UNDEFINED)&M" )
        ELSE DO
            M _ GETM\CONV ( DB\PN, VAL, 0 )
        
```

```
IF MSG\NULL ( M ) THEN
    CRLF ( )
ELSE
    MST _ SETREFSIZE ( @M[MSG\DISP], 3 )
    SOUT ( MST ); CRLF()
    FREE ( MSG\SPACE, M )
ENDIF
ENDIF
ELSE DO
    FOUND _ FALSE
ENDIF
ENDIF
IF NOT FOUND DO
    SOUT ( VNAME ); ERRMSG ( " NOT FOUND&M" )
ENDIF
RETURN
END
```

```
FUNCTION QDELIVER()  
    CRASH()  
END
```

```
FUNCTION QQDELIVER()  
  CRASH()  
END
```

```
FUNCTION QRECEIVE()  
  CRASH()  
END
```

```
FUNCTION QRECEIVE()  
    CRASH()  
END
```

```

FUNCTION RANKCURSE ( STRING MSTR, VECTOR ABODY, ..
                    RLEVEL, AINDEX, PROC CONV, WIDTH )
*
* RECURSIVE FUNCTION TO CALL 'STR\VEC' TO CONVERT RANK-N-ARRAY
* IN 'ABODY' INTO A CHARACTER STRING, WHICH IS PUT IN 'MSTR'.
* 'RLEVEL' IS RANK WE ARE ON AND 'AINDEX' IS NEXT WORD IN 'ABODY'
* TO CONVERT.
*
    SCALAR L
    FIELD RANK ( 0: 8,15 )
    STRING CRLF _ "&M&J"
    *
    IF RLEVEL = ABODY [ 0 ] $ RANK THEN
        AINDEX _ STR\VEC ( MSTR, ABODY, AINDEX, CONV, WIDTH )
        APPEND ( MSTR, CRLF )
    ELSE
        L _ ABODY [ RLEVEL + 1 ] $ APL\VALUE
        FOR I _ 1 TO L DO
            AINDEX _ RANKCURSE ( MSTR, ABODY, RLEVEL+1, AINDEX, CONV, WIDTH )
        ENDFOR
        APPEND ( MSTR, CRLF )
    ENDIF
    RETURN AINDEX
END

```

```
FUNCTION RANK\ERR ( N )  
* APL RANK ERROR  
    MESSAGE ( "RANK ERROR&M" )  
    RETURN  
END
```

```

FUNCTION REAL\TIME()
*
*  SERVICE CALL:
*  PARAM: NONE
*  ACTION: RETURN AN 8 ELEMENT VECTOR OF THE TIME
*          (YEAR, MONTH, DAY, MINUTES, SECONDS, MILLISECONDS,
*          MICROSECONDS )
*
CONSTANT TS\SIZE _ 8
VECTOR DTSEG, AVEC
REFERENCE PD
SCALAR T
DOUBLE TIME0, TIME1
FIELD RANK( 0 : 8,15 ), REFCNT( 0 : 0,15 ), SIZE( 0 : 16,31 )
STRING TSTR [ 30 ]
MACRO LDIV( A, B ) _ ( B & CODE(4114B) & A & CODE(14040B) )
*
  PD _ @ PROC\TABLE [ TRAP\PN ]
  DTSEG _ PD . PD\DTSEG
  AVEC _ APLMAKE ( DTSEG, TS\SIZE + 3 )
  AVEC[ 0 ] $ RANK _ 1
  AVEC [ 1 ] $ REFCNT _ 1
  AVEC [ 1 ] $ SIZE _ TS\SIZE
  AVEC [ 2 ] _ APL\INTEGER ( TS\SIZE )
*
  CLEAR ( TSTR )
  TIME0 _ READ\CLOCK ( )
  DATE\TIME ( TSTR, TIME0 )
  WCI ( 'X', TSTR )
  FOR I _ 3 TO 8 DO
    AVEC [ I ] _ APL\INTEGER ( CSN ( TSTR ) )
    GCI ( TSTR )
  ENDFOR
  * CALCULATE MILLISECOND AND MICROSECOND
  TIME1 _ LMUL ( LDIV ( TIME0, 1000000 ), 1000000 )
  TIME1 _ LSUB ( TIME0, TIME1 )
  T _ TIME1 $ WORD0
  AVEC [ 9 ] _ APL\INTEGER ( T / 1000 )
  AVEC [ 10 ] _ APL\INTEGER ( T REM 1000 )
  EAT ( 1 )
  PUSH\RESUME ( APL\ARRAY (AVEC ), TRAP\PD )
  RETURN
END

```

```

FUNCTION RECEIVE()
*
* SERVICE CALL: RECEIVE MESSAGE FROM SPECIFIED MAILBOX(ES)
*
  SCALAR BOXES, MBOX, LENGTH, I
  BOOL HAVE\MSG
  VECTOR BOXVEC
  BOOL NO\BOX
  *
  BOXES _ PARAM ( 1 )
  *
  IF LEGAL ( MBOX _ SINGLE\INTEGER ( BOXES ) ) DO
    MBOX _ MBOX $ APL\VALUE
    IF BOX\EXISTS ( MBOX ) DO
      EAT ( 2 )
      ENQUEUE\RECEIVER ( TRAP\PN, MBOX )
      PAIR\OFF ( MBOX )
    ELSE DO
      BADNEWS ( NO\OBJ\ERR )
    ENDIF
  ELSEIF LEGAL ( BOXVEC _ INTEGER\VECTOR ( BOXES ) ) DO
    LENGTH _ BOXVEC [ 2 ] $ APL\VALUE
    NO\BOX _ FALSE
    FOR I _ 0 TO LENGTH-1 DO
      MBOX _ BOXVEC [ I+3 ] $ APL\VALUE
      UNLESS BOX\EXISTS ( MBOX ) DO NO\BOX _ TRUE
    ENDFOR
    IF NO\BOX DO
      BADNEWS ( NO\OBJ\ERR )
    ELSE DO
      EAT ( 2 )
      HAVE\MSG _ FALSE
      FOR I _ 0, I+1 UNTIL ( HAVE\MSG OR I = LENGTH ) DO
        MBOX _ BOXVEC [ I + 3 ] $ APL\VALUE
        ENQUEUE\RECEIVER ( TRAP\PN, MBOX )
        HAVE\MSG _ PAIR\OFF ( MBOX )
      ENDFOR
    ENDIF
  ELSE DO
    BADNEWS ( PARAM\ERR )
  ENDIF
  RETURN
END

```

```

FUNCTION REMOVE ( RP, MBOX )
*
* REMOVE PROCESS 'RP' FROM RECEIVER QUEUE
* OF MAILBOX NUMBER 'MBOX'
*
  SCALAR P
  REFERENCE MBRQ
  STRING NEWQ [ MAXPROCS 8 BIT ]
  *
  MBRQ _ @ ( MAILBOX\TABLE [ MBOX ] $ MBD\RCVQ )
  CLEAR ( NEWQ )
  WHILE ( P _ GCI ( MBRQ :-1 ) ) >= 0 DO
    IF P # RP DO WCI ( P, NEWQ )
  ENDWHILE
  SCOPY ( MBRQ, NEWQ )
  RETURN
END

```

```

FUNCTION RESET\TIMER()
*
*SERVICE CALL:
*PARAM: APL INTEGER (BOX NUM)
*ACTION CHECK BOX EXIST AND ATTACHED AS TIMER
*      IF BOX ON LIST , REMOVE IT AND EMPTY BOX OF ALARMS
*FALAIL IF: NO BOX, BOXATTACHED, BAD PARAM
*
SCALAR TBOX, TREMAIN
DOUBLE TIME
FIELD OV( 0: 0,8 ), WD( 0 ), WD1( 1 )
STRING T\LIST[MAXBOXES 2 WORD]
SCALAR OLDLIST
MACRO LDIV(A,B) _ (B&CODE(4114B)&A&CODE(14040B))
CONSTANT CLOCK\FACTOR _ 1000
*
  IF LEGAL(TBOX_SINGLE\INTEGER(PARAM(1))) THEN
    TBOX_TBOX$APL\VALUE
    IF BOX\EXISTS(TBOX) THEN
      IF MAILBOX\TABLE[TBOX]$MBD\ASTAT = MBD\TIMER THEN
        TREMAIN _ 0
        OLDLIST _ TIMER\LIST.SDRS
        IF FIND\TBOX(TBOX,T\LIST) THEN
          TIME _ GDD(TIMER\LIST)
          APPEND(TIMER\LIST,T\LIST)
          TIME$TL\TBOX _ 0
          TIME _ LSUB( TIME, RTCLOCK )
          TREMAIN _ LDIV(TIME, CLOCK\FACTOR)
          IF TREMAIN$OV # 0 THEN TREMAIN _ 37777777B
        ELSE
          TIMER\LIST.SDRS _ OLDLIST
          DEQUEUE\MESSAGE( TBOX : )
        ENDIF
        TIMER()
        EAT(2)
        PUSH\RESUME(APL\INTEGER(TREMAIN), TRAP\PD)
      ELSE
        BADNEWS(BOX\ATTACH\ERR)
      ENDIF
    ELSE
      BADNEWS(NO\OBJ\ERR)
    ENDIF
  ELSE
    BADNEWS(PARAM\ERR)
  ENDIF
  RETURN
END

```

```

FUNCTION REWIND\INFILE()
*
*SERVICE CALL:
*  PARAM: FILE IN-MAILBOX NUMBER
*  ACTION: RESET READ POINTER TO BOF
*  FAILIF: BOX NOT IN-FILE-MAILBOX; PARAM NOT SINGLE INTEGER; NO MAILBOX
*
SCALAR MBOX
REFERENCE MBD,FD
*
  IF LEGAL(MBOX_SINGLE\INTEGER(PARAM(1))) THEN
    MBOX_MBOX$APL\VALUE
    IF BOX\EXISTS(MBOX) DO
      MBD_@MAILBOX\TABLE[MBOX]
      IF MBD.MBD\ASTAT=MBD\FILE THEN
        FD_@FILE\TABLE[MBD.MBD\AOBJ]
        IF FD.FD\IN\BOX=MBOX THEN
          FD.FD\RING\IN_1
          EAT(2)
          ATTN\RESUME(TRAP\PD)
        ELSE
          BADNEWS(NOT\FINBOX\ERR)
        ENDIF
      ELSE
        BADNEWS(NOT\FINBOX\ERR)
      ENDIF
    ELSE
      BADNEWS(NO\OBJ\ERR)
    ENDIF
  ELSE
    BADNEWS(PARAM\ERR)
  ENDIF
  RETURN
END

```

FUNCTION RSCALL\ERR (N)

*ARS_CALL ERROR

```
SOUT ( "BAD SUPERVISOR CALL, " )
IF N = 0 DO
    MESSAGE ( "INVALID PARAMETER&M" )
ELSEIF N = 1 DO
    MESSAGE ( "TOO MANY OBJECTS&M" )
ELSEIF N = 2 DO
    MESSAGE ( "NO SUCH OBJECT&M" )
ELSEIF N = 3 DO
    MESSAGE ( "FILE IS OPEN&M" )
ELSEIF N = 4 DO
    MESSAGE ( "FILE ALREADY EXISTS&M" )
ELSEIF N = 5 DO
    MESSAGE ( "NAME TOO LONG&M" )
ELSEIF N = 6 DO
    MESSAGE ( "NOT FILE-INPUT MAILBOX&M" )
ELSEIF N = 7 DO
    MESSAGE ( "CAN NOT OPEN FILE&M" )
ELSEIF N = 8 DO
    MESSAGE ( "BOX ATTACHMENT ERROR&M" )
ELSEIF N = 9 DO
    MESSAGE ( "IN AND OUT MAILBOXES ARE THE SAME&M" )
ELSEIF N = 10 DO
    MESSAGE ( "ALARM ALREADY SET&M" )
ELSEIF N = 11 DO
    MESSAGE ( "NOT TERMINAL-INPUT MAILBOX&M" )
ELSEIF N = 12 DO
    MESSAGE ( "NO SUCH TERMINAL&M" )
ELSE DO
    SOUT ( "TYPE " ); IOUT ( N )
ENDIF
RETURN
END
```

```
FUNCTION RSEGNAME ( STRING S, STRING TAIL )
*
* CONSTRUCTS RUNTIME SEGMENT NAME FROM 'RUNTNAME' AND 'TAIL'.
* RETURNS RESULT IN 'S'.
*
  SCOPY ( S, TAIL )
  APPEND ( S, RUNTNAME )
  RETURN
END
```

```

FUNCTION RSINIT()
*
* INITIALIZE APL RUNTIME SUPERVISOR
*
  INITIALIZE()
  INIT ( FREEPOOL )
  *
  * SET TERMINAL I/O ESCAPE FUNCTIONS
  *
  \CIN _ TEST\INBUF
  \COUT _ TEST\OUTBUF
  *
  ILLEGAL\VEC $ WRD1 _ ILLEGAL
  NULL\MSG $ VDBASE _ 0
  IPTR1 _ PTRADDR ( 0 )
  *
  * WINDOWS
  *
  BUFINIT()
  TXWINDOW _ WINADDR(0)
    TXWINDOW.BUFFER _ CTXBUF
  GDBWINDOW _ WINADDR(1)
    GDBWINDOW.BUFFER _ GDBBUF
  CDBWINDOW _ WINADDR(2)
    CDBWINDOW.BUFFER _ CDBBUF
  CCDWINDOW _ WINADDR(3)
    CCDWINDOW.BUFFER _ CCDBUF
  *
  STE _ SETREFSIZE ( @ STEVEC [ 0 ], 2 )
  *
  CTRL\TERM _ CONSOLE\TERM()
  CRLF ( 1 )
  PROMPTED _ FALSE
  *
  ABSIZE _ 2000
  STSIZE _ 500
  *
  RUNFLAG _ FALSE
  LOADED _ FALSE
  RTLOADED _ FALSE
  RTSELECT ( "ARUN" )
  *
  OBJINIT()
  *
  RETURN
END

```

```
FUNCTION RTDROP()  
*  
* GET RID OF CURRENT RUNTIME  
*  
  CLEAR ( RUNTIME )  
  IF RTLOADED DO  
    CLOSE ( RTXSEG $ VDCAP )  
    CLOSE ( RCDSEG $ VDCAP )  
    CLOSE ( RDBSEG $ VDCAP )  
  ENDIF  
  RTLOADED _ FALSE  
  RETURN  
END
```

```

FUNCTION RTSELECT ( STRING RTNAME )
*
* SELECT RUNTIME LIBRARY
*
    SCALAR RGLOBLK
    REFERENCE ISTE
    IF RUNFLAG DO
        ERRMSG ( "ILLEGAL IN RUNTIME MODE&M" )
        RETURN
    ENDIF
    *
    IF STREQL ( RTNAME , PROGNAME ) DO
        ERRMSG ( "SAME AS USER PROGRAM&M" )
        RETURN
    ENDIF
    *
    IF LENGTH ( RTNAME ) > PNAMEMAX DO
        ERRMSG ( "NAME TOO LONG&M" )
        RETURN
    ENDIF
    *
    RTDROP()
    *
    SCOPY ( RUNTNAME , RTNAME )
    *
    RSEGNAME ( RTEXTNAME, TEXTTAIL )
    RSEGNAME ( RCODENAME, CODETAIL )
    RSEGNAME ( RDEBUGNAME, DEBUGTAIL )
    *
    OPEN\WINDOWED ( RTXSEG $ VDCAP, RTEXTNAME :GOTO BADTEXT )
    OPEN\LOCKED ( RCDSEG $ VDCAP, RCODENAME :GOTO BADCODE )
    OPEN\WINDOWED ( RDBSEG $ VDCAP, RDEBUGNAME :GOTO BADDEBUG )
    *
    IF RTINVALID ( GLOBLK ) DO
        SOUT ( RTNAME )
        ERRMSG ( " GLOBALLY INVALID&M" )
        GOTO BADGLOB
    ENDIF
    *
    RGLOBLK _ TRUE @ FNRUNT + GLOBLK @ FNBLK
    ISTE _ STEFIND ( "@INIT" , RGLOBLK :GOTO BADGLOB )
    INITPFD _ STEFUNCD ( ISTE )
    ISTE _ STEFIND ( "@INIT0" , RGLOBLK :GOTO BADGLOB )
    INITOPFD _ STEFUNCD ( ISTE )

    RTLOADED _ TRUE
    RETURN
    *
BADGLOB:
    CLOSE ( RDBSEG $ VDCAP )
BADDEBUG:
    CLOSE ( RCDSEG $ VDCAP )
BADCODE:
    CLOSE ( RTXSEG $ VDCAP )

```

```
BADTEXT:
  ERRMSG ( "BAD RUNTIME&M" )
  CLEAR ( RUNTNAME )
  RETURN
END
```

```

FUNCTION SBPROC()
*
* SET-BREAKPOINT COMMAND
*
*     NOTE: DOES NOT FIX SOURCE
*           DOES NOT HANDLE MULT LINES
*
  CMDMOVE ( IPTR1 :FRETURN )
  *
  UNLESS LTEGET ( DBSEG, IPTR1.BLOCK, IPTR1.LINE :FRETURN ) DO FRETURN
  IF LTE $ LTECODEPTR = LTENULL DO
    ERRMSG ( " NOT A STATEMENT-LINE&M" )
  ELSEIF LTE $ LTEBREAK = 1 DO
    ERRMSG ( "BREAKPOINT ALREADY SET&M" )
  ELSE DO
    LTE _ INBREAK ( IPTR1.BLOCK, LTE )
    LTE $ LTEBREAK _ 1
    LTEPUT ( DBSEG, IPTR1.BLOCK, IPTR1.LINE :CRASH() )
  ENDIF
  RETURN
END

```

```

FUNCTION SCALAR\CONV ( OBJ, MBOX ) RETURNING VECTOR
*
* CONVERT APL OBJECT 'OBJ' INTO A CHARACTER STRING. THE
* STRING ( PREFIXED BY ITS DESCRIPTOR ) IS STORED IN
* A NEW MESSAGE VECTOR, WHICH IS RETURNED AS VALUE.
*
    FIELD BYTE4 ( 0: 24,31 )
    SCALAR WSIZE
    VECTOR MVEC
    REFERENCE MSTR
    STRING STR [ 20 ]
    STRING CRLF _ "&M&J"
    *
    CLEAR ( STR )
    IF OBJ$APL\TYPE = TYPE\CHAR THEN
        WCI(OBJ$APL\VALUE,STR)
    ELSE
        MAX\CONV _ CFORMAT
        GWD(STR, OBJ)
    ENDIF
    IF MAILBOX\TABLE[MBOX]$MBD\GETM = GETM\QUAD THEN ..
        APPEND( STR, CRLF)
    WSIZE _ ( LENGTH ( STR ) + 3 ) / 4
    MVEC _ MAKE ( MSG\SPACE, MSG\SIZE ( WSIZE + 3 ) )
    MSTR _ MSG\SETSTRING ( MVEC )
    SCOPY ( MSTR, STR )
    *
    RETURN MVEC
END

```

```
FUNCTION SEGNAME ( STRING S, STRING TAIL )
*
* CONSTRUCTS SEGMENT NAME FROM 'PROGNAME' AND 'TAIL'.
* RETURNS NAME IN 'S'.
*
  SCOPY ( S, TAIL )
  APPEND ( S, PROGNAME )
  RETURN
END
```

```

FUNCTION SELECTPROGRAM ( STRING PNAME )
*
* SELECTPROGRAM DROPS THE CURRENT PROGRAM AND PICKS UP
* A NEW PROGRAM NAMED 'PNAME'. IT FAILS IF THERE IS
* NO SUCH PROGRAM.
*
  IF LENGTH ( PNAME ) > PNAMEMAX DO
    ERRMSG ( "NAME TOO LONG&M" )
    RETURN
  ENDIF
  *
  IF STREQ ( PNAME , RUNTIME ) DO
    ERRMSG ( "SAME AS RUNTIME&M" )
    RETURN
  ENDIF
  *
  DROP()
  *
  SCOPY ( PROGNAME , PNAME )
  *
  SEGNAME ( TEXTNAME, TEXTTAIL )
  SEGNAME ( CODENAME, CODETAIL )
  SEGNAME ( DEBUGNAME, DEBUGTAIL )
  *
  OPEN\WINDOWED ( TXSEG $ VDCAP, TEXTNAME :GOTO BADTEXT )
  OPEN\LOCKED ( CDSEG $ VDCAP, CODENAME :GOTO BADCODE )
  OPEN\WINDOWED ( DBSEG $ VDCAP, DEBUGNAME :GOTO BADDEBUG )
  *
* NEWPOINTER ( CURRENT , GLOBLK , 1 , 0 )
*
  LOADED _ TRUE
  RETURN
  *
  * INVALID PROGRAM NAME .. CLEAN UP AND FAIL.
BADDEBUG:
  CLOSE ( CDSEG $ VDCAP )
BADCODE:
  CLOSE ( TXSEG $ VDCAP )
BADTEXT:
  CLEAR ( PROGNAME )
  FRETURN
  *
END

```

```
FUNCTION SELWINDOW ( REFERENCE W , VECTOR SEG , SCALAR BLK )
*
* SELECTS CONTENTS OF WINDOW W
*
  SCALAR BUF
  *
  BUF _ W.BUFFER
  W.SEGMENT _ SEG
  W.BLOCK _ BLK
  BUFSET ( SEG , BLK , BUF )
  W.BARRAY _ BUFBASE [ BUF ]
  RETURN
END
```

```
FUNCTION SEND()  
*  
* SERVICE CALL: SEND MESSAGE TO SPECIFIED MAILBOX(ES)  
*  
  SEND\DELIVER ( FALSE )  
  ATTN\RESUME ( TRAP\PD )  
  RETURN  
END
```

```
FUNCTION SEND\ALARM(TBOX)
*
*SEND ALARM MESSAGE TO ' TBOX'
*
CONSTANT ALARM\LEN_ 4, ..
          ALARM\MSG _ 2000000376B
VECTOR ALARM _ 0, 0, 200000B, ALARM\MSG
VECTOR MSG
*
  MSG_MAKE(MSG\SPACE,ALARM\LEN)
  VCOPY(MSG,ALARM)
  ENQUEUE\MESSAGE(MSG,TBOX)
  PAIR\OFF(TBOX)
  RETURN
END
```

```

FUNCTION SEND\DELIVER ( BOOL DELIVER )
*
* ACTUAL BODY OF SEND AND DELIVER ROUTINES. PLACES MESSAGE
* IN DESIGNATED MAILBOX(ES) AND SERVICES RECEIVERS, IF ANY.
*

SCALAR BOXES, MSG, MBOX, LENGTH, I
VECTOR BOXVEC
BOOL NO\BOX
*
BOXES _ PARAM ( 1 )
MSG _ PARAM ( 2 )
*
IF LEGAL ( MBOX _ SINGLE\INTEGER ( BOXES ) ) DO
    MBOX _ MBOX $ APL\VALUE
    IF BOX\EXISTS ( MBOX ) DO
        EAT ( 3 )
        PLACE\MESSAGE ( MSG, MBOX, DELIVER )
        PAIR\OFF ( MBOX )
    ELSE DO
        BADNEWS ( NO\OBJ\ERR )
    ENDIF
ELSEIF LEGAL ( BOXVEC _ INTEGER\VECTOR ( BOXES ) ) DO
    LENGTH _ BOXVEC [ 2 ] $ APL\VALUE
    NO\BOX _ FALSE
    FOR I _ 0 TO LENGTH-1 DO
        MBOX _ BOXVEC [ I+3 ] $ APL\VALUE
        UNLESS BOX\EXISTS ( MBOX ) DO NO\BOX _ TRUE
    ENDFOR
    IF NO\BOX DO
        BADNEWS ( NO\OBJ\ERR )
    ELSE DO
        EAT ( 3 )
        FOR I _ 0 TO LENGTH-1 DO
            MBOX _ BOXVEC [ I+3 ] $ APL\VALUE
            PLACE\MESSAGE ( MSG, MBOX, DELIVER )
        ENDFOR
        FOR I _ 0 TO LENGTH-1 DO
            MBOX _ BOXVEC [ I+3 ] $ APL\VALUE
            PAIR\OFF ( MBOX )
        ENDFOR
    ENDIF
ELSE DO
    BADNEWS ( PARAM\ERR )
ENDIF
RETURN
END

```

```

FUNCTION SERVICES()
*
* CHECKS FOR SERVICE-REQUESTS
* A) ATTENTION TRAPS
* B) ATTACHED TERMINAL HAS
*   1) NON-EMPTY INPUT BUFFER
*   2) NON-FULL OUTPUT BUFFER
*
* TRAPS OTHER THAN ATTENTION TRAPS ARE PASSED
* ON TO "ERRORS()"
*
SCALAR PROCESS, TRAP\CLASS, TRAP\NUMBER
SCALAR BOX, REQ, C, I, J, T
REFERENCE TD
MACRO ATTN\TRAP _ 4 @ SA\TRAPCLASS
CONSTANT NCALLS _ 24;      * NUMBER OF SERVICE CALLS
VECTOR ATTN\PROC [ NCALLS ONE\BASED ] _ ..
    CREATE\PROCESS, DESTROY\PROCESS, ..
    CREATE\MAILBOX, DESTROY\MAILBOX, ..
    CREATE\FILE, DESTROY\FILE, ..
    ATTACH\FILE, ATTACH\TERMINAL, ..
    RECEIVE, SEND, DELIVER, ..
    SET\CONVERT, DETACH, REWIND\INFILE, ..
    SET\TIMER, RESET\TIMER, ATTACH\TIMER, ..
    SET\SUBJ\ECHO, REAL\TIME, CRECEIVE, ..
    QRECEIVE, QCRECEIVE, QDELIVER, QQDELIVER
*
* CHECK FOR TRAPS
*
T _ CATCH\TRAP()
IF T > 0 DO
    IF T = ATTN\TRAP DO
        IF LEGAL ( REQ _ SINGLE\INTEGER ( PARAM ( 0 ) ) ) DO
            REQ _ REQ $ APL\VALUE
            IF REQ < 1 OR NCALLS < REQ DO
                BADNEWS ( PARAM\ERR )
            ELSE DO
                ATTN\PROC [ REQ ] ( )
            ENDIF
        ELSE DO
            BADNEWS ( PARAM\ERR )
        ENDIF
    ELSE DO
        TRAP\PD.PD\TRAPWORD _ T
    ENDIF
ENDIF
*

```

```

* CHECK FOR ATTACHED TERMINAL I/O
*
FOR I _ 0 TO MAXTERMS-1 DO
  TD _ @ TERM\TABLE [ I ]
  UNLESS TD . D\FREE DO
    J _ TD . TD\PORT
    BOX _ TD . TD\IN\BOX
    UNTIL ( TEST\INPUT\EMPTY ( J ) OR RCVQ\EMPTY ( BOX ) ) DO
      C _ CIN ( J )
      ECHO ( C, J )
      WCI\MSG ( C, TD )
    ENDUNTIL
    *
    BOX _ TD . TD\OUT\BOX
    UNTIL ( TEST\OUTPUT\FULL ( J ) OR BOX\EMPTY ( BOX ) ) DO
      C _ GCI\MSG ( TD : EXIT )
      COU ( C, J )
    ENDUNTIL
  ENDUNLESS
ENDFOR
*
* CHECK FOR TIMER ALARMS
*
  TIMER()
*
RETURN
END

```

FUNCTION SET\CONVERT()

*

* SET INPUT-CONVERSION MODE OF SPECIFIED MAILBOX

*

SCALAR B, L

REFERENCE MBD, TD

*

IF LEGAL (B _ SINGLE\INTEGER (PARAM (1))) DO

B _ B \$ APL\VALUE

IF BOX\EXISTS (B) DO

MBD _ @ MAILBOX\TABLE [B]

IF MBD . MBD\ASTAT = MBD\TERM DO

TD _ @ TERM\TABLE[MBD.MBD\AOBJ]

IF B = TD . TD\IN\BOX DO

IF LEGAL (L _ SINGLE\INTEGER (PARAM (2))) DO

L _ L \$ APL\VALUE

IF L = 0 DO

MBD . MBD\PUTM _ PUTM\QUOTEQUAD

ATTN\RESUME (TRAP\PD)

ELSEIF L = 1 DO

MBD . MBD\PUTM _ PUTM\QUAD

ATTN\RESUME (TRAP\PD)

ELSE DO

BADNEWS (PARAM\ERR)

ENDIF

ELSE DO

BADNEWS (NOT\TINBOX\ERR)

ENDIF

ELSE DO

BADNEWS (NOT\TINBOX\ERR)

ENDIF

ELSE DO

BADNEWS (PARAM\ERR)

ENDIF

ELSE DO

BADNEWS (NO\OBJ\ERR)

ENDIF

ELSE DO

BADNEWS (PARAM\ERR)

ENDIF

RETURN

END

```

FUNCTION SET\SUBJ\ECHO()
*
* SET INPUT-CONVERSION MODE OF SPECIFIED MAILBOX
*
  SCALAR B, L
  REFERENCE MBD, TD
  *
  IF LEGAL ( B _ SINGLE\INTEGER ( PARAM ( 1 ) ) ) DO
    B _ B $ APL\VALUE
    IF BOX\EXISTS ( B ) DO
      MBD _ @ MAILBOX\TABLE [ B ]
      IF MBD . MBD\ASTAT = MBD\TERM DO
        TD _ @ TERM\TABLE[MBD.MBD\AOBJ]
        IF B = TD . TD\IN\BOX DO
          IF LEGAL ( L _ SINGLE\INTEGER ( PARAM ( 2 ) ) ) DO
            L _ L $ APL\VALUE
            IF L = 0 OR L = 1 DO
              SET\ECHO ( TD . TD\PORT, L )
            ELSE DO
              BADNEWS ( PARAM\ERR )
            ENDIF
          ELSE DO
            BADNEWS ( NOT\TINBOX\ERR )
          ENDIF
        ELSE DO
          BADNEWS ( NOT\TINBOX\ERR )
        ENDIF
      ELSE DO
        BADNEWS ( PARAM\ERR )
      ENDIF
    ELSE DO
      BADNEWS ( NO\OBJ\ERR )
    ENDIF
  ELSE DO
    BADNEWS ( PARAM\ERR )
  ENDIF
  RETURN
END

```

```

FUNCTION SET\TIMER()
*
*SERVICE CALL:
*PARAM: 2 INTEGERS (MAILBOX NUMBER, DELAY TIME)
*ACTION: CHECK BOXEXIST AND ATTACHED AS TIMER AND IS EMPTY
*
*          NO SAME BOX IN LIST
*          INSERT BOX AND TIME INTO LIST
*          CALL 'TIMER'
*FAILIF: TIME OR BOX ILLEGALE
*          BOX ALREADY ON LIST
*
CONSTANT CLOCK\FACTOR _ 1000 ;* APL CLOCK UNIT =(10^3*10^-6) .001 SEC
SCALAR TBOX, DELAY
STRING T\LIST[MAXBOXES 2 WORD]
SCALAR OLDLIST
DOUBLE TIME
FIELD NELTS(0:16,31), WD(0)
*
  IF LEGAL(TBOX _ SINGLE\INTEGER(PARAM(1))) AND ..
    LEGAL(DELAY _ SINGLE\INTEGER(PARAM(2))) THEN
      TBOX _ TBOX$APL\VALUE
      DELAY _ DELAY$APL\VALUE
      IF BOX\EXISTS(TBOX) THEN
        IF MAILBOX\TABLE[TBOX]$MBD\ASTAT = MBD\TIMER THEN
          OLDLIST _ TIMER\LIST.SDRS
          IF (NOT BOX\EMPTY(TBOX)) OR FIND\TBOX(TBOX, T\LIST) THEN
            TIMER\LIST.SDRS _ OLDLIST
            BADNEWS(ALARM\SET\ERR)
          ELSE
            TIMER\LIST.SDRS _ OLDLIST
            TIME_LMUL(DELAY,CLOCK\FACTOR)
            TIME_LADD(TIME,RTCLOCK)
            TIME$TL\TBOX_TBOX
            INSERT\TIME(TIME)
            TIMER()
            EAT(2)
            ATTN\RESUME(TRAP\PD)
          ENDIF
        ELSE
          BADNEWS(BOX\ATTACH\ERR)
        ENDIF
      ELSE
        BADNEWS(NO\OBJ\ERR)
      ENDIF
    ELSE
      BADNEWS(PARAM\ERR)
    ENDIF
  RETURN
END

```

```
FUNCTION SFREE ( VECTOR OBJECT )
*
* STANDARD FREE FUNCTION
*
  IF OBJECT $ VDBASE # 0 DO
    FREE ( FREEPOOL, OBJECT )
  ENDIF
  RETURN
END
```

```

FUNCTION SINGLE\INTEGER ( OBJ )
*
* CHECK PARAMETER: IS IT AN INTEGER SCALAR OR A
* ONE-ELEMENT INTEGER ARRAY ?
*
  SCALAR R, N
  VECTOR AVEC
  FIELD RANK ( 0: 8,15 ), NELTS ( 0: 16,31 )
  *
  IF OBJ $ APL\TYPE = TYPE\INTEGER DO
    RETURN OBJ
  ELSEIF OBJ $ APL\TYPE = TYPE\ARRAY DO
    AVEC _ ARRAY\BLOCK ( OBJ, TRAP\PD )
    IF AVEC [ 1 ] $ NELTS = 1 DO
      R _ AVEC [ 0 ] $ RANK
      N _ AVEC [ R + 2 ]
      IF N $ APL\TYPE = TYPE\INTEGER DO
        RETURN N
      ENDIF
    ENDIF
  ENDIF
  RETURN ILLEGAL
END

```

```
FUNCTION SHAKE ( SCALAR LENGTH ) RETURNING VECTOR
*
* STANDARD MAKE FUNCTION
*
  VECTOR NULLVEC
  *
  NULLVEC $ VDBASE _ 0
  *
  IF LENGTH > 0 DO
    RETURN MAKE ( FREEPOL, LENGTH )
  ELSE DO
    RETURN NULLVEC
  ENDIF
END
```

```

FUNCTION SMAKE\STRING ( STRING S, NCHARS )
*
* ALLOCATE 'NCHARS' BYTES OF STORAGE FOR
* STRING S FROM STANDARD FREEPOL
*
    VECTOR SVEC
    *
    SVEC _ SMAKE ( NCHARS/CPW + 1 )
    SVEC _ BYTEVEC ( SVEC )
    SVEC _ SUBVEC ( SVEC, 0, NCHARS+1 )
    SETSTRING ( S, SVEC )
    CLEAR ( S )
    RETURN
END

```

```
FUNCTION SPPROC()  
*  
* SELECT-PROGRAM COMMAND  
*  
  XPPROC ( SELECTPROGRAM :FRETURN )  
  RETURN  
END
```

```
FUNCTION SRPROC()  
*  
* SELECT-RUNTIME-LIBRARY COMMAND  
*  
  XPPROC ( RTSELECT :FRETURN )  
  RETURN  
END
```

```

FUNCTION STEFIND ( STRING NAME, SCALAR FD ) RETURNING REFERENCE
*
* STEFIND: FIND ENTRY IN SYMBOL TABLE
* LOCATES SYMBOL TABLE FOR FUNCTION DESCRIBED BY 'FD'.
* IT LOOKS UP 'NAME' AND RETURNS A
* REFERENCE TO THE SYMBOL TABLE ENTRY FOUND.
*
* FAILS: IF SYMBOL NOT FOUND
*
    SCALAR INDEX
    REFERENCE THISSTE
    *
    SELWINDOW ( GDBWINDOW, DEBUGSEG ( FD $ FNRUNT ), FD $ FNBLK )
    INDEX _ LOOKUP ( NAME :GOTO NOSUCH )
    THISSTE _ @ ( GDBWINDOW.BARRAY ) [ INDEX ]
    BCOPY ( STE, THISSTE, SDISP )
    RETURN STE
NOSUCH:
    FRETURN
END

```

FUNCTION STLK(BUFNO,VECTOR NAME,SIZE)

*LOOK UP SIZE WORD SYMBOL NAME STARTING AT NAME

*IN DEBUGGER BLOCK BUFFER NUMBER BUFNO

*FAILS IF NOT FOUND, SUCCEEDS AND RETURNS SYMBOL TABLE RELATIVE

*DISPLACEMENT OF ENTRY IF FOUND

DI ENTRY,I

DV ENAME, BUF

*COMPUTE HASH CODE

BUF _ BUFBASE[BUFNO]

ENTRY _ ABS(NAME[0] REM BUFREF[BUFNO].HCSIZE) + HCBASE

*SCAN CHAIN FOR NAME

FOR ENTRY _ BUF[ENTRY]\$SCHAIN WHILE ENTRY # 0 DO

*COMPARE SIZE OF NAMES FOR EQUALITY

IF BUF[ENTRY]\$SSIZE # SIZE THEN GOTO STLK1

*COMPARE NAMES FOR EQUALITY

ENAME _ SUBVEC(BUF,ENTRY+SDISP,SIZE)

FOR I _ SIZE-1 BY -1 TO 0 DO

IF NAME[I] # ENAME[I] THEN GOTO STLK1

ENDFOR

SRETURN ENTRY

STLK1: ENDFOR

FRETURN

END

```
FUNCTION STOVF\ERR ( N )  
*  
* STORAGE-OVERFLOW ERROR  
*  
  IF N = 1 DO  
    MESSAGE ( "DEPTH ERROR&M" )  
  ELSE DO  
    MESSAGE ( "WS FULL ERROR&M" )  
  ENDIF  
  RETURN  
END
```

```

FUNCTION STPROC()
*
* DUMP SYMBOL-TABLE-ENTRY
*
REFERENCE STE
SCALAR FD
STRING S [ IDLENGTH ], B [ IDLENGTH ]
BOOL FOUND
*
CMDARG ( S )
CMDARG ( B )
*
FOUND _ FALSE
FD _ FNFIND ( B: FRETURN )
IF FD > 0 DO
    FOUND _ TRUE
    STE _ STEFIND ( S, FD :FOUND _ FALSE & LEAVE )
ENDIF
*
IF NOT FOUND DO
    FOUND _ TRUE
    STE _ STEFIND ( S, GLOBLK :FOUND _ FALSE & LEAVE )
ENDIF
*
IF FOUND DO
    LVOUT ( "STYPE = ", STE.STYPE )
    LVOUT ( ", SFTYPE = ", STE.SFTYPE )
    LVOUT ( ", SLG = ", STE.SLG )
    LVOUT ( ", SPRIMITIVE = ", STE.SPRIMITIVE )
    CRLF ( 1 )
    LVOUT ( "SVAL1 = ", STE.SVAL1 )
    LVOUT ( ", SVAL2 = ", STE.SVAL2 )
    CRLF ( 2 )
ELSE DO
    SOUT ( S )
    ERRMSG ( " NOT FOUND&M" )
ENDIF
RETURN
END

```

```

FUNCTION STREQL ( STRING S1 , STRING S2 )
*
* STREQL COMPARES TWO STRINGS FOR EQUALITY
*
  STRING T1, T2
  *
  IF LENGTH ( S1 ) = LENGTH ( S2 ) DO
    SDCOPY ( T1, S1 ); SDCOPY ( T2, S2 )
    WHILE GCI ( T1 :RETURN TRUE ) = GCI ( T2 :RETURN TRUE ) DO
      ENDWHILE
    ENDIF
    RETURN FALSE
  END
END

```

```

FUNCTION STR\VEC ( STRING MSTR, VECTOR ABODY, AINDEX, PROC CONV, WIDTH)
*
*   CONVERT ONE ROW OF ARRAY
*
SCALAR BINDEX
FIELD RANK(0:8,15)
  BINDEX _ ABODY[ABODY[0]$RANK + 1]$APL\VALUE + AINDEX - 1
  IF CONV = CFORMAT THEN
    FOR I _ AINDEX TO BINDEX DO
      CMAT(MSTR, ABODY[I]: CMAT(MSTR, ABODY[I]))
    ENDFOR
  ELSEIF CONV = IFORMAT THEN
    FOR I _ AINDEX TO BINDEX DO
      IMAT(MSTR, ABODY[I], WIDTH, FALSE: CMAT(MSTR, ABODY[I]))
    ENDFOR
  ELSEIF CONV = FFORMAT THEN
    FOR I _ AINDEX TO BINDEX DO
      FMAT(MSTR, ABODY[I], WIDTH,SAVE\FRAC\WIDTH, FALSE : ..
        CMAT(MSTR, ABODY[I]))
    ENDFOR
  ELSE
    FOR I _ AINDEX TO BINDEX DO
      EMAT(MSTR, ABODY[I], WIDTH, SIGDIG-1,SAVE\EXP\WIDTH,FALSE:FRETURI
    ENDFOR
  ENDIF
  RETURN BINDEX + 1
END

```

```
FUNCTION SW(String STR, SCALAR NUM, SCALAR W:12)
*32 BIT SIGNED SIMPLE NUMBER OUTPUT AS OCTAL
  OCT(STR,ABS(NUM),NUM$SIGNF,W,TRUE :FRET(7))
  RETURN
END
```

```
FUNCTION SYSTEM\ERR ( N )
```

```
* SYSTEM ERROR
```

```
  IF N = 5 DO  
    MESSAGE ( "DOMAIN ERROR&M" )  
  ELSEIF N = 12 DO  
    MESSAGE ( "DONE&M" )  
  ELSE DO  
    SOUT ( "SYSTEM ERROR = 3," )  
    IOUT ( N ); CRLF()  
  ENDIF  
  RETURN  
END
```

```
FUNCTION SZPROC()  
*  
*  "SIZES" COMMAND  
*  
  STRING S [ 10 ]  
  SCALAR AB, ST  
  *  
  CMDARG ( S : FRETURN )  
  AB _ CSN ( S : FRETURN )  
  *  
  CMDARG ( S : FRETURN )  
  ST _ CSN ( S : FRETURN )  
  *  
  ABSIZE _ AB  
  STSIZE _ ST  
  RETURN  
END
```

```
FUNCTION TEST\INBUF ( T )
*
* DOES BUSY WAIT FOR
* SERVICE-REQUESTS UNTIL CHAR ARRIVES
*
  WHILE TEST\INPUT\EMPTY ( T ) DO
    SERVICES()
  ENDWHILE
  RETURN
END
```

```
FUNCTION TEST\OUTBUF ( T )
*
* DOES BUSY WAIT FOR SERVICE
* REQUESTS UNTIL SPACE IN T'S
* OUTPUT BUFFER IS AVAILABLE
*
  WHILE TEST\OUTPUT\FULL ( T ) DO;
    SERVICES()
  ENDWHILE
  RETURN
END
```

```

FUNCTION TIMER()
*
*CHECK TIMER\LIST AND SEND ALARM IF NECESSARY
*
BOOL AGAIN
SCALAR TBOX
FIELD WD1(1)
DOUBLE TIME, CLOCK
*
    AGAIN_TRUE
    CLOCK_RTCLOCK
    WHILE AGAIN DO
        TIME_GDND(TIMER\LIST:RETURN)
        TBOX_TIMESL\TBOX
        TIMESL\TBOX_0
        TIME_LSUB(TIME,CLOCK)
        IF TIMESWD1 >= 0 THEN
            AGAIN_FALSE
        ELSE
            GDD(TIMER\LIST)
            IF BOX\EXISTS(TBOX) THEN
                SEND\ALARM(TBOX)
            ENDIF
        ENDIF
    ENDWHILE
    RETURN
END

```

```
FUNCTION TRAP\WAIT ( REFERENCE PD )
*
* WAIT FOR PROCESS WITH GIVEN PD TO GET AN ERROR TRAP
*
  WHILE PD . PD\TRAPWORD = 0 DO
    SERVICES()
  ENDWHILE
  RETURN PD . PD\TRAPWORD
END
```

```
FUNCTION TWD(String STR, SCALAR NUM, SCALAR W:8, SCALAR D:3)
*APL NUMBER OUTPUT IN TRIMMED F FORMAT
  FMAT(STR,NUM,W,D,TRUE:FRET(81))
  RETURN
END
```

FUNCTION TYPE\ERR (N)

* CHAR-WHERE-NUMBER-EXPECTED ERROR

MESSAGE ("CHARACTER WHERE NUMBER EXPECTED&M")

RETURN

END

```
FUNCTION UNPROMPT()  
*  
* RETRACT PREVIOUS PROMPT, IF ANY  
*  
  IF PROMPTED DO  
    COUT ( CARRET )  
    PROMPTED _ FALSE  
  ENDIF  
  RETURN  
END
```

```
FUNCTION UW(STRING STR, SCALAR NUM, SCALAR W:11)
*32 BIT UNSIGNED SIMPLE NUMBER OUTPUT AS OCTAL
  OCT(STR, NUM,0,W,TRUE :FRET(6))
  RETURN
END
```

FUNCTION VALUE\ERR (N)

* UE ERROR

MESSAGE ("VALUE ERROR&I")

RETURN

END

```

FUNCTION VECTOR\CONV ( VECTOR ABODY, MBOX ) RETURNING VECTOR
*
*   CONVERT VECTOR
*
SCALAR ASIZE, WSIZE, PNUM
STRING CRLF _ "&M&J"
STRING BLK2 _ "  "
FIELD NELTS(0:16,31)
VECTOR MVEC
REFERENCE MSTR
*
  ASIZE _ ABODY[1]$NELTS
  WSIZE _ (ASIZE * 15) / 4 + 2
  MVEC _ MAKE(MSG\SPACE, MSG\SIZE(WSIZE + 3))
  MSTR _ MSG\SETSTRING ( MVEC)
*
  ASIZE _ ASIZE +2
  PNUM _ FALSE
  FOR I _ 3 TO ASIZE DO
    IF ABODY[I]$APL\TYPE = TYPE\CHAR THEN
      WCI(ABODY[I]$APL\VALUE, MSTR)
      PNUM _ FALSE
    ELSE
      IF PNUM THEN
        APPEND(MSTR,BLK2)
      ELSE
        PNUM _ TRUE
      ENDIF
      MAX\CONV _ CFORMAT
      GWD(MSTR,ABODY[I])
    ENDIF
  ENDFOR
  IF MAILBOX\TABLE[MBOX]$MBD\GETM = GETM\QUAD THEN ..
    APPEND( MSTR, CRLF)
  RETURN CONTRACT(MSG\SPACE, MVEC, ..
    MSG\SIZE((LENGTH(MSTR) + 3)/4+3))
END

```

```

FUNCTION WCI\MSG ( C, REFERENCE TD )
*
* APPEND CHARACTER 'C' TO CURRENT INPUT MESSAGE
* OF TERMINAL 'TD'
*
    SCALAR MBOX
    REFERENCE MSTRING
    VECTOR MVEC
    *
    MVEC _ TD . TD\MVEC
    IF MSG\NULL ( MVEC ) DO
        MVEC _ MAKE ( MSG\SPACE, MSG\SIZE ( LSZW + 3 ) )
        TD.TD\MVEC _ MVEC
        MSTRING _ MSG\SETSTRING ( MVEC )
    ELSE DO
        MSTRING _ SETREFSIZE ( @ MVEC [ MSG\DISP ], 3 )
    ENDIF
    *
    MBOX _ TD . TD\IN\BOX
    *
    IF ACTION\CHAR ( C, MBOX ) DO
        UNLESS C = CARRET DO
            WCI ( C, MSTRING : 0 )
        ENDUNLESS
        MSG\HDR ( MVEC ) . MSG\DELIVER _ FALSE
        *
        ENQUEUE\MESSAGE ( MVEC, MBOX )
        PAIR\OFF ( MBOX )
        *
        TD . TD\MVEC _ NULL\MSG
    ELSEIF C = '&Q' DO
        COUT ( '&133' , TD . TD\PORT )
        GCD ( MSTRING : 0 )
    ELSEIF C = '&Y' DO
        COUT ( '&135' , TD . TD\PORT )
        CLEAR ( MSTRING )
    ELSE DO
        WCI ( C, MSTRING : 0 )
    ENDIF
    RETURN
END

```

```
FUNCTION WDD(DOUBLE V, REFERENCE S)
*
*WRITE DOUBLE WORD V ON FRONT OF STRING AND DECREMENT
*
SCALAR RS
*
  RS_S.SDRS
  PUTF(S.SDVD,RS:FRETURN)_V
  S.SDRS_RS
  SRETURN
END
```

```
FUNCTION WDI(DOUBLE V, REFERENCE S)
*
*WRITE DOUBLE-WORD V ON REAR OF STRING AND INCREMENT
*
SCALAR RS
  RS_S.SDRS
  PUTR(S.SDVD,RS:FRETURN)_V
  S.SDRS_RS
  SRETURN
END
```

```

FUNCTION XPPROC ( PROC GETPROGRAM )
*
* XPPROC EXTRACTS PROGRAM NAME FROM COMMAND LINE,
* AND GETS THAT PROGRAM USING THE FUNCTION PASSED,
* WHICH SHOULD BE EITHER NEWPROGRAM, SELECTPROGRAM, OR
* RTSELECT.
*
  SCALAR CHAR
  *
  CLEAR ( TARGET )
  IF ALPHABETIC ( CHAR _ GNBCI ( CSTRING :FRETURN )) DO
    WHILE ALPHANUMERIC ( CHAR ) DO
      WCI ( CHAR , TARGET )
      CHAR _ GCI ( CSTRING :NONCHAR )
    ENDWHILE
    GETPROGRAM ( TARGET :GOTO BADNAME )
    RETURN
  ENDIF
BADNAME:
  ERRMSG ( "BAD PROGRAM NAME&M" )
  RETURN
END

```