```
**********************************************************************************
**********************************************************************************
**
**
**
**      JJJ    III    M    M   M    M   Y    Y
**       J      I     MM  MM   MM  MM   Y    Y
**       J      I     M M M M  M M M    Y Y
**       J      I     M M M M  M M M     Y
**   J   J      I     M   M M  M   M     Y
**   J   J      I     M   M M  M   M     Y
**    JJ       III    M   M M  M   M     Y
**
**
**
**
**   RRRR    EEEEE   V   V    1      666          222
**   R   R   E       V   V   11     6            2   2
**   R   R   E       V   V    1    6                 2
**   RRRR    EEEE    V   V    1    6666            2
**   R R     E       V V      1    6   6          2
**   R  R    E       VV       1    6   6   ..     2
**   R   R   EEEEE   V       111    666    ..     22222
**
**
**
**********************************************************************************
**********************************************************************************
```

DATE: NOVEMBER 21, 1978

SUBJECT: PMA_REV_16.2

1 SCOPE

THIS DOCUMENT DESCRIBES THE CHANGES MADE TO PMA AT SOFTWARE REVISION 16.2. IT SUPPLEMENTS ALL PREVIOUS PMA-RELATED DOCUMENTS RELEASED FROM ENGINEERING.

2 ERROR HANDLING

WHEN THE ASSEMBLER ENCOUNTERS AN ERROR, A MORE PRECISE AND INFORMATIVE ERROR DIAGNOSTIC IS PRINTED FOLLOWING THE OFFENDING LINE. FOR EXAMPLE:

```
                          (0001)  REL
  000000:        000000   (0002)  DATA 0
**                        (0003)  SEG
ERROR V21:  SEG/SEGR PSEUDO-OP SPECIFIED AFTER CODE HAS BEEN GENERATED

               000001     (0004)  END


TEXT SIZE:  000001 WORDS

ERRORS IN:
  0003 (V21)


0001 ERRORS (PMA-REV 16.3)
```

THE ERROR MESSAGE TEXT RESIDES IN SYSOVL>PMAERR. IF THE FILE IS MISSING OR INACCESSIBLE, AN ABBREVIATED DIAGNOSTIC IS ISSUED, OMITTING THE EXPLANATORY TEXT. (E.G., IN THE ABOVE EXAMPLE, "ERROR V21" WOULD BE PRINTED.)

3 NEW OPCODES

THE NEW OPCODES FOR THE PRIME 550 (STPM, LIOT, PTLB) HAVE BEEN ADDED.

DATE: SEPTEMBER 7, 1978

SUBJECT: REV. 16 - FTN

THIS MEMO DESCRIBES THE CHANGES AND ENHANCEMENTS TO FTN FOR REV 16.

1. ENHANCEMENTS

A. FTN GENERATES FLX AND DFLX INSTRUCTIONS FOR CERTAIN CLASSES AT ARRAY EXPRESSIONS, REPLACING PREVIOUS, MORE VERBOSE OBJECT CODE.

B. COMPILE SPEED IS INCREASED SIGNIFICANTLY DUE TO THE USE OF I/O ROUTINES WHICH TAKE ADVANTAGE OF THE PHYSICAL I/O STRUCTURE UNDER PRIMOS. THIS SPEED INCREASE IS MORE DRAMATIC IN COMPILATIONS WHICH GENERATE LISTING FILES, AND IN COMPILATIOINS ACROSS A NETWORK.

C. A NEW OPTION, - PBECB (B REGISTER BIT 2) INSTRUCTS FTN TO LOAD ECB'S INTO THE PROCEDURE FRAME IN 64V MODE PROGRAMS, THUS ALLOWING ECB'S TO BE SHARED. THIS FEATURE IS LIMITED TO SUBROUTINES: MAIN PROGRAMS ARE NOT ABLE TO TAKE ADVANTAGE OF THIS OPTION.

D. THE FOLLOWING LIBRARY ROUTINES ARE CALLED USING THE SHORT CALL (JSXB) SEQUENCE:

SIN	DSIN
COS	DCOS
ATAN	DATAN
SORT	DSORT
EXP	DEXP
ALOG	DLOG
ALOG10	DLOG10
DLOG2	

THIS FEATURE GARNERS EXECUTION SPEED INCREASES FOR PROGRAMS USING THESE SCIENTIFIC FUNCTIONS.

2. FIXES

THE FOLLOWING BUG FIXES, DIVIDED INTO THOSE REQUESTED VIA TAR'S, AND THOSE WHICH WERE MANIFESTED IN A LESS OFFICIAL MANNER, HAVE BEEN INTRODUCED INTO FTN FOR REV 16.

A. NO - TAR FIXES

1. GENERALIZED SUBSCRIPTS IN ARRAY EXPRESSIONS ARE ACCEPTED TO THE LEFT OF THE EQUALS SIGN IN ASSIGNMENT STATEMENTS. HOWEVER, DIVISION TO THE LEFT OF AN EQUALS SIGN STILL CAUSES A SYNTAX ERROR.

2. THE INCORRECT MATCHING OF EXPRESSIONS FOR PROGRAMS COMPILED IN 64R MODE HAS BEEN FIXED.

3. CERTAIN INCORRECT ARITHMETIC EXPRESSIONS CAUSED THE COMPILER TO LOOP. THIS HAS BEEN CORRECTED.

B. TARED FIXES

THESE ARE LISTED BY NUMBER:

15051 - SEQUENCE NUMBERS
15057 - MULTIPLE INDIRECTS
80276 - BAD CODE FOR COMPARISON IN 64R MODE
15229 - INT*4 OF FUNCTION PROBLEMS IN 64R MODE
80550 - TOO MANY STATEMENT FUNCTIONS
80443 - $INSERTX WITH NO BLANK

SUBJECT: REV. 16 LOADER CHANGES

1. EDB

4 TIMES FASTER.

INPUT SPECIFICATION IS REQUIRED - I.E., EDB NO LONGER DEFAULTS TO THE PAPER TYPE READER.

(PTR) AND (ASR) WILL NO LONGER BE RECOGNIZED. FOR CONSISTENCY WITH COMMAND LINE SYNTAX, -PRT AND -ASR SHOULD BE USED INSTEAD.

2. SEG

THE INTERNAL TABLES WHICH ARE COPIED INTO SEGMENT 0 OF THE SEG RUN FILE HAVE BEEN CHANGED IN ORDER TO EXPAND THE SYMBOL TABLE AREA. THEREFORE, ALL COMMAND FILES SHOULD BE RUN TO INSURE THAT THERE ARE CONFLICTS. FOR EXAMPLE, R-MODE INTERLUDE COMMANDS IN CMDNCO CAN NOT HANDLE THE NEW FORMAT UNTIL THEY HAVE BEEN REBUILT. OLD FORMAT SEG RUN FILES WILL BE CONVERTED TO THE NEW FORMAT AUTOMATICALLY BY SEG. BUFCTL NOW CONSISTS OF (SEGS*2+2 WORDS): COMMON/BUFCTL/REVFLG,BUFCNT,BUFCTL(SEGS*2). A BIT RATHER THAN A WORD IS USED TO INDICATE WHETHER OR NOT A SEGMENT SUBFILE HAS BEEN LOADED INTO. REVFLG WILL BE PRESENT FROM NOW ON. IT IS SET TO -1 AS A FLAG THAT TABLE CONVERSION WILL NOT BE NECESSARY. CURRENTLY, SEGS=256. THERE ARE 32 SUBFILES PER SEGMENT.

SEG CHECK FOR LOAD* OR VLOAD* TYPING ERRORS WHICH USED TO RESULT IN THE RUN FILE BEING DELETED. COMMON BLOCKS LONGER THAN ONE SEGMENT NO LONGER HAVE TO BEGIN AT UND ZERO. MULTIPLE STACK ALLOCATION WILL NO LONGER RUN. THE MIX OPTION CAN BE USED WITH ARRAYS OVER 64K. THE R-MODE INTERLUDE PROGRAMS WILL FXIT GRACEFULLY SHOULD CONTROL RETURN TO RUNIT.

BUGS FIXED

TAR25528- UPDATE SYMBOL TABLE SIZE PRIOR TO WRITING OUT SEGMENT 0

TAR25724- DO NOT ASSIGN STACK SEGMENT

TAR25532- DOUBLE PRECSION ADD SO THAT COMMON BLOCKS LONGER THAN ONE SEGMENT NO LONGER HAVE TO BEGIN AT WORD 0.

TAR25533- MIX OPTION/LONG COMMON BUG FIXED

TAR12731- CHECK FOR LOAD/VLOAD* TYPING ERROR

CMDMAK AND CM.FILE HAVE BEEN FIXED TO CALL EXIT UPON RETURN FROM RUNIT IN THE R-MODE INTERLUDE PROGRAM

DIRECT COMMON REFERENCE CONVERSION HAS BEEN FIXED.

3. LOAD

SYMBOLS MAY HAVE 8-CHARACTER NAMES.

RR (RESET RANGE) CAN BE USED TO RESET THE SAVE RANGE PRIOR TO EN (ENTIRE SAVE) WHEN OVERLAYS ARE BUILT.

LINKING IN COMMON IS NOW ALLOWED WHILE FORWARD REFERENCES ARE BEING UNSTRUNG.

BUGS FIXED

LOAD ALLOWS LINKING IN COMMON WHEN UNSTRING FORWARD REFERENCES. LOAD WILL NOW GIVE A CORRECT EOF ERROR MESSAGE WHEN AN ATTEMPT IS MADE TO LOAD A NULL FILE.

A FIX HAS BEEN MADE TO REMOVE THE CODE,CODE ARGUMENT SEQUENCE IN PRWF$$ CALLS

LOAD HAS BEEN FIXED SO BITS DIPLAYED IN *UII ARE CORRECT.

SUBJECT: CHANGES TO SORT LIBRARIES A. VSRTLI (V-MODE SORT LIBRARY) CHANGES AT REVS 15.3 AND 16.0

1. FOR CONSISTENCY WITH THE R-MODE SORT LIBRARY, CALLS TO THE SUBROUTINE ASCSRT MAY NOW BE MADE AS CALLS TO THE SUBROUTINE ASCS$$.

2. THE V-MODE SORT LIBRARY'S INTERNAL ROUTINE SPACE HAS BEEN RENAMED SPAC$S TO AVOID NAMING CONFLICTS WITH USERS.

B. PROPOSED NAMING CONVENTION

1. ADOPTION OF A NAMING CONVENTION SIMILAR TO THAT OF THE APPLICATION LIBRARY WOULD BE BENEFICIAL IN AVOIDING THE POSSIBILITY OF A CONFLICT WITH USER WRITTEN ROUTINES AND SYSTEM ROUTINES.

2. EXISTING ENTRY POINTS: SUBSRT, ASCS$$, ASCSRT (V-MODE ONLY), AND COMMON BLOCK NAMES: EB$1, EB$2, EB$3, EB$4, EB$5, WOULD NOT BE CHANGED, BUT ALL OTHER NAMES WOULD END WITH THE SUFFIX "$S".

3. I WOULD APPRECIATE YOUR COMMENTS, PARTICULARLY CONCERNING ANY PROBLEMS THIS SCHEME MIGHT CAUSE.

SUBJECT: EVENT LOGGING IN PRIMOS III, IV, AND V

ABSTRACT

EVENT LOGGING IN PRIMOS IS A MECHANISM WHEREBY MACHINE CHECKS, DISK ERRORS, AND CERTAIN OTHER SIGNIFICANT EVENTS ARE RECORDED IN A DISK FILE CALLED LOGREC. A UTILITY PROGRAM -- LOGPRT -- IS AVAILABLE TO FORMAT AND PRINT THE CONTENTS OF LOGREC. THIS DOCUMENT DESCRIBES THE LOGGING MECHANISM, THE USE OF LOGPRT, AND HOW THE LOGGING MECHANISM MAY BE MODIFIED TO ADD NEW EVENT TYPES.

\NOTE: WHENEVER A NEW REVISION OF PRIMOS IS INSTALLED, THE \CORRESPONDING REVISION OF LOGPRT SHOULD BE INSTALLED, SINCE NEW EVENT \TYPES MAY HAVE BEEN DEFINED THAT AN OLDER LOGPRT DOES NOT UNDERSTAND.

REVISION 3 OF THIS PE-T IS A COMPLETE UPDATE OF REVISION 2; NEW MATERIAL IS INDICATED WITH REVISION BARS.

THIS REVISION CORRESPONDS TO REVISION 16.2 OF PRIMOS IV AND V.

THIS PAGE FOR TABLE OF CONTENTS

1 GENERAL INFORMATION

1.1 FIRST-LEVEL EVENT LOGGER -- LOGEV1

INFORMATION ABOUT AN EVENT IS ENTERED INTO AN EVENT BUFFER -- LOGBUF -- BY LOGEV1 -- AN INTERNAL PRIMOS SUBROUTINE. EACH ENTRY IN THE BUFFER CONTAINS THE TYPE AND LENGTH OF THE ENTRY AND A NUMBER OF DATA WORDS PASSED TO LOGEV1 BY THE ROUTINE WISHING TO RECORD THE EVENT. (THE EXACT FORMAT OF EVENT ENTRIES IS DESCRIBED BELOW.) WHEN LOGBUF FILLS UP, LOGEV1 DISCARDS SUBSEQUENT ENTRIES AND INCREMENTS LOGOVF -- A COUNTER OF THE NUMBER OF EVENTS LOST.

\ LOGEV1 IS CALLED FROM THE CHECK HANDLERS IN SEG4, DOSSUB, DVDISK,
\ AND PABORT.

1.2 SECOND-LEVEL LOGGER -- LOGEV2

EVERY MINUTE THE SECOND-LEVEL HANDLER, LOGEV2, EXAMINES LOGBUF AND, IF IT IS NON-EMPTY, WRITES IT TO A DISK FILE NAMED 'LOGREC' IN THE CURRENT UFD OF USER 1 (NORMALLY CMDNCO ON THE COMMAND DEVICE. LOGEV2 WILL NOT DUMP LOGREC UNTIL THE TIME HAS BEEN SET BY THE SYSTEM OPERATOR. LOGEV2 IS CALLED FROM TWO PLACES IN PRIMOS: PABORT WHEN THE ONE-MINUTE PROCESS ABORT OCCURS, AND DOSSUB WHEN A 'SHUTDN ALL' COMMAND IS ISSUED.

LOGEV2 DOES NOT DUMP LOGBUF IF THE FILE LOGREC DOES NOT EXIST IN CMDNCO OR IF THE CONFIGURATION COMMAND LOGREC HAS BEEN USED TO SET THE LOGREC QUOTA TO A NEGATIVE VALUE (SEE BELOW). THIS ALLOWS OPERATION WITH A WRITE-PROTECTED COMMAND DEVICE. (NOTE: IF THE COMMAND DEVICE IS WRITE-PROTECTED AND A LOGREC FILE EXISTS IN CMDNCO AND A 'LOGREC 177777' HAS NOT BEEN ISSUED, A DISK WRITE-PROTECT ERROR MESSAGE WILL BE PRINTED ON THE SYSTEM CONSOLE EVERY MINUTE.)

THE LOGREC FILE CAN BE CREATED WITH ANY SEQUENCE OF COMMANDS EQUIVALENT TO:

```
L 'CMDNCO PASSWORD>LOGREC'
C 2
```

BEFORE DUMPING LOGBUF, LOGEV2 WRITES AN ENTRY TO LOGREC NOTING THE CURRENT TIME AND DATE. AFTER LOGBUF IS DUMPED, IF LOGOVF (THE OVERFLOW COUNTER) IS NON-ZERO, LOGEV2 WRITES AN ENTRY NOTING THE NUMBER OF LOGBUF OVERFLOWS.

NOTE: WHENEVER POSSIBLE, A WARM START SHOULD BE PERFORMED AFTER A MACHINE HALT. THIS WILL GIVE LOGEV2 A CHANCE TO DUMP LOGBUF, EITHER AFTER ONE MINUTE OR ON A 'SHUTDN ALL' COMMAND.

1.3 THE LOGREC CONFIGURATION COMMAND

CERTAIN ACTIONS OF LOGEV2 CAN BE CONTROLLED BY THE LOGREC CONFIGURATION COMMAND. THE FORMAT OF THIS COMMAND IS:

```
LOGREC <VAL>
```

<VAL>, IF POSITIVE, SPECIFIES THE NUMBER OF WORDS IN THE LOGREC FILE. WHEN LOGREC EXCEEDS <VAL> WORDS, LOGEV2 PRINTS:

```
EXCEEDING QUOTA ON LOGREC
```

ON THE SYSTEM CONSOLE EACH TIME LOGBUF IS WRITTEN TO LOGREC.

SPECIFYING A <VAL> OF 0 WILL INHIBIT THE QUOTA CHECK; NO MESSAGE WILL EVER BE PRINTED.

SPECIFYING A NEGATIVE <VAL> (E.G., 177777) WILL SUPPRESS ALL ATTEMPTS TO WRITE TO THE LOGREC FILE. THIS WILL AVOID DISK WRITE ERRORS IF RUNNING ON A WRITE-PROTECTED DISK.

THE DEFAULT VALUE OF <VAL> IS 10000 (4096 DECIMAL). THIS COMMAND IS USED TO SET THE VARIABLE LRQUOT IN FIGCOM.

1.4 LOGPRT -- DUMP CONTENTS OF LOGREC

THE THIRD LEVEL OF THE EVENT LOGGING MECHANISM IS LOGPRT -- A PROGRAM THAT DUMPS THE CONTENTS OF LOGREC TO A DISK FILE OR A USER TERMINAL. THE LOGPRT PROGRAM IS IN THE UFD SYSTEM ON VOLUME 1 OF THE MASTER DISK. THE COMMAND LINE TO INVOKE LOGPRT IS AS FOLLOWS ([] INDICATES OPTIONAL PARAMETER):

```
R *LOGPRT [<OUTTREENAME>] [<OPT> <OPT> ...]
```

<OUTTREENAME> THE DESTINATION FOR LOGPRT'S OUTPUT. IF 'TTY' IS SPECIFIED, THE OUTPUT WILL BE TO THE USER'S TERMINAL. IF <OUTTREENAME> IS OMITTED, OUTPUT WILL BE TO THE FILE 'LOGLST' IN THE CURRENT UFD. ANY OTHER SPECIFICATION WILL BE TAKEN AS A TREENAME TO WHICH THE OUTPUT WILL BE DIRECTED.

<OPT> AN OPTION KEYWORD, POSSIBLY FOLLOWED BY SUBFIELDS. ALL OPTION KEYWORDS BEGIN WITH A HYPHEN AND MAY BE ABBREVIATED TO A UNIQUE LEFT SUBSTRING (WITH THE EXCEPTION OF THE -PURGE OPTION).

-HELP - A LIST OF LOGPRT OPTIONS IS PRINTED. THE LOGPRT COMMAND MUST BE RETYPED AFTER THE OPTIONS ARE PRINTED.

-INPUT <TRNAME> - SPECIFY TREENAME OF LOGREC FILE TO PROCESS. IF THIS OPTION IS OMITTED, A PROMPT IS ISSUED FOR THE TREENAME.

-FROM MMDDYY - ONLY LOGREC ENTRIES FROM THE SPECIFIED DATE TO THE LATEST ENTRY ARE PROCESSED.

-TYPE T1 T2 ... - PROCESS ENTRIES ONLY OF THE INDICATED TYPES. THE TYPES (T1, T2, ETC) CAN BE ANY OF THE FOLLOWING (ANY UNIQUE ABBREVIATIONS ARE ACCEPTABLE):

COLD	COLD STARTS
WARM	WARM STARTS
TIMDAT	TIME/DATE ENTRIES
CHECKS	MACHINE CHECKS (INCLUDING MEMORY PARITY)
POWERF	POWER FAIL CHECKS
DISK	DISK ERRORS
DSKNAM	ADDISK OR STARTU ENTRIES
OVERFL	LOGREC OVERFLOW ENTRIES
SHUTDN	OPERATOR SHUTDOWNS
CHK300	P300 MACHINE CHECKS
PAR300	P300 MEMORY PARITY CHECKS
MOD300	P300 MISSING MEMORY MODULE CHECKS
TYPE10-TYPE15	ENTRIES FOR TYPES 10-15

NOTE THAT THE TIME/DATE STAMPS ASSOCIATED WITH THE SELECTED ENTRIES WILL NOT BE PROCESSED UNLESS TIMDAT IS EXPLICITLY SELECTED, FOR EXAMPLE, '-T D T' WILL PROCESS ALL DISK ERRORS AND THEIR ASSOCIATED TIME/DATE STAMPS. IF TIMDAT ALONE IS SPECIFIED, ALL TIME/DATE STAMPS IN LOGREC WILL BE PROCESSED. IF TIMDAT IS SPECIFIED IN CONJUNCTION WITH ONE OR MORE OTHER TYPES, ONLY THE TIME/DATES OF THE SELECTED TYPES WILL BE PROCESSED. IF THE -TYPE OPTION IS NOT SPECIFIED, ALL ENTRIES WILL BE PROCESSED.

-SPOOL - (PRIMOS III AND IV ONLY) SPOOL THE OUTPUT FILE WHEN DONE. LOGPRT WILL PRINT THE NAME OF THE OUTPUT SPOOL FILE AND A LONG/SHORT INDICATION.

-DELETE - DELETE THE OUTPUT FILE WHEN DONE (MAKES SENSE ONLY WHEN USING THE -SPOOL OPTION).

-PURGE - EMPTY LOGREC WHEN DONE (THIS OPTION CANNOT BE ABBREVIATED). OWNER RIGHTS ARE REQUIRED ON LOGREC.

-CONTIN - CONTINUE AFTER BAD ENTRY IS FOUND. LOGPRT WILL NORMALLY HALT IF AN INVALID ENTRY IS ENCOUNTERED IN LOGREC. IF THIS OPTION IS SPECIFIED, LOGPRT WILL CONTINUE PROCESSING IN AN ATTEMPT TO FIND THE NEXT VALID ENTRY.

-DRUG - THIS OPTION CAUSES LOGPRT TO READ ENTRIES FROM THE TERMINAL AND CAN BE USED FOR TESTING LOGPRT'S FORMATTING FOR NEW (OR OLD) ENTRY TYPES. EACH ENTRY SHOULD BE ENTERED AS A SERIES OF TOKENS (USING RDTK$$'S RULES). OCTAL TOKENS ARE CONVERTED TO BINARY; ALL OTHERS ARE TAKEN AS ASCII STRINGS AND TRUNCATED TO THE LEFTMOST TWO CHARACTERS. LOGPRT LEAVES THIS MODE OF OPERATION WHENEVER A TOKEN

STARTING WITH A HYPHEN IS ENTERED. THE -DBUG OPTION ALSO TURNS ON TTY OUTPUT AND THE -CONTIN OPTION.

IF LOGPRT FINDS THAT THE OUTPUT FILE ALREADY EXISTS, IT WILL PRINT THE MESSAGE:

OK TO DELETE OLD <OUTTREENAME> (Y OR N):

THE REPLY SHOULD BE 'Y' TO DELETE THE FILE OR 'N' TO ENTER A NEW DESTINATION. IF 'N' IS ENTERED, THE MESSAGE

NEW SPECIFICATION:

IS PRINTED. ALL PARAMETERS FOLLOWING THE 'R LOGPRT' MAY BE REENTERED.

FINALLY, IF NO '-I' OPTION WAS SPECIFIED, LOGPRT PRINTS THE MESSAGE:

INPUT TREENAME:

THE TREENAME OF THE LOGREC FILE TO BE PRINTED SHOULD BE ENTERED. IF A NULL LINE IS ENTERED, <0>CMDNC0>LOGREC WILL BE ASSUMED.

2 LOGPRT PROCESSING

LOGPRT FIRST OUTPUTS A HEADER LINE CONTAINING THE TREENAME OF THE INPUT FILE AND THE CURRENT TIME AND DATE. FOR EXAMPLE:

***** <0>CMDNC0>LOGREC, 09:23:44 TUE 12 DEC 1978 *****

THE HEADER IS FOLLOWED BY FORMATTED ENTRIES, ONE OR MORE LINES PER ENTRY. THE FOLLOWING ENTRIES ARE CURRENTLY DEFINED. (ALL NUMBERS ARE IN OCTAL EXCEPT WHERE NOTED. BRACKETS ([]) SURROUND INFORMATION THAT MAY NOT BE PRESENT FOR ALL CPU MODELS OR REVISIONS OF PRIMOS.)

09:01:20 WED 16 FEB 1977

THIS IS A DATE/TIME ENTRY ENTERED BY LOGEV2 WHEN LOGBUF WAS DUMPED TO LOGREC. ALL EVENTS FOLLOWING THIS ENTRY AND BEFORE THE NEXT DATE/TIME ENTRY OCCURRED DURING THE MINUTE JUST PRIOR TO THE TIME SHOWN.

COLD START [CPU TYPE= T MICROCODE REV= MM ID= IIIIII ...]

A COLD START OF PRIMOS WAS PERFORMED. IF RUNNING UNDER REV 16.2 (OR LATER) OF PRIMOS, A COLD START ENTRY CONTAINS 8 WORDS OF INFORMATION OBTAINED FROM THE STORE PROCESSOR MODEL NUMBER (STMP) INSTRUCTION (SEE PE-TN-204). 'CPU TYPE' INDICATES THE CPU AS FOLLOWS:

```
\
\          TYPE   MODEL NUMBER
\            0      P400
\            1      RESERVED
\            2      RESERVED
\            3      P350
\            4      P450I (P450, P400T)
\            5      P560 (P550, P520T)
\            6      P500X (P500)
\            7      RESERVED
\
\     'MM' INDICATES THE REVISION OF MICROCODE RUNNING;  'XXXXXX ...' IS
\     THE FULL 8-WORD ID FROM THE STMP INSTRUCTION.
```

WARM START

```
      A WARM START OF PRIMOS WAS PERFORMED.

MACHINE CHECK (XXX) DSWSTAT= SSSSSS SSSSSS DSWRMA= YYYYY RRRRRR  RRRRRR
\     DSWPB= PPPPPP PPPPPP [ DWPARITY= XXXXXX XXXXXX ... ]

\     A  MACHINE  CHECK OCCURRED.  DSWSTAT, DSWRMA, DSWPB, AND DSWPARITY
\     CONSTITUTE THE DSW AT THE TIME OF THE  CHECK.  DSWPARITY  IS  NOT
\     PRESENT  ON ALL CPU MODELS.  IF DSWPARITY IS NOT PRESENT, 'XXX' IS
\     AN ENCODING OF THE MACHINE CHECK CODE  AND  'NOT  RCM  PARITY'  IN
\     DSWSTATH AS FOLLOWS:

\        BPD    PERIPHERAL DATA OUTPUT
\        BPAI   PERIPHERAL ADDRESS INPUT
\        BMD    MEMORY DATA OUTPUT
\        RCD    CACHE DATA
\        BPAO   PERIPHERAL ADDRESS OUTPUT
\        RDXI   RDX-BPD INPUT
\        BMA    MEMORY ADDRESS
\        RF     REGISTER FILE
\        RCM    RCM PARITY ERROR (XCS ONLY)

\     IF  THE  RMA  INVALID  BIT  IS SET (BIT 9 OF DSWSTATL), 'YYYYY' IS
\     '(INV)', OTHERWISE 'YYYYY' IS ABSENT.

\     IF DSWPARITY IS PRESENT, IT IS BROKEN DOWN BY REPORTING BOARD  (A,
\     C,  CS,  D)  AND  SIGNAL NAME AS FOLLOWS.  (NOTE:  ALL SIGNALS ARE
\     REPORTED IN THE  POSITIVE  SENSE.   FOR  EXAMPLE,  IF  'RCMPE'  IS
\     PRINTED, IT MEANS THAT THE SIGNAL 'RCMPE-' WAS 0.)

\     DSWPARITYH
\
\     01 - RPARERR1+     CS  DMX INPUT E6: BPD OR BURST- R0,R2
\                                    E5: BPD OR BURST- R0,R1,R2,R3
\                            DMX OUTPUT  : BMD
\     02 - RPARERR2+     CS  DMX INPUT F6: BPD OR BURST- R1,R3
\                                    E5: BPD
\                            DMX OUTPUT  : BMA
```

```
\      03 - FBDMX+          CS  BURST-MODE DMX TRANSFER
\      04 - BURST-INPUT+ CS  1=DMX INPUT, 0=DMX OUTPUT
\
\      05,06,07 - 0 - FPDPE+     D   PERIPHERAL REPORTS BPD ERROR (OUTPUT)
\                 1 - FBRFHPE+   D   BASE REGISTER FILE HIGH
\                 2 - FMDPE+     D   MEMORY REPORTS BMD ERROR (WRITE)
\                 3 - FIPBAPE+   D   PREFETCH BUFFER ADDRESS
\                 4 - FPAPE+     D   PERIPHERAL REPORTS BPA ERROR (OUTPUT)
\                 5 - FBRFLPE+   D   BASE REGISTER FILE LOW
\                 6 - FMAPE+     D   MEMORY REPORTS BMA ERROR
\                 7 - FIPBIPE+   D   PREFETCH BUFFER INSTRUCTION
\      08 - RCMPE-          A   RCM PARITY IF NO BOARD REPORTED ERROR
\      09 - FMDECCU+        D   MEMORY REPORTS ECC UNCORRECTABLE READ ERROR
\      10 - GDBDPE-         D   PREFFTCH BOARD DETECTED ERROR
\      11 - BPAIPE+         A   BPA INPUT ERROR (DMX OR INTERRUPT)
\      12 - FRDXPE+         A   RDX ERROR WHEN MOST RECENTLY CLOSED
\      13 - FRFPE+          A   REGISTER FILE ERROR
\      14 - FREAPE+         A   REAH OR REAL ERROR
\      15 - FDMX+           D   DMX CYCLE AT TIME OF ERROR
\      16 -
\
\
\      DSWPARITYL
\
\      01 - GCBDPE-         C   C BOARD DETECTED ERROR
\      02 - FBMDEVPE+       C   BMD INPUT EVEN WD
\      03 - FBMDODPE+       C   BMD INPUT ODD WD
\      04 - LMMOD+          C   MISSING MEMORY MODULE AT CACHE-MISS
\      05 - LBMAPE+         C   MEMORY REPORTS BMA ERROR AT CACHE-MISS
\      06 - LFERNEXT-       C   LSB ADDR TO MEMORY AT ERROR (CACHE-MISS)
\      07 - LFLRMAL15+      C   LSB ADDR TO MEMORY AT START OF CACHE-MISS
\      08 - LMISFL16+       C   INDICATOR OF WHICH MEMORY MODULE WAS ACTIVATE
\      D
\      09 - LBMDECCU+       C   MFMORY REPTS ECC UNCORRECTABLE ON CACHE-MISS
\      10 - LBMDECCC+       C   MEMORY REPTS ECC CORRECTABLE ON CACHE-MISS
\      11 - LRCIAPE+        C   CACHE-INDEX ERROR ON CACHE-READ
\      12 - LRCDODPE+       C   CACHE-DATA-ODD WORD ERROR ON CACHE-READ
\      13 - LRCDEVPE+       C   CACHE-DATA-EVEN WORD ERROR ON CACHE-READ
\      14 - LFSERVDBD-      C   PURPOSE OF CACHE CYCLE: 1=EXECUTE, 0=PREFETCH
\      15 -
\      16 -
```

MISSING MEMORY DSWSTAT= ...

A MISSING MEMORY MODULE CHECK OCCURRED. INFORMATION IS AS FOR A MACHINE CHECK EXCEPT THE MACHINE CHECK CODE (XXX) DOES NOT APPEAR
\ AND DSWPARITY IS NOT DECODED.

MEMORY PARITY (XXXX) DSWSTAT= ... PPN,WN= PPPPPP WWWWW

A MEMORY PARITY ERROR OCCURRED. 'XXXX' IS EITHER 'ECCC' (CORRECTED) OR 'ECCU' (UNCORRECTED). 'PPN,WN=PPPPPP WWWWW' IDENTIFIES THE PHYSICAL PAGE AND WORD NUMBER OF THE ERROR. FOR AN

ECCC ERROR, THE PPN IS FOLLOWED BY 'BIT=XX', WHERE 'XX' IDENTIFIES THE BIT IN ERROR -- 1-15 FOR BITS 1-15, RP FOR RIGHT PARITY, C2, C4, C5 FOR OTHER CHECK BITS, MB FOR MULTIBIT, NE FOR NO ERROR. (THIS IS TAKEN FROM THE ECCC SYNDROME FIELD IN DSWSTATL.) FOLLOWING THE BIT IDENTIFICATION IS 'OP=X', WHERE X IS 0 OR 1 AND REFLECTS THE SETTING OF DSWSTATL BIT 6 (OVERALL PARITY). DSWPARITY IS DISPLAYED BUT NOT DECODED.

\POWER FAIL CHECK
\
\ A POWER FAIL CHECK OCCURRED.

DISK XX ERROR DVNO= DDDDDD (TYPECODE) CRA= RRRRRR RRRRRR CYL= CCC HEAD= HH RECORD= RR RCRA= AAAAAA AAAAAA STATUS (ACCUM)= SSSSSS STATUS (LAST)= LLLLLL RETRIES= TT MMMMMM

A DISK ERROR OCCURRED DURING AN 'XX' OPERATION, WHERE 'XX' IS 'RD' FOR READ OR 'WT' FOR WRITE. DVNO GIVES THE DEVICE NUMBER. 'TYPECODE' GIVES THE CONTROLLER NUMBER AND DEVICE TYPE (MHD => MOVING HEAD DISK, FHD => FIXED HEAD DISK, SM => STORAGE MODULE). CRA GIVES THE RECORD ADDRESS, WHICH IS BROKEN UP INTO CYL (CYLINDER), HEAD, AND RECORD ADDRESS (ALL IN DECIMAL). FOR A READ OPERATION, RCRA GIVES THE CRA READ ON A CRA ERROR. STATUS (ACCUM) IS THE OR OF ALL STATUS BITS OBTAINED DURING RETRIES. STATUS (LAST) IS THE STATUS OF THE LAST OPERATION.

RETRIES GIVES THE NUMBER OF RETRIES ATTEMPTED. IF RETRIES IS LESS THAN 10, THE OPERATION WAS COMPLETED SUCCESSFULLY -- MMMMMM WILL BE '(RECOVERED)'. IF RETRIES = 10 AND THE ERROR COULD NOT BE CORRECTED BY ECC, MMMMMM IS '(UNCORRECTABLE)'. IF AN ECC ERROR HAS BEEN SUCCESSFULLY CORRECTED BY THE SOFTWARE, MMMMMM IS WORDNO= AND CORRECTION=, WHICH GIVE THE WORD NUMBER IN THE RECORD AND THE 32-BIT CORRECTION PATTERN USED.

DISK MOUNT: PACKNAME ON DVNO

AN ADDISK OR STARTU COMMAND WAS ISSUED. THE INDICATED PACKNAME WAS MOUNTED ON THE DISK IDENTIFIED BY 'DVNO'.

MACHINE CHECK USER= NN PC= PPPPPP

A PRIME 300 MACHINE CHECK OCCURRED. USER GIVES THE USER NUMBER (DECIMAL), PC GIVES HIS PC AT THE TIME OF THE CHECK.

MEMORY PARITY

A PRIME 300 MEMORY PARITY ERROR OCCURRED (SEE ALSO NEXT ENTRY).

MEMORY PARITY PPN= PPPPPP WN= WWWWWW CONTENTS= CCCCCC

A PRIME 300 MEMORY PARITY ERROR WAS ENCOUNTERED DURING A WARM START MEMORY SCAN. GIVEN ARE THE PHYSICAL PAGE NUMBER (PPN), WORD NUMBER OFFSET IN THE PAGE (WN), AND INCORRECT CONTENTS.

MISSING MEMORY

A PRIME 300 MISSING MEMORY CHECK OCCURRED.

LOGBUF OVERFLOW -- NNNNN ENTRIES LOST

'NNNNN' (DECIMAL) EVENT ENTRIES WERE LOST DUE TO OVERFLOW OF LOGBUF.

SHUTDOWN BY OPERATOR

THE OPERATOR ISSUED A 'SHUTDN ALL' COMMAND. (THIS AUTOMATICALLY DUMPS LOGBUF.)

\TYPE= TT DATA= DDDDDD ...
\
\ A LOGREC ENTRY OF TYPE 10-15 WAS ENCOUNTERED. 'TT' INDICATES THE
\ TYPE OF THE ENTRY; 'DDDDDD ...' IS A DISPLAY OF UP TO 9 WORDS OF
\ INFORMATION FROM THE ENTRY.

*** LOGREC EMPTY ***

THIS MESSAGE IS PRINTED IF LOGPRT FINDS NO ENTRIES IN LOGREC.

*** END OF LOGREC -- NNNNN ENTRIES, PPPPP PROCESSED ***

THIS MESSAGE IS PRINTED WHEN LOGPRT REACHES THE END OF LOGREC. 'NNNNN' (DECIMAL) GIVES THE NUMBER OF ENTRIES IN LOGREC NOT INCLUDING DATE/TIME AND LOGBUF OVERFLOW ENTRIES. 'PPPPP' GIVES THE NUMBER OF ENTRIES PROCESSED.

WHEN ALL THE ENTRIES IN LOGREC (OR OTHER INPUT FILE) HAVE BEEN PROCESSED, LOGPRT WILL NORMALLY CLOSE THE FILE AND EXIT. IF, HOWEVER, THE -PURGE OPTION HAS BEEN SPECIFIED LOGPRT WILL POSITION TO THE BEGINNING OF THE FILE BEFORE CLOSING, IN EFFECT EMPTYING THE FILE.

FINALLY, IF THE SPOOL OPTION IS IN EFFECT, LOGPRT SENDS THE OUTPUT FILE TO THE SPOOL PROGRAM AND PRINTS THE NAME OF THE RESULTING SPOOL FILE. IF THE DELETE OPTION IS IN EFFECT, THE OUTPUT FILE IS THEN DELETED.

3 MODIFYING THE EVENT LOGGING MECHANISM

THE FOLLOWING TELLS HOW TO MAKE MODIFICATIONS TO THE EVENT LOGGING MECHANISM. THE RELEVANT MODULES ARE FOUND AS FOLLOWS: FOR PRIMOS IV, LOGEV1 AND LOGBUF ARE IN PRI400>KS>SEG4. LOGEV2 IS PRI400>KS>LOGEV2. FOR PRIMOS III, LOGEV1 AND LOGBUF ARE IN PRI300>KS>TMAIN, LOGEV2 IS PRI300>KS>LOGEV2. FOR BOTH PRIMOS III AND IV, LOGPRT IS IN SYSTEM.

3.1 INCREASING THE SIZE OF LOGBUF

LOGBUF IS DEFINED IN SEG4 (PRIMOS IV) OR TMAIN (PRIMOS III). THE FIRST ENTRY IN THE BUFFER (LABEL LOGBUF) IS A 9-WORD COLD START ENTRY. THE FOLLOWING BSZ DEFINES THE REMAINING SIZE OF LOGBUF (CURRENTLY 63). IT CAN BE REDEFINED AS DESIRED.

3.2 ADDING EVENT TYPES

TO LOG A NEW EVENT TYPE, THREE ACTIONS ARE NECESSARY:

1) AN EVENT MESSAGE MUST BE BUILT THAT CONTAINS THE EVENT TYPE, LENGTH OF THE MESSAGE, AND (OPTIONAL) DATA WORDS.

2) LOGEV1 MUST BE CALLED TO ENTER THE MESAGE INTO LOGBUF.

3) LOGPRT MUST BE MODIFIED TO RECOGNIZE THE NEW EVENT TYPE AND APPROPRIATELY FORMAT THE DATA ASSOCIATED WITH THE EVENT. (NOTE THAT LOGEV1 AND LOGEV2 DO NOT EXAMINE THE TYPE FIELD.)

3.2.1 EVENT MESSAGE FORMAT

AN EVENT MESSAGE CONSISTS OF A HEADER WORD FOLLOWED BY UP TO 23 OPTIONAL DATA WORDS. THE HEADER WORD CONSISTS OF THE EVENT TYPE IN BITS 1-8 AND THE TOTAL MESSAGE LENGTH IN BITS 9-16. IN PMA, A MESSAGE COULD BE DEFINED BY:

```
MSG   DATA   (5.LS.8)+3,DATA1,DATA2
```

THIS DEFINES A MESSAGE FOR EVENT TYPE 5, LENGTH OF MESSAGE (INCLUDING HEADER WORD) IS 3 WORDS.

3.2.2 CURRENTLY DEFINED EVENT TYPES

CURRENTLY, THE FOLLOWING EVENT TYPES ARE DEFINED.

```
 0 - COLD START
 1 - WARM START
 2 - DATE/TIME STAMP (LOGEV2)
 3 - CHECKS (MACHINE, MEMORY PARITY, MISSING MEMORY)
 4 - DISK ERRORS
 5 - LOGBUF OVERFLOW (LOGEV2)
 6 - SHUTDN ALL
 7 - PRIME 300 MACHINE CHECK
 8 - PRIME 300 MEMORY PARITY
 9 - PRIME 300 MISSING MEMORY
16 - DISK MOUNT
17 - POWER FAIL CHECK
```

IN ADDITION, EVENT TYPES 10-15 ARE ACCEPTED BY LOGPRT. (SEE

LISTING OF LOGPRT.)

3.2.3 CALLING LOGEV1 -- PRIMOS III

IN PMA:

```
CALL LOGEV1
DAC  MESSAGE
```

IN FORTRAN:

```
CALL LOGEV1(MESSAGE)
```

3.2.4 CALLING LOGEV1 -- PRIMOS IV

IN PMA, CODE INSIDE SEG4:

```
JSXB LOGEVL (NOTE DIFFERENT NAME)
IP   MESSAGE
```

IN PMA, CODE OUTSIDE SEG4:

```
CALL LOGEV1
AP   MESSAGE,SL
```

IN FORTRAN:

```
CALL LOGEV1(MESSAGE)
```

3.2.5 MODIFYING LOGPRT

CURRENTLY, LOGPRT RECOGNIZES AND FORMATS DATA FOR EVENT TYPES 0-9, 16, AND 17. TYPES 10-15 ARE ACCEPTED, BUT RESULT IN A PRINTOUT OF ONLY

TYPE=<TYPE> DATA=<WORD1> <WORD2> ... <WORD9>

(NOTE THAT ONLY 9 DATA WORDS ARE PRINTED FOR THESE TYPES.) TO ADD A NEW TYPE, ADD A LABEL TO THE COMPUTED GOTO FOLLOWING STATEMENT $400. AT THE NEW LABEL (BETWEEN $1950 AND $2000), CALL THE STORE ROUTINE TO PERFORM THE REQUIRED FORMATTING.

THE CALLING SEQUENCE FOR STORE IS AS FOLLOWS:

```
CALL STORE (TEXT,TXTLEN,ARRAY,NW,DEC)
```

TEXT A TEXT STRING TO BE PRINTED.

TXTLEN THE LENGTH IN CHARACTERS IN TEXT. IF ZERO, NO TEXT IS

PRINTED.

ARRAY — AN ARRAY OF WORDS TO BE TRANSLATED AND ENTERED IN THE OUTPUT LINE. ENTRY(1) IS THE FIRST DATA WORD OF THE EVENT MESSAGE. ENTTYP AND ENTLEN CONTAIN THE TYPE AND LENGTH OF THE ENTRY.

NW — THE NUMBER OF WORDS IN ARRAY. IF ZERO, NO WORDS ARE TRANSLATED.

DEC — OCTAL/DECIMAL FLAG. IF ZERO, TRANSLATION IS TO OCTAL WITH NO LEADING ZERO SUPPRESSION. IF NON-ZERO, TRANSLATION IS TO DECIMAL WITH LEADING ZEROES SUPPRESSED.

NOTE THAT THE TOTAL LENGTH OF THE TEXT TO BE STORED (=TXTLEN+NW*7) SHOULD NOT EXCEED 67 -- THE MAXIMUM LENGTH THAT CAN BE PRINTED ON A TTY WITH AN INDENT IN EFFECT. (ALL LINES
\ AFTER THE FIRST FOR AN ENTRY ARE INDENTED 5 SPACES.) IF THE
\ LENGTH OF TEXT IS TOO LONG, IT WILL BE TRUNCATED.

\ THE BREAK SUBROUTINE (NO ARGUMENTS) CAN BE USED TO START A NEW
\ LINE. INDENTING MUST BE PERFORMED EXPLICITLY AFTER BREAK IS
\ CALLED (BY STARTING THE NEXT TEXT STRING WITH 4 BLANKS).

AFTER FORMATTING THE ENTRY, GOTO 2000. CODE AT THAT LABEL FINISHES THE FORMATTING AND OBTAINS THE NEXT ENTRY FROM LOGREC.

\ TO REBUILD LOGPRT, RUN THE COMMAND FILE C_LOGPRT IN SYSTEM (OR
\ C_LLOGPRT IF A FULL LISTING IS DESIRED). THIS WILL CREATE A RUN
\ FILE CALLED *LOGPRT. *LOGPRT CAN THEN BE MOVED TO CMDNCO AND RENAMED TO LOGPRT.

3.3 CHANGING THE SIZE OF LOGREC

\ THE SIZE OF LOGREC (OVER WHICH THE 'EXCEEDING...' MESSAGE IS
\ PRINTED) IS DEFINED BY THE FIGCOM VARIABLE LRQUOT IN SEG14. THE
\ VALUE OF LRQUOT IS SET BY THE CONFIGURATION COMMAND LOGREC (SEE
\ SECTION 1.3).

3.4 CHANGING LOGPRT'S DEFAULT INPUT/OUTPUT FILENAMES

THE DEFAULT INPUT NAME -- '<0>CMDNC0>LOGREC' -- IS IN THE ARRAY INPNAM. THE SIZE OF INPNAM AND LENGTH OF THE NAME, INNAML, SHOULD BE SET TO THE NUMBER OF WORDS AND CHARACTERS IN INPNAM RESPECTIVELY. THE DEFAULT OUTPUT NAME (LOGLST) IS IN THE ARRAY OUTNAM.

TABLE OF CONTENTS

1 GENERAL INFORMATION....3
1.1 FIRST-LEVEL EVENT LOGGER -- LOGEV1....3
1.2 SECOND-LEVEL LOGGER -- LOGEV2....3
1.3 THE LOGREC CONFIGURATION COMMAND....4
1.4 LOGPRT -- DUMP CONTENTS OF LOGREC....4

2 LOGPRT PROCESSING....6

3 MODIFYING THE EVENT LOGGING MECHANISM....10
3.1 INCREASING THE SIZE OF LOGBUF....11
3.2 ADDING EVENT TYPES....11
3.3 CHANGING THE SIZE OF LOGREC....13
3.4 CHANGING LOGPRT'S DEFAULT INPUT/OUTPUT FILENAMES....13

DATE: JANUARY 18, 1979

REV 3

SUBJECT: MIDAS FOR REV 16

ABSTRACT

THIS DOCUMENT DESCRIBES A MAJOR PERFORMANCE IMPROVEMENT FOR REV. 16 MIDAS. A DISCUSSION OF SEVERAL ENHANCEMENTS AND BUG FIXES IS ALSO INCLUDED.

OVERVIEW

MIDAS APPLICATIONS ON P350, P400, AND P500 COMPUTERS WILL OPERATE SIGNIFICANTLY FASTER AT REV 16. IN ADDITION, THE PERFORMANCE IMPROVEMENT REQUIRES NO MODIFICATION OF EITHER USER PROGRAMS OR EXISTING MIDAS FILES.

THE PERFORMANCE IMPROVEMENT IS DUE TO A CHANGE IN THE WAY INDEX ENTRIES ARE ADDED TO A MIDAS FILE. AS ENTRIES ARE INSERTED, MIDAS DYNAMICALLY RESTRUCTURES THE APPROPRIATE INDICIES. PREVIOUS VERSIONS OF MIDAS INSERTED NEW INDEX ENTRIES INTO OVERFLOW CHAINS. AS THE CHAINS GREW LONGER, PERFORMANCE WAS DEGRADED. TO IMPROVE PERFORMANCE, USERS PERIODICALLY EXECUTED UTILITY PROGRAM REMAKE WHICH ELIMINATED THE OVERFLOW CHAINS. REV 16 MIDAS HAS ELIMINATED THE USE OF OVERFLOW CHAINS AND THEREFORE THE NECESSITY TO EXECUTE REMAKE. BECAUSE INDICIES ARE DYNAMICALLY MODIFIED, INSERTION AND DELETION OPERATIONS OPERATE FASTER THAN IN PREVIOUS MIDAS RELEASES.

AFTER INSTALLING REV 16, REMAKE MUST BE EXECUTED ONCE TO ELIMINATE ANY OVERFLOW CHAINS. THEREAFTER REMAKE MUST NOT BE USED. NOTE THAT REV 16 MIDAS WILL PRINT 'STOP! REMAKE THIS FILE!' AND ABORT IF ANY OVERFLOW ENTRIES ARE ENCOUNTERED.

FILES PROCESSED BY REV 16 MIDAS MAY NOT BE USED WITH REV 15 MIDAS. HOWEVER, A NEW UTILITY, *REVERT, WILL CONVERT SUCH A FILE TO A FORMAT WHICH IS COMPATIBLE WITH REV 15 MIDAS.

IN ADDITION TO GREATER SPEED, REV 16 MIDAS HAS IMPROVED FILE SPACE UTILIZATION. EARLIER VERSIONS OF MIDAS SIMPLY 'MARKED' DELETED INDEX ENTRIES. THE SPACE OCCUPIED BY THE DELETED ENTRIES WAS NOT REUSABLE UNTIL THE FILE HAD BEEN RESTRUCTURED BY THE REMAKE UTILITY. MIDAS NOW DYNAMICALLY RECOVERS AND MAY REUSE THE SPACE OCCUPIED BY DELETED INDEX ENTRIES.

MIDAS USER INTERFACE CHANGES

THE CALLING SEQUENCE OF EXISTING MIDAS FORTRAN USER INTERFACE SUBROUTINES HAS NOT BEEN CHANGED. HOWEVER, TWO NEW SUBROUTINES HAVE BEEN ADDED AND A NEW FUNCTION HAS BEEN ADDED TO NEXT$. IN ADDITION, THE OPERATION OF DELET$ HAS BEEN MODIFIED.

UMODE$

THE FIRST SUBROUTINE, UMODE$, ALLOWS AN APPLICATION PROGRAM TO INDICATE THAT IT WILL BE THE ONLY PROGRAM ACCESSING A PARTICULAR MIDAS FILE. THE MIDAS FILE HANDLER NORMALLY ASSUMES THAT A MIDAS FILE MAY BE ACCESSED BY SEVERAL APPLICATIONS SIMULTANEOUSLY. AS A RESULT, INDEX AND DATA SEGMENT SUBFILES ARE OPENED AND CLOSED WITH EACH CALL TO MIDAS. IF ONLY ONE APPLICATION IS TO ACCESS A MIDAS FILE, THEN A CALL TO UMODE$ ALLOWS MIDAS TO AVOID UNNECESSARY OPENING AND CLOSING OF

SEGMENTS. PRELIMINARY STUDIES INDICATE THAT UMODE$ MAY PROVIDE ABOUT A 50% PERFORMANCE IMPROVEMENT. THUS, UMODE$ MAY BE ESPECIALLY USEFUL FOR FORTRAN, MQL, COBOL, BASIC, AND RPG BENCHMARKS.

CALLING SEQUENCE:

```
     CALL UMODE$(FILE_UNIT,FUNCTION,STATUS)
```

FILE_UNIT --> UNIT UPON WHICH THE MIDAS DIRECTORY IS OPEN.

FUNCTION --> 1 (MM$SUM) INDICATES THAT THE FILE IS TO BE ACCESSED ONLY BY THE CALLING PROGRAM. MIDAS WILL THEN LEAVE SEGMENTS OPEN BETWEEN CALLS.

--> 0 (MM$MUM) INDICATES THAT THE FILE WILL BE ACCESSED BY POSSIBLY MORE THAN ONE PROGRAM. MIDAS WILL CLOSE SEGEMENTS BETWEEN CALLS. THIS IS THE DEFAULT MODE.

STATUS --> 0 NO ERROR.

--> 1 TOO MANY FILES. PARAMTER MFILES IN KPARAM DEFINES THE NUMBER OF FILES WHICH MAY BE IN SINGLE USERS MODE AT ONE TIME.

--> 2 ILLEGAL FUNCTION.

--> 3 ILLEGAL FILE UNIT.

NOTE THAT IT IS THE RESPONSIBILITY OF THE USER TO CONTROL THE NUMBER OF PROGRAMS ACCESSING A MIDAS FILE. MIDAS HAS NO WAY OF KNOWING HOW MANY PROGRAMS ARE ACTUALLY USING A MIDAS FILE.

BEFORE AN APPLICATION PROGRAM CLOSES A MIDAS FILE, THE FILE MUST BE RETURNED TO THE MULTIUSER MODE VIA A CALL TO UMODE$. THIS IS NECESSARY TO ALLOW MIDAS TO CLOSE SEGMENTS WITHIN THE MIDAS FILE.

EXAMPLE

```
      CALL UMODE$(FILUN,1,STATUS)
C     SETS THE FILE OPEN ON FILUN TO SINGLE USER MODE.
C
      CALL UMODE$(FILUN,0,STATUS)
C     INDICATES TO MIDAS TO CLOSE THE SEGMENTS FOR THE
C     MIDAS DIRECTORY OPEN ON FILUN. MIDAS WILL
C     SUBSEQUENTLY CLOSE SEGMENTS AFTER EACH CALL TO
C     MIDAS.
C
      CALL UMODE$(0,0,STATUS)
C     MIDAS WILL CLOSE ALL OF THE SEGMENTS THAT IT HAS
C     OPENED. ALL FILES ARE RETURNED TO MULTI-USER MODE.
```

GDATA$

WITH THE SECOND NEW USER INTERFACE SUBROUTINE, USERS CAN SERIALLY ACCESS DATA RECORDS IN A MIDAS FILE.

CALLING SEQUENCE:

```
CALL GDATA$(FILE_UNIT,FUNCTION,BUFFER,SIZE,STATUS)

FILE UNIT  --> MIDAS DIRECTORY FILE UNIT.

FUNCTION   --> :200 (FL$FST) RETRIEVE THE FIRST DATA RECORD.
           --> :100 (FL$NXT) RETRIEVE THE NEXT RECORD.

BUFFER     --> USER DATA BUFFER.

SIZE       --> SIZE IN BYTES OF THE BUFFER.

STATUS     -->  0 NO ERROR.
           --> >0 SYSTEM ERROR CODE.
           --> -1 ILLEGAL FUNCTION CODE
           --> -2 BAD INDEX DESCRIPTOR. (MIDAS FILE)
           --> -3 INVALID RECORD POSITION.

UPON RETURN FROM GDATA$, THE BUFFER CONTAINS THE DATA RECORD. NOTE
THAT CALLS TO GDATA$ SHOULD NOT BE MIXED WITH CALLS TO OTHER MIDAS DATA
ACCESS ROUTINES SUCH AS FIND$.
```

NEXT$ FUNCTIONALITY

NEXT$ MAY NOW BE USED TO RETRIEVE THE RECORD CORRESPONDING TO THE INDEX ENTRY PRECEDING THE CURRENT ENTRY. THIS FUNCTION IS DETERMINED BY BIT 11 IN THE FLAGS PARAMETER. (FLAGS = :40) A TYPICAL CALL TO NEXT$ WOULD USE A FLAGS VALUE OF :140040. THAT IS, BIT 1 --> USE ARRAY, BIT 2 --> RETURN ARRAY, BIT 11 --> RETRIEVE PRECEDING RECORD.

\ADD1$ MODIFICATION

\WHEN INSERTING A SECONDARY INDEX ENTRY VIA A CALL TO ADD1$, MIDAS WILL \NOT MODIFY THE CONTROL ARRAY PARAMETER SUPPLIED BY THE USER EVEN IF THE \FL$RET BIT IS SET IN THE FLAGS PARAMETER.

DELET$ MODIFICATION

SINCE DELET$ CAUSES A DELETED INDEX ENTRY TO BE REMOVED FROM THE INDEX, THE CURRENT POSITION IN THE INDEX IS MODIFIED BY THE DELET$ OPERATION. IN APPLICATIONS IN WHICH BIT 1 OF THE FLAGS PARAMETER (USE ARRAY BIT) IS SET, USERS MUST ALSO SET BIT 2 (RETURN ARRAY) IN ORDER TO MAINTAIN A VALID POSITION IN THE INDEX. THIS CASE OCCURS FREQUENTLY WHEN USING DELET$ IN CONJUNCTION WITH NEXT$ TO DELETE RECORDS WHILE SEQUENTIALLY TRAVERSING AN INDEX.

FILE_UNIT_HANDLING

SEVERAL CHANGES HAVE BEEN MADE TO THE MANNER IN WHICH MIDAS HANDLES FILE UNITS. INTERNALLY, MIDAS USES FILE UNITS TO ACCESS INDEX AND DATA SEGMENTS OF A MIDAS FILE. WHEN ASSIGNING FILE UNITS, MIDAS SEARCHES FROM UNIT 63 DOWNWARD UNTIL AN AVAILABLE UNIT IS FOUND. THIS FILE UNIT VALUE IS THEN INSERTED INTO TABLE SEG.

PARAMETER STSIZ ,DEFINED IN FILES LONGPL, LDPOOL, AND MIDPOL, DETERMINES THE SIZE OF THE TABLE AND THEREFORE THE NUMBER OF FILE UNITS WHICH MIDAS MAY UTILIZE SIMULTANEOUSLY. IF THE TABLE IS FULL AND ANOTHER UNIT IS NEEDED, MIDAS CLOSES THE LEAST RECENTLY OPENED FILE UNIT AND OPENS THE NEW SEGMENT ON THAT FILE UNIT.

MIDAS_UTILITY_PROGRAM_CHANGES

KBUILD -- SEVERAL BUGS HAVE BEEN FIXED. SEE THE SECTION DESCRIBING T.A.R.'S.

REPAIR -- THE REPAIR UTILITY HAS BEEN ELIMINATED.

REMAKE -- REMAKE MAY ONLY BE USED WITH FILES EARLIER THAN REV 16. NOTE THAT REMAKE MUST BE USED ON EACH REV 15 (OR EARLIER) FILE BEFORE THE FILE MAY BE USED WITH REV 16 MIDAS.

CREATK -- SEVERAL BUGS HAVE BEEN FIXED. IN ADDITION, CREATK NOW OPENS AND CLOSES ONLY THOSE FILE UNITS THAT IT ACTUALLY NEEDS. AS A RESULT, CREATK CAN BE RUN FROM A COMINPUT FILE.

*REVERT -- *REVERT IS NEW AT REV 16. THIS UTILITY ALLOWS USERS TO CONVERT FILES WHICH HAVE BEEN PROCESSED BY REV 16 MIDAS TO A STRUCTURE WHICH IS COMPATIBLE WITH EARLIER RELEASES. *REVERT IS CREATED BY C_RVRT IN UFD MIDAS. NOTE THAT *REVERT SETS THE FILE REV STAMP TO 15.0.

T.A.R._S_PROCESSED

T.A.R. NUMBER	PRODUCT
10941	KBUILD. BUILDING FROM A BINARY FILE.
13105	KBUILD. BINARY OPTION.
13128	KBUILD.
20458	KBUILD.
12815	MIDAS LIBRARY. FILE UNIT CONFLICTS

ELIMINATED FOR P400 USERS.

12816 CREATK. USE OF FORTRAN I/O WAS NOT ELIMINATED.

15431 CREATK. KEY TYPE OPTION 'S' WORKS. 'R' IS INVALID.

12636 REMAKE. THE PROBLEM WITH ATTEMPTS TO REMAKE AN EMPTY INDEX HAS BEEN ELIMINATED.

MODIFICATIONS HAVE BEEN MADE TO ELIMINATE SEVERAL PROBLEMS WHICH COULD ARISE IN MULTI-USER APPLICATIONS.

A) IF A 'DEADLOCK' OCCURS DURING A SECONDARY INDEX SEARCH, MIDAS WILL NOT DELETE ANY INDEX ENTRIES WHICH POINT TO DELETED DATA RECORDS.

B) IF A 'DEADLOCK' OCCURS DURING A DELETE OPERATION, MIDAS WILL RESTART DELET$.

C) IF A 'DEADLOCK' OCCURS DURING AN INSERT (ADD1$) OPERATION, MIDAS WILL NOT RELEASE ITS FILE UNITS. IN ADDITION, THE SEGMENT CONTAINING THE ROOT INDEX BLOCK IS NEVER CLOSED UNTIL THE INSERT HAS COMPLETED. NOTE THAT TWO PROCESSES, DOING INSERTS, CAN NEVER DEADLOCK.

USERS SHOULD BE AWARE THAT IN MULTIUSER APPLICATIONS THE CURRENT POSITION OF A PROCESS IN A MIDAS FILE MAY BE MODIFIED BY ANOTHER PROCESS.

\NEW_ERROR_CODE

\WHEN A CALL TO MIDAS REQUESTS THAT THE CONTROL ARRAY PARAMETER BE USED, \MIDAS NOW CHECKS TO BE CERTAIN THAT THE INDEX ENTRY LOCATED BY MIDAS IS \ACTUALLY THE ENTRY SPECIFIED BY THE CONTROL ARRAY PARAMETER. IF THE \ENTRIES ARE NOT THE SAME, THEN MIDAS RETURNS AN ERROR CODE OF 13. \"MIDAS ERROR 13" IS ALSO PRINTED AT THE TERMINAL EXCEPT IN BASICV. \NOTE THAT THIS CHECKING DOES NOT OCCUR DURING AN INSERTION OPERATION. \(IE. CALLS TO ADD1$, COBOL WRITE'S, BASICV ADD'S)

\THE CONTROL ARRAY PARAMETER IS ESSENTIALLY A "CURRENT ENTRY" \DESCRIPTOR. ERROR 13 MAY OCCUR IF THE "CURRENT INDEX ENTRY" HAS BEEN \CHANGED SINCE THE POSITION WAS ESTABLISHED. IN A TWO USER APPLICATION \FOR EXAMPLE, AN ERROR 13 MAY OCCUR FOR PROCESS A IF PROCESS B INSERTS \OR DELETES AN INDEX ENTRY IN THE INDEX BLOCK IN WHICH PROCESS A HAS A \CURRENT ENTRY.

\ERROR CODE 13 MAY ALSO BE GENERATED BY A SINGLE PROCESS IF THE PROCESS \INSERTS OR DELETES AN INDEX ENTRY IN THE SAME BLOCK IN WHICH IT HAS A \CURRENT ENTRY. A PROCESS WHICH ENCOUNTERS AN ERROR 13 MAY ATTEMPT TO \RE-ESTABLISH THE "CURRENT POSITION" BY DOING A KEYED ACCESS.

TAILORING MIDAS

THE SIZE OF THE INTERNAL BUFFER POOL IS DETERMINED BY TWO PARAMETERS. TO MODIFY THE DEFAULT SIZE, SET THE PARAMETER CTLASZ TO THE NUMBER OF INDEX BLOCKS TO BE IN THE BUFFER POOL. THE MINIMUM VALUE IS 2 AND THE DEFAULT IS 6. PARAMETER RECLNT DETERMINES THE SIZE OF A BUFFER. THE DEFAULT VALUE IS 1024 WORDS.

INSTALLATION NOTES

1. MIDAS IS SUPPLIED WITH THE 64V MODE LIBRARY, VKDALB, INSTALLED AS A SHARED LIBRARY. COMMAND FILE C_SKLB BUILDS THE SHARED LIBRARY.

2. EACH MIDAS FILE MUST BE RESTRUCTURED WITH THE MIDAS UTILITY PROGRAM REMAKE BEFORE THE NEW LIBRARY CAN BE USED.

SUBJECT: RUNOFF FOR RELEASE 16.0.

TWO NEW COMMANDS ARE AVAILABLE: .EODD (EJECT ODD) AND .EEVEN (EJECT EVEN), MINIMAL ABBREVIATIONS ARE .EO AND .EE. THESE COMMANDS CAUSE AN EJECT TO A NEW PAGE. A SUBSEQUENT EJECT IS THEN CAUSED IF THE NUMBER OF THE NEW PAGE IS EVEN (FOR .EO) OR ODD (FOR .EE). THESE COMMANDS FUNCTION INDEPENDENTLY OF WHETHER THE PAGE NUMBER HAS BEEN SET WITH THE .PAGE COMMAND OR IS BEING DISPLAYED.

SUBJECT: AVAIL - REV. 16

THE AVAIL COMMAND, USED TO GENERATE THE CURRENT RECORD AVAILABILITY STATUS OF STARTED-UP DISKS, HAS BEEN MODIFIED SO AS TO ACCEPT VOLUMENAMES UP TO 32 CHARACTERS IN LENGTH. THE COMMAND WILL YET CONTINUE TO SUPPORT OLD STYLE PARTITIONS AND THEIR 6 CHARACTER NAMES.

IN ADDITION, THE AVAIL COMMAND WILL NOW ACCEPT AS AN ARGUMENT THE LOGICAL DEVICE NUMBER, STATED IN DECIMAL DIGIT NOTATION, FOR THE PARTITION DESIRED. THE FORMAT FOR WHICH, MUST BE ENTERED AS '-LDEV DD', WHERE DD REPRESETS THE LDEV. CURRENTLY, 18 PARTITIONS ARE SUPPORTED BY THE SYSTEM. WHEN A LDEV IS GIVEN WHICH DOES NOT SUPPORT A PARTITION, ONE OF TWO ERROR MESSAGES WILL BE FORTHCOMING,

'DISKRAT NOT FOUND FOR LDEV'
OR
'SYSTEM SUPPORTS LDEV 0:17

SUBJECT: APPLIB (VAPPLB) - APPLICATIONS LIBRARY

ATTACHED IS A DETAILED DESCRIPTION OF THE REV.16 APPLICATIONS LIBARAY (APPLIB) AND ITS V-MODE VERSION (VAPPLB) LOCATED IN UFD = LIB. FOR FURTHER INFORMATION CONTACT ERIC STANMYER (SUBSYSTEMS).

ALL SUBMISSIONS, COMMENTS, CRITICISMS AND SUGGESTIONS ARE APPRECIATED.

THE REV.16 APPLICATIONS LIBRARY CONTAINS SEVERAL NEW SECTIONS. SUBSECTION 2.4, TITLED "SERVICE ROUTINES" HAS BEEN RESTUCTURED INTO THE FOLLOWING SUBSECTIONS:

2.4 STRING MANIPULATION ROUTINES
2.5 USER QUERY ROUTINES
2.6 SYSTEM INFORMATION ROUTINES
2.7 CONVERSION ROUTINES
2.8 MATHEMATICAL ROUTINES
2.9 PARSING ROUTINES

SEVERAL NEW ROUTINES HAVE ALSO BEEN ADDED FOR REV.16, THESE ARE:

FILE SYSTEM:

1. TSCN$A(KEY,UNITS,ENTRY,MAXSIZ,ENTSIZ,MAXLEV,LEV,CODE)

TSCN$A IS A LOGICAL FUNCTION THAT SCANS THE FILE SYSTEM TREE STRUCTURE (STARTING WITH THE HOME UFD) USING RDEN$$ AND SGDR$$ TO READ UFD AND SEGMENT DIRECTORY ENTRIES INTO THE ENTRY ARRAY. EACH CALL TO TSCN$A RETURNS THE NEXT FILE ON THE CURRENT LEVEL OR THE FIRST FILE ON THE NEXT LOWER LEVEL OF THE STRUCTURE. THE VARIABLE LEV IS USED TO KEEP TRACK OF THE CURRENT LEVEL. FOR EXAMPLE, AFTER THE FIRST CALL TO TSCN$A (WITH LEV=0), LEV WILL BE RETURNED AS 1, AND ENTRY(1,1) WILL CONTAIN THE UFD ENTRY DESCRIBING THE FIRST FILE IN THE HOME UFD. IF THIS FILE IS A SUBUFD, FOLLOWING THE NEXT CALL TO TSCN$A, LEV WILL BE 2, AND ENTRY(1,2) WILL CONTAIN THE ENTRY FOR THE FIRST FILE IN THE SUBUFD.

STRING MANIPULATION:

1. TYPE$A(KEY,STRING,LENGTH)

TYPE$A IS A LOGICAL FUNCTION THAT WILL TEST A CHARACTER STRING TO DETERMINE IF IT CAN BE INTERPRETED AS THE TYPE SPECIFIED BY KEY. A STRING IS NAME IF IT CONTAINS AT LEAST ONE ALPHABETIC OR SPECIAL CHARACTER (OTHER THAN A LEADING + OR -), A DECIMAL NUMBER IF IT CONTAINS ONLY THE DIGITS 0 - 9, AN OCTAL NUMBER IF IT CONTAINS ONLY THE DIGITS 0 - 7, AND A HEXADECIMAL NUMBER IF IT CONTAINS ONLY THE DIGITS 0 - 9 AND THE CHARACTERS A - F (UPPER CASE ONLY). A NUMBER MAY HAVE A LEADING SIGN AND ANY NUMBER OF BLANKS BETWEEN THE SIGN AND THE FIRST DIGIT, HOWEVER IMBEDDED BLANKS WITHIN THE NUMBER ITSELF ARE NOT ALLOWED. A NUMBER MUST ALSO HAVE AT LEAST ONE DIGIT.

LEADING AND TRAILING BLANKS ARE IGNORED. THE FUNCTION IS TRUE IF STRING SATISFIES THE CONDITIONS REQUIRED BY THE KEY USED, OTHERWISE IT IS FALSE. A NULL STRING (IE. LENGTH EQUAL TO ZERO) WILL ONLY RETURN A FUNCTION VALUE OF TRUE IF KEY IS A$NAME.

TYPE$A AVOIDS THE PROBLEM OF DECIMAL OVERFLOW THAT CNVA$A HAS WHEN IT IS USED TO DETERMINE IF A STRING IS A DECIMAL NUMBER (CNVA$A IS FALSE IF DECIMAL OVERFLOW OCCURS).

2. MSTR$A(A,ALEN,B,BLEN)

MSTR$A IS AN INTEGER FUNCTION THAT WILL MOVE THE SOURCE STRING, A, TO THE DESTINATION STRING, B. IF THE SOURCE STRING IS LONGER THAN THE DESTINATION STRING IT WILL BE TRUNCATED AND IF IT IS SHORTER IT WILL BE PADDED WITH BLANKS. THE SOURCE AND DESTINATION STRINGS MAY OVERLAP. THE FUNCTION VALUE WILL BE EQUAL TO THE NUMBER OF CHARACTERS MOVED (EXCLUDING BLANK PADDING). IF EITHER STRING IS NULL (IE. LENGTH EQUAL TO ZERO) NO CHARACTERS ARE MOVED AND THE FUNCTION WILL BE EQUAL TO ZERO.

3. MSUB$A(A,ALEN,AFC,ALC,B,BLEN,BFC,BLC)

MSUB$A IS AN INTEGER FUNCTION THAT WILL MOVE THE SOURCE SUBSTRING CONTAINED IN A TO THE DESTINATION SUBSTRING CONTAINED IN B. IF THE SOURCE SUBSTRING IS LONGER THAN THE DESTINATION SUBSTRING IT WILL BE TRUNCATED AND IF IT IS SHORTER IT WILL BE PADDED WITH BLANKS. THE SOURCE AND DESTINATION SUBSTRINGS MAY OVERLAP.

IF EITHER SUBSTRING IS NULL (IE. LENGTH EQUAL TO ZERO) NO CHARACTERS ARE MOVED AND THE FUNCTION WILL BE EQUAL TO ZERO, OTHERWISE IT IS EQUAL TO THE NUMBER OF CHARACTERS MOVED (EXCLUDING BLANKS USED FOR PADDING).

4. CSTR$A(A,ALEN,B,BLEN)

CSTR$A IS A LOGICAL FUNCTION THAT WILL COMPARE TWO STRINGS FOR EQUALITY. THE FUNCTION WILL BE TRUE IF EACH CHARACTER IN STRING A MATCHES THE CORRESPONDING CHARACTER IN STRING B, OR IF BOTH STRINGS ARE NULL (IE. LENGTH EQUAL TO ZERO), OTHERWISE THE FUNCTION WILL BE FALSE. COMPARISION IS ONLY MADE ON THE NUMBER OF CHARACTERS EQUAL TO THE OPERATIONAL LENGTH OF EACH STRING (IE. TRAILING BLANKS ARE IGNORED).

CSTR$A AVOIDS THE RESTRICTIONS PLACED ON NAMEQ$ CONCERNING NUMERIC FIELDS AND TRAILING BLANKS.

5. CSUB$A(A,ALEN,AFC,ALC,B,BLEN,BFC,BLC)

CSUB$A IS A LOGICAL FUNCTION THAT WILL COMPARE TWO SUBSTRINGS FOR EQUALITY. IF EACH CHARACTER IN THE A SUBSTRING MATCHES THE CORRESPONDING CHARACTER IN THE B SUBSTRING, OR BOTH SUBSTRINGS ARE NULL (IE. LENGTH EQUAL TO ZERO) THE FUNCTION WILL BE TRUE. IF TWO CORRESPONDING CHARACTERS DO NOT MATCH, OR IF THE LENGTHS OF THE SUBSTRINGS ARE NOT EQUAL THE FUNCTION WILL BE FALSE.

6. LSTR$A(A,ALEN,B,BLEN,FCP,LCP)

LSTR$A IS A LOGICAL FUNCTION THAT WILL SEARCH STRING B FOR THE FIRST OCCURENCE OF STRING A. IF STRING A IS FOUND THE FUNCTION WILL BE TRUE AND FCP AND LCP WILL BE EQUAL TO THE CHARACTER POSITIONS OF THE SUBSTRING IN B THAT MATCHES STRING A. IF STRING A IS NOT FOUND OR IF EITHER STRING IS NULL (IE. LENGTH EQUAL TO ZERO) THE FUNCTION WILL BE FALSE AND FCP AND LCP WILL BE EQUAL TO ZERO.

EACH STRING IS LOGICALLY TRUNCATED TO ITS OPERATIONAL LENGTH BEFORE THE SEARCH IS PERFORMED.

7. LSUB$A(A,ALEN,AFC,ALC,B,BLEN,BFC,BLC,FCP,LCP)

LSUB$A IS A LOGICAL FUNCTION THAT WILL SEARCH THE SUBSTRING CONTAINED IN B FOR THE FIRST OCCURENCE OF THE SUBSTRING CONTAINED IN A. IF A MATCH IS FOUND FCP AND LCP WILL BE EQUAL TO THE CHARACTER POSITIONS IN B OF THE MATCHING SUBSTRING AND THE FUNCTION WILL BE TRUE. IF A MATCHING SUBSTRING CANNOT BE FOUND OR IF EITHER SUBSTRING IS NULL (IE. LENGTH EQUAL TO ZERO) THE FUNCTION WILL BE FALSE AND FCP AND LCP WILL BE EQUAL TO ZERO.

8. JSTP$A(KEY,STRING,LENGTH)

JSTR$A IS A LOGICAL FUNCTION THAT WILL LEFT OR RIGHT JUSTIFIY A STRING WITHIN ITSELF. THE FUNCTION WILL BE TRUE IF JUSTIFICATION IS SUCCESSFUL, FALSE IF THE STRING LENGTH IS LESS THAN ZERO OR IF A BAD KEY IS SPECIFIED.

CONVERSION:

1. CNVB$A(NUMKEY,VALUE,NAME,NAMLEN)

CNVB$A IS AN INTEGER FUNCTION USED TO CONVERT AN INTEGER*4 BINARY NUMBER INTO AN ASCII DIGIT STRING FOR DECIMAL, OCTAL, OR HEXADECIMAL NUMBERS. THE RETURNED DIGIT STRING WILL BE RIGHT JUSTIFIED IN NAME PRECEEDED BY LEADING BLANKS OR ZEROS (DEPENDING ON KEY). IF VALUE IS NEGATIVE AND TO BE TREATED AS SIGNED DECIMAL, NAME WILL BEGIN WITH AN INITIAL '-' SIGN. IF THE NUMBER OF DIGITS CONVERTS SUCCESSFULLY, THE FUNCTION VALUE IS THE NUMBER OF DIGITS IN NAME, IF NOT THE FUNCTION VALUE IS ZERO AND NAME WILL BE BLANK.

PARSING:

1. CMDL$A(KEY,KWLIST,KWINDX,OPTBUF,BUFLEN,OPTION,VALUE,KWINFO)

CMDL$A IS A LOGICAL FUNCTION FOR PARSING PRIMOS TYPE COMMAND LINES (IE. A LINE COMPOSED OF -KEYWORDS OPTIONALLY FOLLOWED BY A SINGLE ARGUMENT). EACH CALL TO THE ROUTINE RETURNS INFORMATION ABOUT THE NEXT -KEYWORD (AND ITS ARGUMENT, IF ONE IS PRESENT) ON THE COMMAND LINE.

THE USER DEFINES AN ARRAY OF -KEYWORDS AND AND DESCRIBES THE TYPE OF ARGUMENT THAT MAY FOLLOW EACH KEYWORD. AN OPTIONAL LIST OF DEFAULT KEYWORDS MAY ALSO BE DEFINED. KEYWORD SYNONYMS ARE ALSO PROVIDED FOR AND ABBREVIATIONS ARE HANDLED USING A MINIMUM NUMBER OF CHARACTERS TO MATCH SCHEME.

CMDL$A RETURNS THE FOLLOWING INFORMATION FOR EACH -KEYWORD [ARGUMENT] ENTRY IN THE COMMAND LINE:

1) INTEGER THAT IDENTIFIES THE -KEYWORD (KWINDX),
2) TEXT OF THE KEYWORD ARGUMENT (OPTBUF),
3) ARGUMENT TYPE (OPTION),
4) RESULTS OF NUMERIC CONVERSION (VALUE),
5) NUMBER OF CHARACTERS IN OPTBUF (KWINFO(1)).

CMDL$A DOES NOT PERFORM ANY ACTION OTHER THAN RETURNING INFORMATION ABOUT THE COMMAND LINE.

TABLE OF CONTENTS

1 INTRODUCTION....6

2 GENERAL DESCRIPTION....6
2.1 NAMING CONVENTIONS....7
2.2 SYSCOM>A$KEYS....7
2.3 FILE SYSTEM ROUTINES....7
2.4 STRING MANIPULATION ROUTINES....9
2.5 USER QUERY ROUTINES....10
2.6 SYSTEM INFORMATION ROUTINES....10
2.7 CONVERSION ROUTINES....10
2.8 MATHEMATICAL ROUTINES....10
2.9 PARSING ROUTINES....11

3 LIBRARY IMPLEMENTATION AND POLICIES....12
3.1 SOURCE LANGUAGE....12
3.2 LIBRARY BUILDING....12
3.3 LIBRARY SUBMISSIONS....13

4 THE ROUTINES....15
4.1 FILE SYSTEM....15
4.2 STRING MANIPULATION....26
4.3 USER QUERY....37
4.4 SYSTEM INFORMATION....39
4.5 MATHEMATICAL....42
4.6 CONVERSION....44
4.7 PARSING....47

5 SUMMARY AND KEYS....55
5.1 SUMMARY....55
5.2 SYSCOM>A$KEYS....57

1 INTRODUCTION

APPLIB IS A NEW USER ORIENTED LIBRARY WHICH IS INTENDED TO FILL AN EVER WIDENING GAP IN PRIME SOFTWARE. AT PRESENT, APPLICATIONS AND SYSTEMS PROGRAMMERS MUST CREATE USER INTERFACES FOR THEIR PROGRAMS BASED UPON EITHER HIGH LEVEL LANGUAGE ROUTINES OR LOW LEVEL SYSTEM ROUTINES. IN MOST CASES, THE HIGH LEVEL CONSTRUCTS ARE EITHER INADEQUATE OR PRESENT A "ROUGH" APPEARANCE TO THE TERMINAL USER, AND THE LOW LEVEL ROUTINES, THOUGH ALLOWING ALMOST ANYTHING TO BE DONE, ARE TYPICALLY DIFFICULT TO USE. A SUBSEQUENT PROBLEM ARISES WHEN EVERYONE WRITES THEIR OWN INTERFACES: NO TWO PROGRAMS LOOK ALIKE TO THE USER. AS A RESULT, PRIME PRESENTS AN INCONSISTENT AND OFTEN SLOPPY FACE TO THE USER PUBLIC.

THE PRIMARY GOAL OF APPLIB IS TO PROVIDE USERS WITH AN EASY TO USE LIBRARY OF SERVICE ROUTINES WHICH FALLS BETWEEN THE VERY HIGH AND VERY LOW LEVEL ROUTINES. IN MANY CASES, THE ROUTINES DO LITTLE MORE THAN CALL A LOWER LEVEL ROUTINE, FILLING IN THE EXTRA ARGUMENTS THAT THE CALLER DOESN'T CARE ABOUT AND SOMETIMES REFORMATS WHAT THE LOW LEVEL ROUTINE RETURNS. IN OTHER CASES, THE APPLIB ROUTINES ARE FAIRLY COMPLEX, EITHER BECAUSE THEIR FUNCTIONALITY DEMANDS IT, OR BECAUSE CAREFUL CODING IS REQUIRED TO PERFORM A SEEMINGLY SIMPLE OPERATION CORRECTLY. THE SECONDARY BENEFITS OF THIS LIBRARY ARE THAT IT AVOIDS DUPLICATION OF EFFORT AND AUTOMATICALLY PROVIDES A CONSISTENT FACE TO THE TERMINAL USER.

THE APPLIB ROUTINES ARE NOT INCLUDED IN FTNLIB FOR TWO REASONS. FIRST, THEY DO NOT LOGICALLY BELONG THERE AS FTNLIB IS PRIMARILY FOR THE LOW LEVEL SYSTEM ROUTINES. SECOND, FTNLIB IS VERY LARGE AND REQUIRES A LONG TIME TO LOAD. THIS TIME IS ALREADY A SOURCE OF USER COMPLAINT. THEREFORE APPLIB, AND ITS V-MODE VERSION VAPPLB, EXIST AS INDEPENDENT LIBRARIES IN UFD=LIB ON THE SYSTEM.

2 GENERAL DESCRIPTION

ALL APPLIB ROUTINES ARE WRITTEN AS FORTRAN FUNCTIONS WHOSE VALUES ARE EITHER A STATUS INDICATION (.TRUE. OR .FALSE.), AN APPROPRIATE VALUE, OR AN ALTERNATE VALUE OR FORMAT OF A RETURNED ARGUMENT. IN ADDITION, THE CALLER IS NEVER RETURNED A "CODE" TYPE ARGUMENT WHICH MUST THEN BE DECODED. ALL ERROR DETECTION, REPORTING, AND, IF POSSIBLE, RECOVERY ARE PERFORMED IN THE ROUTINE, RETURNING ONLY THE INFORMATION OF SUCCESS OR FAILURE. ALTHOUGH THIS SEEMS LIMITING, AND IN A SENSE IT IS, MOST USERS DON'T WANT TO KNOW THE DETAILS AS LONG AS THE ERROR IS REPORTED AND ALL POSSIBLE RECOVERY PROCEDURES HAVE BEEN TRIED. IN MOST CASES, THE EXACT REASON FOR FAILURE COMES UNDER THE HEADING OF "IRRELAVENT DIFFERENCE" AND IS IGNORED ANYWAY.

2.1 NAMING CONVENTIONS

AS MENTIONED ABOVE, APPLIB ROUTINES ARE DESIGNED TO BE SIMPLE TO USE. IN ADDITION, THEY ARE ALSO INTENDED TO BE RELATIVELY INDEPENDENT OF SYSTEM REVISIONS. TO FACILITATE THESE GOALS, ALL APPLIB ROUTINES FOLLOW A CONSISTENT NAMING CONVENTION DESIGNED TO AVOID THE POSSIBILITY OF CONFLICT BOTH WITH USER WRITTEN ROUTINES AND SYSTEM ROUTINES. ALL APPLIB ROUTINES HAVE A FOUR LETTER MNEMONIC NAME AND THE SUFFIX "$A". THUS, FOR EXAMPLE, THE ROUTINE TO OPEN A TEMPORARY FILE IS NAMED "TEMP$A". ALSO, IN MANY CASES ROUTINES HAVE OPTIONS WHICH ARE SPECIFIED BY NAMED "PARAMETER" KEYS WHICH ALL BEGIN WITH THE PREFIX "A$".

SUBROUTINES THAT ARE USED INTERNALLY BY APPLIB ROUTINES HAVE A SUFFIX OF "$$A" AND SHOULD NOT BE USED UNDER ORDINARY CIRCUMSTANCES. NO DOCUMENTATION IS PROVIDED FOR THESE ROUTINES.

2.2 SYSCOM>A$KEYS

ALL "PARAMETER" KEYS ARE DEFINED IN A $INSERT FILE NAMED SYSCOM>A$KEYS. THE KEY NAMES, FOLLOWING THE "A$" PREFIX ARE THREE OR FOUR LETTER MNEMONICS SPECIFYING THE ALLOWABLE OPTIONS FOR THE VARIOUS ROUTINES. THE KEYS ARE ORGANIZED ACCORDING TO THE DESCRIPTIONS IN THIS DOCUMENT. IN ADDITION, THIS FILE SUPPLIES ALL THE APPROPRIATE FUNCTION TYPE DECLARATIONS FOR THE APPLIB ROUTINES. A COMPLETE LISTING OF SYSCOM>A$KEYS IS INCLUDED IN SECTION 5 AND THE DETAILED DESCRIPTIONS OF THE KEYS ARE LEFT FOR THE DESCRIPTIONS OF THE APPLICABLE ROUTINES.

2.3 FILE SYSTEM ROUTINES

THE FILE SYSTEM ROUTINES IN APPLIB GIVE THE USER A SIMPLE AND CONSISTENT WAY TO SPECIFY THE MOST COMMON FILE SYSTEM OPERATIONS. ACCORDINGLY, APPLIB DOES NOT PROVIDE THE USER WITH THE FULL CAPABILITIES OF THE FILE SYSTEM SINCE FOR MORE COMPLICATED OPERATIONS, THE FILE SYSTEM ROUTINES THEMSELVES ARE THE BEST ROUTINES TO CALL. APPLIB SUPPORTS BOTH SEQUENTIAL ACCESS METHOD (SAM) AND DIRECT ACCESS METHOD (DAM) FILES. THERE IS NO SUPPORT FOR SEGMENT DIRECTORY TYPE FILES AS THE MIDAS SUBSYSTEM PROVIDES THE HIGHER LEVEL FUNCTIONS WITH THESE FILES.

THE OPERATIONS PROVIDED IN APPLIB ARE:

1. OPEN - NOTE, THERF ARE SEVERAL POSSIBILITIES HERE
2. CLOSE
3. REWIND
4. GO TO END-OF-FILE
5. TRUNCATE
6. DELETE
7. CHECK FOR FILE EXISTENCE
8. CHECK FOR UNIT OPEN
9. READ CURRENT POSITION
10. SFT POSITION

ALL ROUTINES EXCEPT OPEN, DELETE AND EXISTENCE USE ONLY THE DOS FILE UNIT AND NOT THE FILE NAME. ALSO, EACH ROUTINE CARRIES THE NAME OF ITS FUNCTION, AS ABOVE, WITH ARGUMENTS CONSISTING OF ONLY THE RELAVENT INFORMATION, USUALLY JUST THE UNIT NUMBER. NOTE THAT ALL FILE NAMES, EXCEPT SCRATCH FILES, MAY BE TREE NAMES.

THE ONLY ROUTINES WHICH ARE AT ALL COMPLICATED ARE THE VARIOUS (5) OPEN ROUTINES DUE MOSTLY TO THE MULTITUDE OF WAYS IN WHICH PROGRAMS CAN ORTAIN THE NAME OF THE FILE THEY WISH TO OPEN AND THE VARIOUS POSSIBLE ACTIONS THEY MAY WANT TO TAKE BY WAY OF VERIFICATION OR ERROR RECOVERY. RATHER THAN PACK ALL POSSIBILITIES INTO A SINGLE CALLING SEQUENCE, THUS MAKING IT ALWAYS DIFFICULT TO USE AND TO REMEMBER, FIVE DIFFERENT ROUTINES EXIST TO PERFORM THE VARYING LEVELS OF COMPLEXITY. IN THIS WAY, THE SIMPLE OPERATIONS CARE RFPRESENTED BY SIMPLE CALLING SEQUENCES AND ONLY THE COMPLEX OPERATIONS NEFD TO SPECIFY COMPLEX ARGUMENT LISTS.

THF VARIOUS OPEN OPERATIONS ARE, BRIEFLY:

1. TEMP$A - OPEN A SCRATCH FILE WITH UNIQUE NAME
2. OPEN$A - OPEN SUPPLIED NAME
3. OPNP$A - READ NAME AND OPEN
4. OPNV$A - OPEN SUPPLIED NAME WITH VERIFICATION AND DELAY
5. OPVP$A - READ NAME AND OPEN WITH VERIFICATION AND DELAY

ALL ROUTINES ALLOW SELECTION OF THE FILE TYPE (SAM OR DAM) AND ALL BUT TEMP$A ALLOW SPECIFICATION OF THE OPEN MODE (READ, WRITE, OR READ/WRITE). SCRATCH FILES ARE ALWAYS OPENED FOR READ/WRITE.

VERIFICATION CONSISTS OF THE FOLLOWING OPTIONS:

1. VERIFY THAT THE FILE IS NEW; THAT IS, VERIFY THAT IT IS O.K. TO MODIFY A FILE WHICH ALREADY EXISTS.

2. SAME AS 1. ABOVE BUT IF THE FILE ALREADY EXISTS AND THE USER SAYS IT IS O.K. TO MODIFY IT, ASK WHETHER THE OLD FILE IS TO BE OVERWRITTEN OR APPENDED TO.

3. VERIFY THAT THE FILE IS OLD; THAT IS, DO NOT ALLOW CREATION OF A NEW FILE. NOTF THAT IF THE OPEN MODE IS READ, THIS IS THE ONLY POSSIBLE VERIFICATION OPTION.

DELAY CONSISTS OF THE FOLLOWING OPTIONS:

1. IF AND ONLY IF THE FILE IS "IN USE", WAIT A SUPPLIED NUMBER OF SECONDS (ELAPSED TIME) AND TRY AGAIN.

2. THE ABILITY TO RETRY 1. ABOVE A SPECIFIED NUMBER OF TIMES.

2.4 STRING MANIPULATION ROUTINES

THE STRING MANIPULATION ROUTINES ARE DESIGNED TO FACILITATE THE HANDLING OF CHARACTER STRINGS. UNLESS NOTED OTHERWISE IT WILL BE ASSUMED THAT ALL OF THESE ROUTINES OPERATE ON PACKED (2 CHARACTERS PER WORD) STRINGS AND THAT THE DATA TYPE OF THE STRING DOES NOT MATTER. MOST OF THE ROUTINES IN THIS SECTION CHECK THE VALIDITY OF STRING SUBSCRIPTS (CHARACTER POSITIONS) AND IF AN ERROR IS DETECTED WILL CAUSE A MESSAGE TO BE DISPLAYED.

THESE ROUTINES ARE:

FILL$A - FILL A STRING WITH A CHARACTER (E.G. FILL A NAME BUFFER WITH SPACES)

NLEN$A - DETERMINE THE OPERATIONAL LENGTH OF A STRING (NAME), NOT INCLUDING TRAILING BLANKS.

MCHR$A - MOVE A CHARACTER FROM ONE PACKED STRING TO ANOTHER.

GCHR$A - GET A CHARACTER FROM A PACKED STRING.

TREE$A - TEST FOR TREE NAME

TYPE$A - DETERMINE STRING TYPE

MSTR$A - MOVE ONE STRING TO ANOTHER

MSUB$A - MOVE ONE SUBSTRING TO ANOTHER

CSTR$A - COMPARE TWO STRINGS FOR EQUALITY

CSUB$A - COMPARE TWO SUBSTRINGS FOR EQUALITY

LSTR$A - LOCATE ONE STRING WITHIN ANOTHER

LSUB$A - LOCATE ONE SUBSTRING WITHIN ANOTHER

JSTR$A - JUSTIFY A STRING

2.5 USER QUERY ROUTINES

YSNO$A - ASK QUESTION AND OBTAIN A YES OR NO ANSWER

RNAM$A - PROMPT AND READ A NAME

RNUM$A - PROMPT AND READ A NUMBER (DECIMAL, OCTAL, OR HEXADECIMAL) INTO AN INTEGER*4 VARIABLE.

2.6 SYSTEM INFORMATION ROUTINES

TIME$A - TIME OF DAY

CTIM$A - CPU TIME SINCE LOGIN

DTIM$A - DISK TIME SINCE LOGIN

DATE$A - TODAY'S DATE, AMERICAN STYLE

EDAT$A - TODAY'S DATE, EUROPEAN (MILITARY) STYLE

DOFY$A - TODAY'S DATE AS DAY OF YEAR ("JULIAN" DATE)

2.7 CONVERSION ROUTINES

ENCD$A - ENCODE FUNCTION THAT ADJUSTS THE "FORMAT" TO MAKE THE NUMBER PRINTABLE IF POSSIBLE. IF NOT, THE FIELD IS FILLED WITH ASTERISKS.

CNVA$A - CONVERT ASCII NUMBER TO BINARY.

CNVB$A - CONVERT BINARY TO ASCII NUMBER.

2.8 MATHEMATICAL ROUTINES

RNDI$A - INITIALIZE RANDOM NUMBER GENERATOR "SEED".

RAND$A - GENERATE RANDOM NUMBER AND UPDATE "SEED". THIS GENERATOR IS BASED UPON A 32-BIT WORD SIZE AND USES THE LINEAR CONGRUENTIAL METHOD.

2.9 PARSING ROUTINES

CMDL$A - PARSE PRIMOS TYPE COMMAND LINE.

3 LIBRARY IMPLEMENTATION AND POLICIES

A STRONG EFFORT IS BEING MADE TO KEEP APPLIB BOTH CONSISTENT IN ITS USAGE AND EASY TO BUILD, EXPAND, AND MAINTAIN. TO THIS END, SEVERAL GUIDING PRINCIPLES HAVE BEEN FOLLOWED IN ITS IMPLEMENTATION AND A SET OF RULES ESTABLISHED TO CONTROL ITS FUTURE GROWTH.

3.1 SOURCE LANGUAGE

ALL ROUTINES IN APPLIB ARE WRITTEN IN FORTRAN TO FACILITATE THEIR INCLUSION IN BOTH APPLIB AND VAPPLB. IN GENERAL, ANY LANGUAGE WHICH CANNOT BE EITHER R-MODE OR V-MODE AS A COMPILER OPTION SHOULD BE AVOIDED AS THE PROLIFERATION OF MULTIPLE SOURCES OF THE SAME ROUTINE IS GUARENTEED, SOONER OR LATER, TO CAUSE THE TWO LIBRARIES TO FALL OUT OF SYNCHRONY. AS A MAJOR PREMISE OF APPLIB IS CONSISTENCY, INCOMPATIBLITIES BETWEEN THE R-MODE AND V-MODE LIBRARIES ARE UNACCEPTIBLE.

THE ROUTINES HAVE BEEN CODED IN SUCH A WAY AS TO MAKE THEM EASILY CALLABLE FROM MOST OTHER LANGUAGES, INCLUDING PLP AND 1976 ANSI FORTRAN, BOTH OF WHICH CAN AUTOMATICALLY GENERATE STRING LENGTH ARGUMENTS FOLLOWING STRING ARGUMENTS. AS A RESULT, IN THE ARGUMENT PAIR "NAME,NAMLEN", THE NAME IS OFTEN UPDATED BY AN APPLIB ROUTINE, BUT THE NAMLEN ARGUMENT IS NEVER TOUCHED. THE FUNCTION NLEN$A CAN BE USED TO DETERMINE THE OPERATIONAL LENGTH OF A RETURNED NAME.

ALL APPLIB ROUTINES WHICH EITHER ACCEPT KEYS AS ARGUMENTS OR CALL OTHER APPLIB ROUTINES WHICH DO, USE THE SYSCOM>A$KEYS FILE TO DEFINE THOSE KEYS. ALSO, THESE ROUTINES DO NOT TAKE ADVANTAGE OF ANY PARTICULAR NUMERICAL VALUES THESE KEYS MAY HAVE IN CASE IT BECOMES NECESSARY EITHER TO CHANGE THESE VALUES OR TO ADD NEW KEYS WITH NUMERICAL VALUES WHICH DO NOT FIT THE PREVIOUS PATTERN. FOR EXAMPLE, THERE ARE NO COMPUTED GOTO'S ON KEYS AND NO RANGE CHECKS FOR VALIDITY OF A KEY. IN THIS WAY, IF A NEW SYSCOM>A$KEYS FILE IS CREATED, BOTH THE USER PROGRAMS USING THEM AND THE ROUTINES THEY CALL WILL ALWAYS AGREE AS TO WHAT KEY MEANS WHAT. THE SAME IS TRUE OF THE DECLARED TYPES OF THE APPLIB FUNCTIONS.

3.2 LIBRARY BUILDING

ALL ROUTINES ARE COMPILED INTO A SINGLE BINARY FILE WHICH IS THEN CONVERTED INTO THE APPROPRIATE LIBRARY FILE WITH THE EDB UTILITY. AT PRESENT, THE ONLY DIFFERENCE BETWEEN THE R-MODE AND V-MODE BUILD PROCEDURES IS THE FTN COMPILE OPTION USED. FOR APPLIB, ALL ROUTINES ARE COMPILED FOR 64R MODE LOADING AND FOR VAPPLB, ALL ROUTINES ARE COMPILED FOR 64V MODE LOADING (SEG). IN ADDITION, ALL ROUTINES INCLUDED IN VAPPLB ARE PURE PROCEDURE AND MAY BE LOADED INTO THE SHARED PORTION OF A SHARED PROCEDURE.

SINCE SEVERAL OF THE APPLIB ROUTINES CALL OTHER APPLIB ROUTINES, THE LOAD ORDER IS IMPORTANT. THIS ORDER IS SPECIFIED IN THE COMMAND FILES "C_APPL" AND "C_VAPP" LOCATED IN UFD = APPLIB>SOURCE.

3.3 LIBRARY SUBMISSIONS

APPLIB IS BY NO MEANS COMPLETE OR STATIC AND SUBMISSIONS ARE WELCOME. HOWEVER, TO GUARANTEE THE GOALS OF APPLIB AS OUTLINED ABOVE, STRICT CONTROL WILL BE MAINTAINED OVER THE LIBRARY AND ALL SUBMISSIONS MUST CONFORM TO THE RULES SET OUT BELOW. THESE RULES, THOUGH STRICT, ARE NOT MEANT TO DISCOURAGE SUBMISSIONS, BUT TO PRESERVE THE INTEGRITY OF THE LIBRARY WHILE NOT REQUIRING AN EXCESIVE AMOUNT OF WORK ON THE PART OF THE LIBRARY ADMINISTRATOR.

IF SUBMISSIONS ARE MADE WHICH DO NOT CONFORM TO THE RULES, THEY WILL BE PLACED IN A "PENDING" FILE OR AN "IDEA" FILE, DEPENDING UPON THEIR RELATIVE STATES OF COMPLETION. NO GUARANTEE IS MADE THAT ANY SUCH SUBMISSIONS WILL BE INCORPORATED INTO THE LIBRARY.

THE SPIRIT OF APPLIB SHOULD BE KEPT IN MIND WHEN SUBMITTING A ROUTINE. FOR EXAMPLE, A ROUTINE TO PERFORM A MATHEMATICAL FUNCTION MAY BE VERY USEFUL AND DESIREABLE, BUT PROBABLY BELONGS IN MATHLB, NOT APPLIB. IN A SIMILAR WAY, A ROUTINE WHICH DOES TABLE BUILDING, LOOK-UP, OR SORTING PROBABLY BELONGS IN EITHER THE MSORTS OR SRTLIB LIBRARY.

THE LIST OF APPLIB "GROUND RULES" ARE:

1. THE ROUTINE MUST BE IN FORTRAN SUITABLE FOR BOTH APPLIB AND VAPPLB.

2. THE ROUTINE SOULD NOT HAVE "CODE" AS AN ARGUMENT - THE ROUTINE SHOULD HANDLE ALL ABNORMAL SITUATIONS.

3. IF REASONABLE, THE ROUTINE SHOULD BE A FUNCTION WHERE THE VALUE OF THE FUNCTION IS AN ALTERNATE FORM OF THE RETURNED ARGUMENT(S) OR A STATUS INDICATION (SEE #2).

4. THE ROUTINES SHOULD CONFORM TO THE FOLLOWING CONVENTIONS:

   A. ALL ROUTINE NAMES SHOULD END WITH "$A".

   B. ALL ROUTINES WHICH ACCEPT A KEY OR CALL OTHER APPLIB ROUTINES WHICH DO, SHOULD USE SYSCOM>A$KEYS. ANY NEW KEYS WILL BE ADDED TO SYSCOM>A$KEYS BY THE LIBRARY ADMINISTRATOR AND SHOULD BEGIN WITH THE PREFIX "A$". ALSO, NO USE SHOULD BE MADE OF ANY NUMERICAL RELATION BETWEEN KEYS.

   C. ALL FILE SYSTEM CALLS SHOULD BE TO "$$" ROUTINES WITH CODE RATHER THAN LOC(CODE) AS AN ARGUMENT.

   D. RDTK$$ SHOULD BE USED INSTEAD OF CMREAD. IF THE 80 CHARACTER LIMIT FOR RDTK$$ IS INSUFFICIENT, USE I$AA12.

   E. IF REASONABLE, DO NOT USE FORTRAN READ'S AND WRITE'S.

   F. THE USE OF "2-WAY" ARGUMENTS SHOULD BE AVOIDED IF POSSIBLE.

5. ALL ROUTINES SHOULD BE THOROUGHLY TESTED.

6. ALL SUBMISSIONS MUST BE ACCOMPANIED BY A LISTING WITH A STANDARD PRIME HEADER. ALSO, THE LISTING SHOULD CONTAIN A DESCRIPTION OF THE ARGUMENTS AS WELL AS ANY LIMITATIONS OR RESTRICTIONS EITHER ON THEIR USE OR ON THEIR LOADING.

7. ALL SUBMISSIONS MUST BE ACCOMPANIED BY A DOCUMENT DESCRIBING THEIR USE, ALL ARGUMENTS, AND ANY RESTRICTIONS OR LIMITATIONS ON THEIR USE. THIS DOCUMENT WILL BE INCLUDED IN THE LIBRARY DESCRIPTION.

8. ALL SUBMITTED ROUTINES ARE SUBJECT TO MODIFICATION FOR THE PURPOSE OF CONSISTENCY OR GENERALITY.

9. ALL SUBMISSIONS ARE SUBJECT TO REVIEW AND FINAL APPROVAL BY THE LIBRARY ADMINISTRATOR BEFORE THEY ARE INCORPORATED INTO APPLIB.

4 THE ROUTINES

BELOW ARE THE DETAILED DESCRIPTIONS OF EACH ROUTINE IN APPLIB, GROUPED BY FUNCTION.

4.1 FILE SYSTEM

TEMP$A

TEMP$A IS A LOGICAL FUNCTION WITH THE FOLLOWING CALLING SEQUENCE:

```
LOG= TEMP$A(TYPKEY,NAME,NAMLEN,UNIT)
CALL TEMP$A(TYPKEY,NAME,NAMLEN,UNIT)
```

```
WHERE TYPKEY = A$SAMF, SAM FILE (.NE. A$DAMF)
               A$DAMF, DAM FILE
      NAME   = RETURNED NAME (6 CHARACTERS)
      NAMLEN = LENGTH OF NAME BUFFER IN CHARACTERS (.GE. 6)
      UNIT   = DOS FILE UNIT
```

ALL ARGUMENTS ARE INTEGER*2 EXCEPT NAME WHICH DOESN'T MATTER.

THIS ROUTINE OPENS A UNIQUE TEMPORARY FILE IN THE CURRENT UFD FOR READING AND WRITING. THIS FILE WILL BE NAMED T$XXXX WHERE XXXX IS A 4 DIGIT DECIMAL NUMBER BETWEEN 0000 AND 9999 INCLUSIVE. THE ACTUAL NAME OPENED WILL BE RETURNED IN THE NAME BUFFER. IF THE OPERATION IS SUCCESSFUL, THE FUNCTION VALUE IS .TRUE. AND IF THE OPERATION IS UNSUCCESSFUL, THE FUNCTION VALUE IS .FALSE..

OPEN$A

OPEN$A IS A LOGICAL FUNCTION WITH THE FOLLOWING CALLING SEQUENCE:

```
LOG= OPEN$A(OPNKEY+TYPKEY,NAME,NAMLEN,UNIT)
CALL OPEN$A(OPNKEY+TYPKEY,NAME,NAMLEN,UNIT)

WHERE OPNKEY = A$READ, OPEN FOR READING (.NE. A$WRIT OR A$RDWR)
               A$WRIT, OPEN FOR WRITING
               A$RDWR, OPEN FOR READING AND WRITING
      TYPKEY = A$SAMF, SAM FILE (.NE. A$DAMF)
               A$DAMF, DAM FILE
      NAME   = FILE NAME (MAY BE A TREE NAME)
      NAMLEN = LENGTH OF NAME IN CHARACTERS
      UNIT   = DOS FILE UNIT
```

ALL ARGUMENTS ARE INTEGER*2 EXCEPT NAME WHICH DOESN'T MATTER.

THIS ROUTINE OPENS A FILE OF THE GIVEN NAME ON UNIT. IF THE OPERATION IS SUCCESSFUL, THE FUNCTION VALUE IS .TRUE. AND IF THE OPERATION IS UNSUCCESSFUL, THE FUNCTION VALUE IS .FALSE..

OPNP$A

OPNP$A IS A LOGICAL FUNCTION WITH THE FOLLOWING CALLING SEQUENCE:

```
LOG= OPNP$A(MSG,MSGLEN,OPNKEY+TYPKEY,NAME,NAMLEN,UNIT)
CALL OPNP$A(MSG,MSGLEN,OPNKEY+TYPKEY,NAME,NAMLEN,UNIT)

WHERE MSG    = PROMPT FOR NAME MESSAGE
      MSGLEN = LENGTH OF MSG IN CHARACTERS
      OPNKEY = A$READ, OPEN FOR READING (.NE. A$WRIT OR A$RDWR)
               A$WRIT, OPEN FOR WRITING
               A$RDWR, OPEN FOR READING AND WRITING
      TYPKEY = A$SAMF, SAM FILE (.NE. A$DAMF)
               A$DAMF, DAM FILE
      NAME   = FILE NAME (MAY BE A TREE NAME)
      NAMLEN = LENGTH OF NAME IN CHARACTERS
      UNIT   = DOS FILE UNIT
```

ALL ARGUMENTS ARE INTEGER*2 EXCEPT NAME AND MSG WHICH DON'T MATTER.

THIS ROUTINE GETS A NAME FROM THE USER AND OPENS IT ON UNIT. IF THE OPERATION IS SUCCESSFUL, THE FUNCTION VALUE IS .TRUE. AND IF THE OPERATION IS UNSUCCESSFUL OR NO NAME IS SUPPLIED, THE FUNCTION VALUE IS .FALSE..

OPNV$A

OPNV$A IS A LOGICAL FUNCTION WITH THE FOLLOWING CALLING SEQUENCE:

```
LOG= OPNV$A(OPNKEY+TYPKEY,NAME,NAMLEN,UNIT,VERKEY,WTIME,RETRYS)
CALL OPNV$A(OPNKEY+TYPKEY,NAME,NAMLEN,UNIT,VERKEY,WTIME,RETRYS)
```

WHERE OPNKEY = A$READ, OPEN FOR READING (.NE. A$WRIT OR A$RDWR)
A$WRIT, OPEN FOR WRITING
A$RDWR, OPEN FOR READING AND WRITING

TYPKEY = A$SAMF, SAM FILE (.NE. A$DAMF)
A$DAMF, DAM FILE

NAME = FILE NAME (MAY BE A TREE NAME)

NAMLEN = LENGTH OF NAME IN CHARACTERS

UNIT = DOS FILE UNIT

VERKEY = A$NVER, NO VERIFICATION
A$VNEW, VERIFY NEW (OK TO MODIFY OLD)
A$OVAP, A$VNEW + OVERWRITE OR APPEND IF WRITING
A$VOLD, VERIFY OLD (ALREADY EXISTS)

WTIME = NUMBER OF SECONDS TO WAIT IF FILE IN USE

RETRYS = NUMBER OF TIMES TO RETRY IF FILE IN USE

ALL ARGUMENTS ARE INTEGER*2 EXCEPT NAME WHICH DOESN'T MATTER.

THIS ROUTINE OPENS A FILE OF THE GIVEN NAME ON UNIT. NOTE THAT THE FUNCTIONS OF VERIFICATION AND DELAY AS DESCRIBED BELOW ARE INDEPENDENT OF EACH OTHER.

IF WTIME AND RETRYS ARE SPECIFIED NON-ZERO AND THE FILE TO BE OPENED IS IN USE, THE OPEN WILL BE RETRIED THE SPECIFIED NUMBER OF TIMES, WITH WTIME SECONDS (ELAPSED TIME) BETWEEN EACH ATTEMPT. IF THE NUMBER OF RETRIES EXPIRES, OR IF EITHER WTIME OR RETRYS IS INITIALLY 0 AND THE FILE IS IN USE, THE FUNCTION RETURNS .FALSE..

IF VERIFICATION IS REQUESTED (VERKEY .NE. A$NVER), THE FOLLOWING ACTIONS WILL BE TAKEN:

A$VNEW IF THE FILE ALREADY EXISTS AND OPNKEY IS EITHER A$WRIT OR A$RDWR, THE USER WILL BE ASKED IF IT IS OK TO MODIFY THE OLD FILE. IF THE ANSWER IS "NO", THE FUNCTION RETURNS .FALSE.. IF THE ANSWER IS "YES", THE FILE IS OPENED.

A$OVAP THIS IS THE SAME AS A$VNEW EXCEPT THAT IF AN OLD FILE IS TO BE MODIFIED, THE USER IS ALSO ASKED IF THE FILE SHOULD BE OVERWRITTEN OR APPENDED TO. IF THE ANSWER IS "APPEND", THE FILE WILL BE POSITIONED TO END-OF-FILE.

A$VOLD THIS IS THE DEFAULT CASE IF OPNKEY=A$READ. IF NOT, AND IF THE NAMED FILE DOES NOT ALREADY EXIST, A NEW FILE WILL NOT BE CREATED AND THE FUNCTION RETURNS .FALSE..

IF ANY ERRORS NOT COVERED ABOVE OCCUR WHILE OPENING THE FILE OR POSITIONING IT (A$OVAP), THE FUNCTION RETURNS .FALSE.. IF THE OPEN IS ULTIMATELY SUCCESSFUL, THE FUNCTION RETURNS .TRUE..

OPVP$A

OPVP$A IS A LOGICAL FUNCTION WITH THE FOLLOWING CALLING SEQUENCE:

```
LOG= OPVP$A(MSG,MSGLEN,OPNKEY+TYPKEY,NAME,NAMLEN,UNIT,
           VERKEY,WTIME,RETRYS)
CALL OPVP$A(MSG,MSGLEN,OPNKEY+TYPKEY,NAME,NAMLEN,UNIT,
           VERKEY,WTIME,RETRYS)
```

WHERE MSG = PROMPT FOR NAME MESSAGE
MSGLEN = LENGTH OF MSG IN CHARACTERS
OPNKEY = A$READ, OPEN FOR READING (.NE. A$WRIT OR A$RDWR)
A$WRIT, OPEN FOR WRITING
A$RDWR, OPEN FOR READING AND WRITING
TYPKEY = A$SAMF, SAM FILE (.NE. A$DAMF)
A$DAMF, DAM FILE
NAME = FILE NAME (MAY BE A TREE NAME)
NAMLEN = LENGTH OF NAME IN CHARACTERS
UNIT = DOS FILE UNIT
VERKEY = A$NVER, NO VERIFICATION
A$VNEW, VERIFY NEW (OK TO MODIFY OLD)
A$OVAP, A$VNEW + OVERWRITE OR APPEND IF WRITING
A$VOLD, VERIFY OLD (ALREADY EXISTS)
WTIME = NUMBER OF SECONDS TO WAIT IF FILE IN USE
RETRYS = NUMBER OF TIMES TO RETRY IF FILE IN USE

ALL ARGUMENTS ARE INTEGER*2 EXCEPT NAME AND MSG WHICH DON'T MATTER.

THIS ROUTINE GETS A NAME FROM THE USER AND OPENS IT ON UNIT. NOTE THAT THE FUNCTIONS OF VERIFICATION AND DELAY AS DESCRIBED BELOW ARE INDEPENDENT OF EACH OTHER.

IF WTIME AND RETRYS ARE SPECIFIED NON-ZERO AND THE FILE TO BE OPENED IS IN USE, THE OPEN WILL BE RETRIED THE SPECIFIED NUMBER OF TIMES, WITH WTIME SECONDS (ELAPSED TIME) BETWEEN EACH ATTEMPT. IF THE NUMBER OF RETRIES EXPIRES, OR IF EITHER WTIME OR RETRYS IS INITIALLY 0 AND THE FILE IS IN USE, THE FUNCTION RETURNS .FALSE..

IF VERIFICATION IS REQUESTED (VERKEY .NE. A$NVER), THE FOLLOWING ACTIONS WILL BE TAKEN:

A$VNEW IF THE FILE ALREADY EXISTS AND OPNKEY IS EITHER A$WRIT OR A$RDWR, THE USER WILL BE ASKED IF IT IS OK TO MODIFY THE OLD FILE. IF THE ANSWER IS "NO", A NEW FILE NAME WILL BE REQUESTED. IF THE ANSWER IS "YES", THE FILE IS OPENED.

A$OVAP THIS IS THE SAME AS A$VNEW EXCEPT THAT IF AN OLD FILE IS TO BE MODIFIED, THE USER IS ALSO ASKED IF THE FILE SHOULD BE OVERWRITTEN OR APPENDED TO. IF THE ANSWER IS "APPEND", THE FILE WILL BE POSITIONED TO END-OF-FILE.

A$VOLD THIS IS THE DEFAULT CASE IF OPNKEY=A$READ. IF NOT, AND IF THE NAMED FILE DOES NOT ALREADY EXIST, A NEW FILE WILL NOT BE CREATED AND A NEW NAME WILL BE REQUESTED.

IF ANY ERRORS NOT COVERED ABOVE OCCUR WHILE OPENING THE FILE OR POSITIONING IT (A$OVAP), OR A NAME IS NOT SUPPLIED WHEN REQUESTED, THE FUNCTION RETURNS .FALSE.. IF THE OPEN IS ULTIMATELY SUCCESSFUL, THE FUNCTION RETURNS .TRUE..

CLOS$A

CLOS$A IS A LOGICAL FUNCTION WITH THE FOLLOWING CALLING SEQUENCE:

```
LOG= CLOS$A(UNIT)
CALL CLOS$A(UNIT)

WHERE UNIT   = DOS FILE UNIT

UNIT IS INTEGER*2.
```

THIS ROUTINE CLOSES THE FILE OPEN ON FILE UNIT UNIT. IF THE OPERATION IS SUCCESSFUL, THE FUNCTION IS .TRUE. AND IF UNSUCCESSFUL, THE FUNCTION IS .FALSE..

RWND$A

RWND$A IS A LOGICAL FUNCTION WITH THE FOLLOWING CALLING SEQUENCE:

```
LOG= RWND$A(UNIT)
CALL RWND$A(UNIT)

WHERE UNIT   = DOS FILE UNIT

UNIT IS INTEGER*2.
```

THIS ROUTINE REWINDS THE FILE OPEN ON FILE UNIT UNIT. IF THE OPERATION IS SUCCESSFUL, THE FUNCTION IS .TRUE. AND IF UNSUCCESSFUL, THE FUNCTION IS .FALSE..

GEND$A

GEND$A IS A LOGICAL FUNCTION WITH THE FOLLOWING CALLING SEQUENCE:

```
LOG= GEND$A(UNIT)
CALL GEND$A(UNIT)

WHERE UNIT   = DOS FILE UNIT

UNIT IS INTEGER*2.
```

THIS ROUTINE POSITIONS TO END-OF-FILE THE FILE OPEN ON FILE UNIT UNIT. IF THE OPERATION IS SUCCESSFUL, THE FUNCTION IS .TRUE. AND IF UNSUCCESSFUL, THE FUNCTION IS .FALSE..

TRNC$A

TRNC$A IS A LOGICAL FUNCTION WITH THE FOLLOWING CALLING SEQUENCE:

```
LOG= TRNC$A(UNIT)
CALL TRNC$A(UNIT)

WHERE UNIT   = DOS FILE UNIT

UNIT IS INTEGER*2.
```

THIS ROUTINE TRUNCATES THE FILE OPEN ON FILE UNIT UNIT. IF THE OPERATION IS SUCCESSFUL, THE FUNCTION IS .TRUE. AND IF UNSUCCESSFUL, THE FUNCTION IS .FALSE..

DELE$A

DELE$A IS A LOGICAL FUNCTION WITH THE FOLLOWING CALLING SEQUENCE:

```
LOG= DELE$A(NAME,NAMLEN)
CALL DELE$A(NAME,NAMLEN)

WHERE NAME   = FILE NAME (MAY BE A TREE NAME)
      NAMLEN = LENGTH OF NAME IN CHARACTERS

NAMLEN IS INTEGER*2, BUT THE TYPE OF NAME DOESN'T MATTER.
```

THIS ROUTINE WILL DELETE THE FILE IN NAME. IF THE OPERATION IS SUCCESSFUL, THE FUNCTION IS .TRUE. AND IF UNSUCCESSFUL, THE FUNCTION IS .FALSE..

EXST$A

EXST$A IS A LOGICAL FUNCTION WITH THE FOLLOWING CALLING SEQUENCE:

```
LOG= EXST$A(NAME,NAMLEN)

WHERE NAME   = FILE NAME (MAY BE A TREE NAME)
      NAMLEN = LENGTH OF NAME IN CHARACTERS

NAMLEN IS INTEGER*2, BUT THE TYPE OF NAME DOESN'T MATTER.
```

THIS ROUTINE WILL RETURN .TRUE. IF THE FILE EXISTS AND .FALSE. IF THE FILE DOES NOT EXIST OR AN ERROR WAS ENCOUNTERED.

UNIT$A

UNIT$A IS A LOGICAL FUNCTION WITH THE FOLLOWING CALLING SEQUENCE:

```
LOG= UNIT$A(UNIT)

WHERE UNIT   = DOS FILE UNIT

UNIT IS INTEGER*2.
```

THIS ROUTINE WILL RETURN .TRUE. IF THE UNIT IS OPEN AND .FALSE. IF THE UNIT IS NOT OPEN.

RPOS$A

RPOS$A IS A LOGICAL FUNCTION WITH THE FOLLOWING CALLING SEQUENCE:

```
LOG= RPOS$A(UNIT,POS)
CALL RPOS$A(UNIT,POS)

WHERE UNIT   = DOS FILE UNIT
      POS    = RETURNED ABSOLUTE POSITION

UNIT IS INTEGER*2 AND POS IS INTEGER*4.
```

THIS ROUTINE WILL RETURN THE CURRENT ABSOLUTE POSITION OF THE FILE OPEN ON UNIT UNIT. IF THE OPERATION IS SUCCESSFUL, THE FUNCTION IS .TRUE. AND IF UNSUCCESSFUL, THE FUNCTION IS .FALSE..

POSN$A

POSN$A IS A LOGICAL FUNCTION WITH THE FOLLOWING CALLING SEQUENCE:

```
LOG= POSN$A(POSKEY,UNIT,POS)
CALL POSN$A(POSKEY,UNIT,POS)

WHERE POSKEY = A$ABS, ABSOLUTE POSITION (.NE. A$REL)
               A$REL, RELATIVE POSITION
      UNIT   = DOS FILE UNIT
      POS    = POSITION (RELATIVE OR ABSOLUTE)

POSKEY AND UNIT ARE INTEGER*2 AND POS IS INTEGER*4.
```

THIS ROUTINE WILL POSITION THE FILE OPEN ON FILE UNIT UNIT TO THE SUPPLIED POSITION. IF THE OPERATION IS SUCCESSFUL, THE FUNCTION IS .TRUE. AND IF UNSUCCESSFUL, THE FUNCTION IS .FALSE..

TSCN$A

TSCN$A IS A LOGICAL FUNCTION WITH THE FOLLOWING CALLING SEQUENCE:

```
LOG= TSCN$A(KEY,UNITS,ENTRY,MAXSIZ,ENTSIZ,MAXLEV,LEV,CODE)
CALL TSCN$A(KEY,UNITS,ENTRY,MAXSIZ,ENTSIZ,MAXLEV,LEV,CODE)

WHERE KEY    = A$TREE, SCAN FULL TREE
               A$NUFD, DO NOT SCAN SUBUFDS
               A$NSEG, DO NOT SCAN SEGMENT DIRECTORIES
               A$CUFD, SCAN CURRENT UFD ONLY
               A$DLAY, PAUSE WHEN POPPING UP TO DIRECTORY
               A$BACK, BACK UP ONE LEVEL (FOR ERROR HANDLING)
      UNITS  = ARRAY OF UNIT NUMBERS MAXLEV LONG
      ENTRY  = ARRAY MAXSIZ * MAXLEV LONG
      MAXSIZ = SIZE OF EACH ENTRY IN ENTRY ARRAY
      ENTSIZ = SET TO SIZE OF CURRENT ENTRY
      MAXLEV = MAXIMUM NUMBER OF LEVELS TO SCAN
      LEV    = CURRENT LEVEL
      CODE   = RETURNED FILE SYSTEM CODE

ALL PARAMETERS ARE INTEGER*2.
```

TSCN$A SCANS THE FILE SYSTEM TREE STRUCTURE (STARTING WITH THE HOME UFD) USING RDEN$$ AND SGDR$$ TO READ UFD AND SEGMENT DIRECTORY ENTRIES INTO THE ENTRY ARRAY. EACH CALL TO TSCN$A RETURNS THE NEXT FILE ON THE CURRENT LEVEL OR THE FIRST FILE ON THE NEXT LOWER LEVEL OF THE STRUCTURE. THE VARIABLE LEV IS USED TO KEEP TRACK OF THE CURRENT LEVEL. FOR EXAMPLE, AFTER THE FIRST CALL TO TSCN$A (WITH LEV=0), LEV WILL BE RETURNED AS 1, AND ENTRY(1,1) WILL CONTAIN THE UFD ENTRY DESCRIBING THE FIRST FILE IN THE HOME UFD. IF THIS FILE IS A SUBUFD, FOLLOWING THE NEXT CALL TO TSCN$A, LEV WILL BE 2, AND ENTRY(1,2) WILL CONTAIN THE ENTRY FOR THE FIRST FILE IN THE SUBUFD.

THE VALUES OF KEY HAVE THE FOLLOWING MEANINGS:

A$TREE ALL ENTRIES IN THE TREE STRUCTURE ARE RETURNED UP TO MAXLEV LEVELS DEEP. (LEVELS BELOW LEVEL MAXLEV ARE IGNORED.)

A$NUFD WHEN A SUBUFD IS ENCOUNTERED (IN THE HOME UFD), ITS ENTRY IS RETURNED, BUT NO FILES UNDER THAT SUBUFD ARE RETURNED. IN THE ABSENSE OF SEGMENT DIRECTORIES, THIS EFFECTIVELY LIMITS THE TREE SCAN TO THE HOME UFD.

A$NSEG FILES INSIDE SEGMENT DIRECTORIES ARE NOT RETURNED.

A$CUFD THIS IS A LOGICAL COMBINATION OF A$NUFD AND A$NSEG -- ONLY FILES IN THE HOME UFD ARE RETURNED.

A$DLAY THIS KEY IS IDENTICAL TO A$TREE EXCEPT THAT DIRECTORY ENTRIES ARE RETURNED TWICE, ONCE ON THE WAY DOWN (AS FOR A$TREE), AND AGAIN ON THE WAY UP. (THIS IS NECESSARY, FOR EXAMPLE, TO IMPLEMENT TREE-DELETE FUNCTIONALITY, SINCE A DIRECTORY CANNOT BE DELETED UNTIL IT HAS BEEN EMPTIED.)

A$BACK THIS KEY IS USED TO BACK UP ONE LEVEL IN THE TREE, USED FOR ERROR HANDLING.

NOTES ON USING TSCN$A

1) FOR THE FIRST CALL OF TSCN$A, LEV SHOULD BE EQUAL TO 0. THEREAFTER IT SHOULD NOT BE MODIFIED UNTIL EOF IS REACHED ON THE TOP LEVEL UFD AT WHICH POINT LEV WILL BE RESET TO 0.

2) THE ENTRIES IN THE ENTRY ARRAY ARE IN RDEN$$ FORMAT. FOR "ENTRIES" INSIDE A SEGMENT DIRECTORY, ALL INFORMATION FROM THE DIRECTORY ENTRY IS FIRST COPIED DOWN A LEVEL. ENTRY(2,LEV) IS SET TO 0 AND ENTRY(3,LEV) IS THEN SET TO A 16-BIT ENTRY NUMBER. FOR NESTED SEGMENT DIRECTORIES, THE TYPE FIELD OF THE ENTRY IS SET APPROPRIATELY BY OPENING THE FILE WITH SRCH$$. (THE FILE IS THEN IMMEDIATELY CLOSED AGAIN.)

3) THE PARAMETER ENTSIZ IS SET TO THE NUMBER OF WORDS RETURNED BY RDEN$$. INSIDE SEGMENT DIRECTORIES, IT SHOULD BE IGNORED.

4) THE TYPE FIELDS IN THE ENTRY ARRAY -- ENTRY(20,I) -- SHOULD NOT BE MODIFIED. (TSCN$A USES THEM TO WALK UP AND DOWN THE TREE.)

5) WHEN TSCN$A REQUIRES A FILE UNIT, IT USES UNITS(LEV). BY USING RDEN$$ AND SGDR$$ READ-POSITION AND SET-POSITION FUNCTIONS CAREFULLY, IT IS POSSIBLE TO DYNAMICALLY REUSE FILE UNITS (E.G., TO SCAN TREES MORE THAN 16 LEVELS DEEP).

6) TSCN$A RETURNS .TRUE. UNTIL A NON-ZERO FILE SYSTEM CODE IS RETURNED OR UNTIL E$EOF IS RETURNED WITH LEV=0 (TOP LEVEL). E$EOF ON LOWER LEVELS OF THE TREE IS "SUPPRESSED", AND CODE IS RETURNED AS ZERO.

7) TSCN$A REQUIRES OWNER RIGHTS IN THE HOME UFD.

SAMPLE USE OF TSCN$A

THE FOLLOWING PROGRAM ILLUSTRATES HOW TSCN$A CAN BE USED TO PERFORM A TREE LISTF.

```
$INSERT SYSCOM>ERRD.F
$INSERT SYSCOM>KEYS.F
$INSERT SYSCOM>A$KEYS
C
      INTEGER MAXLEV,MAXSIZ
      PARAMETER MAXLEV=16    /* MAXIMUM LEVELS TO SCAN
      PARAMETER MAXSIZ=24    /* MAXIMUM SIZE OF EACH SLICE IN ENTRY
      INTEGER I,L,ENTRY(MAXSIZ,MAXLEV),UNITS(MAXLEV),CODE,NLEV$A
      LOGICAL TSCN$A
      DATA UNITS/1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16/
C
10    L=0  /* INITIALIZE LEVEL COUNTER
100   IF(TSCN$A(A$TREE,UNITS,ENTRY,MAXSIZ,I,MAXLEV,L,CODE))GOTO 105
      IF (CODE.NE.E$EOF) CALL ERRPR$(E$NRTN,CODE,0,0,0,0)
      CALL EXIT      /* ALL DONE IF E$EOF
      GOTO 10        /* RESTART IF 'S' TYPED
C
105   DO 200 I=1,L  /* CONSTRUCT TREENAME
         IF (ENTRY(2,I).EQ.0) GOTO 150  /* BRANCH IF SEGDIR
         CALL TNOUA(ENTRY(2,I),NLEN$A(ENTRY(2,I),32))
         GOTO 170
C
150      CALL TNOUA('(',1)    /* FORMAT SEGDIR ENTRY NUMBER
         CALL TODEC(ENTRY(3,I))
         CALL TNOUA(')',1)
C
170      IF (I.NE.L) CALL TNOUA(' > ',3)  /* TREENAME SEPARATOR
200      CONTINUE
      CALL TONL
      GOTO 100
      END
```

4.2@STRING MANIPULATION

FILL$A

FILL$A IS AN INTEGER FUNCTION WITH THE FOLLOWING CALLING SEQUENCE:

```
INT= FILL$A(NAME,NAMLEN,CHAR)
CALL FILL$A(NAME,NAMLEN,CHAR)

WHERE NAME   = NAME BUFFER TO FILL
      NAMLEN = LENGTH OF NAME IN CHARACTERS
      CHAR   = FILL CHARACTER IN FORTRAN A1 FORMAT
```

NAMLEN AND CHAR ARE INTEGER*2 AND THE TYPE OF NAME DOESN'T MATTER.

THIS ROUTINE WILL FILL THE NAME BUFFER WITH THE FILL CHARACTER SUPPLIED. THE FUCTION IS INTEGER, BUT THE VALUE IS ALWAYS 0.

NLEN$A

NLEN$A IS AN INTEGER*2 FUNCTION WITH THF FOLLOWING CALLING SEQUENCE:

```
I*2= NLEN$A(NAME,NAMLEN)
CALL NLEN$A(NAME,NAMLEN)

WHERE NAME   = NAME BUFFER TO TEST
      NAMLEN = LENGTH OF NAME IN CHARACTERS
```

NAMLEN IS INTEGER*2 AND THE TYPE OF NAME DOESN'T MATTER.

THIS ROUTINE WILL RETURN AS ITS FUNCTION VALUE THE OPERATIONAL LENGTH OF THE NAME IN NAME, NOT INCLUDING TRAILING BLANKS.

MCHR$A

MCHR$A IS AN INTEGER FUNCTION WITH THE FOLLOWING CALLING SEQUENCE:

```
INT= MCHR$A(TARRAY,TCHAR,FARRAY,FCHAR)
CALL MCHR$A(TARRAY,TCHAR,FARRAY,FCHAR)
```

WHERE TARRAY = RECEIVING ("TO") PACKED ARRAY
TCHAR = CHARACTER POSITION IN TARRAY
FARRAY = SOURCE ("FROM") PACKED ARRAY
FCHAR = CHARACTER POSITION IN FARRAY

TCHAR AND FCAHR ARE INTEGER*2, BUT THE TYPES OF TARRAY AND FARRAY DON'T MATTER.

THIS ROUTINE REPLACES THE FORTRAN STATEMENT:

```
TARRAY(TCHAR)=FARRAY(FCHAR)
```

WHEN TARRAY AND FARRAY ARE DECLARED LOGICAL*1 (IBN FORTRAN) OR OF A 1 CHARACTER DATA TYPE. ONLY THE TCHAR'TH CHARACTER IN TARRAY IS REPLACED.

THE FUNCTION VALUE WILL BE THE CHARACTER THAT WAS MOVED IN FORTRAN A1 FORMAT; I.E., THE CHARACTER IN THE LEFT MOST BYTE, RIGHT PADDED WITH BLANKS.

GCHR$A

GCHR$A IS AN INTEGER FUNCTION WITH THE FOLLOWING CALLING SEQUENCE:

```
INT= GCHR$A(FARRAY,FCHAR)
CALL GCHR$A(FARRAY,FCHAR)
```

WHERE FARRAY = SOURCE ("FROM") PACKED ARRAY
FCHAR = CHARACTER POSITION IN FARRAY

FCAHR IS INTEGER*2, BUT THE TYPE OF FARRAY DOESN'T MATTER.

THIS ROUTINE REPLACES THE FORTRAN STATEMENT:

```
CHAR=FARRAY(FCHAR)
```

WHEN FARRAY IS DECLARED LOGICAL*1 (IBN FORTRAN) OR OF A 1 CHARACTER DATA TYPE.

THE FUNCTION VALUE WILL BE THE ACCESSED CHARACTER IN FORTRAN A1 FORMAT; I.E., THE CHARACTER IN THE LEFT MOST BYTE, RIGHT PADDED WITH BLANKS.

TREE$A

TREE$A IS AN LOGICAL FUNCTION WITH THE FOLLOWING CALLING SEQUENCE:

```
LOG= TREE$A(NAME,NAMLEN,FSTART,FLEN)

WHERE NAME   = FILE NAME
      NAMLEN = LENGTH OF NAME IN CHARACTERS
      FSTART = FIRST CHARACTER OF FINAL FILE NAME IN TREE
      FLEN   = LENGTH OF FINAL FILE NAME IN CHARACTERS
```

ALL ARGUMENTS ARE INTEGER*2 AND THE TYPE OF NAME DOESN'T MATTER.

THIS ROUTINE WILL SCAN A FILE NAME AND DETERMINE IF IT IS A TREE NAME. IF IT IS A TREE NAME, THE FUNCTION IS .TRUE. AND IF NOT, IT IS .FALSE.. IN ADDITION, THE FINAL NAME (OR ENTIRE NAME IF NOT IN A TREE) IS LOCATED IN THE STRING. NOTE THAT IF THE NAME IS EMPTY, FSTART=FLEN=0.

TYPE$A

TYPE$A IS A LOGICAL FUNCTION WITH THE FOLLOWING CALLING SEQUENCE:

```
LOG = TYPE$A(KEY,STRING,LENGTH)
```

ARGUMENTS:

```
KEY    = STRING TYPE TO BE TESTED FOR, POSSIBLE KEYS ARE:
         A$NAME - CAN STRING BE INTERPRETED AS A NAME,
         A$DEC - CAN STRING BE INTERPRETED AS A DECIMAL NUMBER,
         A$OCT - CAN STRING BE INTERPRETED AS AN OCTAL NUMBER,
         A$HEX - CAN STRING BE INTERPRETED AS A HEXADECIMAL NUMBER.
STRING = STRING TO BE TESTED, PACKED TWO CHARACTERS PER WORD.
LENGTH = LENGTH OF STRING, IN CHARACTERS.
```

FUNCTION:

TYPE$A WILL TEST A CHARACTER STRING TO DETERMINE IF IT CAN BE INTERPRETED AS THE TYPE SPECIFIED BY KEY. A STRING IS NAME IF IT CONTAINS AT LEAST ONE ALPHABETIC OR SPECIAL CHARACTER (OTHER THAN A LEADING + OR -), A DECIMAL NUMBER IF IT CONTAINS ONLY THE DIGITS 0 - 9, AN OCTAL NUMBER IF IT CONTAINS ONLY THE DIGITS 0 - 7, AND A HEXADECIMAL NUMBER IF IT CONTAINS ONLY THE DIGITS 0 - 9 AND THE CHARACTERS A - F (UPPER CASE ONLY). A NUMBER MAY HAVE A LEADING SIGN AND ANY NUMBER OF BLANKS BETWEEN THE SIGN AND THE FIRST DIGIT, HOWEVER IMBEDDED BLANKS WITHIN THE NUMBER ITSELF ARE NOT ALLOWED. A NUMBER MUST ALSO HAVE AT LEAST ONE DIGIT.

LEADING AND TRAILING BLANKS ARE IGNORED. THE FUNCTION IS TRUE IF STRING SATISFIES THE CONDITIONS REQUIRED BY THE KEY USED, OTHERWISE IT IS FALSE. A NULL STRING (IE. LENGTH EQUAL TO ZERO) WILL ONLY RETURN A FUNCTION VALUE OF TRUE IF KEY IS A$NAME.

ALL ARGUMENTS ARE INTEGER*2 EXCEPT STRING WHOSE TYPE DOES NOT MATTER.

APPLIB CALLS:

GCHR$A, NLEN$A

MSTR$A

MSTR$A IS AN INTEGER FUNCTION USED TO MOVE ONE STRING TO ANOTHER, IT HAS THE FOLLOWING CALLING SEQUENCE:

```
INT = MSTR$A(A,ALEN,B,BLEN)
CALL  MSTR$A(A,ALEN,B,BLEN)
```

ARGUMENTS:

```
A    = SOURCE STRING, PACKED TWO CHARACTERS PER WORD,
ALEN = LENGTH OF A, IN CHARACTERS, MUST BE .GE. ZERO,
B    = DESTINATION STRING, PACKED,
BLEN = LENGTH OF B, IN CHARACTERS, MUST BE .GE. ZERO.
```

FUNCTION:

MSTR$A WILL MOVE THE SOURCE STRING TO THE DESTINATION STRING. IF THE SOURCE STRING IS LONGER THAN THE DESTINATION STRING IT WILL BE TRUNCATED AND IF IT IS SHORTER IT WILL BE PADDED WITH BLANKS. THE SOURCE AND DESTINATION STRINGS MAY OVERLAP. THE FUNCTION VALUE WILL BE EQUAL TO THE NUMBER OF CHARACTERS MOVED (EXCLUDING BLANK PADDING). IF EITHER STRING IS NULL (IE. LENGTH EQUAL TO ZERO) NO CHARACTERS ARE MOVED AND THE FUNCTION WILL BE EQUAL TO ZERO.

ALL ARGUMENTS ARE INTEGER*2 EXCEPT A AND B WHOSE TYPES DO NOT MATTER.

APPLIB CALLS:

MSUB$A, NLEN$A

MSUB$A

MSUB$A IS AN INTEGER FUNCTION USED TO MOVE ONE SUBSTRING TO ANOTHER, IT HAS THE FOLLOWING CALLING SEQUENCE:

```
INT = MSUB$A(A,ALEN,AFC,ALC,B,BLEN,BFC,BLC)
CALL  MSUB$A(A,ALEN,AFC,ALC,B,BLEN,BFC,BLC)
```

ARGUMENTS:

A = ARRAY CONTAINING SOURCE SUBSTRING, PACKED TWO CHARACTERS PER WORD,
ALEN = LENGTH OF A, IN CHARACTERS, MUST BE .GE. ZERO,
AFC = FIRST CHARACTER POSITION OF SUBSTRING IN A,
ALC = LAST CHARACTER POSITION OF SUBSTRING IN A,
B = ARRAY CONTAINING DESTINATION SUBSTRING, PACKED TWO CHARACTERS PER WORD,
BLEN = LENGTH OF B, IN CHARACTERS, MUST BE .GE. ZERO,
BFC = FIRST CHARACTER POSITION OF SUBSTRING IN B,
BLC = LAST CHARACTER POSITION OF SUBSTRING IN B.

FUNCTION:

MSUB$A WILL MOVE THE SOURCE SUBSTRING CONTAINED IN A TO THE DESTINATION SUBSTRING CONTAINED IN B. IF THE SOURCE SUBSTRING IS LONGER THAN THE DESTINATION SUBSTRING IT WILL BE TRUNCATED AND IF IT IS SHORTER IT WILL BE PADDED WITH BLANKS. THE SOURCE AND DESTINATION SUBSTRINGS MAY OVERLAP.

IF EITHER SUBSTRING IS NULL (IE. LENGTH EQUAL TO ZERO) NO CHARACTERS ARE MOVED AND THE FUNCTION WILL BE EQUAL TO ZERO, OTHERWISE IT IS EQUAL TO THE NUMBER OF CHARACTERS MOVED (EXCLUDING BLANKS USED FOR PADDING).

THIS ROUTINE CHECKS THE VALIDITY OF THE STRING SUBSCRIPTS AND WILL DISPLAY AN ERROR MESSAGE IF AN ILLEGAL SUBSCRIPT IS ENCOUNTERED. A SUBSCRIPT MUST BE GREATER THAN ZERO (UNLESS BOTH ARE ZERO, IN WHICH CASE THE SUBSTRING IS NULL) AND THE SECOND SUBSCRIPT MUST BE GREATER THAN OR EQUAL TO THE FIRST. BOTH SUBSCRIPTS MUST BE LESS THAN OR EQUAL TO THE STRING LENGTH.

ALL ARGUMENTS ARE INTEGER*2 EXCEPT A AND B WHOSE TYPES DO NOT MATTER.

APPLIB CALLS:

MCHR$A

CSTR$A

CSTR$A IS A LOGICAL FUNCTION USED TO COMPARE TWO STRINGS FOR EQUALITY, IT HAS THE FOLLOWING CALLING SEQUENCE:

```
LOG = CSTR$A(A,ALEN,B,BLEN)
```

ARGUMENTS:

```
A    = STRING TO BE COMPARED, PACKED TWO CHARACTERS PER WORD,
ALEN = LENGTH OF A, IN CHARACTERS, MUST BE .GE. ZERO,
B    = STRING TO BE COMPARED AGAINST, PACKED,
BLEN = LENGTH OF B, IN CHARACTERS, MUST BE .GE. ZERO.
```

FUNCTION:

CSTR$A WILL COMPARE TWO STRINGS FOR EQUALITY. THE FUNCTION WILL BE TRUE IF EACH CHARACTER IN STRING A MATCHES THE CORRESPONDING CHARACTER IN STRING B, OR IF BOTH STRINGS ARE NULL (IE. LENGTH EQUAL TO ZERO), OTHERWISE THE FUNCTION WILL BE FALSE. ONLY THE OPERATIONAL LENGTHS ARE USED IN THE COMPARISION (IE. TRAILING BLANKS ARE IGNORED). CSTR$A AVOIDS THE RESTRICTIONS IMPOSED BY NAMEQ$ CONCERNING TRAILING BLANKS AND NUMERIC FIELDS.

ALL ARGUMENTS ARE INTEGER*2 EXCEPT A AND B WHOSE TYPES DO NOT MATTER.

APPLIB CALLS:

CSUB$A, NLEN$A

CSUB$A

CSUB$A IS A LOGICAL FUNCTION USED TO COMPARE SUBSTRINGS FOR EQUALITY, IT HAS THE FOLLOWING CALLING SEQUENCE:

```
LOG = CSUB$A(A,ALEN,AFC,ALC,B,BLEN,BFC,BLC)
```

ARGUMENTS:

```
A    = ARRAY CONTAINING SUBSTRING TO BE COMPARED, PACKED TWO
       CHARACTERS PER WORD,
ALEN = LENGTH OF A, IN CHARACTERS, MUST BE .GE. ZERO,
AFC  = FIRST CHARACTER POSITION OF SUBSTRING IN A,
ALC  = LAST CHARACTER POSITION OF SUBSTRING IN A,
B    = ARRAY CONTAINING SUBSTRING TO BE COMPARED AGAINST, PACKED
       TWO CHARACTERS PER WORD,
BLEN = LENGTH OF B, IN CHARACTERS, MUST BE .GE. ZERO,
BFC  = FIRST CHARACTER POSITION OF SUBSTRING IN B,
BLC  = LAST CHARACTER POSITION OF SUBSTRING IN B.
```

FUNCTION:

CSUB$A WILL COMPARE TWO SUBSTRINGS FOR EQUALITY. IF EACH CHARACTER IN THE A SUBSTRING MATCHES THE CORRESPONDING CHARACTER IN THE B SUBSTRING, OR BOTH SUBSTRINGS ARE NULL (IE. LENGTH EQUAL TO ZERO) THE FUNCTION WILL BE TRUE. IF TWO CORRESPONDING CHARACTERS DO NOT MATCH, OR IF THE LENGTHS OF THE SUBSTRINGS ARE NOT EQUAL THE FUNCTION WILL BE FALSE.

THIS ROUTINE CHECKS THE VALIDITY OF THE STRING SUBSCRIPTS AND WILL DISPLAY AN ERROR MESSAGE IF AN ILLEGAL SUBSCRIPT IS ENCOUNTERED. A SUBSCRIPT MUST BE GREATER THAN ZERO (UNLESS BOTH ARE ZERO, IN WHICH CASE THE SUBSTRING IS NULL) AND THE SECOND SUBSCRIPT MUST BE GREATER THAN OR EQUAL TO THE FIRST. BOTH SUBSCRIPTS MUST BE LESS THAN OR EQUAL TO THE STRING LENGTH.

ALL ARGUMENTS ARE INTEGER*2 EXCEPT A AND B WHOSE TYPES DO NOT MATTER.

APPLIB CALLS:

GCHR$A

LSTR$A

LSTR$A IS A LOGICAL FUCTION USED TO LOCATE ONE STRING WITHIN ANOTHER, IT HAS THE FOLLOWING CALLING SEQUENCE:

```
LOG = LSTR$A(A,ALEN,B,BLEN,FCP,LCP)
CALL  LSTR$A(A,ALEN,B,BLEN,FCP,LCP)
```

ARGUMENTS:

A = STRING TO BE LOCATED, PACKED TWO CHARACTERS PER WORD,
ALEN = LENGTH OF A, IN CHARACTERS, MUST BE .GE. ZERO,
B = STRING TO BE SEARCHED, PACKED,
BLEN = LENGTH OF B, IN CHARACTERS, MUST BE .GE. ZERO,
FCP = FIRST CHARACTER POSITION IN B OF SUBSTRING THAT MATCHES STRING A,
LCP = LAST CHARACTER POSITION IN B OF SUBSTRING THAT MATCHES STRING A.

FUNCTION:

LSTR$A WILL SEARCH STRING B FOR THE FIRST OCCURENCE OF STRING A. IF STRING A IS FOUND THE FUNCTION WILL BE TRUE AND FCP AND LCP WILL BE EQUAL TO THE CHARACTER POSITIONS OF THE SUBSTRING IN B THAT MATCHES STRING A. IF STRING A IS NOT FOUND OR IF EITHER STRING IS NULL (IE. LENGTH EQUAL TO ZERO) THE FUNCTION WILL BE FALSE AND FCP AND LCP WILL BE EQUAL TO ZERO. EACH STRING IS LOGICALLY TRUNCATED TO ITS OPERATIONAL LENGTH BEFORE THE SEARCH IS PERFORMED (IE. TRAILING BLANKS ARE IGNORED).

ALL ARGUMENTS ARE INTEGER*2 EXCEPT A AND B WHOSE TYPES DO NOT MATTER.

APPLIB CALLS:

LSUB$A, NLEN$A

LSUB$A

LSUB$A IS A LOGICAL FUNCTION USED TO LOCATE ONE SUBSTRING WITHIN ANOTHER, IT HAS THE FOLLOWING CALLING SEQUENCE:

```
LOG = LSUB$A(A,ALEN,AFC,ALC,B,BLEN,BFC,BLC,FCP,LCP)
CALL  LSUB$A(A,ALEN,AFC,ALC,B,BLEN,BFC,BLC,FCP,LCP)
```

ARGUMENTS:

A = ARRAY CONTAINING SUBSTRING TO BE LOCATED, PACKED TWO CHARACTERS PER WORD,
ALEN = LENGTH OF A, IN CHARACTERS, MUST BE .GE. ZERO,
AFC = FIRST CHARACTER POSITION OF SUBSTRING IN A,
ALC = LAST CHARACTER POSITION OF SUBSTRING IN A,
B = ARRAY CONTAINING SUBSTRING TO BE SEARCHED, PACKED TWO CHARACTERS PER WORD,
BLEN = LENGTH OF B, IN CHARACTERS, MUST BE .GE. ZERO,
BFC = FIRST CHARACTER POSITION OF SUBSTRING IN B,
BLC = LAST CHARACTER POSITION OF SUBSTRING IN B,
FCP = FIRST CHARACTER POSITION IN B OF SUBSTRING THAT MATCHES SUBSTRING IN A,
LCP = LAST CHARACTER POSITION IN B OF SUBSTRING THAT MATCHES SUBSTRING IN A.

FUNCTION:

LSUB$A WILL SEARCH THE SUBSTRING CONTAINED IN B FOR THE FIRST OCCURENCE OF THE SUBSTRING CONTAINED IN A. IF A MATCH IS FOUND FCP AND LCP WILL BE EQUAL TO THE CHARACTER POSITIONS IN B OF THE MATCHING SUBSTRING AND THE FUNCTION WILL BE TRUE. IF A MATCHING SUBSTRING CANNOT BE FOUND OR IF EITHER SUBSTRING IS NULL (IE. LENGTH EQUAL TO ZERO) THE FUNCTION WILL BE FALSE AND FCP AND LCP WILL BE EQUAL TO ZERO.

THIS ROUTINE CHECKS THE VALIDITY OF THE STRING SUBSCRIPTS AND WILL DISPLAY AN ERROR MESSAGE IF AN ILLEGAL SUBSCRIPT IS ENCOUNTERED. A SUBSCRIPT MUST BE GREATER THAN ZERO (UNLESS BOTH ARE ZERO, IN WHICH CASE THE SUBSTRING IS NULL) AND THE SECOND SUBSCRIPT MUST BE GREATER THAN OR EQUAL TO THE FIRST. BOTH SUBSCRIPTS MUST BE LESS THAN OR EQUAHE STRING LENGTH.

ALL ARGUMENTS ARE INTEGER*2 EXCEPT A AND B WHOSE TYPES DO NOT MATTER.

APPLIB CALLS:

CSUB$A

JSTR$A

JSTR$A IS A LOGICAL FUNCTION USED TO LEFT OR RIGHT JUSTIFY A STRING, IT HAS THE FOLLOWING CALLING SEQUENCE:

```
LOG = JSTR$A(KEY,STRING,LENGTH)
CALL  JSTR$A(KEY,STRING,LENGTH)
```

ARGUMENTS:

KEY = DETERMINES DIRECTION OF JUSTIFICATION, POSSIBLE VALUES ARE:
A$RGHT - RIGHT JUSTIFY,
A$LEFT - LEFT JUSTIFY,
STRING = STRING TO BE JUSTIFIED, PACKED TWO CHARACTERS PER WORD,
LENGTH = LENGTH OF STRING IN CHARACTERS, MUST BE .GE. ZERO.

FUNCTION:

JSTR$A WILL LEFT OR RIGHT JUSTIFIY A STRING WITHIN ITSELF. THE FUNCTION WILL BE TRUE IF JUSTIFICATION IS SUCCESSFUL, FALSE IF THE STRING LENGTH IS LESS THAN ZERO OR IF A BAD KEY IS USED.

ALL ARGUMENTS ARE INTEGER*2 EXCEPT STRING WHOSE TPYE DOES NOT MATTER.

APPLIB CALLS:

NLEN$A, FILL$A, MSUB$A, GCHR$A

4.3 USER QUERY

YSNO$A

YSNO$A IS A LOGICAL FUNCTION WITH THE FOLLOWING CALLING SEQUENCE:

```
LOG= YSNO$A(MSG,MSGLEN,DEFKEY)

    WHERE MSG    = MESSAGE TEXT
      MSGLEN = MESSAGE LENGTH IN CHARACTERS
      DEFKEY = A$NDEF, NO DEFAULT ACCEPTED
               A$DNO , DEFAULT = "NO" (.FALSE.)
               A$DYES, DEFAULT = "YES" (.TRUE.)
```

MSGLEN AND DEFKEY ARE INTEGER*2. THE TYPE OF MSG DOESN'T MATTER.

THIS ROUTINE WILL PRINT THE SUPPLIED MESSAGE AND APPEND THE CHARACTERS "? " TO IT. IT THEN READS A USER RESPONSE. IF THE ANSWER IS "YES" OR "OK", THE FUNCTION VALUE IS .TRUE.. IF THE ANSWER IS "NO", THE FUNCTION VALUE IS .FALSE.. IF AN ILLEGAL ANSWER IS PROVIDED OR IF NO DEFAULT IS ACCEPTED, MSG WILL BE REPEATED.

NOTE, USER RESPONSES MAY BE ABBREVIATED TO FIRST 1 OR 2 CHARACTERS.

RNAM$A

RNAM$A IS A LOGICAL FUNCTION WITH THE FOLLOWING CALLING SEQUENCE:

```
LOG= RNAM$A(MSG,MSGLEN,NAMKEY,NAME,NAMLEN)

    WHERE MSG    = MESSAGE TEXT
      MSGLEN = MESSAGE LENGTH IN CHARACTERS
      NAMKEY = A$FUPP, FORCE UPPER CASE (.NE. A$UPLW OR A$RAWI)
               A$UPLW, DO NOT FORCE UPPER CASE
               A$RAWI, READ REST OF LINE
      NAME   = RETURNED NAME
      NAMLEN = LENGTH OF NAME BUFFER IN CHARACTERS (.LE. 80)
```

ALL ARGUMENTS ARE INTEGER*2 EXCEPT MSG AND NAME WHICH DON'T MATTER.

THIS ROUTINE FILLS NAME WITH BLANKS AND THEN PRINTS THE SUPPLIED MESSAGE AND APPENDS THE CHARACTERS ": " TO IT. IT THEN READS A USER RESPONSE. IF THE RESPONSE IS NOT A LEGAL NAME OR IF THE NAME PROVIDED IS TOO LONG FOR THE SUPPLIED BUFFER, THE ERROR WILL BE REPORTED AND MSG WILL BE REPEATED. IF NO NAME IS PROVIDED, THE VALUE OF THE FUNCTION WILL BE .FALSE.. IF A LEGAL NAME IS PROVIDED, THE FUNCTION VALUE WILL BE .TRUE..

RNUM$A

RNUM$A IS A LOGICAL FUNCTION WITH THE FOLLOWING CALLING SEQUENCE:

```
LOG= RNUM$A(MSG,MSGLEN,NUMKEY,VALUE)

    WHERE MSG    = MESSAGE TEXT
      MSGLEN = MESSAGE LENGTH IN CHARACTERS
      NUMKEY = A$DEC, DECIMAL (.NE. A$OCT OR A$HEX)
               A$OCT, OCTAL
               A$HEX, HEXADECIMAL
      VALUE  = RETURNED VALUE
```

ALL ARGUMENTS ARE INTEGER*2 EXCEPT VALUE WHICH IS INTEGER*4.

THIS ROUTINE WILL PRINT THE SUPPLIED MESSAGE AND APPEND THE CHARACTERS ": " TO IT. IT THEN READS A USER RESPONSE. IF THE RESPONSE IS NOT A LEGAL NUMBER OR IF THE NUMBER PROVIDED HAS TOO MANY DIGITS FOR AN INTEGER*4 VALUE, THE ERROR WILL BE REPORTED AND MSG WILL BE REPEATED. IF NO NUMBER IS PROVIDED, THE VALUE OF THE FUNCTION WILL BE .FALSE. AND VALUE=0. IF A LEGAL NUMBER IS PROVIDED, THE FUNCTION VALUE WILL BE .TRUE. AND THE VALUE WILL BE RETURNED IN VALUE.

NOTE, NUMBERS MAY BE PRECEDED BY A "+" OR "-".

4.4 SYSTEM INFORMATION

TIME$A

TIME$A IS A DOUBLE PRECISION FUNCTION WITH THE FOLLOWING CALLING SEQUENCE:

```
R*8= TIME$A(TIME)
CALL TIME$A(TIME)
```

WHERE TIME = TIME OF DAY IN THE FORM "HR:MN:SC"

THE TYPE OF THE TIME ARRAY DOES NOT MATTER AS LONG AS IT IS AT LEAST 8 CHARACTERS LONG.

THIS ROUTINE RETURNS THE TIME OF DAY IN THE FORM "HR:MN:SC".

THE VALUE OF THE FUNCTION IS THE TIME OF DAY IN DECIMAL HOURS. THIS VALUE MAY BE RECEIVED AS EITHER REAL*4 OR REAL*8.

CTIM$A

CTIM$A IS A DOUBLE PRECISION FUNCTION WITH THE FOLLOWING CALLING SEQUENCE:

```
R*8= CTIM$A(CPUTIM)
CALL CTIM$A(CPUTIM)
```

WHERE CPUTIM = CPU TIME IN CENTISECONDS

CPUTIM IS INTEGER*4.

THIS ROUTINE RETURNS CPU TIME SINCE LOGIN AS INTEGER*4 CENTISECONDS IN THE CPUTIM ARGUMENT.

THE FUNCTION VALUE WILL BE CPU TIME SINCE LOGIN IN SECONDS. THIS VALUE MAY BE RECEIVED AS EITHER REAL*4 OR REAL*8.

DTIM$A

DTIM$A IS A DOUBLE PRECISION FUNCTION WITH THE FOLLOWING CALLING SEQUENCE:

```
R*8= DTIM$A(DSKTIM)
CALL DTIM$A(DSKTIM)
```

WHERE DSKTIM = DSK TIME IN CENTISECONDS

DSKTIM IS INTEGER*4.

THIS ROUTINE RETURNS DISK TIME SINCE LOGIN AS INTEGER*4 CENTISECONDS IN THE DSKTIM ARGUMENT.

THE FUNCTION VALUE WILL BE DISK TIME SINCE LOGIN IN SECONDS. THIS VALUE MAY BE RECEIVED AS EITHER REAL*4 OR REAL*8.

DATE$A

DATE$A IS A DOUBLE PRECISION FUNCTION WITH THE FOLLOWING CALLING SEQUENCE:

```
R*8= DATE$A(DATE)
CALL DATE$A(DATE)
```

WHERE DATE = DATE IN THE FORM "DAY, MON DD 19YR"

THE TYPE OF THE DATE ARRAY DOES NOT MATTER AS LONG AS IT IS AT LEAST 16 CHARACTERS LONG.

THIS ROUTINE RETURNS THE DATE IN THE FORM "DAY, MON DD 19YR".

THE VALUE OF THE FUNCTION IS THE DATE IN THE FORM "MM/DD/YR". THIS VALUE MUST BE RECEIVED AS REAL*8.

NOTE THAT THIS ROUTINE IS GOOD FOR THE PERIOD JANUARY 1, 1977 THROUGH DECEMBER 31, 1986.

EDAT$A

EDAT$A IS A DOUBLE PRECISION FUNCTION WITH THE FOLLOWING CALLING SEQUENCE:

```
R*8= EDAT$A(EDATE)
CALL EDAT$A(EDATE)
```

WHERE EDATE = DATE IN THE FORM "DAY, DD MON 19YR"

THE TYPE OF THE EDATE ARRAY DOES NOT MATTER AS LONG AS IT IS AT LEAST 16 CHARACTERS LONG.

THIS ROUTINE RETURNS THE DATE IN THE EUROPEAN (MILITARY) FORM "DAY, DD MON 19YR".

THE VALUE OF THE FUNCTION IS THE DATE IN THE FORM "DD/MM/YR". THIS VALUE MUST BE RECEIVED AS REAL*8.

NOTE THAT THIS ROUTINE IS GOOD FOR THE PERIOD 1 JANUARY 1977 THROUGH 31 DECEMBER 1986.

DOFY$A

DOFY$A IS A DOUBLE PRECISION FUNCTION WITH THE FOLLOWING CALLING SEQUENCE:

```
R*8= DOFY$A(DOFY)
CALL DOFY$A(DOFY)
```

WHERE DOFY = DAY OF YEAR IN THE FORM "DDD "

THE TYPE OF THE DOFY ARRAY DOES NOT MATTER AS LONG AS IT IS AT LEAST 4 CHARACTERS LONG.

THIS ROUTINE RETURNS THE DAY OF THE YEAR IN THE FORM "DDD ".

THE VALUE OF THE FUNCTION IS THE DATE IN THE FORM YR.DDD SUITABLE FOR PRINTING IN FORMAT F6.3. THIS VALUE CAN BE RECEIVED AS EITHER REAL*4 OR REAL*8.

NOTE THAT THIS ROUTINE IS GOOD FOR THE PERIOD JANUARY 1, 1977 THROUGH DECEMBER 31, 1986.

4.5 MATHEMATICAL

RNDI$A

RNDI$A IS A DOUBLE PRECISION FUNCTION WITH THE FOLLOWING CALLING SEQUENCE:

```
R*8= RNDI$A(SEED)
CALL RNDI$A(SEED)
```

WHERE SEED = TIME OF DAY IN CENTISECONDS

SEED IS INTEGER*4.

THIS ROUTINE RETURNS THE TIME OF DAY IN CENTISECONDS. THE FUNCTION VALUE WILL BE THE TIME OF DAY IN SECONDS. THIS VALUE MAY BE RECEIVED AS EITHER REAL*4 OR REAL*8.

NOTE, BECAUSE THIS FUNCTION IS USED TO INITIALIZE A RANDOM NUMBER GENERATOR, IF THE VALUE IS EXACTLY 0, 1234567 OR 12345.67 WILL BE RETURNED INSTEAD.

RAND$A

RAND$A IS A DOUBLE PRECISION FUNCTION WITH THE FOLLOWING CALLING SEQUENCE:

```
R*8= RAND$A(SEED)
CALL RAND$A(SEED)
```

WHERE SEED = INPUT IS PREVIOUS SEED, OUTPUT IS NEW SEED

SEED IS INTEGER*4.

THIS ROUTINE UPDATES A SEED TO A NEW SEED (SEED) BASED UPON THE LINEAR CONGRUENTIAL METHOD:

```
U(I)=FLOAT(K(I))/M
```

WHERE K(I) = B*K(I-1) MODULO M
B = 16807.0
M = 2**31-1 = 2147483647.0

B AND M ARE FROM: LEWIS, GOODMAN, AND MILLER, "A PSEUDO-RANDOM NUMBER GENERATOR FOR THE SYSTEM/360", IBM SYSTEMS JOURNAL, VOL 8, NO 2, 1969, PP 136-145.

K(I-1) IS THE INPUT VALUE OF SEED AND K(I) IS THE RETURNED VALUE.

THE VALUE OF THE FUNCTION IS U(I) WHICH REPRESENTS A PROBABILITY AND IS BETWEEN 0.0 AND 1.0. THIS VALUE MAY BE RECEIVED AS EITHER REAL*4 OR REAL*8.

4.6 CONVERSION

ENCD$A

ENCD$A IS A LOGICAL FUNCTION WITH THE FOLLOWING CALLING SEQUENCE:

```
LOG= ENCD$A(ARRAY,WIDTH,DEC,VALUE)
CALL ENCD$A(ARRAY,WIDTH,DEC,VALUE)
```

WHERE ARRAY = ARRAY TO RECEIVE VALUE
WIDTH = FIELD WIDTH AS IN FORMAT FW.D (SHOULD BE EVEN)
DEC = PLACES TO RIGHT OF DECIMAL PT. AS IN FORMAT FW.D
VALUE = DOUBLE PRECISION VALUE TO BE ENCODED

WIDTH AND DEC ARE INTEGER*2, VALUE IS REAL*8, AND THE TYPE OF ARRAY DOESN'T MATTER.

THIS ROUTINE WILL ATTEMPT TO ENCODE VALUE IN THE SUPPLIED FW.D FORMAT IF IT WILL FIT. IF NOT, THE DEC ARGUMENT IS DECREMENTED (MOVING THE DECIMAL POINT TO THE RIGHT) UNTIL IT WILL FIT. IF DEC REACHES 0, OR IS ORIGINALLY SUPPLIED AS 0, VALUE WILL BE ENCODED IN IW FORMAT IF THE NUMBER WILL FIT INTO A 32-BIT INTEGER. IF NOT, AND IF THE FIELD IS WIDE ENOUGH (WIDTH > 7), THE VALUE WILL BE ENCODED IN E FORMAT. IF THE FIELD IS NOT WIDE ENOUGH, IT WILL BE FILLED WITH ASTERISKS.

NOTE THAT THE LARGEST VALUE OF WIDTH WILL BE 16. IF IT IS LARGER THAN 16, ONLY THE FIRST 16 CHARACTERS OF ARRAY WILL BE USED.

THE FUNCTION VALUE WILL BE .TRUE. IF THE ENCODE WAS SUCCESSFUL AND .FALSE. IF THE FIELD WAS FILLED WITH ASTERISKS.

NOTE THAT ARRAY IS THE ONLY ARGUMENT WHICH IS ACTUALLY MODIFIED IN THE CALLING PROGRAM.

CNVA$A

CNVA$A IS A LOGICAL FUNCTION WITH THE FOLLOWING CALLING SEQUENCE:

```
LOG= CNVA$A(NUMKEY,NAME,NAMLEN,VALUE)
CALL CNVA$A(NUMKEY,NAME,NAMLEN,VALUE)
```

WHERE NUMKEY = A$DEC, DECIMAL (.NE. A$OCT OR A$HEX)
A$OCT, OCTAL
A$HEX, HEXADECIMAL
NAME = ASCII NUMBER STRING
NAMLEN = LENGTH OF NAME IN CHARACTERS
VALUE = RETURNED VALUE

NUMKEY AND NAMLEN ARE INTEGER*2, VALUE IS INTEGER*4, AND THE TYPE OF NAME DOESN'T MATTER.

THIS ROUTINE WILL CONVERT AN ASCII DIGIT STRING INTO ITS BINARY VALUE FOR DECIMAL, OCTAL AND HEXADECIMAL NUMBERS. THE NUMBERS MAY BE EXPLICITLY SIGNED. LEADING AND TRAILING BLANKS ARE IGNORED AS WELL AS BLANKS BETWEEN THE SIGN AND THE NUMBER. HOWEVER, BLANKS WITHIN THE NUMBER ARE NOT ALLOWED. IF THE NUMBER CONVERTS SUCCESSFULLY, THE FUNCTION IS .TRUE. AND IF NOT, IT IS .FALSE. AND VALUE=0. NOTE THAT FOR DECIMAL CONVERSIONS, OVERFLOW WILL BE CONSIDERED AS UNSUCCESSFUL WHEREAS FOR OCTAL AND HEXADECIMAL CONVERSIONS, OVERFLOW IS IGNORED.

CNVB$A

CNVB$A IS AN INTEGER FUNCTION USED TO CONVERT A BINARY NUMBER TO AN ASCII DIGIT STRING, IT HAS THE FOLLOWING CALLING SEQUENCE:

```
I*2 = CNVB$A(NUMKEY,VALUE,NAME,NAMLEN)
```

ARGUMENTS:

NUMKEY = NUMBER BASE TO CONVERT TO, POSSIBLE VALUES ARE:
A$DEC - SIGNED DECIMAL NUMBER WITH LEADING BLANKS,
A$DECU - UNSIGNED DECIMAL NUMBER WITH LEADING BLANKS,
A$DECZ - SIGNED DECIMAL NUMBER WITH LEADING ZEROS,
A$OCT - SIGNED OCTAL NUMBER, LEADING BLANKS
A$OCTZ - SIGNED OCTAL, LEADING ZEROS,
A$HEX - SIGNED HEXADECIMAL, LEADING BLANKS,
A$HEXZ - SIGNED HEXADECIMAL, LEADING ZEROS.
NAME = RETURNED STRING FOR ASCII NUMBER.
NAMLEN = LENGTH OF NAME IN CHARACTERS.
VALUE = INTEGER*4 BINARY NUMBER TO BE CONVERTED.

FUNCTION:

CNVB$A WILL CONVERT A BINARY NUMBER INTO AN ASCII DIGIT STRING FOR DECIMAL, OCTAL, AND HEXADECIMAL NUMBERS. THE RETURNED DIGIT STRING WILL BE RIGHT JUSTIFIED IN NAME AND PRECEDED BY LEADING BLANKS OR ZEROS.

IF VALUE IS NEGATIVE AND TO BE TREATED AS SIGNED DECIMAL, NAME WILL BEGIN WITH AN INITIAL "-" SIGN. IF THE NUMBER CONVERTS SUCCESSFULLY, THE FUNCTION VALUE IS THE NUMBER OF DIGITS AND IF NOT, IT IS ZERO.

ALL ARGUMENTS ARE INTEGER*2 EXCEPT VALUE, WHICH IS INTEGER*4, AND NAME, WHOSE TYPE DOES NOT MATTER.

APPLIB CALLS:

FILL$A, MCHR$A

4.7@PARSING

CMDL$A

CMDL$A IS A LOGICAL FUNCTION FOR PARSING A PRIMOS TYPE COMMAND LINE AND HAS THE FOLLOWING CALLING SEQUENCE:

```
LOG = CMDL$A(KEY,KWLIST,KWINDX,OPTBUF,BUFLEN,OPTION,VALUE,KWINFO)
CALL  CMDL$A(KEY,KWLIST,KWINDX,OPTBUF,BUFLEN,OPTION,VALUE,KWINFO)
```

ARGUMENTS

KEY = A$READ, RETURN THE NEXT KEYWORD ENTRY IN THE COMMAND LINE.
= A$NEXT, CALL COMANL TO GET THE NEXT COMMAND LINE, TURN ON DEFAULT PROCESSING, AND RETURN THE FIRST KEYWORD ENTRY IN THE NEW COMMAND LINE
= A$RSET, RESET THE COMMAND LINE POINTER TO THE BEGINNING OF THE COMMAND LINE AND TURN ON DEFAULT PROCESSING. USE OF THIS KEY DOES NOT RETURN A KEYWORD ENTRY.
= A$RAWI, RETURN THE REMAINDER OF THE COMMAND LINE AS RAW TEXT AND TURN ON THE END OF LINE INDICATOR. TEXT STARTS AT THE TOKEN FOLLOWING THE OPTION (IF PRESENT) OF THE LAST KEYWORD ENTRY READ.
= A$NKWL, TURN ON DEFAULT PROCESSING AND RETURN THE NEXT KEYWORD ENTRY IN THE COMMAND LINE. THIS KEY ALLOWS THE CALLING PROGRAM TO SWITCH KEYWORD LISTS IN THE MIDDLE OF A COMMAND LINE.

KWLIST = A ONE DIMENSIONAL ARRAY CONTAINING CONTROL INFORMATION, A TABLE OF KEYWORD ENTRY DESCRIPTIONS, AND A LIST OF DEFAULT KEYWORDS. SEE SECTION TITLED KWLIST FORMAT FOR A COMPLETE DESCRIPTION.

KWINDX = KEYWORD INDEX, RETURNED INTEGER VARIABLE IDENTIFYING THE KEYWORD IN A KEYWORD ENTRY, POSSIBLE VALUES ARE:

< 0, UNRECOGNIZED KEYWORD OR CMDL$A WAS CALLED WITH A KEY OF A$RSET OR A$RAWI.
= 0, END OF LINE.
> 0, VALID KEYWORD.

OPTBUF = PACKED ARRAY THAT NORMALLY CONTAINS THE TEXT OF A KEYWORD OPTION, HOWEVER IF AN UNRECOGNIZED KEYWORD IS ENCOUNTERED OPTBUF CONTAINS THE TEXT OF THAT KEYWORD.

BUFLEN = SPECIFIED LENGTH OF OPTBUF IN CHARACTERS, MUST BE .GE. ZERO.

OPTION = OPTION TYPE, RETURNED INTEGER VARIABLE THAT DESCRIBES THE OPTION FOLLOWING A KEYWORD, POSSIBLE VALUES ARE:

= A$NONE, NO OPTION, OR OPTION WAS NULL, OPTBUF WILL BE

BLANK.
= A$NAME, OPTION WAS A NAME
= A$NUMB, OPTION WAS A NUMBER, RESULTS OF NUMERIC CONVERSION RETURNED IN VALUE.
= A$NOVF, OPTION WAS A NUMBER AND CONVERSION RESULTED IN OVERFLOW (DECIMAL NUMBERS ONLY).

VALUE = RETURNED INTEGER*4 VARIABLE EQUAL TO THE BINARY VALUE OF AN OPTION IF IT WAS A NUMBER, ZERO OTHERWISE.

KWINFO = AN ARRAY THAT RETURNS MISCELLANEOUS INFORMATION AND MUST BE DIMENSIONED KWINFO(10) IN THE CALLING PROGRAM. KWINFO(1) IS EQUAL TO THE NUMBER OF CHARACTERS IN OPTBUF AND KWINFO(2) - KWINFO(10) ARE RESERVED FOR FUTURE USE.

FUNCTION

CMDL$A WAS DESIGNED TO SIMPLIFY THE PROCESSING OF A PRIMOS TYPE COMMAND LINE WHILE, AT THE SAME TIME, PROVIDING THE USER WITH A GREAT DEAL OF FLEXIBILITY IN DEFINING HIS COMMAND ENVIRONMENT.

THIS ROUTINE WILL PARSE A COMMAND LINE, A KEYWORD ENTRY AT A TIME, AND RETURN INFORMATION ABOUT EACH EACH ENTRY IT ENCOUNTERS. A KEYWORD ENTRY IS DEFINED AS A -KEYWORD FOLLOWED BY AN OPTION. A DEFAULT KEYWORD ENTRY IS DEFINED AS AN OPTION THAT IS NOT PRECEDED BY A -KEYWORD BUT, BY VIRTUE OF ITS POSITION IN THE COMMAND LINE, IMPLIES A SPECIFIED -KEYWORD (EG. FTN SNARF, WHERE SNARF IMPLIES THE DEFAULT KEYWORD -INPUT). DEFAULTS MAY ONLY OCCUR AT THE BEGINNING OF A COMMAND LINE.

CMDL$A RETURNS THE FOLLOWING INFORMATION FOR EACH KEYWORD ENTRY IN THE COMMAND LINE:

1) INTEGER THAT IDENTIFIES THE -KEYWORD (KWINDX).
2) TEXT OF THE KEYWORD OPTION, IF PRESENT (OPTBUF).
3) OPTION TYPE (OPTION).
4) RESULTS OF NUMERIC CONVERSION, IF OPTION WAS A NUMBER (VALUE).
5) NUMBER OF CHARACTERS IN THE TEXT OF AN OPTION (KWINFO(1)).

NOTE THAT CMDL$A DOES NOT PERFORM ANY ACTION OTHER THAN RETURNING INFORMATION ABOUT THE COMMAND LINE.

THE FOLLOWING IS A LIST OF CONSIDERATIONS THAT SHOULD BE TAKEN INTO ACCOUNT WHEN DEFINING A COMMAND ENVIRONMENT:

1) A KEYWORD MAY HAVE, AT MOST, ONE OPTION FOLLOWING IT.
2) A KEYWORD MUST BEGIN WITH A '-'.
3) A KEYWORD MAY NOT BE A DECIMAL NUMBER (EG. -99).
4) REGISTER SETTING PARAMETERS ARE NOT RECOGNIZED AS SUCH.
5) DEFAULT KEYWORDS ARE ONLY ALLOWED AT THE BEGINNING OF A COMMAND LINE. THE FIRST -KEYWORD ENCOUNTERED TURNS OFF DEFAULT PROCESSING AND ALL REMAINING OPTIONS ON THE COMMAND LINE MUST BE PRECEDED BY A -KEYWORD (THIS RESTRICTION CAN BE CIRCUMVENTED BY USING A KEY OF A$NKWL, HOWEVER THE USER MUST BE AWARE OF THE FACT THAT WHEN DEFAULT PROCESSING IS IN EFFECT EACH OPTION IS TREATED AS IF IT WERE PRECEDED BY A -KEYWORD).
6) A KEY OF A$RAWI (OR AN OPTION TYPE OF A$RAWI) WILL TURN ON THE END OF LINE INDICATOR AND ANY FURTHER ATTEMPTS TO READ FROM THE CURRENT COMMAND LINE WILL RETURN AN END OF LINE CONDITION. TO TURN OFF THE END OF LINE INDICATOR CMDL$A MUST BE CALLED WITH A KEY OF A$RSET OR A$NEXT.
7) A BUFFER LENGTH THAT IS TO SMALL TO CONTAIN THE TEXT OF AN OPTION WILL CAUSE THAT OPTION TO BE TRUNCATED AND AN ERROR MESSAGE TO BE DISPLAYED.
8) DEFAULT KEYWORD ENTRIES THAT HAVE A NUMERIC OPTION SHOULD BE AVOIDED AS PRIMOS MAY INTERCEPT THEM AS REGISTER SETTINGS.
9) A NEGATIVE HEXADECIMAL OPTION THAT CONSISTS OF ONLY

ALPHABETIC CHARACTERS (EG. -FF) WILL ALWAYS BE INTERPRETED AS A -KEYWORD.

10) KEYWORD ENTRIES IN THE KEYWORD TABLE WITH THE SAME KEYWORD INDICIES ARE CONSIDERED SYNONYMS. A KEYWORD MAY HAVE ANY NUMBER OF SYNONYMS, EACH HAVING DIFFERENT OPTION SPECIFICATIONS. HOWEVER, IF A KEYWORD WITH SYNONYMS IS ALSO A DEFAULT AND DEFAULT PROCESSING IS IN EFFECT, THE OPTION SPECIFICATIONS FOR THE SYNONYMS WILL BE IGNORED (IE. A DEFAULT KEYWORD OPTION ALWAYS IMPLIES THE FIRST KEYWORD IN A SYNONYM CHAIN).

11) NULL ENTRIES IN THE COMMAND LINE ARE ONLY PERMITTED FOR KEYWORDS THAT HAVE AN OPTION STATUS OF A$OPTL, ALL OTHER NULL ENTRIES WILL BE TREATED AS EITHER A MISSING OPTION OR AN UNRECOGNIZED KEYWORD.

12) CALLS TO CMDL$A AND RDTK$$ ON THE SAME COMMAND LINE SHOULD BE AVOIDED, AS CMDL$A USES RDTK$$ TO PERFORM A LOOK-AHEAD WHEN A -KEYWORD IS ENCOUNTERED.

13) ALL TEXT IS FORCED TO UPPER CASE UNLESS ENCLOSED IN QUOTES OR READ AS RAW TEXT (A$RAWI).

ALL ARGUMENTS ARE INTEGER*2 EXCEPT VALUE, WHICH IS INTEGER*4, AND OPTBUF WHOSE TYPE DOES NOT MATTER.

APPLIB CALLS:

CNVA$A, CNVB$A, CSUB$A, FILL$A, JSTR$A, MSUB$A, MSTR$A, NLEN$A, TYPE$A

KWLIST FORMAT

THE KWLIST ARRAY CONSISTS OF THREE SECTIONS, THE FIRST SECTION CONTAINS CONTROL INFORMATION, THE SECOND CONTAINS THE KEYWORD ENTRY TABLE, AND THE THIRD CONTAINS THE DEFAULT LIST.

CONTROL INFORMATION:

WORD 1 - NUMBER OF KEYWORD ENTRIES IN TABLE, MUST BE .GT. ZERO.

WORD 2 - MAXIMUM LENGTH OF KEYWORD TEXT IN CHARACTERS, MUST BE .GE. 2 AND .LE. 80. ALL KEYWORDS MUST HAVE THE SAME LENGTH THEREFORE IT MAY BE NECESSARY TO PAD THEM WITH BLANKS.

KEYWORD TABLE:

WORDS 1 TO N - TEXT OF KEYWORD, THE ACTUAL NUMBER OF CHARACTERS MUST BE EQUAL TO THE MAXIMUM KEYWORD LENGTH.

WORD N+1 - KEYWORD INDEX, MUST BE .GT. ZERO.

WORD N+2 - MINIMUM NUMBER OF CHARACTERS IN THE KEYWORD TO MATCH, MUST BE .GE. 2 AND .LE. MAXIMUM KEYWORD LENGTH. A VALUE THAT IS ZERO OR NEGATIVE CAUSES THE KEYWORD TO BE IGNORED WHEN THE TABLE IS SEARCHED. THIS ALLOWS KEYWORD TEXT TO EXIST AS DOCUMENTATION.

WORD N+3 - OPTION STATUS, POSSIBLE VALUES ARE:
A$NONE, NO OPTION MAY FOLLOW KEYWORD
A$OPTL, OPTION MAY OR MAY NOT FOLLOW KEYWORD
A$REQD, OPTION MUST FOLLOW KEYWORD.

WORD N+4 - OPTION TYPE, POSSIBLE VALUES ARE:
A$NONE, IF STATUS IS A$NONE
A$DEC, OPTION MUST BE A DECIMAL NUMBER
A$OCT, OPTION MUST BE AN OCTAL NUMBER
A$HEX, OPTION MUST BE A HEXADECIMAL NUMBER
A$NAME, OPTION MUST BE A NAME
A$NDEC, OPTION MAY BE A NAME OR A DECIMAL NUMBER
A$NOCT, OPTION MAY BE A NAME OR AN OCTAL NUMBER
A$NHEX, OPTION MAY BE A NAME OR A HEXADECIMAL NUMBER (IF THE OPTION CONSISTS OF ALL ALPHABETIC CHARACTERS, EG. FACE, THAT ALSO CONSTITUTE A VALID HEXADECIMAL NUMBER THEN IT WILL BE INTERPRETED AS SUCH)
A$RAWI, OPTION IS THE REMAINDER OF THE COMMAND LINE AFTER THE CURRENT -KEYWORD READ AS RAW TEXT. USE OF THIS OPTION TYPE WILL TURN ON THE END OF LINE INDICATOR IN THE SAME MANNER AS A KEY OF A$RAWI.

DEFAULT LIST:

WORD 1 - NUMBER OF DEFAULT KEYWORDS, MUST BE .GE. ZERO
WORDS 2 TO N - (WHERE N IS EQUAL TO WORD 1) LIST OF KEYWORD INDICIES PREVIOUSLY DEFINED IN THE KEYWORD ENTRY TABLE, THAT WILL BE USED WHEN DEFAULT PROCESSING IS IN EFFECT. A DEFAULT KEYWORD ENTRY MAY NOT HAVE AN OPTION STATUS OF A$NONE.

ERROR MESSAGES

THE FUNCTION VALUE WILL BE FALSE IF ANY OF THE FOLLOWING ERRORS OCCUR:

BAD KEY.
BUFFER LENGTH LESS THAN ZERO.
NAME TO LONG. (NAME TEXT)
UNRECOGNIZED KEYWORD. (KEYWORD TEXT)
BAD KEYWORD OPTION. (OPTION TEXT)
MISSING KEYWORD OPTION.
NO. OF KEYWORD ENTRIES MUST BE .GT. ZERO.
MAX KEYWORD LENGTH MUST BE .GE. 2 AND .LE. 80.
1ST CHARACTER OF KEYWORD MUST BE '-'. (KEYWORD TEXT)
KEYWORD MAY NOT BE A NUMBER. (KEYWORD TEXT)
KEYWORD INDEX MUST BE .GT. ZERO. (KEYWORD TEXT)
MIN CHARACTERS TO MATCH MUST BE .LE. MAX KEYWORD LENGTH. (KEYWORD TEXT)
INVALID OPTION STATUS. (KEYWORD TEXT)
INVALID OPTION TYPE. (KEYWORD TEXT)
NO. OF DEFAULTS MUST BE .GE. ZERO.
DEFAULT NOT DEFINED IN KEYWORD LIST. (DEFAULT INDEX)
INVALID DEFAULT OPTION STATUS. (KEYWORD TEXT)
MIN CHARACTERS TO MATCH MUST BE .GE. 2. (KEYWORD TEXT)
UNDETERMINED ERROR. (TEXT OF LAST KEYWORD OR OPTION READ)

SAMPLE PROGRAM

```
C       TEST PROGRAM FOR CMDL$A
C
      IMPLICIT INTEGER*2 (A-Z)
      INTEGER*4 VALUE
      DIMENSION BUFFER(10),KWLIST(128),INFO(10)
$INSERT SYSCOM>A$KEYS
C
      DATA KWLIST /11,14,
     * '*ANY TEXT      ',1,0,A$REQD,A$DEC,
     * '-NDECIMAL      ',2,2,A$OPTL,A$NDEC,
     * '-OCTAL         ',4,2,A$REQD,A$NONE,
     * '-NOCTAL        ',4,3,A$OPTL,A$NOCT,
     * '-HEXADECIMAL   ',5,2,A$REQD,A$HEX,
     * '-NHEXADECIMAL  ',6,3,A$OPTL,A$NHEX,
     * '-NAME          ',7,5,A$REQD,A$NAME,
     * '-MAYBE         ',8,6,A$OPTL,A$NAME,
     * '-NONE          ',9,5,A$NONE,A$NONE,
     * '-QUIT          ',10,2,A$NONE,A$NONE,
     * '-TITLE         ',99,2,A$OPTL,A$RAWI,
     * 4,1,2,8,7/
C
C
      BUFLEN = 20
      KEY = A$READ
10    IF (CMDL$A(KEY,KWLIST,KWINDX,BUFFER,BUFLEN,TYPE,VALUE,INFO))
     *    GO TO 15
          PRINT 99
99        FORMAT(/'TRY AGAIN, TURKEY !')
          CALL EXIT
15    IF (KWINDX.EQ.10) CALL EXIT
      IF (KWINDX.NE.A$NONE) GO TO 20
          KEY = A$NEXT
          GO TO 10
20    KEY = A$READ
      PRINT 100 BUFFER,KWINDX,TYPE,VALUE,INFO(1)
100   FORMAT(/10A2/'KWINDX    TYPE    VALUE      CHARS'/2X,4(I3,6X))
      GO TO 10
      END
```

5 SUMMARY AND KEYS

BELOW IS A BRIEF SUMMARY OF THE CALLING SEQUENCES FOR ALL THE APPLIB ROUTINES AND A LISTING OF THE FILE SYSCOM>A$KEYS.

5.1 SUMMARY

IN THE SUMMARY THAT FOLLOWS, THE TYPE CODES ARE DEFINED AS:

L = LOGICAL
I = INTEGER (SUBJECT TO COMPILE TIME OPTION)
I*2 = INTEGER*2
R = REAL
DP = DOUBLE PRECISION

GROUP	NAME	TYPE	ARGUMENTS
FILE SYSTEM	TEMP$A	L	(TYPKEY,NAME,NAMLEN,UNIT)
	OPEN$A	L	(OPNKEY+TYPKEY,NAME,NAMLEN,UNIT)
	OPNP$A	L	(MSG,MSGLEN,OPNKEY+TYPKEY,NAME,NAMLEN, UNIT)
	OPNV$A	L	(OPNKEY+TYPKEY,NAME,NAMLEN,UNIT,VERKEY, WTIME,RETRYS)
	OPVP$A	L	(MSG,MSGLEN,OPNKEY+TYPKEY,NAME,NAMLEN, UNIT,VERKEY,WTIME,RETRYS)
	CLOS$A	L	(UNIT)
	RWND$A	L	(UNIT)
	GEND$A	L	(UNIT)
	TRNC$A	L	(UNIT)
	DELE$A	L	(NAME,NAMLEN)
	EXST$A	L	(NAME,NAMLEN)
	UNIT$A	L	(UNIT)
	RPOS$A	L	(UNIT,POS)
	POSN$A	L	(POSKEY,UNIT,POS)
	TSCN$A	L	(KEY,UNITS,ENTRY,MAXSIZ,ENTSIZ,MAXLEV, LEV,CODE)
STRING	FILL$A	I	(NAME,NAMLEN,CHAR)
	NLEN$A	I*2	(NAME,NAMLEN)
	MCHR$A	I	(TARRAY,TCHAR,FARRAY,FCHAR)
	GCHR$A	I	(FARRAY,FCHAR)
	TREE$A	I	(NAME,NAMLEN,FSTART,FLEN)
	TYPE$A	L	(KEY,STRING,LENGTH)
	MSTR$A	I	(A,ALEN,B,BLEN)
	MSUB$A	I	(A,ALEN,AFC,ALC,B,BLEN,BFC,BLC)
	CSTR$A	L	(A,ALEN,B,BLEN)
	CSUB$A	L	(A,ALEN,AFC,ALC,B,BLEN,BFC,BLC)
	LSTR$A	L	(A,ALEN,B,BLEN,FCP,LCP)
	LSUB$A	L	(A,ALEN,AFC,ALC,B,BLEN,BFC,BLC,FCP,LCP)
	JSTR$A	L	(KEY,STRING,LENGTH)
USER QUERY	YSNO$A	L	(MSG,MSGLEN,DEFKEY)
	RNAM$A	L	(MSG,MSGLEN,NAMKEY,NAME,NAMLEN)
	RNUM$A	L	(MSG,MSGLEN,NUMKEY,VALUE)
INFORMATION	TIME$A	DP	(TIME)

	CTIM$A	DP	(CPUTIM)
	DTIM$A	DP	(DSKTIM)
	DATE$A	DP	(DATE)
	EDAT$A	DP	(EDATE)
	DOFY$A	DP	(DOFY)
MATHEMATICAL	RNDI$A	DP	(SEED)
	RAND$A	DP	(SEED)
CONVERSION	ENCD$A	L	(ARRAY,WIDTH,DEC,VALUE)
	CNVA$A	L	(NUMKEY,NAME,NAMLEN,VALUE)
	CNVB$A	I	(NUMKEY,VALUE,NAME,NAMLEN)
PARSING	CMDL$A	L	(KEY,KWLIST,KWINDX,OPTBUF,BUFLEN,OPTION, VALUE,KWINFO)

5.2 SYSCOM>A$KEYS

FUNCTION DECLARATIONS (TABSET 6 17)

```
 LOGICAL    CLOS$A,RWND$A,GEND$A,TRNC$A,DELE$A,RPOS$A,POSN$A,
X           TEMP$A,OPEN$A,OPNV$A,OPNP$A,OPVP$A,ENCD$A,YSNO$A,
X           RNAM$A,RNUM$A,TREE$A,EXST$A,UNIT$A,CNVA$A,CMDL$A,
X           CSUB$A,CSTR$A,TYPE$A,TSCN$A,JSTR$A,LSUB$A,LSTR$A
 INTEGER    MCHR$A,GCHR$A,FILL$A
 INTEGER*2  NLEN$A,MSUB$A,MSTR$A,CNVB$A
 REAL   *8  DOFY$A,DATE$A,EDAT$A,TIME$A,CTIM$A,DTIM$A,RNDI$A,RAND$A
```

KEY DECLARATIONS (TABSET 6 17)

```
 INTEGER*2  A$READ,A$WRIT,A$RDWR,A$SAMF,A$DAMF,A$NVER,A$VNEW,
X           A$OVAP,A$VOLD,A$ABS ,A$REL ,A$DEC ,A$OCT ,A$HEX ,
X           A$NDEF,A$DNO ,A$DYES,A$FUPP,A$UPLW,A$RAWI,
X           A$NONE,A$OPTL,A$REQD,A$NDEC,A$NOCT,A$NHEX,A$NAME,
X           A$NUMB,A$NEXT,A$RSET,A$NOVF,A$NKWL,A$TREE,A$DLAY,
X           A$NUFD,A$NSEG,A$CUFD,A$DECZ,A$DECU,A$OCTZ,A$HEXZ,
X           A$RGHT,A$LEFT ,A$BACK

 PARAMETER
X
X /*************************************************************/
X /*                                                           */
X /*                                                           */
X /*       KEY DEFINITIONS (TABSET 6 11 28 69)                 */
X /*                                                           */
X /*                                                           */
X /********************** OPEN$A **********************        */
X /********************** OPNP$A **********************        */
X /********************** OPNV$A **********************        */
X /********************** OPVP$A **********************        */
X /********************** TEMP$A **********************        */
X /*                   ****** OPNKEY ******                    */
X     A$READ = 1,         /* READ                              */
X     A$WRIT = 2,         /* WRITE                             */
X     A$RDWR = 3,         /* READ/WRITE                        */
X /*                   ****** TYPKEY ******                    */
X     A$SAMF = 0,         /* OPEN NEW SAM FILE                 */
X     A$DAMF = :2000,     /* OPEN NEW DAM FILE                 */
X /*                   ****** VERKEY ******                    */
X     A$NVER = 1,         /* NO VERIFICATION                   */
X     A$VNEW = 2,         /* VERIFY NEW FILE (OK TO MODIFY)    */
X     A$OVAP = 3,         /* A$VNEW + OVERWRITE/APPEND OPTION  */
X     A$VOLD = 4,         /* VERIFY OLD FILE (DO NOT CREATE NEW) */
X /*                                                           */
X /********************** RPOS$A **********************        */
X /*                   ****** POSKEY ******                    */
X     A$ABS  = 1,         /* ABSOLUTE POSITION                 */
X     A$REL  = 2,         /* RELATIVE POSITION                 */
```

```
X /*                                                                  */
X /********************* YSNO$A **********************                */
X /*                  ****** DEFKEY ******                            */
X     A$NDEF = -1,       /* NO DEFAULT                               */
X     A$DNO  = 0,        /* DEFAULT = 'NO'                           */
X     A$DYES = 1,        /* DEFAULT = 'YES'                          */
X /*                                                                  */
X /********************* RNUM$A **********************                */
X /********************* CNVA$A **********************                */
X /*                  ****** NUMKEY ******                            */
X     A$DEC  = 1,        /* DECIMAL                                  */
X     A$OCT  = 2,        /* OCTAL                                    */
X     A$HEX  = 3,        /* HEXADECIMAL                              */
X /*                                                                  */
X /*                                                                  */
X /********************* CNVB$A **********************                */
X /*                  ****** NUMKEY ******                            */
X /*  A$DEC  = 1,       /* DECIMAL,LEFT PADDED WITH BLANKS          */
X /*  A$OCT  = 2,       /* OCTAL, LEFT PADDED WITH BLANKS           */
X /*  A$HEX  = 3,       /* HEXADECIMAL, LEFT PADDED WITH BLANKS     */
X     A$DECZ = 4,        /* DECIAML, LEFT PADDED WITH ZEROS          */
X     A$OCTZ = 5,        /* OCTAL, LEFT PADDED WITH ZEROS            */
X     A$HEXZ = 6,        /* HEXADECIMAL, LEFT PADDED WITH ZEROS      */
X     A$DECU = 7,        /* UNSIGNED DECIMAL, LEFT PADDED WITH       */
X /*                         BLANKS                                   */
X /*                                                                  */
X /*                                                                  */
X /********************* CMDL$A **********************                */
X /*                  ****** KEY    ******                            */
X /*  A$READ = 1,       /* READ NEXT ENTRY IN COMMAND LINE          */
X     A$NEXT = 2,        /* READ FIRST ENTRY IN NEXT LINE            */
X     A$RSET = 3,        /* RESET TO BEGINNING OF COMMAND LINE       */
X /*  A$RAWI = 4,       /* READ REMAINDER OF LINE AS RAW TEXT       */
X     A$NKWL = 5,        /* ACCEPT NEW KEYWORD LIST                  */
X /*                  ****** OPTYPE ******                            */
X /*  A$DEC  = 1,       /* DECIMAL OPTION                           */
X /*  A$OCT  = 2,       /* OCTAL OPTION                             */
X /*  A$HEX  = 3,       /* HEXADECIMAL OPTION                       */
X /*  A$RAWI = 4,       /* OPTION IS RAW TEXT                       */
X     A$NDEC = 5,        /* NAME OR DECIMAL OPTION                   */
X     A$NOCT = 6,        /* NAME OR OCTAL OPTION                     */
X     A$NHEX = 7,        /* NAME OR HEXADECIMAL                      */
X     A$NAME = 8,        /* NAME                                     */
X /*                  ****** OPTION ******                            */
X     A$NONE = 0,        /* NO OPTION PRESENT OR NULL OPTION         */
X /*  A$NAME = 8,       /* OPTION IS A NAME                         */
X     A$NUMB = 9,        /* OPTION IS A NUMBER (DIGIT STRING)        */
X     A$NOVF = 10,       /* NUMERIC OVERFLOW                         */
X /*                  ****** STATUS ******                            */
X /*  A$NONE = 0,       /* NO OPTION TO FOLLOW KEYWORD              */
X     A$OPTL = 1,        /* OPTION MAY OR MAY NOT FOLLOW KEYWORD     */
X     A$REQD = 2,        /* OPTION MUST FOLLOW KEYWORD               */
X /*                                                                  */
X /********************* RNAM$A **********************                */
```

```
X /*                  ****** NAMKEY ******                              */
X     A$FUPP = 1,       /* FORCE UPPER CASE                            */
X     A$UPLW = 2,       /* READ UPPER AND LOWER CASE                   */
X     A$RAWI = 4,       /* READ REST OF LINE                           */
X /*                                                                    */
X /*                                                                    */
X /********************* TSCN$A **********************                 */
X /*                  ****** KEY    ******                              */
X     A$TREE = 1,       /* ALL ENTRIES IN A TREE                       */
X     A$NUFD = 2,       /* DO NOT SCAN SUBUFDS                         */
X     A$NSEG = 3,       /* DO NOT SCAN SEGDIRS                         */
X     A$CUFD = 4,       /* DO NOT SCAN SUBUFDS OR SEGDIRS              */
X     A$DLAY = 5,       /* STAY AT DIRECTORY WHEN GOING UP TREE        */
X     A$BACK = 6,       /* BACK UP ONE LEVEL (FOR ERROR HANDLING)      */
X /*                                                                    */
X /********************* JSTR$A **********************                 */
X /*                  ****** KEY    ******                              */
X     A$RGHT = 1,       /* RIGHT JUSTIFY                               */
X     A$LEFT = 2        /* LEFT JUSTIFY                                */
X /*                                                                    */
X /*                                                                    */
X /*****************************************************************/
```

SUBJECT: CHANGES FOR CX REVISION 16

THE CHANGES TO CX FOR REVISION 16 ARE:

A. CX NOW RUNS MULTIPLE JOB STREAMS,
B. CX NOW HANDLES JOB PRIORITIES, AND
C. CPU TIME LIMITS ARE NOW SUPPORTED.

MULTIPLE JOB STREAMS REFERS TO ABILITY OF THE CX TO RUN MORE THAN ONE JOB AT A TIME, DYNAMICALLY SPAWNING PHANTOMS AS IT NEEDS TO USE THEM. THERE IS A LIMIT OF 64 SLAVES TO THE CX MASTER, A LIMIT WHICH CANNOT BE REACHED UNDER PRIMOS IV AND V SINCE THESE OPERATING SYSTEMS DO NOT YET SUPPORT MORE THAN 63 USERS. HOWEVER, IF CX IS RUNNING ON AN OLD PARTITION DISK, THE MAXIMIMUM NUMBER OF STREAMS IS 4.

THE FACILITY TO ASSIGN A CX JOB A PRIORITY LEVEL IS IMPLEMENTED IN TWO PLACES; THE CX MONITOR ITSELF, AND THE OPERATING SYSTEM. THIS PRIORITY LEVEL IS THE MAJOR FACTOR IN DETERMINING WHICH CX JOB IS TO BE EXECUTED NEXT, AND IT ALSO AFFECTS THE SCHEDULING OF THE CX JOB WHILE IT IS RUNNING.

CX JOBS CAN ALSO HAVE A CPU TIME LIMIT ON HOW LONG THEY CAN RUN, AND WILL BE LOGGED OUT IF AND WHEN THEY REACH THAT LIMIT. THE LIMIT IS IN CPU SECONDS, AND THEREFORE NOT RELATED TO HOW LONG THE JOB TAKES TO RUN IN WALL CLOCK TIME. THIS LIMIT IS ENFORCED BY THE PRIMOS OPERATING SYSTEM.

I. USER VISIBLE CHANGES

A. MULTIPLE STREAMS

MULTIPLE STREAMS IS A FEATURE THAT IS AUTOMATIC DEPENDING ON HOW THE SYSTEM MANAGER CONFIGURES CX, THAT IS, THE USER DOES NOT NEED TO DO ANYTHING NEW TO USE THE MULTIPLE STREAMS FEATURE. AS A MATTER OF FACT, THE USER CAN DO RELATIVELY LITTLE TO CONTROL THIS FEATURE, WHICH CAN PRESENT DANGERS TO USERS THAT HAVE USED CX IN THE PAST.

THE MAJOR PROBLEM ASSOCIATED WITH MULTIPLE STREAMS IS THAT TWO OR MORE JOBS SUBMITTED BY THE SAME USER MAY NOW RUN AT ONCE - AT REVISION 15, THIS WAS IMPOSSIBLE. THEREFORE, SOME USERS MAY BE SUBMITTING MULTIPLE JOBS AT ONCE WHICH CAN NOT RUN AT THE SAME TIME FOR VARIOUS REASONS, INCLUDING COMOUTPUTING TO THE SAME FILE, USING THE SAME MAGNETIC TAPE UNIT (OR FOR THAT MATTER, PAPER-TAPE, CARD READER UNIT, ETC.), OR HAVING ONE JOB COMPILE AND THE OTHER JOB LOAD THE BINARY FILES PRODUCED BY THE FIRST JOB.

USERS WHO HAVE BEEN DOING THIS IN THE PAST MUST NOW USE OTHER METHODS; FOR INSTANCE, EACH JOB COULD HAVE ITS OWN COMOUTPUT FILE; OR, WHEN ONE JOB IS JUST ABOUT FINISHED, IT COULD SUBMIT THE NEXT JOB TO BE RUN; THIS HAS ADDED ADVANTAGES, ONE OF WHICH IS THAT IF A JOB ABORTS, NO

MORE JOBS IN THE CHAIN WILL RUN (POSSIBLY PREVENTING IMPORTANT FILES FROM BLOWING UP).

THE USER CAN NOW EASILY DETERMINE THE PROCESS NUMBER THAT HIS JOB(S) IS(ARE) RUNNING ON, USING ONE OF THE CX STATUS COMMANDS (-A, -SNN, -Q OR -P); THE NUMBER AFTER THE PRIORITY COLUMN (THE LAST COLUMN IN THE HEADER LINE) IS THE PROCESS NUMBER. IF THERE IS NO NUMBER THERE, THEN THAT JOB IS EITHER STILL WAITING, NO LONGER RUNNING, OR DROPPED. IN THE CASE OF THE -SNN OPTION, THE PROCESS NUMBER IS THE NUMBER IN PARENTHESES AFTER THE PRIORITY LEVEL. AGAIN, IF THERE IS NO NUMBER IN PARENTHESES, THEN THAT JOB IS NOT EXECUTING.

B. JOB PRIORITIES

INDIVIDUAL JOBS IN THE CX MONITOR NOW HAVE INDIVIDUAL PRIORITIES ASSIGNED BY THE USER WHO SUBMITTED THE JOBS. THE PRIORITY LEVEL IS TAKEN INTO ACCOUNT IN TWO PLACES; WHEN CX LOOKS FOR A WAITING JOB TO RUN, AND WHEN THAT JOB IS ACTUALLY RUN. IN THE FIRST PLACE, ANY JOB WILL NEVER BE STARTED UP IF THERE IS ANOTHER JOB WAITING TO EXECUTE WITH A HIGHER PRIORITY; THIS IS TOTALLY INDEPENDENT OF WHEN THE JOBS WERE SUBMITTED. DATE AND TIME OF SUBMITTAL IS ONLY TAKEN INTO CONSIDERATION WHEN MULTIPLE JOBS WITH THE SAME PRIORITY LEVEL ARE IN THE WAIT QUEUE.

ALSO, WHEN THE JOB IS RUN, ITS PRIORITY DETERMINES THE SCHEDULAR QUEUE THAT THE JOB WILL RUN IN, THE SAME QUEUE THAT IS AFFECTED BY THE OPERATOR'S CHAP COMMAND. THE ALGORITHM TO DETERMINE WHAT QUEUE IT WILL BE IN IS COMPLEX; FIRST, IT DEPENDS ON THE QUEUE THAT THE CX MONITOR IS RUNNING IN. ALL SLAVES SPAWNED BY CX WILL RUN IN THAT QUEUE, NOT NECESSARILY IN THE DEFAULT QUEUE (1). THIS IS A REV. 16 OPERATING SYSTEM CHANGE. ALSO, ANY PROCESS CAN LOWER ITS QUEUE LEVEL WITH THE NEW CHAP LOWER COMMAND. THIS COMMAND IS EXECUTED BY CX SLAVE PHANTOMS WHEN THEY RUN A USER'S JOB.

TO DETERMINE HOW MUCH A SLAVE WILL LOWER ITS QUEUE, THE USER SUBMITTAL PROGRAM SUBTRACTS THE CX PRIORITY OF THE JOB FROM A VALUE CALLED THE "MEDIAN PRIORITY", AND PUTS THE RESULTING NUMBER ON THE COMMAND LINE AFTER THE TEXT "CHAP LOWER ". THE MEDIAN PRIORITY IS A NUMBER WHICH REPRESENTS THE LOWEST PRIORITY A CX JOB CAN HAVE AND STILL RUN IN THE SAME QUEUE AS THE CX MONITOR. ANY JOBS WITH A HIGHER PRIORITY THAN THE MEDIAN PRIORITY WILL ALSO RUN IN THE SAME QUEUE, SINCE THE CHAP LOWER COMMAND CANNOT BE USED TO RAISE THE QUEUE OF THE JOB EXECUTING THE COMMAND. IT ALSO CANNOT LOWER IT BELOW QUEUE 0.

THE STANDARD VALUE FOR MEDIAN PRIORITY IS 3; HOWEVER, IT IS PER-INSTALLATION CONFIGURABLE, SO CHECK WITH THE SYSTEM MANAGER TO DETERMINE WHAT THE MEDIAN PRIORITY IS ON THE SYSTEM.

ANOTHER PER-INSTALLATION CONFIGURABLE VALUE IS THE DEFAULT PRIORITY, I.E. THE PRIORITY ASSIGNED TO A CX JOB WHEN THE USER HAS NOT SPECIFIED A VALUE. THE STANDARD IS 3.

TO SPECIFY THE PRIORITY LEVEL FOR A JOB, APPEND THE OPTION -PRIORITY FOLLOWED BY THE PRIORITY LEVEL TO THE COMMAND LINE, I.E. AFTER THE TREENAME. THE STANDARD LIMITS TO THE PRIORITY LEVEL ARE FROM 0 TO 7; HOWEVER, THE SYSTEM MANAGER MAY LIMIT IT TO ANYTHING HE DESIRES, ALTHOUGH CX AS DISTRIBUTED WILL NOT SUPPORT ANY VALUES HIGHER THAN 99 OR LOWER THAN 0.

AN EXAMPLE COMMAND LINE TO SUBMIT THE CX FILE CX_COBOL WITH A PRIORITY OF 2 IS:

```
CX CX_COBOL -PRIORITY 2
```

THE OPTION -PRIORITY MAY BE ABBREVIATED TO -PRIO. IF, AS AN EXAMPLE, THE MEDIAN PRIORITY ON THIS SYSTEM IS 5, AND THE CX MONITOR IS RUNNING IN QUEUE 3, THE FOLLOWING EVENTS WILL TAKE PLACE; FIRST, THE CX PROGRAM WILL SUBTRACT THE PRIORITY OF THE JOB (2) FROM THE MEDIAN PRIORITY (5) PRODUCING 3 AS THE RESULT, AND PUT THE COMMAND CHAP LOWER 3 AT THE TOP OF THE COMMAND FILE WHEN IT COPIES IT OVER TO THE CX UFD.

THEN, WHEN THE JOB IS RUN, THE CX SLAVE PHANTOM, WHICH WAS STARTED UP AND IS STILL RUNNING IN QUEUE 3, WILL EXECUTE THAT COMMAND FILE. WHEN THE CHAP LOWER 3 COMMAND IS EXECUTED, THE PHANTOM WILL THEN BE IN QUEUE 0, THE LOWEST QUEUE ON THE SYSTEM. THEREFORE, THE USER'S JOB WILL BE EXECUTED ENTIRELY IN QUEUE 0.

C. CPU TIME LIMITS

THE CAPABILITY TO LIMIT PARTICULAR CX JOBS TO A CERTAIN AMOUNT OF CPU TIME NOW EXISTS AT REVISION 16. THIS LIMIT IS PASSED ONTO THE OPERATING SYSTEM, I.E. CX DOES NOT TAKE THIS LIMIT INTO CONSIDERATION WHEN LOOKING FOR A JOB TO EXECUTE.

THE UNIT OF TIME IS THE CPU SECOND; THAT IS, THE ACTUAL AMOUNT OF TIME THAT THE JOB HAS BEEN RUNNING IN SECONDS. ON A LIGHTLY LOADED SYSTEM, 30 CPU SECONDS CAN BECOME 1 WALL CLOCK MINUTE. ON A SYSTEM WITH A HEAVIER LOAD, IT CAN BECOME 3 OR 4 MINUTES.

WHEN THIS LIMIT IS REACHED, THE MESSAGE CPU TIME LIMIT EXCEEDED WILL BE OUTPUT TO THE COMOUTPUT FILE (IF THERE IS ONE), AND THE PROCESS WILL BE LOGGED OUT. CX WILL FLAG THIS STATUS AS "ABORTED".

TO LIMIT A CX JOB, APPEND THE OPTION -CPULIMIT FOLLOWED BY EITHER THE NUMBER OF CPU SECONDS TO WHICH THE JOB IS TO BE LIMITED, OR THE STRING "NONE", TO THE COMMAND LINE, I.E.:

```
CX CX_COBOL -CPULIMIT 500
```

OR:

```
CX CX_COBOL -CPULIMIT NONE
```

THE NUMBER FOLLOWING THE -CPULIM OPTION IS READ AS AN INTEGER*4 NUMBER,

BECAUSE THE OPERATING SYSTEM WILL SUPPORT AN INTEGER*4 TIME LIMIT.

THE OPTIONS -PRIORITY AND -CPULIMIT CAN BOTH BE PRESENT ON THE COMMAND LINE, IN ANY ORDER, BUT BOTH OF THEM MUST FOLLOW THE TREE NAME. THE KEYWORD -CPULIMIT MAY BE ABBREVIATED TO -CPULIM. A CPU LIMIT OF 0 IS ILLEGAL.

NOTE THAT THIS VALUE IS RELATIVE TO THE CURRENT STATUS; IN OTHER WORDS, IF A PROCESS HAD BEEN LOGGED IN 7 MINUTES AND HAD USED 13 CPU SECONDS, THEN LIMITED ITS CPU TIME TO 50 SECONDS, THEN THE PROCESS WOULD BE LOGGED OUT AFTER IT HAD CONSUMED A TOTAL OF 63 CPU SECONDS. THEREFORE, A CX JOB WITH A LIMIT OF 500 SECONDS WILL GET ALMOST EXACTLY THAT, AND IT WON'T HAVE TO PAY FOR THE CPU TIME CONSUMED BY THE SLAVE WHILE IT WAS LOOKING FOR WORK ("SLAVE LABOR").

IF THE -CPULIMIT OPTION IS NOT INCLUDED ON THE CX COMMAND LINE, A PER-INSTALLATION CONFIGURABLE DEFAULT WILL BE USED. THE STANDARD IS AN INFINITE AMOUNT OF CPU TIME, I.E. "NONE".

II. SYSTEM MANAGER NOTES

A. MULTIPLE STREAMS

IN THE CX UFD, CALLED 'CX***', THERE IS A FILE NAMED PH_GO. THIS FILE IS THE START-UP FILE FOR CX, AND THE REV. 16 OPERATING SYSTEM COMMAND PHANTOM CX***>PH_GO WILL CAUSE THE CX MONITOR TO START UP.

THE LAST EXECUTABLE LINE OF THE FILE IS THE LINE:

```
RESUME *MASTER 1/1 2/1
```

WHERE PARAMETER 1 SPECIFIES THE MINIMUM NUMBER OF PHANTOMS TO RUN, AND PARAMETER 2 SPECIFIES THE MAXIMUM NUMBER OF PHANTOMS TO START UP. IN OTHER WORDS, THE ABOVE LINE TELLS THE CX MONITOR THAT IT SHOULD ALWAYS HAVE ONE PHANTOM (SLAVE) RUNNING, NO MORE, AND NO LESS, WHETHER IT HAS WORK TO DO OR NOT.

THIS CONFIGURATION WILL CAUSE CX TO ACT IN THE SAME WAY THAT IT DID AT REV. 15, AND USERS WILL NOT HAVE TO WORRY ABOUT TWO OF THEIR JOBS RUNNING AT ONCE IN THIS CASE.

HOWEVER, TO USE THE MULTIPLE STREAMS FEATURE, DEFINE THE SECOND PARAMETER AS THE MAXIMUM NUMBER OF STREAMS THAT CX IS TO RUN, IRREGARDLESS OF HOW MANY JOBS ARE IN THE QUEUE. DEFINE THE FIRST PARAMETER AS THE MINIMUM NUMBER OF SLAVES TO HAVE READY FOR JOBS, IN EFFECT "RESERVING" THOSE PHANTOM SLOTS FOR CX. AN EXAMPLE LINE TO LET CX RUN UP TO 22 STREAMS AT ONCE BUT RESERVE ONLY 7 IS:

```
RESUME *MASTER 1/7 2/26
```

NOTE THAT THE PARAMETER DATA MUST BE OCTAL. A SUBSEQUENT FEATURE OF THIS IS THAT CX CAN BE CONFIGURED TO NOT HAVE TO RUN ANY SLAVES AT ALL, I.E.:

RESUME *MASTER 1/0 2/6

WILL ALLOW 6 STREAMS TO RUN SIMULTANEOUSLY, BUT WHEN THE CX MONITOR HAS NO WORK TO DO, THE ENTIRE CX SUBSYSTEM WILL RUN ONLY ONE PHANTOM. IN THE EARLIER EXAMPLE, WHERE IT WAS TOLD TO RUN A MINIMUM OF 7 SLAVES, THERE WOULD BE A TOTAL OF 8 PROCESSES IN USE BY THE CX SUBSYSTEM (CX AND THE 7 SLAVES).

IF CX CANNOT START UP AS MANY PHANTOMS AS IT WANTS TO, BECAUSE OF THE "NO FREE PHANTOM" ERROR, IT WILL NOT CRASH, BUT SIMPLY ACT AS THOUGH IT HAD ALREADY REACHED THE MAXIMUM CONFIGURED NUMBER OF SLAVES TO RUN. HOWEVER, IF IT CAN'T START UP THE MINIMUM NUMBER OF PHANTOMS DUE TO THIS ERROR, IT WILL GRIPE TO THE SYSTEM CONSOLE EVERY 10 MINUTES WITH THE MESSAGE "MINIMUM PHANTOMS NOT AVAILABLE". IT WILL STILL BE OPERATIVE, THOUGH (UNLESS, OF COURSE, IT CAN'T GET ANY PHANTOMS AT ALL, IN WHICH CASE IT IS EFFECTIVELY INOPERATIVE).

CX IS DESIGNED TO RECOVER AFTER A SYSTEM CRASH; IF A CRASH OCCURS, CX WILL ATTEMPT TO RESTART ANY JOB THAT WAS RUNNING AT THE TIME WHEN THE SYSTEM IS BROUGHT BACK UP.

IT IS POSSIBLE, HOWEVER, FOR CX TO BE UNABLE TO FUNCTION AFTER A SYSTEM CRASH; IN THAT CASE, TRY RUNNING THE PROGRAM *KILL IN CX*** AND RESTARTING CX. IF THAT STILL DOESN'T WORK, THEN RUN CX***>*INIT AND BRING CX BACK UP. ALL JOB DATA WILL BE LOST AFTER RUNNING *INIT.

IF THAT DOESN'T WORK, MAKE SURE THE FOLLOWING FILES EXIST AND LOOK REASONABLE IN THE CX*** UFD: PH_GO, P_SCAN, AND LOGOUT. THEN MAKE SURE THAT THE FOLLOWING RUNFILES EXIST IN CX***: *INIT, *KILL, *MASTER, AND *SLAVE. THEN MAKE SURE THAT THE COMMAND CX -A PRODUCES THE ERROR MESSAGE "?CAN'T - JOB FILE EMPTY", OR SOME OTHER MESSAGE INDICATING THAT IT IS FUNCTIONING.

THEN, USE FUTIL TO UFDPROTECT THE ENTIRE CX*** UFD TO 7 0, THEN TELL IT TO CLEAN CX## AND THEN CLEAN PH_#. WHEN THIS IS DONE, CX SHOULD BE ABLE TO BE BROUGHT UP.

B. JOB PRIORITIES

THERE ARE SEVERAL VALUES THE SYSTEM MANAGER MUST BE CONCERNED WITH WHEN IT COMES TO CX PRIORITIES. THESE VALUES ARE:

MEDIAN PRIORITY (STANDARD 3) MEDPRI
DEFAULT PRIORITY (STANDARD 3) PRIO
MAXIMUM PRIORITY (STANDARD 7) MAXPRI
MASTER QUEUE LEVEL (STANDARD 1)
SLAVE QUEUE LEVEL (STANDARD 1)

CX WILL NOT SUPPORT A MAXIMUM PRIORITY HIGHER THAN 99, NOR WILL IT SUPPORT EITHER A MEDIAN PRIORITY OR A DEFAULT PRIORITY THAT IS LESS THAN ZERO OR GREATER THAN THE MAXIMUM PRIORITY, I.E. IF THE MAXIMUM PRIORITY IS LEFT STANDARD, THE DEFAULT PRIORITY MUST NOT BE SET TO 8 OR -1.

THE MASTER QUEUE LEVEL IS THE QUEUE IN WHICH THE CX MONITOR RUNS. THIS DIRECTLY AFFECTS THE SLAVE QUEUE LEVEL; WHENEVER A SLAVE IS SPAWNED, IT INHERITS THE QUEUE LEVEL THAT ITS FATHER (THE CX MONITOR) HAD. HOWEVER, THIS MEANS THAT IF THE CX MONITOR IS SUPPOSED TO BE IN QUEUE 3 FOR INSTANCE, BUT BEFORE THE SYSTEM CONSOLE IS USED TO CHAP IT UP TO 3 IT SPAWNS A PHANTOM OR TWO, ANY PHANTOMS THAT IT SPAWNED WILL REMAIN IN QUEUE 1 EVEN THOUGH THE CX MONITOR AND ANY NEW PHANTOMS IT SPAWNS WILL RUN IN QUEUE 3.

THEREFORE, THE CX MONITOR MUST BE CHAPED TO THE DESIRED LEVEL BEFORE IT SPAWNS ANY PHANTOMS. TO FACILITATE THIS, CX WILL NOT SPAWN ANY PHANTOMS OR SEARCH THE CX QUEUE FOR 40 SECONDS AFTER IT STARTS UP. THE CHAP COMMAND CAN BE INCLUDED IN THE COMMAND FILE THAT STARTS CX UP AS LONG AS THE PROCESS NUMBER THAT CX WILL RUN IN REMAINS CONSTANT AND IS KNOWN.

THE DEFAULT PRIORITY IS THE PRIORITY ASSIGNED TO A CX JOB IF THE USER DOES NOT EXPLICITLY ASSIGN ONE HIMSELF. THIS SHOULD GENERALLY BE THE MIDDLE-OF-THE-ROAD PRIORITY FOR YOUR SYSTEM.

THE MEDIAN PRIORITY REPRESENTS THE LOWEST PRIORITY A JOB CAN HAVE TO RUN IN THE SAME QUEUE THAT THE CX MONITOR HAS. TO DETERMINE THIS NUMBER, FIRST YOU MUST DECIDE THE FOLLOWING: WHAT RUN QUEUE A CX JOB SHOULD RUN IN IF THE USER DID NOT SPECIFY ANY PRIORITY. CALL THIS NUMBER THE DEFAULT QUEUE LEVEL. THEN YOU MUST DECIDE THE HIGHEST QUEUE LEVEL YOU WILL LET ANY CX JOB RUN IN. CALL THIS THE MAXIMUM QUEUE LEVEL.

SUBTRACT THE DEFAULT QUEUE LEVEL FROM THE MAXIMUM QUEUE LEVEL, THEN ADD THE DEFAULT PRIORITY, AND THE RESULT IS THE MEDIAN PRIORITY THAT YOU WANT. MODIFY MEDPRI, PRIO AND MAXPRI IN THE CX USER PROGRAM AND RE-COMPILE. THEN, CHAP THE CX MONITOR UP TO THE MAXIMUM QUEUE LEVEL WHENEVER YOU BRING IT UP, AND THE SYSTEM IS IN PLACE. EXAMPLE:

THE HIGHEST QUEUE LEVEL YOU EVER WANT A CX JOB TO HAVE IS 3. HOWEVER, IF THE USER DOESN'T SPECIFY A PRIORITY, YOU WANT HIS JOB TO RUN AT QUEUE LEVEL 0. YOU HAVE ALSO DECIDED THAT THE DEFAULT PRIORITY SHOULD BE 2. SO YOU SUBTRACT 0 FROM 3 (HIGHEST QUEUE LEVEL) AND THEN ADD THE DEFAULT PRIORITY (2) TO PRODUCE A MEDIAN PRIORITY OF 5. SO YOU MAKE THE APPROPRIATE CHANGES TO THE USER PROGRAM IN CX AND RECOMPILE THE SUBSYSTEM.

PRIORITIES THAT ARE GIVEN TO JOBS THAT ARE HIGHER THAN THE MEDIAN PRIORITY WILL RUN IN THE SAME QUEUE AS IF THEY WERE GIVEN THE MEDIAN PRIORITY, AND ANY JOB WITH A PRIORITY LESS THAN THE MEDIAN PRIORITY MINUS THE MAXIMUM QUEUE LEVEL WILL RUN IN THE SAME QUEUE AS IT WOULD HAVE IF IT WERE GIVEN THAT VALUE. THE DIFFERENCE IS THAT CX USES PRIORITIES AS THE MAJOR FACTOR IN DECIDING WHICH JOBS TO RUN NEXT;

THEREFORE, A JOB WITH PRIORITY 7 WILL ALWAYS RUN BEFORE A JOB WITH PRIORITY 6, EVEN THOUGH THE MEDIAN PRIORITY MAY BE 5 AND THEY WOULD BOTH RUN IN THE SAME QUEUE.

C. CPU TIME LIMITS

THE ONLY VALUE YOU NEED TO BE CONCERNED ABOUT AS A SYSTEM MANAGER HERE IS THE DEFAULT CPU TIME LIMIT. THE STANDARD IS NONE (I.E. INFINITE AMOUNT OF CPU TIME FOR A CX JOB), BUT YOU CAN SET IT TO ANY VALUE YOU WANT. SIMPLY CHANGE THE PARAMETER CPULIM IN THE DATA STATEMENT IN THE CX USER PROGRAM, RECOMPILE AND INSTALL IN YOUR SYSTEM. THE PARAMETER CPULIM IS AN INTEGER*4 PARAMETER, AND IT REPRESENTS THE NUMBER OF CPU SECONDS THE JOB WILL BE ALLOWED TO HAVE. NOTE THAT USERS MAY CIRCUMVENT THIS DEFAULT ENTIRELY BY ALWAYS SUBMITTING JOBS WITH -CPULIM NONE ON THE COMMAND LINE.

THERE IS A WAY FOR YOU TO CAUSE ALL JOBS LOGGING IN TO HAVE A CERTAIN CPU TIME LIMIT OR CONNECT TIME LIMIT THAT CAN'T BE GOTTEN AROUND; CHECK THE DOCUMENTATION ON THE NEW LIMIT$ CALL TO PRIMOS, AND CONSIDER PUTTING IT IN YOUR EXTERNAL LOGIN PROGRAM.

IF A CX JOB DOES RUN OUT OF TIME, IT WILL BE LOGGED OUT, BUT THE "PHANTOM TTY REQUEST" MESSAGE WON'T BE PRINTED ON THE SYSTEM CONSOLE. HOWEVER, CX WILL DETECT THAT THE JOB HAS TERMINATED WITHOUT A CX -E AND NOTIFY THE SYSTEM CONSOLE, FLAGGING "ABORTED" STATUS ON THE JOB.

D. TRIVIA

THERE ARE FIVE DIFFERENT MESSAGES THAT CX CAN SEND TO THE SYSTEM CONSOLE:

```
*CX*    CX MONITOR, REV 16.0
*CX*    MINIMUM PHANTOMS NOT AVAILABLE.
*CX*    EXECUTING FILENM FOR USER USRNAM (NN).
*CX*    JOB FILENM USER USRNAM (NN) COMPLETED.
*CX*    JOB FILENM USER USRNAM (NN) ABORTED.
```

THE FIRST MESSAGE IS SENT BY THE MASTER PROGRAM WHEN THE CX MONITOR STARTS UP, AFTER THE 40-SECOND DELAY. THE SECOND MESSAGE IS SENT WHENEVER CX IS UNABLE TO HAVE THE MINIMUM NUMBER OF SLAVES SPAWNED, ALSO BY THE MASTER PROGRAM. THE THIRD IS SENT BY THE MASTER PROGRAM WHENEVER IT STARTS UP A CX JOB; FILENM REFERS TO THE COMMAND FILE NAME (I.E. CX##NN), USRNAM REFERS TO THE USERNAME OF THE SUBMITTER, AND NN REFERS TO THE PROCESS NUMBER OF THE SLAVE THAT IS TO EXECUTE THE JOB. THE FOURTH AND FIFTH MESSAGES ARE VARIATIONS ON THE THIRD, EXCEPT THAT THE FOURTH IS SENT BY THE CX USER PROGRAM WHEN IT EXECUTES A CX -E.

SOME BUG FIXES INCLUDE FIXING THE PROBLEM THAT OCCURED WHEN TWO USERS TRIED TO SUBMIT A CX JOB AT ONCE (FILE IN USE TO ONE OF THEM), NOW THE USER THAT WOULD HAVE GOTTEN THE FILE IN USE MESSAGE JUST GETS A HIGHER

CX QUEUE NUMBER. ALSO, CX NOW ATTEMPTS TO RETURN THE USER TO HIS HOME UFD IN MORE CASES THAT IT CAN. HOWEVER, MOST OF THE BUGS IN CX REVISION 15 HAVE BEEN FIXED BY REWRITING THE CODE IN THE MASTER AND SLAVE PROGRAMS TO ALLOW FOR MULTIPLE STREAMS; AN EXAMPLE IS THE "BUG" THAT IF A CX SLAVE WENT DOWN, THE MASTER WOULD NEVER NOTICE IT, BUT WOULD KEEP SENDING JOBS TO IT AND FLAGGING THEM AS "ABORTED" 30 SECONDS LATER. ALSO, IT WOULD TRY TO DELETE THE CX##NN FILE 30 SECONDS AFTER IT WROTE THE BOOTSTRAP FILE FOR THE SLAVE TO PICK UP (NOW CALLED PH_#NN, AT REV. 15 IT WAS C_PHFL), AND SINCE THE SLAVE SLEPT 30 SECONDS BETWEEN CHECKING FOR THE BOOTSTRAP FILE, SOMETIMES THE MASTER PROGRAM WOULD DELETE THE CX##NN FILE BEFORE THE SLAVE EVER SAW IT.

III. CX INTERNALS

THE CX USER PROGRAM RESIDES IN CMDNCO; HOWEVER, EVERYTHING ELSE THAT CX USES RESIDES IN CX***. THE FILES THAT ARE NEEDED TO RUN THE CX SUBSYSTEM ARE:

*MASTER - THE CX MASTER PROGRAM
*SLAVE - THE CX SLAVE PROGRAM
*INIT - THE CX SUBSYSTEM INITIALIZER
*KILL - THE CX SLAVE ACTIVITY FILE INITIALIZER
PH_GO - THE COMMAND FILE TO START UP THE CX SUBSYSTEM
P_SCAN - THE COMMAND FILE USED BY *MASTER TO SPAWN SLAVES
LOGOUT - THE COMMAND FILE INVOKED BY THE USER PROGRAM TO DO CX -E

RUNNING *INIT WILL PRODUCE TWO MORE FILES THAT MUST STAY INTACT ONCE THEY ARE CREATED:

JOBS*T - CONTAINS INFORMATION ON ALL USER'S JOBS
USER#S - CONTAINS INFORMATION ON AL SLAVE ACTIVITY

ALSO, WHENEVER A USER SUBMITS A CX JOB, IT IS GIVEN THE NAME:

CX##NN - WHERE NN IS THE CX JOB NUMBER OF THAT JOB

AND WHEN CX DECIDES TO RUN IT, IT CREATES A FILE CALLED:

PH_#NN - WHERE NN IS THE PROCESS NUMBER OF THE TARGET SLAVE

WHEN THE SLAVE (WHICH WAS SPAWNED OFF OF P_SCAN WHICH RAN *SLAVE) SEES ITS PH_#NN FILE, IT COMINPUTS INTO IT, AND THAT FILE COMINPUTS INTO THE CX##NN FILE THAT IS TO BE RUN. THAT FILE, WHICH WAS GENERATED BY THE CX USER PROGRAM, STARTS THE SLAVE BACK UP AGAIN SO IT CAN DELETE THE PH_#NN FILE (WHICH WAS GENERATED BY THE MASTER PROGRAM). THEN THE SLAVE EXITS FOR THE LAST TIME, LETTING CX##NN TAKE CONTROL.

WHAT CX##NN THEN DOES IS LIMIT THE JOB'S CPU TIME AND LOWER ITS QUEUE LEVEL APPROPRIATELY, THEN IT DOES WHAT THE USER'S COMMAND FILE WAS GOING TO DO, EXCEPT THAT IF IT ABORTS (OR LEAVES THE CX##NN FILE FOR ANY OTHER REASON SUCH AS A CLOSE ALL OR COMINPUT SOME_OTHER_FILE ON THE SAME UNIT), THE MASTER IS THEN ABLE TO DELETE CX##NN AND FLAG "ABORTED"

STATUS ON THE JOB. IF THE COMMAND FILE IS RUN TO COMPLETION, CX -E IS EXECUTED, AND THE USER PROGRAM DELETES THE CX##NN FILE, FLAGS "COMPLETED" STATUS ON THE JOB, AND THEN COMINPUTS TO CX***>LOGOUT.

WHILE THE SLAVES ARE LOOKING FOR WORK (IN THE FORM OF PH_#NN), THEY CONSTANTLY (EVERY 30 SECONDS) UPDATE AND LOOK AT AN ENTRY SPECIFIC TO THEMSELVES IN THE USER#S FILE. IF THEIR ENTRY IS ZERO, THEY COMINPUT INTO CX***>LOGOUT. THE WHOLE FILE IS ZEROED BY *INIT OR *KILL, SO RUNNING EITHER OF THOSE PROGRAMS CAUSES ALL SLAVES TO LOG THEMSELVES OUT. HOWEVER, IF THE MASTER PROGRAM IS RUNNING, IT WILL LOG SOME MORE BACK IN AGAIN (UNLESS THE MINIMUM NUMBER OF SLAVES IS SET TO ZERO OR IT CAN'T SPAWN THE SLAVES FOR SOME OTHER REASON). *KILL WILL NOT WIPE OUT THE JOB DATA FILE, HOWEVER, AND IS THEREFORE PREFERRED.

THE MASTER PROGRAM HANDLES ALL OF THE SCHEDULING OF CX JOBS, ALL OF THE SLAVE SPAWNING, AND ALL OF THE CRASH RECOVERY PROCEDURES, INCLUDING RECOVERING FROM ITSELF BEING LOGGED OUT, LEAVING OTHER SLAVES RUNNING. IT DOES NOT HANDLE THE CPU LIMIT PARAMETER, AND THE ONLY TIME IT REFERS TO A CX JOB'S PRIORITY IS WHEN IT IS DECIDING WHICH JOB TO SCHEDULE NEXT. THERFFORE, IF THE CX QUEUE CONTROL FILE JOBS*T IS CHANGED AFTER A JOB IS SUBMITTED SO THAT THE PRIORITY OR CPU TIME LIMIT PARAMETERS OF THAT JOB ARE CHANGED, THE ONLY EFFECT THAT CHANGE WILL HAVE IS IF THE CX PRIORITY IS DIFFERENT WHEN THE CX MONITOR LOOKS AT IT. THE PRIORITY AT WHICH THE JOB WILL RUN, AND THE CPU LIMIT OF THE JOB, WILL REMAIN AS BEFORE, SINCE THAT INFORMATION IS IN THE CX COMMAND FILE FOR THAT JOB.

AS MENTIONED EARLIER, EACH SLAVE UPDATES ITS ENTRY IN USER#S EVERY 30 SECONDS; WHAT IT UPDATES IT WITH IS THE TIME OF DAY IN MINUTES PLUS 1 (SO THAT ZERO WILL NOT OCCUR).

IF THE MASTER SEES A SLAVE REGISTERED IN THE USER#S FILE THAT HAS NOT UPDATED ITS ENTRY IN THE LAST 4 MINUTES, IT WILL NULLIFY THAT ENTRY. IF A SLAVE EVER FINDS ITS ENTRY NULLIFIED AFTER IT WRITES IT INTO THE USER#S FILE WHEN IT FIRST RUNS, IT WILL LOG OUT WITH NO COMPLAINTS (I.E. COMINPUT INTO CX***>LOGOUT). THIS METHOD GENERALLY GUARANTEES THE MASTER PROGRAM TO BE IN CLOSE AND CONSTANT TOUCH WITH ITS SLAVES.

WHEN A CX -E IS EXECUTED, AND THE USER PROGRAM FINDS THAT ITS ENTRY IN USER#S IS ZEROED (AS A RESULT OF RUNNING *INIT OR *KILL), IT WILL BOMB OUT WITH A BAD USER#S ERROR, BUT THE STATUS OF THE JOB IT WAS RUNNING WILL BE "COMPLETED", BECAUSE THE JOB WAS, AFTER ALL, COMPLETE.

ON NEW PARTITION DISKS, THE READ-WRITE LOCK ON JOBS*T WILL BE 2, AS SET BY *INIT, SO THAT USERS WON'T HAVE TO WAIT TO DO A CX -A OR SOME OTHER STATUS COMMAND. WHEN THE MASTER PROGRAM RUNS, IT SETS THE READ-WRITE LOCK ON USER#S TO 3 SO THAT MANY SLAVES WON'T END UP FIGHTING FOR THE RIGHT TO WRITE THE FILE EVERY 30 SECONDS; IT WAS DISCOVERED THAT AFTER A CERTAIN NUMBER OF SLAVES WERE RUNNING, SOME WOULD NOT GET A CHANCE TO WRITE THE USER#S FILE FOR AS MUCH AS 5 MINUTES IF ITS READ-WRITE LOCK WAS 2 OR LESS.

IF THE MASTER PROGRAM CAN'T SET THE READ-WRITE LOCK DUE TO AN OLD PARTITION ERROR (E$OLDP), IT WILL LIMIT THE MAXIMUM NUMBER OF STREAMS TO 4.

IV. COMPATIBILITY

ON THE USER LEVEL, CX REVISION 16 IS ENTIRELY COMPATIBLE WITH CX REVISION 15 ASSUMING THAT CERTAIN DEFAULT VALUES ARE LEFT STANDARD. HOWEVER, NO CX REVISION 15 PROGRAM IS IN ANY WAY COMPATIBLE WITH CX REVISION 16 DATA FILES, OR VICE VERSA, DUE TO THE CHANGED STRUCTURE OF THE DATA FILES JOBS*T AND USER#S, ESPECIALLY SINCE AT REVISION 15 THESE FILES WERE CALLED JOBS* AND USER#, SO IT IS IMPORTANT TO INSTALL ALL OF CX AT ONCE.

THE COMMAND FILES AND PROGRAMS ARE DISTRIBUTED IN SUCH A WAY THAT THE ACTIONS TAKEN WILL BE COMPATIBLE; FOR INSTANCE, THE MASTER PROGRAM IS TOLD TO RUN EXACTLY ONE SLAVE IN THE RELEASED PH GO FILE, JUST AS REVISION 15 DID. ALSO, CPU LIMITS ARE DEFAULTED TO INFINITE AS A STANDARD, AND THE VALUES MAXPRI AND MEDPRI ARE BOTH SET TO 3, CAUSING CX JOBS TO DEFAULT TO THE SAME QUEUE LEVEL THAT THEY WERE RUNNING AT AT REVISION 15.

SUBJECT: CHANGES FOR SPOOL REVISION 16

THE CHANGES TO SPOOL FOR REVISION 16 ARE MINOR. THEY CONSIST OF ONE BUG FIX IN THE SPOOLER PHANTOM AND A CHANGE TO THE LIBRARY SUBROUTINES AND SPOOL TO ALLOW FOR THE REV. 16 OPERATING SYSTEM ABILITY TO USE 63 UNITS INSTEAD OF JUST 16.

THE BUG FIX THAT WENT INTO THE SPOOLER PHANTOM IS THAT IT NO LONGER BOMBS OUT WITH PRO NOT ASSIGNED WHEN IT ATTEMPTS TO PRINT AN EMPTY FILE ON A CENTRONICS PRINTER OR A PLOTTER.

THE -TUNIT AND -FUNIT OPTIONS IN THE SPOOL PROGRAM NOW ACCEPT UNIT NUMBERS RANGING FROM 1 TO 63 INSTEAD OF LIMITING THEM TO BETWEEN 1 AND 16. ALSO, THE SPOOL$ SUBROUTINE (IN BOTH R-MODE AND V-MODE) WILL ALSO ACCEPT UNIT NUMBER SPECIFICATIONS IN THE RANGE FROM 1 TO 63 IN THE INFO ARRAY.

IF THIS INCREASED RANGE IS ATTEMPTED ON AN OPERATING SYSTEM EARLIER THAN REV. 16, IT WILL CAUSE A BAD UNIT (E$BUNT) ERROR TO OCCUR. THIS WILL ONLY OCCUR WHEN THE LIBRARY ROUTINES ARE GIVEN UNIT NUMBERS GREATER THAN 16.

UBJECT: PRMPC FOR RELFASE 16.0.

UG FIXES:

PRMPC NO LONGER REQUIRES ITS NRD VARIABLE LINE COUNTER.
TAR #20193.

ABSTRACT

REVISION 16 OF PRIMOS IV HAS SEVERAL NEW FEATURES AND EXTENSIONS. AMONG THESE ARE THE INTRODUCTION OF TREENAMES TO INTERNAL COMMANDS, 63 FILE UNITS PER USER, THE ABILITY TO DYNAMICALLY OBTAIN A FILE UNIT, AND IMPROVEMENTS IN THE AREAS OF PROTECTION. SEVERAL PROBLEMS HAVE BEEN FIXED, AND THE TOOLS FOR BUILDING PRIMOS HAVE BEEN SIMPLIFIED AND IMPROVED. THIS DOCUMENT DESCRIBES THESE AND OTHER TOPICS RELATED TO REVISION 16 OF PRIMOS.

REVISION 16.2 CONTAINS SUBSTANTIAL NEW FUNCTIONALITY AS WELL AS ERROR CORRECTIONS. THE NETWORK PRIMITIVES HAVE BEEN CHANGED TO USE THE X.25 PROTOCOL. COMPUTERS USING REV 16.2 PRIMOS CANNOT BE NETWORKED WITH MACHINES USING EARLIER REVISIONS. DOCUMENTATION OF THE NETWORK CHANGES IS IN OTHER DOCUMENTS.

1 CONFIGURATION AND OPERATIONAL MODIFICATIONS

1.1 BUILDING PRIMOS IV

THE BASIC PROCEDURES FOR BUILDING PRIMOS IV HAVE BEEN SIMPLIFIED. THE ONLY COMMAND FILE WHICH MUST BE RUN TO BUILD PRIMOS IS C_ALL. IF IT IS NOT NECESSARY TO RECOMPILE (OR REASSEMBLE) ALL SOURCE MODULES, SIMPLY RUN THE COMMAND FILE C_LOAD.

THE RUN FILES OF PRIMOS ARE LEFT IN THE UFD NAMED PRI400. THE COMMAND FILE C_COPY (ALSO IN PRI400) IS PROVIDED TO COPY THE RUN FILES INTO PRIRUN. PRIMOS IS NOW STARTED UP BY ATTACHING TO PRIRUN, AND TYPING R PRIMOS.

THE COMMAND FILE C_COLD HAS BEEN SIMPLIFIED TO USE A NEW VERSION OF MAPGEN. SEE SECTION 8 FOR COMPLETE DETAILS ON MAPGEN.

RUNNING C_LOAD WILL RESULT IN THE CREATION OF A NEW PRXXXX FILE -- PR0014. ALL DATABASES WHICH ARE INVOLVED WITH VIRTUAL MEMORY PAGING AND SEGMENTATION, THE TAPE DUMP PROGRAM, THE CRASH REGISTER SAVE AREA, AND SOME UTILITY CODE USED BY THE PAGING SYSTEM ARE CONTAINED IN SEGMENT 14. (FOR COMPLETE DETAILS ON SEGMENT 14, SEE SECTION 9.2.)

IMPORTANT NOTE: AT REVISION 16 OF PRIMOS, THE FORTRAN, KIDA (ALSO KNOWN AS MIDAS), COBOL, AND FORMS LIBRARIES, ED (IN CMDNCO), AND THE UII PACKAGE ARE SHARED BY DEFAULT.

1.1.1 C_PRMO_TEMPLATE

THE FOLLOWING IS A TEMPLATE THAT CAN BE USED BY A SITE TO CREATE THE COMMAND FILE C_PRMO. C_PRMO IS THE COMMAND FILE THAT LIVES IN CMDNCO (OF LOGICAL DISK 0), AND IS USED TO BRING UP REVISION 16 OF PRIMOS.

THE TEMPLATE WHICH APPEARS BELOW IS INCOMPLETE, AND IS COMPLETED ON A PER SITE BASIS. FOR CONVENIENCE, A COPY OF THIS TEMPLATE CALLED C_PRMO.TEMPLATE IS IN THE UFD PRIRUN. ONCE THE CHANGES HAVE BEEN MADE TO C_PRMO.TEMPLATE, SIMPLY FUTIL IT TO CMDNCO OF LOGICAL DISK 0 AS C_PRMO.

THE INFORMATION THAT MUST BE SUPPLIED IN THIS FILE IS AS FOLLOWS:

1) THE NAME OF THE CONFIG DATA FILE. THIS FILE SHOULD BE NAMED CONFIG (THE DEFACTO PRIME STANDARD NAME FOR THIS FILE).

2) THE LOCAL DISK(S) TO BE ADDED WHEN PRIMOS IS STARTED UP. (SOME SITES MAY NEED TO SPECIFY MORE THAN ONE ADDISK COMMAND IN THIS FILE.)

3) THE AMLC LINES AND THE SPEED AT WHICH THEY ARE TO BE SET TO WHEN PRIMOS COMES UP. (SOME SITES MAY NEED TO SPECIFY MORE THAN ONE AMLC COMMAND IN THIS FILE.)

IN ADDITION, A SITE SHOULD INCLUDE (AT THE END OF THIS FILE) ANY COMMANDS NECCESSARY TO BRING UP ANY SEPARATELY PRICED (OR OTHER) SOFTWARE WHEN PRIMOS IS BROUGHT UP (E.G. DBMS, NETWORKS, ETC.).

```
CONFIG -DATA                    /* SPECIFY CONFIG FILE AFTER -DATA
ADDISK                          /* SPECIFY LOCAL DISKS TO BE ADDED
AMLC TTY                        /* SPECIFY AMLC LINES
OPR 1                           /* SHARE REQUIRES OPR 1
SHARE SYSTEM>ED2000 2000        /* SHARE THE EDITOR - ED
SHARE SYSTEM>UI2000 2000        /* SHARE THE UII PACKEAGE
SHARE SYSTEM>S2014A 2014 700/* SHARE FORTRAN LIBRARY
SHARE SYSTEM>S2014B 2014 700
R SYSTEM>S4000
SHARE SYSTEM>K2014A 2014 700/* SHARE MIDAS LIBRARY
SHARE SYSTEM>K2014B 2014 700
R SYSTEM>K4000
SHARE SYSTEM>C2014A 2014 700/* SHARE COBOL LIBRARY
SHARE SYSTEM>C2014B 2014 700
R SYSTEM>C4000
SHARE SYSTEM>F2014A 2014 700/* SHARE FORMS LIBRARY
SHARE SYSTEM>F2014B 2014 700
R SYSTEM>F4000
SHARE 2014
OPR 0
PH CX***>PH_GO                  /* START 'CX' MONITOR
PH SPOOLQ>PH_PRO                /* START SPOOLER PHANTOM
A CMDNC0
/* SET THE DATE AND TIME **********
CO TTY
```

1.2 SINGLE VERSION PRIMOS IV

THE PRIMOS IV OPERATING SYSTEM HAS BEEN MODIFIED TO ALLOW A SINGLE VERSION OF THE SYSTEM TO BE CONFIGURABLE AT COLD-START TO RUN BETWEEN 1 AND 64 USERS. THIS NEW SYSTEM OBSOLETES THE 64 USER SYSTEM, THE 16 USER SYSTEM, AND THE LARGE ADDRESS SPACE 16 USER SYSTEM. IN THE NEW SINGLE VERSION SYSTEM, EACH USER MAY BE CONFIGURED TO HAVE ACCESS TO 32 M-BYTES (256 SEGMENTS) OF VIRTUAL ADDRESS SPACE, WITH A LIMIT OF 40 M-BYTES (320) SEGMENTS OF VIRTUAL ADDRESS SPACE FOR ALL USERS COMBINED. CONFIG DIRECTIVES ARE USED TO SPECIFY THE NUMBER OF USERS TO BE CONFIGURED, THE NUMBER OF SEGMENTS TO ALLOW EACH USER TO ACCESS, AND THE TOTAL NUMBER OF USER SEGMENTS AVAILABLE IN THE SYSTEM.

THE TOTAL NUMBER OF USERS TO BE CONFIGURED IS SPECIFIED BY THREE CONFIG DIRECTIVES: NTUSR (NUMBER OF TERMINAL USERS), NPUSR (NUMBER OF PHANTOM USERS), AND NRUSR (NUMBER OF REMOTE USERS). THE SUM OF THESE THREE VALUES MUST NOT EXCEED 64.

THE NUMBER OF SEGMENTS AVAILABLE TO EACH USER IS SPECIFIED BY A NEW CONFIG DIRECTIVE, NUSEG. (NEW CONFIG DIRECTIVES ARE DESCRIBED IN SECTION 7.) THIS DIRECTIVE IS USED TO SET THE SIZE OF EACH USER'S DESCRIPTOR TABLE FOR DTAR2, AND THUS, SPECIFIES THE NUMBER OF SEGMENTS EACH USER CAN REFERENCE. HOWEVER, THE SYSTEM HAS SPACE FOR A MAXIMUM OF 4096 SDW'S FOR ALL USERS. THEREFORE, THE USERS*NUSEG PRODUCT CANNOT EXCEED 4096.

THE NSEG DIRECTIVE SPECIFIES THE NUMBER OF SEGMENTS TO BE ALLOCATED FOR USE BY ALL USERS. IT SETS THE SIZE OF THE AREA TO BE USED BY THE SYSTEM FOR PAGE MAPS. THERE MAY BE FEWER PAGE MAPS AVAILABLE THAN THE NUMBER OF POSSIBLE USER SEGMENTS. THUS, ALTHOUGH A 64 USER SYSTEM CAN ALLOW 64 POSSIBLE SEGMENTS TO BE ADDRESSED BY EACH USER, THERE IS A LIMIT OF NSEG SEGMENTS WHICH CAN ACTUALLY BE IN USE BY ALL USERS AT ANY GIVEN TIME. THE SYSTEM ALLOCATES SPACE FOR A MAXIMUM OF 320 PAGE MAPS. THUS, NSEG CANNOT EXCEED 320.

THE FOLLOWING TABLE SHOWS THE CORRESPONDENCE BETWEEN THE PREVIOUS VERSIONS OF PRIMOS IV AND THE VALUES TO BE USED WITH THE NEW SINGLE VERSION SYSTEM TO GET THE SAME CONFIGURATION:

VERSION	NSEG	NUSEG	
64	192	32	DEFAULT
16	144	8	
16L	320	256	

1.3 NUMBER OF SEGMENTS, PAGING SPACE

THE NUMBER OF SEGMENTS REQUIRED BY PRIMOS IS GIVEN BY:

NSEG = N + 10 + USERSEGS

WHERE N IS THE TOTAL NUMBER OF CONFIGURED USERS AND USERSEGS IS THE TOTAL NUMBER OF SEGMENTS TO BE AVAILABLE TO USERS. IF IT IS DESIRED TO LIMIT NSEG TO A NUMBER LESS THAN 192, 144, OR 320 (TO CONSERVE PAGING SPACE, FOR EXAMPLE), THE NSEG, PAGDEV, AND ALTDEV CONFIGURATION DIRECTIVES CAN BE USED (SEE SECTION 7). IF NSEG IS NOT MODIFIED, USERSEGS DEFAULTS AS FOLLOWS:

USERSEGS = 118 = (192 - 64 - 10)

GIVEN USERSEGS FROM THE ABOVE, THE PAGING DISK SPACE REQUIREMENTS ARE GIVEN BY:

RECORDS = (64*USERSEGS + 8*N + 280) * RECORDS/PAGE

WHERE N IS AGAIN THE TOTAL NUMBER OF CONFIGURED USERS.

NOTE: IF IT IS DESIRED TO START WITH A SPECIFIED AMOUNT OF PRIMARY AND ALTERNATE PAGING SPACE, THE CALCULATION OF NSEG CAN BE PERFORMED AUTOMATICALLY BY USING THE <RECORDS> PARAMETER ON THE PAGDEV AND ALTDEV CONFIGURATION DIRECTIVES -- SEE SECTION 7.

1.4 PRIMOS IV DIRECTORY REORGANIZATION - PRI400

THE DIRECTORIES THAT CONTAIN PRIMOS IV SOURCE, OBJECT, AND RUNFILES HAVE BEEN UNIFIED INTO A SINGLE DIRECTORY. THIS HAS BEEN MADE POSSIBLE BY THE CREATION OF THE SINGLE VERSION OF PRIMOS IV WHICH ENABLES ONE SET OF OBJECT AND RUN FILES TO BE USED IN ALL CONFIGURATIONS.

THE DIRECTORY WHICH CONTAINS ALL OF PRIMOS IV IS NAMED PRI400. IT CONTAINS SUBDIRECTORIES FOR SOURCE, OBJECT, AND OTHER FILES. THE RUNFILE IMAGES ARE FOUND IN PRI400, ITSELF. ALL COMINPUT FILES FOR GENERATING PRIMOS IV ARE ALSO FOUND IN PRI400.

THE SOURCE AND OBJECT FOR PRIMOS IV IS BROKEN INTO 4 PARTS: KERNEL, FILE SYSTEM, NETWORK, AND COMMUNICATIONS. THE SOURCE FILES FOR THESE FOUR PARTS ARE FOUND IN THE SUBDIRECTORIES KS, FS, NS, AND CS, RESPECTIVELY. THE COMPILED (OR ASSEMBLED) OBJECT FILES ARE FOUND IN KO, FO, NO, AND CO.

1.4.1 PRI400>INSERT

ALL $INSERT FILES WHICH ARE USED IN COMPILING OR ASSEMBLING SOURCE PROGRAMS HAVE BEEN PLACED IN PRI400>INSERT. THUS, SOURCE PROGRAM STATEMENTS OF THE FORM

```
$INSERT DVMCOM
```

HAVE BEEN CHANGED TO

```
$INSERT *>INSERT>DVMCOM
```

IF PRIMOS SOURCE PROGRAMS ARE TO BE COMPILED OR ASSEMBLED IN SOME DIRECTORY OTHER THAN PRI400, A SUBDIRECTORY INSERT MUST EXIST IN THE PRESENT HOME DIRECTORY; IN THE SUB-UFD INSERT MUST BE ANY $INSERT FILES REQUIRED.

1.4.2 PRI400>UTILS

THE SOURCES FOR CERTAIN UTILITIES USED BY PRIMOS IV HAVE BEEN MOVED INTO "PRI400>UTILS". THESE INCLUDE "PRIMOS" (THE PRIMOS PRELOADER), MAPGEN (THE PAGE MAP AND COLD START IMAGE GENERATOR), AND THE VERSION OF VPSD THAT IS LOADED WITH PRIMOS IV FOR DEBUGGING PURPOSES (SEE SECTION 2.3 FOR COMPLETE DETAILS ON VPSD FOR KERNEL DEBUGGING). THE COMINPUT FILES FOR GENERATING THESE UTILITIES ARE FOUND IN PRI400.

1.5 RUNNING PRIMOS IV

THE DIRECTORIES PR4.64, PR4.16, PR4L16, PRINET>NR4.64, PRINET>NR4.16, AND PRINET>NR4L16 NO LONGER EXIST. THE PRIMOS IV SYSTEM IS NOW LOADED BY ATTACHING TO PRIRUN AND ISSUING THE COMMAND "R PRIMOS". FOR THOSE INSTALLATIONS SUPPORTING PRIMENET, SECTION 1.7 EXPLAINS HOW TO INSTALL NETWORKS.

1.6 CONFIGURATION MODIFICATIONS AND ADDITIONS

SEVERAL CHANGES HAVE BEEN MADE TO THE CONFIG DIRECTIVES WHICH ARE USED TO SPECIFY HOW PRIMOS WILL BE INITIALIZED. BELOW IS A LIST OF THE CONFIG DIRECTIVES THAT HAVE BEEN ADDED, DELETED, OR MODIFIED FOR REVISION 16 OF PRIMOS. FOR COMPLETE DETAILS, SEE SECTION 7.

FAM -- OBSOLETED & NOW ILLEGAL
FILUNT -- ADDED
MYNAME -- OBSOLETED & NOW ILLEGAL
NET -- MODIFIED
NSEG -- MODIFIED
NUSEG -- ADDED
RLOGIN -- OBSOLETED & NOW ILLEGAL

1.7 CONFIGURATION AND INSIALLAIION OF NETWORKS

AS THE SIZE AND COMPLEXITY OF PRIMENET NETWORKS EXPANDS, THE SYSTEM MANAGER'S TASK OF NETWORK CONFIGURATION GROWS INCREASINGLY MORE DIFFICULT. IN ORDER TO PROVIDE A FLEXIBLE AND SIMPLE INTERFACE FOR NETWORK CONFIGURATION, A TWO STEP PROCESS HAS BEEN INTRODUCED FOR REVISION 16. THE FIRST STEP OF THE NETWORK CONFIGURATION IS FOR THE SYSTEM MANAGER TO CREATE A NETWORK CONFIGURATION FILE USING THE SUPPLIED EXTERNAL COMMAND NETCFG. (SEE SEPARATE DOCUMENT FOR COMPLETE DETAILS ON NETCFG.) THIS PROGRAM WILL INTERACTIVELY GUIDE A SYSTEM ADMINISTRATOR THROUGH NODE, LINK, AND OPTION SPECIFICATIONS REQUIRED TO DESCRIBE A PRIMENET NETWORK. THE RESPONSES ARE VALIDATED AND WRITTEN INTO THE NETWORK CONFIGURATION FILE NETCON. AT PRIMOS IV INITIALIZATION NETCON (WHICH IS ASSUMED TO BE IN CMDNCO) IS OPENED AND THE INFORMATION PROCESSED.

TO INSTALL NETWORKS WITH REVISION 16 THE FOLLOWING PROCEDURE IS USED.

```
FUTIL
>F PRINET>CMDNCO
>I CMDNCO <PASSWORD>
>C NETCFG
>QU
```

NEXT, THE OBSOLETE CONFIG DIRECTIVES MYNAME, NET, FAM, RLOGIN MUST BE REMOVED FROM THE PRIMOS IV CONFIGURATION FILE, AND REPLACED WITH THE SINGLE CONFIG DIRECTIVE 'NET ON'. FINALLY THE COLD START NETWORK CONFIGURATION FILE MUST BE CREATED WITH THE FOLLOWING PROCEDURE:

```
OK, AT CMDNCO <PASSWORD>
OK, NETCFG
  <ANSWER QUESTIONS TO DESCRIBE YOUR NETWORK>
OK,
```

ONCE ALL QUESTIONS DESCRIBING THE NETWORK HAVE BEEN ANSWERED IN THE DIALOG WITH NETCFG, THE BINARY FILE NETCON WILL BE PLACED IN CMDNCO (ASSUMING ONE HAS ATTACHED THERE AS INDICATED ABOVE). NETCON WILL CONTAIN THE INFORMATION FORMERLY SUPPLIED BY THE CONFIG DIRECTIVES NET, FAM, MYNAME, AND RLOGIN. IN ADDITION, NOTE THAT 'CONFIG <MYNAME>' IS ALSO OBSOLETE. SPECIFICATION OF ANY OBSOLETE CONFIG DIRECTIVE RELATED TO NETWORKS WILL RESULT IN COLD START ERROR MESSAGE.

THE SMLC CONFIG DIRECTIVES ARE NOT RECOMMENDED WHEN CONFIGURING NETWORKS, AS THEY WILL DISABLE ALL SMLC MAPPING FROM THE CONFIGURATION FILE. THE SMLC DIRECTIVES ARE INTENDED FOR THOSE SITES THAT USE SMLC'S WITHOUT NETWORKS.

1.7.1 CONFIGURATION AND INSTALLATION OF FAM

THE SOURCE, OBJECT, RUN, AND COMMAND FILES FOR THE FILE ACCESS MANAGER FAM ARE CONTAINED IN THE DIRECTORY CHAIN PRINET>FAM. THE FILES IN THE UFD FAM THAT ARE OF SPECIAL IMPORTANCE TO THE FAM INSTALLER ARE AS FOLLOWS:

PH_FAM	PHANTOM COMMAND FILE
*FAM	RUN FILE
C_BLD	COMPILE AND LOAD
C_LOAD	LOAD FROM BINARIES

TO INSTALL THE FAM, THE FOLLOWING MUST BE DONE:

1) CREATE A UFD CALLED FAM (WHICH MAY BE LOGGED INTO). THIS UFD MUST NOT HAVE A PASSWORD.

2) FUTIL THE FILES PH_FAM AND *FAM TO THE NEWLY CREATED UFD.

TO ENABLE FAM, SIMPLY DO ONE OF THE FOLLOWING:

1) LOGIN UNDER THE USERNAME OF FAM:

```
OK, LOGIN FAM
FAM (XX) LOGGED IN AT ...
OK, R *FAM 1000
GO
```

FAM WILL NOW RUN, AND NO FURTHER COMMANDS WILL BE READ FROM THE TERMINAL.

2) RUN THE FAM AS A PHANTOM:

```
OK, A_FAM
OK, PH_PH_FAM
PHANTOM IS USER ...
OK,
```

TO ENABLE FAM TO COMMUNICATE WITH A PARTICULAR REMOTE NODE, SEE SEPARATE DOCUMENT DESCRIBING NETCFG. IF REMOTE NODES ARE NOT SPECIFIED PROPERLY WITH NETCFG, FAM WILL TERMINATE WITH THE MESSAGE '****ST 26' AT THE TERMINAL WHICH ENABLED FAM. THE MESSAGE 'FAMSTOP AT 000026' WILL BE PRINTED AT THE OPERATOR (USER 1) CONSOLE.

1.8 THE DISK BOOT

THE DISK BOOTSTRAP PROGRAM, BOOT, HAS BEEN CHANGED TO CATCH MORE ERRORS AND TO EITHER HALT WITH AN ERROR CODE IN THE CONTROL PANEL DATA LIGHTS OR TO PRINT AN ERROR MESSAGE. THE ADDITIONAL CHECKS INCLUDE RUNNING IN MACHINE CHECK MODE, CHECKING UP TO 64K WORDS OF MEMORY AND A MULTI-RECORD CONSISTENCY CHECK.

PRIOR TO REVISION 16 OF PRIMOS, THE BOOT PROGRAM FOR A STORAGE MODULE WAS CONTAINED IN A SINGLE 1040 WORD RECORD. AT REVISION 16, THE BOOT PROGRAM HAS BEEN MODIFIED TO USE MORE THAN ONE RECORD WHEN BOOTING OFF A STORAGE MODULE.

THE DISK BOOT, BOOT, OPERATES IN TWO STEPS. ONLY ONE RECORD IS READ IN BY THE CONTROL PANEL BOOT, CPBOOT. THIS RECORD IS ONLY A DISK INPUT ROUTINE THAT LOADS THE REST OF THE DISK BOOT. BOOT THEN INITIALIZES THE SYSTEM CONSOLE AND TYPES 'PHYSICAL DEVICE='. AFTER THE PHYSICAL DEVICE NUMBER IS TYPED BY THE USER, BOOT ATTEMPTS TO FIND AND LOAD THE APPROPRIATE DOS INTO MEMORY AND TRANSFERS CONTROL TO DOS.

1.8.1 ERRORS

ERRORS DETECTED WHILE LOADING BOOT USING ITS OWN FIRST RECORD WILL CAUSE A HALT WITH AN ERROR CODE IN THE CONTROL PANEL DATA LIGHTS. THE ERRORS CHECKED AND PUT INTO THE LIGHTS AT THIS STAGE WILL BE:

ERROR	OCTAL # IN LIGHTS
PARITY	100
MACHINE CHECK	101
NON-OCTAL PHYSICAL DEVICE NUMBER	102
BAD DEVICE TYPE	103
BAD STATUS OPTION B, B', STORAGE MODULE, DISKETTE	104
BAD RECORD ID - BAD CRA (HIGH-LOW)	105
INCOMPATIBLE BOOT RECORDS	106
'FILE' NOT FOUND	107
MEMORY TEST FAILURE	110

PARITY ERROR AND MACHINE CHECK ERROR, 100, 101

IF A PARITY OR MACHINE CHECK ERROR OCCURS WHILE LOADING THE BOOT PROGRAM ITSELF, THEN A HALT WILL OCCUR WITH THE CODE 100 OR 101 RESPECTIVELY IN THE CONTROL PANEL DATA LIGHTS. PARITY AND MACHINE CHECK ERRORS ARE CAUGHT BY THE HARDWARE. NO FURTHER INFORMATION IS AVAILABLE ON THE P100, P200 OR P300. ADDITION INFORMATION CAN BE FOUND IN THE DIAGNOSTIC STATUS WORD ON THE P400 OR P500. AFTER THE MEMORY TEST, THE ERROR MESSAGES, 'PARITY ERROR', OR, 'MACHINE CHECK', WILL BE PRINTED. IF THE ERRORS PERSIST, THE MESSAGES PERSIST.

NON OCTAL PHYSICAL DEVICE NUMBER, 102

THE MESSAGE, 'OCTAL ONLY', WILL BE PRINTED IF THE USER ENTERS A NON OCTAL CHARACTER FOR THE PHYSICAL DEVICE NUMBER THE 'PHYSICAL DEVICE=' PROMPT IS ISSUED AGAIN AT THE SYSTEM CONCOLE..

BAD DEVICE TYPE, 103

THE BAD DEVICE TYPE CODE WILL APPEAR IN THE DATA LIGHTS IF A DEVICE TYPE OF 7 IS DETECTED. THE 'PHYSICAL DEVICE=' PROMPT IS ISSUED AGAIN AT THE SYSTEM CONCOLE..

BAD STATUS, 104

WHENEVER BAD STATUS IS DECTECTED, THE STATUS IS STORED IN LOCATION 40 OCTAL. DURING THE FIRST PHASE, LOADING THE BOOT PROGRAM ITSELF, A HALT THEN OCCURS WITH THE CODE 104 IN THE CONTROL PANEL DATA LIGHTS. WHILE TRYING TO LOAD DOS, THE MESSAGE 'BAD STATUS' IS PRINTED FOLLOWED BY THE STATUS WORD.

BAD RECORD ID, 105

AS EACH RECORD IS READ, THE RECORD ADDRESS REQUESTED IS CHECKED AGAINST THE ADDRESS OF THE RECORD READ AS FOUND IN THE RECORD ITSELF. IF THESE ADDRESSES DO NOT MATCH, THEN A HALT WILL OCCUR WITH THE CODE 105 IN THE CONTROL PANEL DATA LIGHTS. THE REQUESTED ADDRESS IS IN LOCATIONS 723 AND 724 OCTAL AND THE ADDRESS IN THE RECORD IS IN LOCATIONS 760 AND 761 OCTAL.

WHEN SEARCHING FOR OR LOADING DOS, A MESSAGE WILL BE PRINTED 'BAD RECORD ID, RRRRRR RRRRRR FFFFFF FFFFFF', WHERE THE R'S ARE TWO WORDS OF REQUESTED OCTAL ADDRESS AND THE F'S ARE TWO WORDS OF FOUND OCTAL ADDRESS. THE 'PHYSICAL DEVICE=' PROMPT IS ISSUED AGAIN AT THE SYSTEM CONCOLE..

INCOMPATIBLE BOOT RECORDS, 106

THE FIRST AND SECOND RECORDS ARE CHECKED TO SEE IF THEY COME FROM THE SAME VERSION OF THE BOOT PROGRAM. THEY MAY COME FROM DIFFERENT VERSIONS IF AN OLD (CONTROL PANEL) CPBOOT WHICH ALWAYS READS FROM UNIT ONE GETS THE FIRST RECORD OF A NEW (DISK) BOOT. THE NEW BOOT GETS ITS SECOND RECORD FROM THE UNIT DESIGNATED BY SWITCHES 8 AND 9. THE SECOND AND SUBSEQUENT RECORDS MAY THEREFORE COME FROM A DIFFERENT VERSION OF BOOT. IF SUCH AN INCOMPATIBILITY IS RECOGNIZED, THEN THE BOOT PROGRAM WILL HALT WITH A 106 OCTAL IN THE DATA LIGHTS.

'FILE' NOT FOUND, 107

IF THE REQUIRED VERSION OF DOS OR THE DOS UFD IS NOT ON THE REQUESTED PHYSICAL DEVICE, THEN THE MESSAGE, "'FILE' NOT FOUND", WILL BE PRINTED, WHERE 'FILE' IS THE NAME OF THE REQUESTED FILE. THE 'PHYSICAL DEVICE=' PROMPT IS ISSUED AGAIN AT THE SYSTEM CONCOLE..

MEMORY TEST FAILURE, 110

WHILE TESTING THE MEMORY, IF THE TEST PATTERN WRITTEN AND THAT READ DO NOT MATCH, THEN A MESSAGE WILL BE PRINTED, 'MEM TEST MISMATCH LOC XXXXXX', WHERE XXXXXX IS THE LOCATION OF THE WORD BEING TESTED. DURING MEMORY TEST, IF EITHER A PARITY ERROR OR A MACHINE CHECK IS DETECTED, THEN THE ADDRESS OF THE WORD BEING TESTED WILL BE PRINTED FOLLOWED BY THE MESSAGE 'PARITY ERROR' OR 'MACHINE CHECK'. THE 'PHYSICAL DEVICE=' PROMPT IS ISSUED AGAIN AT THE SYSTEM CONCOLE..

1.9 HALTS

UNDER CERTAIN UNUSUAL CIRCUMSTANCES (HARDWARE OR SOFTWARE MALFUNCTION), PRIMOS WILL HALT (AFTER HAVING EXECUTED A HLT INSTRUCTION).

THE HALTS THAT ARE RELATED TO HARDWARE ARE CALLED CHECKS. (FOR A COMPLETE DISCUSSION OF CHECKS IN THE P400, SEE MAN2798.) WHEN PRIMOS HALTS DUE TO A CHECK OF SOME KIND, AN ADDRESS (OCTAL) IS LEFT IN THE DATA LIGHTS ON THE CONTROL PANNEL. WHEN THIS OCCURS, THE HALT IS SAID TO BE A CODED HALT.

FOR OTHER HALTS (SOFTWARE RELATED), A LOAD MAP OF PRIMOS (M_PRMOS) AND THE CONTENTS OF THE DATA LIGHTS ARE USED TO DETERMINE THE LOCATION OF THE HALT.

1.9.1 CHECKS

CHECKS INDICATE VARIOUS (AND SOMETIMES SERIOUS) EXCEPTIONAL CONDITIONS THAT HAVE OCCURED IN THE HARDWARE. WHEN A CHECK OCCURS, FOUR WORDS OF INFORMATION (PB HIGH, PB LOW, KEYS, AND MODALS) ARE SAVED IN A CHECK HEADER AND CONTROL IS TRANSFERRED TO THE WORD FOLLOWING THE CHECK HEADER. THE CHECK HEADERS ARE WIRED DOWN IN THE SEG4 MODULE, AND HENCE CAN BE EXPECTED NOT TO MOVE. CURRENTLY DEFINED CHECKS ARE:

SYMBOL	HEADER LIGHTS		DESCRIPTION
PWRFL_	200	206	POWER FAILURE
MEMPA_	270	277	UNCORRECTED MEMORY PARITY ERROR
MCHK_	300	306	MACHINE CHECK
MMOD_	310	316	MISSING MEMORY MODULE

1.9.2 MEMORY ERRORS

THE FOLLOWING ARE HALT LOCATIONS IN PRIMOS WHEN MEMORY ERRORS OCCUR:

SYMBOL	DESCRIPTION
BDMEM_	BAD MEMORY AT COLD START. THE PAGE IS AUTOMATICALLY MAPPED OUT BY DEPRESSING THE START SWITCH ON THE CONTROL PANEL. THE HALT IS IN SEG14.
MEMPA_	SEE CHECKS (ABOVE).
MMOD_	SEE CHECKS (ABOVE).

2 NEW AND MODIFIED PRIMOS IV FACILITIES

2.1 FILE SYSTEM MODIFICATIONS

2.1.1 ADDITIONAL FILE UNITS

THE NUMBER OF FILE UNITS AVAILABLE TO EACH USER HAS BEEN INCREASED TO 63; UNITS 1 THRU 62 MAY BE USED FOR ANY PURPOSE, AND UNIT 63 IS RESERVED AS THE COMOUTPUT FILE UNIT.

2.1.2 NEW SRCH$$ KEY - SYSTEM SUPPLIED FILE UNIT

IT IS NOW POSSIBLE TO HAVE PRIMOS CHOOSE AN UNUSED FILE UNIT FOR OPERATIONS PERFORMED BY SRCH$$.

K$GETU PRIMOS CHOOSES AN UNUSED FILE UNIT NUMBER AND RETURNS IT TO THE CALLING PROGRAM IN UNIT.

WHEN REQUESTED TO SUPPLY A FILE TO UNIT NUMBER WITH THE USE OF THE KEY K$GETU, SRCH$$ SUPPLIES THE HIGHEST UNIT NUMBER THAT IS CURRENTLY NOT IN USE. THIS POLICY WILL TEND TO AVOID CONFLICT WITH EXISTING COMMON USAGE, SUCH AS UNIT 6 IS "THE" COMINPUT UNIT.

THE USER SHOULD NOT BUILD ANY DEPENDENCIES ON THE ABOVE POLICY INTO ANY OF HIS PROGRAMS AS SRCH$$ IS SPECIFIED TO RETURN ANY UNUSED UNIT. IN FACT, THE USER IS ENCOURAGED TO ALWAYS USE THE K$GETU FEATURE TO AVOID ANY FUTURE CONFLICT WITH UNITS USED BY PRIME SUPPLIED SUB-SYSTEMS (MIDAS, ETC.).

K$GETU (:40000) IS AN ADDITIVE KEY AND IS ADDED TO THE KEY(S) SUPPLIED TO SRCH$$.

EXAMPLE:

```
      INTEGER*2 CODE, TYPE, UNIT
$INSERT SYSCOM>KEYS.F
      CALL SRCH$$(K$READ+K$GETU,'FILE',4,UNIT,TYPE,
     X            CODE)
      IF (CODE .NE. 0) GOTO ERROR_PROCESSOR
```

THE ABOVE FORTRAN CALL WILL ATTEMPT TO OPEN THE FILE NAMED 'FILE' IN THE USER'S CURRENTLY ATTACHED UFD. IF SUCCESSFUL, THE FILE UNIT NUMBER ON WHICH 'FILE' HAS BEEN OPENED IS RETURNED IN UNIT. THE TYPE OF THE FILE OPENED IS RETURNED IN TYPE, AND CODE IS SET TO ZERO IF THERE ARE NO ERRORS. IF THERE ARE ANY ERRORS, CODE WILL BE NONZERO, AND THE VALUES OF TYPE AND UNIT ARE UNDEFINED.

IF NO FILE UNITS ARE AVAILABLE, THE ERROR CODE E$FUIU (ALL UNITS IN USE) IS RETURNED. THIS CODE IS RETURNED IF EITHER 1) THE PROCESS (USER) HAS EXCEEDED THE MAXIMUM NUMBER OF FILE UNITS THE PROCESS (USER) MAY HAVE, OR 2) THE TOTAL NUMBER OF FILE UNITS IN USE FOR ALL PROCESSES (USERS) EXCEEDS THE MAXIMUM NUMBER OF FILE UNITS AVAILABLE TO ALL PROCESSES (USERS).

2.1.3 NEW PRWF$$ KEY - GUARANTEED WRITE TO DISK

IT IS NOW POSSIBLE TO GUARANTEE THAT PRWF$$ WHEN CALLED WITH THE KEY K$WRIT WILL NOT RETURN UNTIL THE DISK RECORD(S) INVOLVED ARE WRITTEN TO DISK.

K$FRCW — ACTUALLY PERFORM THE WRITE TO DISK BEFORE EXECUTING THE NEXT INSTRUCTION IN THE PROGRAM. SINCE THE K$FRCW KEY DEFEATS THE DISK BUFFERING MECHANISM (ASSOCIATIVE BUFFERS) IT SHOULD BE USED WITH CARE AS IT INCREASES THE ACTUAL AMOUNT OF DISK I/O. IT SHOULD ONLY BE USED WHEN A PROGRAM MUST KNOW THAT DATA IS PHYSICALLY ON A DISK (E.G., AS WHEN IMPLEMENTING ERROR RECOVERY SCHEMES).

THE PROGRAMMER IS RESPONSIBLE FOR ENSURING THAT ONLY ONE PROCESS (USER) IS INVOLVED IN THE PRWF$$ CALL CONCURRENTLY. THE FILE MAY BE OPEN FOR USE BY SEVERAL PROCESSES (DEPENDING ON THE SETTING OF THE FILE'S READ/WRITE CONCURRENCY LOCK, RWLOCK). THE FORCED WRITE APPLIES ONLY TO THE DATA WRITTEN BY THE PROCESS PERFORMING THE OPERATION.

K$FRCW (:40000) IS AN ADDITIVE KEY AND IS ADDED TO THE K$WRIT KEY SUPPLIED TO PRWF$$.

EXAMPLE:

```
      INTEGER*2 ARRAY(40), CODE, UNIT, RET
$INSERT SYSCOM>KEYS.F
      CALL PRWF$$(K$WRIT+K$FRCW+K$PREA,UNIT,LOC(ARRAY),
     X            10,INTL(10),RET,CODE)
      IF (CODE .NE. 0) GOTO ERROR_PROCESSOR
```

THE ABOVE FORTRAN CALL WILL CAUSE THE FILE OPEN ON UNIT NUMBER UNIT TO BE POSITIONED TO THE TENTH WORD IN THE FILE, AND THE FIRST 10 WORDS OF ARRAY WILL BE WRITTEN TO IT. THE NEXT INSTRUCTION IN THE USER'S PROGRAM WILL NOT BE EXECUTED UNTIL THE DATA HAS ACTUALLY BEEN WRITTEN TO DISK. IF AN ERROR IS ENCOUNTERED WHILE WRITING TO DISK, THE ERROR CODE E$DISK (DISK I/O ERROR) IS RETURNED. IF MORE THAN ONE CONCURRENT USER OF THE DISK RECORD IS DETECTED, THE ERROR CODE E$FIUS (FILE IN USE) IS RETURNED. IN THIS CASE, THE WRITE IS NOT LOST, BUT WILL NOT BE PERFORMED IMMEDIATELY.

2.1.4 KEYS.F UPDATE

THE FOLLOWING IS AN UPDATED LISTING OF SYSCOM>KEYS.F.

```
C SYSCOM>KEYS.F      MNEMONIC KEYS FOR FILE SYSTEM (FTN)      07/25/78
      NOLIST
C
C     TABSET 6 11 28 69
C
      INTEGER*2 K$READ,K$WRIT,K$POSN,K$TRNC,K$RPOS,K$PRER,K$PREA,
     X    K$POSR,K$POSA,K$CONV,K$RDWR,K$CLOS,K$DELE,K$EXST,K$GETU,
     X    K$IUFD,K$ISEG,K$CACC,K$NSAM,K$NDAM,K$NSGS,K$NSGD,K$CURR,
     X    K$IMFD,K$ICUR,K$SETC,K$SETH,K$ALLD,K$SPOS,K$GOND,K$MSIZ,
     X    K$MENT,K$ENTR,K$SENT,K$GPOS,K$UPOS,K$NAME,K$FRCW,
     X    K$PROT,K$DTIM,K$DMPB,K$RWLK,K$NRTN,K$SRTN,K$IRTN,K$HOME,
     X    K$MVNT,K$RSUB,K$FULL,K$FREE
      PARAMETER
     X
     X /*************************************************************/
     X /*                                                           */
     X /*      KEY DEFINITIONS                                      */
     X /*                                                           */
     X /*                                                           */
     X /********************* PRWF$$ *********************           */
     X /*                 ****** RWKEY  ******                      */
     X    K$READ = :1,      /* READ                                 */
     X    K$WRIT = :2,      /* WRITE                                */
     X    K$POSN = :3,      /* POSITION ONLY                        */
     X    K$TRNC = :4,      /* TRUNCATE                             */
     X    K$RPOS = :5,      /* READ CURRENT POSITION                */
     X /*                 ****** POSKEY ******                      */
     X    K$PRER = :0,      /* PRE-POSITION RELATIVE                */
     X    K$PREA = :10,     /* PRE-POSITION ABSOLUTE                */
     X    K$POSR = :20,     /* POST-POSITION RELATIVE               */
     X    K$POSA = :30,     /* POST-POSITION ABSOLUTE               */
     X /*                 ****** MODE   ******                      */
     X    K$CONV = :400,    /* CONVENIENT NUMBER OF WORDS           */
     X    K$FRCW = :40000,  /* FORCED WRITE TO DISK                 */
     X /*                                                          */
     X /********************* SRCH$$ *********************          */
     X /*                 ****** ACTION ******                      */
     X /* K$READ = :1,      /* OPEN FOR READ                        */
     X /* K$WRIT = :2,      /* OPEN FOR WRITE                       */
     X    K$RDWR = :3,      /* OPEN FOR READING AND WRITING         */
     X    K$CLOS = :4,      /* CLOSE FILE UNIT                      */
     X    K$DELE = :5,      /* DELETE FILE                          */
     X    K$EXST = :6,      /* CHECK FILE'S EXISTENCE               */
     X    K$GETU = :40000,  /* SYSTEM RETURNS UNIT NUMBER           */
     X /*                 ****** REF    ******                      */
     X    K$IUFD = :0,      /* FILE ENTRY IS IN UFD                 */
     X    K$ISEG = :100,    /* FILE ENTRY IS IN SEGMENT DIRECTORY   */
     X    K$CACC = :1000,   /* CHANGE ACCESS                        */
     X /*                 ****** NEWFIL ******                      */
     X    K$NSAM = :0,      /* NEW SAM FILE                         */
```

```
X     K$NDAM = :2000,   /* NEW DAM FILE                                  */
X     K$NSGS = :4000,   /* NEW SAM SEGMENT DIRECTORY                     */
X     K$NSGD = :6000,   /* NEW DAM SEGMENT DIRECTORY                     */
X     K$CURR = :177777,/* CURRENTLY ATTACHED UFD                         */
X  /*                                                                    */
X  /********************* ATCH$$ **********************                  */
X  /*                  ****** KEY    ******                             */
X     K$IMFD = :0,      /* UFD IS IN MFD                                 */
X     K$ICUR = :2,      /* UFD IS IN CURRENT UFD                         */
X  /*                  ****** KEYMOD ******                             */
X     K$SETC = :0,      /* SET CURRENT UFD (DO NOT SET HOME)             */
X     K$SETH = :1,      /* SET HOME UFD (AS WELL AS CURRENT)             */
X  /*                  ****** NAME   ******                             */
X     K$HOME = :0,      /* RETURN TO HOME UFD (KEY=K$IMFD)               */
X  /*                  ****** LDISK  ******                             */
X     K$ALLD = :100000,/* SEARCH ALL DISKS                               */
X  /* K$CURR = :177777,/* SEARCH MFD OF CURRENT DISK                     */
X  /*                                                                    */
X  /********************* SGDR$$ **********************                  */
X  /*                  ****** KEY    ******                             */
X     K$SPOS = :1,      /* POSITION TO ENTRY NUMBER IN SEGDIR            */
X     K$GOND = :2,      /* POSITION TO END OF SEGDIR                     */
X     K$GPOS = :3,      /* RETURN CURRENT ENTRY NUMBER                   */
X     K$MSIZ = :4,      /* MAKE SEGDIR GIVEN NR OF ENTRIES               */
X     K$MVNT = :5,      /* MOVE FILE ENTRY TO DIFFERENT POSITION         */
X     K$FULL = :6,      /* POSITION TO NEXT NON-EMPTY ENTRY              */
X     K$FREE = :7,      /* POSITION TO NEXT FREE ENTRY                   */
X  /*                                                                    */
X  /********************* RDEN$$ **********************                  */
X  /*                  ****** KEY    ******                             */
X  /* K$READ = :1,      /* READ NEXT ENTRY                               */
X     K$RSUB = :2,      /* READ NEXT SUB-ENTRY                           */
X  /* K$GPOS = :3,      /* RETURN CURRENT POSITION IN UFD                */
X     K$UPOS = :4,      /* POSITION IN UFD                               */
X     K$NAME = :5,      /* READ ENTRY SPECIFIED BY NAME                  */
X  /*                                                                    */
X  /********************* SATR$$ **********************                  */
X  /*                  ****** KEY    ******                             */
X     K$PROT = :1,      /* SET PROTECTION                                */
X     K$DTIM = :2,      /* SET DATE/TIME MODIFIED                        */
X     K$DMPB = :3,      /* SET DUMPED BIT                                */
X     K$RWLK = :4,      /* SET PER FILE READ/WRITE LOCK                  */
X  /*                                                                    */
X  /********************* ERPR$$ **********************                  */
X  /*                  ****** KEY    ******                             */
X     K$NRTN = :0,      /* NEVER RETURN TO USER                          */
X     K$SRTN = :1,      /* RETURN AFTER START COMMAND                    */
X     K$IRTN = :2       /* IMMEDIATE RETURN TO USER                      */
X  /*                                                                    */
X  /********************************************************************/
 LIST
```

2.1.5 ERRD.F UPDATE

THE FOLLOWING IS AN UPDATED LISTING OF SYSCOM>ERRD.F.

```
C SYSCOM>ERRD.F       MNEMONIC CODES FOR FILE SYSTEM (FTN)      07/25/78
      NOLIST
C
C     TABSET 6 11 23 56 65
C
      INTEGER*2 E$EOF,E$BOF,E$UNOP,E$UIUS,E$FIUS,E$BPAR,E$NATT,
     X          E$FDFL,E$DKFL,E$NRIT,E$FDEL,E$NTUD,E$NTSD,E$DIRE,
     X          E$FNTF,E$FNTS,E$BNAM,E$EXST,E$DNTE,E$SHUT,E$DISK,
     X          E$BDAM,E$PTRM,E$BPAS,E$BCOD,E$BTRN,E$OLDP,E$BKEY,
     X          E$BUNT,E$BSUN,E$SUNO,E$NMLG,E$SDER,E$BUFD,E$BFTS,
     X          E$FITB,E$NULL,F$IREM,E$DVIU,E$RLDN,E$FUIU,E$DNS,
     X          E$TMUL,E$FBST,E$BSGN,E$FIFC,E$TMRU,E$NASS,E$BFSV,
     X          E$SEMO,E$NTIM,E$FABT,E$FONC,E$NPHA,E$ROOM,E$WTPR,
     X          E$ITRE,E$LAST
C
      PARAMETER
     X
     X /*************************************************************/
     X /*                                                           */
     X /*                                                           */
     X /*       CODE DEFINITIONS                                    */
     X /*                                                           */
     X /*                                                           */
     X     E$EOF=  1,   /* END OF FILE                  PE         */
     X     E$BOF = 2,   /* BEGINNING OF FILE            PG         */
     X     E$UNOP= 3,   /* UNIT NOT OPEN                PD,SD      */
     X     E$UIUS= 4,   /* UNIT IN USE                  SI         */
     X     E$FIUS= 5,   /* FILE IN USE                  SI         */
     X     E$BPAR= 6,   /* BAD PARAMETER                SA         */
     X     E$NATT= 7,   /* NO UFD ATTACHED              SL,AL      */
     X     E$FDFL= 8,   /* UFD FULL                     SK         */
     X     E$DKFL= 9,   /* DISK FULL                    DJ         */
     X     E$NRIT=10,   /* NO RIGHT                     SX         */
     X     E$FDEL=11,   /* FILE OPEN ON DELETE          SD         */
     X     E$NTUD=12,   /* NOT A UFD                    AR         */
     X     E$NTSD=13,   /* NOT A SEGDIR                 --         */
     X     E$DIRE=14,   /* IS A DIRECTORY               --         */
     X     E$FNTF=15,   /* (FILE) NOT FOUND             SH,AH      */
     X     E$FNTS=16,   /* (FILE) NOT FOUND IN SEGDIR   SQ         */
     X     E$BNAM=17,   /* ILLEGAL NAME                 CA         */
     X     E$EXST=18,   /* ALREADY EXISTS               CZ         */
     X     E$DNTE=19,   /* DIRECTORY NOT EMPTY          --         */
     X     E$SHUT=20,   /* BAD SHUTDN (FAM ONLY)        BS         */
     X     E$DISK=21,   /* DISK I/O ERROR               WB         */
     X     E$BDAM=22,   /* BAD DAM FILE (FAM ONLY)      SS         */
     X     F$PTRM=23,   /* PTR MISMATCH (FAM ONLY)      PC,DC,AC   */
     X     E$BPAS=24,   /* BAD PASSWORD (FAM ONLY)      AN         */
     X     E$BCOD=25,   /* BAD CODE IN ERRVEC           --         */
     X     E$BTRN=26,   /* BAD TRUNCATE OF SEGDIR       --         */
     X     E$OLDP=27,   /* OLD PARTITION                --         */
```

```
X     E$BKEY=28,   /* BAD KEY                                --        */
X     E$BUNT=29,   /* BAD UNIT NUMBER                        --        */
X     E$BSUN=30,   /* BAD SEGDIR UNIT                        SA        */
X     E$SUNO=31,   /* SEGDIR UNIT NOT OPEN                   --        */
X     E$NMLG=32,   /* NAME TOO LONG                          --        */
X     E$SDER=33,   /* SEGDIR ERROR                           SQ        */
X     E$BUFD=34,   /* BAD UFD                                --        */
X     E$BFTS=35,   /* BUFFER TOO SMALL                       --        */
X     E$FITB=36,   /* FILE TOO BIG                           --        */
X     E$NULL=37,   /* (NULL MESSAGE)                         --        */
X     E$IREM=38,   /* ILL REMOTE REF                         --        */
X     E$DVIU=39,   /* DEVICE IN USE                          --        */
X     E$RLDN=40,   /* REMOTE LINE DOWN                       --        */
X     E$FUIU=41,   /* ALL UNITS IN USE                       --        */
X     E$DNS=42,    /* DEVICE NOT STARTED                     --        */
X     E$TMUL=43,   /* TOO MANY UFD LEVELS                    --        */
X     E$FBST=44,   /* FAM - BAD STARTUP                      --        */
X     E$BSGN=45,   /* BAD SEGMENT NUMBER                     --        */
X     E$FIFC=46,   /* INVALID FAM FUNCTION CODE              --        */
X     E$TMRU=47,   /* MAX REMOTE USERS EXCEEDED              --        */
X     E$NASS=48,   /* DEVICE NOT ASSIGNED                    --        */
X     E$BFSV=49,   /* BAD FAM SVC                            --        */
X     E$SEMO=50,   /* SEM OVERFLOW                           --        */
X     E$NTIM=51,   /* NO TIMER                               --        */
X     E$FABT=52,   /* FAM ABORT                              --        */
X     E$FONC=53,   /* FAM OP NOT COMPLETE                    --        */
X     E$NPHA=54,   /* NO PHANTOMS AVAILABLE                  --        */
X     E$ROOM=55,   /*  NO ROOM                               --        */
X     E$WTPR=56,   /*  DISK WRITE-PROTECTED                  --        */
X     E$ITRE=57,   /*  ILLEGAL TREENAME                      --        */
X     E$LAST=57    /*  THIS ***MUST*** BE LAST               --        */
X  /*                                                                   */
X  /*                                                                   */
X  /*******************************************************************/
 LIST
```

2.2 EIGHT MEGABYTE SUPPORT

AT REVISION 16, PRIMOS IV HAS BEEN MODIFIED TO SUPPORT SYSTEMS WITH UP TO EIGHT MEGABYTES OF MEMORY.

2.3 VPSD FOR KERNEL DEBUGGING

THE PRIMOS IV OPERATING SYSTEM KERNEL HAS A BUILT-IN VERSION OF THE VPSD DEBUGGER. IT IS LOADED AS PART OF SEGMENT NUMBER 4, AND A TOEHOLD TO ENTER IT IS LOCATED AT '600 IN SEGMENT 14. THE TOEHOLD SERVES TO ENTER 64V MODE AND LOAD DTARO BEFORE TRANSFERRING CONTROL TO VPSD. THUS, AFTER AN OPERATING SYSTEM CRASH, THE MACHINE CAN BE MASTER-CLEARED, '600 ENTERED IN THE SWITCHES, AND THE START SWITCH DEPRESSED IN LOAD MODE. VPSD WILL BE ENTERED AND WILL BE ABLE TO ACCESS ANY KERNEL SEGMENT. SEGMENTS NOT IN DESCRIPTOR TABLE 0, HOWEVER, CANNOT BE DIRECTLY ACCESSED BY VPSD.

THIS VERSION OF VPSD IS USABLE ONLY IF THE PAGES OF SEGMENT 4 THAT CONTAIN VPSD ARE WIRED (MADE NON-PAGABLE, ALSO KNOWN AS LOCKED).

VPSD BAUD RATE

THE VPSD SUPPLIED AS PART OF PRIMOS IV IS SET TO RUN THE SYSTEM TERMINAL AT 300 BAUD. IN SOME CASES, IT MAY BE DESIRABLE TO CHANGE THIS RATE. VPSD, ITSELF, HAS THREE CONTROL WORDS ASSEMBLED INTO IT THAT AFFECT THE BAUD RATE OF THE SYSTEM TERMINAL. THE VALUES OF THESE THREE WORDS CAN BE PATCHED IF THE SYSTEM TERMINAL CANNOT RUN AT 300 BAUD, OR IF A DIFFERENT BAUD RATE IS DESIRED.

VPSD IS LOADED AS PART OF SEGMENT NUMBER 4 STARTING AT A WORD OFFSET OF 2000(OCTAL). THE THREE WORDS TO PATCH ARE LOCATED STARTING AT 2004(OCTAL) IN SEG 4. THE FOLLOWING TABLE GIVES THE VALUES OF THESE WORDS FOR VARIOUS BAUD RATES:

	BAUD RATE	2004	2005	2006
	110	110	27	74000
DEFAULT	300	1010	76	34000
	1200	2010	373	34000
	9600	3410	3735	34000

THESE WORDS CAN BE PATCHED FROM THE CONTROL PANEL, OR THEY CAN BE PATCHED AFTER THE SYSTEM IS BROUGHT UP BY USING THE SHARE COMMAND AND THE VPSD COMMAND.

2.4 FAM EXTENSIONS

2.4.1 REMOTE DISK ACCESS

IT IS NOW POSSIBLE TO PERMIT OR DENY ACCESS TO LOCAL FILE SYSTEM DISK PARTITIONS FROM SPECIFIC OR ALL REMOTE NODES. (SEE SECTION 3 FOR DETAILS ON THE NEW INTERNAL COMMAND 'REMOTE'.)

2.4.2 EXPANDED FUNCTIONALITY

FAM SUPPORTS THE EXPANDED NUMBER OF FILE UNITS AND REMOTE NODES AVAILABLE AT REVISION 16 OF PRIMOS IV.

2.4.3 BETTER MULTIPLEXING OF REMOTE USERS

FAM NOW RELEASES ANY OF ITS OWN INTERNAL RESOURCES IT HAS RESERVED FOR A REMOTE USER WHEN THAT USER NO LONGER HAS FILE UNITS OPEN OR A VALID HOME/CURRENT ATTACH POINT ON THAT SYSTEM. THIS ENABLES FAM TO SUPPORT MORE REMOTE ACTIVITY WITHOUT EXHAUSTING RESOURCES.

2.5 DISK WRITE-PROTECT VIA SOFTWARE

AT REVISION 16 OF PRIMOS IV, A FEATURE HAS BEEN ADDED THAT ALLOWS INDIVIDUAL PARTITIONS OF A STARTED DISK TO BE SOFTWARE WRITE PROTECTED. THE PROTECTION IS ACHIEVED BY DETERMINING THE NATURE OF THE FILE OPERATION ON A DISK PARTITION AT THE TIME THE FILE IS ACCESSED, AND THE EFFECT OF THAT OPERATION ON THE DISK. IF THE FINAL EFFECT OF THE OPERATION IS TO MODIFY THE DISK, THE OPERATION IS NOT PERMITTED.

FOR EXAMPLE, PROTECTION CHECKING WOULD BE PERFORMED FOR A CNAME COMMAND WHEN THE FILE IS ACCESSED. IF THE ATTEMPTED FILE OPERATION WAS TO READ FROM THE FILE, THE PROTECTION CHECKING WOULD OCCUR WHEN THE FILE WAS OPENED, NOT FOR EVERY READ PERFORMED. IN ESSENCE, AS FEW CHECKS AS POSSIBLE ARE MADE TO PROVIDE THE NECESSARY PROTECTION. FOR MORE DETAILS, SEE THE ADDISK AND STARTUP COMMANDS IN SECTION 3.

2.6 IMPROVED DISK RECOVERY

THE DISK ERROR RECOVERY SCHEMES HAVE BEEN MODIFIED TO CORRECT PROBLEMS OCCURING DURING A WARM START. FOR COMPLETE DETAILS, SEE SECTION 9.

3 INTERNAL COMMAND MODIFICATIONS AND ADDITIONS

3.1 TREENAMES

AT REVISION 16 OF PRIMOS, ALL INTERNAL PRIMOS COMMANDS THAT FORMERLY ACCEPTED A FILENAME AS THEIR FIRST ARGUMENT, NOW ACCEPT A TREENAME AS WELL. THE EFFECT OF THIS EXTENDED FUNCTIONALITY IS A MORE GENERAL ENVIRONMENT FOR PROGRAM DEVELOPMENT UNDER PRIMOS. THESE COMMANDS ARE:

ATTACH	BINARY	CLOSE	CNAME
COMINPUT	COMOUTPUT	CREATE	DELETE
INPUT	LISTING	OPEN	PHANTOM
PROTEC	RESTORE	RESUME	SAVE
SHARE			

TERMINOLOGY

TREENAME: THE COMPLETE DESCRIPTION OF A DIRECTORY TREE, STARTING WITH A SPECIFIED DISK VOLUME OR PARTITION AND ENDING WITH A FILENAME. THE GENERAL FORMAT OF A TREENAME IS:

PATHNAME>FILENAME

NOTE: BLANKS ARE NOT ALLOWED IN TREENAMES EXCEPT FOR SEPARATION OF DIRECTORY NAMES AND PASSWORDS. TREENAMES WHICH CONTAIN PASSWORDS MUST BE QUOTED.

PATHNAME: A CHAIN OF DIRECTORIES OPTIONALLY STARTING WITH THE DISK VOLUME NAME AND ENDING WITH THE NAME OF THE DIRECTORY CONTAINING THE FILES TO BE ACCESSED. THE FORMAT OF A PATHNAME IS:

```
[ <VOLUME> ]
[ <LDISK>  ]   DIRECTORY-CHAIN
[ <*>      ]
```

ONLY ONE OF THE DISK SPECIFIERS <VOLUME>, <LDISK>, OR <*> MAY BE PRESENT. THE ANGLE BRACKETS ARE REQUIRED FOR DISK SPECIFICATION. THE DISK SPECIFIERS ARE INTERPRETED AS FOLLOWS:

<VOLUME> IS THE NAME OF A DISK.
<LDISK> IS THE LOGICAL NUMBER OF A DISK (IN OCTAL).
<*> IS THE DISK OF THE CURRENT ATTACH POINT.

DIRECTORY-CHAIN: PART OF A PATHNAME; A SERIES OF DIRECTORIES AND OPTIONAL PASSWORDS SEPARATED BY THE SYMBOL '>', AS IN:

DIRECTORY [PASSWORD] [>SUBDIRECTORY [PASSWORD]] ...

IF THE FIRST ELEMENT OF A DIRECTORY-CHAIN IS AN '*', IT IS INTERPRETED AS THE MOST RECENTLY SET HOME DIRECTORY. THE ASTERISK CONVENTION FOR HOME DIRECTORY MUST NOT BE SPECIFIED IF A

LOGICAL DISK IS SPECIFIED.

DIRECTORY: A DIRECTORY MAY BE THE MFD, A UFD, OR A SUB-UFD. DIRECTORIES WITH NAMES IN THE MFD ARE UFDS; ALL OTHER DIRECTORIES ARE SUB-UFDS.

NOTE: IF A TREENAME CONTAINS A PASSWORD(S), IT MUST BE ENCLOSED WITHIN SINGLE QUOTATION MARKS AS IN: 'MFD XXXXXX>MYUFD'. IN ALL OTHER CASES, A TREENAME NEED NOT BE SPECIFIED WITHIN SINGLE QUOTATION MARKS AS IN: MYUFD>MYSUBUFD>TEST.LST. NOTE THAT BLANKS DO NOT APPEAR IN THIS TREENAME.

THE FOLLOWING DESCRIBES THE EXTENDED SYNTAX AND ILLUSTRATES SOME EXAMPLES FOR EACH OF THE ABOVE COMMANDS:

ATTACH TREENAME [KEY]

ATTACHES TO TREENAME AS CURRENT DIRECTORY. NONOWNER PASSWORDS MAY BE GIVEN. DEFAULT = SET AS HOME DIRECTORY.

EXAMPLE: A <1>MYUFD>MYSUBUFD>BINS

ATTACHES TO THE SUB-UFD 'BINS' IN THE DIRECTORY-CHAIN 'MYUFD>MYSUBUFD'. THE UFD 'MYUFD' IS SEARCHED FOR IN THE MFD OF LOGICAL DISK 1. 'BINS' BECOMES THE HOME DIRECTORY.

EXAMPLE: A '*>LISTINGS SECRET'

ATTACHES TO THE SUB-UFD 'LISTINGS' IN THE HOME DIRECTORY. THE SUB-UFD 'LISTINGS' HAS A PASSWORD OF 'SECRET'. 'LISTINGS' BECOMES THE HOME DIRECTORY.

EXAMPLE: A MYUFD>MYSUBDIR>BACKUPS 1/1

ATTACHES TO THE SUB-UFD 'BACKUPS' IN THE DIRECTORY-CHAIN 'MYUFD>MYSUBDIR'. 'MYUFD' IS IN THE CURRENT DIRECTORY ATTACHED TO. 'BACKUPS' DOES NOT BECOME THE HOME DIRECTORY.

EXAMPLE: A <REV16>MFD

ATTACHES TO THE MFD OF THE DISK WITH 'REV16' AS ITS VOLUME NAME AS NONOWNER. THE 'MFD' DOES NOT HAVE A NONOWNER PASSWORD.

EXAMPLE: A <MD16A1>LIB 1/177777

ATTACHES TO THE UFD 'LIB' ON THE DISK WITH 'MD16A1' AS ITS VOLUME NAME. THE UFD 'LIB' DOES NOT BECOME THE HOME DIRECTORY.

BINARY TREENAME

OPENS TREENAME FOR WRITING ON FILE UNIT 3 FOR OUTPUT. EQUIVALENT TO TO THE COMMAND 'OPEN TREENAME 3 2'.

EXAMPLE: B MYUFD>MYSUBUFD>TEST.BIN

THE FILE 'TEST.BIN' IS OPENED ON FILE UNIT 3 FOR WRITING IN THE DIRECTORY-CHAIN 'MYUFD>MYSUBUFD'. ALL MFDS (STARTING WITH LOGICAL DISK 0) ARE SEARCHED FOR THE UFD NAMED 'MYUFD'.

CLOSE [TREENAME] [FILE-UNIT] ... [FILE-UNIT]

CLOSES TREENAME AND/OR FILE UNITS.

EXAMPLE: C MYUFD>L_TEST

THE FILE 'L_TEST' IN THE UFD 'MYUFD' IS CLOSED. ALL MFDS (STARTING WITH LOGICAL DISK 0) ARE SEARCHED FOR THE UFD NAMED 'MYUFD'.

CNAME TREENAME FILENAME

CHANGES THE LAST NAME IN TREENAME TO FILENAME. REQUIRES OWNER RIGHTS. THE NEW NAME MUST BE A FILENAME; OTHERWISE ONE WOULD BE MOVING FILES.

EXAMPLE: CN TOOLS>FORTRAN>TEST OLDTEST

THE FILE NAMED 'TEST' IN THE DIRECTORY-CHAIN 'TOOLS>FORTRAN' IS CHANGED TO 'OLDTEST'. ALL MFDS (STARTING WITH LOGICAL DISK 0) ARE SEARCHED FOR THE UFD NAMED 'TOOLS'.

EXAMPLE: CN 'MEMOS>CONFIDENTIAL SECRET>CURRENT' OLD

THE FILE NAMED 'CURRENT' IN THE DIRECTORY-CHAIN 'MEMOS>CONFIDENTIAL' IS CHANGED TO 'OLD'. THE SUB-UFD 'CONFIDENTIAL' HAS A PASSWORD OF 'SECRET'. ALL MFDS (STARTING WITH LOGICAL DISK 0) ARE SEARCHED FOR THE UFD NAMED 'MEMOS'.

COMINPUT TREENAME [FILE-UNIT]

READS COMMAND INPUT FROM TREENAME INSTEAD OF TERMINAL.

EXAMPLE: CO MYUFD>MYSUBDIR>C_TEST1 33

COMMAND INPUT IS SWITCHED TO THE FILE 'C_TEST1' IN THE DIRECTORY-CHAIN 'MYUFD>MYSUBDIR' ON FILE UNIT 33 (OCTAL). ALL MFDS (STARTING WITH LOGICAL DISK 0) ARE SEARCHED FOR THE UFD NAMED 'MYUFD'. NOTE THAT WHILE THE NEXT COMMAND COMES FROM THE FILE 'C_TEST1', THE ATTACH POINT OF THE PROCESS REMAINS UNCHANGED.

COMOUTPUT TREENAME

SENDS OUTPUT STREAM TO SPECIFIED TREENAME ON THE COMOUTPUT FILE UNIT.

EXAMPLE: COMO *>UTOPIA84>TESTRUN

THE OUTPUT STREAM IS SENT TO THE FILE 'TESTRUN' IN THE DIRECTORY-CHAIN '*>UTOPIA84'. THE SUB-UFD 'UTOPIA84' IS CONTAINED IN THE DIRECTORY WHERE HOME WAS MOST RECENTLY SET.

CREATE TREENAME

CREATES A NEW UFD IN THE DIRECTORY SPECIFIED BY TREENAME.

EXAMPLE: CR '<1>MFD XXXXXX>ACCOUNTS>RECEIVABLE'

THE SUB-UFD 'RECEIVABLE' IS CREATED IN THE UFD 'ACCOUNTS'. THE UFD 'ACCOUNTS' IS IN THE 'MFD' OF LOGICAL DISK 1. THE MFD HAS A PASSWORD OF 'XXXXXX'.

DELETE TREENAME

DELETES A FILE OR EMPTY DIRECTORY.

EXAMPLE: DELETE LISTINGS>TEST.FTN.LST

THE FILE NAMED 'TEST.FTN.LST' IS DELETED FROM THE UFD 'LISTINGS'. ALL MFDS (STARTING WITH LOGICAL DISK 0) ARE SEARCHED FOR THE UFD NAMED 'LISTINGS'.

INPUT TREENAME

OPENS TREENAME FOR READING ON FILE UNIT 1. EQUIVALENT TO THE COMMAND 'OPEN TREENAME 1 1'.

EXAMPLE: I <*>INVENTORY>ONHAND>DATA

THE FILE NAMED 'DATA' IN THE DIRECTORY-CHAIN 'INVENTORY>ONHAND' IS OPENED FOR READING ON FILE UNIT 1. THE UFD 'INVENTORY' IS SEARCHED FOR IN THE MFD OF THE CURRENT DISK.

LISTING TREENAME

OPENS TREENAME FOR WRITING ON FILE UNIT 2. EQUIVALENT TO THE COMMAND 'OPEN TREENAME 2 2'.

EXAMPLE: L <BACKUP>PAYROLL>THIS_WEEK

THE FILE NAMED 'THIS_WEEK' IN THE UFD 'PAYROLL' IS OPENED FOR WRITING ON FILE UNIT 2. THE UFD 'PAYROLL' IS SEARCHED FOR IN THE MFD OF THE DISK WITH THE VOLUME NAME OF 'BACKUP'.

OPEN TREENAME UNIT KEY

OPENS A TREENAME ON THE SPECIFIED UNIT WITH A DISPOSITION SPECIFIED BY KEY.

EXAMPLE: O MYUFD>MYSUBFD>MYDATA 1 1

THE FILE 'MYDATA' IN THE DIRECTORY 'MYUFD>MYSUBFD' IS OPENED ON FILE UNIT 1 FOR READING. ALL MFDS (STARTING WITH LOGICAL DISK 0) ARE SEARCHED FOR THE UFD NAMED 'MYUFD'.

PHANTOM TREENAME [FILE-UNIT]

RUNS THE SPECIFIED TREENAME AS A PHANTOM USER. THE LAST ELEMENT IN THE TREENAME IS A COMMAND-INPUT FILE.

EXAMPLE: PH *>PRODUCTION>DAILY.CO 52

RUNS THE COMMAND-INPUT FILE 'DAILY.CO' IN THE DIRECTORY-CHAIN '*>PRODUCTION' AS A PHANTOM USER. THE SUB-UFD 'PRODUCTION' IS CONTAINED IN THE DIRECTORY WHERE HOME WAS MOST RECENTLY SET. THE PHANTOM'S HOME UFD IS 'PRODUCTION'. FILE UNIT 52 (OCTAL) IS USED AS THE COMMAND-INPUT FILE UNIT.

PROTEC TREENAME [OWNER-RIGHTS [NONOWNER-RIGHTS]]

SETS PROTECTION RIGHTS ON TREENAME.

EXAMPLE: PRO <OLD>MYUFD>SECRET 0 0

SETS PROTECTION RIGHTS TO THE FILE 'SECRET' IN THE UFD 'MYUFD' TO NO ACCESS FOR BOTH THE OWNER AND NONOWNER. THE UFD 'MYUFD' IS SEARCHED FOR IN THE MFD OF THE DISK WITH THE VOLUME NAME OF 'OLD'.

RESTORE TREENAME

RESTORES THE RUNFILE CONTAINED IN TREENAME INTO MEMORY.

EXAMPLE: REST *>*TEST

RESTORES THE RUNFILE *TEST IN THE MOST RECENTLY SET HOME DIRECTORY TO MEMORY. THIS IS EQUIVALENT TO THE COMMAND 'REST *TEST'.

RESUME TREENAME [P] [A] [B] [X] [KEYS]

RUNS (RESTORES AND STARTS) THE EXTERNAL PROGRAM CONTAINED IN TREENAME.

EXAMPLE: R CMDNCO>DATE

RUNS (RESTORES AND STARTS) THE EXTERNAL PROGRAM NAMED 'DATE' IN THE UFD 'CMDNCO'. ALL MFDS (STARTING WITH LOGICAL DISK 0) ARE SEARCHED FOR THE UFD NAMED 'CMDNCO'.

SAVE TREENAME START-ADDRESS END-ADDRESS [A] [B] [X] [KEYS]

SAVES MEMORY IMAGE/CONTENTS FROM THE SPECIFIED START-ADDRESS TO END-ADDRESS AS TREENAME. DO NOT USE WITH SEG FORMAT (64V OR 32I) RUNFILES.

EXAMPLE: SA 'MFD XXXXXX>MYUFD>*NEW' 100 177777

SAVES THE MEMORY IMAGE IN LOCATIONS 100-177777 (OCTAL) AS THE FILE '*NEW' IN UFD 'MYUFD'. THE UFD 'MYUFD' IS IN THE 'MFD' OF LOGICAL DISK 0. THE 'MFD' HAS A PASSWORD OF 'XXXXXX'.

SHARE TREENAME SEGMENT-NUMBER [SEGMENT-DESCRIPTOR-WORD]

SHARES TREENAME IN THE SPECIFIED SEGMENT-NUMBER. THIS IS AN OPERATOR COMMAND, AND MUST BE ISSUED FROM THE SYSTEM CONSOLE.

EXAMPLE: SHA SYSTEM>UI2000 2000

THE FILE 'UI2000' IN THE UFD 'SYSTEM' IS PLACED INTO SEGMENT 2000. ALL MFDS (STARTING WITH LOGICAL DISK 0) ARE SEARCHED FOR THE UFD NAMED 'SYSTEM'.

3.2 ADDISK COMMAND MODIFICATION

THE ADDISK COMMAND HAS BEEN EXTENDED TO ALLOW A DISK PARTITION TO BE SOFTWARE WRITE-PROTECTED.

A DISK IS WRITE-PROTECTED BY SPECIFYING PROTECT IN THE ADDISK COMMAND AS FOLLOWS:

ADDISK PROTECT DEVN01 [DVN02 ... DVN09]

PROTECT MAY ONLY BE SPECIFIED FOR DISKS WHICH ARE ADDED LOCALLY, AND DOES NOT GOVERN THE RIGHTS OF REMOTELY ADDED DISKS. REMOTELY ADDED DISKS ASSUME THE WRITE-PROTECTION STATUS OF THE LOCAL SYSTEM.

THE STATUS OF THE WRITE-PROTECT FEATURE MAY BE CHANGED FOR A GIVEN PARTITION BY RESPECIFYING THE STARTUP OR ADDISK COMMAND WITH OR WITHOUT PROTECT.

IF AN SUBSEQUENT ADDISK COMMAND IS ISSUED FOR THE SAME DISK, AND PROTECT IS NOT SPECIFIED, THE WRITE-PROTECT FEATURE IS DISABLED. (AN ADDISK PROTECT TO AN ALREADY PROTECTED DISK DOES NOT CHANGE THE PROTECTION.) IF AN ADDISK PROTECT COMMAND IS ISSUED FOR A DISK THAT DOES NOT HAVE PROTECTION ENABLED, IT IS IMPORTANT THAT THE DISK BE SHUTDOWN FIRST, TO INSURE THAT THE DISK IS NOT INADVERTENTLY WRITTEN UPON.

3.3 LOGOUT COMMAND MODIFICATION

THE LOGOUT COMMAND HAS BEEN MODIFIED SO THAT WHEN 'LOGOUT ALL' IS SPECIFIED FROM THE SYSTEM CONSOLE (USER 1) THE REMOTE FILE ACCESS MANAGER (FAM) IS NOT LOGGED OUT IF IT IS A RUNNING PROCESS.

3.4 LOOK COMMAND MODIFICATION

THE LOOK COMMAND HAS BEEN MODIFIED SO THAT A 'REALLY?' PROMPT IS ISSUED FOR ANY LOOK COMMAND WHOSE REQUEST IS CONSIDERED TO BE RISKY OR DANGEROUS TO SYSTEM INTEGRITY. (IF THE LOOK COMMAND INVOLVES AN ATTEMPT TO DO A FROM FROM A SEGMENT THAT DOES NOT EXIST, AN ATTEMPT TO DO A TO TO A SEGMENT THAT DOES EXIST, OR ATTEMPTS TO MAP EITHER SHARED OR STACK SEGMENTS WITH WRITE PERMISSION, THE COMMAND IS CONSIDERED RISKY OR DANGEROUS TO SYSTEM INTEGRITY.) A SIMPLE 'YES' WILL ALLOW THE OPERATION TO PROCEED.

3.5 MAXUSR COMMAND MODIFICATION

THE MAXUSR COMMAND HAS BEEN MODIFIED SO THAT THE VALUE OF THE VARIABLE MAXUSR IS IGNORED FOR PHANTOMS STARTED FROM THE SYSTEM CONSOLE (USER 1).

3.6 PHANTOM COMMAND MODIFICATION

THE PHANTOM COMMAND HAS BEEN MODIFIED SO THAT THE PRIORITY OF A SPAWNED PROCESS HAS THE SAME PRIORITY AS THE SPAWNING PROCESS. IF THE SPAWNING PROCESS IS PROCESS 1 (THE SYSTEM CONSOLE), THE PRIORITY IS SET TO 1.

3.7 REMOTE COMMAND

THE REMOTE COMMAND ENABLES USER 1 (THE SYSTEM CONSOLE USER) TO PERMIT OR DENY ACCESS TO LOCAL FILE SYSTEM DISK PARTITIONS FROM SPECIFIC OR ALL REMOTE NODES.

```
REMOTE  PERMIT  <OPTION>
        DENY
```

PERMIT PERMITS ACCESS TO SPECIFIC OR ALL LOCAL DISKS BY SPECIFIC OR ALL REMOTE NODES.

DENY DENIES ACCESS TO SPECIFIC OR ALL LOCAL DISKS BY SPECIFIC OR ALL REMOTE NODES.

OPTIONS CAN BE:

```
NODENAME DVN01 [DVN02 ... DVN09]
NODENAME -ALL
-NET     DVN01 [DVN02 ... DVN09]
-NET     -ALL
```

THE FOLLOWING EXAMPLES ILLUSTRATE HOW THIS COMMAND IS USED TO PERMIT ACCESS TO SPECIFIC OR ALL LOCAL DISKS.

```
REMOTE PERMIT NODENAME DVN01 [DVN02 ... DVN09]
```

THIS COMMAND PERMITS NODE NODENAME TO STARTUP OR ADDISK ANY OF THE LOCAL PHYSICAL DISK DEVICES DVN01 THROUGH DVN09. AT LEAST DVN01 MUST BE SPECIFIED. ALL SPECIFIED LOCAL DISK PARITITIONS MUST ALREADY BE STARTED-UP WITH A PREVIOUS ADDISK OR STARTUP COMMAND.

```
REMOTE PERMIT NODENAME -ALL
```

THIS COMMAND PERMITS NODE NODENAME TO STARTUP OR ADDISK ALL PRESENTLY STARTED UP LOCAL DISK PARTITIONS. IT HAS NO EFFECT ON LOCAL PARTITIONS ADDED AFTER THIS COMMAND IS EXECUTED.

```
REMOTE PERMIT -NET DVN01 [DVN02 ... DVN09]
```

THIS COMMAND PERMITS ALL NETWORK NODES CONFIGURED TO ACCESS THE SPECIFIED LOCAL DISK PARTITIONS.

```
REMOTE PERMIT -NET -ALL
```

THIS COMMAND PERMITS ALL NETWORK NODES TO ACCESS ALL PRESENTLY STARTED UP DISK PARITIONS.

PERMIT AND DENY AFFECT ONLY DISK PARTITIONS ALREADY STARTED UP AT THE TIME OF THE REMOTE COMMAND. DISKS SHUT DOWN AND STARTED UP AGAIN WILL GET THE SYSTEM DEFAULT PERMISSIONS UNTIL AN EXPLICIT REMOTE PERMIT OR REMOTE DENY COMMAND CHANGES THEM. THE SYSTEM DEFAULT PERMISSIONS ARE DETERMINED FROM THE FILE NETCON WHICH IS CREATED BY NETCFG. THE REMOTE PERMIT COMMAND WILL NOT AUTOMATICALLY ADD A DISK TO ANY SYSTEM. THE REMOTE DENY COMMAND WILL NOT REVOKE A SYSTEM'S EXISTING ACCESS TO A DISK.

3.8 STARTUP COMMAND MODIFICATION

THE STARTUP COMMAND HAS BEEN EXTENDED TO PERMIT A DISK TO BE SOFTWARE WRITE-PROTECTED.

A DISK IS WRITE-PROTECTED BY SPECIFYING PROTECT IN THE STARTUP COMMAND AS FOLLOWS:

```
STARTUP PROTECT DVN01 [DVN02 ... DVN09]
```

PROTECT MAY ONLY BE SPECIFIED FOR DISKS WHICH ARE STARTED LOCALLY, AND DOES NOT GOVERN THE RIGHTS OF REMOTELY ADDED DISKS. REMOTELY ADDED DISKS ASSUME THE WRITE-PROTECTION STATUS OF THE LOCAL SYSTEM.

THE STATUS OF THE WRITE-PROTECT FEATURE MAY BE CHANGED FOR A GIVEN PARTITION BY RESPECIFYING THE STARTUP OR ADDISK COMMAND WITH OR WITHOUT PROTECT.

IF AN SUBSEQUENT STARTUP COMMAND IS ISSUED FOR THE SAME DISK, AND PROTECT IS NOT SPECIFIED, THE WRITE-PROTECT FEATURE IS DISABLED. (AN STARTUP PROTECT TO AN ALREADY PROTECTED DISK DOES NOT CHANGE THE PROTECTION.) IF AN STARTUP PROTECT COMMAND IS ISSUED FOR A DISK WHICH DOES NOT HAVE PROTECTION ENABLED, IT IS IMPORTANT THAT THE DISK BE SHUTDOWN FIRST, TO INSURE THAT THE DISK IS NOT INADVERTENTLY WRITTEN UPON.

4 CORRECTED REVISION 15.1, 15.2 PROBLEMS

THE FOLLOWING ARE PROBLEMS WHICH HAVE BEEN CORRECTED FOR REVISION 16 OF PRIMOS. WHERE APPLICABLE, TAR #'S ARE INLCUDED.

4.1 SLEEP$

CALLS TO SLEEP$ FOR LONG PERIODS OF TIME RESULTED IN INACCURATE DELAYS IF SYSTEM USAGE WAS HEAVY.

4.2 CONFIG COMMAND NEEDED IN CONFIG FILE

THE CONFIG COMMAND WAS NOT OPTIONAL IF NETWORKS WERE CONFIGURED. THE SYSTEM WOULD FAIL TO COLD START.

4.3 DELAYED LOGIN

A USER'S LOGIN COMMAND WAS SOMTIMES DELAYED FOR AS LONG AS ONE MINUTE.

4.4 SECURITY PROBLEM

USER-RING (RING 3) PROGRAMS COULD USE THE RING 0 PRIVILEGED RETURN OF E$BPAS (BAD PASSWORD) FROM ATCH$$. THIS ERROR ALLOWED USERS TO WRITE A PROGRAM WHICH ITERATED THROUGH ALL POSSIBLE PASSWORDS IN FINITE AMOUNT OF TIME.

4.5 WRONG LINE NUMBER IN STATUS COMMAND

THE STATUS COMMAND PRINTED THE INCORRECT LINE NUMBER WHEN THE USER'S AMLC LINE NUMBER WAS GREATER THAN THE NUMBER OF CONFIGURED TERMINAL USERS. (TAR # 12860)

4.6 ATCH$$ PROBLEM

CALLS TO ATCH$$ TO ATTACH TO HOME DIRECTORY FAILED IF THE HOME DIRECTORY WAS ON A REMOTE DISK AND THE LOGICAL DEVICE ARGUMENT WAS K$ALLD (100000 OCTAL).

4.7 CARD READER-PUNCH PROBLEM

CALLS TO T$CMPC AND T$PMPC (CARD READER-PUNCH) COULD RESULT IN SPURIOUS 'NO MPC' ERROR MESSAGES. (TAR # 25725)

4.8 GARBLED COLD START MESSAGE

AT COLD START, THE PRIMOS HEADER MESSAGE COULD SOMETIMES BE GARBLED DUE TO INCORRECT BAUD RATE SETTING.

4.9 9600 BAUD CONSOLE

SETTING THE BAUD RATE OF THE SYSTEM CONSOLE TO 9600 BAUD VIA THE B REGISTER SETTING OF *COLDS CAUSED SYSTEM CRASH DURING COLD START.

4.10 REMOTE LOGIN

THE FORCED LOGOUT OF A USER WHO WAS REMOTELY LOGGED IN CAUSED ANOMALOUS BEHAVIOUR.

4.11 REMOTE ATTACH

AN ATTACH TO A NONEXISTANT UFD ON A REMOTE DISK INCORRECTLY RETURNED THE ERROR CODE F$IREM INSTEAD OF E$FNTF.

4.12 SYSTEM HALTS

IF THE COMOUTPUT UNIT WAS OPEN, AND THE SYSTEM MESSAGE 'BAD RTNREC' WAS ISSUED, THE SYSTEM WOULD HALT IN N1LOCK.

OTHER PROBLEMS RELATED TO THE SYSTEM HANGING IN N1LOCK HAVE BEEN CORRECTED.

4.13 LOC POINTER WEAKENING

A USER SUPPLIED LOC POINTER WAS NOT BEING WEAKENED FOR CALLS TO T$AMLC, T$CMPC, T$LMPC, T$PMPC, AND T$VG.

4.14 SCHEDULING

A USER ATTEMPTING TTY INPUT VIA 'INA 4' FAILED TO GET 'INTERACTIVE' TIMESLICE.

4.15 COMINPUT

THE COMINPUT COMMAND INCORRECTLY TREATED -FILENAME AS THE COMINPUT FILE, WHERE "FILENAME" WAS THE NAME OF A FILE.

CO TTY DID NOT CHECK TO SEE IF THE COMINPUT UNIT WAS OTHER THAN UNIT 6. (TAR # 25476, 80564)

4.16 DISK ASSIGNMENT

PRIMOS IV WOULD INCORRECTLY ACCEPT ADDISK, DISK, AND ASSIGN COMMANDS ON OVERLAPPING DISK PARTITIONS.

4.17 SETIME

THE SETIME COMMAND WOULD FAIL IF A DISK ERROR OCCURRED WHILE WRITING THE COLD START MESSAGE TO THE LOGREC FILE.

4.18 LOGLOG

THE LOGLOG DIRECTIVE OF CONFIG DID NOT WORK AS PREVIOUSLY DOCUMENTED.

4.19 UNASSIGN

THE UNASSIGN COMMAND DID NOT WAIT UNTIL THE BUFFER WAS CLEARED FOR A CARD READER, CARD PUNCH, PAPERTAPE READER/PUNCH, ETC.

4.20 LOGREC FILE

THE PACKNAME IN THE LOGREC FILE MAY HAVE HAD A NONPRINTING CHARACTER IF THE NAME HAD AN ODD NUMBER OF CHARACTERS.

4.21 REMOTE UFD RIGHTS

WHEN OPERATING ON A REMOTE DISK, NONOWNERS OF A UFD WERE ILLEGALLY ALLOWED TO CREATE NEW FILES.

4.22 MAXSCH

THE MAXSCH COMMAND INCORRECTLY DEFAULTED TO 0 INSTEAD OF 3. (TAR # 24959)

4.23 MESSAGE ALL NOW

THE COMMAND 'MESSAGE ALL NOW' COULD HANG THE SYSTEM CONSOLE FOR LONG PERIODS OF TIME. COMMAND PROCESSING HAS BEEN CHANGED SO THAT THE SYSTEM ONLY WAITS A SHORT PERIOD OF TIME FOR ROOM IN THE TTY OUTPUT BUFFERS. IF A MESSAGE CAN NOT BE PLACED IN A BUFFER, THE SYSTEM CONSOLE USER (OPERATOR) IS INFORMED AS TO WHICH USERS DID NOT RECEIVE THE MESSAGE. (TAR # 11324)

4.24 REMOTE PHANTOM

PHANTOMS WHOSE COMMAND FILES WERE ON REMOTE DISKS FAILED TO LOGIN, AND NO ERROR WAS SENT TO THE SPAWNING PROCESS. THIS OPERATION NOW RESULTS IN THE ERROR E$IREM (ILLEGAL REMOTE REFERENCE) IF THE PHANTOM CAN NOT BE STARTED.

4.25 SKS 604

THE 'SKS 604' INSTRUCTION (SKIP ON TRANSMITTER READY) ALWAYS SKIPPED IF THE USER WAS A PHANTOM. THIS PREVENTED PHANTOMS FROM RUNNING MULTIPLE DEVICES (SERIAL LINE PRINTERS, ETC.) WHICH WERE CONNECTED TO THE SOC OR OPTION-A CONTROLLER. (TAR # 20022)

4.26 DELAY

THE DELAY COMMAND RESET TERMINAL CHARACTERISTICS BEFORE ALL OF THE OUTPUT BUFFER WAS FLUSHED CAUSING SOME CHARACTERS TO BE LOST. (TAR # 24726)

4.27 XOFF

IF TERMINAL OUTPUT WAS TURNED OFF, AND A USER WAS FORCED OFF (LOGOUT OR DISCONNECT COMMAND), THAT USER TERMINAL WOULD HANG AND COULD NOT BE LOGGED OUT. (TAR # 24726)

4.28 REMOTE ATTACHES & SPAS$$

IF SPAS$$ WAS CALLED WHILE ATTACHED TO A REMOTE UFD, A USER THAT HAD OWNER PERMISSION HAD THE RIGHTS CHANGED TO THAT OF NONOWNER. FAM NOW RETAINS THE NEW PASSWORD TO USE IN RE-ATTACHING TO THAT UFD FOR SUBSEQUENT OPERATIONS.

5 CORRECTED REVISION 16.0 PROBLEMS

THE FOLLOWING IS A LIST OF PROBLEMS WHICH WERE CORRECTED AT REV16.1. WHERE APPLICABLE, TAR NUMBERS ARE INCLUDED.

5.1 FORCEW

CALLS TO FORCEW GAVE SPURIOUS E$IREM.

5.2 UNIT 0 AND MUNIT

RING 3 CALLS (NOT DOSSUB) ARE PROHIBITED FROM OPENING FILE ON UNIT 0 (SYSUN - RESERVED FOR SYSTEM USE) AND ON THE HIGHEST UNIT NUMBER (MUNIT - RESERVED FOR COMOUTPUT).

5.3 SPURIOUS E$FIUS

THE ERROR MESSAGE "FILE IN USE" WAS GIVEN WHEN THE FILE WAS IN FACT NOT IN USE.

5.4 SPURIOUS CHARACTERS IN COMOUTPUT FILE

IF SRCH$$ WAS USED TO CLOSE THE COMOUTPUT FILE INSTEAD OF COMO$$, THEN THE NEXT COMOUTPUT FILE OPENED COULD HAVE EXTRA CHARACTERS INSERTED AT THE BEGINNING OF THE FILE.

5.5 ERROR MESSAGES FROM INA'S AND OTA'S

TERMINAL OUTPUT WAS FORCED ON IF AN ERROR MESSAGE RESULTED FROM AN INA OR OTA IN A USER PROGRAM.

5.6 SYSTEM HANG DURING PRINTER UNASSIGN

THE SYSTEM WOULD HANG IF THE LINE PRINTER BEING UNASSIGNED WAS POWERED OFF (TAR #25477).

5.7 BROKEN DISK FILE STRUCTURE

A PROBLEM IN THE LOCATE LOCKING STRATEGY ALLOWED AN OPPORTUNITY FOR POINTER MIS-MATCHES TO BE CREATED ON THE DISK.

5.8 MAXSCH

THE DEFAULT VALUE FOR THE MAXSCH COMMAND WAS CORRECTED TO BE THREE.

5.9 DMQ BUFFERS

THE AMLBUF CONFIGURATION PARAMETER FAILED TO SET-UP THE DMQ BUFFERS CORRECTLY IF THE DEFAULT SIZE WAS CHANGED. ON SOME SYSTEMS, THE DEFAULT BUFFERS WERE NOT CORRECTLY INITIALIZED. (TAR #23422, #24788)

5.10 VERSATEC PRINTER-PLOTTER

PRINT MODE AFTER SIMULTANEOUS PRINT-PLOT MODE FAILED. (TAR #12068)

5.11 SYSCOM>KEYS.P

THE KEYS INSERT FILE WAS INCORRECTLY FORMATTED. COMMENTS WERE NOT RECOGNIZED AS SUCH.

6 CORRECTED REVISION 16.1 PROBLEMS

THE FOLLOWING IS A LIST OF PROBLEMS WHICH WERE CORRECTED AT REV16.2 BUT NOT AT REV16.1. WHERE APPLICABLE, TAR NUMBERS ARE INCLUDED.

6.1 PAPER TAPE READER AND PUNCH

PROCESSES USING PTR OR PUNCH WERE GIVEN A DISPROPORTIONATE SHARE OF CPU TIME.

6.2 SRCH$$ USING K$GETU

CALLS TO SRCH$$ TO OPEN FILES ON REMOTE DISKS WHICH USED THE SUB-KEY K$GETU (SYSTEM CHOOSES UNIT NUMBER) WOULD HANG FAM.

6.3 COMOUTPUT COMMAND

COMOUTPUT COMMAND IGNORED ALL EXCEPT FILENAME IN TREENAME IF "-OPTION" SPECIFIED.

6.4 SAVE AND RESTORE

ATTEMPTS TO SAVE OR RESTORE MEMORY IMAGES WHICH WERE AN ENTIRE SEGMENT (0 THROUGH 177777 OCTAL) WOULD NOT WORK. (TAR #15791)

6.5 I$AMLC

CALLS TO T$AMLC TO RETURN STATUS RETURNED INCORRECT INFORMATION. (TAR #14792) OUTPUT BUFFER EMPTIED TOO SLOWLY. (TAR #23421)

6.6 PRWF$$

PRWF$$ SOMETIMES FAILED TO POSITION FILE CORRECTLY ON LARGE DAM FILES.

6.7 GARBLED COLD START MESSAGE

IF AN ASRATE PARAMETER WAS NOT INCLUDED IN THE CONFIG COLD START FILE, THE SYSTEM TERMINAL COULD BE SET TO THE WRONG BAUD RATE. OUTPUT WAS MISSING OR GARBLED. (TAR #14514, #80695)

6.8 MAX REMOTE USERS EXCEEDED

ATTACH HOME TO A LOCAL DISK FROM A REMOTE CURRENT ATTACH POINT FAILED TO INVALIDATE ATTACH POINT ON REMOTE SYSTEM.

6.9 COMINP FILE EOF

THE ERROR MESSAGE "COMINP FILE EOF" WAS RETURNED IF ANY ERROR WAS ENCOUNTERED WHILE READING FROM A COMMAND FILE. THE CORRECT ERROR MESSAGE IS NOW PRINTED. (TAR #80698)

6.10 SYSTEM HANG

IF THE AMLC TRANSPARENT INPUT PROTOCOL WAS USED AND THE INPUT BUFFER WAS NOT EMPTIED BY THE PROGRAM WHILE A LARGE AMOUNT OF INPUT WAS DONE, THE SYSTEM COULD HANG.

6.11 LOST INPUT

TERMINAL INPUT WAS SOMETIMES LOST. THE PROBLEM WAS MOST APPARENT WHEN USING BLOCK MODE TERMINALS.

7 CONFIG - A TOOL FOR CONFIGURING PRIMOS

\NOTE: FOR CONVENIENCE, THIS SECTION HAS BEEN REPEATED (IN LARGE \PART) FROM THE PRIMOS REVISION 15 DOCUMENT.

\AT REVISION 15 OF PRIMOS, IT BECAME POSSIBLE TO SPECIFY CONFIG \PARAMETERS AS A SERIES OF CONFIG DIRECTIVES. THE DIRECTIVES ARE \KEPT IN A DATA FILE IN CMDNCO AND ARE PROCESSED BY THE PRELOADER \TO SET UP MOST SYSTEM PARAMETERS. MOST CONFIGURATION PARAMETERS \MAY STILL BE SPECIFIED VIA THE OLD-STYLE REGISTER SETTINGS, BUT \THE CHANGE TO THE NEW-STYLE IS RECOMMENDED. NEW CONFIGURATION \FACILITIES ARE AVAILABLE ONLY BY SPECIFYING SYSTEM PARAMETERS AS \CONFIG DIRECTIVES.

\NOTE: CONFIG, ITS DATABASES, ANY DATABASES IT ACCESSES, AND \ERROR MESSAGES ARE SUBJECT TO CHANGE AT ANY REVISION OF PRIMOS.

\THIS SECTION INCLUDES THE UPDATES MENTIONED IN SECTION 1, AND \INDICATES WHICH CONFIG DIRECTIVES ARE NO LONGER SUPPORTED. \CHANGES TO THIS SECTION HAVE BEEN MARKED WITH REVISION BARS.

\THE INFORMATION CONFIG PROCESSES WITH RESPECT TO NETWORKS HAS \BEEN REDUCED TO A VERY SIMPLE NET ON DIRECTIVE. ALL OTHER \NETWORK CONFIGURATION INFORMATION IS NOW PROCESSED BY NETCFG. \(SEE SEPARATE DOCUMENT ON NETCFG.) THE CONFIG DIRECTIVES FAM, \MYNAME, AND RLOGIN ARE NO LONGER SUPPORTED, AND THE OLD STYLE \CONFIG COMMAND NO LONGER PERMITS A <NODE> TO BE SPECIFIED. SOME \NEW CONFIG DIRECTIVES HAVE ALSO BEEN ADDED AT REVISION 16.

7.1 OVERVIEW OF PRELOADER ACTIONS

AS IS DONE CURRENTLY, THE PRELOADER ATTACHES TO CMDNCO AND LOOKS FOR THE FILE C_PRMO. IF THE FILE EXISTS, IT IS OPENED FOR COMMAND INPUT; IF IT DOESN'T, THE 'PLEASE ENTER CONFIG' PROMPT IS ISSUED. THE FIRST EXECUTABLE DIRECTIVE IS READ (FROM THE TERMINAL OR FROM C_PRMO), AND A 'CO TTY' IS ISSUED. THE DIRECTIVE IS EXAMINED TO ENSURE IT IS A CONFIG DIRECTIVE.

N.B.: NOTE THAT COMMENTS -- LINES STARTING WITH '*' OR '/*' MAY NOW PRECEDE THE CONFIG COMMAND IN C_PRMO.

AT THIS POINT, THE NEW PRELOADER MAKES AN ADDITIONAL CHECK FOR THE KEYWORD '-DATA' AS THE FIRST NAME ON THE CONFIG COMMAND. IF THIS KEYWORD IS PRESENT, THE SECOND NAME FOLLOWING THE COMMAND IS TAKEN AS THE NAME OF A CONFIGURATION DATA FILE. THE FILE IS OPENED FOR INPUT, AND CONFIGURATION DIRECTIVES ARE PROCESSED AS DESCRIBED BELOW. A NEW-STYLE CONFIG COMMAND APPEARS AS:

```
CONFIG -DATA <CONFIGURATION-DATA-FILENAME>
```

NOTE: WHILE NO RESTRICTIONS ARE PLACED ON <CONFIGURATION-DATA-FILENAME> -- THE NAME OF THE CONFIGURATION DATA FILE -- IT IS SUGGESTED THAT THE NAME CONFIG BE ADAPTED AS A DEFACTO STANDARD.

7.2 CONFIGURATION DIRECTIVES

FOLLOWING THE ABOVE SEQUENCE, THE PRELOADER EITHER HAS READ AN OLD-STYLE CONFIG COMMAND OR HAS THE NAME OF A DATA FILE CONTAINING NEW-STYLE CONFIGURATION DIRECTIVES. THE FOLLOWING DESCRIBES ALL POSSIBLE CONFIGURATION DIRECTIVES IN ALPHABETICAL ORDER.

CORRESPONDENCE TO CURRENT CONFIG PARAMETERS IS NOTED WHERE APPROPRIATE. DIRECTIVES (WHICH CANNOT BE ABBREVIATED) AND LITERAL STRINGS ARE SHOWN IN UPPER CASE. SYNTACTIC VARIABLES ARE SHOWN IN LOWER-CASE AND ENCLOSED IN ANGLE BRACKETS (<>). OPTIONAL PARAMETERS ARE ENCLOSED IN SQUARE BRACKETS ([]). DEFAULTS, WHICH OCCUR IF THE DIRECTIVE IS NOT SPECIFIED OR IF A PARAMETER IS OMITTED, ARE UNDERLINED. THE CONFIGURATION DIRECTIVES CAN APPEAR IN THE CONFIGURATION DATA FILE IN ANY ORDER WITH THE EXCEPTION OF THE 'GO' DIRECTIVE, WHICH MUST BE THE LAST DIRECTIVE IN THE CONFIGURATION DATA FILE.

ALL NUMERIC PARAMETERS ARE IN OCTAL UNLESS OTHERWISE SPECIFIED.

ALTDEV -- SPECIFY ALTERNATE PAGING DEVICE AND SIZE

ALTDEV <DVNO> [<RECORDS>]

<DVNO> IS THE DEVICE NUMBER OF THE DISK TO BE USED AS AN ALTERNATE PAGING DEVICE. A <DVNO> OF 0 IS NOW ACCEPTABLE. THIS DIRECTIVE CORRESPONDS TO THE OLD-STYLE CONFIG PARAMETER 4/<DVNO>.

THE OPTIONAL PARAMETER <RECORDS> SPECIFIES THE SIZE OF THE ALTERNATE PAGING DEVICE. <RECORDS> IS INTERPRETED AS A 16-BIT POSITIVE INTEGER AND MUST BE GREATER THAN ZERO. IF THE <RECORDS> PARAMETER IS ALSO SPECIFIED ON THE PAGDEV DIRECTIVE, THE SUM OF THE TWO <RECORDS> PARAMETERS IS USED TO CALCULATE NSEG -- THE TOTAL NUMBER OF SEGMENTS IN THE SYSTEM.

NOTE: THE ALTERNATE PAGING DEVICE WILL BE USED FOR PAGING ONLY IF THE SIZE OF THE PRIMARY PAGING DEVICE (PAGDEV) IS SET WITH THE <RECORDS> PARAMETER -- SEE DESCRIPTION OF PAGDEV DIRECTIVE.

AMLBUF -- SET TERMINAL I/O BUFFER SIZES

AMLBUF <LINE> [<IBUFSZ>] [<OBUFSZ>] [<DMQSIZ>]

THE TERMINAL INPUT AND OUTPUT BUFFERS FOR AMLC LINE NUMBER <LINE> ARE SET TO THE NUMBER OF WORDS GIVEN BY <IBUFSZ> AND <OBUFSZ>. FOR SYSTEMS WITH DMQ AMLC CONTROLLERS, <DMQSIZ> CAN BE USED TO SPECIFY THE SIZE OF THE DMQ BUFFER FOR THE LINE. OMITTING <IBUFSZ>, <OBUFSZ>, OR <DMQSIZ> OR SPECIFYING 0 WILL RESULT IN NO CHANGE TO THE DEFAULT BUFFER SIZE. A 'TERMINAL I/O BUFFERS TOO LARGE' MESSAGE WILL BE PRINTED IF THE TOTAL

SIZE OF THE I/O BUFFERS (NOT INCLUDING THE DMQ BUFFER SIZES) IS MADE TO EXCEED 32K WORDS. A 'BAD LINE # IN AMLBUF CMND' MESSAGE WILL BE PRINTED IF <LINE> IS LESS THAN 0 OR GREATER THAN THE NUMBER OF LINES CONFIGURED FOR THE SYSTEM. A 'BAD DMQ AMLC CONFIGURATION' MESSAGE WILL BE PRINTED IF A DMQ BUFFER SIZE THAT IS NOT A POWER OF 2 IS SPECIFIED OR IF THE TOTAL SIZE OF THE I/O BUFFERS PLUS DMQ BUFFERS EXCEEDS 64K WORDS. THE DEFAULT BUFFER SIZES ARE 200, 300, AND 40 (DECIMAL 128, 192, 32).

ASRATE -- SET SYSTEM CONSOLE BAUD RATE

ASRATE <CTRL>

<CTRL> SPECIFIES THE BAUD RATE OF THE SYSTEM CONSOLE AS FOLLOWS:

110	110 BAUD
1010	300 BAUD
2010	1200 BAUD
3410	9600 BAUD

THIS DIRECTIVE IS EQUIVALENT TO (AND WILL OVERRIDE) THE B-REGISTER SETTING OF *COLDS. THE DEFAULT VALUE IS 110. IF THE ASRATE DIRECTIVE IS OMITTED AND THE SYSTEM INCLUDES A SOC CONTROLLER THE SPEED OF THE SYSTEM CONSOLE (USER 1) WILL BE THE SAME AS IT WAS UNDER PRIMOS II. THIS IS NOT TRUE IF THE SYSTEM HAS AN OPTION-A CONTROLLER.

ASRBUF -- SET ASR TERMINAL I/O BUFFER SIZE

ASRBUF <LINE> [<IBUFSZ>] [<OBUFSZ>]

THE TERMINAL INPUT AND OUTPUT BUFFERS FOR THE ASR ARE SET TO THE NUMBER OF WORDS GIVEN BY <IBUFSZ> AND <OBUFSZ>. OMITTING <IBUFSZ> OR <OBUFSZ> OR SPECIFYING 0 WILL RESULT IN NO CHANGE TO THE DEFAULT BUFFER SIZE. A 'TERMINAL I/O BUFFERS TOO LARGE' MESSAGE WILL BE PRINTED IF THE TOTAL SIZE OF THE I/O BUFFERS (INCLUDING AMLC BUFFERS) EXCEEDS 32K WORDS. A 'BAD LINE # IN ASRBUF CMND' MESSAGE WILL BE PRINTED IF <LINE> IS NOT 0. DEFAULT BUFFER SIZES ARE 200 AND 300 (DECIMAL 128 AND 192).

COMDEV -- SPECIFY COMMAND DEVICE

COMDEV <DVNO>

<DVNO> SPECIFIES THE DEVICE ON WHICH THE SYSTEM UFD CMDNCO RESIDES. THE COMMAND DEVICE MUST BE SPECIFIED, EITHER WITH THE COMDEV DIRECTIVE OR WITH A CONFIG DIRECTIVE. THIS DIRECTIVE CORRESPONDS TO CONFIG PARAMETER 2/<DVNO>.

CONFIG -- SPECIFY CONFIGURATION PARAMETERS

```
\   CONFIG <NTUSR> <PAGDEV> <COMDEV> [<OTHER PARMS>]

\   WITH THE EXCEPTION OF <NODE> (WHICH IS NO LONGER A VALID
\   OLD-STYLE CONFIG DIRECTIVE), AN OLD-STYLE CONFIG DIRECTIVE CAN
\   BE INCLUDED ANYWHERE IN A CONFIGURATION DATA FILE.  (IT WILL
    NOT, HOWEVER, BE PRINTED ON THE SYSTEM CONSOLE AS IS THE
    CONFIG COMMAND IN C_PRMO UNLESS 'TYPOUT YES' IS IN EFFECT --
    SEE TYPOUT DIRECTIVE.)  A COMPLETE SPECIFICATION OF PARAMETERS
    FOR THE OLD-STYLE CONFIG COMMAND IS AS FOLLOWS:

        0/<NTUSR>     NUMBER OF TERMINAL USERS
        1/<PAGDEV>    PAGING DEVICE
        2/<COMDEV>    COMMAND DEVICE
        3/<MAXPAG>    NUMBER PAGES PHYSICAL MEMORY TO USE
        4/<ALTDEV>    ALTERNATE PAGING DEVICE
        5/<NAMLC>     NUMBER ASSIGNABLE AMLC LINES
        6/<NPUSR>     NUMBER PHANTOM USERS
        7/<NRUSR>     NUMBER REMOTE USERS (NEW AT REV 15)
        10/<SMLCON>   NON-ZERO => ENABLE SMLC
```

DISLOG -- SET DISCONNECT LOGOUT OPTION

```
    DISLOG YES
           NO

    IF 'YES' IS SPECIFIED, A LOGOUT WILL BE PERFORMED WHEN
    DISCONNECT OCCURS ON AN AMLC LINE.  THIS DIRECTIVE IS USED TO
    SET THE FIGCOM VARIABLE DLOGOT.  THE DEFAULT SETTING DOES NOT
    LOGOUT ON DISCONNECT.
```

ERASE -- SPECIFY SYSTEM DEFAULT ERASE CHARACTER

```
    ERASE [<CHAR>] [<OCTAL-VAL>]

    <CHAR> IS USED TO SET THE SYSTEM DEFAULT CHARACTER-ERASE
    CHARACTER.  THE CHARACTER CAN OPTIONALLY BE SPECIFIED AS
    <OCTAL-VAL>.  FOR EXAMPLE:

        ERASE A          IS EQUIVALENT TO:
        ERASE 301

    THIS DIRECTIVE IS USED TO SET THE FIGCOM VARIABLE DEFERA
    (DEFAULT VALUE IS '"').
```

\FAM -- SPECIFY FAM NETWORK CONFIGURATION

\ FAM <NODENAME> <NETTYPE>

\ FAM IS NO LONGER A SUPPORTED CONFIG DIRECTIVE AND ITS USE IS
\ ILLEGAL. USE THE NETCFG COMMAND TO SPECIFY FAM INFORMATION.
\ (SEE SEPERATE DOCUMENT DESCRIBING NETCFG.)

\FILUNT -- SPECIFY NUMBER OF SYSTEM FILE UNITS

\ FILUNT <RSVUNT> <MAXUNT> <TOTUNT>

\ THE FILUNT DIRECTIVE IS USED TO DEFINE THE NUMBER OF FILE
\ UNITS AVAILABLE TO A USER, AND TO PRIMOS. <RSVUNT> DEFINES
\ THE MAXIMUM NUMBER OF FILE UNITS GUARRANTEED TO BE AVAILABLE
\ TO EACH USER. <MAXUNT> DEFINES THE MAXIMUM NUMBER OF UNITS
\ ANY ONE USER MAY HAVE OPEN AT ONE TIME. <TOTUNT> DEFINES THE
\ TOTAL NUMBER OF UNITS THAT BE SIMULTANEOUSLY OPEN IN THE
\ SYSTEM. IF FILUNT IS NOT SPECIFIED IN THE CONFIGURATION FILE,
\ THE DEFAULTS ARE AS FOLLOWS:

\ <RSVUNT> 16
\ <MAXUNT> 64
\ <TOTUNT> 2048

\ THE MAXIMUM TOTAL NUMBER OF UNITS THAT MAY BE OPEN
\ SIMULTANEOUSLY BY ALL USERS IS 2048. <TOTUNT> MAY BE USED TO
\ REDUCE THIS NUMBER. BY REDUCING THE TOTAL NUMBER OF FILE UNIT
\ TABLE ENTRIES IN THE SYSTEM, THE EFFECT WILL BE TO REDUCE THE
\ AMOUNT OF VIRTUAL MEMORY USED BY THE FILE SYSTEM. PRIMOS DOES
\ ATTEMPT TO KEEP THE ACTUAL NUMRER OF FILE UNIT TABLE ENTRIES
\ IN USE TO A MINIMUM IN ORDER TO KEEP DOWN THE SIZE OF THE
\ WORKING SET. FOR EACH CONFIGURED USER, THREE FILE UNITS ARE
\ ALLOCATED AT COLD-START.

\ THE MAXIMUM NUMBER OF UNITS THAT ANY ONE USER MAY HAVE OPEN
\ SIMULTANEOUSLY IS 64. OF THE 64 UNITS, 2 ARE RESERVED FOR
\ EXCLUSIVE USE BY THE SYSTEM. <MAXUNT> MAY BE USED TO REDUCE
\ THIS NUMBER, BUT NOT BELOW 2. THE HIGHEST NUMBERED FILE UNIT
\ AVAILABLE IS "<MAXUNT> - 1". IT MAY BE DESIRABLE IN SPECIAL
\ CIRCUMSTANCES TO RESTRICT <MAXUNT> TO 16, THUS PROVIDING
\ COMPATABILITY WITH PRIMOS II AND PRIMOS III.

\ THE NUMBER OF FILE UNITS GUARANTEED TO BE AVAILABLE TO EACH
\ USER IS 16. <RSVUNT> MAY BE USED TO INCREASE OR DECREASE THIS
\ QUANTITY. SINCE THERE ARE NOT ENOUGH FILE UNIT TABLE ENTRIES
\ TO PERMIT ALL USERS TO HAVE 64 FILE UNITS OPEN SIMULTANEOUSLY
\ (64*64=4096), SRCH$$ MAY RETURN THE ERROR CODE E$FUIU (ALL
\ UNITS IN USE). IF MULTIPLE COOPERATING PROCESSES (USERS)
\ DEPEND ON HAVING A CERTAIN NUMBER OF FILE UNITS AVAILABLE, THE
\ POSSIBILITY OF A DEADLOCK EXISTS. <RSVUNT> SHOULD BE
\ SPECIFIED SO THAT THERE ARE SUFFICIENT UNITS AVAILABLE TO
\ PREVENT DEADLOCK. THAT IS, <TOTUNT> MUST BE GREATER THAN OR

\ EQUAL TO <RSVUNT>*N, WHERE "N" IS THE NUMBER OF CONFIGURED
\ USERS, AND <TOTUNT> IS LESS THAN OR EQUAL TO 2048.

GO -- MARK END OF CONFIGURATION FILE

GO

THE GO DIRECTIVE MARKS THE END OF THE CONFIGURATION DATA FILE. ANY SUBSEQUENT LINES IN THE CONFIGURATION FILE ARE IGNORED. THE CONFIGURATION DATA FILE MUST INCLUDE A GO DIRECTIVE.

KILL -- SPECIFY SYSTEM DEFAULT KILL CHARACTER

KILL [<CHAR>] [<OCTAL-VAL>]

<CHAR> IS USED TO SET THE SYSTEM DEFAULT LINE-KILL CHARACTER. THE CHARACTER CAN OPTIONALLY BE SPECIFIED AS <OCTAL-VAL>. THIS DIRECTIVE IS USED TO SET THE FIGCOM VARIABLE DEFKIL. THE DEFAULT WOULD BE SPECIFIED AS:

```
KILL ?      OR EQUIVALENTLY:
KILL 277
```

LOGLOG -- ALLOW LOGINS WHILE LOGGED IN

LOGLOG YES
NO

IF 'YES' IS SPECIFIED, THE LOGIN COMMAND WILL BE PERMITTED WHILE A USER IS LOGGED IN. IF 'NO' IS SPECIFIED, THE LOGIN COMMAND WILL BE INHIBITED WHILE A USER IS LOGGED IN. THIS DIRECTIVE IS USED TO SET THE FIGCOM VARIABLE LOGOVR. THE
\ DEFAULT SETTING ALLOWS LOGINS WHILE LOGGED IN. THE EXTERNAL
\ LOGIN PROGRAM (IF PRESENT) IS RUN ONLY ONCE IF A USER LOGS IN
\ WHILE ALREADY LOGGED IN (AND LOGLOG YES HAS BEEN SPECIFIED FOR
\ CONFIGURATION).

LOGMSG -- PRINT LOGIN/LOGOUT MESSAGES

LOGMSG YES
NO

THIS DIRECTIVE CONTROLS THE PRINTING OF LOGIN AND LOGOUT MESSAGES ON THE SYSTEM CONSOLE. 'YES' IS THE DEFAULT, WHICH CAUSES THE MESSAGES TO BE PRINTED. SPECIFYING 'NO' WILL CAUSE THE MESSAGES TO BE SUPPRESSED. THIS DIRECTIVE IS USED TO SET THE FIGCOM VARIABLE NLGPRT.

LOGREC -- SPECIFY MAXIMUM SIZE OF LOGREC FILE

LOGREC <VAL>

<VAL>, IF POSITIVE, SPECIFIES THE NUMBER OF WORDS IN THE LOGREC FILE. WHEN LOGREC EXCEEDS <VAL> WORDS, THE 'EXCEEDING QUOTA ON LOGREC' MESSAGE IS PRINTED AS EACH NEW ENTRY IS ADDED TO LOGREC. SPECIFYING AN <VAL> OF 0 WILL INHIBIT THE QUOTA CHECK; NO MESSAGE WILL EVER BE PRINTED. SPECIFYING A NEGATIVE <VAL> WILL SUPPRESS ALL ATTEMPTS TO WRITE TO THE LOGREC FILE. (THIS WILL AVOID DISK WRITE ERRORS IF RUNNING ON A WRITE-PROTECTED DISK.) THE DEFAULT VALUE IS 10000 (4096 DECIMAL). THIS DIRECTIVE IS USED TO SET THE VARIABLE LRQUOT IN FIGCOM.

LOUTQM -- SPECIFY INACTIVITY-LOGOUT QUANTUM

LOUTQM <MINS>

THIS DIRECTIVE SPECIFIES THE NUMBER OF MINUTES OF INACTIVITY TO BE ALLOWED TO PASS BEFORE A USER IS AUTOMATICALLY LOGGED OUT. THE DEFAULT VALUE IS 1750 (1000 DECIMAL) MINUTES. THIS DIRECTIVE IS USED TO SET THE FIGCOM VARIABLE LOUTQM. <MINS> MUST BE GREATER THAN ZERO.

MAXPAG -- SPECIFY NUMBER PAGES OF MEMORY TO VALIDATE

MAXPAG <NPAGES>

<NPAGES> IS THE NUMBER OF PAGES OF PHYSICAL MEMORY TO VALIDATE FOR USE. THE DEFAULT VALUE IS 400 (256 DECIMAL). THIS DIRECTIVE CORRESPONDS TO THE OLD-STYLE CONFIG PARAMETER 3/<NPAGES>. (MEMORY VALIDATION OCCURS AT COLD START. EACH PAGE IS 1024 WORDS.)

\MYNAME -- SPECIFY NETWORK NAME OF LOCAL NODE

\ MYNAME <NODENAME>

\ MYNAME IS NO LONGER A SUPPORTED CONFIG DIRECTIVE AND ITS USE
\ IS ILLEGAL. USE THE NETCFG COMMAND TO SPECIFY <NODENAME>
\ INFORMATION. (SEE SEPERATE DOCUMENT DESCRIBING NETCFG.)

NAMLC -- SPECIFY NUMBER ASSIGNABLE AMLC LINES

NAMLC <NLINES>

<NLINES> SPECIFIES THE NUMBER OF ASSIGNABLE AMLC LINES IN THE SYSTEM. THIS DIRECTIVE CORRESPONDS TO THE OLD-STYLE CONFIG PARAMETER 5/<NLINES>. THE DEFAULT VALUE IS 0.

\NET -- SPECIFY NETWORK CONFIGURATION

\ NET ON

\ THIS DIRECTIVE SPECIFIES THAT NETWORKS ARE TO BE CONFIGURED. IF THIS DIRECTIVE IS NOT SPECIFIED, THEN NETWORKS WILL NOT BE CONFIGURED. THE PREVIOUS QUALIFIERS OF THIS DIRECTIVE ARE NO LONGER SUPPORTED AND ARE ILLEGAL. (SEE SEPARATE DOCUMENT ON NETCFG.)

NPUSR -- SPECIFY NUMBER OF PHANTOM USERS

NPUSR <N>

<N> SPECIFIES THE NUMBER OF PHANTOM USERS TO BE CONFIGURED. IT IS ADDED TO NTUSR AND NRUSR TO DETERMINE THE TOTAL NUMBER OF USERS ON THE SYSTEM. THIS DIRECTIVE CORRESPONDS TO THE OLD-STYLE CONFIG PARAMETER 6/<N>. THE DEFAULT IS 0.

NRUSR -- SPECIFY NUMBER REMOTE USERS

NRUSR <N>

<N> SPECIFIES THE NUMBER OF PROCESSES TO BE RESERVED FOR REMOTE LOGINS (THE DEFAULT NUMBER IS 0). THE NRUSR DIRECTIVE ALLOWS UP TO <N> CONCURRENT REMOTE USERS TO CONNECT TO THIS SYSTEM USING THE -ON KEYWORD OF THE LOGIN COMMAND (MAXIMUM VALUE IS 40 -- DECIMAL 32). THE NUMBER OF REMOTE USERS IS ADDED TO NPUSR AND NTUSR TO DETERMINE THE TOTAL NUMBER OF USERS ON THE SYSTEM.

\NSEG -- SPECIFY NUMBER AVAILABLE SEGMENTS IN SYSTEM

\ NSEG <NUMBER>

\ THIS DIRECTIVE SETS THE TOTAL VIRTUAL ADDRESS SPACE FOR A SYSTEM (THE VARIABLE NSEG IN SEGMENT 4). <NUMBER> SPECIFIES THE NUMBER OF PAGE MAPS TO BE ALLOCATED DURING SYSTEM INITIALIZATION. THERE MAY BE FEWER PAGE MAPS AVAILABLE THAN THE NUMBER OF POSSIBLE USER SEGMENTS. THUS, ALTHOUGH A 64 USER SYSTEM CAN ALLOW 64 POSSIBLE SEGMENTS TO BE ADDRESSED BY EACH USER, THERE IS A LIMIT OF <NUMBER> SEGMENTS WHICH CAN ACTUALLY BE IN USE BY ALL USERS AT ANY GIVEN TIME. THE SYSTEM ALLOWS A MAXIMUM OF 320 DECIMAL (500 OCTAL) PAGE MAPS. THE DEFAULT VALUE OF <NUMBER> IS 192 DECIMAL (300 OCTAL).

IF THE AMOUNT OF PAGING SPACE SPECIFIED IN THE PAGDEV AND ALTDEV DIRECTIVES WILL NOT PERMIT NSEG SEGMENTS TO BE ALLOCATED, NSEG IS REDUCED TO CONFORM WITH THE AMOUNT OF PAGING SPACE AVAILABLE. (SEE ALSO THE ALTDEV AND PAGDEV DIRECTIVES.)

NTUSR -- SPECIFY NUMBER OF TERMINAL USERS

NTUSR <N>

<N> SPECIFIES THE NUMBER OF TERMINAL USERS TO BE CONFIGURED. THE NUMBER OF USERS MUST BE SPECIFIED, EITHER WITH THE NTUSR DIRECTIVE OR WITH THE CONFIG COMMAND. THIS DIRECTIVE \ CORRESPONDS TO THE OLD-STYLE CONFIG PARAMETER 0/<N>. NTUSR \ MUST BE GREATER THAN 1 AND LESS THAN 65. NTUSR IS ADDED TO \ NPUSR AND NRUSR TO DETERMINE THE TOTAL NUMBER OF USERS ON THE \ SYSTEM.

\NUSEG -- SET NUMBER OF USER SEGMENTS PER USER

\ NUSEG <NUMBER>

\ THIS DIRECTIVE SETS THE SIZE OF THE VIRTUAL ADDRESS SPACE FOR \ EACH USER BY SETTING THE SIZE OF EACH PROCESS' DESCRIPTOR \ TABLE 2. <NUMBER> SPECIFIES (IN OCTAL) THE NUMBER OF SEGMENTS \ AVAILABLE TO EACH USER PROCESS. THE PRIMOS IV SYSTEM RESERVES \ ROOM FOR A TOTAL OF 4096 USER SEGMENTS. THEREFORE, THE \ PRODUCT OF <NUMBER> TIMES THE TOTAL NUMBER OF USERS (INCLUDING \ PHANTOMS AND REMOTE LOGIN USERS) CANNOT EXCEED 4096. THE \ DEFAULT VALUE OF <NUMBER> IS 32 DECIMAL (40 OCTAL).

PAGDEV -- SPECIFY PAGING DEVICE AND SIZE

PAGDEV <DVNO> [<RECORDS>]

<DVNO> SPECIFIES THE PHYSICAL DISK ON WHICH PAGING IS TO TAKE PLACE. THE PAGING DEVICE MUST BE SPECIFIED, EITHER WITH THE PAGDEV DIRECTIVE OR WITH THE CONFIG COMMAND. THIS DIRECTIVE CORRESPONDS TO THE OLD-STYLE CONFIG PARAMETER 1/<DVNO>.

THE OPTIONAL PARAMETER <RECORDS> IS USED TO SPECIFY THE SIZE OF THE PAGING DISK. IT IS INTERPRETED AS A 16-BIT POSITIVE INTEGER AND MUST BE GREATER THAN ZERO. SPECIFYING <RECORDS> HAS TWO CONSEQUENCES. FIRST, <RECORDS>, POSSIBLY IN CONJUNCTION WITH A <RECORDS> SPECIFICATION ON AN ALTDEV DIRECTIVE, IS USED TO LIMIT NSEG -- THE TOTAL NUMBER OF SEGMENTS IN THE SYSTEM. SECOND, IF AN ALTERNATE PAGING DEVICE HAS BEEN SPECIFIED (ALTDEV), <RECORDS> WILL DEFINE THE POINT AT WHICH PAGE SPACE ALLOCATION SWITCHES FROM THE PRIMARY TO THE ALTERNATE PAGING DEVICE.

NOTE: <RECORDS> CAN BE AS SMALL AS 1 TO FORCE ALMOST ALL PAGING TO OCCUR ON THE ALTERNATE PAGING DEVICE. THE PRIMARY DEVICE, HOWEVER, WILL ALWAYS BE USED TO PAGE THE SEGMENTS USED BY PRIMOS IV (SEGMENT NUMBERS 0-12 AND USER 1'S SEGMENT 6000).

PREPAG -- SPECIFY NUMBER OF PAGES TO PREPAGE

```
PREPAG <N>

<N> SPECIFIES THE NUMBER OF PAGES TO PREPAGE OUT WHEN A PAGE
FAULT OCCURS. THE DEFAULT VALUE IS 3. THIS DIRECTIVE SETS
THE VARIABLE PREPGK IN PAGCOM.
```

\RLOGIN -- SPECIFY REMOTE LOGIN NETWORK CONFIGURATION

```
\   RLOGIN <NODENAME> <NETTYPE>

\   RLOGIN IS NO LONGER SUPPORTED AS A CONFIG DIRECTIVE AND ITS
\   USE IS ILLEGAL. USE THE NETCFG COMMAND TO SPECIFY REMOTE
\   LOGIN INFORMATION. (SEE SEPERATE DOCUMENT DESCRIBING NETCFG.)
```

RWLOCK -- SPECIFY FILE SYSTEM READ/WRITE LOCK SETTING

```
RWLOCK <VAL>

<VAL> IS USED TO SET THE FIGCOM VARIABLE RWLOCK -- THE
SYSTEM-WIDE FILE READ/WRITE LOCK. VALID VALUES OF <VAL> ARE:

0 - 1 READER   OR 1 WRITER (WRITER HAS EXCLUSIVE CONTROL)
1 - N READERS  OR 1 WRITER (WRITER HAS EXCLUSIVE CONTROL)
3 - N READERS AND 1 WRITER
5 - N READERS AND N WRITERS

THE DEFAULT SETTING OF RWLOCK IS 1.

\   NOTE: MANY SUBSYSTEMS (SUCH AS SPOOL, CX, ETC.) DO NOT
\   PERMIT MULTIPLE WRITERS.
```

SMLC -- ENABLE AND CONFIGURE SMLC LINES

```
SMLC ON
SMLC CNTRLR <CTRLR-NUMBER> <DEVADR>
SMLC SMLCNN <CTRLR-NUMBER> <LINE-NUMBER>

SMLC DIRECTIVES ARE USED TO ENABLE AND CONFIGURE SMLC LINES.
SPECIFYING 'ON' ENABLES THE SMLC IN THE DEFAULT CONFIGURATION.
THIS CORRESPONDS TO THE OLD-STYLE CONFIG SPECIFICATION 10/1.
THE DEFAULT VALUE LEAVES THE SMLC DISABLED.

THE SMLC CNTRLR FORM IS USED TO SPECIFY THE PHYSICAL DEVICE
NUMBER(S) OF THE SMLC CONTROLLERS. <CTRLR-NUMBER> IS 0 OR 1;
<DEVADR> IS THE PHYSICAL DEVICE ADDRESS OF THE SPECIFIED
CONTROLLER NUMBER. DEFAULT VALUES FOR CONTROLLER ADDRESSES
ARE CONTROLLER 0 AT 50 AND CONTROLLER 1 UNDEFINED.

THE SMLC SMLCNN FORM IS USED TO MAP LOGICAL LINE NUMBERS
(SMLC00-SMLC03) ONTO PHYSICAL CONTROLLERS AND LINE NUMBERS.
```

<CTRLR-NUMBER> IS AS FOR THE THE SMLC CNTRLR DIRECTIVE;
<LINE-NUMBER> IS THE PHYSICAL LINE NUMBER ON THE CONTROLLER FROM 0 TO 3. THE DEFAULT VALUES MAP SMLC00-SMLC03 ONTO CONTROLLER 0, PHYSICAL LINES 0-3.

TYPOUT -- CONTROL PRINTING OF CONFIGURATION COMMANDS

TYPOUT YES
NO

PRINTING OF THE CONFIGURATION DIRECTIVES ON THE SYSTEM CONSOLE IS UNDER THE CONTROL OF THE TYPOUT DIRECTIVE. SPECIFYING 'YES' WILL CAUSE THE DIRECTIVES TO BE PRINTED AS THEY ARE PROCESSED. THE DEFAULT OR ANY OTHER SPECIFICATION WILL CAUSE PRINTING OF THE DIRECTIVES TO BE SUPPRESSED. (SEVERAL TYPOUT DIRECTIVES CAN BE USED TO PRINT SELECTED CONFIGURATION DIRECTIVES.)

7.3 PRIMOS IV INITIALIZATION ERROR MESSAGES

THE FOLLOWING LISTS ALL ERROR MESSAGES GENERATED BY THE PRIMOS IV PRELOADER ('PRIMOS') AND THE PRIMOS IV AND NETWORK INITIALIZATION \SEQUENCES. THE MAJORITY OF THE CONFIG MESSAGES ARE FATAL, AND \CAUSE CONFIGURATION TO TERMINATE. ANY ERROR MESSAGES WHICH DO \NOT COME FROM THE PRELOADER ('PRIMOS'), REQUIRE THAT PRIMOS II BE \BOOTED AGAIN FROM THE CONTROL PANEL (I.E., START OVER FROM THE \BEGINNING).

7.3.1 PRELOADER ('PRIMOS') ERROR MESSAGES

<FILE-SYSTEM-ERROR-MESSAGE> CMDNCO (PRIMOS)

A FILE SYSTEM ERROR WAS ENCOUNTERED BY THE PRELOADER WHILE ATTEMPTING TO ATTACH TO CMDNCO.

<FILE-SYSTEM-ERROR-MESSAGE> C_PRMO (PRIMOS)

A FILE SYSTEM ERROR (OTHER THAN FILE NOT FOUND) WAS ENCOUNTERED BY THE PRELOADER WHILE ATTEMPTING TO OPEN THE FILE C_PRMO FOR COMMAND INPUT.

FIRST COMMAND MUST BE CONFIG

THE COMMAND TYPED IN RESPONSE TO THE 'PLEASE ENTER CONFIG' PROMPT OR THE FIRST EXECUTABLE COMMAND IN C_PRMO IS NOT THE EXTERNAL COMMAND CONFIG.

<FILE-SYSTEM-ERROR-MESSAGE> <CONFIG-FILE> (PRIMOS)

A FILE SYSTEM ERROR WAS ENCOUNTERED BY THE PRELOADER WHILE ATTEMPTING TO OPEN THE CONFIGURATION FILE <CONFIG-FILE>.

\ MISSING NTUSR, PAGDEV, OR COMDEV

\ THE CONFIGURATION DATA FILE DID NOT SPECIFY THESE REQUIRED \ PARAMETERS.

ILLEGAL PAGDEV

THE DEVICE SPECIFIED FOR PAGING IS NOT A LEGAL PAGING DEVICE.

USE <DVNO> FOR PAGING?

THE DISK <DVNO> HAS BEEN SPECIFIED AS THE PAGING DEVICE, BUT IS FORMATTED AS A STANDARD PRIMOS DISK. A REPLY OF 'YES' IS REQUIRED TO ENABLE PAGING ACTIVITY ON <DVNO>.

ILLEGAL COMDEV

THF DEVICE SPECIFIED FOR THE COMMAND DEVICE IS NOT LEGAL.

ILLEGAL ALTDEV

THE DEVICE SPECIFIED AS THE ALTERNATE PAGING DEVICE IS NOT LEGAL.

<FILE-SYSTEM-ERROR-MESSAGE> PRNNNN (PRIMOS)

A FILE SYSTEM ERROR WAS ENCOUNTERED BY THE PRELOADER WHILE ATTEMPTING TO OPEN OR READ THE INDICATED PRNNNN FILE.

END OF FILE. MISSING 'GO' CMND (PRIMOS)

THE CONFIGURATION DATA FILE DOES NOT INCLUDE A GO DIRECTIVE AS REQUIRED.

TPIOS ERROR

AN I/O ERROR OCCURRED WHILE PRELOADING THE PAGING DEVICE.

7.3.2 PRIMOS IV INITIIALIZATION ERROR MESSAGES

NTUSR+NPUSR+NRUSR TOO BIG (AINIT)

THE NUMBER OF TERMINAL PLUS PHANTOM PLUS REMOTE USERS EXCEEDS THE MAXIMUM NUMBER OF CONFIGURABLE USERS.

NRUSR INVALID (AINIT)

THE NUMBER OF REMOTE USERS SPECIFIED BY AN NRUSR DIRECTIVE EXCEEDS THE MAXIMUM NUMBER OF CONFIGURABLE REMOTE USERS (40, DECIMAL 32).

NTUSR, NPUSR, OR NRUSR INVALID (AINIT)

THE VALUE OF NTUSR, NPUSR, OR NRUSR IS INCORRECT.

SEEK FAILURE ON PAGDEV (AINIT)

THE INITIAL SEEK TO CYLINDER 0 ON THE PAGING DEVICE FAILED.

SEEK FAILURE ON ALTDEV (AINIT)

THE INITIAL SEEK TO CYLINDER 0 ON THE ALTERNATE PAGING DEVICE FAILED.

<FILE-SYSTEM-MESSAGE> CAN'T ATTACH TO CMDNC0 (AINIT)

A FILE SYSTEM ERROR WAS ENCOUNTERED WHILE ATTEMPTING TO ATTACH TO CMDNC0 FOR USER 1.

BAD CONFIG COMMAND: <XXXXXX> (AINIT)

THE DIRECTIVE <XXXXXX> IN THE CONFIGURATION DIRECTIVE FILE IS NOT A RECOGNIZED CONFIGURATION DIRECTIVE.

BAD <CMND> PARAMETER (AINIT)

ONE OR MORE OF THE PARAMETERS SPECIFIED FOR THE CONFIGURATION DIRECTIVE <CMND> IS INVALID.

BAD LINE # IN AMLBUF CMND (AINIT)

AN AMLBUF DIRECTIVE SPECIFIES AN INVALID LINE NUMBER.

BAD DMQ AMLC CONFIGURATION (AINIT)

A DMQ BUFFER SIZE IN AN AMLBUF DIRECTIVE WAS TOO LARGE OR NOT EQUAL TO A POWER OF 2.

BAD LINE # IN ASRBUF CMND (AINIT)

AN ASRBUF DIRECTIVE SPECIFIED AN INVALID LINE NUMBER.

\ FILUNT INVALID (AINIT)

\ THE FILUNT DIRECTIVE SPECIFIES INCORRECT INFORMATION FOR
\ PROPER CONFIGURATION.

TERMINAL I/O BUFFERS TOO LARGE (AINIT)

THE TOTAL SIZE OF THE TERMINAL I/O BUFFERS EXCEEDS 32K WORDS.

SMLC CTRLR # OUT OF RANGE (AINIT)

AN SMLC DIRECTIVE SPECIFIES AN INVALID CONTROLLER NUMBER.

SMLC LINE # OUT OF RANGE (AINIT)

AN SMLC DIRECTIVE SPECIFIES AN INVALID LINE NUMBER.

RESTART PLEASE

THIS MESSAGE APPEARS FOLLOWING ANY ERROR MESSAGE PRINTED BY THE PRIMOS IV INITIALIZATION LOGIC (AINIT). THE SYSTEM WILL HALT THE LOCATION BOOTO_ IN SEGMENT 6. PRIMOS II MUST BE RELOADED. THE OFFENDING DIRECTIVE IN THE CONFIGURATION DATA FILE MUST BE CORRECTED.

\7.3.3 NETWORK INITIALIZATION ERROR MESSAGES

\ NETWORK NOT CONFIGURED (AINIT)

THIS MESSAGE IS NO LONGER ISSUED.

\ <FILE-SYSTEM-ERROR-MESSAGE> NETCON (NETFIG)

\ A FILE SYSTEM ERROR HAS OCCURRED WHILE OPENING OR READING
\ THE NETWORK CONFIGURATION FILE.

\ BAD NETWORK CONFIGURATION FILE FORMAT (NETFIG)

\ THE NETWORK CONFIGURATION FILE HAS AN ILLEGAL FORMAT.
\ RECREATE THE NETWORK CONFIGURATION FILE USING THE MOST
\ RECENT VERSION OF NETCFG.

\ NO TABLE SYSGEN'D FOR RING # <N> (NETFIG)
\ NO TABLE SYSGEN'D FOR IPC #<N> (NETFIG)
\ NO TABLE SYSGEN'D FOR SMLC #<N> (NETFIG)

\ THERE ARE TOO MANY NODES OF THE SPECIFIED LINE TYPE.
\ RECREATE THE NETWORK CONFIGURATION FILE SPECIFYING FEWER
\ NODES OF THAT TYPE.

\ TOO MANY NETWORK NODES
\ ONLY <N> NODES ALLOWED (NETFIG)

\ THERE ARE TOO MANY TOTAL NODES CONFIGURED. RECREATE THE
\ NETWORK CONFIGURATION FILE SPECIFYING FEWER TOTAL NODES.

\ WARNING -- <REVISION TEXT> NETWORK CONFIGURATION FILE (NETFIG)

\ THE NETWORK CONFIGURATION FILE WAS CREATED FOR A PREVIOUS
\ VERSION OF THE OPERATING SYSTEM. IF THERE ARE NO
\ SUBSEQUENT ERRORS THEN NETWORKS HAVE BEEN CONFIGURED
\ SUCCESSFULLY. IN ANY CASE THE NETWORK CONFIGURATION FILE
\ SHOULD BE RECREATED WITH THE MOST RECENT VERSION OF NETCFG.

8 MAPGEN - A TOOL FOR BUILDING PRIMOS IV

INTRODUCTION

THE MAPGEN TOOL IS A UTILITY PROGRAM EMPLOYED TO BUILD THE PRIMOS IV OPERATING SYSTEM. THIS UTILITY TAKES INFORMATION AVAILABLE FROM THE SYSTEM LOAD (PERFORMED USING SEG) AND CONSTRUCTS THE INITIAL PAGING MAPS, SEGMENT DESCRIPTORS, AND OTHER DATA NECESSARY TO THE PAGING MANAGEMENT WITHIN THE SYSTEM. THE UTILITY WILL ALSO BUILD THE INITIAL COLD-START R-MODE OBJECT FILE THAT IS EXECUTED WHEN BOOTING THE SYSTEM.

NOTE: MAPGEN, ITS DATABASES, ANY DATABASES IT ACCESSES, AND ERROR MESSAGES ARE SUBJECT TO CHANGE AT ANY REVISION OF PRIMOS.

CONVENTIONS

TWO CONVENTIONS ARE ADOPTED IN THIS SECTION FOR THE NOTATION WHICH DESCRIBES THE DIRECTIVES. THESE CONVENTIONS ARE: 1) ALL NUMERIC VALUES SPECIFIED TO THE MAPGEN PROGRAM ARE OCTAL BASED, AND 2) DIRECTIVES MAY BE ABBREVIATED TO A LEFT-MOST UNIQUE STRING WHICH IS UNDERLINED IN THE SYNTAX DEFINITIONS.

OPERATION

THE MAPGEN PROGRAM BUILDS THE INITIAL PAGING DATABASE WITH INFORMATION FROM DIRECTIVES SUPPLIED BY THE USER AND OTHER INFORMATION AVAILABLE FROM THE SYSTEM LOAD. THE DIRECTIVES THAT THE USER MAY SUPPLY ARE OF THREE BASIC TYPES:

A) SEGMENT DESCRIPTIONS
B) INFORMATION STORE
C) SUPPORT

EACH OF THESE THREE TYPES OF DIRECTIVES ARE DISCUSSED IN THE SECTIONS WHICH FOLLOW.

8.1 SEGMENT DESCRIPTION DIRECTIVES

THE SEGMENT DESCRIPTION DIRECTIVES TELL THE MAPGEN UTILITY ABOUT THE INITIAL SEGMENTS AVAILABLE IN THE PRIMOS OPERATING SYSTEM. THE USER MUST PROVIDE ALL THIS INFORMATION BEFORE USING ANY OF THE OTHER TYPES OF DIRECTIVES (WITH THE EXCEPTION OF THE QUIT AND TABLE DIRECTIVES). THE USER FIRST ESTABLISHES A SEGMENT TO BE DEFINED. ONE DOES THIS BY USING THE SEGMENT DIRECTIVE WHOSE SYNTAX IS GIVEN BELOW.

SEGMENT <SEGNO> <FILE>
*

THE <SEGNO> ARGUMENT IS THE NUMBER OF THE SEGMENT BEING DEFINED. THE <FILE> ARGUMENT IS THE TREENAME OF A R-MODE OBJECT FILE TEMPLATE WHICH IS TO BE INITIALLY LOADED INTO THE SPECIFIED SEGMENT. IF NO OBJECT FILE IS TO BE LOADED, USER MUST SPECIFY AN ASTERISK (*) IN PLACE OF THE <FILE> ARGUMENT.

THE EFFECT OF THE SEGMENT DIRECTIVE IS TO CAUSE ALLOCATION OF PAGE-MAP FOR THIS SEGMENT. IF A <FILE> ARGUMENT WHICH IS NOT AN ASTERISK IS GIVEN, THE PAGES DEFINED WITHIN THE LOAD RANGE OF THE R-MODE OBJECT FILE ARE DEFINED IN THE PAGE-MAP AS BEING REFERENCABLE AND PRELOADED. ALL PAGES OUTSIDE OF THE OBJECT FILE RANGE WILL BE UNREFERENCABLE. THE USE OF AN ASTERISK IN PLACE OF THE <FILE> ARGUMENT WILL CAUSE ALL OF THE PAGES OF THE SEG TO BE UNREPLACABLE. EACH OF THE ATTRIBUTES ASSOCIATED WITH PAGES WITHIN THE SEGMENT BEING DEFINED MAY BE ALTERED USING THE ATTRIBUTE MODIFIERS DESCRIBED BELOW.

TO PROVIDE FOR SPECIAL CHARACTERISTICS OF SOME SEGMENTS IN THE OPERATING SYSTEM (E.G., PAGES 'WIRED' TO MEMORY), A SET OF DIRECTIVES MAY APPEAR AFTER THE SEGMENT DIRECTIVE. THESE DIRECTIVES ARE CALLED ATTRIBUTE MODIFIERS. EACH SUCH DIRECTIVE APPLIES ONLY TO THE SEGMENT DEFINED BY THE PREVIOUS SEGMENT DIRECTIVE. THE DIRECTIVES ALL CONTAIN AN ADDRESS RANGE WITHIN THE SEGMENT. THE BNF SYNTAX OF A <RANGE> ARGUMENT IS GIVEN BELOW:

```
<RANGE>  ::= <FADDR> <LADDR> \ <FADDR> <LADDR> <OFFSET> \ *
<FADDR>  ::= <SYMBOL> \ <NUMBER> \ $LOW
<LADDR>  ::= <SYMBOL> \ <NUMBER> \ $HIGH
<OFFSET> ::= <NUMBER>
```

THE IDEA OF A <RANGE> ARGUMENT IS THAT IT INDICATE THE LOWER AND UPPER BOUNDS OF PAGES FOR WHICH THE ATTRIBUTE MODIFIER IS TO APPLY. THE <FADDR> ARGUMENT SPECIFIES THE FIRST ADDRESS FOR WHICH THIS MODIFIER APPLIES. THE <LADDR> ARGUMENT SPECIFIES THE LAST ADDRESS PLUS_ONE FOR WHICH THIS MODIFIER APPLIES. THE <SYMBOL> ARGUMENT MAY BE ANY EXTERNAL SYMBOL (I.E., ALPHANUMERIC STRING) FROM THE LOAD. THE EFFECT OF THE <SYMBOL> ARGUMENT IS THAT THE ADDRESS ASSOCIATED WITH THE SYMBOL IS USED. THE STRINGS $LOW AND $HIGH REPRESENT THE ADDRESSES FOR THE BEGINING AND THE END (PLUS ONE), RESPECTIVELY, OF THE R-MODE OBJECT FILE SPECIFIED IN THE PREVIOUS SEGMENT DIRECTIVE. THE OPTIONAL <OFFSET> ARGUMENT MAY BE A VALUE WHICH IS ADDED TO THE <LADDR> ARGUMENT IN DETERMINING THE END OF THE RANGE. SPECIFYING AN ASTERISK AS THE RANGE INDICATES THAT THE RANGE FROM THE PREVIOUS MODIFIER DIRECTIVE IS TO BE USED FOR THIS DIRECTIVE ALSO.

THE ATTRIBUTE MODIFIER DIRECTIVES AND THEIR MEANING ARE LISTED BELOW. THESE DIRECTIVES MAY APPEAR IN ANY ORDER WITHIN THE SCOPE OF THE SEGMENT DIRECTIVE FOR WHICH THEY APPLY. IN ADDITION, IF THE MEANING OF ONE MODIFIER CONTRADICTS THE

MEANING OF A MODIFIER WHICH HAS ALREADY APPEARED FOR ANY GIVEN RANGE, ONLY THE MEANING OF THE LAST MODIFIER SHALL APPLY.

NOTE: THE SCOPE OF A SEGMENT DIRECTIVE IS TERMINATED BY THE OCCURENCE OF ANY DIRECTIVE WHICH IS NOT AN ATTRIBUTE MODIFIER.

RESIDE -- SPECIFY RANGE IN COLD-START MODULE

RESIDE <RANGE>

THE SPECIFIED RANGE IS RESIDENT IN THE COLD-START R-MODE OBJECT FILE.

WIRE -- SPECIFY RANGE LOCKED IN COLD-START MODULE

WIRE <RANGE> -OR- LOCK <RANGE>

THE SPECIFIED RANGE IS RESIDENT IN THE COLD-START R-MODE OBJECT FILE AND THE RESPECTIVE WIRE-BITS OF THE PAGE-MAP ARE TURNED ON.

ONE -- SPECIFY IDENTICAL VIRTUAL/PHYSICAL ADDRESS

ONE <RANGE>

THE SPECIFIED RANGE IS RESIDENT IN THE COLD-START R-MODE OBJECT FILE AND IT IS PLACED AT SUCH A LOCATION IN THAT FILE SUCH THAT BOTH THE VIRTUAL- AND PHYSICAL-ADDRESSES ARE IDENTICAL.

PAGE -- SPECIFY RANGE FOR PAGING SPACE

PAGE <RANGE>

THE SPECIFIED RANGE IS ALLOCATED PAGING SPACE ON THE DISK AND THE 'NO COPY'-BIT IN THE PAGE-MAP IS TURNED ON.

EMPTY -- SPECIFY RANGE FOR NO PAGING SPACE

EMPTY <RANGE>

THE SPECIFIED RANGE IS NOT ALLOCATED ANY PAGING SPACE ON THE DISK.

LOAD -- SPECIFY RANGE FOR PRELOADING

LOAD <RANGE>

THE SPECIFIED RANGE IS ALLOCATED PAGING SPACE ON THE DISK AND THE 'NO COPY'-BIT IN THE PAGE-MAP IS TURNED OFF INDICATING THAT THE RANGE IS PRELOADED ON THE DISK.

SHARE -- SPECIFY RANGE NOT CACHEABLE

SHARE <RANGE>

THE SPECIFIED RANGE WILL HAVE THE HARDWARE SHARE-BIT IN THE PAGE-MAP TURNED ON INDICATING THAT THE MEMORY IS NOT CACHEABLE.

8.2 INFORMATION STORE DIRECTIVES

THE INFORMATION STORE DIRECTIVES CAUSE SOME INFORMATION GIVEN BY THE SEGMENT DESCRIPTION DIRECTIVES TO BE STORED WITHIN THE R-MODE OBJECT FILE TEMPLATES USED IN PRELOADING THE PAGING DISK. THE USER MUST SPECIFY A LOCATION FOR EACH OF THESE DIRECTIVES THAT DETERMINES WHERE THE DATA IS TO BE STORED. THE BNF SYNTAX OF THE <LOCATION> ARGUMENT IS GIVEN BELOW.

<LOCATION> ::= <SYMBOL> \ <ADDRESS> <SEGNO>

THE <SYMBOL> ARGUMENT MAY BE ANY SYMBOL THAT WAS RECOGNIZED USING THE TABLE DIRECTIVE (I.E., ANY EXTERNAL SYMBOL NAME). THE USE OF THE <ADDRESS> AND <SEGNO> ARGUMENTS SPECIFIES AN OFFSET WITHIN A SEGMENT AND A SEGMENT NUMBER WHERE THE INFORMATION IS TO BE STORED.

EACH OF THESE DIRECTIVES WILL CAUSE ONLY THE AMOUNT OF INFORMATION THAT HAS BEEN SPECIFIED TO BE STORED, THAT IS, ONLY PART OF THE DATA-BASE FOR A PARTICULAR DIRECTIVE IS STORED. HENCE THE DIRECTIVE WILL MODIFY ONLY AS MUCH OF A DATA-BASE IS AS DEFINED BY THE SEGMENT DESCRIPTION DIRECTIVES AND NO MORE. THIS ASSUMES THAT THE USER HAS INITIALIZED THE REMAINDER OF THE DATA-BASES TO THE APPROPRIATE VALUES AND THAT SUFFICIENT SPACE EXISTS IN THE DATA-BASE FOR STORING THE DEFINED INFORMATION.

IN EACH OF THE DIRECTIVES, THE DATA IS UPDATED WITHIN AN R-MODE OBJECT FILE. THIS OBJECT FILE IS THE SAME AS THAT GIVEN IN THE <FILE> ARGUMENT SPECIFIED IN THE SEGMENT DIRECTIVE FOR THE SEGMENT NUMBER ASSOCIATED WITH THE LOCATION. IF AN ASTERISK WAS SPECIFIED FOR THE <FILE> ARGUMENT OR THE SEGMENT WAS NOT DEFINED, AN ERROR MESSAGE IS ISSUED. WHEN <SYMBOL> IS SPECIFIED AS THE LOCATION, MAPGEN WILL DETERMINE THE SEGMENT NUMBER FROM THE SYMBOL TABLE.

EACH OF THE INFORMATION STORE DIRECTIVES IS DISCUSSED BELOW. THE HMAP (HARDWARE MAP) DIRECTIVE MUST APPEAR BEFORE ANY OF THE OTHER DIRECTIVES GIVEN IN THIS LIST SINCE THE LOCATION SPECIFIED IN THIS DIRECTIVE IS OF IMPORTANCE TO THE OTHER DIRECTIVES.

HMAP -- SPECIFY LOCATION FOR PAGE MAPS

HMAP <LOCATION>

THE INITIAL PAGE-MAPS (HMAP, THE HARDWARE MAP, AND LMAP, THE LOGICAL ADDRESS MAP) FOR THE SYSTEM ARE STORED AT THE SPECIFIED LOCATION; THE FORMAT OF THE PAGE-MAPS IS COMPATIBLE WITH THE THEN CURRENT DEFINITION OF THE MAPS.

MMAP -- SPECIFY LOCATION FOR MMAP

MMAP <LOCATION>

THE INITIAL MMAP (MEMORY UTILIZATION MAP) FOR THE SYSTEM IS STORED AT THE SPECIFIED LOCATION.

SDW -- SPECIFY LOCATION FOR SDW TABLE

SDW <LOCATION> <GROUP>

THE INITIAL SDW TABLE FOR THE SYSTEM IS STORED AT THE SPECIFIED LOCATION. THE ADDITIONAL <GROUP> ARGUMENT AFTER THE RANGE APPEARS WHICH DETERMINES WHICH SDW GROUP IS TO BE STORED. THE ARGUMENT MUST BE A 0, 1, 2 OR 3 AND CAUSES SDW DATA FOR SEGMENT RANGES 0-1777, 2000-3777, 4000-5777 AND 6000-7777, RESPECTIVELY, TO BE STORED. THE DIRECTIVE WILL PRESERVE THE PER-RING ACCESS INFORMATION IN THE INDIVIDUAL SDWS AND WILL CAUSE THE FAULT-BIT TO BE TURNED OFF. THE PHYSICAL-ADDRESS FOR THE PAGE-MAP IS STORED WITHIN THE SDW BASED ON THE LOCATION GIVEN IN THE HMAP DIRECTIVE DESCRIBED ABOVE.

PTUSEG -- SPECIFY LOCATION FOR PTUSEG ARRAY

PTUSEG <LOCATION>

THE INITIAL PTUSEG ARRAY FOR THE SYSTEM IS STORED AT THE SPECIFIED LOCATION. THE OWNER FOR THE SEGMENTS IS GIVEN AS THE SUPERVISOR.

8.3 SUPPORT DIRECTIVES

THE SUPPORT DIRECTIVES ARE SUPPLIED TO ALLOW THE USER TO EITHER GIVE OR RECEIVE ADDITIONAL INFORMATION ABOUT THE MAPGEN BUILD OPERATION. THE DIRECTIVES (WITH THE EXCEPTION OF THE TABLE DIRECTIVE) MUST BE SPECIFIED AFTER THE SEGMENT DESCRIPTION DIRECTIVES LISTED ABOVE. THE DIRECTIVES ALONG WITH THE SYNTAX AND SEMANTICS OF EACH ARE LISTED BELOW.

TABLE -- SPECIFY WHERE FIRST SYMBOLS ARE OBTAINED

TABLE <FILE>

THIS DIRECTIVE CAUSES THE FIRST SET OF SYMBOLS TO BE READ FROM THE <FILE> ARGUMENT, WHICH MAY BE A TREENAME. THE FILE IS ASSUMED TO BE A SEG LOAD MAP. MAPGEN WILL READ THE SYMBOLS FOR ECBS, COMMON BLOCKS AND OTHER SYMBOLS. FOR SYMBOLS THAT REPRESENT AN ECB, THE ADDRESS OF THE ECB ITSELF IS USED, NOT THE PROCEDURE ADDRESS. USE OF A SYMBOL WHICH REPRESENTS AN ECB WILL CAUSE A WARNING MESSAGE TO BE ISSUED. THIS DIRECTIVE MUST BE SPECIFIED PRIOR TO USE OF ANY SYMBOL AND MAY APPEAR BEFORE THE SEGMENT DESCRIPTION DIRECTIVES.

COLDS -- SPECIFY NAME OF COLD-START FILE

COLDS <FILE> <SAVE-REGS>

THIS DIRECTIVE CAUSES THE COLD-START R-MODE OBJECT FILE TO BE BUILT AND SAVED AS THE TREENAME GIVEN IN THE <FILE> ARGUMENT. THE <SAVE-REGS> ARGUMENT REPRESENTS A NOTATION FOR THE REGISTER VECTOR CONTENTS OF THE OBJECT FILE. THIS ARGUMENT MAY BE A STRING OF OCTAL VALUES WHICH REPRESENT THE INITIAL BOUNDS AND REGISTER CONTENTS. THE ORDER AND MEANING OF THESE ARGUMENTS ARE IDENTICAL TO THAT OF THE DOSSUB SAVE COMMAND.

QUIT -- SPECIFY TERMINATION OF MAPGEN

QUIT

THIS DIRECTIVE CAUSES THE MAPGEN PROGRAM TO TERMINATE AND RETURN TO THE PRIMOS COMMAND LEVEL. THE NUMBER OF ERRORS AND WARNINGS ARE PRINTED.

MAP -- SPECIFY PRINTING OF SEGMENT/COLD-START MAPS

MAP

THIS DIRECTIVE CAUSES THE SEGMENT MAP AND COLD-START RESIDENT MAP TO BE PRINTED. THESE MAPS GIVE ATTRIBUTES OF EACH SEGMENT BY ADDRESS. ADDITIONAL STATISTICS ABOUT THE NUMBER OF PAGES WIRED AND PAGING DISK RECORDS USED ARE ALSO PRINTED.

DUMP -- SPECIFY PRINTING OF HMAP AND LMAP DATA

DUMP

THIS DIRECTIVE CAUSES THE DATA DEFINED WITHIN THE HMAPS AND LMAPS TO BE PRINTED. IT IS CONSIDERED A DEBUG DIRECTIVE ONLY.

8.4 MAPGEN EXAMPLE

THE FOLLOWING IS AN EXAMPLE OF HOW MAPGEN IS USED TO BUILD A COLD START MODULE. (THE EXAMPLE IS TAKEN FROM THE FILE C_COLD IN PRI400.)

```
*
*  BUILD PAGE MAPS AND CREATE *COLDS MEMORY IMAGE.
*
*
*    THIS FILE CONTAINS THE DIRECTIVES FOR THE *MAPGEN PROGRAM
*    FOR GENERATING PRIMOS IV REV 16 - SINGLE 64 USER VERSION.
*
*
FILMEM ALL                          /* INITIALIZE
R *MAPGEN
*
TABLE M_PRMOS
****************
*  SEGMENT 00   * CONTAINS I/O WINDOWS, DVDISK, CONTROL BLOCKS.
****************
SEGMENT      0   PR0000
EMPTY        0   177777             /* MAKE ALL REFERENCABLE...
LOAD      $LOW   $HIGH               /* ASSURE MODULE HAS PAGE SPACE.
WIRE      SEG0   AMLQCB              /* I/O WINDOWS, DVDISK.
EMPTY     SEG0   AMLQCB
****************
*  SEGMENT 01   * CONTAINS ASSOCIATIVE BUFFERS.
****************
SEGMENT      1       *
PAGE         0   177777             /*ALLOC PAGE SPACE FOR BUFFERS.
****************
*  SEGMENT 04   * CONTAINS PHANTOM INTS., SEMAPHORES, PCB,
*               * FAULT HANDLERS.
```

```
****************
SEGMENT     4   PR0004
EMPTY   SEG4    SEG4SZ          /* DO NOT NEED PAGING SPACE.
WIRE    SEG4    VPSD            /* PHANTOM CODE, CHECKS, BASE,
*                               /* SEMCOM, WARM/COLD START, ETC.
WIRE    VPSD    VPSD      12000 /* VPSD WIRED FOR OS DEBUGGING.
WIRE    IFLTB_  U2PCBE          /* INTERRUPT HANDLERS, CHKLOG,
*                               /* LOGEV1, PCBS THRU USER 2.
WIRE    SUPCSK  SEG4SZ          /* SOME CONCEALED STACKS, ISTACK
****************
*  SEGMENT 05  * CONTAINS RING ZERO GATES.
****************
SEGMENT     5   PR0005
****************
*  SEGMENT 06  * CONTAINS O/S KERNAL PROCEDURE AND LINKAGE.
****************
SEGMENT     6   PR0006
WIRE        0   WIRE6           /* WIRED THRU RTNSEG.
EMPTY       0   WIRE6
RESIDE GETSEG   COMO$$          /* COLD-START RES UP TO COMO$$.
****************
*  SEGMENT 07  * CONTAINS TFLIOB BUFFERS.
****************
SEGMENT     7       *
WIRE        0    2000           /* USER 1'S OUTPUT BUFFER WIRED.
WIRE    30000   32000           /* USER 1'S INPUT BUFFER WIRED.
EMPTY       0  177777           /* NO PAGING SPACE.
****************
*  SEGMENT 10  * CONTAINS USRCOM.
****************
SEGMENT    10   PR0010
RESIDE      0    2000           /* USER 1 COMMON COLD-START RES.
PAGE        0  177777           /* REST IS UNINITIALIZED.
****************
*  SEGMENT 11  * CONTAINS FILE SYSTEM PROCEDURE AND LINKAGE.
****************
SEGMENT    11   PR0011
****************
*  SEGMENT 12  * CONTAINS NETWORK SYSTEM PROCEDURE  LINKAGE.
****************
SEGMENT    12   PR0012
****************
*  SEGMENT 14  * CONTAIN ONE-TO-ONE STUFF LIKE PAGCOM, HDRBUF,
*              * SPECIAL CODE.
****************
SEGMENT    14   PR0014
EMPTY   SEG14   SDWE            /* NO PAGING SPACE.
WIRE    SEG14   HMAP      4000  /* HDRBUF, CONFIG, RSAV, FIGCOM,
*                               /* MMAP, INITIAL HMAPS OF SYSTEM
ONE     SEG14   HMAP      4000  /* TAPE-DUMP, MEMORY-SCAN CODE,
*                               /* WARM-COLD
PAGE    HMAP    PAGCOM          /* MUST ALLOCATE PAGING SPACE.
WIRE    PAGCOM  SDWE            /* PAGCOM, PAGDEV, SDW0, SDW1,
*                               /* SDW3 TABLES
ONE     PAGCOM  SDWE
```

```
****************
* SEGMENT 6000 * CONTAINS SUPERVISOR'S RING-0 STACK.
****************
SEGMENT  6000  PR6000
WIRE          0    2000          /* FIRST PAGE IS WIRED.
PAGE          0    2000          /* NEEDS PAGE DISK.
****************
*   END OF SEGMENT DEFINITIONS.
****************
*
*   FILL IN THE HMAP AND MMAP.
HMAP  HMAP
MMAP  MMAP
*
*   PRINT A MEMORY MAP.
*
MAP
*
*   DUMP OF PAGE MAPS FOR DEBUGGING ONLY.
*
*  DUMP
*
*   SAVE COLD-START IMAGE.
*
COLD  *COLDS  1/135777
*
QUIT
*
CO  CONTINUE
CO  TTY
```

9 MODIFICATIONS TO PRIMOS IV INTERNAL LOGIC

THE FOLLOWING DESCRIBES THE MAJOR MODIFICATIONS THAT HAVE BEEN MADE TO THE INTERNAL LOGIC OF PRIMOS IV. THIS INFORMATION IS REQUIRED NORMALLY ONLY BY THOSE INVOLVED IN THE MODIFICATION OR MAINTENANCE OF PRIMOS IV.

9.1 THE SPLITTING OF SEGMENT 4

THE LARGEST CHANGE TO PRIMOS IV SOFTWARE WAS THE SPLITTING UP OF THE SEG4 PROGRAM INTO TWO SEPARATE PIECES: SEG4 AND SEG14. IN REVISION 15 OF PRIMOS IV, SEG4 CONTAINED ALL OF THE DATABASES AND SOME OF THE CODE FOR MANAGING BOTH THE VIRTUAL MEMORY ENVIRONMENT AND THE PROCESS EXCHANGE MECHANISM. ALLOCATION OF THE VARIOUS DATABASES IN SEGMENT 4 WAS FAST BECOMING AN IMPOSSIBLE JUGGLING ACT. SYSTEMS THAT NEEDED LARGE USER ADDRESS SPACES HAD TO SUPPORT A MINIMAL NUMBER OF USERS IN ORDER TO MAKE ROOM IN SEGMENT 4 FOR THE INCREASED SIZE OF PAGING AND SEGMENTATION DATABASES. WHEN THE SINGLE VERSION OF PRIMOS IV WAS DEVELOPED, IT WAS NECESSARY TO PROVIDE SPACE FOR PROCESS EXCHANGE AND VIRTUAL MEMORY DATABASES AT THE LARGEST SIZE THEY WOULD EVER BE CONFIGURED. THERE WAS NOT ENOUGH ROOM IN ONE SEGMENT FOR THIS, SO SEGMENT 4 HAD TO BE SPLIT UP.

ON THE PRIME 400 AND PRIME 500, THE HARDWARE REQUIRES SOME CODE AND DATA TO RESIDE IN SEGMENT NUMBER 4. THIS INCLUDES PHANTOM INTERRUPT CODE AND MACHINE CHECK HANDLING. IN ADDITION, THE HARDWARE REQUIRES THAT ALL PROCESS EXCHANGE DATABASES RESIDE IN THE SAME SEGMENT. SINCE BOTH INTERRUPTS AND MACHINE CHECKS ARE CLOSELY COUPLED TO THE PROCESS EXCHANGE MECHANISM, IT MAKES SENSE TO HAVE SEGMENT NUMBER 4 CONTAIN BOTH THE PROCESS EXCHANGE DATABASES AND THE INTERRUPT AND MACHINE CHECK HANDLING CODE.

THE PRIMOS IV OPERATING SYSTEM REQUIRES SOME CODE AND DATA TO RESIDE IN MEMORY PAGES THAT ARE LOADED SUCH THAT THEY ARE ONE-TO-ONE WITH PHYSICAL MEMORY. (ONE-TO-ONE PAGES HAVE THEIR PHYSICAL MEMORY ADDRESS EQUAL TO THE WORD NUMBER OF THEIR VIRTUAL MEMORY ADDRESS.) SUCH CODE INCLUDES THE COLD START AND WARM START ROUTINES, THE TAPE DUMP PROGRAM, AND THE TOEHOLD TO ENTER VPSD. DATABASES WHICH MUST BE ONE-TO-ONE WITH PHYSICAL MEMORY INCLUDE THE CRASH REGISTER FILE AREA, THE SEGMENT DESCRIPTOR TABLES FOR ALL USERS, AND THE PAGE MAPS USED BY THE KERNEL SEGMENTS OF PRIMOS IV.

ALL CODE AND DATA WHICH HAD TO RESIDE ONE-TO-ONE WITH PHYSICAL MEMORY WAS MOVED TO A NEW SEGMENT, SEGMENT NUMBER 14. THIS NEW SEGMENT CONTAINS ALL THE DATABASES WHICH ARE INVOLVED WITH VIRTUAL MEMORY PAGING AND SEGMENTATION. IT ALSO CONTAINS THE TAPE DUMP PROGRAM, THE CRASH REGISTER SAVE AREA, AND SOME UTILITY CODE USED BY THE PAGING SYSTEM.

FIGCOM HAS ALSO BEEN MOVED TO SEGMENT 14. IT RETAINS LOCATION 700 AS IT WAS IN SEGMENT 4.

BOTH THE UNIT TABLE ENTRIES AND ATTACH POINT ENTRIES CONTAIN "POINTERS" TO UNIT TABLE ENTRIES IN A SEPARATE COMMON AREA, UTCOM. THE LOGIN NAME REMAINS IN USRCOM AS A 32 CHARACTER STRING. AT REVISION 16 OF PRIMOS, ONLY SIX CHARACTERS (3 WORDS) ARE USED; THE REMAINING WORDS ARE RESERVED FOR FUTURE USE.

AS IN PREVIOUS REVISIONS, VARIABLES IN USRCOM SUCH AS UNITAB (START OF UNIT TABLE POINTERS) ARE ARRAYS EQUIVALENCED TO LIST THUS PROVIDING BASICALLY THE SAME FUNCTION AS PL/1 BASED OVERLAYS. THE EXPRESSION:

```
PTR = UNITAB ((USR-1)*USRSIZ)+UNITNR)
```

WHERE: USR IS THE PROCESS OR USER NUMBER
USRSIZ IS THE PER PROCESS LENGTH OF "USRCOM".
UNITNR IS THE FILE UNIT NUMBER.

THUS BECOMES A POINTER TO THE UNIT TABLE ENTRY FOR THE UNIT. IF PTR HAS A VALUE OF 0, THIS INDICATES THAT THE UNIT IS CLOSED AND THAT A UNIT TABLE ENTRY IN UTCOM DOES NOT EXIST.

THE EXPRESSIONS:

```
PTR = CURATT ((USR-1)*USRSIZ)
PTR = HOMATT ((USR-1)*USRSIZ)
```

ARE USED TO OBTAIN THE POINTER TO UNIT TABLE ENTRIES REPRESENTING CURRENT AND HOME ATTACH POINTS.

THE VARIABLE LUSR IN THE COMMON AREA PUDCOM IS INITIALIZED AT COLD START TO THE VALUE (USR-1)*USRSIZ.

9.5.2 NEW STRUCTURE OF UTCOM

THE COMMON AREA UTCOM CONTAINS UNIT TABLE ENTRIES. THE VARIABLES IN UTCOM SUCH AS VSTAT, VBR, ETC. (WHICH WERE IN USRCOM PRIOR TO REVISION 16) ARE ALSO EQUIVALENCED TO LIST. THE EXPRESSION:

```
STATUS = VSTAT (PTR)
```

RESULTS IN STATUS CONTAINING THE OPEN STATUS ASSOCIATED WITH A PARTICULAR UNIT OR ATTACH POINT WHEN PTR IS OBTAINED USING THE ABOVE STATEMENTS.

VARIABLES IN UTCOM WHEN USED TO REPRESENT OPEN FILE UNITS RETAIN THE SAME MEANINGS AS IN PREVIOUS PRIMOS IV AND V RELEASES. THE VARIABLES AND MEANINGS ARE:

```
VSTAT     BIT    1:  IF SET FILE MODIFIED
          BIT    2:  IF SET OPEN FOR SYSTEM USE,
                     EXCLUDE FROM CONCURRENCY CHECK
          BITS 3-8:  FILE TYPE
```

	BITS 9-16: OPEN STATUS 1 = READ 2 = WRITE 3 = READ/WRITE 4 = ATTACH
VBRA	BEGINNING RECORD ADDRESS OF FILE
VDVNO	LOGICAL DISK NUMBER
VDCRA	CURRENT DISK RECORD ADDRESS IN DAM INDEX; 0 IF SAM FILE, -1 IF INVALID BUT A DAM FILE.
VDRWP	ORDINAL RECORD NUMBER IN FILE
VCRA	CURRENT DISK RECORD ADDRESS IN FILE
VRWP	WORD OFFSET OF CURRENT POSITION IN CURRENT DISK RECORD
VPRIV	BITS 1-8: ACTUAL RWLOCK VALUE 0 = ONLY ONE USER 1 = ONE WRITER XOR N READERS 3 = N WRITERS XOR N READERS 5 = N WRITERS AND N READERS BITS 9-16: PRIVILEGE BITS 1 = READ 2 = WRITE 4 = TRUNCATE/DELETE
VPOPRA	DISK RECORD ADDRESS OF FILE ENTRY IN FATHER UFD ENTRY WHOSE DATE-TIME MODIFIED (DTM) FIELDS ARE TO BE UPDATED UPON CLOSE IF THE FILE WAS MODIFIED.
VPRPRW	POSITION IN VPOPRA RECORD OF ENTRY CONTROL WORD (ECW) OF FILE ENTRY.

IF THE UNIT TABLE ENTRY IS USED TO REPRESENT AN ATTACH POINT RATHER THAN AN OPEN FILE, THE FOLLOWING DEFINITIONS APPLY:

VSTAT	SAME AS FOR FILES; OPEN STATUS IS 4 WHEN "OPEN FOR ATTACH"
VBRA	SAME AS FOR FILES
VDVNO	SAME AS FOR FILES
VDCRA	NOT VALID
VDRWP	NOT VALID
VCRA	NOT VALID
VRWP	NOT VALID

VPRIV	BITS 1-8: RESERVED BITS 9-16: 0 = NONOWNER; 1 = OWNER
VPOPRA	SAME AS FILE, DTM INDICATED IS UPDATED WHEN ATTACH POINT UFD IS MODIFIED.
VPOPRW	SAME AS FILE, DTM INDICATED IS UPDATED WHEN ATTACH POINT IS MODIFIED.

THE ATTACH POINT NAME IS NO LONGER STORED IN ANY SYSTEM TABLE AS AN ASCII CHARACTER STRING. THE NAME MAY BE CONVENIENTLY READ IN FROM DISK BY USING THE RING 0 SUBROUTINE UFDNAM. THE SOURCE FOR UFDNAM MAY BE FOUND IN PRI400>FS.

9.5.3 ALLOCATION OF UNIT TABLE ENTRIES IN UTCOM

UNIT TABLE ENTRIES ARE ALLOCATED AT COLD START BY AINIT AND ALLOCATED AND FREED WHILE THE SYSTEM IS RUNNING. AINIT GARNERS A UNIT TABLE ENTRY FOR SYSUN (FILE UNIT 0) AND CURRENT AND HOME ATTACH POINTS FOR EVERY CONFIGURED USER AT COLD START. THESE UNIT TABLE ENTRIES ARE NEVER FREED. SYSUN MAY BE CLOSED AND THE ATTACH POINTS MADE INVALID BY SETTING VSTAT = 0, HOWEVER. AT FILE OPEN TIME, UNIT TABLE ENTRIES ARE ALLOCATED BY THE SURROUTINE GETUN AND FREED (RETURNED) BY THE SUBROUTINE RTNUN. THE SOURCES FOR GETUN AND RTNUN MAY BE FOUND IN PRI400>FS. THE ARRAY UTBITS IN COMMON FSCOM IS A BIT MAP WITH 1 BIT PER UNIT TABLE ENTRY IN UTCOM. A TRUE BIT(1) INDICATES A FREE TABLE ENTRY.

THE UNIT TABLE RESERVATION STRATEGY USES THE FOLLOWING VARIABLES:

RUFREE	TOTAL NUMBER OF RESERVED UNITS IN SYSTEM THAT HAVE NOT BEEN ALLOCATED.
RUCNT	THE NUMBER OF RESERVED UNITS PER USER
NUFREE	NUMBER OF FREE UNIT TABLE ENTRIES IN UTCOM
UUCNT(USR)	AN ARRAY EACH OF WHOSE ELEMENTS CONTAIN THE NUMER OF UNIT TABLE ENTRIES CURRENTLY IN USE FOR THE GIVEN USER.

10 APPLICATION NOTE - I$MT

THIS SECTION DESCRIBES USE OF THE T$MT WAIT SEMAPHORE AND ERROR RECOVERY SCHEMES FOR READING AND WRITING TAPE WITH T$MT.

10.1 USE OF THE I$MT WAIT SEMAPHORE

LOOPING ON THE STATUS DONE WORD STATV(1) USES UP CPU TIME WHILE THE PROCESS WAITS FOR THE TAPE OPERATION TO COMPLETE. THIS IS NOT A GOOD PRACTICE FOR TWO REASONS. FIRST, IT TIES UP THE CPU NEEDLESSLY AND SLOWS DOWN SYSTEM PERFORMANCE IN GENERAL. SECOND, IT CAUSES THE PROCESS TO WASTE SOME OF ITS TIME SLICE WITHOUT DOING USEFUL WORK. THIS WILL RESULT IN THE PROCESS BEING SCHEDULED EXTRA TIMES AND THE REAL TIME OF PROGRAM EXECUTION WILL BE LONGER THAN NECESSARY.

THIS PROBLEM CAN BE SOLVED BY USING A SEMAPHORE. IF THE PROCESS WAITS ON A SEMAPHORE, THE WAIT TIME IS NOT COUNTED AGAINST ITS TIME SLICE. THEREFORE, AS SOON AS THE TAPE OPERATION COMPLETES, THE PROCESS WILL BE SCHEDULED TO RUN AGAIN TO FINISH UP ITS TIME SLICE.

THE PROGRAM T$MT CONTAINS A WAIT SEMAPHORE THAT CAN BE USED FOR THIS PURPOSE. THIS SEMAPHORE IS USED TO QUEUE TAPE REQUESTS. IF THE PROCESS MAKES A TAPE REQUEST WHEN THE CONTROLLER IS BUSY WITH ANOTHER OPERATION, THE PROCESS IS PUT ON THE WAIT SEMAPHORE.

WHENEVER THE PROGRAM WANTS TO WAIT FOR A TAPE OPERATION TO COMPLETE, IT CAN CALL T$MT WITH A REQUEST FOR STATUS. SINCE THE TAPE CONTROLLER IS ALREADY BUSY WITH THE PREVIOUS OPERATION, THE PROCESS WILL BE PUT ON THE T$MT WAIT SEMAPHORE.

SINCE THE STATUS REQUEST IS FAST AND DOESN'T AFFECT THE TAPE, IT IS A CONVENIENT TAPE OPERATION TO USE TO PROVIDE THE SEMAPHORE WAIT. A SCRATCH STATUS VECTOR SHOULD BE USED SO THAT THE STATUS FROM THE ORIGINAL CALL IS NOT DESTROYED.

EXAMPLE OF WAIT CODE:

```
      . . .

      INTEGER STATV(3)          /* STATUS VECTOR SET BY T$MT
      INTEGER UNIT              /* MAG TAPE DRIVE NUMBER (0-7)
      INTEGER BUF (1024)        /* OUTPUT BUFFER
      INTEGER XSTATV (3)        /* SCRATCH VECTOR FOR WAIT

      . . .
      CALL T$MT (UNIT,LOC(BUF),,:042620,STATV) /* WRITE 1024

      . . .                     /* OVERLAP EXECUTION WITH IO

C     WAIT FOR TAPE WRITE TO COMPLETE.
```

```
100   IF (STATV(1).EQ.0) GOTO 120 /* SEE IF IO IS ALREADY DONE
         CALL T$MT (UNIT,LOC(0),0,:100000,XSTATV) /* WAIT
         GOTO 100
120   . . .
```

10.2 ERROR RECOVERY FOR TAPE WRITES

THERE ARE MANY POSSIBLE ERROR RECOVERY SCHEMES. THE TWO THAT ARE DESCRIBED HERE ARE BASED ON DIFFERENT RECORD FORMATS. THE FIRST ALGORITHM CAN BE USED WHEN RECORDS CONTAIN ONLY DATA. THE OTHER SCHEME REQUIRES THAT THE RECORDS CONTAIN EXTRA INFORMATION FOR ERROR RECOVERY.

10.2.1 SIMPLE WRITE ERROR RECOVERY

THE AIM OF THE SIMPLE ERROR RECOVERY PROGRAM IS TO GET BY A POSSIBLE BAD SPOT ON THE TAPE BY ERASING PART OF THE TAPE WHERE THE ERROR OCCURRED AND REWRITING THE RECORD AFTER THAT GAP.

THE PROGRAM DOES NOT TRY TO REWRITE THE RECORD ON THE SAME SPOT ON THE TAPE EVEN THOUGH REPEATED TRIES ON THE SAME SPOT MAY IMPROVE THE TAPE ENOUGH TO PERMIT THE WRITE TO SUCCEED. THE TAPE IS CONSIDERED MARGINAL AT THAT SPOT AND MAY NOT BE READABLE AT A LATER DATE.

THE TAPE CAN BE ERASED BY WRITING A FILE MARK AND THEN BACKSPACING OVER THE FILE MARK. THIS WILL CAUSE THREE INCHES OF TAPE TO BE ERASED.

PROGRAM STEPS FOR WRITE ERROR RECOVERY:

1. CHECK THAT ERROR RECOVERY IS POSSIBLE. DON'T ATTEMPT ERROR RECOVERY IF THE TAPE DRIVE IS OFFLINE OR NOT READY, OR THE TAPE IS FILE PROTECTED.

2. BACKSPACE OVER THE RECORD.

3. ERASE A THREE INCH GAP ON THE TAPE.

   A. WRITE A FILE MARK.

   B. BACKSPACE A RECORD AND CHECK THAT THE FILE MARK DETECTED BIT IS SET IN THE STATUS WORD.

4. ATTEMPT TO WRITE THE RECORD AGAIN.

5. IF THE RECORD WAS NOT WRITTEN SUCCESSFULLY, REPEAT STEPS 1-4 UP TO TWENTY TIMES (A MAXIMUM OF FIVE FEET OF ERASED TAPE).

10.2.2 WRITE ERROR RECOVERY WITH SEQUENCE NUMBERS

THERE IS A DRAWBACK TO THE FIRST SCHEME. SINCE THE TAPE IS BAD AT THE SPOT WHERE THE ERROR RECOVERY IS BEING DONE, IT IS POSSIBLE FOR ERRORS TO OCCUR WHILE BACKSPACING. FOR EXAMPLE, IF THE BAD RECORD HAS A GAP IN THE MIDDLE OF IT, THE PROGRAM MIGHT DETECT TWO SHORT RECORDS WHEN BACKSPACING. IF THE PROGRAM HAS SOME WAY OF IDENTIFYING RECORDS, THE PROGRAM CAN BE SURE THAT IT HAS NOT LOST POSITION DURING ERROR RECOVERY.

ONE WAY TO DO THIS IS TO INCLUDE A SEQUENCE NUMBER WITH EVERY RECORD. THEN WHEN ERROR RECOVERY IS ATTEMPTED, THE PROGRAM BACKSPACES TWO RECORDS AND THEN READS A RECORD. THIS RECORD SHOULD CONTAIN THE SEQUENCE NUMBER OF THE LAST GOOD RECORD BEFORE THE ERROR RECORD.

PROGRAM STEPS FOR ERROR RECOVERY:

1. CHECK THAT ERROR RECOVERY IS POSSIBLE. DON'T ATTEMPT ERROR RECOVERY IF THE TAPE DRIVE IS OFFLINE OR NOT READY, OR THE TAPE IS FILE PROTECTED.

2. POSITION THE TAPE AFTER THE LAST GOOD RECORD.

   A. BACKSPACE TWO RECORDS. THIS WILL PLACE THE TAPE BEFORE THE LAST GOOD RECORD.

   B. READ A RECORD AND VERIFY THAT ITS SEQUENCE NUMBER MATCHES THE ONE EXPECTED FOR THE LAST GOOD RECORD.

   C. IF THE 'GOOD' RECORD CAN'T BE READ, THEN IT IS

POSSIBLE THAT THE TAPE IS NOT POSITIONED CORRECTLY. BACKSPACE SEVERAL RECORDS AND READ THOSE RECORDS TO FIND THE SEQUENCE NUMBER OF THE LAST GOOD RECORD WRITTEN.

3. ERASE A THREE INCH GAP ON THE TAPE.

   A. WRITE A FILE MARK.

   B. BACKSPACE A RECORD AND CHECK THAT THE FILE MARK DETECTED BIT IS SET IN THE STATUS WORD.

4. ATTEMPT TO WRITE THE RECORD AGAIN.

5. IF THE RECORD WAS NOT WRITTEN SUCCESSFULLY, REPEAT STEPS 1-4 UP TO TWENTY TIMES, LENGTHENING THE GAP EACH TIME.

10.3 ERROR RECOVERY FOR TAPE READS

ERROR RECOVERY WHEN READING A TAPE INVOLVES REPEATEDLY REREADING THE RECORD. THE SAME PROBLEM OF LOSING POSITION CAN OCCUR WHEN DOING ERROR RECOVERY SO THE ALGORITHM CAN BE IMPROVED BY VERIFYING THE SEQUENCE NUMBER EACH TIME A RECORD IS READ.

PROGRAM STEPS FOR READ ERROR RECOVERY:

1. CHECK THAT ERROR RECOVERY IS POSSIBLE. DON'T ATTEMPT ERROR RECOVERY IF THE TAPE DRIVE IS OFFLINE OR NOT READY.

2. BACKSPACE AND REREAD THE RECORD EIGHT TIMES.

3. IF UNSUCCESSFUL, BACKSPACE EIGHT RECORDS (OR TO THE LOAD POINT IF LESS THAN EIGHT RECORDS AWAY), SPACE FORWARD SEVEN RECORDS AND THEN READ THE PROBLEM RECORD. THIS SEQUENCE DRAWS THE TAPE OVER THE TAPE CLEANER AND COULD DISLODGE A POSSIBLE DIRT PARTICLE.

4. REPEAT STEPS 1-3 EIGHT TIMES.

SUBJECT: FTNOPT REV. 16.1

THE REV. 16.1 MASTER DISK RELEASE IS UNUSUAL IN THAT TWO FORTRAN COMPILERS ARE INCLUDED IN IT. FTN IS THE REV. 16.0 FTN WITH LITTLE CHANGE. THE OTHER VERSION, FTNOPT, ALLOWS TWO NEW OPTIONS WHICH INSTRUCT THE COMPILER TO PERFORM CERTAIN OPTIMIZATIONS UPON DO LOOPS. THESE TWO VERSIONS OF FTN WILL REMAIN SEPARATE ON SUBSEQUENT "POINT" REVS., BUT WILL BE MERGED INTO A SINGLE COMPILER AT REV. 17.

THE DO LOOP OPTIMIZATION PERFORMED BY FTNOPT IS OPTIONAL, AND MUST BE EXPLICITLY REQUESTED BY THE USER IN THE FTNOPT COMMAND LINE. ALTHOUGH THE LOOP OPTIMIZATION ALGORITHMS ARE GENERAL-PURPOSE, THE EFFECT OF OPTIMIZATION ON WELL-CODED FORTRAN PROGRAMS WILL BE TO REMOVE SOME SUBSCRIPT CALCULATIONS FROM THE DO LOOP.

THIS DOCUMENT DESCRIBES THE LOOP OPTIMIZATIONS DONE, AND THE COMMAND LINE OPTIONS THAT INVOKE THE FTNOPT DO LOOP OPTIMIZER.

OPTIMIZATION

TWO TYPES OF OPTIMIZATIONS ARE DONE:

1. REMOVAL OF INVARIANT OPERATIONS
2. STRENGTH REDUCTION OF EXPRESSIONS THAT INVOLVE THE DO LOOP INDEX

1. REMOVAL OF INVARIANT OPERATIONS

INVARIANT OPERATIONS ARE OPERATIONS ON OPERANDS WHOSE VALUES DO NOT CHANGE WITHIN THE DO LOOP, (E.G., THE OPERANDS ARE NOT SET BY AN ASSIGNMENT STATEMENT WITHIN THE LOOP). SINCE THE OPERANDS DO NOT CHANGE WITHIN THE LOOP, IT IS NOT NECESSARY FOR THE CODE THAT PERFORMS THE OPERATION TO BE CONTAINED WITHIN THE LOOP. THEREFORE, THE CODE FOR THESE OPERATIONS IS MOVED OUTSIDE OF THE LOOP, AND IS EXECUTED ONLY ONCE, IMMEDIATELY BEFORE LOOP SET-UP AND ENTRY, INSTEAD OF EVERY TIME THE LOOP IS EXECUTED.

THE CURRENT IMPLEMENTATION DOES INVARIANT OPERATION REMOVAL FOR ARITHMETIC AND LOGICAL OPERATIONS ON INTEGER (INTEGER*2 AND INTEGER*4) OPERANDS, FOR INTEGER MODE CONVERSIONS, AND FOR THE INTEGER INTRINSICS.

2. STRENGTH REDUCTION

STRENGTH REDUCTION IS THE CONVERSION OF AN EXPRESSION IN THE DO LOOP THAT INVOLVES THE LOOP INDEX INTO A SIMPLER EXPRESSION, THAT EXECUTES FASTER THAN THE ORIGINAL EXPRESSION. USUALLY, STRENGTH REDUCTION IS DONE ON EXPRESSIONS THAT INVOLVE A MULTIPLICATION OF

THE DO LOOP INDEX. SUCH EXPRESSIONS ARE CONVERTED, OR REDUCED, TO EXPRESSIONS THAT DO SEVERAL ADDITIONS, INSTEAD OF THE MULTIPLICATION. SOME OF THESE ADDITIONS CAN BE MOVED OUTSIDE OF THE LOOP FURTHER DECREASING THE EXECUTION TIME OF THE LOOP. STRENGTH REDUCTION OF EXPRESSIONS IN A DO LOOP CAN BE DONE ONLY IF THE FOLLOWING CONDITIONS ARE TRUE:

1. THE LOOP INDEX IS ALTERED ONLY IN THE NORMAL LOOP INCREMENTING MANNER, (I.E., IT IS NOT MODIFIED BY AN ASSIGNMENT STATEMENT IN THE DO LOOP).

2. THE LOOP INCREMENT IS INVARIANT WITHIN THE DO LOOP.

3. COMMAND_OPTIONS

OPTIMIZATION OF DO LOOPS IS DONE ONLY IF EXPLICITLY REQESTED BY THE USER IN THE FTNOPT COMMAND LINE. EVEN THOUGH OPTIMIZATON IS REQUESTED, IT IS POSSIBLE THAT MANY OF THE DO LOOPS (OR ALL OF THEM), IN ANY GIVEN PROGRAM DO NOT SATISFY ALL OF THE CRITERIA FOR OPTIMIZATION, OR CONTAIN CERTAIN STATEMENTS (E.G., READ, WRITE) THAT MAKE OPTIMIZATION MEANINGLESS. THEREFORE, COMPILING WITH AN OPTIMIZATION OPTION MAY NOT RESULT IN ANY INCREASED PERFORMANCE FOR SOME FORTRAN PROGRAMS.

THERE ARE TWO OPTIMIZATION COMMAND LINE OPTIONS:

OPTION	FUNCTION
-OPT	PERFORM OPTIMIZATION ON ALL DO LOOPS THAT DO NOT CONTAN ANY GO TO STATEMENTS.
-UNCOPT	PERFORM OPTIMIZATION UNCONDITIONALLY

THE CORRESPONDING REGISTER SETTINGS ARE:

B-REG		ON OPTION	OFF OPTION
BIT 5	004000	OPT	---
BIT 6	002000	UNCOPT	---

DESCRIPTION_OF_IHE_OPTIONS

-OPT IS A "SAFE" OPTION. ANY LOOP THAT IS OPTIMIZED BY THE COMPILER WHEN THIS OPTION IS SPECIFIED WILL EXECUTE CORRECTLY. ANY DO LOOPS THAT COULD POTENTIALLY NOT WORK AFTER OPTIMIZATION ARE NOT OPTIMIZED.

-UNCOPT CAUSES THE COMPILER TO ATTEMPT TO OPTIMIZE ALL DO LOOPS, EVEN THOSE THAT CONTAIN GO TO STATEMENTS. THE OPTIMIZED CODE GENERATED BY FTNOPT FOR DO LOOPS THAT CONTAIN GO TO STATEMENTS THAT TRANSFER CONTROL ENTIRELY WTHIN THE DO LOOPS, OR THAT SIMPLY EXIT

FROM THE LOOPS WILL EXECUTE CORRECTLY. HOWEVER, IF ANY LOOP CONTAINS ANY GO TO STATEMENT THAT EXITS TO A CODE SEQUENCE THAT EVENTUALLY RETURNS INTO THE LOOP, THE OPTIMIZED CODE MAY NOT (AND MOST LIKELY WILL NOT) EXECUTE CORRECTLY IF ANY OF THE OPERANDS THAT ARE INVARIANT WITHIN THE LOOP OR THE LOOP INDEX OR INCREMENT VARIABLE ARE MODIFIED BY THE CODE SEQUENCE OUTSIDE OF THE LOOP. IF THIS OPTION IS USED, IT IS THE USER'S RESPONSIBILITY TO INSURE THAT NO OPERANDS THAT ARE CONSIDERED TO BE INVARIANT BY THE OPTIMIZER ARE MODIFED BY THE CODE SEQUENCE OUTSIDE OF THE LOOP.

SUBJECT: REV. 16 LOADER CHANGES

1. EDB

4 TIMES FASTER.

INPUT SPECIFICATION IS REQUIRED - I.E., EDB NO LONGER DEFAULTS TO THE PAPER TYPE READER.

(PTR) AND (ASR) WILL NO LONGER BE RECOGNIZED. FOR CONSISTENCY WITH COMMAND LINE SYNTAX, -PRT AND -ASR SHOULD BE USED INSTEAD.

2. SEG

THE INTERNAL TABLES WHICH ARE COPIED INTO SEGMENT 0 OF THE SEG RUN FILE HAVE BEEN CHANGED IN ORDER TO EXPAND THE SYMBOL TABLE AREA. THEREFORE, ALL COMMAND FILES SHOULD BE RUN TO INSURE THAT THERE ARE CONFLICTS. FOR EXAMPLE, R-MODE INTERLUDE COMMANDS IN CMDNCO CAN NOT HANDLE THE NEW FORMAT UNTIL THEY HAVE BEEN REBUILT. OLD FORMAT SEG RUN FILES WILL BE CONVERTED TO THE NEW FORMAT AUTOMATICALLY BY SEG. BUFCTL NOW CONSISTS OF (SEGS*2+2 WORDS): COMMON/BUFCTL/REVFLG,BUFCNT,BUFCTL(SEGS*2). A BIT RATHER THAN A WORD IS USED TO INDICATE WHETHER OR NOT A SEGMENT SUBFILE HAS BEEN LOADED INTO. REVFLG WILL BE PRESENT FROM NOW ON. IT IS SET TO -1 AS A FLAG THAT TABLE CONVERSION WILL NOT BE NECESSARY. CURRENTLY, SEGS=256. THERE ARE 32 SUBFILES PER SEGMENT.

SEG CHECK FOR LOAD* OR VLOAD* TYPING ERRORS WHICH USED TO RESULT IN THE RUN FILE BEING DELETED. COMMON BLOCKS LONGER THAN ONE SEGMENT NO LONGER HAVE TO BEGIN AT UND ZERO. MULTIPLE STACK ALLOCATION WILL NO LONGER RUN. THE MIX OPTION CAN BE USED WITH ARRAYS OVER 64K. THE R-MODE INTERLUDE PROGRAMS WILL EXIT GRACEFULLY SHOULD CONTROL RETURN TO RUNIT.

BUGS FIXED

TAR25528- UPDATE SYMBOL TABLE SIZE PRIOR TO WRITING OUT SEGMENT 0

TAR25724- DO NOT ASSIGN STACK SEGMENT

TAR25532- DOUBLE PRECSION ADD SO THAT COMMON BLOCKS LONGER THAN ONE SEGMENT NO LONGER HAVE TO BEGIN AT WORD 0.

TAR25533- MIX OPTION/LONG COMMON BUG FIXED

TAR12731- CHECK FOR LOAD/VLOAD* TYPING ERROR

CMDMAK AND CM.FILE HAVE BEEN FIXED TO CALL EXIT UPON RETURN FROM RUNIT IN THE R-MODE INTERLUDE PROGRAM

DIRECT COMMON REFERENCE CONVERSION HAS BEEN FIXED.

3. LOAD

SYMBOLS MAY HAVE 8-CHARACTER NAMES.

RR (RESET RANGE) CAN BE USED TO RESET THE SAVE RANGE PRIOR TO EN (ENTIRE SAVE) WHEN OVERLAYS ARE BUILT.

LINKING IN COMMON IS NOW ALLOWED WHILE FORWARD REFERENCES ARE BEING UNSTRUNG.

BUGS FIXED

LOAD ALLOWS LINKING IN COMMON WHEN UNSTRING FORWARD REFERENCES. LOAD WILL NOW GIVE A CORRECT EOF ERROR MESSAGE WHEN AN ATTEMPT IS MADE TO LOAD A NULL FILE.

A FIX HAS BEEN MADE TO REMOVE THE CODE,CODE ARGUMENT SEQUENCE IN PRWF$$ CALLS

LOAD HAS BEEN FIXED SO BITS DIPLAYED IN *UII ARE CORRECT.

SUBJECT: REV. 16 LOADER CHANGES

1. EDB

4 TIMES FASTER.

INPUT SPECIFICATION IS REQUIRED - I.E., EDB NO LONGER DEFAULTS TO THE PAPER TYPE READER.

(PTR) AND (ASR) WILL NO LONGER BE RECOGNIZED. FOR CONSISTENCY WITH COMMAND LINE SYNTAX, -PRT AND -ASR SHOULD BE USED INSTEAD.

2. SEG

THE INTERNAL TABLES WHICH ARE COPIED INTO SEGMENT 0 OF THE SEG RUN FILE HAVE BEEN CHANGED IN ORDER TO EXPAND THE SYMBOL TABLE AREA. THEREFORE, ALL COMMAND FILES SHOULD BE RUN TO INSURE THAT THERE ARE CONFLICTS. FOR EXAMPLE, R-MODE INTERLUDE COMMANDS IN CMDNCO CAN NOT HANDLE THE NEW FORMAT UNTIL THEY HAVE BEEN REBUILT. OLD FORMAT SEG RUN FILES WILL BE CONVERTED TO THE NEW FORMAT AUTOMATICALLY BY SEG. BUFCTL NOW CONSISTS OF (SEGS*2+2 WORDS): COMMON/BUFCTL/REVFLG,BUFCNT,BUFCTL(SEGS*2). A BIT RATHER THAN A WORD IS USED TO INDICATE WHETHER OR NOT A SEGMENT SUBFILE HAS BEEN LOADED INTO. REVFLG WILL BE PRESENT FROM NOW ON. IT IS SET TO -1 AS A FLAG THAT TABLE CONVERSION WILL NOT BE NECESSARY. CURRENTLY, SEGS=256. THERE ARE 32 SUBFILES PER SEGMENT.

SEG CHECK FOR LOAD* OR VLOAD* TYPING ERRORS WHICH USED TO RESULT IN THE RUN FILE BEING DELETED. COMMON BLOCKS LONGER THAN ONE SEGMENT NO LONGER HAVE TO BEGIN AT UND ZERO. MULTIPLE STACK ALLOCATION WILL NO LONGER RUN. THE MIX OPTION CAN BE USED WITH ARRAYS OVER 64K. THE R-MODE INTERLUDE PROGRAMS WILL EXIT GRACEFULLY SHOULD CONTROL RETURN TO RUNIT.

BUGS FIXED

TAR25528- UPDATE SYMBOL TABLE SIZE PRIOR TO WRITING OUT SEGMENT 0

TAR25724- DO NOT ASSIGN STACK SEGMENT

TAR25532- DOUBLE PRECSION ADD SO THAT COMMON BLOCKS LONGER THAN ONE SEGMENT NO LONGER HAVE TO BEGIN AT WORD 0.

TAR25533- MIX OPTION/LONG COMMON BUG FIXED

TAR12731- CHECK FOR LOAD/VLOAD* TYPING ERROR

CMDMAK AND CM.FILE HAVE BEEN FIXED TO CALL EXIT UPON RETURN FROM RUNIT IN THE R-MODE INTERLUDE PROGRAM

DIRECT COMMON REFERENCE CONVERSION HAS BEEN FIXED.

3. LOAD

SYMBOLS MAY HAVE 8-CHARACTER NAMES.

RR (RESET RANGE) CAN BE USED TO RESET THE SAVE RANGE PRIOR TO EN (ENTIRE SAVE) WHEN OVERLAYS ARE BUILT.

LINKING IN COMMON IS NOW ALLOWED WHILE FORWARD REFERENCES ARE BEING UNSTRUNG.

BUGS FIXED

LOAD ALLOWS LINKING IN COMMON WHEN UNSTRING FORWARD REFERENCES. LOAD WILL NOW GIVE A CORRECT EOF ERROR MESSAGE WHEN AN ATTEMPT IS MADE TO LOAD A NULL FILE.

A FIX HAS BEEN MADE TO REMOVE THE CODE,CODE ARGUMENT SEQUENCE IN PRWF$$ CALLS

LOAD HAS BEEN FIXED SO BITS DIPLAYED IN *UII ARE CORRECT.

SUBJECT: FORTRAN LIBRARY, PSD, AND VPSD FOR REV. 16

PSD

PSD HAS NOT BEEN CHANGED FOR REV. 16. THE LATEST VERSION IS REV. 14.2, DESCRIBED IN THE PMA MANUAL, PDR3059.

VPSD

SEVERAL MINOR BUGS IN REV. 15 VPSD HAVE BEEN CORRECTED, AND A NEW ENTRY POINT FOR THE USE OF THE HIGHER LEVEL LANGUAGE DEBUGGER HAS BEEN ADDED. VPSD FOR REV. 15 IS DESCRIBED IN THE PMA MANUAL.

FORTRAN LIBRARY

THE FORTRAN LIBRARY HAS SOME NEW ROUTINES AND A LOT OF UPDATED ONES. FOR REV. 16 THE SHARED FORTRAN LIBRARY WILL BE THE DEFAULT LIBRARY FOR V-MODE.

NEW ROUTINES:

TOVFD$	CALL TOVFD$(NUM) OUTPUTS THE 16-BIT INTEGER NUM TO THE TERMINAL WITHOUT ANY SPACES, I.E. "123" OR "-17".
"$X" SERIES	- THESE ARE SHORT-CALLABLE (V-MODE ONLY) VERSIONS OF COMMON SCIENTIFIC FUNCTIONS WHICH TAKE A SINGLE ARGUMENT IN THE SINGLE OR DOUBLE PRECISION FLOATING ACCUMULATOR. AVAILABLE ARE: SINX, COSX, ATANX, EXPX, SRQT$X, ALOG$X, DSIN$X, DCOS$X, DATN$X, DEXP$X, DSQR$X, AND DLOG$X.
V-MODE	THE FOLLOWING ROUTINES ARE NOW AVAILABLE IN V-MODE: C$P02, O$AL06, I$AA01, I$AP02, O$AP02, P1IN, P10U, P1IB AND P10B.

MODIFIED ROUTINES:

F$AT	A NEW VERSION OF THIS ROUTINE (THE R-MODE ARGUMENT TRANSFER SUBROUTINE) WRITTEN BY BERNIE STUMPF IS ABOUT TWICE AS FAST AS THE OLD VERSION. A LOT OF TIME IS TRADITIONALLY SPENT IN THIS ROUTINE IN R-MODE PROGRAMS. IT ALSO PRESERVES THE A AND B REGISTERS AND THE FLOATING ACCUMULATOR, ALLOWING SUBROUTINES WITH ARGUMENTS TO STILL PICK UP REGISTER CONTENTS USING THE -SPO OPTION IN FORTRAN.
TBUFIN	THIS ROUTINE, CALLED BY TIDEC, TIOCT AND TIHEX, HAS BEEN UPDATED TO PROPERLY HANDLE ERASE AND KILL CHARACTERS.
I$AA12	TWO BUGS IN ERASE AND KILL HANDLING WERE FIXED, AND SOME

	SPEED-UPS MADE.
O$AL06	A BUG CONCERNING DATA PRODUCTS PRINTERS WAS FIXED.
CONTRL	CHANGED TO TAKE THE ERROR RETURN WHEN PASSED A BAD UNIT NUMBER, OR GIVE A MESSAGE IF NO RETURN IS PROVIDED.
F$IO	INCORRECT OUTPUT OF FN.(N-1) IN THE FORTRAN I/O PACKAGE FIXED.
MAGTAPE	ROUTINES - UPDATED TO (1) NOT LOOP WAITING FOR I/O, (2) DO BETTER ERROR RECOVERY. THE REV. 15 VERSIONS SIMPLY RETRIED 20 TIMES AND GAVE UP IF THEY WERE UNSUCCESSFUL AT DOING THEIR I/O. THE NEW VERSIONS RETRY READ IN THE SAME FASHION BUT ONLY 5 TIMES. WRITES ARE RETRIED ONCE, AND IF THAT FAILS THE TAPE IS ASSUMED TO HAVE A BAD SPOT. THE ROUTINES BACK OVER THE BAD SPOT AND WRITE BLANK TAPE, THEN TRY AGAIN, UP TO 10 TIMES, OR ABOUT 30 INCHES OF TAPE. (3) THE V-MODE VERSIONS OF THE 7-TRACK AND EBCDIC ROUTINES NO LONGER HAVE A LIMITATION ON RECORD SIZE AS THEY ALLOCATE THEIR WORK BUFFER AS NEEDED, RATHER THAN USING ONE OF FIXED SIZE. JOEL JENNINGS HELPED WITH THESE ROUTINES.
E$XX	FIVE OF THE EXPONENTIATION ROUTINES HAVE BEEN REWRITTEN TO REPLACE SLOWER REV. 15 VERSIONS. THEY ARE: E$11, E$51, E$71, E$77, AND E$57.
LTAPS	THIS ROUTINE, CLAIMED TO BE THOROUGHLY OBSOLETE AND UNUSED BY RUSS BARBOUR AND BERNIE STUMPF, WAS DELETED FROM THE LIBRARY.
F$BN	THE FORTRAN REWIND ROUTINE WAS REWRITTEN, IN FORTRAN, TO GIVE BETTER ERROR MESSAGES.
PACK	ANOTHER OBSOLETE ROUTINE, DELETED AFTER CONSULTATION WITH THE ABOVE-MENTIONED STC'S.
F$IOER	REWRITTEN IN FORTRAN.

SUBJECT: CMPF & MRGF BUG FIXES

A FIX IS BEING MADE TO BOTH THE CMPF & THE MRGF COMMANDS TO FIX A BUG IN FILE NAME ARGUMENT PROCESSING. THIS BUG OCCURRED ONLY WHEN MORE THAN THREE INPUT FILE NAMES WERE SPECIFIED TO EITHER CMPF OR MRGF.

RPG FOR REV16.00

I. BUG FIXES

A. TAR81303
ONCE A FILE HAS EXPERIENCED AN UNSUCCESSFUL CHAIN OPERATION, ALL DATA FIELDS PERTAINING TO THAT FILE WILL BE FILLED WITH 9'S THROUGHOUT THE DURATION OF THE PROGRAM RUN.

B. TAR25232 AND TAR25226
IF A FILE IS SPECIFIED AS BEING IN ASCENDING SEQUENCE (AN A IN COLUMN 18 OF THE FILE SPECS) AND IF PACKED DATA IS CONTAINED AND REFERENCED, THE DATA IS TREATED AS IF IT WERE UNPACKED.

C. TAR UNKNOWN
MVR DOES NOT FUNCTION PROPERLY WHEN THE NUMBER OF DIGITS IN THE DIVIDEND IS LESS THAN THE NUMBER OF DIGITS IN THE DIVISOR.

D. TAR25234 AND TAR81301
AN EXECUTION TIME ARRAY WITH PACKED DATA IN CONJUNCTION WITH THE INPUT SPEC STATEMENT, WITHOUT NUMBER OF DECIMAL POSITIONS SPECIFIED (AS IT SHOULD NOT BE WHEN THE ARRAY IS DECLARED BY ONE INPUT SPEC STATEMENT) WILL GIVE AN ==INCONSISTENT USAGE== ERROR MESSAGE.

E. TAR25231
FIELDS WITH A LENGTH OF TWO OR LESS ARE NOT EDITED IN ACCORDANCE WITH THE EDIT CODES.

F. TAR15185 AND TAR25224
EDIT CODES NOT FUNCTIONING PROPERLY.

G. TAR25225
LOKUP OPERATION ONLY WORKS PROPERLY WHEN THE TABLE IS IN ASCENDING SEQUENCE.

H. TAR25227
AN UNSUCCESSFUL CHAIN OPERATION CAUSES A KIDA 07 ERROR MESSAGE TO BE DISPLAYED ON THE USER TERMINAL.

I. TAR15526
NO STATEMENT TYPE SEQUENCING IS DONE DURING COMPILATION.

J. TAR25233
ARITHMETIC OPERATIONS TRUNCATE SIGNIFICANT DIGITS DURING COMPUTATION.

K. TAR25228
DIVISION BY ZERO SETS RESULTING FIELD EQUAL TO DIVIDEND RATHER THAN TO ZERO.

L. TAR UNKNOWN
CHAIN OPERATION RETRIEVES NEXT RECORD IN SEQUENCE IF THE CHAIN WERE UNSUCCESSFUL.

M. TAR25230
SETLL POSITIONS THE DEMAND FILE TO END OF FILE RATHER THAN NEXT HIGHER KEY IF AN EXACT MATCH WERE NOT FOUND ON THE KEY.

N. TAR15183
A RECORD WITH NO FIELDS DEFINED TAKES THE FIELDS FOR THE NEXT RECORD AND DROPS THE RECORD IDENTIFIER APPLICABLE TO THE FIELDS IT TAKES.

II. ENHANCEMENTS

A. AS SUGGESTED BY TAR15180, A HEADING LINE, PRINTED AT THE TOP OF EACH PAGE OF THE RPG COMPILATION LISTING, WHICH CONTAINS RPG REV NUMBER, INPUT SOURCE FILE NAME, AND THE DATE AND TIME OF COMPILATION, HAS BEEN ADDED.

B. THE HALF-ADJUST ('H' IN COLUMN 53 OF THE CALCULATION LINE) HAS BEEN IMPLEMENTED.

NOTES:

=HALF-ADJUST CAN ONLY BE USED WITH ARITHMETIC OPERATIONS.

=HALF-ADJUST CANNOT BE USED WITH AN MVR OPERATION OR A DIVISION FOLLOWED BY

AN MVR.

=HALF-ADJUST IS VALID ONLY FOR NUMERIC FIELDS WITH GREATER THAN ZERO DECIMAL POSITIONS IN THE RESULT FIELD.

OPERATIONALLY, THE HALF-ADJUST OPTION PERFORMS A ROUNDING OPERATION ON THE RESULT FIELD. TO ACHIEVE HALF-ADJUST, + OR - 5 (DEPENDING UPON THE SIGN OF THE RESULT) IS ADDED TO THE DIGIT IMMEDIATELY TO THE RIGHT OF THE RIGHTMOST RESULTING DECIMAL POSITION.

EXAMPLE:

RESULT IS 1.25618

RESULTING FIELD IS TO BE 4 WITH 2 DECIMAL POSITIONS

SO, 1.25618 + 0.00500 = 1.26118

THEN THE DIGITS TO THE RIGHT OF THE RIGHTMOST RESULTANT DIGIT ARE DROPPED, GIVING THE FINAL RESULT OF 1.26.

SUBJECT: MAJOR CHANGES TO THE REV. 16 MASTER DISK

THE FOLLOWING COMMANDS HAVE BEEN REMOVED FROM THE REV. 16 MASTER DISK: LOAD20, MCG. LOAD20 IS A VERSION OF THE LOADER FOR USE UNDER A 16K VERSION OF PRIMOS 2. ANYONE STILL USING IT SHOULD NOW USE LOAD INSTEAD. MCG IS THE MICROCODE ASSEMBLER FOR THE PRIME 300 AND IS NO LONGER SUPPORTED.

THE FOLLOWING COMMANDS WILL BE REMOVED FROM THE MASTER DISK AT REV. 17: CNVTMA, HILOAD, PUSS. CNVTMA IS A UTILITY TO CONVERT LOADMAPS GENERATED BY HILOAD FOR USE BY PSD. IT WILL NO LONGER BE USEFUL WHEN HILOAD IS REMOVED. HILOAD IS AN OBSOLETE VERSION OF THE LOAD COMMAND. USERS SHOULD CONVERT THEIR COMMAND FILES TO USE LOAD BY REV. 17. PUSS IS A SOURCE FILE COMPARE PROGRAM THAT HAS THE SAME FUNCTIONALITY AND MUCH SLOWER SPEED THAN CMPF. USERS SHOULD SWITCH TO CMPF BY REV. 17.

STARTING AT REV. 16, THE CURRENT MASTER DISK WILL NOT BE SHIPPED TO PRIME 100, 200 AND 300 CUSTOMERS. THEY WILL RECEIVE THE REV. 15 MASTER DISK. ANY BUGS REPORTED FOR THAT VERSION OF THE MASTER DISK WILL BE INVESTIGATED, FIXED, AND RELEASED AS AN UPDATE TO THE REV. 15 MASTER DISK. AS A RESULT, THE FOLLOWING SOFTWARE, WHICH IS ONLY USEFUL FOR PRIME 100, 200 AND 300 CUSTOMERS, HAS BEEN REMOVED FROM THE REV. 16 MASTER DISK: ALL VERSIONS OF PRIMOS 3 AND THE RDOS AND SDOS VERSIONS OF PRIMOS 2. FURTHERMORE, THE FOLLOWING SOFTWARE HAS BEEN MOVED FROM THE B1 MASTER DISK PARTITION TO THE A1 MASTER DISK PARTITION: PRIMOS 4, SEG, VPSD, VPSD16, AND FORTRAN LIBRARY. ALL SOFTWARE FORMERLY FOUND IN UFD LIB AND SYSTEM ON PARTITION B1 HAS BEEN MOVED TO THE CORRESPONDING UFD'S ON PARTITION A1.

STARTING AT REV. 16, THE SHARED LIBRARIES WILL BE DEFAULT. THE SHARED LIBRARIES ARE: PFTNLB, VCOBLB, VKDALB, AND VFORMS IN UFD LIB. TO WORK, THE SHARED LIBRARIES MUST BE INSTALLED AT PRIMOS SYSTEM STARTUP TIME USING THE SHARE COMMAND AS DESCRIBED IN THE PRIMOS DOCUMENTATION FOR REV. 16. THE SHARED LIBRARIES CONSIST OF SEVERAL FILES FOR EACH LIBRARY. DO NOT SHARE THE ABOVE NAMED FILES. NOTE THAT UNLESS SHARED LIBRARIES ARE INSTALLED, THE FOLLOWING THINGS WILL NOT RUN: COBOL COMPILER, DBMS COMMANDS, ED, CX, MQL AND MSCH WHICH COMPRISE THE MIDAS QUERY LANGUAGE, AND BASICV.

THE NONSHARED LIBRARIES ARE SUPPLIED IN UFD LIB FOR THOSE THAT WISH TO USE THEM. THEY ARE NPFTNLB AND IFTNLB FOR THE FORTRAN LIBRARY, NVCOBLB FOR THE COBOL LIBRARY, NVKDALB FOR THE KIDA LIBRARY AND NVFORMS FOR THE FORMS LIBRARY. NOTE THAT USING THE SHARED FORMS LIBRARY REQUIRES A SMALL SOURCE CHANGE TO PROGRAMS THAT USE IT.

AT REV. 16, THE COMMAND ED WILL BE SHARED BY DEFAULT. TO USE ED, THE SHARED LIBRARIES MUST BE INSTALLED AND THE FOLLOWING COMMAND MUST BE GIVEN AT THE SUPERVISOR CONSOLE: SHARE SYSTEM>ED2000 2000.

TO EDIT FILES UNDER PRIMOS 2, USE THE COMMAND NSED. AN ATTEMPT TO USE

ED UNDER PRIMOS 2 WILL CAUSE THE MACHINE TO HALT.

SUBJECT: MODIFICATIONS TO RUNOFF FOR 16.2

1) RBAR HAS BEEN MODIFIED AND WILL NOW WORK CORRECTLY. RUNOFF USED TO PRINT AN EXTRA RBAR IF THE RBAR OFF COINCIDED WITH THE BEGINNING OF A NEW OUTPUT LINE. TAR 11200 DEALT WITH THIS PROBLEM

2) UNDERSCORE NO LONGER MISSES ONE SPACE IN ADJUST MODE.

3) ADJUST MODE DID NOT ALWAYS ADJUST QUITE RIGHT WHEN DEALING WITH UNDERLINED TEXT THAT ALSO INCLUDED PHANTOM HYPHENS. THIS HAS BEEN CORRECTED.

4) IF A BREAK HAPPENED TO COINCIDE WITH THE LEFT MARGIN AFTER A HANGING INDENT THE BREAK DID NOT TAKE EFFECT. THIS HAS BEEN CORRECTED.

5) IF THE COMMAND ERRGO HAS BEEN GIVEN AND ERRORS DO OCCUR, RUNOFF WILL NOW EXIT NORMALLY SO THAT COMMAND OR PHANTOM FILES WILL CONTINUE TO RUN. THE ERRORS ARE STILL FLAGGED.

6) ILLEGAL AND UNRECOGNIZED COMMAND MESSAGES WILL NOW LIST THE LINE NUMBER AND FILE IN WHICH THE ERROR WAS FOUND. OTHER ERRORS WILL ALSO NOW GIVE THE NAME OF THE FILE THAT CAUSED THE ERROR.

SUBJECT: CHANGES TO SORT AND SORT LIBRARIES

A. VSRTLI CHANGES AT REVS 15.5 AND 16.2

THE FOLLOWING BUG HAS BEEN FIXED:

IF THE COMMON BLOCK EB$4 WAS LOADED AT WORD 0 OF ANY SEGMENT, THE SORT WOULD HAVE UNPREDICTABLE RESULTS.

THE FIX WAS MADE IN THE ROUTINE VGETLL.

THIS FIX IS IN RESPONSE TO TAR 19975.

B. SORT CHANGES AT REVS 15.4 AND 16.1

THE FOLLOWING BUG HAS BEEN FIXED:

THE SORT COMMAND WOULD NOT ALLOW 8, 9, 18, OR 19 KEYS.

THE FIX WAS MADE IN THE ROUTINE SETSIZ ON UFD LIB7. THE SORT COMMAND SHOULD BE REBUILT WITH THE CORRECTED ROUTINE.

THIS FIX IS IN RESPONSE TO TAR 15451.

C. VSRTLI (V-MODE SORT LIBRARY) CHANGES AT REVS 15.3 AND 16.0

1. FOR CONSISTENCY WITH THE R-MODE SORT LIBRARY, CALLS TO THE SUBROUTINE ASCSRT MAY NOW BE MADE AS CALLS TO THE SUBROUTINE ASCS$$.

2. THE V-MODE SORT LIBRARY'S INTERNAL ROUTINE SPACE HAS BEEN RENAMED SPAC$S TO AVOID NAMING CONFLICTS WITH USERS.

D. NAMING CONVENTION FOR REV 17 AND BEYOND

1. ADOPTION OF A NAMING CONVENTION SIMILAR TO THAT OF THE APPLICATION LIBRARY WILL BE BENEFICIAL IN AVOIDING THE POSSIBILITY OF A CONFLICT WITH USER WRITTEN ROUTINES AND SYSTEM ROUTINES.

2. EXISTING ENTRY POINTS: SUBSRT, ASCS$$, ASCSRT (V-MODE ONLY), AND COMMON BLOCK NAMES: EB$1, EB$2, EB$3, EB$4, EB$5, WILL NOT BE CHANGED, BUT ALL OTHER NAMES WILL END WITH THE SUFFIX "$S".

SUBJECT: BUG FIXES TO MAGSAV AND MAGRST FOR REVISION 16

THE FOLLOWING CORRECTIONS WERE MADE TO MAGSAV AND MAGRST FOR REVISION 16:

1. MAGRST PRINTS AN ERROR MESSAGE AND PAUSES WHEN IT DETECTS THAT THE TAPE HAS GONE OFFLINE. PREVIOUSLY THE PROGRAM ATTEMPTED ERROR RECOVERY INDEFINITELY.

2. MAGRST CONVERTS COMMAND LINES TO UPPERCASE WHEN READING TREENAMES FOR A PARTIAL RESTORE.

3. THE ERROR MESSAGE FOR RECOVERED TAPE ERRORS IS PRINTED OUT BEFORE THE INDEX FILE IS CLOSED. THF ERROR MESSAGE WILL APPEAR BOTH ON THE TERMINAL AND IN THE INDEX FILE. THIS CHANGE WAS MADE FOR MAGSAV AND MAGRST.

4. THE REEL NUMBER IS DISPLAYED WHEN MAGRST ASKS FOR THE NEXT TAPE TO BE MOUNTED.

5. THE PREVIOUS VERSION OF MAGSAV STOPPED WITH 'STOP :1101' WHEN AN ERROR WAS DETECTED WHILE WRITING THE TWO FILE MARKS AT THE LOGICAL END OF TAPE. THE PROGRAM NOW TRIES UP TO TEN TIMES TO WRITE THE FILE MARK. IF IT FAILS, IT STOPS WITH 'STOP :1101' AS BEFORE. IF IT SUCCEEDS, IT PRINTS THE FOLLOWING ERROR MESSAGE AND CONTINUES:

   DUE TO BAD TAPE, ADDITIONAL TAPE MARKS WERE ADDED TO THE END LOGICAL TAPE. DO NOT APPEND OTHER MAGSAV LOGICAL TAPES TO THIS REEL.

SUBJECT: EDITOR CHANGES FOR 16.2

THE FOLLOWING CHANGES HAVE BEEN MADE TO THE EDITOR:

1)FIND AND NFIND WILL NOW WORK IF YOU ARE LOOKING FOR BLANKS OR NON-BLANKS BEYOND THE END OF MOST LINES. PREVIOUSLY IF YOU SAID

NF(73)

LOOKING FOR LINES THAT DID NOT HAVE A BLANK IN COLUMN 73, EDITOR STOPPED AT ANY LINE SHORTER THAN 73 COLUMNS, AS WELL AS AT LINES CONTAINING A CHARACTER IN COLUMN 73. SIMILARLY

F(73)

WOULD NOT HAVE FOUND LINES SHORTER THAN 73 COLUMNS. NOW THE EDITOR WILL ONLY FIND THOSE LINES THAT ACTUALLY HAVE SOMETHING PRINTED IN COLUMN 73 FOR NFIND AND WILL FIND THOSE SHORTER THAN 73 FOR FIND.

2) IF YOU TYPE A COMMAND THAT GIVES YOU THE ERROR MESSAGE BAD L OR YOU WILL STILL BE POSITIONED AT THE SAME PLACE AS WHEN YOU GAVE THE COMMAND. PREVIOUSLY THE POINTER WAS ADVANCED ONE LINE WHEN AN L WAS TYPED BEFORE CHECKING IF THE COMMAND WAS VALID. THE EDITOR WILL NOW CHECK FIRST FOR CORRECT SYNTAX AND LEAVE THE POINTER POINTED TO WHERE IT WAS WHEN THE COMMAND WAS GIVEN.

3) IF YOU HIT CONTROL-P OR BREAK WHILE IN EDITOR THE BREAK WILL NOT TAKE EFFECT WHILE FILE POINTERS ARE BEING UPDATED. THIS SHOULD INSURE THAT THE FILE WILL NOT BE BROKEN BECAUSE OF A BREAK. THE USER MAY HAVE TO WAIT A LITTLE LONGER FOR THE BREAK TO BE ACKNOWLEDGED. HE WILL STILL HAVE TO DO A WHERE TO FIND OUT WHERE IN THE FILE HE IS CURRENTLY LOCATED. IF HE HIT BREAK DURING A GLOBAL CHANGE, SOME OF THE CHANGES MAY HAVE OCCURRED AND SOME NOT.

4) TYPING AN UNLOAD OR DUNLOAD COMMAND AND FORGETTING TO GIVE A FILENAME BUT TYPING A NUMBER FOR LINES TO MOVE WILL NOW CAUSE THE EDITOR TO ISSUE AN ERROR MESSAGE PERTAINING TO A FILE UNIT. IF THE NUMBER IS LESS THAN 5 THE MESSAGE IS "BAD FILE UNIT" BECAUSE THE EDITOR USES THE FIRST FOUR FILE UNITS, IF 5 OR GREATER THE ERROR MESSAGE WILL BE "UNIT NOT OPEN FOR WRITING". THIS IS BECAUSE THE EDITOR ALLOWS ONE TO UNLOAD TO A FILE UNIT THAT HAS BEEN PREVIOUSLY OPENED. (SEE PE-IN-74 REV.3)

5) THE SHARED EDITOR SETS UP ITS OWN STACK HEADER BUT DID NOT SET THE PB AND LB RETURNS, THIS WAS CAUSING PROBLEMS WITH THE REV 17 COMMAND ENVIRONMENT. THIS HAS BEEN CORRECTED.

6) MODE PROMPT CAUSED THE COLUMN HEADING BANNER PRINTED WITH MODE COLUMN TO BE OFF BY 2 COLUMNS WHEN IT WAS PRINTED IN INPUT MODE WHEN THE EDITOR IS IN MODE PROMPT. THIS HAS BEEN CORRECTED BY MOVING THE BANNER OVER 2 COLUMNS WHEN IN MODE PROMPT. IT SHOULD BE REMEMBERED THAT THE BANNER WILL NOT APPEAR LINED UP CORRECTLY WITH LINES THE EDITOR PRINTS IN EDIT MODE SINCE THESE STILL START IN COLUMN 1. (TAR 14505)

EXAMPLE:

```
MODE PROMPT
$ MODE COLUMN
$
INPUT
          1         2         3         4         5         6
 12345678901234567890123456789012345678901234567890123456789012345678
& THIS IS AN EXAMPLE
&
EDIT
$ P
THIS IS AN EXAMPLE
```

DATE: DECEMBER 1, 1977

SUBJECT: CMPF AND MRGF COMMANDS

TWO NEW COMMANDS HAVE BEEN PROVIDED TO HELP EASE THE PROBLEMS OF PARALLEL SOFTWARE DEVELOPMENT. CMPF PROVIDES A FACILITY SIMILAR TO THE PUSS COMMAND, EXCEPT THAT IT RUNS FASTER THAN PUSS AND PRODUCES MORE MEANINGFUL OUTPUT THAN PUSS. THE MRGF COMMAND IS A POWERFUL TOOL DESIGNED TO ALLOW AUTOMATED MERGING OF PROGRAM CHANGES. MRGF OBVIATES THE NEED FOR TEDIOUS EDITING OF PROGRAMS WHEN TWO (OR MORE) SETS OF CHANGES MADE TO A PROGRAM ARE TO BE COMBINED. IT IS EXPECTED, HOWEVER, THAT THE RESULTANT MERGED OUTPUT WILL BE EXAMINED CAREFULLY BEFORE IT IS USED.

CMPF NOEL I. MORRIS NOVEMBER 21, 1977

THE CMPF COMMAND ALLOWS A USER TO COMPARE UP TO FIVE ASCII FILES. ONE FILE IS TREATED AS AN ORIGINAL FILE. THE CMPF COMMAND PRODUCES OUTPUT SHOWING LINES THAT WERE ADDED TO, CHANGED FROM, OR DELETED FROM THE ORIGINAL FILE IN THE OTHER FILES.

USAGE:

CMPF FILEA FILEB [FILEC ... FILEE] [-CONTROL_ARGS]

FILEA THROUGH FILEE ARE THE TREE NAMES OF THE FILES TO BE COMPARED.

CONTROL ARGS:

-MINL # SETS THE MINIMUM NUMBER OF LINES WHICH MUST MATCH FOLLOWING A DISCREPANCY IN ORDER TO RESYNCH ALL FILES. THE DEFAULT VALUE IS -MINL 3.

-BRIEF (OR -BF) SUPPRESSES THE PRINTING OF DIFFERING LINES. ONLY THE FILE IDENTIFICATION AND LINE NUMBERS ARE PRINTED.

-REPORT REPORT_FILE_NAME PRODUCES A FILE CONTAINING THE DISCREPANCIES INSTEAD OF PRINTING THEM OUT ON THE USER'S TERMINAL.

OPERATION:

FILEA IS TREATED AS AN ORIGINAL FILE (I.E. AS A FILE WHICH IS THE COMMON ANCESTOR OF FILEB THROUGH FILEE). FILEA IS COMPARED LINE BY LINE WITH EACH OF THE OTHER FILES. WHEN A DISCREPANCY IS FOUND BETWEEN FILEA AND ANY OTHER FILE, CMPF ATTEMPTS TO GET ALL FILES BACK IN SYNCH. REMATCHING IS COMPLETED ONLY WHEN A CERTAIN MINIMUM NUMBER OF LINES MATCH IN ALL FILES. THIS MINIMUM NUMBER IS SETTABLE WITH THE -MINL CONTROL ARGUMENT. AFTER RESYNCHRONIZATION IS COMPLETE, LINES WHICH DIFFER BETWEEN FILEA AND ANY OF THE OTHER FILES ARE REPORTED, AND THE COMPARISON CONTINUES. WHEN THE DISCREPANCY IS REPORTED, EACH LINE FROM FILEA IS IDENTIFIED BY PRECEDING IT WITH THE LETTER "A" AND THE LINE NUMBER OF THAT LINE. LINES OF FILEB THROUGH FILEE ARE SIMILARLY IDENTIFIED, USING THE LETTERS "B" THROUGH "E", RESPECTIVELY. THE -BRIEF CONTROL ARGUMENT CAUSES ONLY THE FILE IDENTIFICATION LETTER AND THE LINE NUMBERS OF THE DIFFERING LINES TO BE PRINTED.

NOTES:

THE CMFF COMMAND COMPARES COMPRESSED LINES OF ANY LENGTH. IT ASSUMES THE FILES OF COMMON ANCESTRY WILL CONTAIN LINES COMPRESSED IN IDENTICAL FASHION. IT IS, HOWEVER, POSSIBLE FOR A MISMATCH TO OCCUR BETWEEN TWO LINES WHICH APPEAR IDENTICAL, BUT WHICH WERE COMPRESSED DIFFERENTLY. THIS POSSIBILITY IS CONSIDERED TO BE REMOTE.

THE -REPORT CONTROL ARGUMENT OF CMPF MAKES USE OF THE COMMAND OUTPUT (COMO$$) MECHANISM OF PRIMOS. A CMPF WHICH PRODUCES A REPORT FILE SHOULD, THEREFORE, NOT BE INVOKED WITH A COMMAND OUTPUT FILE ALREADY IN USE.

EXAMPLE:

CONSIDER THE FOLLOWING TWO FILES:

FILEA	FILEB
THF	THE
QUICK	NASTY
BROWN	BROWN
FOX	FOX
JUMPS	JUMPS
OVER	OVER
THE	THE
LAZY	DOG
DOG	

A CMPF OF THESE TWO FILES WOULD PRODUCE THE FOLLOWING OUTPUT:

```
A2        QUICK
CHANGED TO
B2        NASTY

A8        LAZY
DELETED BEFORE
B8        DOG

COMPARISON FINISHED.
2 DISCREPANCIES FOUND.
```

MRGF NOEL I. MORRIS NOVEMBER 23, 1977

THE MRGF COMMAND ALLOWS A USER TO MERGE BETWEEN TWO AND FIVE ASCII FILES. ONE FILE IS TREATED AS AN "ORIGINAL" FILE TO WHICH CHANGES HAVE BEEN MADE IN THE OTHER FILES. UNCHANGED LINES AND UNCONFLICTING CHANGES BETWEEN FILES ARE COPIED AUTOMATICALLY INTO THE OUTPUT FILE. WHEN CONFLICTS EXIST, THE USER CAN BE QUERIED TO RESOLVE THE CONFLICT MANUALLY.

MRGF IS ESPECIALLY USEFUL FOR COMBINING DIFFERENT CHANGES TO A PROGRAM WHICH HAVE BEEN MADE IN PARALLEL BY SEVERAL PROGRAMMERS. IT CAN ALSO BE USEFUL FOR DISTRIBUTING SOFTWARE CHANGES TO OTHER SITES.

USAGE:

MRGF ORIGFILE FILEB [FILEC ... FILEE] OUTPUTFILE [-CONTROL_ARGS]

ORIGFILE IS THE TREE NAME OF THE "ORIGINAL" FILE. FILEB THROUGH FILEE ARE TREE NAMES OF FILES WHICH TRACE THEIR ANCESTRY TO ORIGFILE.

OUTPUTFILE IS THE TREE NAME OF THE MERGED OUTPUT FILE.

CONTROL ARGS:

-MINL # SETS THE MINIMUM NUMBER OF LINES WHICH MUST MATCH FOLLOWING A DISCREPANCY IN ORDER TO RESYNCH ALL FILES. THE DEFAULT VALUE IS -MINL 3.

-FORCE CAUSES FILEB TO BE A "PREFERRED" FILE WHEN CONFLICTS EXIST BETWEEN SEVERAL FILES. WHEN -FORCE IS USED, THE USER OF MRGF IS NEVER QUERIED TO RESOLVE A CONFLICT (SEE BELOW).

-BRIEF (OR -BF) SUPPRESSES THE PRINTING OF CONFLICTING LINES. ONLY THE FILE IDENTIFICATION AND LINE NUMBERS ARE PRINTED.

-REPORT REPORT_FILE_NAME PRODUCES A FILE CONTAINING THE DISCREPANCIES FOUND BETWEEN FILES DURING THE MERGE. RESOLVABLE DISCREPANCIES ARE NOT DISPLAYED ON THE USER'S TERMINAL. UNRESOLVABLE DISCREPANCIES WILL BE PLACED IN THE REPORT FILE AS WELL AS DISPLAYED ON THE USER'S TERMINAL.

OPERATION:

ORIGFILE IS TREATED AS AN ORIGINAL FILE (I.E. AS A FILE WHICH IS THE COMMON ANCESTOR OF FILEB THROUGH FILEE). ORIGFILE IS COMPARED LINE BY LINE WITH EACH OF THE OTHER FILES. LINES WHICH MATCH IN ALL FILES ARE COPIED INTO OUTPUTFILE AUTOMATICALLY. WHEN A DISCREPANCY IS FOUND BETWEEN ORIGFILE AND ANY OTHER FILE, MRGF ATTEMPTS TO GET ALL FILES BACK IN SYNCH. REMATCHING IS COMPLETED ONLY WHEN A CERTAIN MINIMUM NUMBER OF LINES MATCH IN ALL FILES. THIS MINIMUM NUMBER IS SETTABLE WITH THE -MINL CONTROL ARGUMENT.

AFTER RESYNCHRONIZATION IS COMPLETE, SELECTION OF LINES TO BE OUTPUT MUST TAKE PLACE. IF ONLY ONE FILE DIFFERED FROM ORIGFILE, THE CHANGES IN THAT FILE ARE COPIED INTO OUTPUTFILE AUTOMATICALLY. IF ALL FILES DIFFERED IDENTICALLY FROM THE ORIGINAL, THOSE CHANGES ARE ALSO COPIED AUTOMATICALLY. IF CONFLICTING CHANGES ARE FOUND IN SEVERAL FILES, (OR IF ONLY ONE FILE IS BEING MERGED WITH THE ORIGINAL), THE USER CAN SELECT MANUALLY WHICH LINES ARE TO BE COPIED INTO OUTPUTFILE. IF THE -FORCE CONTROL ARGUMENT IS USED, SUCH CONFLICTS ARE RESOLVED

AUTOMATICALLY. THE USER IS NOT QUERIED, AND THE CHANGES IN FILEB ARE TAKEN AS THE "PREFERRED" CHANGES TO BE INSERTED INTO OUTPUTFILE.

IF THE -FORCE CONTROL ARGUMENT IS NOT USED, THE DIFFERING LINES FROM EACH OF THE FILES ARE REPORTED. EACH LINE FROM ORIGFILE IS IDENTIFIED BY PRECEDING IT WITH THE LETTER "A" AND THE LINE NUMBER OF THAT LINE. LINES OF FILEB THROUGH FILEE ARE SIMILARLY IDENTIFIED, USING THE LETTERS "B" THROUGH "E", RESPECTIVELY. THE -BRIEF CONTROL ARGUMENT CAUSES ONLY THE FILE IDENTIFICATION LETTER AND THE LINE NUMBERS OF THE DIFFERING LINES TO BE PRINTED. AFTER AN UNRESOLVABLE DISCREPANCY IS REPORTED, EDIT MODE IS ENTERED TO ALLOW THE USER TO SELECT LINES TO BE PLACED IN OUTPUTFILE (SEE BELOW). AFTER SELECTION (EITHER AUTOMATIC OR MANUAL) IS COMPLETED, THE LINE BY LINE COMPARISON CONTINUES.

IF THE -REPORT CONTROL ARGUMENT IS USED, THE RESULTANT REPORT FILE CONTAINS ALL DISCREPANCIES BETWEEN FILES --- THAT IS, BOTH THE RESOLVABLE AND THE UNRESOLVABLE DIFFERENCES. UNRESOLVABLE DIFFERENCES ARE ALWAYS DISPLAYED ON THE USER'S TERMINAL AS WELL. RESOLVABLE DIFFERENCES ARE, HOWEVER, NEVER DISPLAYED ON THE USER'S TERMINAL. THE ACTION TAKEN BY MRGF (OR THE USER) IS PLACED IN THE REPORT FILE FOLLOWING EACH DISCREPANCY.

MANUAL SELECTION:

AFTER EACH UNRESOLVABLE DISCREPANCY IS DISPLAYED, EDIT MODE IS ENTERED. THE USER MUST SELECT WHICH LINES ARE TO BE INSERTED INTO OUTPUTFILE BY ISSUING THE FOLLOWING COMMANDS:

A	INSERT ALL OF THE DIFFERING LINES IN ORIGFILE.
B	INSERT ALL OF THE DIFFERING LINES IN FILEB.
C	INSERT ALL OF THE DIFFERING LINES IN FILEC.
D	INSERT ALL OF THE DIFFERING LINES IN FILED.
E	INSERT ALL OF THE DIFFERING LINES IN FILEE.
AN	INSERT LINE N OF ORIGFILE.
BN	INSERT LINE N OF FILEB. (SIMILARLY FOR FILEC THROUGH FILEE)
AM,N	INSERT LINES M THROUGH N OF ORIGFILE. (SIMILARLY FOR FILEB THROUGH FILEE)
PA	PRINT ALL OF THE DIFFERING LINES IN ORIGFILE. (SIMILARLY FOR FILEB THROUGH FILEE)
PAN	PRINT LINE N OF ORIGFILE. (SIMILARLY FOR FILEB THROUGH FILEE)
PAM,N	PRINT LINES M THROUGH N OF ORIGFILE. (SIMILARLY FOR FILEB THROUGH FILEE)
OOPS	UNDO ALL PREVIOUS EDITING FOR THIS DISCREPANCY.
GO	TERMINATE EDITING AND PROCEED WITH MERGE.
QUIT	TERMINATE EDITING, CLOSE ALL FILES, AND EXIT FROM MRGF.

IN ADDITION TO THE ABOVE, NEW TEXT CAN BE INSERTED AT ANY POINT IN A DISCREPANCY BY ENTERING A BLANK LINE. INPUT MODE IS ENTERED, AND LINES TYPED WILL BE COPIED INTO OUTPUTFILE. A BLANK LINE WILL TERMINATE INPUT. NOTE THAT NO TEXT EDITING CAN BE PERFORMED ON LINES WHICH ARE COPIED OR INPUTTED. NO TAB EXPANSION IS PERFORMED ON INPUTTED LINES.

NOTES:

THE MRGF COMMAND OPERATES ON COMPRESSED LINES OF ANY LENGTH. IT ASSUMES THAT FILES OF COMMON ANCESTRY WILL CONTAIN LINES COMPRESSED IN IDENTICAL FASHION. IT IS, HOWEVER, POSSIBLE FOR A MISMATCH TO OCCUR

BETWEEN TWO LINES WHICH APPEAR IDENTICAL, BUT WHICH WERE COMPRESSED DIFFERENTLY. THIS POSSIBILITY IS CONSIDERED TO BE REMOTE.

THE -REPORT CONTROL ARGUMENT OF MRGF MAKES USE OF THE COMMAND OUTPUT (COMO$$) MECHANISM OF PRIMOS. A MRGF WHICH PRODUCES A REPORT FILE SHOULD, THEREFORE, NOT BE INVOKED WITH A COMMAND OUTPUT FILE ALREADY IN USE.

EXAMPLE:

CONSIDER THE FOLLOWING THREE FILES:

FILEA	FILEB	FILEC
THE	THE	THE
QUICK	QUICK	QUICK
BROWN	RED	BROWN
FOX	FOX	FOX
JUMPS	JUMPS	JUMPS
OVER	OVER	OVER
THE	THE	THE
LAZY	SLEEPING	SNORING
DOG	DOG	DOG

A MRGF OF THESE FILES WOULD PRODUCE THE FOLLOWING:

```
      A8        LAZY
      CHANGED TO
      B8        SLEEPING
      BUT ALSO CHANGED TO
      C8        SNORING
      EDIT
    $ B
    $ GO

      MERGE FINISHED.
      1 MANUAL CHANGE.
      1 AUTOMATIC CHANGE AS FOLLOWS:
           1 FROM FILE B.
```

IN THE ABOVE EXAMPLE, THE LINES PRECEDED BY A "$" WERE TYPED BY THE USER. THE MERGED OUTPUT FILE FROM THE ABOVE MRGF WOULD APPEAR AS FOLLOWS:

```
THE
QUICK
RED
FOX
JUMPS
OVER
THE
SLEEPING
DOG
```

NOTE THAT IF THE -FORCE CONTROL ARGUMENT HAD BEEN USED, THE SAME MERGED OUTPUT WOULD HAVE BEEN PRODUCED. HOWEVER, THE CHANGE FROM FILEB WOULD HAVE BEEN INSERTED AUTOMATICALLY, AND THE USER WOULD NOT HAVE BEEN QUERIED.

ACKNOWLEDGEMENT:

THE MRGF COMMAND IS BASED ON THE MERGE_ASCII COMMAND OF MULTICS, WHICH WAS IMPLEMENTED BY ROBERT E. MULLEN OF HONEYWELL INFORMATION SYSTEMS, INC. THE ALGORITHMS USED IN THE MRGF COMMAND WERE "BORROWED" EXTENSIVELY FROM THOSE DEVELOPED BY MULLEN.

SUBJECT: COBOL, REV 15.0

1. INTRODUCTION

THIS DOCUMENT DESCRIBES THE CHANGES BETWEEN REV 14 AND REV 15 COBOL. THE COMPILER HAS MANY USER VISIBLE ENHANCEMENTS.

2. LARGER ADDRESS SPACE

REV 14 AND BELOW WERE RESTRICTED TO A MAXIMUM OF A 64K BYTE ADDRESS SPACE. THIS WAS FURTHER CUT DOWN BY 4K BYTES FOR EACH FILE DECLARED AND FOR EACH ARGUMENT PASSED. THE SIZE OF A DATA ITEM (GROUP OR ELEMENTARY) COULD NOT EXCEED 4K BYTES.
THESE RESTRICTIONS HAVE BEEN RELAXED OR REMOVED FOR REV 15. THE NEW CHARACTERISTICS ARE:

- THE TOTAL ADDRESS SPACE WHICH A PROGRAM USES NO LONGER HAS AN EXPLICIT LIMIT.
- THE MAXIMUM DATA ITEM SIZE IS NOW 32K BYTES.
- THE OCCURS COUNT MAY NOT EXCEED 32767.
- THERE MAY NOW BE UP TO 126 FILES DECLARED. OBVIOUSLY, THERE ARE INSUFFICIENT FILE SYSTEM UNITS AVAILABLE TO SUPPORT THIS MANY FILES OPEN SIMULTANEOUSLY.

NATURALLY, IN R MODE, THE TOTAL PROGRAM + DATA SIZE MUST NOT EXCEED 64K WORDS. IN 64V MODE, THIS EXTENDED ADDRESSING IS DONE THROUGH COMPILER GENERATED COMMON BLOCKS.

3. STREAMLINED COMPILER

THE REV 15 COBOL COMPILER IS ROUGHLY TWICE AS FAST AS OLDER COMPILERS. WORKING SET SIZE HAS ALSO BEEN SIGNIFICANTLY REDUCED, SO COMPILATION SPEED ON SMALL MEMORY SYSTEMS SHOULD IMPROVE SIGNIFICANTLY.

4. EXTENSIONS

REV 15 COBOL HAS THE FOLLOWING NEW FEATURES:

. OPEN EXTEND FOR SEQUENTIAL DISK FILES

. FULL IF STATEMENTS (EXCEPT ARITHMETIC EXPRESSION OPERANDS)

. V MODE MAG TAPE SUPPORT

5. BUG FIXES

. COPY STATEMENTS MAY NOW CONTAIN TEXT AFTER THE COPY CLAUSE. IN THE LISTING FILE THE LINE NUMBERING OF THE COPY FILE IS NOW INDEPENDANT OF THE LINE NUMBERS OF THE SOURCE. FOR EXAMPLE:

```
(0069)          COPY FILE.   VALUE 213.
[0001]                INSERTED
[0002]                  TEXT
[0003]                   ..
[NNNN]                   ..
(0069)          COPY FILE.   VALUE 213.
(0070)
(0071)          ETC, ETC, ETC.
```

. LEVEL 88 (DECIMAL) IS NOW FUNCTIONING PROPERLY FOR ALL CASES.

. COMP-3 USAGE NO LONGER CAUSES 'INCOMPLETE TREE' PROBLEMS.

. SYNTAX ONLY COMPILATION (-B NO) IS NOW WORKING CORRECTLY.

. 'DECIMAL POINT IS COMMA' IS FUNCTIONAL IN 64V.

6. IMPORTANT NOTE

REV 14 AND EARLIER COBOL LIBRARIES ARE INCOMPATIBLE WITH THE REV 15 COMPILER.

DATE: NOVEMBER 21, 1978

SUBJECT: FORMS, REV. 16.2

1 SCOPE

THIS DOCUMENT DESCRIBES THE CHANGES MADE TO THE FORMS MANAGEMENT SYSTEM AT SOFTWARE REVISION 16.2. IT SUPPLEMENTS REVISION 4 OF PE-T-296 AND REVISION 1 OF PE-T-400.

2 FAP UPDATES

THE TCB COMMAND HAS BEEN UPDATED TO ALLOW THE "CURRENT USER NUMBER" TO BE IDENTIFIED BY AN ASTERISK ("*") IN LIEU OF THE ACTUAL VALUE. FOR EXAMPLE,

```
* TCB * OWL1200        CHANGES THE USER'S TERMINAL TO OWL1200
* TCB *                DROPS THE TCB ENTRY FOR THE CURRENT USER
```

ADDITIONALLY, A NUMERIC ARGUMENT (OR "*") MAY FOLLOW THE TCB LIST COMMAND TO REQUEST THE TERMINAL TYPE FOR THE SPECIFIED USER NUMBER ONLY. F.G.,

```
* TCB LIST 20
VISTAR3
* TCB LIST *
UNDEFINED.
```

3 REMOTE LOGIN SUPPORT

FORMS WILL NOW FUNCTION PROPERLY ACROSS A REMOTE LOGIN USING X.25. AS IT IS IMPOSSIBLE TO IDENTIFY THE TERMINAL (THRU THE USER NUMBER) ON REMOTE LOGIN, USING A PUBLIC DATA NETWORK, OR USING DIAL-UP LINES, A NEW PSEUDO-TERMINAL TYPE, "INQUIRE", HAS BEEN DEFINED. WHEN A USER NUMBER IS ASSOCIATED WITH THIS PSEUDO-TERMINAL TYPE (THRU FAP'S TCB COMMAND), THE OPERATOR IS QUERIED AT PROGRAM EXECUTION TIME FOR THE ACTUAL TYPE OF THE TERMINAL.

4 I/O BUFFER REDEFINITION

THE PROBLEM WHICH PROHIBITED THE REDEFINITION OF THE FORMS I/O BUFFER (COMMON BLOCK IOBCM$) HAS BEEN CORRECTED. USERS ARE REMINDED THAT IT IS IMPOSSIBLE TO REDEFINE THE LENGTH OF THE I/O BUFFER IN THE 64V MODE SHARED VERSION OF THE LIBRARY; THE LIBRARY MUST BE REBUILT AFTER UPDATING THE VALUE OF IOLSIZ DEFINED IN FORMS>RUN>IOLDEF.

THE USER SHOULD IGNORE THE WARNING MESSAGE PRODUCED BY LOAD AFTER REDEFINING IOBCM$.

DATE: 24 AUG 1978

TO: PROGRAMMING AND ENGINEERING STAFF

FROM: LARRY STABILE

SUBJECT: BASIC/VM FOR REV 16 (AND REV 15.5)

PREFACE

REV 16 BASIC/VM IS ENHANCED OVER PREVIOUS VERSIONS BY THE FOLLOWING:

1. MIDAS FILES MAY NOW BE ACCESSED BY A NEW SET OF BASIC STATEMENTS.

2. ARRAY DATA SPACE HAS BEEN INCREASED TO BE LIMITED ONLY BY THE MACHINE CONFIGURATION (MULTI-SEGMENT ARRAYS).

3. SEVERAL RELATIVELY SMALL FUNCTIONAL CHANGES AND THE ADDITION OF A FEW MISCELLANEOUS STATEMENTS.

4. NUMEROUS SMALL BUGS HAVE BEEN REPAIRED, AND THE FOLLOWING TARS HAVE BEEN PROCESSED:

25257 25254 25258 25252 25253 25261 24732 25256 25260
20208 24731 24730

THE FOLLOWING TARS HAVE BEEN RECEIVED BUT NO RESPONSE HAS YET BEEN GENERATED:

20206 24764 25255

THESE TARS ARE BRIEFLY DESCRIBED IN THE LAST SECTION OF THIS PAPER.

LARGE DATA SPACE

ARRAY DATA MAY NOW SPAN MULTIPLE SEGMENTS. IN A CONFIGURATION-DEPENDENT MANNER, A SYSTEM ANALYST MAY ALLOCATE AS MANY SEGMENTS TO ARRAY DATA AS THE PARTICULAR SYSTEM ALLOWS. IN ADDITION, A FULL SEGMENT IS AVAILABLE FOR GENERATED CODE, AS WELL A FULL SEGMENT IS AVAILABLE FOR STRING DATA. THE LARGEST SIZE OF A SINGLE ARRAY DIMENSION (SUBSCRIPT) STILL MAY NOT EXCEED 32767.

ON THE MASTER DISK, TWO COMMAND FILES ARE SUPPLIED TO BUILD A BASIC/VM SYSTEM. C_BASI IS THE STANDARD ONE-SEGMENT VERSION. THIS IS THE VERSION SENT PRE-BUILT WITH THE MASTER DISK. C_BASI.NSEG WILL BUILD A BASIC/VM SYSTEM WITH MULTIPLE DATA SEGMENTS AND A FULL CODE SEGMENT. THE SEG-SEQUENCE FOR C_BASI AND C_BASI.NSEG IS SHOWN BELOW:

C_BASI

```
SEG
LOAD
#BASIC.SHARE
SY LIST$$ 4000 1
SY EDWI$$ 4000 1
SY ESNBAS 0 165777
SY EMAXUN 0 76 /* MAX FILE UNIT NUMBER = 62(DECIMAL)
SY CDESEG 4001 0
SY BOTSEG 4001 0
SY TOPSEG 4001 0
SY F$FLEX 4000 2000
SPLIT 5000 4000 176000
F/S/LO BASICV>BINARY>B_F$FLEX.BV 2000 4000 4000
CO    ABS  4000
A/SY LNKTBL 4000 40
S/LO BASICV>BINARY>B_BASBIN 0 2013 4000
D/LI VKDALB
D/LI VAPPLB
D/PL
S/IL 0 4000 4000
S/LO BASICV>BINARY>B_TOPDAT 0 4000 4000
```

C_BASI.NSEG

```
SEG
LOAD
#BASIC.SHARE
SY LIST$$ 4001 1
SY TOPDAT 4001 0
SY EDWI$$ 4000 1
SY ESNBAS 0 177777
SY EMAXUN 0 76 /* MAX FILE UNIT NUMBER = 62(DECIMAL)
SY CDESEG 4002 0
SY BOTSEG 4003 0
SY TOPSEG 4037 0
SY F$FLEX 4000 2000
SPLIT 5000 4000 176000
F/S/LO BASICV>BINARY>B_F$FLEX.BV 2000 4000 4000
CO    ABS  4000
A/SY LNKTBL 4001 40
S/LO BASICV>BINARY>B_BASBIN 0 2013 4000
D/LI VKDALB
D/LI VAPPLB
D/PL
S/IL 0 4000 4000
```

THE SYMBOL SETTING FOR TOPSEG IS MOST RELEVANT TO TAILORING A SPECIFIC SYSTEM. IT SHOULD BE SY'ED IN THE MANNER SHOWN TO PLACE AN UPPER LIMIT ON THE ARRAY-DATA SEGMENT NUMBER USED. THE TOTAL NUMBER OF SEGMENTS ALLOCATED FOR ARRAY DATA IS THUS DETERMINED BY THIS SYMBOL AND BOTSEG, WHICH SHOULD NOT HAVE TO BE CHANGED. NOTE THAT BASRUN, THE RUN-TIME PACKAGE, MUST BE ASSEMBLED WITH A 2/1 OPTION FOR THE MULTI-SEGMENT SYSTEM. THIS FACT IS INCLUDED IN THE C_BASI.NSEG FILE. ALSO, NOTICE THAT THE TOP SEGMENT ALLOWABLE TO THE DATA SPACE (MAXIMUM SETTING OF TOPSEG) IS LIMITED BY THE PARTICULAR MACHINE/PRIMOS CONFIGURATION ON WHICH BASIC/VM IS RUNNING. BEWARE OF THIS LIMIT OR 'ILLEGAL SEGNO' MAY BE THE RESULT.

MIDAS-ACCESS STATEMENTS

OVERVIEW

BASIC/VM NOW PROVIDES STATEMENTS TO ACCESS MIDAS FILES ON AN ASCII-DATA BASIS. AS SUCH, STRINGS ARE THE FUNDAMENTAL DATA TYPE PASSED BETWEEN A PROGRAM AND A MIDAS FILE. THEY MAY BE USED AS KEYS OR IN DATA RECORDS, AND THE TREATMENT OF THEM IS UNIFORM THROUGHOUT THE MIDAS STATEMENTS.

THIS SECTION DESCRIBES THE CAPABILITIES OF THE MIDAS STATEMENTS, EXPLAINS THEIR USE, AND PROVIDES EXAMPLES OF THEIR USE IN A TYPICAL SMALL APPLICATION.

A MIDAS FILE MAY BE CONSIDERED FROM TWO FUNDAMENTAL ASPECTS. FIRST IS THE SEPARATION OF READING AND WRITING STATEMENTS. READING STATEMENTS ARE READ [KEY], POSITION, AND REWIND. WRITING STATEMENTS ARE UPDATE, ADD, AND REMOVE.

UPDATE IS THE ONLY WRITING OPERATION THAT PERFORMS ITS FUNCTION ON THE CURRENT RECORD. ADD AND REMOVE ARE COMPLETELY INDEPENDENT OF THE CURRENT RECORD POINTER, AND SO THIS POINTER CAN IN GENERAL BE CONSIDERED A READ POINTER.

THE SECOND IMPORTANT ASPECT OF MIDAS IS THE ABILITY TO MODEL THE FILE AS A RECTANGULAR MATRIX OF [INDEX-NUMBER,KEY-POSITION] (SEE FIGURE) WITH A DATA-RECORD POINTER FROM EACH ELEMENT OF THIS MATRIX. READING THE FILE CONSISTS LARGELY OF MOVING THE POINTER AROUND THIS 2-DIMENSIONAL ARRAY. AT EACH POINT, EITHER THE KEY OR THE DATA RECORD POINTED-TO BY THE KEY MAY BE READ. REWIND PLACES THE POINTER AT THE TOP ROW OF THE MATRIX, AT A COLUMN SPECIFIED BY THE KEY (INDEX) NUMBER. INITIALLY, THIS POINTER IS SET TO THE FIRST PRIMARY KEY, IE., TO THE UPPER LEFT CORNER OF THE MATRIX. THIS IS EQUIVALENT TO EXECUTING 'REWIND #N, KEY 0' .

THE CURRENT SET OF MIDAS STATEMENTS ATTEMPTS TO PERFORM A CONSISTENT AND COMPLETE SET OF MOVEMENTS AROUND A MIDAS FILE STRUCTURE. IT IS DESIGNED SO THAT ANY STATEMENTS MAY BE MIXED WITHOUT INCONSISTENT OR UNPREDICTABLE RESULTS. IN PARTICULAR, AN ADD PERFORMED DURING A POSITION-SEQ OR -SAMEKEY SEQUENCE (OR OTHER READ OPERATION) WILL NOT CAUSE THE READ LOCATION TO BE LOST; THERE IS NO STRANGE INTERACTION OF READS WITH WRITES.

STATEMENT FORMATS

DEFINITIONS

[...] - INDICATES ANY ONE OF THE VERTICALLY STACKED ELEMENTS.

\...\ - INDICATES ANY PERMUTATION OF THE ELEMENTS.

[...] - INDICATES OPTIONALITY.

* - INDICATES REPITITION, 0 OR MORE TIMES.

+ - INDICATES REPITITION, 1 OR MORE TIMES.

LOWERCASE - INDICATES A SYNTACTIC TYPE.

****
* DEFINE FILE *
****

DEFINE FILE #NUM_EXPR_1 = STR_EXPR, MIDAS [, NUM_EXPR_2]

OPENS A MIDAS FILE ON GIVEN UNIT. IF NUM_EXPR_2, THE RECORD SIZE, IS GIVEN, THE INTERNAL BUFFERS ARE DIMENSIONED TO THIS VALUE. DEFAULT IS 120 CHARACTERS. MOST EFFICIENT PROCESSING OF A MIDAS FILE WILL BE OBTAINED BY SPECIFYING A RECORDSIZE EQUAL TO THE LENGTH OF THE DATA RECORD, AS DEFINED IN THE MIDAS FILE BY THE CREATK PROCEDURE. NOTE THAT THE RECORDSIZE GIVEN MUST BE IN WORDS.

****
* POSITION *
****

```
                                          SEQ
POSITION #NUM_EXPR_1, [ KEY [NUM_EXPR_2] = STR_EXPR ]
                                        SAME   KEY
```

POSITIONS TO A RECORD IN A MIDAS FILE. POSITION IS SIMILAR TO READ, EXCEPT THAT NO DATA IS RETRIEVED.

```
************
*   READ   *
* READ KEY *
************
```

```
                                                        SEQ
READ [KEY] #NUM_EXPR_1 [, [ KEY [NUMEXPR_2] = STR_EXPR ] ], STR_VAR
                                                      SAME  KEY
```

READS DATA FROM A MIDAS FILE.

NUM_EXPR_1 - FILE UNIT.

STR_VAR - STRING VARIABLE INTO WHICH DATA IS PLACED. IF READ KEY IS SPECIFIED, THE KEY VALUE ITSELF IS READ INTO STR_VAR. THIS IS USEFUL FOR OBSERVING THE ACTUAL KEY AFTER A SEQUENTIAL OR PARTIAL-KEY SEARCH.

NUM_EXPR_2 - NUMBER OF INDEX SUBFILE. IF OMITTED, PRIMARY KEY IS USED.

STR_EXPR - THE KEY BY WHICH TO ACCESS (A STRING). THE KEY SIZE IS DETERMINED BY THE STRING LENGTH. USE THIS FACT FOR PARTIAL-KEY ACCESS.

IF SEQ IS SUPPLIED INSTEAD OF A KEY, THEN THE NEXT SEQUENTIAL RECORD (IN KEY ORDER) IS READ. NOTE THAT IT MAY HAVE THE SAME KEY VALUE AS THAT OF THE RECORD MOST RECENTLY READ (KEY 'DUPLICATES'). SAME KEY WILL RETURN A DATUM ONLY IF THE NEXT KEY MATCHES THE CURRENT ONE, OTHERWISE AN ERROR TRAP IS TAKEN. TO AVOID LOOP PROBLEMS, AND TO SUPPORT AN UNAMBIGUOUS LOCKOUT MECHANISM, AN OPTIONLESS FORM OF READ WILL READ DATA FROM THE CURRENT RECORD, AND DO NO REPOSITIONING. OTHERWISE, A READ STATEMENT WILL PRE-POSITION (AND LOCK) TO THE DESIRED POINT (AS GIVEN BY THE KEY, SAME-KEY, OR SEQUENTIAL OPTIONS) AND THEN READ THE DATA. THIS IS IN ACCORDANCE WITH THE RULE THAT A RECORD WILL BE LOCKED AS SOON AS IT IS POSITIONED TO, AND UNLOCKED BY ANY SUBSEQUENT I/O OPERATION TO THE MIDAS FILE. THEREFORE, NO LOCK OR UNLOCK STATEMENTS ARE PRESENT IN BASIC.

```
*******
* ADD *
*******
```

```
                                        PRIMKEY
ADD #NUM_EXPR_1, STRING_EXPR_1 , [KEY 0_EXP] = STRING_EXPR_2   KEYLIST
                                        KEY

KEYLIST  -->  [ ,KEY NUMERIC_EXPR_2 = STRING_EXPR_3 ]*
```

ADDS A RECORD TO THE MIDAS DATA BASE. PRIMKEY MUST BE SUPPLIED, A SYNONYM FOR WHICH IS KEY. ADD DOES NOT CHANGE THE CURRENT RECORD LOCATION.

```
**********
* UPDATE *
**********
```

```
UPDATE #NUM_EXPR, X$
```

WRITES X$ TO CURRENT RECORD ON FILE UNIT NUM_EXPR. IF THE USER IS STORING KEYS IN THE RECORD, THEN HE MUST BE CAUTIONED AGAINST CHANGING THE KEYS WITH THIS STATEMENT, SINCE, UNLIKE COBOL, BASIC WILL NOT KEEP TRACK OF THE RECORD BREAKDOWN AND HENCE WILL NOT KNOW WHETHER A KEY FIELD HAS BEEN CHANGED. THIS ALSO IMPLIES THAT BASIC CANNOT PERFORM THE UPDATE FUNCTION USING REMOVE FOLLOWED BY ADD.

```
**********
* REMOVE *
**********
```

```
 REMOVE #NUM_EXPR_1 [, KEY [NUM_EXPR_2] = STR_EXPR ]+
```

REMOVES (DELETES) THE GIVEN KEY FROM THE DATA BASE. IF THE KEY IS THE PRIMARY KEY (NUM_EXPR_2 EQUAL TO 0), THEN THE DATA AS WELL AS THE PRIMARY KEY IS REMOVED. TYPICALLY, ONE REMOVES AN ENTIRE LIST OF KEYS, AND THE DATA, IN A SINGLE STATEMENT. THE LANGUAGE DOES PERMIT SPECIFIC-KEY DELETION, HOWEVER.

****
* REWIND *
****

REWIND #NUM_EXPR_1 [, KEY NUM_EXPR_2]

REWINDS THE INDEX SUBFILE NUM_EXPR_2 ON UNIT NUM_EXPR_1. THIS STATEMENT CAN BE THOUGHT OF AS POSITIONING THE 2-DIMENSIONAL FILE POINTER TO THE TOP OF A SPECIFIED COLUMN. NOTE THAT IF THE KEY SPECIFICATION IS OMITTED, KEY 0 IS ASSUMED. THIS ACTION IS EQIVALENT TO POSITIONING THE POINTER IN THE UPPER LEFT CORNER OF THE INDEX-SUBFILE MATRIX.

****
* CLOSE *
****

CLOSE #NUM_EXPR

CLOSES THE MIDAS FILE ON UNIT NUM_EXPR

****
* ERROR HANDLER *
****

ALL MIDAS ERRORS TRAP THROUGH THE NORMAL ON ERROR GOTO MECHANISM.

THE NEW ERROR CODES AND MESSAGES (ERR AND ERR$) FOR MIDAS ARE DEFINED AS FOLLOWS:

56	RECORD NOT FOUND
57	RECORD LOCKED
58	RECORD NOT LOCKED
59	KEY ALREADY EXISTS
60	SEGMENT FILE IN USE
61	INCONSISTENT RECORD LENGTH
62	RECORD FILE FULL
63	KEY FILE FULL
64	IMPROPER FILE TYPE
65	PRIMARY KEY NOT SUPPLIED
66	ILLEGAL OPERATION ON UNIT 0
71	CONCURRENCY ERROR
67	FATAL MIDAS ERROR

THE CATCH-ALL ERROR 67 INDICATES A PROBLEM WITH MIDAS OR THE MIDAS FILE. THESE MAY IN FACT NOT BE COMPLETELY FATAL ERRORS, OR MAY BE CORRECTABLE BY THE USER WITH THE PROPER ANALYSIS. THEREFORE, A NEW SPECIAL FUNCTION, MIDASERR, HAS BEEN DEFINED, WHOSE FUNCTION IS SIMILAR TO ERR, EXECEPT THAT THE TRUE MIDAS PACKAGE ERROR CODE WILL BE CONTAINED THEREIN. USERS CAN THEN REFER TO THE MIDAS MANUAL FOR INTERPRETATION, AND HOPEFULLY FIX THE PROBLEM OR REPORT IT TO PRIME.

FOLLOWING IS AN EXAMPLE OF BASICV MIDAS STATEMENTS IN USE.

```
1 !     ** A VERY SIMPLE 'MIDAS QUERY LANGUAGE' **
2 !
3 !          MIDAS DEMONSTRATION PROGRAM
4 !
5 !     THIS PROGRAM DEMONSTRATES THE USE OF MIDAS IN A SIMPLE
6 !     APPLICATION. CENTRAL IDEAS TO NOTE ARE THE USE OF MULTIPLE
7 !     KEYS, STORAGE OF KEY FIELDS AS DATA, AND THE USE OF BASICV'S
8 !     STRING FUNCTIONS TO AUTOMATICALLY CONTROL STRING LENGTHS,
9 !     TO PERFORM SPACE-PADDING, AND FACILITATE STRING COMPARISONS.
10 !
11 !    THE FUNCTIONS AVAILABLE VIA THIS PROGRAM ARE:
12 !       FIND [ALL] FIELD-NAME FIELD-VALUE
13 !         FINDS ONE OR ALL OF THE RECORDS WITH A THE GIVEN VALUE
14 !         IN THE FIELD SPECIFIED BY FIELD-NAME. FIELD NAMES
15 !         ARE REQUESTED FROM THE USER AT THE START OF THE PROGRAM.
16 !       ADD
17 !         ALLOWS THE USER TO ADD A RECORD TO THE DATA BASE.
18 !         THE USER IS PROMPTED WITH THE FIELD NAMES BEFORE
19 !         BEING REQUIRED TO TYPE IN THE RECORD.
20 !       LIST
21 !         LISTS OUT ALL RECORDS IN THE FILE.
22 !
23 !
100 ON ERROR GOTO 680  ! FIRST SET A SINGLE ERROR HANDLER
110 DIM I$(10)  ! THE INPUT ARRAY
115 !
116 !  FIRST DEFINE ALL NEEDED FUNCTIONS
117 !
120 DEF FNP$(X$,N)  ! PADS X$ WITH SPACES ON RIGHT SUCH THAT TOTAL LENGTH IS N
130   Y$=X$
140   Y$=Y$+' ' UNTIL LEN(Y$)=N
150   FNP$=Y$
160 FNEND
161 !
162 !
```

```
170 DEF FNK(F$)  ! RETURNS A KEY (INDEX SUBFILE) NUMBER GIVEN A FIELD NAME
180   FOR I = 1 TO 10
190     FNK = I-1
200     IF K$(I)=F$ THEN GOTO 220
210   NEXT I
220 FNEND
221 !
222 !
230 DEF FNI   ! INPUT FUNCTION - GETS SPACE-SEPARATED STRINGS FROM TTY AND
231           ! STORES THE SEQUENCE IN I$(1)...I$(N)
240   INPUTLINE '.',X$ ! PROMPT WITH A '.'
250   X$=X$+' '
260   MAT I$=NULL
270   FOR I = 1 STEP 1 UNTIL CVT$$(X$,2)='' ! CVT$$ INSURES NO BLANKS
280     I$(I) = LEFT(X$,INDEX(X$,' ')-1)
290     X$ = RIGHT(X$,INDEX(X$,' ')+1)
300   NEXT I
310 FNEND
311 !
312 !
320 DEFINE FILE #1='DIR',MIDAS,64
330 MATINPUT 'FIELDS:',K$(*) ! FIELD NAMES, IN ORDER FROM KEY 0
331 !
332 !  ** MAIN LOOP **
333 !
340 D=FNI ! INPUT COMMAND STRING
345 !
346 !    FIND ALL
347 !
350 IF I$(1)='FIND' AND I$(2)='ALL' THEN DO
360   POSITION #1, KEY FNK(I$(3))=I$(4)
370   READ #1, X$
380   PRINT CVT$$(X$,16)    ! COMPRESS STRINGS OF BLANKS TO ONE BLANK
390   POSITION #1, SAME KEY  ! WANT TO FIND ALL RECORDS WITH THIS KEY VALUE
400   GOTO 370
410   DOEND
411 !
412 !    FIND
413 !
420 IF I$(1)='FIND' THEN DO
430   READ #1, KEY FNK(I$(2))=I$(3), X$
440   PRINT CVT$$(X$,16)
450   GOTO 340
460   DOEND
461 !
462 !    ADD
463 !
470 IF I$(1)='ADD' THEN DO
480   PRINT K$(I): FOR I = 1 TO 4
490   PRINT ' ';
500   D = FNI
510   I$(1)=FNP$(I$(1),32)  ! WRITE DATA MUST BE PADDED TO CORRECT LENGTH
520   I$(2)=FNP$(I$(2),32)
530   I$(3)=FNP$(I$(3),32)
```

```
540    I$(4)=FNP$(I$(4),32)
550    Z$=I$(1)+I$(2)+I$(3)+I$(4)
560    ADD #1,Z$,KEY0=I$(1),KEY1=I$(2),KEY2=I$(3),KEY3=I$(4)
570    GOTO 340
580    DOEND
581 !
582 !  LIST
583 !
590 IF I$(1)='LIST' THEN DO
600    REWIND #1 ! DEFAULT IS KEY 0
610    READ #1, X$
620    PRINT CVT$$(X$,16)
630    POSITION #1, SEQ
640    GOTO 610
650    DOEND
651 !
652 !
660 PRINT '?'  ! COMMAND ERROR
670 GOTO 340
671 !
680!  A SINGLE ERROR HANDLER  !!!!
681 !
690 IF ERR=56 AND ERL=390 THEN GOTO 340
695 IF ERR=56 AND ERL=630 THEN GOTO 340
700 PRINT ERR$(ERR):'AT LINE':ERL  ! FALL THROUGH TO SYSTEM ERROR
720 END
```

BELOW IS A SAMPLE DIALOGUE USING THE DEMO PROGRAM, WITH
A SIMPLE DATA BASE OF SEQUENCE NUMBERS, NAMES, CITIES, AND STATES.

```
OK, * FIRST MAKE AN EMPTY MIDAS FILE
OK, CO C_CREATK
OK, CREATK
GO
MINIMUM OPTIONS? YES

FILE NAME? DIR
NEW FILE? YES
DIRECT ACCESS? NO

DATA SUBFILE QUESTIONS

KEY TYPE: A
KEY SIZE = : W 16
DATA SIZE = : 64

SECONDARY INDEX

INDEX NO.? 1

DUPLICATE KEYS PERMITTED? YES
KEY TYPE: A
KEY SIZE = : W 16
USER DATA SIZE = :

INDEX NO.? 2

DUPLICATE KEYS PERMITTED? YES
KEY TYPE: A
KEY SIZE = : W 16
USER DATA SIZF = :

INDEX NO.? 3

DUPLICATE KEYS PERMITTED? YES
KEY TYPE: A
KEY SIZE = : W 16
USER DATA SIZE = :
```

INDEX NO.?

```
OK, CO TTY
OK, * NOW RUN THE DEMO
BASICV
GO
```

```
BASICV REV16.0
NEW OR OLD: OLD DEMO
>RUN
DEMO       FRI, SEP 01 1978          17:46:24


FIELDS:NUM,NAME,CITY,STATE
.ADD
NUM NAME CITY STATE  .1 JONES BOSTON MASS
.ADD
NUM NAME CITY STATE  .2 JAMES NEWTON MASS
.ADD
NUM NAME CITY STATE  .3 SMITH NYC NY
.ADD
NUM NAME CITY STATE  .4 AMES ORANGE NJ
.LIST
1 JONES BOSTON MASS
2 JAMES NEWTON MASS
3 SMITH NYC NY
4 AMES ORANGE NJ
.FIND NAME JAMES
2 JAMES NEWTON MASS
.FIND ALL STATE MASS
1 JONES BOSTON MASS
2 JAMES NEWTON MASS
.FIND ALL NAME J     (PARTIAL KEY ACCESS - FINDS ALL NAMES STARTING WITH 'J')
2 JAMES NEWTON MASS
1 JONES BOSTON MASS
.FIND ALL STATE N
4 AMES ORANGE NJ
3 SMITH NYC NY
.CONTROLC
END OF DATA AT LINE 240
>QUIT
```

BASIC/VM ADDENDA

FOLLOWING IS A LIST OF MISCELLANEOUS CHANGES MADE TO BASIC/VM BETWEEN REV 15.0 AND REV 16.0 .

1. COMINP CAN NOW BE USED AS A STATEMENT. AS SUCH, IT ACTS LIKE A STOP FOLLOWED BY A COMINP COMMAND. NOTE THAT THE ARGUMENT TO THE COMINP STATEMENT MUST BE A LEGAL BASIC STRING (AN EXPRESSION, VARIABLE, OR QUOTED CONSTANT) WHEREAS THE COMINP COMMAND TAKES AN UNQUOTED STRING ARGUMENT.

2. THE REPLACE STATEMENT:

   REPLACE #U SEG M BY SEG N

   #U IS A FILE UNIT ON WHICH A SEGMENT DIRECTORY IS OPEN. THE FILE POINTED AT BY SEGMENT M IS DELETED, AND THE POINTER AT SEGMENT N IS MOVED TO SEGMENT M. THE OLD POINTER AT SEGMENT N IS THEN ZEROED.

3. STRING CONSTANTS CAN NOW BE A MAXIMUM OF 160 CHARACTERS LONG.

4. A NEW MAT STATEMENT HAS BEEN ADDED. SIMILAR IN ACTION TO CON OR ZER, IT IS CALLED MAT X$ = NULL[(N1[,N2])] , AND SERVES TO NULL OUT AND OPTIONALLY REDIMENSION A STRING ARRAY.

5. TERMINAL CAN NOW BE ASSIGNED AS AN ARBITRARY FILE UNIT USING THE '(ASR)' FILE NAME, AS IN THE OLD BASIC INTERPRETER.

6. MATINPUT AND INPUTLINE CAN NOW TAKE PROMPT STRINGS, IN THE SAME MANNER AS INPUT.

7. ALTER IS NOW A MODE FROM WHICH ONE ESCAPES TO BASICV BY TYPING 'Q'. THIS WAS DONE BECAUSE VERY OFTEN ONE NEEDS MORE THAN ONE PASS OVER A LINE TO MAKE THE DESIRED CHANGES FELICITOUSLY.

8. THE NUMBER OF AVAILABLE FILE UNITS IS NOW 12 RATHER THAN 8. IN ADDITION, ALL FILE UNITS ARE ALLOCATED FROM THE TOP DOWN (STARTING WITH UNIT 62) AND BASIC/VM CLOSES (UPON EXIT) ONLY THE UNITS THAT IT USES. THIS WILL ALLEVIATE COMMAND-FILE PROBLEMS OF PREVIOUS VERSIONS.

9. LIN(0) NOW OUTPUTS A SINGLE CARRIAGE-RETURN, AS OPPOSED TO SIMPLY DOING NOTHING.

10. THE FOLLOWING TWO SYNTACTICAL FORMS ARE NOW LEGAL:

```
WRITE #N USING X$,...

WRITE USING X$, #N,...
```

BOTH OF THESE ARE SUPPORTED FOR COMPATIBILITY REASONS.

11. PAUSE STATEMENT HAS BEEN ADDED; IT ACTS AS AN EXECUTABLE BREAK. EXAMPLE:

```
>OLD PROG
>LIST
PROG        THU, JUN 01 1978          10:35:25

10 PRINT 1
20 PAUSE
30 PRINT 3
40 END
>RUN
PROG        THU, JUN 01 1978          10:35:27

1
PAUSE AT LINE 20
>CONTINUE
3
>
```

12. SOME CAUTIONS IN USING USER-DEFINED FUNCTIONS:

- GOTO'S OR OTHER SIMILAR TRANSFERS OF CONTROL MUST NOT BE MADE FROM INSIDE A FUNCTION DEFINITION OUT NOR FROM OUTSIDE IN. IF THIS IS DONE, THE SYSTEM STACK WILL INCORRECTLY TRACK, AND RESULTS WILL BE UNPREDICTABLE.

- A CALL TO A USER-FUNCTION THAT PERFORMS I/O MUST NOT BE PLACED AS PART OF AN I/O LIST. UNPREDICTABLE BEHAVIOR WILL RESULT ON THE FILES OR I/O DEVICES ADDRESSED BY THESE STATEMENTS. THIS DOES NOT MEAN THAT FUNCTIONS SHOULD NOT BE USED TO PERFORM I/O, INDEED, THERE ARE ADVANTAGES TO THIS. ONE SHOULD SIMPLY FOLLOW THE RULE STATED ABOVE, AND USE A TEMPORARY VALUE WHERE ONE MIGHT HAVE USED, SAY, A PRINT STATEMENT.

EXAMPLE:

```
10 !  GOODPROG
20 !
100 DEF FNR$(N)  ! READS A STRING FROM FILE UNIT N,
105              !                RETURNS NULL IF EOF
110  ON END #N GOTO 150
120  READ LINE #N, X$
130  FNR$=X$
140  GOTO 160
150  FNR$=''
160 FNEND
170 !
180 ! READ FILE 'XXX', PRINT ON TERMINAL.
190 !   ON EOF, CLOSE FILE AND END
195 !
200 N = 1
210 DEFINE FILE #N = 'XXX'
220 FOR I = 1 UNTIL 1=2
230  D$=FNR$(N)
240  IF D$ = '' THEN DO
250        CLOSE #N
260        END
270  DOEND
280  PRINT D$
290 NEXT I
```

```
10 ! BADPROG
20 !
100 DEF FNR$(N)  ! READS A STRING FROM FILE UNIT N,
105              !                RETURNS NULL IF EOF
110  ON END #N GOTO 150
120  READ LINE #N, X$
130  FNR$=X$
140  GOTO 160
150  FNR$=''
160 FNEND
```

```
170 !
180 ! READ FILE 'XXX', PRINT ON TERMINAL.
190 !   EOF ACTION ACTION NOT DEFINED
195 !
200 N = 1
210 DEFINE FILE #N = 'XXX'
220 FOR I = 1 UNTIL 1=2
230  PRINT FNR$(N)
290 NEXT I
```

LINE 230 OF GOODPROG SHOWS THAT D$=FNR$(N) WILL NOT CONFUSE THE I/O HANDLER. LINE 230 OF BADPROG, HOWEVER, SHOWS A POSSIBLE WAY OF MAKING THE I/O AMBIGUOUS. IN THIS CASE, THE SYSTEM SETS UP TO PRINT THE VALUE OF THE FUNCTION, THEN CALLS THE FUNCTION TO GET THIS VALUE. DURING THE CALL, THOUGH, I/O IS PERFORMED, THEREBY DESTROYING THE PREVIOUS SETUP FOR PRINT. SUBSEQUENT RESULTS WILL THEN BE UNPREDICTABLE.

TARS

FOLLOWING IS A LIST OF TARS, PROCESSED AND UNPROCESSED, APPLICABLE TO BASIC/VM.

25257 - BASICV NON-EXISTENT-FILE CAUSES ACCESS VIOLATION. FIXED

25254 - WRITING >120 CHARS TO A FILE TRUNCATES LINE. SOLUTION: INCREASE RECORD SIZE TO HANDLE LINES OF DESIRED LENGTH.

25258 - GOSUB STACK NOT RESET FROM PROGRAM TO PROGRAM. FIXED

25252 & 25253 - ENTER - SHORT FORM WAS ILLEGAL, ALSO SYNTAX NOT CHECKED FOR CONSTANTS IN VARIABLE SLOTS. FIXED

25261 - SQR OF FRACTIONAL POWERS OF TWO PRODUCED INCORRECT RESULTS. FIXED

24732 - TRACE ON CLOBBERS LINE NUMBER UPON ERROR. FIXED

25256 - NUMERIC CONSTANTS OF FORM XE00 WERE NOT RECOGNIZED. FIXED

25260 - NUMERIC OVERFLOW,UNDERFLOW - INAPPROPRIATE ERROR MESSAGE. FIXED

20208 - BASICV FILE-NAME RETURNS TO BASICV INSTEAD OF PRIMOS. FIXED

24731 - RANDOM CHARACTERS WERE PRINTED IN EXPONENT UPON OVERFLOW. FIXED (NOW FILLS FIELD WITH STARS, AS DOES PRINT USING)

24730 - 0**-X = 1, X>=0. FIXED (GIVES ERROR MESSAGE)

20206 - DATA CHAINS NOT RESET ON EXECUTE. FIXED

24764 - COMPILER-TEMPORARY FILES (T$NNNN) ARE LOST UPON ATTACHING TO ANOTHER UFD FROM WHICH THEY WERE CREATED. IN PROGRESS

25255 - RESEQUENCE DOES NOT REPLACE DUPLICATE LINES. SOLUTION: DELETE LINES BEFORE RESEQUENCING

SUBJECT: RUNOFF FOR RELEASE 16.0.

TWO NEW COMMANDS ARE AVAILABLE: .EODD (EJECT ODD) AND .EEVEN (EJECT EVEN), MINIMAL ABBREVIATIONS ARE .EO AND .EE. THESE COMMANDS CAUSE AN EJECT TO A NEW PAGE. A SUBSEQUENT EJECT IS THEN CAUSED IF THE NUMBER OF THE NEW PAGE IS EVEN (FOR .EO) OR ODD (FOR .EE). THESE COMMANDS FUNCTION INDEPENDENTLY OF WHETHER THE PAGE NUMBER HAS BEEN SET WITH THE .PAGE COMMAND OR IS BEING DISPLAYED.