Rev 17

Sherry M

APPLICATIONS LIBRARY

DATE: MAY 19, 1979

TO:

FROM:

SUBJECT: APPLICATIONS LIBRARY

REFERENCE: NONE

ABSTRACT

ATTACHED IS A DETAILED DESCRIPTION OF THE REV.17.0 APPLICATIONS LIBARAY (APPLIB) AND ITS V-MODE VERSION (VAPPLB) LOCATED IN UFD = LIB. FOR <FURTHER INFORMATION CONTACT ERIC STANMYER (NEWTON - X419). ALL CHANGES <TO THE PREVIOUS REVISION (PE-T-315, REV. 5, FEBUARY 22, 1979) OF THIS <DOCUMENT ARE MARKED WITH REVISION BARS ("<"). ALL SUBMISSIONS, COMMENTS, CRITICISMS AND SUGGESTIONS ARE APPRECIATED AND WILL BE GIVEN CAREFULL CONSIDERATION.

THE REV. 17.0 APPLICATIONS LIBRARY CONTAINS 9 NEW ROUTINES PLUS A NUMBER OF ENHANCEMENTS TO SEVERAL PREVIOUSLY EXISTING ROUTINES.

THE NEW ROUTINES ARE:

FSUB$A - FILL A SUBSTRING WITH A GIVEN CHARACTER, SIMILAR IN FUNCTION TO FILL$A.

SSTR$A - SHIFT A STRING LEFT OR RIGHT A GIVEN NUMBER OF CHARACTER POSITIONS.

SSUB$A - SHIFT A SUBSTRING LEFT OR RIGHT A GIVEN NUMBER OF CHARACTER POSITIONS.

RSTR$A - ROTATE A STRING LEFT OR RIGHT A GIVEN NUMBER OF CHARACTER POSITIONS.

RSUB$A - ROTATE A SUBSTRING LEFT OR RIGHT A GIVEN NUMBER OF CHARACTER POSITIONS.

CASE$A - CONVERT A STRING FROM LOWER CASE TO UPPER CASE OR VICE VERSA.

FTIM$A - CONVERT THE TIMMOD FIELD AS RETURNED BY RDEN$$ TO 'HH:MM:SS' AND DECIMAL HOURS.

FDAT$A - CONVERT THE DATMOD FIELD AS RETURNED BY RDEN$$ TO 'MM/DD/YY' AND 'DAY, MON DD YEAR'.

FEDT$A - SAME AS FDAT$A EXCEPT DATE IS RETURNED AS 'MM.DD.YY'.

ROUTINES WITH FUNCTIONAL IMPROVEMENTS AND ENHANCEMENTS ARE:

CNVB$A, CNVA$A, RNUM$A, TYPE$A - ACCEPT A NUMKEY OF A$BIN FOR SPECIFING BINARY NUMBERS.

JSTR$A - ACCEPTS A KEY OF A$CNTR FOR CENTERING A STRING WITHIN A FIELD.

CMDL$A - THREE NEW KEYS HAVE BEEN ADDED:
A$BIN AND A$NBIN CAN NOW BE USED IN SPECIFING THE OPTION TYPE A KEYWORD MAY HAVE (INDICATING, RESPECTIVELY, THAT AN OPTION IS A BINARY NUMBER, AN OPTION MAY BE A BINARY NUMBER OR A NAME),
A$RCMD USED AS KEY ALLOWS THE USE OF A KEYWORD WITHOUT A PRECEDING MINUS SIGN AS THE FIRST TOKEN ON A LINE (THIS KEY MAY NOT BE USED ON THE COMMAND LINE ITSELF).

OPEN$A, OPNP$A, OPNV$A, OPVP$A, TEMP$A - WILL NOW ACCEPT AN ADDITIONAL KEY, A$GETU, WHICH WILL CAUSE THE FILE SYSTEM TO CHOOSE AN UNUSED UNIT NUMBER UPON WHICH TO OPEN A FILE. THE UNIT NUMBER IS RETURNED IN THE 'UNIT' ARGUMENT OF EACH ROUTINE. THIS KEY IS ADDITIVE AND CORRESPONDS TO THE 'UNTKEY' ARGUMENT IN THE CALLING SEQUENCES.

FOR A MORE DETAILED DESCRIPTION OF THE NEW ROUTINES AND ENHANCEMENTS PLEASE SEE THE APPROPRIATE ENTRY FOR THAT ROUTINE.

TABLE OF CONTENTS

1 INTRODUCTION

APPLIB IS A USER ORIENTED LIBRARY WHICH IS INTENDED TO PROVIDE USERS WITH AN EASY TO USE SET OF SERVICE 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 AND ERROR CHECKING 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 INTERFACE TO THE TERMINAL USER.

2 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.

2.1 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 USE OF A "CODE" TYPE ARGUMENT WHICH MUST BE DECODED BY THE USER IS AVOIDED WHEREVER POSSIBLE. 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.

2.2 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.

2.3 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.

2.4 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 "STRING,LENGTH", THE STRING IS OFTEN UPDATED BY AN APPLIB ROUTINE, BUT THE LENGTH ARGUMENT IS NEVER MODIFIED. 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.

2.5 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_APPLIB" AND "C_VAPPLB" LOCATED IN UFD = APPLIB>SOURCE.

2.6 LIBRARY TESTING

BEFORE A NEW VERSION OF APPLIB IS RELEASED FOR A MASTER/UPDATE DISK BUILD IT SHOULD BE INSTALLED ON THE IN-HOUSE SYSTEMS FOR TESTING. THE NECESSARY COMMAND FILES TO BUILD BOTH VERSIONS (R-MODE AND V-MODE) ARE LOCATED IN UFD=APPLIB>TEST>BUILD ALONG WITH VARIOUS OTHER COMMAND FILES TO PRODUCE COMPILER LISTINGS, ETC. THESE COMMAND FILES FUNCTION MUCH THE SAME AS THOSE USED TO PERFORM AN ACTUAL BUILD, EXCEPT THAT THE RESULTING BINARY FILES ARE NOT COPIED TO UFD=LIB. ONCE THE BINARY FILES HAVE BEEN BUILT THEN THEY MAY BE SUBMITTED TO SDI FOR INSTALLATION ON THE IN-HOUSE SYSTEMS.

THERE IS ALSO A UFD AVAILABLE IN WHICH TEST PROGRAMS, ETC. MAY BE KEPT FOR FUTURE USE (UFD=APPLIB>TEST>PROGRAMS).

2.7 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 (UFD=APPLIB>PENDING) OR AN "IDEA" FILE (UFD=APPLIB>IDEAS), 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 <u>BOTH</u> APPLIB AND VAPPLB.

2. THE ROUTINE SOULD <u>NOT</u> 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 THE 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.

3 THE ROUTINES

THE FOLLOWING SECTIONS DESCRIBE EACH ROUTINE IN THE APPLICATIONS LIBRARY IN DETAIL, INCLUDING THE VARIOUS CALLING SEQUENCES, DESCRIPTIONS OF ALL PASSED ARGUMENTS, AND THE FUNCTION THE ROUTINE WAS DESIGNED TO PERFORM.

3.1 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.

THESE ROUTINES ARE:

CLOS$A - CLOSE FILE

DELE$A - DELETE FILE

EXST$A - CHECK FOR EXISTENCE OF FILE

GEND$A - POSITIONS TO END OF FILE

OPEN$A - OPEN FILE WITH GIVEN NAME

OPNP$A - OPEN FILE WITH NAME READ FROM USER TERMINAL

OPNV$A - OPEN FILE WITH GIVEN NAME WITH VERIFICATION AND DELAY

OPVP$A - OPEN FILE WITH NAME READ FROM USER TERMINAL WITH VERIFICATION AND DELAY

POSN$A - POSITION FILE TO GIVEN POSITION

RPOS$A - RETURN ABSOLUTE POSITION WITHIN FILE

RWND$A - REWIND FILE

TEMP$A - OPEN SCRATCH FILE

TRNC$A - TRUNCATE FILE

TSCN$A - FILE SYSTEM TREE SCAN

UNIT$A - DETERMINE STATUS OF FILE (OPEN OR CLOSED)

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 OBTAIN 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 REPRESENTED BY SIMPLE CALLING SEQUENCES AND ONLY THE COMPLEX OPERATIONS NEED TO SPECIFY COMPLEX ARGUMENT LISTS.

ALL FILE OPEN 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.

ALL FILE OPEN ROUTINES HAVE THE ABILITY TO CHOOSE THE FILE UNIT UPON WHICH A FILE WILL OPENED BY USE OF THE A$GETU KEY. IF THIS KEY IS USED THE FILE UNIT SELECTED WILL BE RETURNED IN THE ARGUMENT 'UNIT'. IF THIS KEY IS NOT USED THEN THE CALLER MUST PROVIDE THE ROUTINE WITH A USABLE FILE UNIT NUMBER.

VERIFICATION CONSISTS OF THE FOLLOWING OPTIONS:

1. VERIFY THAT THE FILE IS NEW; OTHERWISE IF THE FILE ALREADY EXISTS VERIFY THAT IT IS O.K. TO MODIFY IT.

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 ALREADY EXISTS; THAT IS, DO NOT ALLOW CREATION OF A NEW FILE. NOTE 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.

CLOS$A

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

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

ARGUMENTS:

UNIT = DOS FILE UNIT, INTEGER*2

FUNCTION:

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

APPLIB CALLS:

NONE

DELE$A

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

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

ARGUMENTS:

NAME = ARRAY CONTAINING FILENAME (MAY BE A TREE NAME) PACKED TWO CHARACTERS PER WORD, DATA TYPE DOES NOT MATTER
NAMLEN = LENGTH OF NAME IN CHARACTERS, INTEGER*2

FUNCTION:

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

APPLIB CALLS:

TREE$A, UNIT$A, NLEN$A

EXST$A

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

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

ARGUMENTS:

```
NAME   = ARRAY CONTAINING FILENAME (MAY BE A TREE NAME) PACKED TWO
         CHARACTERS PER WORD, DATA TYPE DOES NOT MATTER
NAMLEN = LENGTH OF NAME IN CHARACTERS, INTEGER*2
```

FUNCTION:

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

APPLIB CALLS:

TREE$A, UNIT$A, NLEN$A

GEND$A

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

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

ARGUMENTS:

UNIT = DOS FILE UNIT, INTEGER*2

FUNCTION:

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 IT IS FALSE.

APPLIB CALLS:

NONE

OPEN$A

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

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

ARGUMENTS:

```
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
UNTKEY = A$GETU, CHOOSE A FILE UNIT NUMBER, UNIT NUMBER RETURNED IN
         'UNIT'.  OMISSION OF THIS KEY REQUIRES THAT THE ROUTINE BE
         PROVIDED WITH A UNIT NUMBER.
NAME   = ARRAY CONTAINING FILENAME (MAY BE A TREE NAME) PACKED  TWO
         TWO CHARACTERS PER WORD, DATA TYPE DOES NOT MATTER
NAMLEN = LENGTH OF NAME IN CHARACTERS, INTEGER*2
UNIT   = DOS FILE UNIT, INTEGER*2
```

FUNCTION:

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.

APPLIB CALLS:

TREE$A, UNIT$A, NLEN$A

OPNP$A

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

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

ARGUMENTS:

```
MSG    = ARRAY CONTAINING PROMPT FOR NAME MESSAGE PACKED TWO
         CHARACTERS PER WORD, DATA TYPE DOES NOT MATTER
MSGLEN = LENGTH OF MSG IN CHARACTERS, INTEGER*2
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
UNTKEY = A$GETU, CHOOSE A FILE UNIT NUMBER, UNIT NUMBER RETURNED IN
         'UNIT'. OMISSION OF THIS KEY REQUIRES THAT THE ROUTINE BE
         PROVIDED WITH A UNIT NUMBER.
NAME   = ARRAY CONTAINING FILENAME (MAY BE A TREE NAME) PACKED TWO
         CHARACTERS PER WORD, DATA TYPE DOES NOT MATTER
NAMLEN = LENGTH OF NAME IN CHARACTERS, INTEGER*2
UNIT   = DOS FILE UNIT, INTEGER*2
```

FUNCTION:

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.

APPLIB CALLS:

RNAM$A, NLEN$A, TREE$A, UNIT$A

OPNV$A

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

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

ARGUMENTS:

```
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
UNTKEY = A$GETU, CHOOSE A FILE UNIT NUMBER, UNIT NUMBER RETURNED IN
         'UNIT'.  OMISSION OF THIS KEY REQUIRES THAT THE ROUTINE BE
         PROVIDED WITH A UNIT NUMBER.
NAME   = ARRAY CONTAINING FILENAME (MAY BE A TREE NAME) PACKED  TWO
         CHARACTERS PER WORD, DATA TYPE DOES NOT MATTER
NAMLEN = LENGTH OF NAME IN CHARACTERS, INTEGER*2
UNIT   = DOS FILE UNIT, INTEGER*2
VERKEY = A$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, INTEGER*2
RETRYS = NUMBER OF TIMES TO RETRY IF FILE IN USE, INTEGER*2
```

FUNCTION:

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.

APPLIB CALLS:

RNAM$A, TIME$A, NLEN$A, EXST$A, UNIT$A, TREE$A, GEND$A

COMMENTS:

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 VERIFICATION IS NOT REQUESTED (VERKEY = A$NVER) THEN OPNV$A IS
< IDENTICAL IN FUNCTION TO OPEN$A.

< IF NAMLEN IS LESS THAN OR EQUAL TO ZERO THE ROUTINE WILL RETURN A
< FUNCTION VALUE OF FALSE (IE. NULL FILENAMES OR TREENAMES ARE NOT
< ALLOWED).

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

ARGUMENTS:

```
MSG    = ARRAY CONTAINING PROMPT MESSAGE PACKED TWO CHARACTERS PER
         WORD, DATA TYPE DOES NOT MATTER
MSGLEN = LENGTH OF MSG IN CHARACTERS, INTEGER*2
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
UNTKEY = A$GETU, CHOOSE A FILE UNIT NUMBER, UNIT NUMBER RETURNED IN
         'UNIT'.  OMISSION OF THIS KEY REQUIRES THAT THE ROUTINE BE
         PROVIDED WITH A UNIT NUMBER.
NAME   = ARRAY CONTAINING FILENAME (MAY BE A TREE NAME) PACKED  TWO
         CHARACTERS PER WORD, DATA TYPE DOES NOT MATTER
NAMLEN = LENGTH OF NAME IN CHARACTERS, INTEGER*2
UNIT   = DOS FILE UNIT, INTEGER*2
VERKEY = A$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, INTEGER*2
RETRYS = NUMBER OF TIMES TO RETRY IF FILE IN USE, INTEGER*2
```

FUNCTION:

THIS ROUTINE GETS A FILENAME 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 NAMLEN IS LESS THAN OR EQUAL TO ZERO THE ROUTINE WILL RETURN A
< FUNCTION VALUE OF FALSE (IE. NULL FILENAMES OR TREENAMES ARE NOT
< ALLOWED).

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.

APPLIB CALLS:

RNAM$A, TIME$A, NLEN$A, FXST$A, UNIT$A, TREE$A, GEND$A

COMMENTS:

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 PROMPT MESSAGE WILL BE REPEATED.

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.

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)
```

ARGUMENTS:

```
POSKEY = A$ABS, ABSOLUTE POSITION (.NE.  A$REL)
         A$REL, RELATIVE POSITION
UNIT   = DOS FILE UNIT, INTEGER*2
POS    = POSITION (RELATIVE OR ABSOLUTE), INTEGER*4
```

FUNCTION:

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.

APPLIB CALLS:

NONE

RPOS$A

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

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

ARGUMENTS:

```
UNIT    = DOS FILE UNIT, INTEGER*2
POS     = RETURNED ABSOLUTE POSITION, INTEGER*4
```

FUNCTION:

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.

APPLIB CALLS:

NONE

RWND$A

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

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

ARGUMENTS:

UNIT = DOS FILE UNIT, INTEGER*2

FUNCTION:

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..

APPLIB CALLS:

NONE

TEMP$A

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

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

ARGUMENTS:

```
TYPKEY = A$SAMF, SAM FILE (.NE.  A$DAMF)
         A$DAMF, DAM FILE
UNTKEY = A$GETU, CHOOSE A FILE UNIT NUMBER, UNIT NUMBER RETURNED IN
         'UNIT'.  OMISSION OF THIS KEY REQUIRES THAT THE ROUTINE BE
         PROVIDED WITH A UNIT NUMBER.
NAME   = RETURNED NAME (6 CHARACTERS), PACKED  TWO  CHARACTERS  PER
         WORD, DATA TYPE DOES NOT MATTER
NAMLEN = LENGTH OF NAME BUFFER IN CHARACTERS (.GE.  6), INTEGER*2
UNIT   = DOS FILE UNIT, INTEGER*2
```

FUNCTION:

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 IS FALSE.

APPLIB CALLS:

FILL$A

TRNC$A

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

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

ARGUMENTS:

UNIT = DOS FILE UNIT, INTEGER*2

FUNCTION:

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

APPLIB CALLS:

NONE

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)
```

ARGUMENTS:

```
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,
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 ARGUMENTS ARE INTEGER*2.

FUNCTION:

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.)

APPLIB CALLS:

NONE

COMMENTS:

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 PROGRAM:

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
```

UNIT$A

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

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

ARGUMENTS:

UNIT = DOS FILE UNIT, INTEGER*2

FUNCTION:

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

APPLIB CALLS:

NONE

3.2 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 REQUIRE THAT THE PHYSICAL LENGTH OF A STRING BE PASSED AS AN ARGUMENT, THE LENGTH BEING SPECIFIED IN CHARACTERS. THE PHYSICAL LENGTH OF A STRING IS THE ACTUAL INTERNAL STORAGE ALLOCATED FOR THAT STRING, IN BYTES (OR CHARACTERS), INCLUDING ANY TRAILING BLANKS. THE OPERATIONAL LENGTH OF A STRING DOES NOT INCLUDE TRAILING BLANKS. SINCE THE LENGTH OF A STRING IS SPECIFIED AS AN INTEGER*2 VARIABLE THE MAXIMUM STRING LENGTH IS 32,767 CHARACTERS.

THE MAJORITY OF ROUTINES THAT OPERATE ON ENTIRE STRINGS FIRST TRUNCATE THEM TO THEIR OPERATIONAL LENGTH, WHEREAS THOSE ROUTINES THAT OPERATE ON SUBSTRINGS TREAT ANY TRAILING BLANKS AS PART OF THE SUBSTRING.

THESE ROUTINES ARE:

CSTR$A - COMPARE TWO STRINGS FOR EQUALITY

CSUB$A - COMPARE TWO SUBSTRINGS FOR EQUALITY

FILL$A - FILL A STRING WITH A GIVEN CHARACTER

FSUB$A - FILL SUBSTRING WITH A GIVEN CHARACTER.

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

JSTR$A - LEFT JUSTIFY, RIGHT JUSTIFY, OR CENTER A STRING WITHIN A FIELD.

LSTR$A - LOCATE ONE STRING WITHIN ANOTHER

LSUB$A - LOCATE ONE SUBSTRING WITHIN ANOTHER

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

MSTR$A - MOVE ONE STRING TO ANOTHER

MSUB$A - MOVE ONE SUBSTRING TO ANOTHER

NLEN$A - DETERMINE THE OPERATIONAL LENGTH OF A STRING.

RSTR$A - ROTATE STRING LEFT OR RIGHT.

RSUB$A - ROTATE SUBSTRING LEFT OR RIGHT.

SSTR$A - SHIFT STRING LEFT OR RIGHT.

SSUB$A - SHIFT SUBSTRING LEFT OR RIGHT.

TREE$A - TEST FOR TREE NAME

TYPE$A - DETERMINE STRING TYPE

ALL STRING LENGTH SPECIFICATIONS AND SUBSTRING DELIMITING CHARACTER POSITIONS ARE CHECKED FOR VALIDITY AND MUST CONFORM TO THE FOLLOWING RULES:

1. PHYSICAL STRING LENGTH SPECIFICATIONS MUST BE GREATER THAN OR EQUAL TO ZERO, A VALUE OF ZERO INDICATING A NULL OR EMPTY STRING.

2. SUBSTRING DELIMITING CHARACTER POSITIONS MUST BE GREATER THAN OR EQUAL TO ZERO AND LESS THAN OR EQUAL TO THE PHYSICAL STRING LENGTH. ADDITIONALLY THE BEGINNING CHARACTER POSITION MUST BE LESS THAN OR EQUAL TO THE ENDING CHARACTER POSITION. A NULL SUBSTRING IS SPECIFIED BY A VALUE OF ZERO FOR EITHER THE STARTING OR ENDING CHARACTER POSITIONS.

IF ANY OF THE ABOVE RULES ARE VIOLATED AN ERROR MESSAGE WILL BE DISPLAYED AND THE FUNCTION WILL BE FALSE (LOGICAL FUNCTIONS ONLY).

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, DATA TYPE DOES NOT MATTER.
ALEN	= LENGTH OF A IN CHARACTERS, INTEGER*2.
B	= STRING TO BE COMPARED AGAINST, PACKED TWO CHARACTERS PER WORD, DATA TYPE DOES NOT MATTER.
BLEN	= LENGTH OF B IN CHARACTERS, INTEGER*2.

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. TRAILING BLANKS ARE IGNORED.

APPLIB CALLS:

CSUBA, LENA

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, DATA TYPE DOES NOT MATTER.
ALEN	=	LENGTH OF A IN CHARACTERS, INTEGER*2.
AFC	=	FIRST DELIMITING CHARACTER POSITION OF SUBSTRING IN A, INTEGER*2.
ALC	=	LAST DELIMITING CHARACTER POSITION OF SUBSTRING IN A, INTEGER*2.
B	=	ARRAY CONTAINING SUBSTRING TO BE COMPARED AGAINST, PACKED TWO CHARACTERS PER WORD, DATA TYPE DOES NOT MATTER.
BLEN	=	LENGTH OF B IN CHARACTERS, INTEGER*2.
BFC	=	FIRST DELIMITING CHARACTER POSITION OF SUBSTRING IN B, INTEGER*2.
BLC	=	LAST DELIMITING CHARACTER POSITION OF SUBSTRING IN B, INTEGER*2.

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.

APPLIB CALLS:

NONE

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)
```

ARGUMENTS:

```
NAME   = NAME BUFFER TO FILL PACKED TWO CHARACTERS PER WORD, DATA
         TYPE DOES NOT MATTER.
NAMLEN = LENGTH OF NAME IN CHARACTERS, INTEGER*2.
CHAR   = FILL CHARACTER IN A1 FORMAT, DATA TYPE DOES NOT MATTER.
```

FUNCTION:

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

APPLIB CALLS:

NONE

FSUB$A

FSUB$A IS AN LOGICAL FUNCTION USED TO FILL A CHARACTER SUBSTRING WITH A GIVEN CHARACTER, IT HAS THE FOLLOWING CALLING SEQUENCE:

```
LOG = FSUB$A(STRING,LENGTH,FCHAR,LCHAR,FILCHR)
CALL  FSUB$A(STRING,LENGTH,FCHAR,LCHAR,FILCHR)
```

ARGUMENTS:

STRING = STRING CONTAINING SUBSTRING TO BE FILLED, PACKED TWO CHARACTERS PER WORD, DATA TYPE DOES NOT MATTER.
LENGTH = LENGTH OF STRING IN CHARACTERS, INTEGER*2.
FCHAR = FIRST DELIMITING CHARACTER POSITION OF SUBSTRING, INTEGER*2.
LCHAR = LAST DELIMITING CHARACTER POSITION OF SUBSTRING, INTEGER*2.
FILCHR = FILL CHARACTER IN A1 FORMAT, DATA TYPE DOES NOT MATTER.

FUNCTION:

FSUB$A WILL FILL THE SUBSTRING DELIMITED BY FCHAR AND LCHAR WITH THE GIVEN FILL CHARACTER. THE STRING PARAMETERS PASSED TO THE ROUTINE ARE CHECKED FOR VALIDITY AND IF AN ERROR OCCURS A MESSAGE IS PRINTED AND THE FUNCTION WILL BE FALSE. IF ALL PARAMETERS ARE VALID THE FUNCTION WILL BE TRUE.

APPLIB CALLS:

NONE

GCHR$A

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

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

ARGUMENTS:

FARRAY = SOURCE ("FROM") PACKED ARRAY, DATA TYPE DOES NOT MATTER.
FCHAR = CHARACTER POSITION IN FARRAY, INTEGER*2.

FUNCTION:

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.

APPLIB CALLS:

NONE

JSTR$A

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

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

ARGUMENTS:

```
KEY    = DIRECTION OF JUSTIFICATION, POSSIBLE VALUES ARE:
         A$RGHT - RIGHT JUSTIFY,
         A$LEFT - LEFT JUSTIFY,
         A$CNTR - CENTER.
STRING = STRING TO BE JUSTIFIED, PACKED TWO CHARACTERS PER WORD,
         DATA TYPE DOES NOT MATTER.
LENGTH = LENGTH OF STRING IN CHARACTERS, INTEGER*2.
```

FUNCTION:

JSTR$A WILL LEFT JUSTIFY, RIGHT JUSTIFY, OR CENTER A STRING WITHIN ITSELF. THE FUNCTION WILL BE FALSE IF LENGTH IS LESS THAN ZERO, OTHERWISE IT WILL BE TRUE.

APPLIB CALLS:

NLEN$A, FILL$A, MSUB$A, GCHR$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, DATA TYPE DOES NOT MATTER.
ALEN	=	LENGTH OF A IN CHARACTERS, INTEGER*2.
B	=	STRING TO BE SEARCHED PACKED TWO CHARACTERS PER WORD, DATA TYPE DOES NOT MATTER.
BLEN	=	LENGTH OF B IN CHARACTERS, INTEGER*2.
FCP	=	FIRST CHARACTER POSITION IN B OF SUBSTRING THAT MATCHES STRING A, INTEGER*2.
LCP	=	LAST CHARACTER POSITION IN B OF SUBSTRING THAT MATCHES STRING A, INTEGER*2.

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. TRAILING BLANKS ARE IGNORED.

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, DATA TYPE DOES NOT MATTER.
ALEN	=	LENGTH OF A IN CHARACTERS, INTEGER*2.
AFC	=	FIRST DELIMITING CHARACTER POSITION OF SUBSTRING IN A, INTEGER*2.
ALC	=	LAST DELIMITING CHARACTER POSITION OF SUBSTRING IN A, INTEGER*2.
B	=	ARRAY CONTAINING SUBSTRING TO BE SEARCHED, PACKED TWO CHARACTERS PER WORD, DATA TYPE DOES NOT MATTER.
BLEN	=	LENGTH OF B IN CHARACTERS, INTEGER*2.
BFC	=	FIRST DELIMITING CHARACTER POSITION OF SUBSTRING IN B, INTEGER*2.
BLC	=	LAST CHARACTER POSITION OF SUBSTRING IN B, INTEGER*2.
FCP	=	FIRST CHARACTER POSITION IN B OF SUBSTRING THAT MATCHES SUBSTRING IN A, INTEGER*2.
LCP	=	LAST CHARACTER POSITION IN B OF SUBSTRING THAT MATCHES SUBSTRING IN A, INTEGER*2.

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.

APPLIB CALLS:

NONE

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)
```

ARGUMENTS:

```
TARRAY = RECEIVING ("TO") PACKED ARRAY, DATA TYPE DOES NOT MATTER.
TCHAR  = CHARACTER POSITION IN TARRAY, INTEGER*2.
FARRAY = SOURCE ("FROM") PACKED ARRAY, DATA TYPE DOES NOT MATTER.
FCHAR  = CHARACTER POSITION IN FARRAY, INTEGER*2.
```

FUNCTION:

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.

APPLIB CALLS:

NONE

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, DATA TYPE DOES NOT MATTER.
ALEN = LENGTH OF A IN CHARACTERS, INTEGER*2.
B = DESTINATION STRING, PACKED TWO CHARACTERS PER WORD, DATA TYPE DOES NOT MATTER.
BLEN = LENGTH OF B IN CHARACTERS, INTEGER*2.

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.

APPLIB CALLS:

MSUB$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, DATA TYPE DOES NOT MATTER.
ALEN	=	LENGTH OF A IN CHARACTERS, INTEGER*2.
AFC	=	FIRST DELIMITING CHARACTER POSITION OF SUBSTRING, INTEGER*2.
ALC	=	LAST DELIMITING CHARACTER POSITION OF SUBSTRING, INTEGER*2.
B	=	ARRAY CONTAINING DESTINATION SUBSTRING, PACKED TWO CHARACTERS PER WORD, DATA TYPE DOES NOT MATTER.
BLEN	=	LENGTH OF B IN CHARACTERS, INTEGER*2.
BFC	=	FIRST DELIMITING CHARACTER POSITION OF SUBSTRING, INTEGER*2.
BLC	=	LAST DELIMITING CHARACTER POSITION OF SUBSTRING, INTEGER*2.

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).

APPLIB CALLS:

MCHR$A

NLEN$A

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

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

ARGUMENTS:

NAME = NAME BUFFER TO TEST PACKED TWO CHARACTERS PER WORD, DATA TYPE DOES NOT MATTER.
NAMLEN = LENGTH OF NAME IN CHARACTERS, INTEGER*2.

FUNCTION:

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

APPLIB CALLS:

NONE

RSTR$A

RSTR$A IS A LOGICAL FUNCTION USED TO ROTATE A CHARACTER STRING, IT HAS THE FOLLOWING CALLING SEQUENCE:

```
LOG = RSTR$A(STRING,LENGTH,COUNT)
CALL  RSTR$A(STRING,LENGTH,COUNT)
```

ARGUMENTS:

STRING = STRING TO BE ROTATED, PACKED TWO CHARACTERS PER WORD, DATA TYPE DOES NOT MATTER.
LENGTH = LENGTH OF STRING IN CHARACTERS, INTEGER*2.
COUNT = NUMBER OF POSITIONS TO ROTATE STRING, NEGATIVE COUNT CAUSES LEFT ROTATE, POSITIVE COUNT CAUSES RIGHT ROTATE, INTEGER*2.

FUNCTION:

RSTR$A WILL ROTATE A CHARACTER STRING LEFT OR RIGHT THE NUMBER OF POSITIONS SPECIFIED BY COUNT. THE STRING IS TRUNCATED TO ITS OPERATIONAL LENGTH BEFORE THE ROTATE IS PERFORMED THEREFORE TRAILING BLANKS ARE NOT INCLUDED. IF LENGTH IS LESS THAN ZERO AN ERROR MESSAGE WILL BE PRINTED AND THE FUNCTION WILL BE FALSE, OTHERWISE IT WILL BE TRUE.

APPLIB CALLS:

MCHR$A, NLEN$A

COMMENTS:

THIS ROUTINE USES AN ALGORITHM THAT MINIMIZES BOTH TEMPORARY STORAGE AND EXECUTION TIME. TEMPORARY STORAGE USED CONSISTS OF ONE WORD AND THE NUMBER OF ITERATIONS NECESSARY TO ROTATE A STRING IS EQUAL TO THE LENGTH (IN CHARACTERS) OF THE STRING. THIS IS ACCOMPLISHED BY MOVING A CHARACTER DIRECTLY FROM ITS ORIGINAL POSITION TO ITS FINAL DESTINATION POSITION.

RSUB$A

RSUB$A IS A LOGICAL FUNCTION USED TO ROTATE A CHARACTER STRING, IT HAS THE FOLLOWING CALLING SEQUENCE:

```
LOG = RSUB$A(STRING,LENGTH,FCHAR,LCHAR,COUNT)
CALL RSUB$A(STRING,LENGTH,FCHAR,LCHAR,COUNT)
```

ARGUMENTS:

STRING = STRING CONTAINING SUBSTRING TO BE ROTATED, PACKED TWO CHARACTERS PER WORD, DATA TYPE DOES NOT MATTER.
LENGTH = LENGTH OF STRING IN CHARACTERS, INTEGER*2.
FCHAR = FIRST DELIMITING CHARACTER POSITION OF SUBSTRING, INTEGER*2.
LCHAR = LAST DELIMITING CHARACTER POSTION OF SUBSTRING, INTEGER*2.
COUNT = NUMBER OF POSITIONS TO ROTATE SUBSTRING, NEGATIVE COUNT CAUSES LEFT ROTATE, POSITIVE COUNT CAUSES RIGHT ROTATE, INTEGER*2.

FUNCTION:

RSUB$A WILL ROTATE THE SUBSTRING CONTAINED IN STRING LEFT OR RIGHT THE NUMBER OF POSITIONS SPECIFIED BY COUNT. ONLY THE CHARACTERS IN THE SUBSTRING ARE AFFECTED. THE STRING PARAMETERS PASSED TO THE ROUTINE ARE CHECKED FOR VALIDITY AND IF AN ERROR OCCURS A MESSAGE IS PRINTED AND THE FUNCTION VALUE WILL BE FALSE. IF EVERYTHING IS IN ORDER THE FUNCTION WILL BE TRUE.

APPLIB CALLS:

MCHR$A

COMMENTS:

THIS ROUTINE USES AN ALGORITHM THAT MINIMIZES BOTH TEMPORARY STORAGE AND EXECUTION TIME. TEMPORARY STORAGE USED CONSISTS OF ONE WORD AND THE NUMBER OF ITERATIONS NECESSARY TO ROTATE A STRING IS EQUAL TO THE LENGTH (IN CHARACTERS) OF THE STRING. THIS IS ACCOMPLISHED BY MOVING A CHARACTER DIRECTLY FROM ITS ORIGINAL POSITION TO ITS FINAL DESTINATION POSITION.

SSTR$A

SSTR$A IS A LOGICAL FUNCTION USED TO SHIFT A CHARACTER STRING LEFT OR RIGHT COUNT CHARACTERS, IT HAS THE FOLLOWING CALLING SEQUENCE:

```
LOG = SSTR$A(STRING,LENGTH,COUNT,FILCHR)
CALL  SSTR$A(STRING,LENGTH,COUNT,FILCHR)
```

ARGUMENTS:

STRING = CHARACTER STRING TO BE SHIFTED, PACKED TWO CHARACTERS PER WORD, DATA TYPE DOES NOT MATTER.
LENGTH = LENGTH OF STRING IN CHARACTERS, MUST BE GREATER THAN OR EQUAL TO ZERO, INTEGER*2.
COUNT = SHIFT COUNT, NEGATIVE COUNT CAUSES LEFT SHIFT, POSITIVE COUNT CAUSES RIGHT SHIFT, INTEGER*2.
FILCHR = FILL CHARACTER, VACATED CHARACTER POSITIONS WILL BE PADDED WITH THIS CHARACTER, SPECIFIED IN A1 FORMAT, DATA TYPE DOES NOT MATTER.

FUNCTION:

SSTR$A WILL SHIFT A CHARACTER STRING LEFT OR RIGHT THE SPECIFIED NUMBER OF CHARACTERS AND PAD VACATED POSITIONS WITH THE FILL CHARACTER. IF LENGTH IS LESS THAN ZERO AN ERROR MESSAGE WILL BE PRINTED AND THE FUNCTION WILL BE FALSE, OTHERWISE THE FUNCTION WILL BE TRUE. NO CHARACTERS ARE SHIFTED IF THE FUNCTION IS FALSE.

APPLIB CALLS:

FSUB$A, MCHR$A, NLEN$A

COMMENTS:

SSTR$A TRUNCATES STRING TO ITS OPERATIONAL LENGTH BEFORE PERFORMING THE SHIFT THEREFORE TRAILING BLANKS WILL NOT BE INCLUDED IN THE SHIFT.

SSUB$A

SSUB$A IS A LOGICAL FUNCTION USED TO SHIFT A CHARACTER SUBSTRING LEFT OR RIGHT COUNT CHARACTERS, IT HAS THE FOLLOWING CALLING SEQUENCE:

```
LOG = SSUB$A(STRING,LENGTH,FCHAR,LCHAR,COUNT,FILCHR)
CALL  SSUB$A(STRING,LENGTH,FCHAR,LCHAR,COUNT,FILCHR)
```

ARGUMENTS:

STRING = STRING CONTAINING SUBSTRING TO BE SHIFTED, PACKED TWO CHARACTERS PER WORD, DATA TYPE DOES NOT MATTER.
LENGTH = LENGTH OF STRING IN CHARACTERS, INTEGER*2.
FCHAR = FIRST DELIMITING CHARACTER POSITION OF SUBSTRING, INTEGER*2.
LCHAR = LAST DELIMITING CHARACTER POSITION OF SUBSTRING, INTEGER*2.
COUNT = SHIFT COUNT, NEGATIVE COUNT CAUSES LEFT SHIFT, POSITIVE COUNT CAUSES RIGHT SHIFT, INTEGER*2.
FILCHR = FILL CHARACTER, VACATED CHARACTER POSITIONS WILL BE PADDED WITH THIS CHARACTER, SPECIFIED IN A1 FORMAT, DATA TYPE DOES NOT MATTER.

FUNCTION:

SSUB$A WILL SHIFT A CHARACTER SUBSTRING LEFT OR RIGHT THE SPECIFIED NUMBER OF CHARACTERS AND PAD VACATED POSITIONS WITH THE FILL CHARACTER. THE STRING PARAMETERS ARE CHECKED FOR VALIDITY AND AN ERROR WILL CAUSE A MESSAGE TO BE PRINTED AND THE FUNCTION TO RETURN A VALUE OF FALSE. IF EVERYTHING IS IN ORDER THE FUNCTION WILL BE TRUE.

APPLIB CALLS:

FSUB$A, MCHR$A

COMMENTS:

ANY TRAILING BLANKS IN THE SUBSTRING WILL BE INCLUDED IN THE SHIFT. IF THE SUBSTRING IS NULL OR LENGTH IS EQUAL TO ZERO NO SHIFT WILL BE PERFORMED.

TREE$A

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

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

ARGUMENTS:

```
NAME   = ARRARY CONTAINING FILE NAME PACKED TWO CHARACTERS PER
         WORD, DATA TYPE DOES NOT MATTER.
NAMLEN = LENGTH OF NAME IN CHARACTERS, INTEGER*2.
FSTART = CHARACTER POSITION IN NAME OF FIRST CHARACTER IN FILENAME,
         INTEGER*2.
FLEN   = LENGTH OF FINAL FILE NAME IN CHARACTERS, INTEGER*2.
```

FUNCTION:

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.

APPLIB CALLS:

GCHR$A, NLEN$A

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$NAME, CAN STRING BE INTERPRETED AS A NAME,
         A$BIN, CAN STRING BE A BINARY NUMBER ?
         A$DEC, CAN STRING BE A DECIMAL NUMBER ?
         A$OCT, CAN STRING BE AN OCTAL NUMBER ?
         A$HEX, CAN STRING BE A HEXADECIMAL NUMBER ?
STRING = STRING TO BE TESTED, PACKED TWO CHARACTERS PER WORD,
         THDATEHA TYPE DOES NOT MATTER.
LENGTH = LENGTH OF STRING IN CHARACTERS, INTEGER*2.
```

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 BINARY NUMBER IF IT CONTAINS ONLY THE DIGITS 0 - 2, A DECIMAL NUMBER IF IT CONTAINS ONLY THE DIGITS 0 - 9, AN OCTAL NUMBER IF IT CONTAINS ONLY THE DIGITS 0 - 7, 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.

APPLIB CALLS:

GCHR$A, NLEN$A

3.3 USER QUERY ROUTINES

THE USER QUERY ROUTINES PROVIDE A CONVENIENT MEANS TO INPUT DATA FROM THE USERS TERMINAL. EACH ROUTINE AS THE ABILITY TO PROMPT THE TERMINAL USER WITH A SUPPLIED MESSAGE AND THEN INPUT HIS RESPONSE.

RNAM$A - PROMPT AND READ A NAME.

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

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

RNAM$A

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

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

ARGUMENTS:

```
MSG    = MESSAGE TEXT PACKED TWO CHARACTERS PER WORD, DATA TYPE
         DOES NOT MATTER.
MSGLEN = MESSAGE LENGTH IN CHARACTERS, INTEGER*2.
NAMKEY = A$FUPP, FORCE UPPER CASE (.NE. A$UPLW OR A$RAWI)
         A$UPLW, DO NOT FORCE UPPER CASE
         A$RAWI, READ ENTIRE LINE AS RAW TEXT.
NAME   = RETURNED NAME PACKED TWO CHARACTERS PER WORD, DATA TYPE
         DOES NOT MATTER.
NAMLEN = LENGTH OF NAME BUFFER IN CHARACTERS (.LE. 80), INTEGER*2.
```

FUNCTION:

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..

APPLIB CALLS:

NONE

RNUM$A

RNJM$A IS A LOGICAL FUNCTION USED TO INPUT NUMERIC DATA FROM THE USER TERMINAL, IT HAS THE FOLLOWING CALLING SEQUENCE:

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

ARGUMENTS:

MSG = ARRAY CONTAINING PROMPT MESSAGE, PACKED TWO CHARACTERS PER WORD, TYPE DOES NOT MATTER.
MSGLEN = LENGTH OF MSG IN CHARACTERS, INTEGER*2.
NUMKEY = BASE OF THE NUMBER BEING INPUT, POSSIBLE VALUES ARE:
A$BIN, BINARY NUMBER
A$OCT, OCTAL NUMBER
A$DEC, DECIMAL NUMBER
A$HEX, HEXADECIMAL NUMBER.
VALUE = BINARY VALUE RETURNED AS A RESULT OF CONVERSION ATTEMPTS, INTEGER*4.

APPLIB CALLS:

NONE

FUNCTION:

THIS ROUTINE WILL PRINT THE USER SUPPLIED PROMPT AND APPEND ': ' TO IT. IT THEN READS THE USER RESPONSE. IF THE RESPONSE IS NOT A LEGAL NUMBER OR IF THE NJMBER 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 WILL BE ZERO. IF A LEGAL NUMBER IS PROVIDED, THE FUNCTION WILL BE TRUE AND THE CONVERTED VALUE WILL BE RETURNED IN VALUE.

COMMENTS:

NUMBERS MAY BE PRECEDED BY AN OPTIONAL PLUS OR MINUS SIGN, IF A SIGN IS PRESENT THERE MUST BE NO INTERVENING SPACES BETWEEN IT AND THE FIRST DIGIT OF THE NUMBER. BINARY NJMBERS MAY HAVE A MAXIMUM OF 31 DIGITS, OCTAL A MAXIMUM OF 11, DECIMAL A MAXIMUM OF 10, AND HEXADECIMAL A MAXIMUM OF 8. NOTE THAT NEGATIVE BINARY, OCTAL, OR HEX NUMBERS SHOULD NOT BE ENTERED IN TWO'S COMPLEMENT FORM, BUT RATHER THE SAME AS YOU WOULD ENTER A NEGATIVE DECIMAL NUMBER.

YSNO$A

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

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

ARGUMENTS

```
MSG    = MESSAGE TEXT PACKED TWO CHARACTERS PER WORD, DATA TYPE
         DOES NOT MATTER.
MSGLEN = MESSAGE LENGTH IN CHARACTERS, INTEGER*2.
DEFKEY = A$NDEF, NO DEFAULT ACCEPTED
         A$DNO, DEFAULT = "NO" (.FALSE.)
         A$DYES, DEFAULT = "YES" (.TRUE.)
```

FUNCTION:

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.

APPLIB CALLS:

NONE

3.4 SYSTEM INFORMATION ROUTINES

THE SYSTEM INFORMATION ROUTINES RETURN THE SYSTEM DATE, SYSTEM TIME, CPU TIME, DISK TIME, ETC. IN CHARACTER STRING FORMAT.

CTIM$A - CPU TIME SINCE LOGIN.

DATE$A - TODAY'S DATE, AMERICAN STYLE.

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

DTIM$A - DISK TIME SINCE LOGIN.

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

TIME$A - TIME OF DAY.

CTIM$A

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

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

ARGUMENTS:

CPUTIM = CPU TIME IN CENTISECONDS, INTEGER*4.

FUNCTION:

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.

APPLIB CALLS:

NONE

DATE$A

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

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

ARGUMENTS:

DATE = DATE IN THE FORM 'DAY, MON DD YEAR', DATA TYPE DOES NOT MATTER AS LONG AS DATE ARRAY IS AT LEAST 16 CHARACTERS LONG.

FUNCTION:

THIS ROUTINE RETURNS THE DATE IN THE FORM 'DAY, MON DD YEAR'. 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.

APPLIB CALLS:

NONE

DOFY$A

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

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

ARGUMENTS:

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.

FUNCTION:

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.

APPLIB CALLS:

NONE

DTIM$A

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

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

ARGUMENTS:

DSKTIM = DISK TIME IN CENTISECONDS, INTEGER*4.

FUNCTION:

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.

APPLIB CALLS:

NONE

EDAT$A

EDAT$A IS A REAL*8 FUNCTION THAT RETURNS THE EUROPEAN STYLE CURRENT DATE, IT HAS THE FOLLOWING CALLING SEQUENCE:

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

ARGUMENTS:

EDATE = RETURNED STRING CONTAINING THE DATE IN THE FORM OF 'DAY, DD MON YEAR', PACKED TWO CHARACTERS PER WORD, TYPE DOES NOT MATTER AS LONG AS ARRAY IS AT LEAST 16 CHARACTERS LONG.

APPLIB CALLS:

DATE$A

FUNCTION:

EDAT$A RETURNS THE DATE AS 'DAY, DD MON YEAR' IN THE ARRAY EDATE AND THE FUNCTION VALUE RETURNS THE DATE AS 'DD.MM.YY'.

COMMENTS:

THE FUNCTION VALUE MUST BE RECEIVED IN A REAL*8 VARIABLE. THE ROUTINE IS GOOD FOR THE PERIOD: 1 JANJARY, 1977 - 31 DECEMBER, 2076.

TIME$A

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

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

ARGUMENTS:

TIME = TIME OF DAY IN THE FORM 'HR:MN:SC' PACKED TWO CHARACTERS PER WORD, DATA TYPE DOES NOT MATTER AS LONG AS IT IS AT LEAST 8 CHARACTERS LONG.

FUNCTION:

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.

APPLIB CALLS:

NONE

3.5 MATHEMATICAL ROUTINES

THE MATHEMATICAL ROUTINES PROVIDE MISCELLANEOUS FUNCTIONS NOT AVAILABLE IN MATHLIB. THE ROUTINES PRESENTLY AVAILABLE ARE RANDOM NUMBER FUNCTIONS.

RAND$A - GENERATE RANDOM NUMBER AND UPDATE "SEED".

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

RAND$A

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

```
R*4 = RAND$A(SEED)
R*8 = RAND$A(SEED)
CALL  RAND$A(SEED)
```

ARGUMENTS:

SEED = INPUT IS PREVIOUS SEED, OUTPUT IS NEW SEED, INTEGER*4.

FUNCTION:

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.

APPLIB CALLS:

NONE

RNDI$A

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

```
R*4 = RNDI$A(SEED)
R*8 = RNDI$A(SEED)
CALL  RNDI$A(SEED)
```

ARGUMENTS:

SEED = TIME OF DAY IN CENTISECONDS, INTEGER*4.

FUNCTION:

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.

APPLIB CALLS:

NONE

3.6 CONVERSION ROUTINES

THE CONVERSION ROUTINES PROVIDE A MEANS TO CONVERT A FIELD REPRESENTED BY ONE PARTICULAR FORMAT TO ANOTHER FORMAT.

CNVA$A - CONVERT ASCII DIGIT STRING TO BINARY NUMBER.

CNVB$A - CONVERT BINARY NUMBER TO ASCII DIGIT STRING.

ENCD$A - ENCODE FUNCTION THAT ADJUSTS THE "FORMAT" TO MAKE THE NUMBER PRINTABLE IF POSSIBLE.

FDAT$A - CONVERT DATMOD FIELD RETURNED BY RDEN$$ TO 'MM/DD/YY'.

FEDT$A - CONVERT DATMOD FIELD RETURNED BY RDEN$$ TO 'MM.DD.YY'.

FTIM$A - CONVERT TIMMOD FIELD RETURNED BY RDEN$$ TO 'HH:MM:SS'.

CASE$A - CONVERT A STRING FROM LOWER CASE TO UPPER OR VICE VERSA.

CASE$A

CASE$A IS A LOGICAL FUNCTION THAT WILL CONVERT A STRING FROM UPPER CASE TO LOWER OR VICE VERSA, IT HAS THE FOLLOWING CALLING SEQUENCE:

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

ARGUMENTS:

```
KEY    = A$FUPP, CONVERT LOWER CASE TO UPPER CASE,
         A$FLOW, CONVERT UPPER CASE TO LOWER CASE.
STRING = ARRAY CONTAINING CHARACTER STRING TO BE CONVERTED PACKED
         TWO CHARACTERS PER WORD, TYPE DOES NOT MATTER.
LENGTH = LENGTH OF STRING IN CHARACTERS, INTEGER*2.
```

APPLIB CALLS:

GCHR$A, MCHR$A

FUNCTION:

CASE$A WILL CONVERT ALL LOWER CASE ALPHABETIC CHARACTERS IN STRING TO UPPER CASE WHEN THE KEY A$FUPP IS USED AND WILL CONVERT ALL UPPER CASE ALPHABETIC CHARACTERS TO LOWER CASE WHEN THE KEY A$FLOW IS USED. THE FUNCTION WILL BE FALSE IF LENGTH IS LESS THAN ZERO, OTHERWISE IT WILL BE TRUE. IF NEITHER A$FUPP OR A$FLOW IS SPECIFIED THE DEFAULT IS NO CONVERSION.

CNVA$

CNVA$A IS A LOGICAL FUNCTION USED TO CONVERT AN ASCII DIGIT STRING TO ITS BINARY REPRESENTATION, IT HAS THE FOLLOWING CALLING SEQUENCE:

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

ARGUMENTS:

NUMKEY = BASE OF NUMBER IN NAME, POSSIBLE VALUES ARE:
A$BIN, BINARY,
A$OCT, OCTAL,
A$DEC, DECIMAL,
A$HEX, HEXADECIMAL.

NAME = ARRAY CONTAINING DIGIT STRING PACKED TWO CHARACTERS PER WORD DATA TYPE DOES NOT MATTER.

NAMLEN = LENGTH OF NAME IN CHARACTERS, INTEGER*2.

VALUE = CONVERTED BINARY VALUE, INTEGER*4.

FUNCTION:

THIS ROUTINE WILL CONVERT AN ASCII DIGIT STRING INTO ITS BINARY VALUE FOR BINARY, OCTAL, DECIMAL, 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 VALUE IS THE CONVERTED BINARY VALUE, IF CONVERSION IS NOT SUCCESSFUL THE FUNCTION IS FALSE AND VALUE WILL BE EQUAL TO ZERO. NOTE THAT FOR DECIMAL CONVERSIONS, OVERFLOW WILL BE CONSIDERED AS UNSUCCESSFUL WHEREAS FOR BINARY, OCTAL, AND HEXADECIMAL CONVERSIONS, OVERFLOW IS IGNORED.

FOR CONVERSION TO BE SUCCESSFULL THE MAXIMUM NUMBER OF DIGITS IN NAME MAY BE THE FOLLOWING: FOR BINARY - 31, FOR OCTAL - 11, FOR DECIMAL - 10, AND FOR HEXADECIMAL - 8 (THE MAXIMUM DOES NOT INCLUDE LEADING SIGNS OR BLANKS).

APPLIB CALLS:

GCHR$A, NLEN$A

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$A(NUMKEY,VALUE,NAME,NAMLEN)
CALL  CNVB$A(NUMKEY,VALUE,NAME,NAMLEN)
```

ARGUMENTS:

NUMKEY = NUMBER BASE TO CONVERT TO, POSSIBLE VALUES ARE:
A$BIN, BINARY, LEADING BLANKS,
A$BIN7, BINARY, LEADING ZEROS,
A$DEC, SIGNED DECIMAL, LEADING BLANKS,
A$DECU, UNSIGNED DECIMAL, LEADING BLANKS,
A$DEC7, SIGNED DECIMAL, LEADING ZEROS,
A$OCT, OCTAL, LEADING BLANKS,
A$OCT7, OCTAL, LEADING ZEROS,
A$HEX, HEXADECIMAL, LEADING BLANKS,
A$HEX7, HEXADECIMAL, LEADING ZEROS.

NAME = ARRAY CONTAINING RETURNED ASCII DIGIT STRING PACKED TWO CHARACTERS PER WORD, DATA TYPE DOES NOT MATTTER.

NAMLEN = LENGTH OF NAME IN CHARACTERS, INTEGER*2.

VALUE = BINARY VALUE PASSED TO ROUTINE, INTEGER*4.

FUNCTION:

CNVB$A WILL CONVERT A BINARY INTEGER*4 NUMBER INTO AN ASCII DIGIT STRING IN THE BASE SPECIFIED BY NUMKEY. THE RETURNED DIGIT STRING WILL BE RIGHT JUSTIFIED IN NAME AND PRECEDED BY LEADING BLANKS OR ZEROS DEPENDING ON NUMKEY.

IF THE NUMBER IS TO BE TREATED AS SIGNED DECIMAL AND VALUE IS NEGATIVE, THE DIGIT STRING WILL HAVE A '-' PREFIXED TO IT. FOR BINARY, OCTAL, AND HEXADECIMAL NUMBERS IF VALUE IS NEGATIVE THE DIGIT STRING WILL BE IN TWO'S COMPLEMENT FORM.

IF THE NUMBER CONVERTS SUCCESSFULLY THE FUNCTION WILL BE EQUAL TO THE NUMBER OF DIGITS IN NAME IF NOT, IT WILL BE EQUAL TO ZERO.

NAMLEN REQUIREMENTS FOR CONVERSION ARE AS FOLLOWS: FOR BINARY THE MAXIMUM NUMBER OF DIGITS THAT MIGHT BE RETURNED IN NAME IS 31, FOR OCTAL MAX = 11, FOR DECIMAL MAX = 10, AND FOR HEXADECIMAL MAX = 8 (MAX DOES NOT INCLUDE LEADING SIGNS OR LEADING ZEROS).

APPLIB CALLS:

FILL$A, MCHR$A

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)
```

ARGUMENTS:

ARRAY = ARRAY TO RECEIVE VALUE PACKED TWO CHARACTES PER WORD, DATA TYPE DOES NOT MATTER.
WIDTH = FIELD WIDTH AS IN FORMAT FW.D (SHOULD BE EVEN), INTEGER*2.
DEC = PLACES TO RIGHT OF DECIMAL POINT AS IN FORMAT FW.D, INTEGER*2.
VALUE = DOUBLE PRECISION VALUE TO BE ENCODED, REAL*8.

FUNCTION:

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.

APPLIB CALLS:

NONE

FDAT$A

FDAT$A CONVERTS A DATE FROM THE RDEN$$ DATMOD FORMAT TO 'DAY, MON DD YEAR' AND 'MM/DD/YY', IT HAS THE FOLLOWING CALLING SEQUENCE:

```
CALL   FDAT$A(DATMOD,DATE)
R*8 = FDAT$A(DATMOD,DATE)
```

ARGUMENTS:

DATMOD = THE DATE WHEN THE FILE WAS LAST MODIFIED, AS RETURNED BY RDEN$$, IN THE FORM YYYYYYYMMMMDDDDD, WHERE YYYYYYY IS THE YEAR MODULO 100, MMMM IS THE MONTH, AND DDDDD IS THE DAY, INTEGER*2.

DATE = ARRAY CONTAINING THE DATE AS A CHARACTER STRING, PACKED TWO CHARACTERS PER WORD, IN THE FORM 'DAY, MON DD YEAR', TYPE DOES NOT MATTER AS LONG AS ARRAY IS AT LEAST 16 CHARACTERS LONG.

APPLIB CALLS:

CNVB$A

FUNCTION:

FDAT$A CONVERTS THE DATMOD FIELD AS RETURNED BY RDEN$$ TO THE FORM 'DAY, MON DD YEAR' IN THE ARGUMENT DATE AND THE FUNCTION VALUE WILL BE THE DATE IN THE FORM 'MM/DD/YY'.

COMMENTS:

THE FUNCTION VALUE MUST BE RECEIVED IN A REAL*8 VARIABLE. THE ROUTINE IS GOOD FOR THE PERIOD: 1 JANJARY, 1972 - 31 DECEMBER, 2071.

FEDT$A

FEDT$A CONVERTS A DATE FROM THE RDEN$$ DATMOD FORMAT TO 'DAY, MON DD YEAR' AND 'MM.DD.YY', IT HAS THE FOLLOWING CALLING SEQUENCE:

```
CALL   FEDT$A(DATMOD,DATE)
R*8 = FEDT$A(DATMOD,DATE)
```

ARGUMENTS:

DATMOD = THE DATE WHEN THE FILE WAS LAST MODIFIED, AS RETURNED BY RDEN$$, IN THE FORM YYYYYYYMMMMDDDDD, WHERE YYYYYYY IS THE YEAR MODULO 100, MMMM IS THE MONTH, AND DDDDD IS THE DAY, INTEGER*2.

DATE = ARRAY CONTAINING THE DATE AS A CHARACTER STRING, PACKED TWO CHARACTERS PER WORD, IN THE FORM 'DAY, MON DD YEAR', TYPE DOES NOT MATTER AS LONG AS ARRAY IS AT LEAST 16 CHARACTERS LONG.

APPLIB CALLS:

FDAT$A

FUNCTION:

FEDT$A CONVERTS THE DATMOD FIELD AS RETRUVED BY RDEN$$ TO THE FORM 'DAY, MON DD YEAR' IN THE ARGUMENT DATE AND THE FUNCTION VALUE WILL BE THE DATE IN THE FORM 'MM.DD.YY' (EUROPEAN STYLE).

COMMENTS:

THE FUNCTION VALUE MUST BE RECEIVED IN A REAL*8 VARIABLE. THE ROUTINE IS GOOD FOR THE PERIOD: 1 JANJARY, 1972 - 31 DECEMBER, 2071.

FTIM$A

FTIM$A IS A REAL*8 OR REAL*4 FUNCTION WHICH WILL CONVERT THE 'TIME FILE WAS LAST MODIFIED' FIELD RETURNED BY RDEN$$ TO THE FORM 'HH:MM:SS' AND DECIMAL HOURS, IT HAS THE FOLLOWING CALLING SEQUENCE:

```
CALL  FTIM$A(TIMMOD,TIME)
R*8 = FTIM$A(TIMMOD,TIME)
R*4 = FTIM$A(TIMMOD,TIME)
```

ARGUMENTS:

TIMMOD = THE TIME AT WHICH A FILE WAS LAST MODIFIED, IN THE FORM 'SECONDS SINCE MIDNIGHT' DIVIDED BY FOUR, INTEGER*2.

TIME = ARRAY CONTAINING THE TIME A FILE WAS LAST MODIFIED AS A CHARACTER STRING OF THE FORM 'HH:MM:SS', TYPE DOES NOT MATTER AS LONG AS ARRAY IS AT LEAST 8 CHARACTERS LONG.

APPLIB CALLS:

CNVD$A

FUNCTION:

FTIM$A WILL CONVERT THE TIMMOD FIELD RETURNED BY THE THE FILE SYSTEM ROUTINE RDEN$$ TO THE FORM 'HH:MM:SS'. THE FUNCTION VALUE IS THE TIMMOD FIELD CONVERTED TO DECIMAL HOURS AND MAY BE RECEIVED AS EITHER REAL*4 OR REAL*8.

3.7 PARSING ROUTINES

THE PARSING ROUTINES ARE DESIGNED TO FACILITATE THE DESIGN AND IMPLEMENTATION OF USER INTERFACES IN A PROGRAM. THEY PROVIDE A MEANS TO BREAK A CHARACTER STRING UP INTO TOKENS AND RETURN VARIOUS INFORMATION CONCERNING EACH TOKEN. AT PRESENT THE ONLY ROUTINE AVAILABLE IS CMDL$A WHICH CAN BE USED TO PARSE A PRIMOS TYPE COMMAND LINE.

CMDL$A - PARSE PRIMOS TYPE COMMAND LINE.

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$NKWL, TURN ON DEFAULT PROCESSING AND RETURN THE NEXT KEYWORD ENTRY IN THE COMMAND LINE. THIS KEY ALLOWS THE CALLING PROGRAM TO SWITCH KEYWORD LISTS IN THE MIDDLE OF A COMMAND LINE,
A$PCMD, PERMITS THE USE OF A KEYWORD WITHOUT A PRECEDING MINUS SIGN AS THE FIRST TOKEN ON A LINE (MAY ONLY BE USED FOR LINES SUBSEQUENT TO THE INITIAL 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.

APPLIB CALLS:

CNVA$A, CNVB$A, CSUB$A, FILL$A, JSTR$A, MSJB$A, MSTR$A, NLEN$A, TYPE$A, SSUB$A

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 PRECEEDED 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 PRECEEDED 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 PRECEEDED 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).

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:

WORD 1,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$NONE
A$BIN, OPTION MUST BE A BINARY NUMBER
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$NBIN, OPTION MAY BE A NAME OR A BINARY NUMBER
A$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$RAWI, OPTION IS THE REMAINDER OF THE COMMAND LINE AFTER THE CURRENT -<EYWORD READ AS RAW TEXT. USE OF THIS OPTION TYPE WILL TURN ON THE END OF LINE INDICATOR IN THE SAME MANNER AS A KEY OF A$RAWI.

DEFAULT LIST:

WORD 1 = NUMBER OF DEFAULT KEYWORDS, MUST BE .GE. ZERO
WORD 2,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$REQD,A$NAME,
     * '-NDECIMAL      ',2,2,A$OPTL,A$NDEC,
     * '-OCTAL         ',3,2,A$REQD,A$OCT,
     * '-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,
     * 4,1,2,8,99/
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
```

4 SUMMARY

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

4.1 CALLING SEQUENCES

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	CLOS$A	L	(UNIT)
	DELE$A	L	(NAME,NAMLEN)
	EXST$A	L	(NAME,NAMLEN)
	GEND$A	L	(UNIT)
	OPEN$A	L	(OPNKEY+TYPKEY+UNTKEY,NAME,NAMLEN,UNIT)
	OPNP$A	L	(MSG,MSGLEN,OPNKEY+TYPKEY+UNTKEY,NAME, NAMLEN,UNIT)
	OPNV$A	L	(OPNKEY+TYPKEY+UNTKEY,NAME,NAMLEN,UNIT, VERKEY,WTIME,RETRYS)
	OPVP$A	L	(MSG,MSGLEN,OPNKEY+TYPKEY+UNTKEY,NAME, NAMLEN,UNIT,VERKEY,WTIME,RETRYS)
	POSN$A	L	(POSKEY,UNIT,POS)
	RPOS$A	L	(UNIT,POS)
	RWND$A	L	(UNIT)
	TEMP$A	L	(TYPKEY+UNTKEY,NAME,NAMLEN,UNIT)
	TRNC$A	L	(UNIT)
	TSCN$A	L	(KEY,UNITS,ENTRY,MAXSIZ,ENTSIZ,MAXLEV, LEV,CODE)
	UNIT$A	L	(UNIT)
STRING	CSTR$A	L	(A,ALEN,B,BLEN)
	CSUB$A	L	(A,ALEN,AFC,ALC,B,BLEN,BFC,BLC)
	FILL$A	I	(NAME,NAMLEN,CHAR)
	FSUB$A	L	(STRING,LENGTH,FCHAR,LCHAR,FILCHR)
	GCHR$A	I	(FARRAY,FCHAR)
	JSTR$A	L	(KEY,STRING,LENGTH)
	LSTR$A	L	(A,ALEN,B,BLEN,FCP,LCP)
	LSUB$A	L	(A,ALEN,AFC,ALC,B,BLEN,BFC,BLC,FCP,LCP)
	MCHR$A	I	(TARRAY,TCHAR,FARRAY,FCHAR)
	MSTR$A	I*2	(A,ALEN,B,BLEN)
	MSUB$A	I*2	(A,ALEN,AFC,ALC,B,BLEN,BFC,BLC)
	NLEN$A	I*2	(NAME,NAMLEN)
	RSTR$A	L	(STRING,LENGTH,COUNT)
	RSUB$A	L	(STRING,LENGTH,FCHAR,LCHAR,COUNT)
	SSTR$A	L	(STRING,LENGTH,COUNT,FILCHR)

	SSUB$A	L	(STRING,LENGTH,FCHAR,LCHAR,COUNT,FILCHR)
	TREE$A	I	(NAME,NAMLEN,FSTART,FLEN)
	TYPE$A	L	(KEY,STRING,LENGTH)
USER QUERY	RNAM$A	L	(MSG,MSGLEN,NAMKEY,NAME,NAMLEN)
	RNUM$A	L	(MSG,MSGLEN,NUMKEY,VALUE)
	YSNO$A	L	(MSG,MSGLEN,DEFKEY)
SYSTEM INFO	CTIM$A	DP	(CPUTIM)
	DATE$A	DP	(DATE)
	DOFY$A	DP	(DOFY)
	DTIM$A	DP	(DSKTIM)
	EDAT$A	DP	(EDATE)
	TIME$A	DP	(TIME)
MATHEMATICAL	RAND$A	DP	(SEED)
	RNDI$A	DP	(SEED)
CONVERSION	CASE$A	L	(KEY,STRING,LENGTH)
	CNVA$A	L	(NUMKEY,NAME,NAMLEN,VALUE)
	CNVB$A	I	(NUMKEY,VALUE,NAME,NAMLEN)
	ENCD$A	L	(ARRAY,WIDTH,DEC,VALUE)
	FDAT$A	DP	(DATMOD,DATE)
	FEDT$A	DP	(DATMOD,DATE)
	FTIM$A	DP	(TIMMOD,TIME)
PARSING	CMDL$A	L	(KEY,KWLIST,KWINDX,OPTBUF,BUFLEN,OPTION, VALUE,KWINFO)

4.2 SYSCOM>A$KEYS

```
: A$KEYS, APPLIB>SOURCE, ELS, 01/09/79
: INSERT FILE FOR MNEMONIC APPLIB KEYS (FTN)
: COPYRIGHT 1977, PRIME COMPUTER, INC., FRAMINGHAM, MA.
      NOLIST
:
:     ******************************************************************
:     *                                                                *
:     *              FUNCTION DECLARATIONS (TABSET 6 17)               *
:     *                                                                *
:     ******************************************************************
:
      LOGICAL   CLOS$A,RWND$A,GEND$A,TRNC$A,DELE$A,RPOS$A,POSN$A,TEMP$A,
     X          OPEN$A,OPNV$A,OPNP$A,OPVP$A,ENCD$A,YSNO$A,RNAM$A,RNUM$A,
     X          TREE$A,EXST$A,UNIT$A,CNVA$A,CMDL$A,CSUB$A,CSTR$A,TYPE$A,
     X          TSCN$A,JSTR$A,LSUB$A,LSTR$A,FSUB$A,SSTR$A,SSUB$A,RSTR$A,
     X          RSUB$A,CSP$$A,CASE$A
C
      INTEGER   MCHR$A,GCHR$A,FILL$A
      INTEGER*2 NLEN$A,MSUB$A,MSTR$A,CNVB$A
C
      REAL   *8 DOFY$A,DATE$A,EDAT$A,TIME$A,CTIM$A,DTIM$A,RNDI$A,RAND$A,
     X          FEDT$A,FTIM$A,FDAT$A
C
C     ******************************************************************
C     *                                                                *
C     *                KEY DECLARATIONS (TABSET 6 17)                  *
C     *                                                                *
C     ******************************************************************
C
      INTEGER*2 A$READ,A$WRIT,A$RDWR,A$SAMF,A$DAMF,A$NVER,A$VNEW,A$OVAP,
     X          A$VOLD,A$ABS ,A$REL ,A$DEC ,A$OCT ,A$HEX ,A$NDEF,A$DNO ,
     X          A$DYES,A$FUPP,A$UPLW,A$RAWI,A$NONE,A$OPTL,A$REQD,A$NDEC,
     X          A$NOCT,A$NHEX,A$NAME,A$NUMB,A$NEXT,A$RSET,A$RCMD,A$NKWL,
     X          A$NOVF,A$TREE,A$DLAY,A$NUFD,A$NSEG,A$CUFD,A$DECZ,A$DECU,
     X          A$OCTZ,A$HEXZ,A$RGHT,A$LEFT,A$CNTR,A$BACK,A$FLOW,A$BIN,
     X          A$NBIN
C
      PARAMETER
     X
     X /*****************************************************************/
     X /*                                                               */
     X /*            KEY DEFINITIONS (TABSET 6 11 28 69)                */
     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$DAMF = :2000,    /* OPEN NEW DAM FILE                             */
X  /*                  ****** UNTKEY ******                              */
X     A$GFTU = :40000,   /* OPEN AND GET UNIT                             */
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$HEX  = 3,        /* HEXADECIMAL                                   */
X     A$BIN  = 9,        /* BINARY                                        */
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$BIN  = 9,        /* BINARY, LEFT PADDED WITH BLANKS               */
X     A$DECZ = 4,        /* DECIMAL, LEFT PADDED WITH ZEROS               */
X     A$OCTZ = 5,        /* OCTAL, LEFT PADDED WITH ZEROS                 */
X     A$HEXZ = 6,        /* HEXADECIMAL, LEFT PADDED WITH ZEROS           */
X     A$DECU = 7,        /* UNSIGNED DECIMAL, LEFT PADDED WITH            */
X  /*                       BLANKS                                        */
X     A$BINZ = 8,        /* BINARY, LEFT PADDED WITH ZEROS                */
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     A$RCMD  = 6,        /* FIRST TOKEN IS COMMAND (NO '-')          */
X  /*                  ****** OPTYPE ******                           */
X  /*  A$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$NAME  = 8,        /* NAME                                    */
X  /*  A$BIN   = 9,        /* BINARY OPTION                           */
X     A$NBIN  =10,        /* NAME OR BINARY OPTION                   */
X  /*                  ****** OPTION ******                           */
X     A$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     A$CNTR  = 3,        /* CENTER                                  */
X  /*                                                                */
X  /********************** CASE$A ********************************** */
X  /*                  ****** KEY    ******                          */
X  /*  A$FUPP  = 1,        /* FORCE UPPER CASE                        */
X     A$FLOW  = 5         /* FORCE LOWER CASE                        */
X  /*                                                                */
X  /********************** TYPE$A ********************************** */
X  /*                  ****** KEY    ******                          */
X  /*  A$BIN   = 9,        /* BINARY NUMBER                           */
X  /*  A$DEC   = 1,        /* DECIMAL NUMBER                          */
```

```
Y /* A$OCT   = 2,        /* OCTAL NUMBER                                  */
X /* A$HEX   = 3,        /* HEXADECIMAL NUMBER                            */
X /* A$NAME  = 8         /* NAME                                          */
X /*                                                                       */
X /*************************************************************************/
 LIST
```

4.3 ROUTINE INDEX

```
**********************************************************************************************
**********************************************************************************************
**
**
**
**      JJJ    III    M    M   M    M   Y    Y
**       J      I     MM  MM   MM  MM   Y    Y
**       J      I     M M M    M M M     Y Y
**       J      I     M M M    M M M      Y
**    J  J      I     M   M    M   M      Y
**    J  J      I     M   M    M   M      Y
**     JJ      III    M   M    M   M      Y
**
**
**
**    RRRR    FFFFF  77777  77777
**    R   R   F          7      7
**    R   R   F         7      7
**    RRRR    FFFF     7      7
**    R R     F       7      7
**    R  R    F      7      7
**    R   R   F      7      7
**
**
**
**********************************************************************************************
**********************************************************************************************
```

FORTRAN-77 FOR REV. 17.0A

DATE: AUGUST 16, 1979

TO:

FROM:

SUBJECT: FORTRAN-77 FOR REV. 17.0A

REFERENCE: NONE

ABSTRACT

THIS MEMO DESCRIBES THE OPERATING PROCEDURES OF THE NEW FORTRAN-77 COMPILER. IN PARTICULAR, THE COMMAND LINE OPTIONS OF THE NEW COMPILER ARE LISTED AND NOTES ON THE USE OF THE COMPILER ARE FURNISHED. THIS INFORMATION WILL ALLOW USERS ALREADY FAMILIAR WITH FORTRAN-77 TO COMPILE, LOAD, AND EXECUTE THEIR PROGRAMS. USERS NOT FAMILIAR WITH THE FORTRAN-77 LANGUAGE SHOULD READ THE FORTHCOMING USER'S MANUAL.

1 F77 COMPILER OPTIONS

F77 IS THE FORTRAN-77 COMPILER. IT IS INVOKED BY THE COMMAND:

F77 NAME [OPTIONS]

OPTIONS ARE PRECEDED BY A '-'. THE NAME MAY BE A PATH NAME, BUT NEITHER IT NOR ANY FILE NAME MADE FROM IT MAY EXCEED 32 CHARACTERS. THE COMMAND LINE SYNTAX IS THE SAME AS OTHER PRIME COMPILERS: THE -S, -B, AND -L OPTIONS ARE ALL SUPPORTED.

THE FOLLOWING OPTIONS ARE SUPPORTED:

-XREF	-- PRODUCE A CROSS-REFERENCE LISTING. (IMPLIES L)
-NOXREF	-- NO CROSS REFERENCE.
-OFFSET	-- PRODUCE AN OFFSET MAP IN L_NAME (IMPLIES L)
-EXPLIST	-- PRODUCE A PSEUDO-ASSEMBLY LISTING OF THE GENERATED CODE IN L_NAME (IMPLIES L)
-OPTIMIZE	-- EXECUTE THE OPTIMIZER PHASE
-NOOPTIMIZE	-- DON'T USE THE OPTIMIZER
-STATISTICS	-- PRINT OUT STATISTICS ABOUT THE COMPILATION
-RANGE	-- COMPILE CODE TO CHECK SUBSCRIPT AND SUBSTR RANGES
-UPCASE	-- MAP LOWER CASE TO UPPER CASE IN IDENTIFIERS
-LCASE	-- UPPER AND LOWER CASE ARE DISTINCT IN IDENTIFIERS
-SILENT	-- SUPPRESS LEVEL 1 (WARNING) ERROR MESSAGES
-DEBUG	-- PRODUCE A FULL DEBUGGER (DBG) SYMBOL TABLE AND DEBUGGER LISTING (DEBUGGER LISTING IS CALLED "FILENAME.DBG")
-64V	-- PRODUCE V-MODE CODE
-BIG	-- DENOTES THAT ARRAYS MAY BE LARGER THAN 1 SEGMENT
-NOBIG	-- ASSUMES ARRAYS ARE LESS THAN ONE SEGMENT
-PRODUCTION	-- PRODUCE "PRODUCTION" DEBUGGER SYMBOL TABLE
-INTL	-- MAKE INTEGER*4 THE DEFAULT
-INTS	-- MAKE INTEGER*2 THE DEFAULT
-LOGL	-- MAKE LOGICAL*4 THE DEFAULT
-LOGS	-- MAKE LOGICAL*2 THE DEFAULT
-DO1	-- CAUSES ALL DO LOOPS TO BE AT LEAST ONE TRIP
-DYNM	-- ALLOCATE LOCAL VARIABLES ON THE STACK.
-SAVE	-- ALLOCATE LOCAL VARIABLES IN STATIC SPACE.

THE DEFAULT OPTIONS AS DISTRIBUTED ARE '-B YES -L NO -64V -OPTIMIZE -UPCASE -INTL -LOGL -NOBIG'. THE DEFAULT OPTIONS MAY BE CHANGED BY USE OF THE PROGRAM F77DF, WHICH IS FOUND IN THE TOOLS UFD.

EXAMPLE:

```
F77 FOO -L YES -INTS
```

WILL COMPILE FOO TO PRODUCE AN OBJECT FILE NAMED B_FOO AND A LISTING FILE NAMED L_FOO. INTEGERS WILL BE ASSUMED TO BE INTEGER*2, LOGICALS WILL BE ASSUMED TO BE LOGICAL*4, AND THE CODE WILL BE OPTIMIZED.

EACH COMPILATION PRODUCES TEMPORARY FILES (NAMED "T$XXXX") IN THE CURRENT WORKING DIRECTORY. THESE FILES ARE NORMALLY DELETED AT THE END OF THE COMPILATION.

2 ERROR MESSAGES

ERROR MESSAGES ARE WRITTEN TO THE TERMINAL AND TO FILENAME.ERROR. FOUR LEVELS OF ERRORS ARE REPORTED: LEVEL 1 IS A WARNING, LEVEL 2 IS AN ERROR THAT HAS BEEN FIXED, LEVEL 3 IS AN ERROR THAT HAS NOT BEEN FIXED, AND LEVEL 4 IS AN ERROR THAT PREVENTS CONTINUED COMPILATION. ANY ERROR OF LEVEL 3 PREVENTS OPTIMIZATION AND CODE GENERATION IF DETECTED PRIOR TO THOSE PHASES.

3 PROGRAM LOADING

AT REV. 17.0 THE F77 COMPILER OUTPUTS V-MODE CODE EXCLUSIVELY. THUS, THE SEGMENTED LOADER (SEG) MUST BE USED TO LOAD THE OBJECT MODULES PRODUCED BY THE COMPILER. ONLY THE STANDARD LIBRARY IS NEEDED.

4 MISCELLANEOUS NOTES

4.1 CROSS-REFERENCE OPTION

THE CROSS-REFERENCE OPTION MAY CAUSE THE COMPILER'S VIRTUAL SYMBOL SPACE TO OVERFLOW FOR VERY LARGE SOURCE PROGRAMS.

4.2 DBG INTERFACE

THE LINE NUMBERS GIVEN IN DBG REFER TO THE LINE NUMBERS FOUND IN THE DEBUGGER LISTING FILE (NAMED "FILENAME.DBG"). THESE NUMBERS WILL DIFFER FROM THOSE OF THE SOURCE IF "$INSERT" STATEMENTS ARE USED, SINCE THE INCLUDED TEXT WILL APPEAR IN THE DEBUGGER LISTING.

SINCE COMPILATION IN DEBUG MODE PRODUCES EXTRA INFORMATION IN THE BINARY AND SEG FILES IN ADDITION TO PRODUCING THE DEBUGGER LISTING FILE, USE OF THE "-DEBUG" OPTION FOR LARGE PROGRAMS MAY REQUIRE A SIGNIFICANT AMOUNT OF DISK SPACE.

AT REV 17.0, USE OF DBG ON PROGRAMS IN WHICH MULTIPLE EXTERNAL PROCEDURES EXIST IN A SINGLE SOURCE FILE IS

NOT SUPPORTED.

4.3 SEGMENT USAGE

COMPILATION OF F77 PROGRAMS USES SEGMENTS 4004-4007 AND 4027. IN USER PROGRAMS SEGMENTS 4027 THROUGH 4010 ARE USED - IN DESCENDING ORDER - AS THE SYSTEM FREE STORAGE POOL (IN WHICH ALLOCATE AND FREE REQUESTS OPERATE AND IN WHICH SOME COMPILER-GENERATED TEMPORARIES ARE ALLOCATED).

```
**********************************************************************************************
**********************************************************************************************
*
*
*
*       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 M     Y Y
*         J       I     M  M  M    M  M  M      Y
*     J   J       I     M     M    M     M      Y
*     J   J       I     M     M    M     M      Y
*      JJ       III     M     M    M     M      Y
*
*
*
*     RRRR     RRRR     PPPP      GGG
*     R   R    R   R    P   P    G   G
*     R   R    R   R    P   P    G
*     RRRR     RRRR     PPPP     G
*     R R      R R      P        G  GG
*     R  R     R  R     P        G   G
*     R   R    R   R    P         GGGG
*
*
*
**********************************************************************************************
**********************************************************************************************
```

RPG FOR REV. 17

DATE: APRIL 5, 1979

TO:

FROM:

SUBJECT: RPG FOR REV. 17

REFERENCE: NONE

ABSTRACT

THIS DOCUMENT DESCRIBES THE ENHANCEMENTS AND MODIFICATIONS TO THE PRIME RPG-II COMPILER AND RUNTIME LIBRARIES FOR REV 17 RELEASE. THE ENHANCEMENTS THAT HAVE BEEN INCLUDED ARE REDUCTION IN AN RPG-II PROGRAM WORKING SET SIZE, BIT OPERATIONS, ZONE/DIGIT FUNCTION FOR BOTH INPUT AND CALCULATIONS, INCREASED FUNCTIONALITY OF EDIT CODES AND OUTPUT PRINTING SUPPORT.

TABLE OF CONTENTS

I. COMPILER REVISIONS

A. WORKING SET

THE REV17 RPG COMPILER DETERMINES DURING COMPILATION WHETHER OR NOT CERTAIN PORTIONS OF THE RUNTIME LIBRARY NEED BE INCLUDED IN THE PROGRAM'S MEMORY IMAGE. MODULES WHICH ARE DETERMINED BY THE COMPILER AS NOT TO BE REFERENCED DURING THE EXECUTION OF AN RPG PROGRAM ARE SATISFIED BY A ONE WORD ENTRY POINT.

THE ACTUAL REDUCTION IN SIZE OF EACH RPG PROGRAM DEPENDS ON THE FUNCTIONING OF THE PROGRAM. WITHOUT USING MIDAS FILES, PACKED BINARY DATA, PACKED DECIMAL DATA, THE DEBUG OPTION, THE DISPLAY OPTION, THE SPECIAL FILE TYPE, THE ZONE OPERATIONS, AND THE BIT OPERATIONS, PROGRAM SIZE OF A REV17 RPG PROGRAM WILL BE REDUCED BY MORE THAN 30,000 WORDS. THE ACCOMPANYING CHART INDICATES THE APPROXIMATE SIZE OF EACH MODULE SET THAT MAY OR MAY NOT BE INCLUDED BASED ON THE FUNCTION OF THE PROGRAM.

FUNCTION	SIZE IN OCTAL WORDS
PACKED BINARY	1276
PACKED DECIMAL	1173
DSPLY OPERATION	630
DEBUG OPERATION	1075
EXTERNAL SUBROUTINES	250 + SUBROUTINE SIZE
KIDA FUNCTIONS - (CHAIN, MIDAS FILES)	25500
ZONE OPERATIONS	1112
BIT OPERATIONS	235
TABLES	1530

TO BUILD AN RPG PROGRAM UNDER REV17:

```
OK, LOAD
$ MODE D64R
$ DC
$ LOAD B_PROGRAM NAME
$ LIB RPGLIB
$ LIB KIDALB     (IF MIDAS FILES ARE USED)
$ LIB
$ SAVE *PROGRAM NAME
LOAD COMPLETE
$ MAP M_PROGRAM NAME
$ QUIT
OK,
```

NOTE: THE UNDERSCORED PORTIONS ARE PROMPTED BY THE SYSTEM.

B. INDICATOR SUMMARY ON COMPILER GENERATED LISTING OF THE RPG PROGRAM

THE REV17 RPG COMPILER GENERATES ADDITIONAL INFORMATION ON THE OUTPUT LISTING. DURING COMPILATION, THE COMPILER KEEPS TRACK

OF INDICATORS ASSIGNED AND USED AND AT THE CONCLUSION OF THE COMPILATION, CORRELATES THIS INFORMATION. THE INDICATOR SUMMARY INCLUDES A LIST OF INDICATORS ASSIGNED, INDICATORS USED TO CONDITION OPERATIONS, INDICATORS ASSIGNED BUT NOT USED TO CONDITION OPERATIONS, AND INDICATORS USED TO CONDITION OPERATIONS BUT NOT ASSIGNED.

C. ERROR MESSAGE ADDITION

MESSAGE: END POSITION TOO LOW.
MEANING: THE LENGTH OF THE FIELD TO BE OUTPUT IS GREATER THAN THE SPACE ALLOCATED BY THE RPG PROGRAM.
SEVERITY: ERROR.
SPECIFICATION TYPE: OUTPUT.
CORRECTION: ADJUST THE END POSITION OF THE OUTPUT FIELD.

II. RUNTIME LIBRARIES

A. ZONE OPERATIONS

1. INPUT SPECIFICATIONS

CARD COLUMN 26 OF THE INPUT SPECIFICATIONS MAY CONTAIN A C, Z, OR D. THE C IN 26 SIGNIFIES THAT THE RECORD IDENTIFYING INDICATOR WILL BE TURNED ON IF THERE IS A FULL CHARACTER MATCH ON THE SPECIFIED CHARACTER.

EXAMPLE (19 - 27):

```
: 0: 1:  :  : 1: 0:  : C: D:
:19:20:21:22:23:24:25:26:27:     COLUMN NUMBER
```

MEANING: INDICATOR 01 WILL BE ACTIVATED IF POSITION 10 OF THE INPUT CONTAINS A CHARACTER D (:304).

A 'Z' FOUND IN 26 WILL ACTIVATE THE INDICATOR DESIGNATED IF THE ZONE PORTION OF THE INPUT RECORD CHARACTER MATCHES THE ZONE PORTION OF THE CONTROL CHARACTER.

EXAMPLE (19 - 27):

```
: 0: 2:  :  :  : 9:  : Z: D:
:19:20:21:22:23:24:25:26:27:     COLUMN NUMBER
```

MEANING: INDICATOR 02 WILL BE ACTIVATED IF THE ZONE OF THE CHARACTER IN POSITION 9 OF THE INPUT RECORD MATCHES THE ZONE PORTION OF D. REFER TO THE TABLE OF EQUAL ZONE GROUPINGS. THE FOUR LEFT BITS OF THE EBCDIC BIT STRING FOR D ARE 1100. ANY CHARACTER WITH THESE SAME FOUR LEFT BITS WILL ACTIVATE INDICATOR 02

PROVIDED THE CHARACTER IS FOUND IN POSITION 9 OF THE INPUT RECORD. IN THIS CASE, CHARACTERS WITH EQUAL ZONES ARE A, B, C, D, E, F, G, H, AND I.

A 'D' FOUND IN 26 WILL ACTIVATE THE DESIGNATED INDICATOR IF THE DIGIT PORTION OF THE INPUT CHARACTER (THE FOUR RIGHT BITS OF THE EBCDIC BIT STRING) IS IDENTICAL TO THE DIGIT PORTION OF THE CONTROL CHARACTER.

EXAMPLE (19 - 27):

```
: 0: 3:  :  :  : 5:  : D: D:
:19:20:21:22:23:24:25:26:27:      COLUMN NUMBER
```

MEANING: INDICATOR 03 WILL BE TURNED ON IF THE DIGIT PORTION OF THE CHARACTER FOUND IN POSITION 5 OF THE INPUT RECORD MATCHES THE DIGIT PORTION OF THE CHARACTER 'D'. REFERRING TO THE TABLE OF EQUAL DIGIT GROUPINGS, THE DIGIT PORTION OF THE EBCDIC BIT STRING OF THE CHARACTER D IS 0100. THE CHARACTERS THAT WILL CAUSE THIS CHARACTER TO BE TURNED ON ARE D, M, U, AND 4.

NOTES ON ZONE OPERATIONS

A. UNLIKE A SYSTEM THAT EMPLOYS THE EBCDIC CHARACTER SET RATHER THAN THE ASCII CHARACTER SET, UTILIZING ANY ZONE OR DIGIT FUNCTIONS UNDER PRIME RPG WILL RESULT IN NEITHER A TIME NOR A SPACE SAVING. THE REASON FOR THIS IS THAT IN ORDER TO PROVIDE THIS CAPABILITY, THE CHARACTERS MUST BE TRANSLATED FROM ASCII FORM TO EBCDIC EQUIVALENT.

B. BECAUSE THERE IS A GREATER VARIETY OF ASCII CHARACTERS THAN EBCDIC CHARACTERS, ANY ASCII CHARACTER THAT DOES NOT HAVE AN EBCDIC EQUIVALENT IS GIVEN THE VALUE OF AN EBCDIC BLANK CHARACTER.

TABLE OF EQUAL DIGITS

CHARACTER	ASCII	EBCDIC BIT STRING	HEXIDECIMAL EQUIVALENT	EBCDIC BIT STRING ASCII VALUE
(SP)	:240	01000000	40	:100
&	:246	01010000	50	:120
-	:255	01100000	60	:140
=	:375	11010000	D0	:320
0	:260	11110000	F0	:360
A	:301	11000001	C1	:301
J	:312	11010001	D1	:321
1	:261	11110001	F1	:361
/	:257	01100001	61	:141
B	:302	11000010	C2	:302
K	:313	11010010	D2	:322
S	:323	11100010	E2	:342
2	:262	11110010	F2	:362
C	:303	11000011	C3	:303
L	:314	11010011	D3	:323
T	:324	11100011	E3	:343
3	:263	11110011	F3	:363
D	:304	11000100	C4	:304
M	:315	11010100	D4	:324
U	:325	11100100	E4	:344
4	:264	11110100	F4	:364
E	:305	11000101	C5	:305
N	:316	11010101	D5	:325
V	:326	11100101	E5	:345
5	:265	11110101	F5	:365
F	:306	11000110	C6	:306
O	:317	11010110	D6	:326
W	:327	11100110	E6	:346
6	:266	11110110	F6	:366
G	:307	11000111	C7	:307
P	:320	11010111	D7	:327
X	:330	11100111	E7	:347
7	:267	11110111	F7	:367

CHARACTER:	ASCII:	EBCDIC BIT STRING:	HEXIDECIMAL: EQUIVALENT	EBCDIC BIT STRING: ASCII VALUE
H	:310	11001000	C8	:310
Q	:321	11011000	D8	:330
Y	:331	11101000	E8	:350
8	:270	11111000	F8	:370
I	:311	11001001	C9	:311
R	:322	11011001	D9	:331
Z	:332	11101001	E9	:351
9	:271	11111001	F9	:371
(CENT)	:336	01001010	4A	:112
!	:241	01011010	5A	:132
:	:272	01111010	7A	:172
.	:256	01001011	4B	:132
$	:244	01011011	5B	:133
,	:254	01101011	6B	:153
#	:243	01111011	7B	:173
<	:274	01001100	4C	:114
*	:252	01011100	5C	:134
%	:245	01101100	6C	:154
@	:300	01111100	7C	:174
(	:250	01001101	4D	:115
)	:251	01011101	5D	:135
	:337	01101101	6D	:155
'	:247	01111101	7D	:175
+	:253	01001110	4E	:116
;	:273	01011110	5E	:136
>	:276	01101110	6E	:156
=	:275	01111110	7E	:176
<	:374	01001111	4F	:117
(BAR)<	:XXX	____1111	5F	:137
?	:277	01101111	6F	:157
"	:242	01111111	7F	:177
===	===	===	===	===

TABLE OF EQUAL ZONES

CHARACTER	ASCII	EBCDIC BIT STRING	HEXIDECIMAL EQUIVALENT	EBCDIC BIT STRING ASCII VALUE
A	:301	11000001	C1	:301
B	:302	11000010	C2	:302
C	:303	11000011	C3	:303
D	:304	11000100	C4	:304
E	:305	11000101	C5	:305
F	:306	11000110	C6	:306
G	:307	11000111	C7	:307
H	:310	11001000	C8	:310
I	:311	11001001	C9	:311
=	:375	11010000	D0	:375
J	:312	11010001	D1	:312
K	:313	11010010	D2	:322
L	:314	11010011	D3	:323
M	:315	11010100	D4	:324
N	:316	11010101	D5	:325
O	:317	11010110	D6	:326
P	:320	11010111	D7	:327
Q	:321	11011000	D8	:330
R	:322	11011001	D9	:331
S	:323	11100010	E2	:342
T	:324	11100011	E3	:343
U	:325	11100100	E4	:344
V	:326	11100101	E5	:345
W	:327	11100110	E6	:346
X	:330	11100111	E7	:347
Y	:331	11101000	E8	:350
Z	:332	11101001	E9	:351

CHARACTER:	ASCII:	EBCDIC BIT STRING:	HEXIDECIMAL: EQUIVALENT	EBCDIC BIT STRING: ASCII VALUE
0	:260	11110000	F0	:360
1	:261	11110001	F1	:361
2	:262	11110010	F2	:362
3	:263	11110011	F3	:363
4	:264	11110100	F4	:364
5	:265	11110101	F5	:365
6	:266	11110110	F6	:366
7	:267	11110111	F7	:367
8	:270	11111000	F8	:368
9	:271	11111001	F9	:371
(SP)	:240	01000000	40	:100
(CENT)	:336	01001010	4A	:112
.	:256	01001011	4B	:113
<	:274	01001100	4C	:114
(	:250	01001101	4D	:115
+	:253	01001110	4E	:116
<	:374	01001111	4F	:117
&	:246	01010000	50	:120
!	:241	01011010	5A	:132
$	:244	01011011	5B	:133
*	:252	01011100	5C	:134
)	:251	01011101	5D	:135
;	:273	01011110	5E	:136
(BAR<)	:XXX	01011111	5F	:137
-	:255	01100000	60	:140
/	:257	01100001	61	:141
,	:254	01101011	6B	:153
%	:245	01101100	6C	:154
_	:337	01101101	6D	:155
>	:276	01101110	6E	:156
?	:277	01101111	6F	:157
:	:272	01111010	7A	:172
#	:243	01111011	7B	:173
@	:300	01111100	7C	:174
'	:247	01111101	7D	:175
=	:275	01111110	7E	:176
"	:242	01111111	7F	:177
===	===	===	===	===

2. CALCULATION SPECIFICATIONS

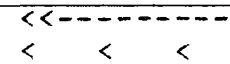
OPERATIONS INVOLVED:

1. MHHZO
2. MHLZO
3. MLHZO
4. MLLZO
5. TESTZ

A. RESTRICTIONS

1. MHHZO - FACTOR 2 AND RESULT FIELD MUST BE ALPHA.
2. MHLZO - FACTOR 2 MUST BE ALPHA.
3. MLHZO - RESULT FIELD MUST BE ALPHA.
4. MLLZO - EITHER FIELD MAY BE EITHER ALPHA OR NUMERIC.
5. FOR MHHZO, MHLZO, MLHZO, AND MLLZO THE RESULTING INDICATORS MUST BE LEFT BLANK.
6. TESTZ - FACTOR 1 AND FACTOR 2 MUST BE BLANK; AT LEAST ONE RESULTING INDICATOR MUST BE SPECIFIED.

B. METHOD OF OPERATION



		FACTOR2	RESULT	FUNCTION
1.	MHHZO	ALPHA	ALPHA	BITS 0-3 OF THE LEFTMOST BYTE OF FACTOR2 ARE MOVED TO BITS 0-3 OF THE LEFTMOST BYTE OF THE RESULT.
2.	MHLZO	ALPHA	NUMERIC	BITS 0-3 OF THE LEFTMOST BYTE OF FACTOR2 ARE MOVED TO BITS 4-7 OF THE RIGHTMOST BYTE OF THE RESULT.
		ALPHA	ALPHA	BITS 0-3 OF THE LEFTMOST BYTE OF FACTOR2 ARE MOVED TO BITS 0-3 OF THE RIGHTMOST BYTE OF THE RESULT FIELD.
3.	MLHZO	ALPHA	ALPHA	BITS 0-3 OF THE RIGHTMOST BYTE OF FACTOR2 ARE MOVED

				TO BITS 0-3 OF THE LEFTMOST BYTE OF THE RESULT FIELD.
		NUMERIC	ALPHA	BITS 4-7 OF THE RIGHTMOST BYTE OF FACTOR2 ARE MOVED TO BITS 0-3 OF THE LEFTMOST BYTE OF THE RESULT FIELD.
4.	MLLZO	ALPHA	ALPHA	BITS 0-3 OF THE RIGHTMOST BYTE OF FACTOR2 ARE MOVED TO BITS 0-3 OF THE RIGHTMOST BYTE OF THE RESULT FIELD.
		ALPHA	NUMERIC	BITS 0-3 OF THE RIGHTMOST BYTE OF FACTOR2 ARE MOVED TO BITS 4-7 OF THE RIGHTMOST BYTE OF THE RESULT FIELD.
		NUMERIC	ALPHA	BITS 4-7 OF THE RIGHTMOST BYTE OF FACTOR2 ARE MOVED TO BITS 0-3 OF THE RIGHTMOST BYTE OF THE RESULT FIELD.
		NUMERIC	NUMERIC	BITS 4-7 OF THE RIGHTMOST BYTE OF FACTOR2 ARE MOVED TO BITS 4-7 OF THE RIGHTMOST BYTE OF THE RESULT FIELD.

IN SUMMARY, IF THE FIELD IS ALPHA, BITS 0-3 ARE USED AND IF THE FIELD IS NUMERIC, BITS 4-7 ARE USED.

5. THE TESTZ OPERATION EXAMINES THE LEFTMOST CHARACTER IN THE RESULT FIELD AND TURNS ON THE HI INDICATOR IF THE CHARACTER IS AMPERSAND, OR A-I; TURNS ON THE LO INDICATOR IF THE CHARACTER IS =, -, OR J-R; OR TURNS ON EQ IF THE CHARACTER IS ANYTHING ELSE.

B. OVERPRINT

THE OVERPRINT OPTION HAS BEEN ADDED TO RPG FOR REV17 FOR PRINTER FILES. THE OVERPRINT OPTION APPLIES ONLY TO FILES WHOSE DEVICE HAS BEEN SPECIFIED AS 'PRINTER' IN COLUMNS 40-47 OF THE FILE SPECIFICATION. THE OVERPRINT WILL NOT FUNCTION FOR FILES SPECIFIED AS 'DISK' IN COLUMN 40-47 OF FILE SPECIFICATIONS, EVEN IF SPOOLING WITH FORTRAN FORMAT CONTROL IS DESIGNATED.

C. EDIT FORMAT DECLARATION IN THE HEADER SPECIFICATION LINE

THE FUNCTIONALITY OF OTHER THAN UNITED STATES EDIT CODE FORMAT HAS BEEN ADDED FOR REV17. USING THE HEADER SPECIFICATION LINE, COLUMN 21, THE FOLLOWING WILL BE GENERATED FOR OUTPUT EDIT CODES:

COLUMN 21 OF H-SPEC	EDIT CODES 3, 4, C, D, L, M	EDIT CODES 1, 2, A, B, J, K	UDATE Y EDIT CODE	Z EDIT (WHERE ALL ZEROES TO LEFT OF DECIMAL)
BLANK	NNNN.NN	N,NNN.NN	MM/DD/YY	.NN
D	NNNN.NN	N,NNN.NN	DD/MM/YY	.NN
I	NNNN,NN	N.NNN,NN	DD.MM.YY	,NN
J	NNNN,NN	N.NNN,NN	DD.MM.YY	,NN

NOTE: THE INPUT FOR UDATE COMES FROM THE SYSTEM DATE AND INVERTED FOR RPG PURPOSES IF REQUESTED BY COLUMN 21 OF THE HEADER SPECIFICATION OF THE RPG PROGRAM.

D. COMINPUT FILENUMBER CONFLICT

RPG RUNTIME NO LONGER ASSIGNS UNIT 6 AS AN RPG FILE.

E. BIT OPERATIONS

ALL OF THE FOLLOWING ENHANCEMENTS ARE UTILIZED IN THE CALCULATION PORTION OF AN RPG PROGRAM.

1. BITON

A. DEFINITION: THE BITON OPERATION IS USED IN THE CALCULATION SECTION OF AN RPG PROGRAM AND CAUSES BITS IDENTIFIED IN FACTOR2 TO TURN ON IN A FIELD NAMED IN THE RESULT FIELD.

B. CONDITIONS AND LIMITATIONS:

I. FACTOR1 MUST BE BLANK.

II. RESULTING INDICATORS MUST BE BLANK.

III. FIELD SIZE OF FACTOR2 AND RESULT MUST BE 1 BYTE.
IV. FACTOR2 MAY BE EITHER A VARIABLE OR A LITERAL.

2. BITOF

A. DEFINITION: THE BITOF OPERATION CAUSES THE BITS IDENTIFIED IN FACTOR2 TO TURN OFF IN A FIELD NAMED IN THE RESULT FIELD.

B. CONDITIONS AND LIMITATIONS:

I. FACTOR1 MUST BE BLANK.
II. RESULTING INDICATORS MUST BE BLANK.
III. FIELD SIZE OF FACTOR2 AND RESULT MUST BE 1 BYTE.
IV. FACTOR2 MAY BE EITHER A VARIABLE OR A LITERAL.

3. TESTB

A. DEFINITION: TESTB OPERATION CAUSES BITS IDENTIFIED IN FACTOR2 TO BE TESTED FOR AN ON OR OFF CONDITION IN THE FIELD NAMED AS THE RESULT FIELD. THE INDICATOR IN THE HI COLUMN OF THE RESULTING INDICATORS IS TURNED ON IF EACH BIT SPECIFIED IN FACTOR2 IS OFF IN THE RESULT FIELD. AN INDICATOR IN LO IS TURNED ON IF TWO OR MORE BITS WERE TESTED AND FOUND TO BE OF MIXED STATUS. AN INDICATOR IN THE EQ COLUMN IS TURNED ON IF EACH BIT SPECIFIED IN FACTOR2 IS ON IN THE RESULT FIELD.

B. CONDITIONS AND LIMITATIONS:

I. FACTOR1 MUST BE BLANK.
II. FIELD SIZE OF FACTOR2 AND RESULT MUST BE 1 BYTE.
III. FACTOR2 MAY BE EITHER A VARIABLE OR A LITERAL.
IV. AT LEAST ONE RESULTING INDICATOR MUST BE USED.
V. TWO RESULTING INDICATORS MAY BE THE SAME BUT ALL THREE RESULTING INDICATORS CANNOT BE THE SAME.
VI. IF THE FIELD SPECIFIED IN FACTOR2 CONTAINS BITS WHICH ARE ALL OFF, NO RESULTING INDICATOR IS TURNED ON.

F. ERROR MESSAGE ADDITION

MESSAGE: IMPROPER FIELD TYPE FOR MZONE OPERATION.
MEANING: FOR THE REQUESTED MZONE OPERATION, (MHHZO, MHLZO, MLLZO, MLHZO, OR TESTZ) A FIELD TYPE IS EITHER ALPHA AND MUST BE NUMERIC OR IS NUMERIC AND MUST BE ALPHA.
SUGGESTION: CHECK THE FIELD TYPES ON THE MZONE OPERATION AND BE SURE THEY COINCIDE WITH THE ALLOWABLE FIELD TYPES FOR THAT OPERATION.

III. TECHNICAL ACTION REQUESTS RESPONDED TO AND BUGS FIXED

81303 - 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.

25232 AND 25226 - 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.

- MVR DOES NOT FUNCTION PROPERLY WHEN THE NUMBER OF DIGITS IN THE DIVIDEND IS LESS THAN THE NUMBER OF DIGITS IN THE DIVISOR.

25234 AND 81301 - 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.

25231 - FIELDS WITH A LENGTH OF TWO OR LESS ARE NOT EDITED IN PROPER FORMAT ACCORDING TO THE EDIT CODE SPECIFIED.

15185 AND 25224 - EDIT CODES NOT FUNCTIONING PROPERLY.

25225 - LOKUP OPERATION ONLY WORKS PROPERLY WHEN THE TABLE IS IN ASCENDING SEQUENCE.

25227 - AN UNSUCCESSFUL CHAIN OPERATION CAUSES A 'KIDA 07 ERROR MESSAGE TO BE DISPLAYED ON THE USER TERMINAL.

15526 - NO STATEMENT TYPE SEQUENCING IS DONE DURING COMPILATION.

25233 - ARITHMETIC OPERATIONS TRUNCATE SIGNIFICANT DIGITS DURING COMPUTATION.

25228 - DIVISION BY ZERO SETS RESULTING FIELD EQUAL TO DIVIDEND RATHER THAN TO ZERO.

- CHAIN OPERATION RETRIEVES NEXT RECORD IN SEQUENCE IF THE CHAIN WERE UNSUCCESSFUL.

25230 - SETLL POSITIONS THE DEMAND FILE TO END OF FILE RATHER THAN NEXT HIGHER KEY IF AN EXACT MATCH WERE NOT FOUND ON THE KEY.

15183 - A RECORD WITH NO FIELDS DEFINED TAKES THE FIELDS FOR THE NEXT RECORD AND DROPS THE RECORD IDENTIFIER APPLICABLE TO THE FIELDS IT TAKES.

- WHEN MATCHING RECORDS ARE SPECIFIED FOR PACKED DATA FIELDS, RANDOM OUT-OF-SEQUENCE ERRORS OCCUR.

20404 - WHEN AN "ILLEGAL SUBSCRIPT PRESS S AND <CR> TO CONTINUE" ERROR MESSAGE IS DISPLAYED, THERE IS NO PAUSE.

ARRAY LOOK-UP -
A. RETURNS SUBSCRIPT 1 GREATER THAN THE RESULT SHOULD BE IF THE ARRAY IS A SINGLE ELEMENT ARRAY.
B. FOR ALL ARRAY LOOK-UP, AFTER THE FIRST LOOK-UP OPERATION HAS BEEN PERFORMED, THE FIRST WORD OF THE ARRAY IS BEING WRITTEN OVER.

80579 - FLOATING DOLLAR SIGN COMBINED WITH ZERO SUPPRESS IN AN

EDIT WORD IS MALFUNCTIONING BY APPENDING A ZERO AND LEAVING OUT THE $.

81422 - SUBSEQUENT TO AN OUTPUT FILE SPECIFICATION LINE, THE NEXT SEQUENCE OF DATA AND LITERALS ARE PRINTED WHEN ONLY A BLANK LINE SHOULD BE GENERATED.

15187 - ALL 'OR' LINES TAKE THE QUOTA OF SKIP/SPACING FROM THE FIRST PRECEEDING NON-'OR' LINE. ALSO, FETCH OVERFLOW IS NOT ALLOWED FOR 'OR' LINES.

13541 - RPG COMPILER DOES NOT ALLOW SPECIAL CHARACTERS #,$, AND @. NOTES AND WARNINGS @ IS NOT A VALID PRIME FILENAME CHARACTER. A SEG FILENAME IS TYPICALLY STARTED WITH #.

20167 AND 15386 - ARRAY LOOKUP FAILS WHEN THE LOOKUP ARRAY IS NUMERIC AND THE ELEMENT SIZE IS DIFFERENT FROM THE RESULTING TABLE SIZE.

20178 - OUTPUT BINARY FIELDS ARE WRITTEN OFFSET ONE CHARACTER TO THE RIGHT OF THE POSITION NOMINATED.

22443 - ARGUMENT RANGE ERROR OCCURS IN LARGE RPG PROGRAMS WHICH UTILIZE THE MVR OPERATION.

11416 - WHEN THE END POSITION IN THE OUTPUT FILE IS LESS THAN THE LENGTH OF THE FIELD TO BE OUTPUT, THE PROGRAM HALTS ABRUPTLY.

20161 - CONTROL BREAK ON PACKED NUMERIC FIELDS IS NOT ALWAYS PROPERLY DETECTED.

25756 - ALTHOUGH RPG DOES NOT OPEN FILE UNIT 6 (COMINPUT FILE NUMBER), UPON PROGRAM COMPLETION, RPG CLOSES LUN 6.

```
**********************************************************************************************
**********************************************************************************************
**
**
**
**      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      Y
**   J   J      I     M   M   M   M      Y
**    JJ       III    M   M   M   M      Y
**
**
**
**    III   N   N  FFFFF   000          PPPP   EEEEE  RRRR   FFFFF          1     7
**     I    NN  N  F      0   0         P   P  E      R   R  F             11
**     I    N N N  F      0   0         P   P  E      R   R  F              1
**     I    N N N  FFFF   0   0         PPPP   EEEE   RRRR   FFFF           1
**     I    N N N  F      0   0         P      E      R R    F              1
**     I    N  NN  F      0   0   ..    P      E      R  R   F        ..    1     7
**    III   N   N  F       000    ..    P      EEEEE  R   R  F        ..  111     7
**
**
**
**
**********************************************************************************************
**********************************************************************************************
```

DATE: 15 MAR 1979

TO: R & D PERSONNEL

FROM: LARRY STABILE

SUBJECT: BASIC/VM PERFORMANCE MEASUREMENT PACKAGE

ABSTRACT

A NEW COMMAND HAS BEEN IMPLEMENTED IN BASIC/VM WHICH ALLOWS USERS TO OBTAIN DATA, AT THE BASIC COMMAND LEVEL, REGARDING THE PERFORMANCE OF THEIR PROGRAMS. THE DATA IS PRESENTED IN THE FORM OF EITHER A TABLE OR A HISTOGRAM.

THE PURPOSE OF THIS PACKAGE IS TWOFOLD:

(A) TO AID THE IMPLEMENTORS OF THE COMPILER IN BEST OPTIMIZING BASIC ITSELF.

(B) TO PERMIT USERS TO OPTIMIZE THEIR OWN BASIC CODE.

THE PACKAGE IS GENERALLY AVAILABLE AT REV 17.

THIS REV OF PE-T-554 INCLUDES MINOR STYLISTIC CHANGES AND A SINGLE FUNCTIONAL CHANGE: THE 'TOT' KEYWORD HAS BEEN CHANGED TO 'TTL'.

DESCRIPTION

```
********
* PERF *
********
```

```
PERF ON
PERF OFF

PERF TABLE [SNLO [- SNHI ]]

PERF HIST [SCREEN_SIZE] <[CNT] [AVG] [TTL]< [SNLO [- SNHI ]]
```

BEFORE COMPILING A PROGRAM TO BE MEASURED, THE PERF ON COMMAND IS GIVEN. THIS INFORMS THE SYSTEM THAT MEASUREMENT CALLS ARE TO BE INSERTED IN THE COMPILED CODE. THE PROGRAM IS THEN EXECUTED, THEN A FURTHER PERF COMMAND IS INVOKED TO PRINT TO THE USER THE PERFORMANCE DATA DESIRED. THE TABLE OPTION PRODUCES OUTPUT LIKE THIS:

SN	CNT	AVG	DEV	TTL	SQSUM
100	1	0.00	0.00	0	0
110	1	1.00	0.00	1	1
120	1	2.00	0.00	2	4
130	1	0.00	0.00	0	0

THE COLUMNS REPRESENT THE STATEMENT NUMBER, STATEMENT COUNT, AVERAGE STATEMENT EXECUTION TIME, STANDARD DEVIATION OF EXECUTION TIME (FOR ASSESSING THE DATA- OR TIMESHARING-DEPENDENCE OF A STATEMENT'S EXECUTION TIME), THE TOTAL RUNNING TIME OF THE STATEMENT, AND THE TOTAL SQUARED-SUM OF THE RUNNING TIME OF THE STATEMENT (ALL TIMES ARE IN TICKS).

THE HIST OPTION OUTPUTS SIMILAR INFORMATION IN HISTOGRAM FORMAT. THE DATA IS SCALED IN TERMS OF THE SCREEN SIZE, THE DEFAULT OF WHICH IS THE CURRENT BASIC MARGIN.

UP TO THREE GRAPHS CAN BE DISPLAYED AT ONCE, WITH THE SCREEN EVENLY PARTITIONED INTO SPACE FOR EACH GRAPH (SEE EXAMPLE).

THE REMAINING KEYWORDS TO THE HIST OPTION SELECT THE DATA TO BE GRAPHED: CNT - EXECUTION COUNT ('*'[GRAPH-CHARACTER]); AVG - AVERAGE TIME ('.'); TTL - RUNNING-TIME TOTAL ('+').

EXAMPLE

```
OK, BASICV

BASICV REV17.0
NEW OR OLD: OLD TEST
>LIST
TEST       FRI, MAR 02 1979          11:25:35


10 X = 0
20 X = X + 1 FOR I = 1 TO 50
30 PRINT X
40 FOR I = 1 TO 10
50   X = X + SIN(I)
60 NEXT I
70 PRINT X,X**2,I,I**2
80 END
>PERF ON
>RUN
TEST       FRI, MAR 02 1979          11:25:45


50
51.41118837122          2643.110289741          11                    121
>MARGIN 70
>PERF TABLE
     SN          CNT          AVG          DEV          TTL          SQSUM
     10            1         0.00         0.00            0              0
     20            1         1.00         0.00            1              1
     30            1         4.00         0.00            4             16
     40            1         0.00         0.00            0              0
     50           10         0.20         0.40            2              2
     60           10         0.10         0.30            1              1
     70            1        11.00         0.00           11            121
     80            1         2.00         0.00            2              4
>PERF HIST CNT AVG TTL
COUNT   AVERAGE   TOTAL :
   10   ***                  .                    +
   20   ***                  ..                   ++
   30   ***                  ........             ++++++++
   40   ***                  .                    +
   50   **********.....      .                    ++++
   60   **********.....      .                    ++
   70   ***                  ....................+++++++++++++++++++
   80   ***                  ....                 ++++
```

```
>PERF HIST CNT AVG
COUNT  AVERAGE  :
   10  ****                                .
   20  ****                                ...
   30  ****                                ...........
   40  ****                                .
   50  ********************************.
   60  ********************************.
   70  ****                                ................................
   80  ****                                ......
>PERF HIST CNT TTL 40-70
COUNT  TOTAL :
   40  ****                                +
   50  ************************************++++
   60  ********************************++
   70  ****                                ++++++++++++++++++++++++++++++++
>MAEHRGIN 30
>PERF HIST CNT AVG TTL
COUNT  AVERAGE  TOTAL :
   10  *       .       +
   20  *       .       +
   30  *       ...     +++
   40  *       .       +
   50  *******.       ++
   60  *******.       +
   70  *       .......+++++++
   80  *       ..      ++
>PERF HIST SS AVG CNT
COUNT  AVERAGE  :
   10  ***                             .
   20  ***                             ...
   30  ***                             ..........
   40  ***                             .
   50  ****************************.
   60  ****************************.
   70  ***                             ..............................
   80  ***                             .....
>QUIT

OK,
```

NOTES ON USAGE

COMPILE (HENCE RUN) CLEARS THE PERFORMANCE MEASUREMENT STATISTICS AND STARTS THE MEASUREMENT AFRESH. EXECUTE, HOWEVER, DOES NOT, AND THUS PROVIDES A WAY OF CONTINUING TO GATHER DATA AFTER OBSERVING SOME INTERMEDIATE RESULTS. THE SAME IS TRUE OF CONTINUE, SO THAT THESE DATA MAY BE VIEWED DURING A BREAK OR PAUSE.

NOTE THAT THE TICK (TYPICALLY 3.03 MSEC.) IS THE SMALLEST POSSIBLE UNIT OF TIME MEASUREMENT. AVERAGE STATEMENT TIMES NEAR 1 TICK WILL PROBABLY HAVE HIGH STANDARD DEVIATIONS; THIS IS UNAVOIDABLE, AND SIMPLY MEANS THAT ONE IS DOWN AS CLOSE AS POSSIBLE TO THE SMALLEST MEASUREMENT GRANULARITY. AVERAGE-TIME MEASUREMENTS SHOULD BE GIVEN CREDIBILITY BASED ON THE STANDARD-DEVIATION VALUE.

THE STATEMENT COUNT CAN BE EFFECTIVELY USED IN PROGRAM TESTING TO SPOT 'DEAD REGIONS'. SIMPLY EXAMINE THE CNT COLUMN IN THE TABLE FOR STATEMENTS WITH A COUNT OF ZERO; THIS MEANS THAT THEY WERE NEVER TESTED.

COMPILE-ING A PROGRAM WITH A GIVEN BINARY-FILE NAME WILL RESULT IN AN ERROR MESSAGE IF PEPF IS ON. THE MEASUREMENT DATA IS NOT PERMITTED IN A BINARY PROGRAM FILE.

```
**********************************************************************
**********************************************************************
*
*
*
*      JJJ    III    M    M   M    M   Y     Y
*       J      I     MM  MM   MM  MM   Y     Y
*       J      I     M M M    M M M     Y Y
*       J      I     M M M    M M M      Y
*    J  J      I     M   M    M   M      Y
*    J  J      I     M   M    M   M      Y
*     JJ      III    M   M    M   M      Y
*
*
*
*    RRRR    PPPP    RRRR     III    M   M   000     SSS
*    R   R   P   P   R   R     I     MM MM  0   0   S   S
*    R   R   P   P   R   R     I     M M M  0   0   S
*    RRRR    PPPP    RRRR      I     M M M  0   0    SSS
*    R R     P       R R       I     M   M  0   0       S
*    R  R    P       R  R      I     M   M  0   0   S   S
*    R   R   P       R   R    III    M   M   000     SSS
*
*
*
**********************************************************************
**********************************************************************
```

ABSTRACT

REVISION 17 OF PRIMOS HAS SEVERAL NEW FEATURES AND EXTENSIONS. NEW FEATURES INCLUDE CARD PROCESSOR SUPPORT, A NEW COMMAND ENVIRONMENT, A CONDITION MECHANISM, AND MANY NEW DIRECT ENTRANCE CALLS. ENCLOSED IN THIS DOCUMENT IS A LIST OF PROBLEMS FIXED BY THIS RELEASE. THIS DOCUMENT DESCRIBES THESE AND OTHER TOPICS RELATED TO REVISION 17 OF PRIMOS.

1 CONFIGURATION AND OPERATIONAL MODIFICATIONS

1.1 RUNNING PRIMOS

THE PRIMOS SYSTEM IS NOW COLD STARTED BY ATTACHING TO PRIRUN AND ISSUING THE COMMAND "R PRIMOS". THE DEFAULT SYSTEM TERMINAL SPEED HAS BEEN CHANGED FROM 110 TO 300 BPS.

IMPORTANT NOTE: AT REVISION 17 OF PRIMOS, THE FORTRAN, MIDAS, COBOL, AND FORMS LIBRARIES, ED (IN CMDNCO), AND THE UII PACKAGE ARE ASSUMED TO BE SHARED.

REVISION 17.1 OF PRIMOS REQUIRES THE SUPPORT OF A NEW VERSION OF DOS. THIS VERSION IS *DOS64 REV. 16.8. PRIMOS REV. 17.1 WILL ONLY RUN WITH THE ABOVE VERSION OF DOS. IT WILL FAIL WITH PREVIOUS VERSIONS OF DOS. (REF: SECTION 2.17.4 NEW DOS SUPPORT).

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, IF PRESENT, IN CMDNCO OF THE COMMAND DEVICE IS USED TO BRING UP REVISION 17 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 INCLUDED IN THE UFD PRIRUN. ONCE THE CHANGES HAVE BEEN MADE TO C_PRMO.TEMPLATE, SIMPLY FUTIL IT TO CMDNCO OF THE COMMAND DEVICE 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>S2050 2050 700/* SHARE FORTRAN LIBRARY
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 BATCHQ>PH_GO                     /* STARTUP BATCH MONITOR
CHAP -NN 3 143                      /* NN IS USER # OF ABOVE PHANTOM
PROP -START                         /* STARTUP SPOOLER
A CMDNCO
/* SET THE DATE AND TIME **********
CO TTY
```

1.2 BUILDING PRIMOS

THE BASIC PROCEDURES FOR BUILDING PRIMOS 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 FILES: C_R3LOAD FOLLOWED BY 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.

RUNNING C_LOAD WILL RESULT IN THE CREATION OF NEW PRXXXX FILES -- PR0013, PR0015, AND PR6002.

1.3 SINGLE VERSION PRIMOS

REF: SECTION 6 CONFIGURATION DIRECTIVES

PRIMOS OPERATING SYSTEM IS DELIVERED IN ONLY A SINGLE VERSION STARTING AT REV. 16. IT CAN BE CONFIGURED AT COLD-START TO RUN BETWEEN 1 AND 64 USERS. EACH USER MAY BE CONFIGURED TO HAVE A NON-SHARED (DTAR2) VIRTUAL ADDRESS SPACE OF UP TO 32 M-BYTES.

(256 SEGMENTS) A LIMIT OF 64 M-BYTES (511 SEGMENTS) OF VIRTUAL ADDRESS SPACE IS AVAILABLE TO ALL USERS. CONFIG DIRECTIVES ARE USED TO SPECIFY THE NUMBER OF USERS TO BE CONFIGURED, THE NUMBER OF NON-SHARED SEGMENTS IN EACH USER'S WORKING AREA, AND THE TOTAL NUMBER OF SEGMENTS AVAILABLE FOR USE IN THE SYSTEM. (NOTE THAT THE NUMBERS USED HERE ARE DECIMAL AND THE NUMBERS SPECIFIED ON ALL CONFIG DIRECTIVES ARE OCTAL NUMBERS)

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 NTUSR, NPUSR, AND NRUSR WHICH IS NUSR, MUST NOT BE GREATER THAN 64 NOR LESS THAN 1. (USER 1, WHICH IS THE SYSTEM USER, MUST ALWAYS BE IN THE SYSTEM)

THE NUMBER OF SEGMENTS AVAILABLE IN EACH USER'S NON-SHARED (DTAR2) ADDRESS SPACE IS SPECIFIED BY THE CONFIG DIRECTIVE NUSEG. IT SETS THE SIZE OF EACH USER'S DESCRIPTOR TABLE FOR DTAR2. THE MAXIMUM NUMBER NUSEG MAY HAVE IS 256. THE SYSTEM HAS SPACE FOR A MAXIMUM OF 4096 DTAR2 SDWS FOR ALL USERS. THEREFORE, THE PRODUCT NUSR*NUSEG MUST NOT EXCEED 4096.

THE CONFIG DIRECTIVE NSEG SPECIFIES THE TOTAL NUMBER OF SEGMENTS AVAILABLE FOR USE THROUGHOUT THE SYSTEM. A NUMBER OF SEGMENTS (CURRENTLY 17) ARE USED FOR THE OPERATING SYSTEM, AND EACH CONFIGURED USER USES ONE SEGMENT FOR THE RING 0 STACK. THEREFORE, NSEG MUST NOT BE LESS THAN THE SUM OF THE SEGMENTS USED BY PRIMOS AND (NUSR-1). (THE RING 0 STACK SEGMENT FOR USER 1 IS INCLUDED IN THE SEGMENTS USED BY PRIMOS.) THE SYSTEM HAS SPACE FOR 511 PAGE MAPS. THEREFORE, NSEG MUST NOT BE GREATER THAN 511.

1.4 PAGING SPACE REQUIREMENTS

THE TOTAL NUMBER OF AVAILABLE SEGMENTS (TOTAL VIRTUAL ADDRESS SPACE) OF THE SYSTEM IS LIMITED BY BOTH THE NSEG CONFIG DIRECTIVE AND THE AMOUNT OF AVAILABLE PAGING SPACE ON THE PAGING DEVICE. IT IS THE MINIMUM OF THE:

A. NUMBER OF SEGMENTS THAT FIT IN THE SPECIFIED PAGING SPACE. (AS SPECIFIED BY USING THE <RECORDS> PARAMETER IN THE PAGDEV AND ALTDEV CONFIGURATION DIRECTIVES)

B. NUMBER THAT IS SPECIFIED IN THE NSEG CONFIG DIRECTIVE.

C. DEFAULT NSEG SETTING. (DEFAULT = 192)

THEREFORE, THE NUMBER OF RECORDS REQUIRED FOR PAGING IS GIVEN BY THE FOLLOWING EQUATION:

$$RECORDS = [(S_T - (S_P + (NUSR-1)))*64 + (NUSR-1)*8 + P_{OS}]*[RECORD/PAGE]$$

S_T TOTAL NUMBER OF SEGMENTS AVAILABLE

S_P NUMBER OF SEGMENTS USED BY PRIMOS

NUSR NUMBER OF CONFIGURED USERS

P_{OS} NUMBER OF PAGES USED BY OPERATING SYSTEM SEGMENTS

(SP + (NUSR-1)) IS THE NUMBER OF SEGMENTS USED BY THE OPERATING SYSTEM AND CONFIGURED USERS' RING 0 STACKS. SUBSTRACT THIS NUMBER FROM THE TOTAL NUMBER OF SEGMENTS AVAILABLE (ST) GIVES THE NUMBER OF SEGMENTS AVAILABLE TO BE USED BY USERS. ALTHOUGH EACH CONFIGURED USER USES ONE SEGMENT FOR RING 0 STACK, ONLY 8 RECORDS ARE ACTUALLY USED WITHIN THE SEGMENT. (NUSR-1)*8 GIVES THE NUMBER OF RECORDS USED BY ALL CONFIGURED USERS (EXCEPT USER 1) FOR RING 0 STACKS.

THE NUMBER OF RECORDS PER PAGE IS 1 FOR STORAGE MODULE, 2 FOR FIXED HEAD DISKS AND 3 FOR MOVING HEAD DISKS.

1.5 PRIMOS DIRECTORY ORGANIZATION - PRI400

IN AN ATTEMPT TO MAKE IT EASIER TO PERFORM INSTALLATIONS, UPDATES, AND OTHER MAINTENANCE ACTIVITIES UPON THE LIBRARY DIRECTORIES THAT CONTAIN PRIMOS, THE OPERATING SYSTEMS GROUPS HAVE ADOPTED THE FOLLOWING CONVENTIONS FOR USE IN ALL SUCH DIRECTORIES.

1.5.1 NAMING CONVENTIONS

PROGRAM SOURCE FILES ARE TO BE NAMED "X.LANG", WHERE "X" IS THE NAME OF THE PROCEDURE CONTAINED IN THE FILE (THE FIRST SUCH IF THERE IS MORE THAN ONE), AND "LANG" IS "FTN" FOR FORTRAN, "PMA" FOR ASSEMBLER, AND "PL1" FOR PL/P OR PL/I. IN THIS WAY, IT WILL BE EASY TO ANSWER THE QUESTION, "WHICH SOURCES IN THIS DIRECTORY ARE WRITTEN IN PL/I"?

THE BINARY FILE CORRESPONDING TO A SOURCE FILE "X.LANG" WILL BE NAMED "x.BIN".

THE LISTING FILE (IF ANY) CORRESPONDING TO SOURCE FILE "X.LANG" WILL BE NAMED "x.LIST".

AN INSERT FILE USED BY A SOURCE FILE "X.LANG" WILL BE NAMED

"Y.INS.LANG", WHERE "Y" IS AN APPROPRIATELY DESCRIPTIVE NAME. COMMAND FILES WILL BE NAMED "C_Y", WHERE "Y" IS AN APPROPRIATELY DESCRIPTIVE NAME. MAPS WILL BE NAMED "M_Y", WHERE "Y" IS AN APPROPRIATE NAME.

1.5.2 LIBRARY DIRECTORY HIERARCHY

THE STRUCTURE OF THE HIERARCHY USED TO CONTAIN PRIMOS MODULES IS THE FOLLOWING:

TOP_LEVEL_DIR	CONTAINS COMMAND , SYSTEM IMAGE FILES (*COLDS, PRNNN, ETC.), AND UTILITY PUNFILE FILES (*MAPGEN, ETC.)
KS	CONTAINS KERNEL SOURCE FILES.
FS	CONTAINS FILE SYSTEM SOURCE FILES.
NS	CONTAINS NETWORK SUBSYSTEM SOURCE FILES.
CS	CONTAINS COMMUNICATIONS SUBSYSTEM SOURCES.
R3S	CONTAINS RING THREE OPERATING SYSTEM SOURCES.
SE	CONTAINS THE SOURCES THAT SUPPORT DPTX DEVICES.
PLPLIB	CONTAINS PLP LIBERARY SOURCES THAT ARE USED BY THE OPERATING SYSTEM.
OBJ	CONTAINS ALL THE OBJECT MODULES.

NOTE THAT NO COMMAND FILE SHOULD CONTAIN DEPENDENCIES ON A PARTICULAR NAME OR LOCATION WITHIN THE HIERARCHY OF TOP_LEVEL_DIR.

1.5.3 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.INS.XXX
                WHERE:    (XXX IS FTN, PMA, ...)
```

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-JFD INSERT MUST BE ANY $INSERT FILES REQUIRED.

1.5.4 PRI400>UTILS

THE SOURCES FOR CERTAIN UTILITIES USED BY PRIMOS 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 FOR DEBUGGING PURPOSES. THE COMINPUT FILES FOR GENERATING THESE UTILITIES ARE FOUND IN PR1400.

1.6 REQUIRED HARDWARE UPGRADES

REVISION 17 SUPPORTS OVERLAPPED DISK TRANSFERS IF TWO DISK CONTROLLERS ARE PRESENT. IF THE SYSTEM CONTAINS TWO TYPE 4004 DCU'S (STORAGE MODULE CONTROLLERS), BOTH MUST BE UPDATED TO THE LATEST ECO REVISION.

THE DEVICE ADDRESS OF THE RING NET CONTROLLER (RNC) WAS CHANGED FROM :61 (OCTAL) TO :07. EXISTING RNC CONTROLLERS MUST BE ECO TO THE NEW ADDRESS. IT IS BELIEVED THAT NO CONTROLLERS WITH THE OLD ADDRESS HAVE BEEN SHIPPED TO CUSTOMERS. NOTE THAT THE OLD ADDRESS CONFLICTED WITH THE DEVICE ADDRESS OF THE GPIB (GENERAL PURPOSE INTERFACE BOARD).

1.7 CONFIGURATION AND INSTALLATION 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 IS USED. 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 INITIALIZATION NETCON (WHICH MUST BE IN CMDNCO) OF THE COMMAND DEVICE IS OPENED AND THE INFORMATION PROCESSED.

TO INSTALL NETWORKS WITH REVISION 17 THE FOLLOWING PROCEDURE IS USED.

```
FUTIL                                  FUTIL
>F PRINET>CMDNCO                       >F X.25>CMDNCO
>T CMDNCO <PASSWORD>        OR         >T CMDNCO <PASSWORD>
>C NETCFG                              >C NETCFG
>QU                                    >Q
```

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

THE COLD START CONFIG DIRECTIVES 'SMLC SMLC' AND 'SMLC CNTRLR' MUST BE USED TO CHANGE THE LOGICAL TO PHYSICAL LINE MAPPING OF SYNCHRONOUS LINES. THE 'SMLC DSC' DIRECTIVE IS NOT IMPLEMENTED IN NETWORKS AT THIS REV. USE THE -DSC OPTION TO NETCFG TO SPECIFY DATASET CONTROL OPTIONS.

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 OR X.25>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	SEGMENTED RUN FILE
C_FAM	COMPILE AND LOAD
C_SEG	LOAD FROM BINARIES

TO INSTALL THE FAM, THE FOLLOWING MUST BE DONE:

1) CREATE A TOP LEVEL 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, SEG #FAM
```

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

2) RUN THE FAM AS A PHANTOM:

```
OK, PH FAM>PH_FAM
PHANTOM IS USER ...
OK,
```

TO ENABLE FAM TO COMMUNICATE WITH A PARTICULAR REMOTE NODE, SEE SEPARATE DOCUMENT DESCRIBING NETCFG.

2 NEW FEATURES

2.1 CONDITION MECHANISM

2.1.1 INTRODUCTION TO THE CONDITION MECHANISM

THE NOTION OF THE "CONDITION" COMES FROM THE CORRESPONDING CONCEPT IN THE PL/I LANGUAGE. A CONDITION IS BASICALLY AN UNSCHEDULED SOFTWARE PROCEDURE CALL (OR BLOCK ACTIVATION) WHICH IS BROUGHT ABOUT BY THE OCCURRENCE OF SOME UNUSUAL EVENT IN A PROCESS. EXAMPLES OF THIS ARE HARDWARE-DEFINED FAULTS (SUCH AS ARITHMETIC EXCEPTIONS), DETECTION BY A SUBROUTINE OF A HOPELESS ERROR SITUATION WHICH CANNNOT BE ADEQUATELY DESCRIBED TO THE SUBROUTINE'S CALLER THROUGH AVAILABLE PARAMETERS, AND QUITS FROM THE USER TERMINAL (CONVERTED BY PRIMOS INTO A HARDWARE-DEFINED FAULT).

CONDITIONS ARE PARTICULARLY USEFUL IN TWO BASIC CIRCUMSTANCES: WHEN CAUSED BY AN UNPREDICTABLE HARDWARE FAULT, AND WHEN THE CALL-RETURN FLOW OF CONTROL IS NOT KNOWN BY THE ROUTINE DETECTING THE UNUSUAL HAPPENING.

THE IMPLEMENTATION OF THE CONDITION MECHANISM DESCRIBED HERE IS INTENDED TO SERVE THREE PURPOSES: TO PROVIDE A CONSISTENT AND USEFUL MEANS FOR SYSTEM SOFTWARE TO HANDLE ERROR CONDITIONS AND TO MANAGE A REENTRANT/RECURSIVE COMMAND ENVIRONMENT; TO PROVIDE USER PROGRAMS WITH THE CAPABILITY TO HANDLE ERROR CONDITIONS THAT HERETOFORE FORCED A RETURN TO COMMAND LEVEL; AND TO PROVIDE SUPPORT FOR THE CONDITION MECHANISM OF ANSI PL/I.

2.1.2 ON-UNITS

AN "ON-UNIT" IS A HANDLER FOR A CONDITION, AND MAY EITHER BE A PROCEDURE (AN "ENTRY VARIABLE" IN PL/I TERMINOLOGY), OR A BEGIN-BLOCK. THE LATTER RESULT FROM EXECUTION OF THE PL/I <ON STATEMENT>, WHILE THE FORMER RESULT FROM EXPLICIT INVOCATION OF THE SYSTEM PRIMITIVES MKONU$ AND MKON$F. THE ONLY WAY TO CAUSE CREATION OF AN ON-UNIT IN A NON-PL/I PROGRAM IS TO EXPLICITLY CALL THE SYSTEM PRIMITIVES MKONU$ OR MKON$F. AT VARIOUS TIMES, SYSTEM SOFTWARE WILL CREATE ITS OWN ON-UNITS FOR SYSTEM-DEFINED CONDITIONS.

A PROCEDURE MAY ALSO ACT TO INVALIDATE, OR "REVERT", AN ON-UNIT IT HAD PREVIOUSLY CREATED. IN PL/I, THIS CAN BE DONE BY MEANS OF THE <REVERT STATEMENT>. THE REVERSION OF AN ON-UNIT CAN ALSO BE ACCOMPLISHED BY CALLING THE SYSTEM PRIMITIVES RVONU$ OR RVON$F. NOTE THAT SUCH A REVERSION APPLIES TO THE CURRENT ACTIVATION ONLY; THE ON-UNIT(S) FOR THE SAME CONDITION CREATED BY ACTIVATIONS EARLIER IN THE STACK ARE NOT AFFECTED.

EVERY ON-UNIT IS ASSOCIATED WITH A GIVEN ACTIVATION, WHICH IS SIMPLY THE PARTICULAR INVOCATION OF THE PROCEDURE (OR BEGIN BLOCK) THAT REQUESTED CREATION OF THE ON-UNIT. ASSOCIATED WITH EVERY ON-UNIT IS THE NAME OF THE CONDITION FOR WHICH THE ON-UNIT IS A HANDLER. THESE CONDITION NAMES ARE CHARACTER STRINGS OF UP TO 32 CHARACTERS, AND MAY REPRESENT SYSTEM-DEFINED CONDITIONS IF THE NAME IS ONE OF THOSE RESERVED FOR SYSTEM USE, OR IT MAY BE A USER-DEFINED CONDITION.

THE CONDITION MECHANISM IS ACTIVATED WHENEVER A CONDITION IS RAISED. IN PL/I TERMS, A CONDITION IS RAISED EITHER EXPLICITLY AS THE RESULT OF A <SIGNAL STATEMENT>, OR IMPLICITLY AS THE RESULT OF SOME ERROR HAVING BEEN DETECTED DURING REGULAR COMPUTATION. NOTE THAT IN THIS CONTEXT, "ERROR" IS TO BE TAKEN LOOSELY: "END OF FILE" IS ONE SUCH "ERROR".

A CONDITION MAY BE RAISED BY THE EXECUTION OF A PL/I <SIGNAL STATEMENT>, OR BY AN EXPLICIT CALL TO THE SYSTEM PRIMITIVES SIGNL$ OR SGNL$F. IN SOME CASES, SUCH AS CONVERSION OF HARDWARE FAULTS TO CONDITIONS, THIS CALL IS EXECUTED BY SYSTEM SOFTWARE AND SO IS INVISIBLE TO THE USER: FROM THE USER'S POINT OF VIEW, A HARDWARE FAULT IS A CONDITION SIGNAL.

2.1.3 INVOCATION OF ON-UNITS

WHEN A CONDITION IS RAISED, THE CONDITION MECHANISM MUST FIRST FIND AN ON-UNIT FOR THAT CONDITION. IT DOES THIS BY SEARCHING THE STACK BACKWARD IN TIME, STARTING FROM THE ACTIVATION BELONGING TO THE PRODECURE THAT RAISED THE CONDITION. IF AN ACTIVATION HAS AN ON-UNIT FOR THE SPECIFIC CONDITION NAME THAT WAS RAISED, THAT ON-UNIT IS SELECTED. IF AN ACTIVATION DOES NOT HAVE AN ON-UNIT FOR THE SPECIFIC CONDITION, BUT DOES HAVE AN ON-UNIT FOR THE SPECIAL CONDITION ANY$, THAT ACTIVATION IS SAID TO HAVE A DEFAULT ON-UNIT, AND THE ANY$ ON-UNIT IS SELECTED. SCANNING IS TEMPORARILY SUSPENDED AT THE FIRST ACTIVATION CONTAINING A SELECTABLE ON-UNIT, AND THAT ON-UNIT IS INVOKED.

A SELECTED ON-UNIT IS INVOKED ACCORDING TO THE FOLLOWING SPECIFICATION:

```
DCL ON_UNIT ENTRY (PTR) VARIABLE;

CALL ON_UNIT (PTR_TO_COND_FRAME);
```

THAT IS, ALL ON-UNITS ARE PASSED A SINGLE ARGUMENT WHICH IS A POINTER TO THE STANDARD CONDITION FRAME HEADER THAT DESCRIBES THE CONDITION THAT WAS RAISED. NOTE THAT AN ON-UNIT OPERATES AS IF IT HAD BEEN INVOKED FROM THE ACTIVATION WHICH CREATED IT, SO THAT IF THE ON-UNIT PROCEDURE IS INTERNAL TO THAT ACTIVATION, THE NORMAL PL/I SCOPING RULES FOR AUTOMATIC STORAGE (AND ALL OTHER STORAGE CLASSES) APPLY.

2.1.4 POSSIBLE ACTIONS OF AN ON-UNIT

IN THE GENERAL CASE, AN ON-UNIT HAS SEVERAL OPTIONS AS TO WHAT ACTION IT CAN TAKE. IT MAY, OF COURSE, PERFORM ANY DESIRED APPLICATION-SPECIFIC TASKS, SUCH AS CLOSING FILE UNITS, DELETING TEMPORARY FILES, UPDATING DATABASES, DOING CONSISTENCY CHECKS, AND SO ON IN ORDER TO ABORT THE COMPUTATION THAT HAS BEEN INTERRUPTED. IN MANY CASES, HOWEVER, IT MAY BE POSSIBLE FOR THE ON-UNIT TO REPAIR THE CAUSE OF THE CONDITION (OR EVEN TO DETERMINE THAT THE CONDITION CAN BE SAFELY IGNORED), OR TO DECIDE THAT THE COMPUTATION'S NORMAL FLOW CAN BE INTERRUPTED AND THE PROGRAM REENTERED AT SOME "KNOWN" POINT.

IF PERMITTED BY THE SIGNALLER OF THE CONDITION, THE ON-UNIT MAY SIMPLY RETURN, IN WHICH CASE THE COMPUTATION WILL CONTINUE FROM THE POINT OF THE SIGNAL OR HARDWARE FAULT. IF THE SIGNALLER HAS FORBIDDEN SUCH A RETURN, AN ATTEMPT TO DO IT WILL RESULT IN A SIGNAL OF THE CONDITION ILLEGAL_ONUNIT_RETURN$. THERE ARE INFORMATION BITS IN THE CONDITION FRAME HEADER (SEE BELOW) THAT INFORM THE ON-UNIT AS TO WHETHER RETURN IS PERMITTED (THIS INFORMATION IS IMPLICIT IN THE NAME OF MOST SYSTEM-DEFINED CONDITIONS), AND WHETHER IT WILL IN GENERAL MAKE SENSE TO RETURN WITHOUT HAVING TAKEN "CORRECTIVE ACTION".

AN ON-UNIT MAY PERFORM A NONLOCAL GOTO TO SOME PREVIOUSLY DEFINED LABEL, THAT WILL CAUSE THE PROGRAM TO RESTART IN SOME KNOWN STATE. THIS ACTION MAY ALWAYS BE TAKEN, AS THE ACTIVATION THAT CAUSED THE CONDITION TO BE RAISED WILL USUALLY BE ABORTED BY THE NONLOCAL GOTO, AND HENCE THE ISSUE OF ON-UNIT RETURN DOES NOT ARISE.

AN ON-UNIT MAY ALSO SIGNAL ANOTHER (POSSIBLY THE SAME, BUT BEWARE OF INFINITE RECURSION) CONDITION, AND IT MAY EITHER PERMIT OR DENY RETURN FROM THAT CONDITION.

THE ON-UNIT MAY DECIDE TO GET THE PROCESS TO COMMAND LEVEL, TO ALLOW THE USER TO TAKE CONTROL. THE SYSTEM DEFAULT ON-UNIT IS AN EXAMPLE OF AN ON-UNIT THAT DOES EXACTLY THAT. IF THE USER ENTERS THE "START" COMMAND, THE NEW COMMAND LEVEL WILL RETURN TO THE INVOKING ON-UNIT.

FINALLY, THE ON-UNIT MAY DECIDE THAT IT HAS NOT BEEN ABLE TO HANDLE THE CONDITION AFTER ALL, OR THAT IT HAS ONLY PARTIALLY DONE SO, AND NEEDS HELP FROM AN ON-UNIT ESTABLISHED BY ONE OF ITS DYNAMIC ANCESTORS. THE ON-UNIT INSTRUCTS THE CONDITION MECHANISM THAT IT DESIRES TO "CONTINUE TO SIGNAL", AND THEN SIMPLY RETURNS. THE CONDITION MECHANISM WILL THEN CONTINUE TO SCAN UP THE STACK FOR MORE ON-UNITS FOR THE CONDITION.

2.1.5 USING THE CONDITION MECHANISM FROM FORTRAN

SINCE FORTRAN IS NOT A BLOCK-STRUCTURED LANGUAGE, THE USE OF ON-UNITS (CONDITION HANDLERS) AND OF NONLOCAL GOTO'S FROM FORTRAN IS SOMEWHAT RESTRICTED. IN PARTICULAR, THERE ARE NO INTERNAL PROCEDURES OR BLOCKS IN FORTRAN, LEAVING EXTERNAL PROCEDURES

(SUBROUTINES) AS THE ONLY POSSIBILITIES FOR SERVICE AS ON-UNITS. THE FACT THAT A FORTRAN STATEMENT LABEL VALUE DOES NOT CONTAIN AN ACTIVATION (STACK) POINTER MEANS THAT NONLOCAL GOTO'S WORK IN A WAY DIFFERENT FROM PL/I (SEE BELOW).

TO SUMMARIZE THE RESTRICTIONS:

- FORTRAN ON-UNITS MUST BE SUBROUTINES, WHICH MAY NOT, OF COURSE, BE INTERNAL TO THE SUBROUTINE OR MAIN PROGRAM MAKING THE ON-UNIT.

- NONLOCAL GOTO'S ARE DEFINED IN PRIME FORTRAN TO WORK ONLY IF THE TARGET STATEMENT LABEL BELONGS TO THE CALLER OF THE SUBROUTINE PERFORMING THE NONLOCAL GOTO. THAT IS, NONLOCAL GOTO'S WORK ONLY TO THE PREVIOUS STACK LEVEL.

2.1.5.1 DATATYPE_INCOMPATIBILITIES

THE PL/I INTERFACES TO THE CONDITION MECHANISM UTILIZE THE PL/I DATATYPE "CHARACTER(*) VARYING". THIS DATATYPE IN NOT AVAILABLE IN FORTRAN (EITHER 1966 OR 1977 ANSI STANDARD). THE 1977 ANSI FORTRAN, HOWEVER, INCLUDES A DATATYPE THAT IS THE EQUIVALENT OF PL/I "CHARACTER(*) NONVARYING".

FOR THESE REASONS, A SET OF INTERFACES TO THE CONDITION MECHANISM IS PROVIDED WHICH UTILIZES NONVARYING CHARACTER STRINGS. THE USER IS CAUTIONED THAT THESE INTERFACES WILL NOT BE AS EFFICIENT AS THOSE USING THE VARYING CHARACTER STRINGS. IT IS POSSIBLE TO SIMULATE VARYING CHARACTER STRINGS IN FORTRAN BY USING APPROPRIATE EQUIVALENCES.

THE PRIME REPRESENTATION FOR A VARYING CHARACTER STRING IS EQUIVALENT TO AN INTEGER*2 ARRAY IN WHICH THE FIRST ELEMENT CONTAINS THE CHARACTER COUNT, AND THE REMAINING ELEMENTS CONTAIN THE CHARACTERS IN PACKED FORMAT. TO ILLUSTRATE:

```
PL/I:
  DCL NAME CHAR(5) VARYING STATIC INITIAL ('QUIT$');

FORTRAN:
  INTEGER*2 NAME(4)
  DATA NAME /5, 'QUIT$'/
```

2.1.5.2 INTERFACES_FOR_NONLOCAL_GOTO'S

A FULL-FUNCTION NONLOCAL GOTO REQUIRES THAT THE TARGET LABEL IDENTIFY BOTH A STATEMENT AND AN ACTIVATION (STACK FRAME) OF THE PROGRAM THAT CONTAINS THE STATEMENT. IF SUCH A NONLOCAL GOTO IS REQUIRED IN A FORTRAN PROGRAM, THE FOLLOWING INTERFACES CAN BE USED.

THE PROCEDURE MKLBSF IS CALLED BY A PROGRAM TO CREATE A PL/I-COMPATIBLE LABEL POINTING TO ONE OF ITS STATEMENTS. THE ACTIVATION POINTER IN THE LABEL WILL IDENTIFY THE CALLER'S ACTIVATION.

THE PROCEDURE PL1$NL WILL PERFORM A NONLOCAL GOTO TO A SPECIFIED TARGET LABEL. LABELS PRODUCED BY MKLB$F ARE ACCEPTABLE TO PL1$NL.

THE CALLING SEQUENCES FOR THESE ROUTINES ARE DESCRIBED BELOW.

2.1.6 DEFAULT ON-UNITS AND CLEANUP ON-UNITS

THE CLEANUP$ CONDITION

THE SPECIAL CONDITION CLEANUP$ IS USED ONLY DURING PROCESSING OF A NONLOCAL GOTO (OR A CRAWLOUT FROM AN INNER RING). ANY ACTIVATION MAY MAKE AN ON-UNIT FOR THE CONDITION CLEANUP$, WHICH WILL BE INVOKED ONLY WHEN THAT ACTIVATION IS ABOUT TO BE ABORTED BY A CRAWLOUT OR SIMPLE NONLOCAL GOTO.

THE CLEANUP$ ON-UNITS OF THE ACTIVATIONS ON THE STACK ARE INVOKED IN REVERSE CHRONOLOGICAL ORDER. EACH CLEANUP$ ON-UNIT IS EXPECTED TO RETURN UNLESS IT ENCOUNTERS A FATAL ERROR. NO ACTIVATION'S STACK FRAME IS REMOVED FROM THE STACK UNTIL ALL ACTIVATIONS HAVE BEEN CLEANED UP (I.E. ALL CLEANUP$ ON-UNITS HAVE RETURNED).

AN ON-UNIT FOR CLEANUP$ MAY PERFORM ANY OPERATION DESIRED, BUT MOST COMMON WILL BE SUCH THINGS AS CLOSING FILE UNITS, FREEING GENERATIONS OF STORAGE THAT HAVE BEEN ALLOCATED IN STATIC AREAS, AND SO ON.

THE CONDITION MECHANISM CANNOT GUARANTEE THAT THE CLEANUP$ ON-UNIT IN A GIVEN ACTIVATION WILL NOT BE INVOKED MORE THAN ONCE, BUT IT CAN AND DOES GUARANTEE THAT, ONCE A CLEANUP$ ON-UNIT HAS RETURNED TO THE CONDITION MECHANISM AND THAT ACTIVATION HAS BEEN MARKED "CLEANED UP", INVOCATION OF ANY OF THAT ACTIVATION'S ON-UNITS (INCLUDING CLEANUP$), AS WELL AS TRANSFER OF CONTROL INTO THAT ACTIVATION BY MEANS OF A FURTHER NONLOCAL GOTO, ARE PREVENTED.

THE ANY$ CONDITION

AN ON-UNIT FOR THE CONDITION ANY$ IS CALLED A "DEFAULT ON-UNIT". A PROCEDURE CREATES AN ON-UNIT FOR ANY$ IN THE NORMAL MANNER (PL/I <ON STATEMENT>, OR A CALL TO MKONU$ OR MKON$F), WHENEVER IT WISHES TO INTERCEPT ANY CONDITION THAT MIGHT BE SIGNALLED DURING ITS ACTIVATION.

WHEN A GIVEN ACTIVATION IS REACHED DURING THE STACK SCAN ASSOCIATED WITH THE RAISING OF A CONDITION, IT IS FIRST EXAMINED FOR AN ON-UNIT FOR THAT SPECIFIC CONDITION. THAT ON-UNIT IS SELECTED FOR INVOCATION IF IT EXISTS. IF THE ACTIVATION HAS NO SPECIFIC ON-UNIT, BUT DOES HAVE AN ON-UNIT FOR ANY$, THEN THE ANY$ ON-UNIT IS SELECTED FOR INVOCATION. THE STANDARD CONDITION FRAME HEADER PASSED TO THE ANY$ ON-UNIT DESCRIBES THE ORIGINAL CONDITION BECAUSE OF WHICH THE ON-UNIT IS BEING INVOKED.

HENCE, A PROCEDURE'S DEFAULT (ANY$) ON-UNIT IS INVOKED ONLY IF THE PROCEDURE HAS NO SPECIFIC ON-UNIT FOR THE GIVEN CONDITION.

USER PROGRAMS SHOULD NOT INCLUDE AN ANY$ ON-UNIT UNLESS TRULY NECESSARY. A USER ANY$ ON-UNIT SHOULD NOT ATTEMPT TO HANDLE MOST SYSTEM CONDITIONS, BUT RATHER SHOULD "PASS THEM ON" BY SIMPLY RETURNING. THE CONTINUE SWITCH (CFH.CFLAGS.CONTINUE_SW) IS SET AUTOMATICALLY WHENEVER AN ANY$ ON-UNIT IS INVOKED. ANY USER ANY$ ON-UNIT THAT FAILS TO RETURN WITH THE CONTINUE SWITCH STILL SET, MAY CAUSE IMPROPER OPERATION OF USER OR PRIME SOFTWARE AT SOME FUTURE RELEASE OF THE SYSTEM.

2.2 FILE SYSTEM ENHANCEMENTS

2.2.1 LISTF

LISTF WILL PRINT THE PATHNAME OF THE DIRECTORY RATHER THAN THE DIRECTORY NAME ONLY. IF THE PATHNAME EXCEEDS 80 CHARACTERS, NO PATHNAME OR DIRECTORY NAME WILL BE PRINTED.

2.2.2 FILE UNITS

THE MAXIMUM NUMBER OF AVAILABLE FILE UNITS PER USER HAS INCREASED FROM 64 TO 128. NOTE THAT UNIT 0 IS STILL RESERVED FOR USE BY THE FILE SYSTEM AND THAT UNIT 127 (WAS 63) IS RESERVED FOR COMOUTPUT FILE USAGE.

2.3 STATUS COMMAND

2.3.1 STATUS UNIT

THE STATUS UNIT COMMAND WILL PRINT MORE INFORMATION PERTAINING TO OPEN FILE UNITS. THE FILE UNIT IS NOW LISTED IN DECIMAL. ALSO LISTED FOR EACH OPEN FILE UNIT IS THE FILE POSITION (IN DECIMAL), THE OPEN MODE, THE READ/WRITE CONCURRENCY LOCK SETTING, THE FILE TYPE AND THE PATHNAME (IF PATHNAME IS LESS THAN 80 CHARACTERS LONG). THE UNIT NUMBER AND PATHNAME ONLY ARE PRINTED WHEN LISTING A DIRECTORY ON A REMOTE DISK.

2.3.2 STATUS DEVICE

THIS COMMAND PRINTS THE LOGICAL TO PHYSICAL CORRESPONDENCE OF MAGTAPE UNITS.

USAGE

```
OK, STATUS DEVICE

DEVICE   USRNAM   USRNUM   LDEVICE
MT0      YLEE        8     MT0
MT1      SYSTEM     27     MT2
```

DEVICE	PHYSICAL DEVICE NAME.
USRNAM	USER LOGIN NAME.
USRNUM	USER NUMBER.
LDEVICE	LOGICAL DEVICE NAME USED BY THE USER WHOM THE DRIVE IS ASSIGNED TO.

2.3.3 STATUS USERS

INFORMATION ABOUT THE SUPERVISOR PROCESS (USER 1) IS NOW LISTED. NOTE THAT SMLC LINES USED BY PRIMENET ARE SHOWN AS ASSIGNED TO THE SUPERVISOR PROCESS.

2.3.4 STATUS ME

THIS COMMAND IS SIMILAR TO STATUS USERS, BUT IT ONLY LISTS INFORMATION ABOUT ALL PROCESSES ON THE SYSTEM WITH THE SAME LOGIN NAME AS THE REQUESTING PROCESS.

2.4 THE PRIMOS COMMAND ENVIRONMENT

2.4.1 INTRODUCTION

THIS SECTION DESCRIBES THE PRIMOS RECURSIVE COMMAND ENVIRONMENT. THIS ENVIRONMENT REPRESENTS A SIGNIFICANT DEPARTURE FROM PREVIOUS IMPLEMENTATIONS OF COMMAND PROCESSING AND USER PROGRAM INVOCATION IN THE PRIMOS SYSTEM.

THE PRIMARY GOAL OF THE NEW ENVIRONMENT IS TO PROVIDE A FULLY RECURSIVE COMMAND PROCESSING LOOP (CALLED THE "COMMAND LOOP" BELOW) THAT IS ALSO HIGHLY MODULAR. THE ABILITY TO RECURSIVELY INVOKE THE COMMAND LOOP MEANS THAT COMMAND PROCEDURE LANGUAGES, AND USER SUBSYSTEMS THAT EXECUTE SYSTEM COMMANDS, BECOME RELATIVELY EASY TO IMPLEMENT. ALSO, THIS RECURSION IS A VERY USEFUL PROPERTY DURING PROGRAM DEBUGGING, AND THE PRIME HIGH-LEVEL LANGUAGE DEBUGGER MAKES EXTENSIVE USE OF THIS CAPABILITY. THE FACT THAT THE COMMAND LOOP IS MODULAR MEANS THAT, WHEN A SUITABLE LINKING MECHANISM IS AVAILABLE, IT WILL BE POSSIBLE TO PROVIDE, FOR EACH USER, A CUSTOM-TAILORED VERSION OF THE COMMAND LOOP THAT FITS THE NEEDS OF THAT USER.

EVERY ATTEMPT HAS BEEN MADE TO PRESERVE A USER ENVIRONMENT THAT IS COMPATIBLE WITH EXISTING USER SOFTWARE. THERE ARE, HOWEVER, SOME DIFFERENCES IN THE ENVIRONMENT WHICH CANNOT BE IGNORED BY SOME USERS DOING SOFTWARE DEVELOPMENT WORK (PRIMARILY DEBUGGING). SEE THE ABOVE SECTION ON THE CONDITION MECHANISM).

2.4.2 RECURSIVE MODE AND STATIC MODE

THE IMPLEMENTATION OF THE NEW ENVIRONMENT PARTITIONS SYSTEM AND USER SOFTWARE INTO TWO CATEGORIES KNOWN AS "RECURSIVE MODE" AND "STATIC MODE". ALL PRESENT USER SOFTWARE, AND ALL PRESENT EXTERNAL COMMANDS, ARE STATIC MODE. STATIC MODE IS CHARACTERIZED BY THESE PROPERTIES: (1) IT ALLOCATES ITS OWN SEGMENTS; (2) IT MANAGES ITS OWN STACK; (3) IT MANAGES ITS OWN PER-PROCESS

LINKAGE SEGMENT (SHARED LIBRARIES); (4) IT USES A "MEMORY IMAGE" APPROACH IN WHICH THE PROGRAM IS RELOADED EACH TIME IT IS CALLED; AND (5) PROGRAMS MAY NOT BE RECURSIVELY INVOKED FROM_COMMAND_LEVEL BECAUSE OF (1)-(4).

THE ONLY RECURSIVE MODE SOFTWARE IN THE SYSTEM AT THIS POINT IS THE OPERATING SYSTEM ITSELF (THE RING ZERO SUBROUTINES, THE ALL-RINGS UTILITY SUBROUTINES (SUCH AS THE CONDITION MECHANISM ROUTINES), AND THE INTERNAL COMMANDS), AND USER ON-UNITS TOGETHER WITH THEIR DYNAMIC DESCENDANTS. RECURSIVE MODE IS CHARACTERIZED BY THESE PROPERTIES: (1) IT USES THE SYSTEM STACK (OF THE APPROPRIATE RING); (2) ALL COMMAND PROCEDURES MAY BE RECURSIVELY INVOKED FROM_COMMAND_LEVEL; (3) AND IT IS NOT RELOADED AS A MEMORY IMAGE EACH TIME IT IS CALLED (INDEED, THE CONCEPT OF MEMORY IMAGE DOES NOT APPLY).

OPERATIONALLY, A RECURSIVE MODE PROCEDURE MUST TERMINATE BY RETURNING, NOT BY CALLING EXIT; ARGUMENTS TO RECURSIVE MODE COMMANDS AS PASSED AS PARAMETERS, AND ARE NOT OBTAINED FROM SOME STATIC BUFFER; ERROR INFORMATION IS PASSED EITHER BY SETTING A RETURN PARAMETER (AN ERROR CODE), BY PRINTING AN ERROR MESSAGE AND RETURNING, OR BY SIGNALLING SOME CONDITION: THE ERRRTN CALL AND ERRPR$ WITH OTHER THAN THE IMMEDIATE-RETURN KEY ARE_NOT_USED. WHEN A METHOD EXISTS FOR THE INVOCATION OF USER RECURSIVE MODE PROGRAMS, THE ABOVE RULES REPRESENT THE ONLY CHANGES (OTHER THAN POSSIBLE DEPENDENCIES ON SPECIFIC SEGMENT NUMBERS) THAT NEED TO BE MADE TO A STATIC MODE PROGRAM TO PERMIT IT TO RUN IN RECURSIVE MODE.

IMPORTANT: USER ON-UNITS AND ALL OF THEIR DYNAMIC DESCENDANTS ARE RECURSIVE MODE SOFTWARE, AND SO SHOULD OBEY THE ABOVE RULES.

2.4.3 THE_BASIC_COMMAND_LOOP

EACH USER IS NOW PROVIDED WITH A RING THREE STACK, WHICH IS USED BY THE VARIOUS MODULES OF THE COMMAND LOOP AS WELL AS BY (RING THREE) INTERNAL COMMANDS. STATIC MODE PROGRAMS ARE NOT PERMITTED TO USE THIS STACK, WHICH RESTRICTION IS ALREADY IN DE-FACTO ENFORCEMENT BY SEG. CURRENTLY, THE STACK IS ASSIGNED TO SEGMENT 6002, BUT NO USER PROGRAM SHOULD DEPEND ON THIS FACT. THE STACK IS NOT PERMITTED TO GROW INTO ANOTHER SEGMENT.

WHEN A USER BECOMES LOGGED IN, THE RING ZERO PROCEDURE LOGIN CAUSES A CRAWLOUT TO RING THREE TO OCCUR (VIA AN INTERMEDIARY KNOWN AS INIT$3). WHEN CONTROL ARRIVES IN RING THREE, THE FIRST CALL ON A MODULE KNOWN AS THE LISTENER (LISTEN_) IS MADE. LISTEN_ IS CHARGED WITH READING COMMAND LINES FROM THE COMMAND INPUT STREAM (VIA CALLS TO CL$GET AS DESCRIBED IN THIS DOCUMENT), AND WITH SUBMITTING THEM TO A COMMAND_PROCESSOR FOR DECODING AND EXECUTION. THUS IT CAN BE SEEN THAT FLOW OF CONTROL NOW ORIGINATES IN RING THREE, UNLIKE PRIOR VERSIONS OF PRIMOS IN WHICH CONTROL ORIGINATED IN RING ZERO AND RETURNED TO RING THREE ONLY TO EXECUTE A USER PROGRAM.

LISTEN_ CALLS THE STANDARD COMMAND PROCESSOR, STD$CP, TO PARSE

THE COMMAND AND CAUSE IT TO BE EXECUTED. THE COMMAND PROCESSOR IS AVAILABLE TO USER PROGRAMS (STATIC OR RECURSIVE MODE) VIA THE ENTRY POINT CP$, IN ORDER TO INSULATE USER SOFTWARE FROM POSSIBLE CHANGES TO ITS CALLING SEQUENCE OR NAME.

STD$CP PARSES THE COMMAND USING THE RULES OF SYNTAX OF THE PRIMOS COMMAND LANGUAGE. WHEN THE COMMAND NAME HAS BEEN ISOLATED, IT IS COMPARED TO A LIST OF INTERNAL COMMAND NAMES. IF THE NAME IS IN THIS LIST, STD$CP INVOKES THE (RECURSIVE MODE) PROCEDURE THAT IMPLEMENTS THAT COMMAND. TEMPORARILY, MANY INTERNAL COMMANDS STILL EXECUTE IN RING ZERO IN THE PERSON OF THE PROCEDURE DOSSUB.

IF THE COMMAND IS NOT INTERNAL, STD$CP CALLS A PROCEDURE KNOWN AS INVKSM_ ("INVOKE STATIC MODE"), WHICH RESTORES THE FILE INTO SEGMENT 4000 AND CAUSES CONTROL TO BE PASSED TO THE PROPER LOCATION.

THE COMMAND LOOP IS INVOKED RECURSIVELY SIMPLY BY CAUSING LISTEN_ TO BE INVOKED. IN ORDER TO INSULATE USER PROGRAMS FROM THE NAME AND CALLING SEQUENCE OF LISTEN_, THE ENTRY POINT COMLV$ IS PROVIDED.

2.4.4 RECURSION OF THE COMMAND LOOP

ASSOCIATED WITH EACH (RECURSIVE) INVOCATION OF LISTEN_ IS A NUMBER CALLED THE LISTENER LEVEL. THE BASE LISTEN_ IS AT LEVEL 1. LISTEN_ IS RECURSIVELY CALLED BY THE SYSTEM DEFAULT ON-UNIT (WHICH IS INVOKED WHEN USER SOFTWARE DOES NOT HAVE ITS OWN ON-UNIT FOR A CONDITION THAT WAS RAISED) AFTER THE DIAGNOSTIC IS PRINTED, AND BY (USER) CALLS TO COMLV$. EACH ADDITIONAL LEVEL OF RECURSION INCREMENTS THE LEVEL NUMBER BY 1.

THE FACT THAT THE SYSTEM DEFAULT ON-UNIT INVOKES A NEW LISTENER LEVEL IS USEFUL BECAUSE IT ALLOWS THE USER TO INVOKE ANY RECURSIVE MODE COMMAND(S), AND THEN RETURN TO THE POINT OF INTERRUPTION, OR TRACE THE STACK TO HELP DETERMINE THE CAUSE OF THE INTERRUPTION.

ONE OF THE MOST COMMON INTERRUPTIONS IS, OF COURSE, THE USE OF THE QUIT BUTTON. THIS RESULTS IN A SIGNAL OF THE CONDITION QUIT$. THE DEFAULT ON-UNIT FOR THIS CONDITION FIRST CLEARS THE TERMINAL INPUT AND OUTPUT BUFFERS, PAUSES ANY COMMAND INPUT FILE IN EFFECT, FORCES TERMINAL OUTPUT ON, AND PRINTS "QUIT". THEN, A NEW LISTEN_ LEVEL IS INVOKED.

UNLIKE PREVIOUS VERSIONS OF PRIMOS, IT WILL NOW BE POSSIBLE TO RESTART INTERNAL COMMANDS THAT HAVE BEEN INTERRUPTED BY A QUIT$ OR OTHER CONDITION SIGNAL, WHEN ALL SUCH COMMANDS HAVE BEEN MOVED OUT TO RING THREE. UNTIL THEN, RING ZERO INTERNAL COMMANDS WILL EITHER HAVE EXECUTED TO COMPLETION BEFORE THE CONDITION WAS RAISED, OR WILL BE RE-EXECUTED FROM THE BEGINNING WHEN THE "START" IS ISSUED.

EACH LISTEN_ LEVEL IS MARKED AS "STATIC MODE" OR "RECURSIVE MODE". A GIVEN LEVEL STARTS OUT AS RECURSIVE MODE, SO THAT, FOR

EXAMPLE, WHEN ONE INTERRUPTS A STATIC MODE PROGRAM AT LEVEL 1, AND LEVEL 2 IS INVOKED, LEVEL 1 IS STATIC MODE AND LEVEL 2 IS RECURSIVE. ONCE A LEVEL HAS BEEN MARKED AS STATIC MODE, IT REMAINS THAT WAY UNTIL THAT LEVEL IS ABORTED OR UNTIL AN "RLS" COMMAND IS ISSUED AT THAT LEVEL. AS WILL BE SEEN, THE FACT THAT A LEVEL IS STATIC MODE (OR RECURSIVE MODE) AFFECTS THE INTERPRETATION GIVEN TO THE "START" INTERNAL COMMAND.

2.4.5 CHANGES TO THE "START" COMMAND

IT HAS PROVED NECESSARY TO SLIGHTLY CHANGE THE MEANING OF THE INTERNAL COMMAND "START" IN ORDER TO ENABLE A USER TO PROPERLY INTERACT WITH STATIC MODE AND RECURSIVE MODE SOFTWARE IN THE SAME TERMINAL SESSION. IN MOST CASES, THE AVERAGE USER WILL PERCEIVE NO DIFFERENCE.

THE "START" COMMAND NOW PERFORMS ONE OF THREE ACTIONS DEPENDING ON THE STATIC/RECURSIVE MODE STATE OF THE CURRENT COMMAND LEVEL, AND ON THE ARGUMENTS TO THE START COMMAND.

2.4.5.1 "START" AT A STATIC MODE LEVEL

IN THIS CASE, THE STATIC MODE PROGRAM DEFINED BY THE STATIC MODE STATE VECTOR (PRINTED BY THE PM COMMAND AND KNOWN AS THE "RVEC") IS INVOKED AT THE CURRENT COMMAND LEVEL. SINCE THE RESTORE OPERATION IS WHAT CAUSES A LEVEL TO BECOME MARKED AS STATIC MODE, AN EXTERNAL COMMAND (WHICH IS EQUIVALENT TO THE SEQUENCE "RESTORE CMDNCO>COMMAND" FOLLOWED BY "START COMMAND_ARGS") FALLS UNDER THIS CASE.

2.4.5.2 "START" AT A RECURSIVE MODE LEVEL (NO ARGUMENTS)

IN THIS CASE, THE "START" COMMAND HAS BEEN GIVEN NO ARGUMENTS. ITS EFFECT IS TO CAUSE THE CURRENT LISTEN_ TO RETURNS TO ITS CALLER. IF THIS COMMAND LEVEL WAS INVOKED BY THE SYSTEM DEFAULT ON-UNIT IN RESPONSE TO A CONDITION SIGNAL, LISTEN_ RETURNS TO THE DEFAULT ON-UNIT, WHICH IN TURN RETURNS TO THE CONDITION MECHANISM. IF RETURN FROM THE CONDITION IS PERMITTED, THE CONDITION MECHANISM WILL INVOKE THE NEXT ON-UNIT IF CONTINUATION OF SIGNALLING WAS REQUESTED, OR ELSE WILL RETURN TO THE POINT OF INTERRUPTION. IF THIS COMMAND LEVEL WAS INVOKED BY A USER CALL TO COMLV$, COMLV$ NOW RETURNS TO ITS CALLER.

HENCE, WHEN A STATIC MODE PROGRAM AT LEVEL 1 IS INTERRUPTED BY A QUIT, AND "START" WITHOUT ARGUMENTS IS TYPED AT LEVEL 2, THE LEVEL 2 LISTEN_ RETURNS TO THE DEFAULT ON-UNIT, WHICH RETURNS TO SIGNL$, WHICH RETURNS TO THE POINT WHERE THE QUIT FAULT OCCURRED, CAUSING RESUMPTION OF THE INTERRUPTED COMPUTATION. TO CONTINUE THIS EXAMPLE, IF LEVEL 2 BECOMES MARKED AS STATIC MODE (DUE TO THE USE OF AN EXTERNAL COMMAND, OR THE RESTORE OR RESUME COMMANDS), A SUBSEQUENT "START" WILL FALL UNDER THE STATIC MODE CASE ABOVE. FURTHERMORE, SINCE STATIC MODE OPERATES IN MEMORY-IMAGE FASHION, IT IS NOT IN GENERAL POSSIBLE TO EVER CONTINUE AN INTERRUPTED STATIC MODE PROGRAM AFTER HAVING INVOKED ANOTHER STATIC MODE PROGRAM (OR RESTORING ONE) AT SOME HIGHER

COMMAND LEVEL. THE COMMAND LOOP ENFORCES THIS RESTRICTION.

2.4.5.3 "START" AT A RECURSIVE MODE LEVEL (ARGUMENTS)

THE USE OF THIS CASE OF START SHOULD BE RESTRICTED TO THE CASE WHERE A "RESTART ADDRESS" IS BEING SUPPLIED. FOR EXAMPLE, WHEN ONE HAS QUIT OUT OF THE EDITOR, RESTARTING AT ADDRESS '1000 PRESERVES THE WORKING BUFFER. EVENTUALLY, MOST PRIME SOFTWARE WILL SUPPORT THE USE OF THE "REN" (REENTER) COMMAND, WHICH DOES NOT INVOLVE THE NEED TO KNOW SUCH RESTART ADDRESS VALUES.

IN THIS CASE, THE CURRENT COMMAND LEVEL IS MARKED STATIC MODE, THE PARAMETERS FROM THE "START" COMMAND MODIFY THE STATIC MODE RVEC, AND THE RESULTING STATIC MODE PROGRAM IS INVOKED AT THE CURRENT COMMAND LEVEL, AS FOR THE STATIC MODE CASE.

2.4.6 THE "RLS" COMMAND

THE NEW INTERNAL COMMAND "RLS" CAN BE USED TO RELEASE COMMAND LEVELS ON THE STACK THAT ARE NO LONGER USEFUL.

IF THE STACK HISTORY IS NOT TO BE OR HAS ALREADY BEEN EXAMINED, THE RLS COMMAND SHOULD BE USED TO RELEASE THE UNWANTED HISTORY AND THUS CONSERVE STACK SPACE AND REDUCE PAGING ACTIVITY ON IT.

IF, HOWEVER, THE CURRENT LISTENER LEVEL IS STATIC MODE, THE RLS COMMAND CAN BE USED TO RELEASE THE STATIC MODE PROGRAM BY MAKING THE CURRENT LISTENER LEVEL RECURSIVE MODE. SEE THE WRITEUP ON THE RLS COMMAND FOR MORE DETAILS.

2.4.7 THE MEANING OF THE "PM" AND "PRERR" COMMANDS

THE DATA PRINTED OUT BY THE INTERNAL COMMAND "PM" (POSTMORTEM) IS SUPPOSED TO REPRESENT THE MACHINE STATE OF THE STATIC MODE IMAGE. THIS IS STILL TRUE IN ALL CASES EXCEPT THOSE IN WHICH THE STATIC MODE PROGRAM IS INTERRUPTED BY A CONDITION SIGNAL, BUT HAS ITS OWN ON-UNIT (OR DEFAULT ON-UNIT) WHICH DOES NOT PERMIT THE STATIC MODE OVERSEER PROCEDURE TO GAIN CONTROL AND HENCE UPDATE THE "PM" DATA. THUS, IF A USER STATIC MODE PROGRAM'S ON-UNIT FOR SOME CONDITION CALLS COMLV$ TO GET TO COMMAND LEVEL, AND THEN THE USER ENTERS THE "PM" COMMAND, THE DATA DISPLAYED WILL NOT ACCURATELY REFLECT THE MACHINE STATE OF THE STATIC MODE PROGRAM. THE DMSTK COMMAND, HOWEVER, ALWAYS PRODUCES AN ACCURATE DISPLAY.

THE "PRERR" COMMAND, WHICH DISPLAYS THE CONTENTS OF THE OLD ERROR VECTOR ERRVEC, WILL NO LONGER DISPLAY ANY USEFUL INFORMATION FOLLOWING SUCH FAULTS AS ACCESS VIOLATION OR ILLEGAL SEGMENT NUMBER. THIS INFORMATION, HOWEVER, IS NOW PRINTED BY THE SYSTEM AS PART OF THE DIAGNOSTIC, AND IS ALSO AVAILABLE IN THE COMMAND LOOP STACK.

2.4.8 OTHER NEW COMMANDS

THREE OTHER NEW INTERNAL COMMANDS WERE ADDED TO SUPPORT THE NEW ENVIRONMENT. THE COMMAND "DMSTK" (DUMP STACK) PRODUCES A FORMATTED TRACE OF A USER OR COMMAND PROCESSOR STACK. THE COMMAND "RDY" (READY) ALLOWS THE USER TO CHANGE THE FORMAT OF THE "OK" MESSAGE, OR TO TURN IT OFF ALTOGETHER. THE COMMAND "REN" (REENTER) GENERATES A CONDITION SIGNAL THAT WILL ALLOW SUBSYSTEMS TO BE REENTERED GRACEFULLY WITHOUT THE NEED TO SUPPLY A RESTART ADDRESS IN A START COMMAND.

THESE COMMANDS ARE DOCUMENTED IN A SECTION FOLLOWING.

2.4.9 CHANGES TO COMMAND LOOP MESSAGES

EACH USER MAY ELECT TO USE AN EXPANDED FORM OF THE "OK" AND "ER" MESSAGES THAT WILL INCLUDE THE TIME OF DAY, THE CURRENT LISTENER LEVEL, AND OTHER USEFUL INFORMATION. SEE THE WRITEUP ON THE RDY COMMAND BELOW FOR DETAILS.

THE "GO" MESSAGE IS NO LONGER PRINTED FOLLOWING THE "START" INTERNAL COMMAND, NOR AFTER A RESUME OR EXTERNAL COMMAND.

THE "RESERVED CHAR" MESSAGE FROM THE SUBROUTINE RDTK$$ HAS BEEN REMOVED. THIS DOES NOT IMPLY THAT THE CHARACTERS "()[];=!;^?%+:= <\", COMMERICAL-AT, QUOTE, TILDE, AND RUBOUT, ARE NO LONGER RESERVED. ANY OR ALL OF THESE CHARACTERS MAY BE USED IN THE FUTURE FOR EXPANSION OF THE PRIMOS COMMAND LANGUAGE AND/OR ITS INTERFACES AND UTILITIES.

2.4.10 ABBREV -- NEW INTERNAL COMMAND

THE INTERNAL COMMAND ABBREV CONTROLS THE ABBREVIATION PREPROCESSOR FOR COMMANDS. THIS FACILITY ALLOWS A USER TO PERSONALIZE HIS TERMINAL COMMAND LANGUAGE AS HE SEES FIT. ABBREVIATIONS CAN BE USED TO SHORTEN NAMES OF COMMANDS, OR TO ENCAPSULATE HARD-TO-REMEMBER ARGUMENT LISTS FOR COMMANDS. SINCE THE USER HIMSELF DEFINES THE ABBREVIATION NAMES AND VALUES, IT FOLLOWS THAT HE WILL CONSIDER THEM EASY TO REMEMBER AND TO USE.

2.4.10.1 OVERVIEW OF ABBREV

ABBREV IS AN INTERNAL COMMAND WHICH ALLOWS THE PRIMOS USER TO DEFINE HIS OWN ABBREVIATIONS FOR COMMAND-LINE TOKENS; FOR EXAMPLE, WITH ITS USE, ONE COULD INVOKE THE EXTERNAL COMMAND X.PRINT WITH THE COMMAND-ABBREVIATION PR. THIS CORRESPONDENCE IS REMEMBERED BY THE SYSTEM IN THE FORM:

```
(C) PR          X.PRINT
```

THIS INFORMATION IS HELD IN A TABLE CALLED AN "ABBREVIATION FILE". THE USER INVOKES ABBREV IN ORDER TO CREATE, ACTIVATE, OR ALTER SUCH AN ABBREVIATION FILE. TO ACTIVATE THE ABBREVIATION FILE SLUTZ>SLUTZ.ABBREV, SLUTZ WOULD TYPE

```
OK, AB SLUTZ>SLUTZ.ABBREV

TO CREATE AN EMPTY ABBREVIATION FILE, THE PRIMOS COMMAND LINE

OK, AB SLUTZ>SLUTZ_NEW -CREATE

IS USED.  THE FILE MUST NOT ALLREADY EXIST.
```

EACH PRIMOS COMMAND TYPED AT THE USERS TERMINAL WILL BE SCANNED FOR ANY ABBREVIATIONS FOUND IN THE CURRENTLY ACTIVE ABBREVIATION FILE. EACH ABBREVIATION FOUND IN THE PRIMOS COMMAND LINE IS SUPPLANTED BY ITS EXPANDED VALUE; THE EXPANDED COMMAND LINE IS THEN PASSED TO THE COMMAND PROCESSOR.

```
     THE PRIMOS COMMAND LINE

OK, AB -ADD_COMMAND PR X.PRINT

WOULD INSERT THE ABOVEMENTIONED ABBREVIATION PR INTO THE ACTIVE
ABBREVIATION FILE.
```

AN ABBREVIATION IS FORMED OF TWO PARTS, ITS NAME AND ITS VALUE. AN ABBREVIATION NAME IS 1 TO 8 CHARACTERS IN LENGTH. ALL ASCII CHARACTERS ARE LEGAL EXCEPT FOR SPACE, TAB, QUOTE('), COMMA, GREATER THAN (>), AND VERTICAL BAR (<). LOWERCASE LETTERS CAN BE USED INTERCHANGEABLY WITH UPPERCASE LETTERS IN AN ABBREVIATION NAME.

IN AN ABBREVIATION VALUE, ALL ASCII CHARACTERS ARE LEGAL. THE CHARACTER PERCENT SIGN (%) IS USED AS AN ESCAPE CHARACTER. THE SEQUENCE %I% IS REPLACED BY THE ITH TOKEN FOLLOWING THE ABBREVIATION NAME ON THE COMMAND LINE. THE SEQUENCE %% IS REPLACED BY %. THIS ALLOWS REORDERING AND DUPLICATION OF TOKENS IN THE COMMAND LINE. FOR EXAMPLE:

```
IF T = "%1% *>%1%>%2%.%1% -B *>%1%.BIN>B_%2% -L *>%1%.LIST>L_%2%"
THEN THE COMMAND LINE:

T PMA FOO -EXPLIST

WOULD EXPAND INTO THE FOLLOWING TO BE PASSED TO THE COMMAND
PROCESSOR:

PMA *>PMA>FOO.PMA -B *>PMA.BIN>B_FOO -L *>PMA.LIST>L_FOO -EXPLIST
```

ABBREVIATIONS ARE AUTOMATICALLY TURNED OFF WHEN COMMAND LINES ARE READ FROM A COMMAND FILE, AND TURNED ON WHEN THE COMMAND COMES FROM THE USER.

2.4.10.2 ABBREV

NAMES: ABBREV, AB

PURPOSE:

THE COMMAND ABBREV IS USED TO ACTIVATE, DEACTIVATE, AND EDIT PER-USER COMMAND ABBREVIATIONS. WHEN ACTIVATED, THE LISTENER WILL CALL THE ABBREVIATION PREPROCESSOR BEFORE CALLING THE STANDARD COMMAND PROCESSOR. INCLUDED IN THE VALUE OF AN ARBREVIATION ARE ESCAPE SEQUENCES THAT ALLOWS REORDERING OF TOKENS ON THE COMMAND LINE. THERE IS ALSO A WAY TO CONCATENATE TWO TOKENS TOGETHER SO THAT EACH CAN BE EXPANDED BUT CAN STILL APPEAR AS A SINGLE TOKEN TO THE STANDARD COMMAND PROCESSOR.

USAGE:

ABBREV [<TREE_NAME>] [-CONTROL_ARGUMENT(S)]

IF <TREE_NAME> IS SUPPLIED THEN THE CONTROL ARGUMENT -ON IS ASSUMED AND THE ABBREVIATION TABLE IS LOADED FROM THE FILE SPECIFIED BY <TREE_NAME>. IF THE FILE SPECIFIED BY <TREE_NAME> DOES NOT EXIST OR IS NOT A VALID ABBREVIATION FILE THEN AN ERROR MESSAGE WILL BE PRINTED AND THE FILE WILL NOT BE USED AS AN ABBREVIATION FILE.

WHILE ARBREVIATIONS ARE IN EFFECT THE USER CAN SUPPRESS EXPANSION OF A PARTICULAR STRING BY ENCLOSING IT IN QUOTES('). TO SUPPRESS EXPANSION OF THE ENTIRE COMMAND LINE SEE THE -EXECUTE CONTROL ARGUMENT REQUEST.

ABBREVIATIONS ARE NOT EXPANDED WHEN COMMAND LINES ARE READ FROM A COMMAND FILE.

CONTROL_ARGUMENT_REQUESTS

THE CONTROL ARGUMENTS TO ABBREV HANDLE THE MANAGEMENT OF THE ABBREVIATION TABLE. THE TEXT SHOWN BELOW IN BRACES (;=) IS THE ABBREVIATION FOR THE CONTROL REQUEST. THE FOLLOWING CONTROL ARGUMENT REQUESTS ARE IMPLEMENTED:

-OFF
TURN ABBREVIATION EXPANSION OFF.

-ON
TURN ABBREVIATION EXPANSION ON. WILL USE THE OLD <TREENAME> IF NOT SUPPLIED.

-CREATE ;-CR=
CREATE AN EMPTY FILE IF THE <TREE_NAME> DOES NOT EXISTS. HAS NO EFFECT IF <TREE_NAME> ALLREADY EXISTS.

-VERIFY ;-VFY=
TURN ON VERIFY MODE. THIS CAUSES THE LISTENER TO PRINT THE EXPANDED COMMAND LINE JUST BEFORE CALLING THE STANDARD COMMAND PROCESSOR.

-NO_VERIFY ;-NVFY=
TURN OFF VERIFY MODE. THIS IS THE DEFAULT SETTING OF

VERIFY MODE.

-ADD ;-A= <NAME> <VALUE>
ADD THE ABBREVIATION <NAME> TO THE CURRENT ABBREVIATION FILE, AND GIVE IT THE VALUE <VALUE>. ALL LEADING SPACES AND/OR TABS ARE REMOVED FROM THE VALUE. NOTE THAT <VALUE> MAY CONTAIN ANY CHARACTERS INCLUDING SPACES. IF AN ABBREVIATION FOR <NAME> ALREADY EXISTS THEN THE USER WILL BE ASKED IF HE WANTS TO REPLACE IT. AN ABBREVIATION NAME MAY BE AT MOST EIGHT CHARACTERS. ALL ABBREVIATION NAMES ARE CONVERTED TO UPPERCASE, TO MAKE THEM CASE INDEPENDENT.

-ADD_ARGUMENT ;-AA= <NAME> <VALUE>
ADD AN ABBREVIATION THAT IS EXPANDED ONLY IN THE ARGUMENT POSITION IN THE COMMAND LINE.

-ADD_COMMAND ;-AC= <NAME> <VALUE>
ADD AN ABBREVIATION THAT IS EXPANDED ONLY IN THE COMMAND POSITION IN THE COMMAND LINE.

-CHANGE ;-C= <NAME1> ... <NAMEN>
CHANGE THE SPECIFIED ABBREVIATIONS <NAME1>, ..., <NAMEN>, TO BE EXPANDABLE IN ALL POSITIONS ON THE COMMAND LINE.

-CHANGE_ARGUMENT ;-CA= <NAME1> ... <NAMEN>
CHANGE THE SPECIFIED ABBREVIATIONS TO BE EXPANDABLE ONLY IN THE ARGUMENT POSITION ON THE COMMAND LINE.

-CHANGE_COMMAND ;-CC= <NAME1> ... <NAMEN>
CHANGE THE SPECIFIED ABBREVIATIONS TO BE EXPANDABLE ONLY IN THE COMMAND POSITION ON THE COMMAND LINE.

-CHANGE_NAME ;-CN= <OLD>NAME> <NEW>NAME>
CHANGE THE NAME OF THE ABBREVIATION <OLD>NAME> TO <NEW>NAME>.

-DELETE ;-DL= <NAME1> ... <NAMEN>
DELETE THE SPECIFIED ABBREVIATIONS FROM THE ABBREVIATION FILE.

-LIST <NAME1> ... <NAMEN>
LIST THE SPECIFIED ABBREVIATIONS IN THE CURRENT ABBREVIATION FILE. IF THERE ARE NO <NAMEI> SUPPLIED THEN ALL OF THE ABBREVIATIONS ARE LISTED.

-STATUS ;-ST=
PRINT OUT THE NAME OF THE CURRENT ABBREVIATION FILE AND THE NUMBER OF DEFINED ABBREVIATIONS.

-NO_QUERY ;-NQ=
DO NOT ASK TO REPLACE AN OLD ABBREVIATION IF IT EXISTS. ONLY USEFUL IF FOLLOWED BY ONE OF THE ADD REQUESTS.

-EXECUTE ;-EX= <REST>OF>LINE>

THIS REQUEST JUST PASSES THE <REST>OF>LINE> TO THE STANDARD COMMAND PROCESSOR WITHOUT MAKING ANY CHANGES.

-EXPAND ;-EXP= <REST>OF>LINE>
THIS REQUEST EXPANDS <REST>OF>LINE> AND PRINTS THE RESULT ON THE USERS TERMINAL INSTEAD OF PASSING IT TO THE STANDARD COMMAND PROCESSOR.

BREAK_CHARACTERS:

WHEN A COMMAND LINE IS EXPANDED CERTAIN CHARACTERS ARE TREATED AS BREAK CHARACTERS BETWEEN TOKENS THAT ARE CANDIDATES FOR EXPANSION. THESE CHARACTERS ARE SPACE, TAB, COMMA, ">", AND " THERE CAN BE ANY NUMBER OF SPACES AND/OR TABS SEPARATING TOKENS. THE VERTICAL BAR IS USED TO CONCATENATE THE (POSSIBLY EXPANDED VALUE OF THE) TWO TOKENS TOGETHER, AND THE BAR IS DELETED FROM THE EXPANDED LINE. QUOTES ARE HANDLED IN THE STANDARD MANNER AND A TOKEN THAT IS QUOTED WILL NEVER BE EXPANDED. THE QUOTED TOKEN WILL HAVE ALL OF THE QUOTES IN THE EXPANDED LINE THAT IT HAD IN THE INPUT COMMAND LINE.

TOKEN_REPOSITIONING_FACILITY:

THE CHARACTER AT SIGN(%) IS USED AS AN ESCAPE CHARACTER WITHIN THE VALUE OF AN ABBREVIATION. AN ABBREVIATION WHOSE VALUE CONTAINS ONE OR MORE AT SIGNS IS SAID TO PARAMETERIZED. WHEN SUCH AN ABBREVIATION IS EXPANDED, THE SEQUENCE %I% IN ITS VALUE IS REPLACED BY THE ITH TOKEN FOLLOWING ON THE COMMAND LINE. CURRENTLY I IS IN THE RANGE ONE TO NINE. THE CHARACTER SEQUENCE %% IS REPLACED BY A SINGLE AT SIGN. THE CHARACTERS %1% ARE REPLACED BY THE FIRST TOKEN FOLLOWING THE NAME, %2% BY THE SECOND, ETC. IF THE END OF THE COMMAND LINE IS REACHED BEFORE THE ITH TOKEN IS FOUND THEN %I% BECOMES THE NULL STRING.

THE NUMBER OF TOKENS FOLLOWING THE PARAMETERIZED ABBREVIATION NAME THAT ARE COMBINED WITH ITS VALUE IS EQUAL TO THE MAXIMUM %I% CONTAINED IN THE PARAMETERIZED ABBREVIATION. HENCE, IF X = "%1% %4%", THEN "X A B C D E" EXPANDS TO "A D E", EVEN THOUGH THE SECOND AND THIRD PARAMETER TOKENS ARE NOT ACTUALLY USED. IF PARAMETERIZED ABBREVIATIONS ARE NESTED (I.E. A PARAMETER OF ONE IS ITSELF PARAMETERIZED), THE NEST IS EXPANDED IN INSIDE-OUT ORDER.

NOTES:

COMMAND LINES THAT ARE BEING EXPANDED WHOSE FIRST TOKEN IS "AB" OR "ABBREV" ARE TREATED DIFFERENTLY. THE REMAINDER OF THE COMMAND LINE IS NOT EXPANDED. THE COMMAND LINE IS CHECKED AFTER EXPANSION OF THE FIRST TOKEN. THIS MEANS THAT ONE CAN HAVE ABBREVIATINS FOR ABBREV CONTROL ARGUMENT REQUESTS (I.E. THE ABBREVIATION .L = "ABBREV -LIST" IS USEFUL IN LISTING THE CURRENT ABBREVIATIONS).

AT PRESENT THE ABBREVIATION TABLE IS KEPT IN PART OF A SEGMENT. IF ANY CHANGES ARE MADE TO THE SEGMENT BECAUSE OF

CONTROL REQUESTS, THEN THE ABBREVIATION FILE IS UPDATED. THIS FILE IS SIMPLY AN IMAGE OF PART OF THE SEGMENT THAT CONTAINS THE ABBREVIATIONS. SINCE PART OF A SEGMENT IS USED TO HOLD THE ABBREVIATIONS THERE IS A LIMIT ON THE NUMBER OF ABBREVIATIONS ONE MAY HAVE. WITH 40-CHARACTER VALUES FOR EACH NAME IT IS POSSIBLE TO HAVE OVER 200 ABBREVIATIONS. THIS SHOULD BE MORE THAN IS EVER NEEDED IN PRACTICE.

THE ABBREVIATION NAMES ARE KEPT SORTED IN THE SEGMENT, AND A BINARY SEARCH IS USED TO LOOK THEM UP. THIS MEANS THAT AT MOST EIGHT COMPARISIONS HAVE TO BE MADE TO SEE IF THERE IS AN ABBREVIATION FOR A GIVEN TOKEN. THE ACTUAL NUMBER OF COMPARISIONS IS INT(LOG(N)/LOG(2)+1) WHERE N IS THE NUMBER OF ABBREVIATIONS.

2.4.10.3 ABBREV_EXAMPLE

THE FOLLOWING IS AN EXAMPLE OF USING ABBREVIATIONS. THE USER'S INPUT IS THE TEXT FOLLOWING THE 'OK,'. [IF A COMMENT WERE TYPED ON THE SAME LINE WHERE AN ABBREVIATION IS ADDED THEN THAT COMMENT WOULD BE TAKEN AS PART OF THE VALUE OF THE ABBREVIATION].

```
OK, ABBREV SLUTZ>SLUTZ.EXAMPLE /* SPECIFY ABBREVIATION FILE
CREATING NEW ABBREVIATION FILE: SLUTZ>SLUTZ.EXAMPLE (AB_FILE_)

OK, /* ADD THE ABBREVIATION T EXPANDABLE ONLY IN COMMAND POSITION
.
ABBREV -ADD COMMAND T %1% *>%1%>%2%.%1% -B *>%1%.BIN>B_%2% -L *>%
1%.LIST>L_%2%

OK, /* ADD ABBREVIATION RV EXPANDABLE ANYWHERE.
ABBREV -ADD RV %3% %2% %1%

OK, /* ADD ABBREVIATION A EXPANDABLE ANYWHERE.
ABBREV -ADD A FOO%1%BAR

OK, /* ADD ABBREVIATION B EXPANDABLE ANYWHERE.
ABBREV -ADD B %1%000

OK, ABBREV -LIST                /* LIST ALL OF THE ABBREVIATIONS.
ABBREVIATION FILE: SLUTZ>SLUTZ.EXAMPLE
ABBREVIATIONS: 4

    A           FOO%1%BAR
    B           %1%000
    RV          %3% %2% %1%
(C) T           %1% *>%1%>%2%.%1% -B *>%1%.BIN>B_%2% -L *>%1%.LIST>L
_%2%

OK, /* SEE WHAT THE COMMAND LINE 'T PMA FX -EXP' WOULD EXPAND TO
ABBREV -EXPAND T PMA FX -EXP
(LISTEN_) "PMA *>PMA>FX.PMA -B *>PMA.BIN>B_FX -L *>PMA.LIST>L_FX
-EXP"
```

```
OK, /* CHECK EXPANSION OF 'RV ONE TWO THREE'
ABBREV -EXPAND RV ONE TWO THREE
(LISTEN_) "THREE TWO ONE"

OK, /* SHOW THE DIFFERENCE BETWEEN NON QUOTING AND QUOTING.
ABBREV -EXPAND A B C D
(LISTEN_) "FOOCOOOBAR D"

OK, ABBREV -EXPAND A 'B' C D
(LISTEN_) "FOO'B'BAR C D"

OK, ABBREV -DELETE A B RV      /* DELETE THE ABBREVIATIONS A, B, A
ND RV

OK, ABBREV -LIST    /* LIST THE REMAINING ABBREVIATIONS
ABBREVIATION FILE: SLUTZ>SLUTZ.EXAMPLE
ABBREVIATIONS: 1

(C) T         %1% *>%1%>%2%.%1% -B *>%1%.BIN>B_%2% -L *>%1%.LIST>L
_%2%


OK, ABBREV -VERIFY              /* TURN ON VERIFY MODE

OK, /* ADD AN ABBREVIATION TO ADD ABBREVIATIONS
ABBREV -ADD_COMMAND .AC ABBREV -ADD_COMMAND
(LISTEN_) "ABBREV -ADD_COMMAND .AC ABBREV -ADD_COMMAND"

OK, /* USE IT TO ADD AN ABBREVIATION TO LIST ABBREVIATIONS
.AC .L ABBREV -LIST
(LISTEN_) "ABBREV -ADD_COMMAND .L ABBREV -LIST"

OK, /* ADD AN ABBREVIATION TO DELETE ABBREVIATIONS
.AC .D ABBREV -DELETE
(LISTEN_) "ABBREV -ADD_COMMAND .D ABBREV -DELETE"

OK, .L                /* LIST ALL ABBREVIATIONS
(LISTEN_) "ABBREV -LIST"
ABBREVIATION FILE: SLUTZ>SLUTZ.EXAMPLE
ABBREVIATIONS: 4

(C) .AC       ABBREV -ADD_COMMAND
(C) .D        ABBREV -DELETE
(C) .L        ABBREV -LIST
(C) T         %1% *>%1%>%2%.%1% -B *>%1%.BIN>B_%2% -L *>%1%.LIST>L
_%2%

OK, ABBREV -NO_VERIFY              /* TURN OFF VERIFY MODE
(LISTEN_) "ABBREV -NO_VERIFY"

OK, /* THIS IS THE END OF THE EXAMPLE.
```

2.5 PRIMENET

PRIMENET PROVIDES A WIDE RANGE OF NETWORK COMMUNICATIONS SERVICES FOR PRIME SYSTEMS. THE SYSTEMS IN A PRIMENET NETWORK MAY BE EITHER PHYSICALLY LOCAL (HUNDREDS OF FEET) AND CONNECTED IN A HIGH SPEED RING, OR SCATTERED AROUND THE WORLD AND CONNECTED VIA SYNCHRONOUS TELEPHONE LINES.
IN ADDITION, PRIMENET HAS (AT REV 16.2) BEEN EXTENDED TO SUPPORT NETWORK CONNECTIONS THROUGH CCITT RECOMMENDATION X.25 COMPLIANT PACKET NETWORKS (E.G. TELENET). SINCE THE INTRODUCTION OF THIS NEW FUNCTIONALITY, ALL OF THE PRIMENET SERVICES ARE AVAILABLE BETWEEN TWO PRIME SYSTEMS EACH CONNECTED TO A PACKET NETWORK.

PRIMENET SOFTWARE PROVIDES THREE DISTINCT SETS OF SERVICES.
THE INTER-PROCESS COMMUNICATIONS FACILITY (IPCF) LETS PROGRAMS RUNNING UNDER PRIMOS ESTABLISH FULL DUPLEX NETWORK DATA LINKS (VIRTUAL CIRCUITS) TO PROGRAMS IN ANY PRIMENET SYSTEM AND (WHEN CONNECTED TO A PACKET NETWORK) TO THE EQUIPMENTS OF OTHER VENDORS SUPPORTING X.25.
THE INTERACTIVE TERMINAL SUPPORT (ITS) FACILITY PERMITS TERMINALS ATTACHED TO ONE PRIMENET SYSTEM TO LOG-IN TO ANOTHER PRIMENET SYSTEM AND APPEAR AS A NORMAL PRIMOS USER IN THAT SECOND MACHINE. WHEN A PRIMENET SYSTEM IS CONNECTED TO A PACKET NETWORK, USERS MAY ACCESS PRIMOS FROM TERMINALS CONNECTED TO THE PACKET NETWORK.
THE FILE ACCESS MANAGER (FAM) ALLOWS PROGRAMS RUNNING UNDER PRIMOS TO ACCESS FILES PHYSICALLY STORED AT OTHER PRIME SYSTEMS IN THE NETWORK.
REMOTE FILE OPERATIONS ARE LOGICALLY TRANSPARENT TO THE APPLICATION PROGRAM. WITH THE FAM, USERS NEED NOT REPROGRAM OR LEARN NEW COMMANDS FOR TRANS-NETWORK FILE ACCESS.

PRIMENET SUPPORTS NETWORK CONNECTIONS THROUGH TWO STANDARD PRIME HARDWARE CONTROLLERS. THE FIRST IS THE PRIMENET NODE CONTROLLER (PNC).
THE PNC IS USED TO PROVIDE NETWORK COMMUNICATIONS BETWEEN PHYSICALLY LOCAL PRIME SYSTEMS. THE MACHINES MUST BE CABLED INTO A "RING" WITH NO MORE THAN 750 FEET BETWEEN ADJACENT RUNNING SYSTEMS, BUT THE RESULT IS A FULLY INTERCONNECTED NETWORK WITH AN AVAILABLE BANDWIDTH OF TEN MEGABITS. INDIVIDUAL SYSTEMS CAN BE POWERED OFF AND RESTORED TO SERVICE WITHOUT DISTURBING THE NETWORK OPERATIONS OF THE OTHER MACHINES ON THE RING.

THE OTHER CONTROLLER TYPE IS THE MULTI-LINE DATA LINK CONTRO
LLER (MDLC).
THE MDLC PROVIDES THE INTERFACE BETWEEN A PRIME SYSTEM AND UP TO
FOUR
SERIAL SYNCHRONOUS COMMUNICATION LINES. MANY COMMUNICATION PROTO
COL
FUNCTIONS ARE HANDLED BY THE MDLC AND SO THE BOARD MUST BE CONFIG
URED TO
USE THE CONVENTIONS OF PARTICULAR PROTOCOLS. FOR USE WITH PRIMEN
ET,
THE MDLC IS CONFIGURED WITH EITHER THE BINARY SYNCHRONOUS OR HDLC
/X.25
PROTOCOL OPTIONS. (IT CAN SUPPORT A DIFFERENT PROTOCOL OVER EACH
LINE.)
THE MDLC CAN BE ORDERED IN EITHER A TWO OR FOUR LINE
VERSION. CURRENTLY, A SYSTEM MAY CONFIGURE ONLY TWO SYNCHRONOUS
PRIMENET
LINES, BUT OTHER MDLC LINES MAY BE USED TO SUPPORT SYNCHRONOUS TE
RMINALS
OR ANY OF PRIME'S REMOTE JOB ENTRY EMULATOR SUBSYSTEMS CONCURRENT
LY.

PRIMENET SOFTWARE IS AVAILABLE IN TWO FORMS. THE FIRST IS S
HIPPED
IN THE MASTER DISK UFD PRINET. PRINET INCLUDES ALL OF THE
PRIMENET SOFTWARE REQUIRED TO SUPPORT PRIME-TO-PRIME CONNECTIONS
USING THE HIGH-SPEED RING NETWORK OR DEDICATED TELEPHONE COMPANY
SYNCHRONOUS LEASED LINES.
THE ADDITIONAL SOFTWARE NEEDED TO SUPPORT
PRIMENET CONNECTIONS TO AND THROUGH AN X.25 COMPLIANT PACKET NETW
ORK
(THE PACKET_NETWORK_INTERFACE) IS INCLUDED WITH ALL OF THE
REGULAR PRIMENET SOFTWARE IN THE MASTER DISK UFD X.25.
AT REV 17, EACH MASTER DISK SHIPPED WITH NETWORKS WILL HAVE ONE O
F THESE
TWO UFDS.

BOTH THE PRINET AND X.25 MASTER DISK UFDS HAVE WITHIN THEM NINE S
UBUFDS.
SUBUFDS CONTAIN ALL OF THE SOURCE AND COMMAND FILES REQUIRED TO B
UILD
THE NON-PRIMOS PRIMENET SOFTWARE. THE FOLLOWING IS AN ANNOTATED
LIST OF
THESE SUBUFDS.
IN ALL CASES BELOW, CUSTOMERS WITH THE PACKET NETWORK INTERFACE S
HOULD
REPLACE "PRINET" WITH "X.25".

FAM - THE FILE ACCESS MANAGER (FAM) RUNS AS A PRIMOS PHANTOM T
O SUPPORT
TRANSPARENT FILE ACCESS AMONG PRIMENET SYSTEMS. THIS SU
BUFD CONTAINS THE
FAM SOURCE MODULES, A COMMAND FILE WITH WHICH TO BUILD T

HE EXECUTABLE
FAM MODULE AND A COMMAND FILE THAT INITIATES THE FAM PHA
NTOM.
NOTE: THE FAM MUST RESIDE IN A UFD-LEVEL DIRECTORY (DIRE
CTLY UNDER AN MFD).
IT CANNOT BE RUN FROM PRINET.

NETPRT - PRIMENET CURRENTLY SUPPORTS NETWORK EVENT LOGGING IN A S
EPARATE
BUT IDENTICAL WAY TO PRIMOS OS EVENT LOGGING. EVENTS AR
E OBSERVED AND
WRITTEN OUT TO A DISK FILE NAMED 'NETREC' BY PRIMENET.
[TO ENABLE EVENT LOGGING TO DISK,
THE SYSTEM ADMINISTRATOR MUST CREATE AN EMPTY FILE IN TH
E COMMAND UFD
NAMED 'NETREC.' THIS FILE IS AN EXACT PARALLEL TO 'LOGR
EC.']
THE CODED ENTRIES IN NETREC MAY BE EXAMINED WITH THE EXT
ERNAL COMMAND
NETPRT. THE NETPRT SUBUFD CONTAINS THE SOURCE FOR THIS
EXTERNAL COMMAND
ALONG WITH THE COMMAND FILE NEEDED TO CREATE ITS EXECUTA
BLE MODULE.
IN ADDITION THE COMMAND FILE COPIES THE EXECUTABLE MODUL
E TO PRINET>CMDNCO
IN PREPARATION FOR COPYING TO THE NORMAL SYSTEM COMMAND
DIRECTORY.

NETCON - IN ORDER TO INCLUDE HIS SYSTEM IN A PRIMENET NETWORK, A
SYSTEM
ADMINISTRATOR MUST CREATE A NETWORK CONFIGURATION FILE B
EFORE PRIMOS COLD START.
WE PROVIDE THE EXTERNAL PROGRAM 'NETCFG' TO SIMPLIFY THE
SPECIFICATION
OF THE NETWORK CONFIGURATION. THE SOURCE AND COMMAND FI
LES FOR NETCFG
ARE INCLUDED IN THE NETCON SUBUFD. AS IN THE CASE OF NE
TPRT, THE BUILDING
COMMAND FILE COPIES THE EXECUTABLE MODULE FOR THE EXTERN
AL COMMAND TO
PRINET>CMDNCO IN PREPARATION FOR ITS COPY TO THE SYSTEM
COMMAND UFD.

NETEST - AS A DIAGNOSTIC TOOL, THE NETEST SUBUFD CONTAINS MODULES
THAT FORM
A 'TESTER' FOR PRIMENET. BY RUNNING A MASTER/SLAVE PAIR
, A SYSTEM ADMINISTRATOR
CAN RUN TESTS ON THE IPCF PRIMITIVES ACROSS ANY CONFIGUR
ED NETWORK PATHS.

VNETLB - ANY PRIMOS USER LOGGED INTO A SYSTEM RUNNING PRIMENET MA
Y USE
THE INTER-PROCESS COMMUNICATION FACILITY (IPCF) TO PERFO
RM PROGRAM-TO-PROGRAM
DATA EXCHANGE ACROSS THE NETWORK. IPCF CONSISTS OF A SE

T OF NETWORK
'PRIMITIVES' WHICH MAY BE CALLED FROM ANY USER-LEVEL V-M
ODE PROGRAM.
THE LIBRARY THAT SUPPORTS THE CALLING OF THESE PRIMITIVE
S IS CALLED
'VNETLB.' THE SOURCE AND COMMAND FILE FOR VNETLB ARE I
N THE SUBUFD
VNETLB. THE COMMAND FILE COPIES THE LOADABLE VERSION OF
THE LIBRARY
INTO PRINET>LIB FOR EVENTUAL COPYING TO THE SYSTEM LIB U
FD.

LIB - THE LIB SUBUFD CONTAINS THE BINARY MODULE OF THE V-MODE
NETWORK PRIMITIVES LIBRARY VNETLB. ONCE THE COMMAND FIL
E IN THE VNETLB
SUBUFD IS RUN, THE LIB SUBUFD MAY BE FUTIL UFDCPY-ED TO
THE SYSTEM LIB UFD.

CMDNCO - THE CMDNCO SUBUFD CONTAINS THE EXECUTABLE MODULES FOR TH
E EXTERNAL
PRIMENET COMMANDS NETCFG AND NETPRT. ONCE THE COMMAND F
ILES IN THE NETCON AND
NETPRT SUBUFDS ARE RUN, THE CMDNCO SUBUFD MAY BE FUTIL U
FDCPY-ED TO THE
SYSTEM COMMAND UFD.

SYSCOM - THE SYSCOM SUBUFD CONTAINS THE INSERT FILE THAT DEFINES
THE KEYS
AND CODES USED WITH THE IPCF NETWORK PRIMITIVES. THIS S
UBUFD MAY BE
FUTIL UFDCPY-ED TO THE SYSTEM SYSCOM UFD.

INFO - THE INFO SUBUFD CONTAINS THE RUNOFF FILES THAT DESCRIBE
THE
FACILITIES AVAILABLE TO USERS OF PRIMENET SYSTEMS. THES
E INCLUDE A COMPLETE
DESCRIPTION OF THE IPCF PRIMITIVE CALLING SEQUENCES AND
DIRECTIONS ON THE
USE OF NETPRT, NETCFG AND THE NETWORK TESTER.
IN ADDITION, THE INFO SUBUFD CONTAINS A DESCRIPTION OF T
HE 'CHANGES' TO
PRIMENET SINCE THE LAST PRIMOS REV.
FOR FURTHER CLARIFICATION
ON ANY OF THE TOPICS DESCRIBED ABOVE, REFER TO THE APPRO
PRIATE INFO FILE.

2.5.1 CHANGES TO PRIMENET SINCE 16.4

PRIMENET NODE CONTROLLER (PNC)

AT REV. 16.4 THE HARDWARE DEVICE ADDRESS OF THE PRIMENET NODE CON
TROLLER
(PNC) WAS CHANGED FROM ITS ORIGINAL ADDRESS OF '61 TO '07. THIS
CHANGE

IS A PERMANENT ONE. PNC'S AND PNC SUPPORTING PRIME SOFTWARE WILL USE
DEVICE ADDRESS '07 FOR ALL PRIMOS REVISIONS LATER THAN 16.4.

INTERACTIVE TERMINAL SUPPORT (ITS)

AT REV 16.4, A USER REMOTELY LOGGING IN COULD BE CONNECTED TO A USER-PROCESS JUST AS IT WAS LOGGING OUT A PREVIOUS ITS USER.

WHEN THE LOGOUT WAS COMPLETE, THE SECOND USER WAS IMMEDIATELY DISCONNECTED. USERS ARE NOW PREVENTED FROM CONNECTING TO PROCESSES THAT ARE LOGGING OUT.

ALSO REFER TO CHAGES MADE TO THE IPCF DESCRIBED IN THIS SECTION. A NEW KEY HAS BEEN ADDED TO THE PRIMENET STATUS GATHERING ROUTINE TO RETURN INFORMATION ABOUT ITS NETWORK CONNECTIONS.

FILE ACCESS MANAGER (FAM)

- NOW IN V-MODE.

- MAXIMUM MESSAGE SIZE BETWEEN FAMS IS NOW 267 WORDS (UP FROM 248).

- FAM SIZES ITS FREE BLOCK POOL ACCORDING TO THE NUMBER OF FAM ENABLED MODES.

- FAM MESSAGES FORMERLY PRINTED ONLY ON THE SYSTEM CONSOLE NOW ALSO APPEAR IN THE COMOUTPUT FILE FAMOUT.

- A NEW A-REGISTER BIT (:1000) HAS BEEN DEFINED. WHEN :1000 IS SET IN THE A-REGISTER WORD IN THE LINE SEG #FAM 1/1XXX, THE FAM RUNS IN "NOISY" MODE, AND MAY PRINT THE FOLLOWING MESSAGES ON THE SYSTEM CONSOLE.

****FAM UNEXPECTED CC$ = NNNNN
****FAM UNEXPECTED CALL FROM <NODE>NNNNNNNNN

NORMALLY, PRINTING OF THESE TWO MESSAGES IS SURPRESSED.

SYNCHRONOUS NETWORK (LEVEL 2) MODIFICATIONS

LEVEL 2 SYNCHRONOUS NETWORKS HAVE BEEN MODIFIED TO SUPPORT THE MDLC CONTROLLER. SINCE SLCINI PLACES THE MODEL NUMBER OF THE CONTROLLER IN AN ACCESSIBLE TABLE, PRSMLC WILL DETERMINE IF THE CONTROLLER CAN SUPPORT THE CONFIGURED PROTOCOL. IF NOT THE LINE IS DISABLED. THE MODEL NUMBERS SUPPORTING EACH PROTOCOL ARE (IN OCTAL):

BISYNC - 0 (HSSMLC), 5646, 5647, 5650
HDLC - 5646, 5651, 5653

AT REV 17 THE POLL/FINAL BIT IS NOW SUPPORTED. THIS WILL ELIMINATE A NUMBER OF PRSMLC RESETS AND PREVENT OUT OF SEQUENCE

LEVEL 3 MESSAGES FROM OCCURRING. PRSMLC WILL NO LONGER ASSUME A DEFAULT FOR THE LINE CONFIGURATION WORD BUT WILL REQUIRE THAT IT BE PROVIDED BY THE CONFIGURATOR. THE DEFAULT DATA SET CONTROL WORD HAS BEEN CHANGED TO '31403. THE DATA SET ORDER IS NOW DATA TERMINAL READY (DTR) AND REQUEST TO SEND (RTS). THE TRANSMIT AND RECEIVE DATA SET PATTERNS ARE BOTH NOW DATA SET READY (DSR) AND CLEAR TO SEND (CTS).

NETPRT

MORE NETWORK EVENTS ARE LOGGED FOR NETPRT. FOR THE COMPLETE LIST OF THESE ALONG WITH THE DESCRIPTION OF NETPRT SEE THE NETPRT DOCUMENT OR READ PRINET>INFO>RNETPRT.

NETCFG

ADD SUPPORT FOR MDLC PROTOCOL AND DROP QUESTIONS FOR SMLC LOGICAL TO PHYSICAL LINE MAPPING. FOR A COMPLETE DESCRIPTION OF NETCFG SEE THE NETCFG DOCUMENT OR READ PRINET>INFO>RNETCFG.

INTER-PROCESS COMMUNICATIONS FACILITY (ICF)

THE X$STAT DOCUMENTATION HAS BEEN EDITED TO CLARIFY THE USE OF SEVERAL OF THE ORIGINAL KEYS. IN ADDITION, A NEW KEY, XI$RLG HAS BEEN DEFINED TO RETURN INFORMATION TO REMOTE LOGIN USERS ABOUT THEIR OWN ITS CIRCUITS. FOR COMPLETE DETAILS SEE THE IPCF DOCUMENT OR READ PRINET>INFO>RIPCF.

SYSCOM

THE X$KEYS FILE IN PRINET>SYSCOM HAS BEEN CHANGED TO INCLUDE A DEFINITION FOR THE NEW X$STAT KEY XI$RLG.

2.6 NEW PROCESSOR SUPPORT

REVISION 17 SUPPORTS THE NEW P750 CENTRAL PROCESSOR AS WELL AS THE COMPANION WIDE WORD MEMORIES AND BURST MODE DISK CONTROLLER. THE NEW PROCESSOR AND ASSOCIATED HARDWARE IS FULLY COMPATIBLE WITH ALL USER RING SOFTWARE.

2.7 NEW SYNCHRONOUS LINE CONTROLLER

A NEW CONTROLLER FOR SYNCHRONOUS COMMUNICATIONS, MDLC, IS SUPPORTED AT REV. 17. THIS CONTROLLER, IN ITS VARIOUS VERSIONS, SUPPORTS BISYNC, SDLC, PACKET, AND SEVERAL RJE PROTOCOLS.

2.8 NEW CARD PROCESSOR

BEGINNING AT REV. 17, PRIME'S STANDARD CARD PROCESSOR WILL BE THE DECISION DATA MODEL 8010. PREVIOUSLY, THE STANDARD CARD PROCESSOR WAS THE DOCUMATION MODEL LC-50.

CURRENTLY PRIMOS SUPPORTS TWO UNIT RECORD CONTROLLERS (URC). ANY COMBINATION OF TWO OF THE FOLLOWING CONTROLLERS CAN BE RUN ON A PRIME SYSTEM.

CONTROLLER	RUNS:	DEVICE
URC1	2 LINE-PRINTERS	STANDARD
	1 CARD READER	DOCUMATION
URC2	1 LINE-PRINTER	STANDARD
	1 CARD PROCESSOR	DOCUMATION LC-50
URC3	1 CARD PROCESSOR	DECISION DATA 8010

THE 8010 MAY REPLACE THE DOCUMATION LC-50 CARD PROCESSOR, THOUGH THE FOLLOWING CONFIGURATION DIFFERENCES SHOULD BE NOTED.

SYSTEMS UPGRADING TO THE 8010 FROM AN LC-50 WILL BE ABLE TO RUN ONE LESS LINE-PRINTER.

SYSTEMS WITH A UNIT RECORD CONTROLLER 2 (URC2) RUNNING A DOCUMATION LC-50 CARD PROCESSOR/STANDARD LINE-PRINTER CAN UPGRADE TO THE MODEL 8010, THOUGH TWO CONTROLLERS MUST BE USED INSTEAD OF ONE.

NOTE THAT DUE TO DEVICE I.D.S (BOTH SOFTWARE AND HARDWARE) BEING DEPENDENT ON THE CONTROLLER NUMBER (NOT TYPE, BUT CONTROLLER NUMBER 1 OR 2) SOFTWARE REFERENCES WILL CHANGE WITH THE ABOVE CHANGE IN CONFIGURATION.

NOTE THAT THE LC-50 IS STILL SUPPORTED AND THAT URC3 IS NOT SUPPORTED ON PRIME 300 SYSTEMS.

2.9 RING-0 STACK OVERFLOW DETECTION

A CHECK HAS BEEN INSTALLED IN THE PAGE-FAULT HANDLER TO DETECT A PAGE-FAULT WITH ANY VIRTUAL ADDRESS IN THE RING-0 STACK SEGMENT, I.E., SEGMENT 6000 (OCTAL). THE OCCURRENCE OF THIS EVENT IS AN INDICATION OF A FATAL SYSTEM ERROR. IF THIS EVENT OCCURS, THE MACHINE WILL HALT WITH A PB THAT IS ONE LOCATION AFTER THE LABEL "R0OVR_". THIS CHECK WAS INSTALLED TO HELP PRESERVE INFORMATION ABOUT THE ERROR AND TO HALT THE MACHINE.

NOTE: A WARM START MAY BE PERFORMED. THE FAULTING PROCESS WILL RECEIVE A FATAL ERROR AND BE REINITIALIZED.

2.10 AINIT DIRECTIVE PROCESSING CHANGE

THE AINIT DIRECTIVE PROCESSING HAS BEEN CHANGED TO GIVE AN ERROR MESSAGE WITH UNRECOGNIZED DIRECTIVES AND TO CONTINUE PROCESSING. PREVIOUSLY, THE SEQUENCE WAS TO PRINT THE ERROR MESSAGE AND TO DISCONTINUE FURTHER PROCESSING. THIS CHANGE WAS DONE TO ALLOW DOWNWARD COMPATIBILTY OF CONFIGURATION FILES.

2.11 AINIT MEMORY SIZE SPECIFICATION

THE AINIT PROCESSING WILL NOW PRINT THE AMOUNT OF MEMORY AVAILABLE IN BYTES INSTEAD OF WORDS. THE MESSAGE INDICATES THAT THE VALUE IS NOW CLEARLY BYTES.

2.12 MAG TAPE ASSIGNMENT MECHANISM

IMPORTANT NOTE: AN ASSIGN COMMAND WHICH REQUIRES THE ASSISTANCE OF THE SYSTEM OPERATOR CAN NOT BE ENTERED AT THE SYSTEM CONSOLE BECAUSE THE REPLY COMMAND HAS TO BE ENTERED AT THE SYSTEM CONSOLE.

2.12.1 OVERVIEW

THE FOLLOWING SECTION DESCRIBES A NEW ASSIGNMENT MECHANISM FOR MAGNETIC TAPE DRIVE UNITS. IT ENABLES A USER TO ASSIGN PHYSICAL TAPE DRIVE WITH LOGICAL TAPE UNIT NUMBER. A USER MAY REQUEST SYSTEM OPERATOR'S ASSISTANCE IN ASSIGNING A DRIVE, AND/OR MOUNTING A TAPE. SYSTEM OPERATOR MAY ASSIGN A DRIVE TO A USER BASED ON THE ATTRIBUTES SUPPLIED BY THE USER. (TRACK, DENSITY, ETC.) THIS MECHANISM PROVIDES A SIMPLE (POSSIBLY TEMPORARY) FIRST PASS SOLUTION TO THE GENERAL RESOURCE ASSIGNMENT PROBLEM. THIS ASSIGNMENT MECHANISM ONLY APPLIES TO MAGNETIC TAPE DRIVES. IT DOES NOT AFFECT THE CURRENT ASSIGNMENT MECHANISM FOR OTHER PERIPHERAL DEVICES SUCH AS CARD READERS AND PRINTERS.

2.12.2 OBJECTIVES

A) PROVIDE A LEVEL OF INDIRECTION (MAPPING) BETWEEN PHYSICAL AND LOGICAL DEVICE NUMBERS. A USER MAY ARBITRARILY ASSIGN A TAPE DRIVE BY SUPPLYING THE DRIVE CHARACTERISTICS. IF THE PARTICULAR DRIVE IS IN USE, ANOTHER DRIVE OF THE SAME CHARACTERISTICS CAN BE USED.

B) ALLOW, AND OPTIONALLY REQUIRE, OPERATOR INTERVENTION IN TAPE UNIT ASSIGNMENT.

C) PROVIDE A MEANS BY WHICH A USER MAY REQUEST OPERATOR ASSISTANCE IN MOUNTING A TAPE OR DOING OTHER TAPE OPERATIONS.

2.12.3 ASSIGN

ASSIGN [-TPID ID] [-TRACK] [-PROTECT]
[-DENSITY] [-WAIT]

PDN MIN ($0 \leq N \leq 7$), PHYSICAL DRIVE NUMBER.

LDN MIN ($0 \leq N \leq 7$), LOGICAL DRIVE NUMBER. THIS NUMBER, IF SUPPLIED IN THE ASSIGN COMMAND, WILL BE USED AND MAPPED INTO THE PHYSICAL DRIVE NUMBER 'PDN' IN SUBSEQUENT TAPE

	OPERATIONS BY CALLS TO T$MT. EACH USER HAS 8 LOGICAL DRIVE NUMBERS AT HIS DISPOSAL.
MTX	MTX INDICATES 'ANY DRIVE'. IF THIS IS GIVEN INSTEAD OF THE PHYSICAL DRIVE NUMBER PDN, IT MEANS THAT THE USER JUST WANTS ANY ONE OF THE 8 PHYSICAL DRIVES. ASSIGNMENT OF A DRIVE IS BASED ON SUBSEQUENT INFORMATION (IF ANY) IN THE ASSIGN COMMAND. AN ALIAS (LOGICAL DRIVE NUMBER LDN) MUST BE GIVEN WHEN USING MTX.
TPID	CONTROL ARGUMENT FOR TAPE OR VOLUME IDENTIFIER.
ID	A LIST OF TOKENS (OR WORDS) THAT IDENTIFIES A PARTICULAR REEL OF TAPE TO BE MOUNTED, AND/OR THE TYPE OF TAPE DRIVE. A TOKEN IS ANY STRING OF CHARACTERS NOT CONTAINING A DELIMITER. (CURRENT DELIMITERS ARE COMMA, SPACE, A NEWLINE, AND '/*') NO TOKEN (OR WORD) IS ALLOWED TO BEGIN WITH A '-'. A TOKEN WHICH BEGINS WITH A '-' WILL BE INTERPRETED AS THE NEXT CONTROL ARGUMENT.
TPACK	9TRK - 9 TRACK TAPE DRIVE 7TRK - 7 TRACK TAPE DRIVE
PROTECT	RINGON - READ AND WRITE. RINGOFF - READ ONLY (WRITE PROTECT).
DENSITY	6250 6250BPI 1600 1600BPI 800 800BPI
WAIT	QUEUES THE ASSIGNMENT IF THE DEVICE IS IN USE.

A USER MAY ASSIGN A MAGNETIC TAPE DRIVE BY ISSUING THE PRIMOS COMMAND ASSIGN, ALONG WITH A PHYSICAL DRIVE NUMBER. TO MAP A PHYSICAL DRIVE NUMBER TO A DIFFERENT LOGICAL NUMBER, THE OPTION '-ALIAS' MAY BE SPECIFIED. IF THE KEYWORD 'MTX' IS SPECIFIED INSTEAD OF THE PHYSICAL DRIVE NUMBER, THEN ANY DRIVE WHICH MEETS THE REQUIREMENTS SPECIFIED IN THE SUBSEQUENT OPTIONS IS ASSIGNED. THE DEGREE OF OPERATOR INTERVENTION IN THE TAPE ASSIGNMENT PROCESS DEPENDS ON THE MODE OF OPERATION, WHICH IS DESCRIBED IN THE FOLLOWING SECTION.

2.12.4 MODES OF OPERATION

SOME TAPE OPERATIONS REQUIRE THE ASSISTANCE OF THE OPERATOR, AND SOME DO NOT. THE ASSIGN COMMAND AT REV. 16 ALLOWS USERS TO ASSIGN TAPE DRIVES AT THEIR TERMINALS WITHOUT ANY OPERATOR INTERVENTION.

A RUN TIME SETTABLE SWITCH IS NOW AVAILABLE TO THE SYSTEM

OPERATOR TO DEFINE A MODE OF OPERATION WHICH PERMITS TAPE ASSIGNMENT AT THE PRESENCE AND ABSENCE OF THE OPERATOR. MODE OF OPERATION IS DEFINED BY A NEW OPERATOR COMMAND SETMOD WHICH CAN ONLY BE ENTERED AT THE SYSTEM CONSOLE.

SETMOD [-OPERATOR -NOASSIGN]

A) OPERATOR

IN THIS MODE, ALL TAPE ASSIGNMENTS MUST BE ACKNOWLEDGED BY THE OPERATOR. A USER IS NOT ALLOWED TO ASSIGN TAPE DRIVES AT THE TERMINAL WITHOUT INTERVENTION BY THE OPERATOR. ALL TAPE ASSIGNMENTS WILL RESULT IN REQUEST MESSAGES TO THE OPERATOR, WHO WILL RESPOND TO THE REQUEST USING THE REPLY COMMAND. (DESCRIBED IN LATER SECTION)

B) USER

IN THIS MODE, A USER MAY ASSIGN TAPE DRIVES AT THE TERMINAL, AND MAP PHYSICAL DRIVE NUMBER TO LOGICAL DRIVE NUMBER WITHOUT INTERVENTION OF THE OPERATOR. THE USER CAN ALSO REQUEST OPERATOR ASSISTANCE IN ASSIGNING AND MOUNTING A TAPE USING THE '-TPID', '-TRACK', '-PROTECT', AND '-DENSITY' OPTIONS. THE SYSTEM IS IN USER MODE WHEN COLD STARTED. THIS IS THE DEFAULT MODE.

C) NOASSIGN

IN THIS MODE, TAPE DRIVES CANNOT BE ASSIGNED. A USER MAY NOT ASSIGN A TAPE DRIVE AT THE TERMINAL, AND NO OPERATOR IS AVAILABLE TO ASSIST THE TAPE ASSIGNMENT OR MOUNTING OF A TAPE. THIS MODE OF OPERATION MAY BE USED IN AN ENVIRONMENT WHERE THE SYSTEM OPERATOR HAS TOTAL CONTROL OVER TAPE DRIVE ASSIGNMENT TO INDICATE TEMPORARY ABSENCE OF THE OPERATOR.

FROM A USER POINT OF VIEW, THERE IS NO DIFFERENCE IN THE USAGE OF THE ASSIGN COMMAND IN ANY OF THESE THREE MODES.

2.12.5 MESSAGE TO THE OPERATOR

IN 'OPERATOR' MODE, ALL ASSIGN COMMANDS WILL RESULT IN MESSAGES TO THE OPERATOR. IN 'USER' MODE, THERE WILL ONLY BE MESSAGES TO THE OPERATOR WHEN THE USER SPECIFIES OPTIONS SUCH AS '-TPID', '-TRACK', AND 'PROTECT' IN THE ASSIGN COMMAND.

THE MESSAGE IS SENT TO THE OPERATOR BY THE ASSIGN PROGRAM. IT SERVES TWO PURPOSES:

A) REQUEST ASSISTANCE OF THE OPERATOR TO ASSIGN A NEEDED TAPE DRIVE.

B) REQUEST ASSISTANCE OF THE OPERATOR TO MOUNT A TAPE ON A DRIVE.

THE FORMAT OF THE MESSAGE THAT APPEARS AT THE SYSTEM CONSOLE IS AS FOLLOWS:

```
***** MAGTAPE REQUEST *****
FROM <USRNAM> (<USRNUM>):  <MESSAGE>
```

<USRNAM> IS THE NAME OF THE USER REQUESTING THE TAPE ASSIGNMENT.

<USRNUM> IS THE USER NUMBER .

<MESSAGE> IS THE SAME AS THE COMMAND LINE WHICH THE USER TYPED LESS THE COMMAND WORD ASSIGN.

THIS MESSAGE IS REPEATED PERIODICALLY UNTIL THE OPERATOR RESPONDS USING THE REPLY COMMAND. THE MESSAGE REPEAT FREQUENCY IS SETTABLE USING THE REPLY COMMAND WITH THE '-REPEAT' OPTION. THE DEFAULT MESSAGE REPEAT FREQUENCY IS 3 MINUTES. (180 SECONDS)

2.12.6 REPLY

A NEW COMMAND IS CREATED TO ESTABLISH COMMUNICATION BETWEEN THE ASSIGN PROGRAM AND THE SYSTEM OPERATOR. THIS COMMAND CAN ONLY BE ENTERED AT THE SYSTEM CONSOLE. (THIS COMMAND MAY BE DEVELOPED INTO A MORE GENERAL PURPOSE OPERATOR/USER COMMUNICATION FACILITY IN THE FUTURE.) THE FORMAT OF THE COMMAND IS AS FOLLOWS:

```
REPLY -USRNUM -TAPE [ RESEND  GO
```

USRNUM	USER NUMBER.
TAPE	INDICATING THAT THIS IS A REPLY TO TAPE OPERATION REQUESTS.
RESEND	REQUEST RESENDING OF ASSIGN'S MESSAGE.
ABORT	THIS IS USED WHEN NO TAPE DRIVE IS ASSIGNABLE, OR THE OPERATOR WANTS TO ABORT THE REQUEST FOR ANY REASON. (DRIVE NOT AVAILABLE, TAPE NOT FOUND, ETC.)
GO	THIS IS A REPLY TO THE REQUEST OF MOUNTING A TAPE AND ASSIGNING A PHYSICAL DRIVE. IT INDICATES THE REQUESTED TAPE HAS BEEN MOUNTED ON THE SPECIFIED DRIVE, OR THE PHYSICAL DRIVE WHICH THE USER PICKED IS ASSIGNED AND READY.
PDN	PHYSICAL DRIVE NUMBER. THIS IS A REPLY TO THE REQUEST OF THE ASSIGNMENT OF A TAPE DRIVE. IT INDICATES THAT A DRIVE HAS BEEN ASSIGNED TO THE USER, AND THE NUMBER OF THE ASSIGNED DRIVE IS 'PDN'. THIS REPLY IS USED IF USER SPECIFIED 'MTX' IN THE ASSIGN COMMAND.

THE OPERATOR MAY REQUEST RESENDING OF ALL OUTSTANDING TAPE REQUESTS. HE CAN ACCOMPLISH THIS BY USING THE '-TAPE RESEND' OPTION WITHOUT SUPPLYING THE USER NUMBER OR SIMPLY USING THE

'-ALL' OPTION. THE FORMATS ARE AS FOLLOWS:

```
REPLY -TAPE RESEND      OR
REPLY [ -USRNUM
```

THE OPERATOR HAS THE OPTION OF SETTING THE MESSAGE REPEAT FREQUENCY. THE MESSAGE REPEAT PERIOD IS THE TIME BETWEEN THE PRINTING OF TWO SUCCESSIVE MESSAGES ON THE SYSTEM CONSOLE FROM THE SAME REQUEST.

```
REPLY -REPEAT <SECONDS>
```

<SECONDS> A DECIMAL NUMBER SPECIFING THE MESSAGE REPEAT PERIOD IN SECONDS.

2.12.7 UNASSIGN

```
UNASSIGN MTN

UNASSIGN -ALIAS MTN
```

IF THE OPTION '-ALIAS' IS NOT SUPPLIED, 'MTN' IS THE PHYSICAL DRIVE NUMBER. THE USER UNASSIGNS THE DRIVE BY PHYSICAL DEVICE NUMBER. IF THE OPTION '-ALIAS' IS SUPPLIED, 'MTN' IS THE LOGICAL DRIVE NUMBER.

THE SYSTEM OPERATOR HAS THE PRIVILEGE TO UNASSIGN ANY DRIVE EVEN HE DOES NOT OWN THE DRIVE. HOWEVER, HE MUST UNASSIGN THE DRIVE BY PHYSICAL DEVICE NUMBER. HE CANNOT UNASSIGN THE DRIVE BY LOGICAL DEVICE NUMBER (USING THE '-ALIAS' OPTION) IF HE DOES NOT OWN THE DRIVE BECAUSE A LOGICAL DEVICE NUMBER IS NOT UNIQUE (TWO USERS MAY OWN TWO DIFFERENT PHYSICAL DRIVES WITH THE SAME LOGICAL DEVICE NUMBER).

2.12.8 USAGE

THE FOLLOWING IS A LIST OF EXAMPLES ILLUSTRATING POSSIBLE USE OF THE NEW ASSIGN MAGTAPE COMMAND.

```
ASSIGN MTX -AL MT0 -TPID ABC -9TRK -RINGON -6250BPI
```

ASSIGN ANY DRIVE, 9 TRACK, READ AND WRITE (NO PROTECT), AND 6250 B.P.I.. ALSO MOUNT TAPE ABC ON THE DRIVE IF A DRIVE IS ASSIGNED. MT0 IS THE NUMBER WHICH THE USER WILL USE TO REFER TO THE ASSIGNED DRIVE. ALL SUBSEQUENT TAPE OPERATIONS OF THE ASSIGNED DRIVE BY CALLS TO T$MT WILL USE MT0 AS THE UNIT NUMBER. IF FOR ANY REASON NO DRIVE IS ASSIGNABLE, AN ERROR MESSAGE WILL BE RETURNED TO THE USER.

ASSIGN MT1 -AL MT2 -TPID XYZ

ASSIGN PHYSICAL DRIVE MT1 AND MAP PHYSICAL DRIVE NUMBER MT1 INTO LOGICAL NUMBER MT2. LOGICAL DRIVE NUMBER MT2 WILL BE USED IN SUBSEQUENT TAPE OPERATIONS TO PHYSICAL DRIVE MT1. ALSO MOUNT TAPE XYZ ON PHYSICAL DRIVE MT1 IF IT IS ASSIGNED.

ASSIGN MT0

ASSIGN PHYSICAL DRIVE MT0. NO MAPPING IS REQUESTED. IF THE DRIVE IS ASSIGNED, ALL SUBSEQUENT TAPE OPERATIONS TO THE ASSIGNED DRIVE WILL USE MT0 AS THE TAPE DRIVE NUMBER.

ASSIGN MTX -AL MT1

ASSIGN ANY DRIVE. USER WILL USE MT1 AS THE NAME OF THE DRIVE IF ONE IS ASSIGNED. NO MOUNT TAPE REQUEST.

ASSIGN MTX

ILLEGAL COMMAND SYNTAX. USER REQUEST TO ASSIGN A DRIVE, BUT NO LOGICAL NAME IS GIVEN.

2.13 AMLC ENHANCEMENTS

TWO ENHANCEMENTS HAVE BEEN MADE TO PRIMOS AMLC PROCESSING REVISION 17. THE FIRST IS THE INSTALLATION OF A DISCONNECT WATCHDOG TIMER WHICH IS USED TO FORCE DISCONNECTION OF INACTIVE TELEPHONE LINES. THE SECOND IS THE AVAILABILITY OF NEW PROTOCOL HANDLERS USED TO MAP LOWER-CASE TO UPPER-CASE.

THE DISCONNECT WATCHDOG TIMER IS ENABLED WITH THE "AMLTIM" CONFIG DIRECTIVE DESCRIBED LATER IN THIS DOCUMENT. THE FEATURE MONITORS THOSE TERMINAL, NOT (I.E., NOT ASSIGNABLE) LINES FOR WHICH CARRIER IS ACTIVE. WHEN CARRIER GOES ACTIVE ON ONE OF THESE LINES, THE AMLC PROCESS WILL CHECK THE LINE TO SEE IF THE PROCESS ASSOCIATED WITH THE LINE IS LOGGED-IN. IF THE LINE HAS FAILED TO BECOME ESTABLISHED WITH A LOGGED-IN PROCESS IN THE "GRACE" TIME, AS GIVEN IN THE "AMLTIM" DIRECTIVE, THE DATA TERMINAL READY (DTR) SIGNAL TO THE MODEM IS FORCED INACTIVE. IN MOST INSTALLATIONS, THIS WILL CAUSE THE TELEPHONE LINE INTO THE MODEM TO BECOME DISCONNECTED. THE DTR SIGNAL REMAINS INACTIVE FOR THE TIME PERIOD GIVEN BY THE "AMLTIM" DIRECTIVE. THE USE OF THIS FEATURE RELIES ON A TELEPHONE EXCHANGE WHICH WILL FREE THE TELEPHONE LINE IF DISCONNECTION OCCURS.

THE NEW PROTOCOL HANDLERS ARE USED TO MAP LOWER-CASE ALPHABETIC CHARACTERS TO UPPER-CASE ALPHABETIC CHARACTERS. THE PROTOCOLS SHOULD BE USED TO AVOID SENDING LOWER-CASE OUTPUT TO TERMINALS WHICH CANNOT PRINT THESE CHARACTERS. THE PROTOCOLS ARE ACTIVATED FOR EACH AMLC LINE BY USE OF THE "AMLC" OPERATOR COMMAND. SPECIFYING "TTYUPC" FOR THE PROTOCOL TYPE WILL ENABLE THE

STANDARD-SPEED OUTPUT PROTOCOL WITH LOWER- TO UPPER-CASE MAPPING AND THE STANDARD INPUT PROTOCOL HANDLING. SPECIFYING "TTYHUP" FOR THE PROTOCOL TYPE WILL ENABLE THE HIGH-SPEED OUTPUT PROTOCOL WITH LOWER- TO UPPER-CASE MAPPING AND THE STANDARD INPUT PROTOCOL HANDLING. THE PROTOCOL CAN ONLY BE ESTABLISHED BY THE OPERATOR AND NOT BY THE USER.

2.14 NEW LOW-SPEED BUFFERING MECHANISM

THE LOW-SPEED BUFFERING MECHANISM IN PRIMOS HAS BEEN ENHANCED TO ALLOW MORE SPACE FOR THE TOTAL ACCUMULATION OF TERMINAL BUFFERS. THE PREVIOUS RESTRICTION OF ALLOWING ONLY 32K WORDS TOTAL BUFFER SPACE HAS BEEN CHANGED TO ALLOW 64K WORDS OF TOTAL BUFFER SPACE. THE INSTALLATION MAY NOW SPECIFY MORE OR LARGER BUFFERS THAN WERE PREVIOUSLY POSSIBLE. EACH INDIVIDUAL BUFFER IS RESTRICTED TO A MAXIMUM SIZE OF 4K WORDS. THE AMLC DMQ BUFFERS, WHICH PREVIOUSLY FOLLOWED THE TERMINAL BUFFERS IN THE SAME SEGMENT, HAVE BEEN MOVED TO A SEPARATE SEGMENT. THEIR SIZE IS NOW INDEPENDENT OF THE TERMINAL BUFFER SPACE USED.

2.15 NEW SYNCHRONOUS COMMUNICATIONS ENVIRONMENT

THERE ARE NEW TOOLS FOR DOING SYNCHRONOUS COMMUNICATIONS.

THE NEW ENVIRONMENT COMPRISES

-- A NEW SYNCHRONOUS "INTERRUPT" PROCESS,

-- A LIBRARY OF SUBROUTINE CALLS.

THE MAJOR NEW FEATURES ARE

-- DYNAMIC DMC ALLOCATION AND SEGMENT) WINDOWING

-- DYNAMIC "INTERRUPT SUB-PROCESS" CONNECTION

-- FREE STORAGE AND QUEUEING MECHANISMS

CURRENT SOFTWARE USING THE NEW TOOLS ARE SYNCHRONOUS NETWORKING AND THE T$SLC RAW DATA MOVER.

TWO NEW COMMUNICATIONS PROTOCOLS HAVE BEEN IMPLEMENTED WITHIN THE NEW ENVIRONMENT. THEY ARE BOTH DIRECTIONS OF AN IBM SYNCHRONOUS TERMINAL (I.E., 3271/3277 SUPPORT AND EMULATION).

WHY

THE CHANGES HAVE BEEN MADE TO SOLVE TWO MAJOR PROBLEMS IN THE PREVIOUS COMMUNICATIONS SUPPORT. THESE ARE:

-- HIGH CPU OVERHEAD

-- COSTLY MAINTENANCE AND MODIFICATION

THE SOLUTION IS TO SIMPLIFY THE CONSTRUCTION OF TAILORED COMMUNICATIONS PROTOCOL CODE AT ALL LEVELS. IN THIS WAY HIGH FREQUENCY EVENTS CAN BE RESPONDED TO "CLOSE TO HARDWARE", THUS LOWERING THE AMOUNT OF MEMORY TOUCHED BY THE EVENT. THIS REDUCES THE OVERHEAD ASSOCIATED WITH HIGH-FREQUENCY HARDWARE STATUS. TAILORED PROTOCOL CODE IS ALSO SIMPLER THAN GENERAL-PURPOSE CODE, THUS LOWERING MAINTENANCE AND EVOLUTIONARY COSTS.

THE CONSTRAINTS ARE THAT A GIVEN COMMUNICATIONS LINE MUST NOT BE DEDICATED TO A SINGLE PROTOCOL DURING THE COURSE OF A PRIMOS SESSION ("COLD-START" TO "SHUTDOWN"); THAT PROTOCOL TAILORING EXTEND TO THE INTERRUPT PROCESS; THAT THE INTERRUPT PROCESS BE ABLE TO ACCESS I/O BUFFERS; THAT SCHEDULING BE SIMPLIFIED. THESE CONSTRAINTS COMPEL A NUMBER OF FACILITIES:

NON-DEDICATED LINES REQUIRE:

-- DYNAMIC PROCESS CONNECTION TO A LINE

-- DYNAMIC DMC CHANNEL ALLOCATION

-- DYNAMIC SEGMENT 0 WINDOW OPERATIONS

TAILORED INTERRUPT PROCESSING REQUIRES

-- VARIABLE INTERRUPT PROCESS CONNECTION

-- A NEW SYNCHRONOUS DIM

SIMPLICITY AND/OR EFFICIENCY REQUIRE(S)

-- REMOVAL OF COMMUNICATIONS PROCEDURE FROM USER 1 CONSTRAINTS

-- MULTIPLE PROCESSES AT "MUX-DEMUX" JUNCTIONS OR TO LOCALIZE HIGH-FREQUENCY EVENTS

-- QUEUED PROCESS INTERFACE

A GENERALIZED TAILORED PROTOCOL HANDLER INCLUDES CODE EXECUTED AS PART OF THE SLCDIM CHEAP (INTERRUPT) PROCESS AND A (STOLEN) USER PROCESS WHICH RESPONDS TO SIGNALS FROM THE CHEAP PROCESS CODE. THERE ARE CURRENTLY THREE TAILORED PROTOCOL HANDLERS:

SLCCMP SERVICING T$SLC1 AND SMLCEX

SLCNET SERVICING PRSMLC

BSCMTR SERVICING BSCMAN

NEW ROUTINES PROVIDE DYNAMIC CONNECTION FACILITIES:

DMCDYN ALLOCATES DMC CHANNELS FROM THE COMMUNICATIONS POOL.

SOMAN PROVIDES MAPPING AND BACKING FOR THE COMMUNICATIONS SEGMENT 0 WINDOWS.

SLCRND LINKS CHEAP PROCESS CODE TO A LOGICAL LINE.

THERE IS ALSO A FAMILY OF QUEUEING ROUTINES.

2.16 THE BSCMAN COMMUNICATIONS UTILITY

BSCMAN IS A PROCESS WHICH WILL AID SOME OTHER PROCESS IN USING A BINARY SYNCHRONOUS LINK. THE PURPOSE OF BSCMAN IS TO IMPROVE EFFICIENCY AND MAINTAINABILITY FOR PRIME-TO-IBM STYLE COMMUNICATION PRODUCTS (BSC PROTOCOL ONLY, NOT SDLC).

MOST OF THE I/O FUNCTIONS FOR A LINK USING BSC CAN BE PUT INTO ONE PROCESS SERVING MULTIPLE LINES.

"MOST I/O FUNCTIONS" MEANS

-- CONTROLLER OPERATIONS

-- MODEM CONTROL

-- VIRTUAL TO PHYSICAL BUFFER MAPPING AND WIRING.

-- SYNTACTIC ANALYSIS

-- TIMING

THE PROFIT IN SUCH A MERGE IS FOURFOLD:

-- SHARED LOGIC

-- COMMON INTERFACE TO LOGIC

-- SIMPLIFIED INTERFACE TO LOGIC

-- FLEXIBLE SCHEDULING

THE LAST TWO ITEMS BEAR CLARIFICATION.

THE INTERFACE IS SIMPLER BECAUSE THE BSCMAN PROCESS CONTAINS ENOUGH LOGIC TO RESPOND TO LOW LEVEL PROTOCOL MESSAGES ON ITS OWN. SINCE THESE MESSAGES DO NOT TRANSFER DATA, THEY NEED NOT CONCERN THE USER OF THE BSC LINK.

SCHEDULING IS MORE FLEXIBLE BECAUSE THE BSCMAN PROCESS CAN SUSTAIN A LINK INDEPENDENTLY OF ITS DRIVING PROCESS BY SENDING LOW-LEVEL PROTOCOL MESSAGES. THIS MEANS THAT BSCMAN IS USUALLY THE ONLY PROCESS WHICH MUST BE SCHEDULED AT A HIGH PRIORITY. ALSO, THE PRIORITY OF VARIOUS PROCESSES DRIVING BSCMAN CAN BE ADJUSTED INDEPENDENTLY OF BSCMAN AND EACH OTHER.

2.17 BAD SPOT HANDLING FOR PAGING PARTITION

2.17.1 OVERVIEW

CURRENTLY, PRIMOS WRITES OVER ALL RECORDS (EXCEPT THE FIRST EIGHT) OF A PAGING PARTITION. IT IS THEREFORE NOT POSSIBLE TO RETAIN DEFECTIVE TRACK INFORMATION IN A PAGING PARTITION. MANY OF THE NEWER (AND LESS COSTLY) DISK MEMORIES CAN HAVE A GREATER NUMBER OF IMPERFECTIONS ON THEIR RECORDING SURFACES. FOR EXAMPLE, THE CARTRIDGE MODULE DEVICE (CMD) SPECIFICATIONS ALLOW UP TO TWO UNCORRECTABLE ERRORS TO EXIST ON EACH FIXED MEDIA RECORDING SURFACE. AT REVISION 17.1 OF PRIMOS IV, THE PRIMOS PAGING PARTITION IS ALLOWED TO RESIDE ON IMPERFECT DISK SURFACES (WITH UNCORRECTABLE ERRORS) BY PROVIDING A MECHANISM TO MAP OUT BAD DISK TRACKS ON THE PAGING PARTITION.

2.17.2 HANDLING BAD SPOTS

CURRENTLY, PRIMOS IS ABLE TO MAP BAD DISK TRACKS OUT OF A FILE SYSTEM DISK PARTITION. BAD DISK TRACK INFORMATION IS KEPT IN A FILE NAMED BADSPT. WHEN THE DISK IS CREATED, (PRIMOS COMMAND MAKE) OR WHEN FIXRAT IS RUN, DEFECTIVE TRACK INFORMATION IS READ FROM THE BADSPT FILE, AND DISK RECORDS AFFECTED BY THE DEFECTIVE TRACKS ARE MARKED AS USED IN THE DSKRAT FILE. THE MECHANISM USED IN HANDLING BAD DISK TRACKS ON PAGING PARTITION IS SIMILAR TO THE ONE DESCRIBED ABOVE. BAD DISK TRACK INFORMATION IS KEPT IN THE BADSPT FILE. DURING START UP, THE PRIMOS PRELOADER (PRIMOS) AND THE COLD START INITIALIZATION PROGRAM (AINIT) WILL USE THE BAD TRACK INFORMATION AS IN THE BADSPT FILE AND REASSIGN PAGE RECORD INDEXES IN SUCH A WAY TO AVOID BAD DISK TRACKS.

2.17.3 USAGE

PRIMOS WILL MAP OUT DEFECTIVE TRACKS ON THE PAGING PARTITION IF A BADSPT FILE WHICH CONTAINS THE BAD TRACKS INFORMATION EXISTS IN THE PARTITION.

2.17.3.1 SYSTEM COMMAND DEVICE IS EQUAL TO PAGING DEVICE

PRIMOS RECOGNIZES A SPLIT PARTITION BY NOTING THAT THE SYSTEM COMMAND DEVICE AND THE PAGING DEVICE ARE THE SAME. (COMDEV=PAGDEV) IF A BADSPT FILE EXISTS IN THE FILE SYSTEM PORTION OF THE PARTITION, IT WILL USE THE INFORMATION IN THE FILE TO AVOID USING PAGING DISK RECORDS WITHIN THE DEFECTIVE TRACKS DURING SYSTEM PRELOADING AND INITIALIZATION.

2.17.3.2 SYSTEM COMMAND DEVICE IS NOT EQUAL TO PAGING DEVICE

IN ORDER TO MAINTAIN BAD DISK TRACK INFORMATION, A PAGING DISK MUST BE SET UP AS A SPLIT PARTITION. (THE PAGING DISK WILL APPEAR TO PRIMOS AS ONE LOGICAL DISK WITH A 'FILE SYSTEM PARTITION' AND A 'PAGING PARTITION'.) THE FILE SYSTEM PORTION OF THIS PAGING DISK WILL CONTAIN THE DEFECTIVE TRACK INFORMATION, (STORED IN A SAVE FORMAT FILE BADSPT) AND THE PAGING PORTION OF THIS PAGING DISK WILL BE USED FOR NORMAL PAGING. THE PACK NAME OF THE PAGING DISK MUST BE NAMED 'PAGING' IN ORDER TO DISTINGUISH IT FROM OTHER FILE SYSTEM PARTITIONS.

TO MAKE SUCH A DISK, RUN THE PRIMOS COMMAND MAKE:

```
MAKE, REV. 17.0
BUILDING NEW PARTITION.
PHYSICAL DISK:
```

WHEN MAKE ASKS:

```
SPLIT DISKS?
```

THE ANSWER IS YES. THEN MAKE ASKS:

```
PAGING RECORDS (DECIMAL)
```

THE NUMBER OF PAGING RECORDS DEPENDS ON THE NUMBER OF RECORDS USED FOR THE FILE SYSTEM. RECORD 0 TO 15 CONTAIN THE BOOTSTRAP (BOOT), DISK RECORD AVAILABILITY TABLE (DSKRAT), MFD, CMDNCO, DOS, AND THE BADSPT FILE. THE FIRST 16 RECORDS ARE THEREFORE NOT USED FOR PAGING. HENCE, THE NUMBER OF PAGING RECORDS IS EQUAL TO THE TOTAL NUMBER OF RECORDS IN THE LOGICAL DISK MINUS 16 FILE SYSTEM RECORDS.

A GOOD WAY TO DETERMINE THE NUMBER OF RECORDS ON A LOGICAL DISK IS TO ASK MAKE TO CREATE AN UNSPLIT PARTITION AND NOTE THE NUMBER OF FILE RECORDS TYPED OUT BY MAKE. THEN, TYPE "NO" TO THE QUESTION "PARAMETERS OK?". AT THIS POINT A SPLIT PARTITION CAN BE SPECIFIED USING THE INFORMATION RECEIVED IN THE PREVIOUS STEP.

WHEN MAKE ASKS:

```
PACK NAME?
```

THE PACK NAME MUST BE 'PAGING'. IF A TRACK IS KNOWN TO BE BAD WHEN THE DISK IS CREATED, THE HEAD AND TRACK NUMBER OF THE BAD TRACK MAY BE ENTERED WHEN MAKE ASKS IF THERE IS ANY BAD SPOT ON DISK.

PRIMOS NOW SUPPOPTS A MAXIMUM OF 16 BAD SPOTS ON PAGING PARTITIONS. I.E. THE SUM OF BAD SPOTS ON THE PRIMARY PAGING PARTITION (PAGDEV) AND AND THE ALTERNATE PAGING PARTITION (ALTDEV) MUST NOT BE GREATER THAN 16.

2.17.4 NEW DOS SUPPORT

A NEW SVC HAS BEEN ADDED TO DOS TO ALLOW THE PRELOADER PRMLD TO ADD ADDITIONAL FILE SYSTEM PARTITIONS WHILE RUNNING UNDER DOS. THIS NEW SVC HAS BEEN INSTALLED IN REVISION 16.X OF DOS (*DOS64). PRIMOS, REV 17.1 WILL ONLY RUN WITH THE ABOVE VERSION OF DOS. IT WILL FAIL WITH PREVIOUS VERSIONS OF DOS. DOS, REV 16.8 IS COMPATIBLE WITH ALL PREVIOUS VERSIONS OF DOS AND CAN BE RUN WITH PRIMOS VERSIONS PRIOR TO REV. 17.1.

3 NEW USER CALLABLE SUBROUTINES

3.1 T$SLCO FOR MDLC SUPPORT

SMLC MODIFICATIONS

THE SMLC ROUTINES OF SLCINI (IN SLCDIM) AND T$SLC1 HAVE BEEN MODIFIED TO SUPPORT THE NEW MDLC CONTROLLER WHICH CAN SUPPORT MULTIPLE HSSMLC ALTHOUGH THE HSSMLC WILL STILL BE SUPPORTED. AT INITIALIZATION SLCINI INPUTS THE DEVICE ID OF THE CONTROLLER. FROM THIS ID, SLCINI CAN DETERMINE THE ROTOCOLS PRESENT. IF THE CONTROLLER IS PRESENT, CERTAIN BITS OF THE ID WORD INDICATE WHETHER THE CONTROLLER IS AN MDLC OR AN HSSMLC. IF IT IS AN MDLC, SLCINI CAN DETERMINE THE MODEL NUMBER. THE ID WORD FOR THE HSSMLC IS THE SAE FOR ALL MODELS. SLCINI TRANSLATES THE ID WORD INTO A MODEL NUMBER, THE OCTAL, WHICH IS PLACED IN A TABLE FOR ACCESS BY T$SLC1 OR THE NETWORKS' PRSMLC. A LIST OF MODEL NUMBERS AND THE ROTOCOLS THEY SPPORT FOLLOWS.

MODEL NUMBER (OCTAL)	PROTOCOLS
0	HSSMLC
5646	BISYNC AND HDLC
5647	BISYNC AND PACKET
5650	BISYNC AND 1004/UT200/7020
5651	HDLC AND 1004/UT200/7020
5652	PACKET AND 1004/UT200/7020
5653	HDLC AND PACKET

SO THAT A USER PROGRAM CAN OBTAIN THE MODEL NUMBER THROUGH A CALL TO T$SLCO, A NEW KEY OF 7 HAS BEEN IMPLEMENTED IN T$SLCO WHICH WILL PASS BACK THE MODEL NUMEER OF THE CONTROLLER. T$SLCO IS THE LIBRARY ROUTINE THAT A USER PROGRAM CALLS. T$SLC1 IS THE OPERATING SYSTEM WHICH DOES THE ACTUAL WORK. THE CALING SEQUENCE FOR T$SLCO IS:

```
CALL T$SLCO(KEY,LINE,LOC(BLOC),NWDS)
```

WHERE:

KEY = 7 - RETURN MODEL NUMBER
LINE - LOGICAL LINE NUMBER (0-3)
LOC(BLOCK) - ADDRESS OF USER'S BLOCKE
MODEL NUMBER IS RETURNED HERE
NWDS - NUMBER OF WORDS (MUST BE 1)

BEFORE CALLING T$SLCO TO CONFIGURE A LINE (<EY = 3) A CALL WITH A KEY OF 7 SHOULD BE MADE TO SEE IF THE MDLC CONTAINS THE PROPER PROTOCOL AND TO DETERMINE WHAT THE LINE CONFIGURATION WORD SHOULD BE. IT IS THE RESPONSIBILITY OF THE CALLER TO SEE THAT THE LINE CONFIGURATION IS CORRECT FOR THE MODEL OF MDLC BEING USED.

TO CONFIGURE A LINE FOR 1004, UT200 OR 7020 ON THE MDLC, A CHANGE MUST BE MADE TO THE LINE CONFIGURATION WORD SPPLIED IN WRDS 0 AND

1 IN THE BLOCK OF A CALL TO T$SLCO WITH A KEY OF 3. BITS 10 - 16 REMAIN THE SAME. BITS 1 - 9 ARE DEFINED AS FOLLOWS:

MODEL_5650 BIT 10 SET TO ENABLE PROTOCOL

```
 1004:    '1407XX
UT200:     '407XX
 7020:     '427XX
```

MODEL_5651

```
 1004:    '1007XX
UT200:      '7XX
 7020:     '27XX
```

MODEL_5652

```
 1004:    '1007XX
UT200:      '7XX
 7020:     '27XX
```

THE LINE CONFIGURATON FOR THE HSSMLC REMAINS THE SAME.

3.2 GET PATH NAME PRIMITIVE SUBROUTINE

THE FILE SYSTEM AT REV17 CONTAINS A NEW FILE SYSTEM PRIMATIVE, GPATH$. GPATH$ OBTAINS A FULLY QUALIFIED PATHNAME FOR AN OPEN FILE UNIT OR FOR CURRENT OR HOME ATTACH POINTS.

GPATH$ IS A DIRECT ENTRANCE CALL AND MAY BE CALLED BY V-MODE PROGRAMS ONLY. ITS USAGE AND CALLING SEQUENCE ARE AS FOLLOWS:

```
CALL GPATH$(KEY,FUNIT,BUFFER,BUFFLEN,PATHLEN,CODE)
```

WHERE:

KEY — SPECIFIES PATHNAME BASE AND CAN BE: (INPUT ARGUMENT) INTEGER*2

- K$UNIT — PATHNAME OF FILE OPEN ON FILE UNIT SPECIFIED BY BY FUNIT IS RETURNED. (K$UNIT = 1)
- K$CURA — PATHNAME OF CURRENT ATTACH POINT IS RETURNED. (K$CURA = 2)
- K$HOMA — PATHNAME OF HOME ATTACH POINT IS RETURNED. (K$HOMA = 3)

FUNIT — SPECIFIES FILE UNIT NUMBER IF KEY IS K$UNIT ELSE IGNORED. (INPUT) INTEGER*2

BUFFER — IS THE BUFFER IN WHICH THE PATHNAME IS RETURNED. (OUTPUT ARGUMENT) INTEGER*2 (BUFLEN)

BUFFLEN	SPECIFIES MAXIMUM BUFFER LENGTH IN CHARACTERS. IF THE PATHNAME EXCEEDS BUFFLEN CHARACTERS THEN DATA IN BUFFER IS UNDEFINED AND A CODE OF E$BFTS IS RETURNED. (INPUT) INTEGER*2
PATHLEN	SPECIFIES THE LENGTH IN CHARACTERS OF THE PATHNAME RETURNED IN BUFFER. CHARACTERS BEYOND PATHLEN IN BUFFER CONTAIN NO USEFUL INFORMATION. (OUTPUT) INTEGER*2

CODE	IS A STANDARD ERROR CODE (OUTPUT) INTEGER*2 AND CAN BE:
000000	NO ERRORS.
E$BKEY	A BAD KEY WAS SPECIFIED.
E$BUNT	A BAD UNIT NUMBER WAS SPECIFIED IN FUNIT.
E$UNOP	UNIT SPECIFIED IS CLOSED, NAME CAN NOT BE RETURNED.
E$NATT	NOT ATTACHED TO ANY NODE. (FOR KEYS K$CURA, K$HOMA)
E$BFTS	BUFFER GIVEN WITH CHARACTER LENGTH BUFFLEN IS TO SMALL TO CONTAIN FULL PATHNAME. BUFFER CONTAINS NO VALID DATA.
	OTHER CODES MAY BE RETURNED BY SUBROUTINES CALLED BY GPATH$. THE ABOVE CODES ARE THE EXCLUSIVE SET THAT GPATH$ CURRENTLY SETS. (OUTPUT ARGUMENT)

BELOW ARE EXAMPLES OF PATHNAME FORMATS WHICH MAY BE RETURNED IN BUFFER. THE SYNTAX OF THE NAME RETURNED IN BUFFER IS SUBJECT TO A FORMAT CHANGE AT A LATER SOFTWARE REVISION. NOTE THAT PASSWORDS ARE NEVER RETURNED AS PART OF THE PATHNAME.

```
<DISK_NAME>MFD

<DISK_NAME>UFD_NAME

<DISK_NAME>UFD_NAME1>UFD_NAME2>FILE_NAME

<DISK_NAME>UFD_NAME>SEGMENT_DIRECTORY_NAME

<DISK_NAME>UFD_NAME>SEGMENT_DIRECTORY_NAME>ENTRY_NUMBER>ENTRY_NUM
BER

<SPOOLD>MFD

<SPOOLD>SPOOLG

<SALESD>WEST.COAST>YTD.1979>MARCH

<OPSYST>PR4.64>VPRMOS

<DBDISK>DICTIONARY>WORDS>22>68
```

3.3 T$MT (MAG TAPE) ROUTINE CHANGES

3.3.1 OVERVIEW

THE T$MT ROUTINE IS A RAW DATA MOVER THAT MOVES INFORMATION FROM A MAGNETIC TAPE TO THE USER ADDRESS SPACE, OR VICE-VERSA. IT CAN ALSO PERFORM OTHER TAPE OPERATIONS SUCH AS BACKSPACING, FORWARD SPACING, AND SET DENSITY WHEN CALLED WITH THE APPROPRIATE INSTRUCTIONS. HOWEVER, WHEN T$MT ENCOUNTERS AN ERROR CONDITION, IT EXITS TO THE USER COMMAND LEVEL INSTEAD OF RETURNING TO THE CALLING PROGRAM. AN EXTRA ARGUMENT HAS BEEN ADDED TO THE CALLING SEQUENCE OF THE T$MT ROUTINE. THIS SECTION DESCRIBES THE NEW CALLING SEQUENCE AND THE POSSIBLE RETURN ERROR CODES.

3.3.2 T$MT ROUTINE DESCRIPTION

T$MT ROUTINE IS USED TO PERFORM OPERATIONS ON MAGNETIC TAPE DRIVE UNITS. IT CAN PERFORM DATA TRANSFER OPERATION ON A NUMBER OF WORDS FROM 0 TO 6144 WORDS. IT IS ALSO USED TO READ STATUS, SET DENSITY, AND PERFORM TAPE MOTIONS (SPACE FORWARD, SPACE BACKWARD) ON MAGNETIC TAPE DRIVES.

USAGE:

```
      CALL T$MT (UNIT, PBA, NW, INSTR, STATV)
  OR
      CALL T$MT (UNIT, PBA, NW, INSTR, STATV, CODE)
```

UNIT	INTEGER*2 MAGNETIC DRIVE UNIT NUMBER. IT MAY BE EITHER PHYSICAL DRIVE NUMBER OR LOGICAL DRIVE NUMBER. NUMBER MUST BE FROM 0 TO 7.
PBA	INTEGER*4 POINTER TO A BUFFER ADDRESS FROM WHICH (WRITE OPERATION), OR TO WHICH (READ OPERATION) DATA IS TRANSFERRED. IF IT IS NEITHER A READ NOR A WRITE OPERATION, PBA IS 0.
NW	INTEGER*2 NUMBER OF WORDS TO TRANSFER. (BOTH READ AND WRITE OPERATIONS) THIS NUMBER MUST BE BETWEEN 0 AND 6144 WORDS. (0 TO 6K WORDS)
INSTR	INTEGER*2 THE INSTRUCTION REQUEST TO THE MAGNETIC TAPE DRIVE UNIT. VALID INSTRUCTIONS ARE AS FOLLOWS:

OCTAL	HEX	FUNCTION
100000	8000	SELECT TRANSPORT. (7 & 9 TRACK)
100020	8010	ERASE A 3 INCH GAP ON THE TAPE. (VER. 2 AND 3 CONTR. ONLY)
100040	8020	UNLOAD. REWIND THE TAPE AND PLACE THE DRIVE OFFLINE. (VER. 2 AND 3 CONTR. ONLY)
100060	8030	SET DENSITY TO 800 BPI. (NRZI) (VERSION 2 CONTR. ONLY)
100100	8040	SET DENSITY TO 1600 BPI. (PE) (VERSION 2 & 3 CONTR.)
100120	8050	SET DENSITY TO 6250 BPI. (GCR) (VERSION 3 CONTR. ONLY)
000040	0020	REWIND TO BOT. (7 & 9 TRACK)
022100	2440	BACKSPACE ONE FILE MARK. (9 TRACK)
020100	2040	BACKSPACE ONE FILE MARK. (7 TRACK)
062100	6440	BACKSPACE ONE RECORD. (9 TRACK)
060100	6040	BACKSPACE ONE RECORD. (7 TRACK)
022220	2490	WRITE FILE MARK. (9 TRACK)
060200	6080	WRITE FILE MARK. (7 TRACK)
022200	2480	FORWARD SPACE ONE FILE MARK. (9 TRACK)
020200	2080	FORWARD SPACE ONE FILE MARK. (7 TRACK)
042220	4490	WRITE RECORD ONE CHARACTER PER WORD. (9 TRACK)
042620	4590	WRITE RECORD TWO CHARACTERS PER WORD. (9 TRACK)
042200	4480	READ RECORD ONE CHARACTER PER WORD. (9 TRACK)
042600	4580	READ RECORD TWO CHARACTERS PER WORD. (9 TRACK)
052200	5480	READ AND CORRECT RECORD, ONE CHAR/WORD. (9 TRACK)
052600	5580	READ AND CORRECT RECORD, TWO CHAR/WORD. (9 TRACK)
040220	4090	WRITE BINARY RECORD, ONE CHAR/WORD. (7 TRACK)
040620	4190	WRITE BINARY RECORD, TWO CHAR/WORD. (7 TRACK)
044220	4890	WRITE BCD RECORD ONE CHAR/WORD. (7 TRACK)
044620	4990	WRITE BCD RECORD TWO CHAR/WORD. (7 TRACK)
040200	4080	READ BINARY RECORD ONE CHAR/WORD. (7 TRACK)
040600	4180	READ BINARY RECORD TWO CHAR/WORD. (7 TRACK)
044200	4880	READ BCD RECORD ONE CHAR/WORD. (7 TRACK)
044600	4980	READ BCD RECORD TWO CHAR/WORD. (7 TRACK)
043500	4740	READ RECORD BACKWARDS. (VERSION 3 CONTR. ONLY)

STATV INTEGER*2(8)
AN 8 WORDS STATUS VECTOR.

STATV	CONTENT
STATV(1)	STATUS FLAG. (1 = OPERATION IN PROGRESS, 0 = OPERATION DONE)
STATV(2)	HARDWARE STATUS WORD FROM MAGNETIC TAPE CONTROLLER.
STATV(3)	NUMBER OF WORDS TRANFERRED. (READ AND WRITE OPERATIONS ONLY)
STATV(4) TO	RESERVED FOR FUTURE USE.

CODE	INTEGER *2 IF SUPPLIED WHEN T$MT IS CALLED, CONTAINS THE RETURN ERROR CODE.

CODE	MEANING
E$NASS	DEVICE SPECIFIED IN 'UNIT' IS NOT ASSIGNED.
E$IVCM	INVALID COMMAND. (E.G. ATTEMPT TO SET DENSITY ON VER. 0 CONTR.)
E$DNCT	DEVICE SPECIFIED IN 'UNIT' IS NOT CONNECTED, OR NO CONTROLLER.
E$BNWD	BAD NUMBER OF WORDS. (NW < 0, OR NW > 6144)

3.3.3 CHANGES

3.3.3.1 RETURN_CODE

AN EXTRA ARGUMENT 'CODE' IS ADDED TO THE CALLING SEQUENCE FOR T$MT TO RETURN TO THE CALLING PROGRAM IN CASE OF ERROR. THIS SIXTH ARGUMENT IS OPTIONAL. IF T$MT IS CALLED WITH THE 'CODE' ARGUMENT, APPROPRIATE ERROR CODE WILL BE RETURNED TO THE CALLING PROGRAM. IF T$MT IS CALLED WITHOUT THE 'CODE' ARGUMENT, ERRRTN IS CALLED AND PROGRAM WILL EXIT TO USER COMMAND LEVEL IN CASE OF ERROR.

3.3.3.2 PHYSICAL_TO_LOGICAL_UNIT_NUMBER_MAPPING

A NEW MAGNETIC TAPE ASSIGNMENT HAS BEEN IMPLEMENTED AT REV. 17.0 OF PRIMOS. LOGICAL DEVICE NUMBER MAY BE USED IN CALLS TO T$MT. WHEN CALLED, T$MT WILL MAP THE UNIT NUMBER GIVEN IN THE 'UNIT' ARGUMENT TO THE USER'S PHYSICAL DEVICE NUMBER. LOGICAL DEVICE NUMBER MAY BE THE SAME AS PHYSICAL DEVICE NUMBER.

3.3.3.3 NEW_INSTRUCTION

A NEW INSTRUCTION HAS BEEN ADDED TO T$MT. THIS INSTRUCTION IS ONLY VALID WITH VERSION 2 MAGNETIC TAPE CONTROLLER. DEPENDING ON WHETHER THE 'CODE' ARGUMENT IS SUPPLIED, USE OF THIS INSTRUCTION WITH OLDER VERSIONS OF THE CONTROLLER (VERSION 0 1) WILL CAUSE EITHER:

1) PROGRAM EXITS TO USER COMMAND LEVEL AND AN ERROR MESSAGE PRINTED, OR

2) RETURN TO CALLING PROGRAM WITH APPROPRIATE ERROR CODE.

OCTAL	HEX	ACTION
100060	8030	SET DENSITY TO 800 BPI (NRZI)

3.4 INTERFACES TO THE CONDITION MECHANISM

THE FOLLOWING SECTION DOCUMENTS NEW DYNAMICALLY-LINKED CALLS THAT HAVE BEEN MADE AVAILABLE SO THAT V-MODE PROGRAMS MAY USE THE CONDITION MECHANISM. NOTE THAT IT IS NOT POSSIBLE FOR R-MODE PROGRAMS TO USE ANY OF THESE INTERFACES.

3.4.1 SIGNAL SPECIFIC CONDITION

NAME: SIGNL$

PURPOSE:

THE PRIMITIVE SIGNL$ IS CALLED IN ORDER TO RAISE A SPECIFIC CONDITION IN THE RING OF THE CALLER. THE STACK IS SCANNED BACKWARDS IN ORDER TO FIND AN ON-UNIT FOR THIS CONDITION, OR A DEFAULT (ANY$) ON-UNIT. THE FIRST SUCH ON-UNIT FOUND IS INVOKED OF ON-UNITS". THE ON-UNIT HAS THE OPTION OF SIGNALLING ANOTHER CONDITION; OF CALLING THE PRIMITIVE CNSIG$ TO REQUEST THAT THE STACK SCAN FOR ON-UNITS CONTINUE AND THEN RETURNING; OF PERFORMING A NONLOCAL GOTO; AND IT MAY HAVE THE OPTION OF SIMPLY RETURNING, IN WHICH CASE THE PROCESSING OF THE CONDITION IS CONSIDERED COMPLETE AND SIGNL$ RETURNS TO ITS CALLER.

IF SIGNL$ IS CALLED FROM A PROCEDURE EXECUTING IN AN INNER RING (A RING LESS THAN 3), AND NO ON-UNIT OR DEFAULT ON-UNIT IS FOUND IN THAT INNER RING, AN EVENT KNOWN AS A CRAWLOUT OCCURS.

USAGE:

```
DCL SIGNL$ ENTRY (CHAR(*) VAR, PRT, FIXED BIN PRT, FIXED
BIN,
                  BIT(16) ALIGNED);

CALL SIGNL$ (CONDITION_NAME, MS_PTR, MS_LEN, INFO_PTR,
             INFO_LEN, ACTION);
```

CONDITION_NAME
IS THE NAME OF THE CONDITION TO BE SIGNALLED. (INPUT)

MS_PTR
IS A POINTER TO AN SFH, FFH OR CFH STRUCTURE DEFINING THE MACHINE STATE AT THE TIME THE EVENT OCCURRED WHICH MAKES NECESSARY THIS CALL TO SIGNL$. IF MS_PTR IS NULL, SIGNL$ WILL USE A POINTER TO THE CFH PRODUCED BY THE CALL TO SIGNL$. (INPUT)

MS_LEN
IS THE LENGTH IN WORDS OF THE STRUCTURE POINTED TO BY MS_PTR. IF MS_PTR IS NULL, THE VALUE OF MS_LEN IS NOT EXAMINED. (INPUT)

INFO_PTR

IS A POINTER TO AN ARBITRARY STRUCTURE CONTAINING AUXILIARY INFORMATION ABOUT THE CONDITION. THE FORMAT OF THIS STRUCTURE NEED BE KNOWN ONLY TO THOSE PROCEDURES THAT WILL RAISE OR HANDLE THIS CONDITION. IF NO AUXILIARY INFORMATION IS AVAILABLE, INFO_PTR SHOULD BE NULL. (INPUT)

INFO_LEN

IS THE LENGTH OF THE STRUCTURE POINTED TO BY INFO_PTR, IN WORDS. IF INFO_PTR IS NULL, THE VALUE OF INFO_LEN IS NOT EXAMINED. (INPUT)

ACTION

HAS THE FOLLOWING INTERNAL STRUCTURE:

```
DCL 1 ACTION,
      2 RETURN_OK BIT(1) UNAL,
      2 INACTION_OK BIT(1) UNAL,
      2 CRAWLOUT BIT(1) UNAL,
      2 SPECIFIER BIT(1) UNAL,
      2 MBZ BIT(12) UNAL;
```

ACTION.RETURN_OK SHOULD BE '1'B IF THE ON-UNIT IS TO BE ALLOWED TO RETURN. ACTION.INACTION_OK, IF '1'B, INFORMS A POTENTIAL ON-UNIT THAT IT MAY RETURN WITHOUT TAKING ANY CORRECTIVE ACTION AND STILL EXPECT "DEFINED" RESULTS. (RETURN_OK MUST BE '1'B IF INACTION_OK IS '1'B). ACTION.CRAWLOUT IS '1'B IF THIS CALL TO SIGNL$ IS THE RESULT OF A CRAWLOUT. THIS BIT SHOULD NEVER BE SET BY A USER PROGRAM IN A CALL TO SIGNL$, BUT SIGNL$ HAS NO WAY TO ENFORCE THIS RESTRICTION. ACTION.SPECIFIER IS '1'B TO SIGNAL A PLIO CONDITION. THE FIRST MEMBER OF THE INFO STRUCTURE MUST BE THE APPROPRIATE SPECIFIER POINTER. ACTION.MBZ MUST BE '0'B. (INPUT)

USER PROGRAMS SHOUD NEVER ATTEMPT TO SIGNAL A PLIO CONDITION (THAT IS, ACTION.SPECIFIER SHOULD NEVER BE '1'B).

THE BLOCKS OF STORAGE IDENTIFIED BY (MS_PTR, MS_LEN) AND (INFO_PTR, INFO_LEN) WILL BE COPIED ONTO THE OUTER RING STACK IF A CRAWLOUT OCCURS. IN GENERAL, HOWEVER, A PLIO CONDITION SHOULD NOT BE SIGNALLED IN AN INNER RING, AS THE FILE CONTROL BLOCK MAY BE INACCESSIBLE TO THE OUTER RING(S).

3.4.2 MAKE AN ON-UNIT

NAME: MKONU$

PURPOSE:

THE PRIMITIVE MKONU$ IS CALLED BY A PROCEDURE OR BEGIN BLOCK WHEN IT WISHES TO CREATE AN ON-UNIT FOR A SPECIFIC CONDITION, OR A DEFAULT ON-UNIT (AN ON-UNIT FOR THE CONDITION ANY$).

USAGE:

```
DCL MKONU$ ENTRY (CHAR(*) VAR, ENTRY)
                 OPTIONS (SHORTCALL (18));

CALL MKONU$ (CONDITION_NAME, ON_UNIT_ENTRY);
```

CONDITION_NAME
IS THE NAME OF THE CONDITION FOR WHICH THE ACTIVATION DESIRES TO CREATE AN ON-UNIT. IF THE ACTIVATION ALREADY HAS AN ON-UNIT FOR THIS CONDITION, THE PREVIOUS ON-UNIT IS OVERWRITTEN BY THIS NEW ONE. (INPUT)

ON_UNIT_ENTRY
IS AN ENTRY VALUE REPRESENTING THE ON-UNIT PROCEDURE TO BE INVOKED WHEN CONDITION_NAME IS RAISED AND THIS ACTIVATION IS REACHED IN THE STACK SCAN. BECAUSE MKONU$ DOES NOT SAVE THE DISPLAY POINTER ASSOCIATED WITH ON_UNIT_ENTRY, THE ENTRY VALUE MUST BE EXTERNAL OR INTERNAL TO THE BLOCK CALLING MKONU$. NOTE THAT AN ENTRY CONSTANT THAT IS DECLARED IN THE BLOCK CONTAINING THE CALL TO MKONU$ MUST NECESSARILY SATISFY THESE RESTICTIONS. (INPUT)

NOTES

THE STACK FRAME OF THE CALLER IS GROWN, IF NECESSARY, TO ADD THE DESCRIPTOR BLOCK FOR THE NEW ON-UNIT.

THE VALUE OF <CONDITION_NAME> MUST NOT CONTAIN TRAILING BLANKS.

THE CALLER MUST GUARANTEE THAT THE GENERATION OF STORAGE OCCUPIED BY CONDITION_NAME WILL NOT BE FREED UNTIL AFTER THE CALLER RETURNS OR ITS ACTIVATION IS ABORTED BY A NONLOCAL GOTO. FOR PL/I CALLERS, THIS IMPLIES THAT CONDITION_NAME MAY NOT BE A CONSTANT.

3.4.3 REVERT AN ON-UNIT

NAME: RVONU$

PURPOSE:

THIS PRIMITIVE IS CALLED BY AN ACTIVATION WHENEVER IT WISHES TO REVERT AN ON-UNIT IT HAD PREVIOUSLY CREATED. A REVERTED ON-UNIT IS IGNORED WHEN SCANNING THE STACK FOR AN ON-UNIT FOR A CONDITION THAT HAS BEEN RAISED. THE ONLY WAY TO RE-INSTATE A REVERTED ON-UNIT IS TO ISSUE ANOTHER CALL TO MKONU$.

USAGE:

```
DCL RVONU$ ENTRY (CHAR(*) VAR);

CALL RVONU$ (CONDITION_NAME);
```

CONDITION_NAME
: IS THE NAME OF THE CONDITION WHOSE ON-UNIT IN THIS ACTIVATION (IF ANY) IS TO BE REVERTED. (INPUT)

NOTES

THERE IS NO EFFECT IF AN ACTIVATION ATTEMPTS TO REVERT AN ON-UNIT FOR A CONDITION WHEN THAT ACTIVATION HAS NO ON-UNIT FOR THE CONDITION, OR IF THAT ACTIVATION HAD ALREADY REVERTED ITS ON-UNIT FOR THE CONDITION. IN NO CASE WILL A CALL TO RVONU$ AFFECT ON-UNITS IN ANY OTHER ACTIVATION.

3.4.4 SET_CONTINUE-TO-SIGNAL_SWITCH

NAME: CNSIG$

PURPOSE:

THE PRIMITIVE CNSIG$ IS CALLED BY AN ON-UNIT WHEN THAT ON-UNIT HAS NOT BEEN ABLE TO COMPLETELY HANDLE THE CONDITION BECAUSE OF WHICH IT WAS INVOKED. AFTER CALLING CNSIG$, THE ON-UNIT SHOULD RETURN, AT WHICH POINT THE CONDITION MECHANISM WILL RESUME SCANNING THE STACK FOR MORE ON-UNITS FOR THE CONDITION THAT WAS RAISED.

USAGE

```
DCL CNSIG$ ENTRY (FIXED BIN);

CALL CNSIG$ (STATUS);
```

STATUS IS A STANDARD SYSTEM ERROR CODE, AND WILL BE NONZERO ONLY IF THERE WAS NO CONDITION FRAME FOUND IN THE STACK IN WHICH TO SET THE CONTINUE_SW.

NOTES

MULTIPLE CALLS TO CNSIG$ BY THE SAME ON-UNIT PRIOR TO RETURNING TO THE CONDITION MECHANISM WILL HAVE THE SAME EFFECT AS A SINGLE CALL. S.K THE CONTINUE SWITCH IS AUTOMATICALLY SET WHENEVER AN ANY$ ON-UNIT IS INVOKED. SUCH AN ON-UNIT, THEREFORE, NEED NOT CALL CNSIG$ IN ORDER TO CONTINUE TO SIGNAL.

3.4.5 MAKE AN ON-UNIT (FORTRAN)

NAME: MKON%F

PURPOSE:

THE PRIMITIVE MKON$F PERFORMS THE SAME FUNCTION AS MKONU$; THAT IS, AN ON-UNIT FOR A SPECIFIED CONDITION NAME IS CREATED FOR THE CALLING PROCEDURE OR BLOCK. THIS INTERFACE, HOWEVER, AVOIDS USE OF VARYING CHARACTER STRINGS AND SO MAY BE MORE CONVENIENT FOR FORTRAN AND OTHER LANGUAGES. WARNING: THIS INTERFACE IS SUBSTANTIALLY LESS EFFICIENT THAN MKONU$, BOTH IN TERMS OF STACK SPACE AND EXECUTION TIME. APPLICATIONS WHERE THESE ARE IMPORTANT SHOULD USE MKONU$.

USAGE:

```
CALL MKON$F (CNAME, CNAMEL, UNIT)
EXTERNAL UNIT
INTEGER*2 CNAME (--), CNAMEL
```

CNAME IS AN ARRAY CONTAINING THE NAME OF THE CONDITION FOR WHICH AN ON-UNIT IS DESIRED. (INPUT)

CNAMEL IS THE LENGTH IN CHARACTERS OF CNAME. (INPUT)

UNIT IS AN EXTERNAL SUBROUTINE (OR PROCEDURE) WHICH IS TO BECOME THE ON-UNIT HANDLER FOR THIS CONDITION. THIS SUBROUTINE WILL BE INVOKED WITH ONE ARGUMENT AS FOLLOWS:

```
CALL UNIT (CP)
INTEGER*4 CP
```

WHERE CP IS A POINTER TO THE CONDITION FRAME HEADER (CFH) THAT DESCRIBES THE CONDITION.

NOTES

IMPORTANT: ANY PROGRAM COMPILED BY THE FTN COMPILER THAT MAKES A CALL TO MKON$F, MUST INCLUDE THE SPECIFICATION STATEMENT "STACK HEADER 34", AND BE COMPILED WITH THE -SPO OPTION. THIS RESERVES THE STACK SPACE NECESSARY FOR ON-UNIT DATA. IF MKONU$ IS USED, ITS SHORTCALL SPECIFICATION WILL RESERVE THE NEEDED SPACE.

THE COMMENTS IN THE WRITEUP ON MKONU$ APPLY TO MKON$F AS WELL, WITH THE EXCEPTION THAT CNAME AND CNAMEL MAY BE OVERWRITTEN BY THE CALLER ONCE MKON$F HAS RETURNED, BECAUSE THEY ARE COPIED INTO A STACK FRAME EXTENSION BY MKON$F.

NOTE THAT EVERY CALL TO MKON$F ALLOCATES ADDITIONAL STACK SPACE TO HOLD A COPY OF CNAME, EVEN IF THE CALLER HAD PREVIOUSLY CALLED MKON$F WITH THE SAME VALUES OF CNAME AND CNAMEL.

3.4.6 REVERT ON-UNIT (FORTRAN)

NAME: RVON$F

PURPOSE:

THE PRIMITIVE RVON$F REVERTS AN ON-UNIT FOR A SPECIFIC CONDITION IN THE CALLER'S ACTIVATION. IT IS IDENTICAL IN EFFECT TO RVONU$. WARNING: THIS INTERFACE IS LESS EFFICIENT IN EXECUTION TIME (AND TEMPORARY STACK SPACE USED) THAN IS RVONU$; TIME- OR SPACE-CRITICAL APPLICATIONS MAY WISH TO USE RVONU$ INSTEAD.

USAGE:

```
CALL RVON$F (CNAME, CNAMEL)
INTEGER*2 CNAMF(--), CNAMEL
```

CNAME IS THE NAME OF THE CONDITION WHOSE ON-UNIT IN THE CALLER'S ACTIVATION IS TO BE REVERTED. (INPUT)

CNAMEL IS THE LENGTH IN CHARACTERS OF CNAME. (INPUT)

NOTES: ALL COMMENTS THAT APPLY TO RVONU$ ALSO APPLY TO RVON$F.

3.4.7 SIGNAL SPECIFIC CONDITION (FORTRAN)

NAME: SGNL$F

PURPOSE:

THE PRIMITIVE SGNL$F IS USED TO SIGNAL A SPECIFIC CONDITION, SUPPLYING OPTIONAL AUXILIARY INFORMATION WITH THE SIGNAL. A CALL TO SGNL$F IS EQUIVALENT IN EFFECT TO A CALL TO SIGNL$. WARNING: THIS INTERFACE IS LESS EFFICIENT IN EXECUTION TIME (AND TEMPORARY STACK SPACE USED) THAN SIGNL$; TIME- OR SPACE-CRITICAL APPLICATIONS MAY WISH TO USE SIGNL$.

USAGE:

```
CALL SGNL$F (CNAME, CNAMEL, MSPTR, MSLEN, INFOPT,
             INFOLN, FLAGS)
INTEGER*2 CNAME(--), CNAMEL, MSLEN, INFOLN, FLAGS
INTEGER*4 MSPTR, INFOPT
```

CNAME IS THE NAME OF THE CONDITION TO BE SIGNALLED. CNAME IS AN INTEGER ARRAY CONTAINING THE CHARACTER STRING. (INPUT)

CNAMEL IS THE LENGTH OF CNAME IN CHARACTERS. (INPUT)

MSPTR IS AN INTEGER*4 DATUM CONTAINING A HARDWARE INDIRECT POINTER TO A STACK FRAME DESCRIBING THE MACHINE STATE AT THE TIME THE CONDITION WAS DETECTED. USER CALLERS WILL NOT USUALLY KNOW THIS VALUE, AND IF NOT SHOULD PASS THE NULL POINTER VALUE 7777/0, WHICH AS AN OCTAL CONSTANT IS :1777600000. (INPUT)

MSLEN IS THE LENGTH IN WORDS OF THE MACHINE STATE STACK FRAME HEADER. (INPUT)

INFOPT IS A POINTER (SAME FORMAT AS MSPTR) TO A USER-SUPPLIED INFORMATION ARRAY. THIS ARRAY CAN BE IN ANY FORMAT. IF THE ARRAY IS CONTAINED IN THE VARIABLE X, A POINTER TO IT IS PASSED BY THE NONSTANDARD EXPRESSION LOC(X). CALLERS SHOULD PASS THE NULL POINTER (SEE ABOVE) IF NO INFORMATION ARRAY IS BEING SUPPLIED. (INPUT)

INFOLN IS THE LENGTH IN WORDS OF THE INFORMATION ARRAY POINTED TO BY INFOPT. (INPUT)

FLAGS IS AN INTEGER DATUM SPECIFYING CERTAIN CONTROL ACTIONS TO SGNL$F. IF BIT 1 (:100000) IS SET, THE ON-UNIT MAY RETURN. IF BIT 2 (:040000) IS SET, THE ON-UNIT NEED NOT TAKE ANY CORRECTIVE ACTION BEFORE RETURNING. ALL OTHER BITS WILL USUALLY BE 0 (BUT SEE THE WRITEUP ON SIGNL$ FOR A FULL DESCRIPTION). (INPUT)

3.4.8 MAKE LABEL VALUE (FORTRAN)

NAME: MKLB$F

PURPOSE:

THE PRIMITIVE MKLB$F IS CALLED TO CONVERT A FORTRAN STATEMENT LABEL OR INTEGER VARIABLE THAT HAS BEEN ASSIGNED A STATEMENT LABEL VALUE, INTO A PL/I-COMPATIBLE LABEL VALUE. THIS VALUE CAN THEN BE USED TO CAUSE A FULL-FUNCTION NONLOCAL GOTO IN A FORTRAN PROGRAM.

USAGE:

```
CALL MKLB$F (STMT, LABEL)
INTEGER*2 STMT
REAL*8 LABEL
```

STMT IS EITHER A VARIABLE TO WHICH A STATEMENT NUMBER HAS BEEN ASSIGNED BY AN ASSIGN STATEMENT, OR ELSE IS A STATEMENT NUMBER CONSTANT OF THE FORM $XXXXX. (INPUT)

LABEL WILL BE SET TO A PL/I-COMPATIBLE LABEL VALUE IDENTIFYING THE STATEMENT STMT IN THE ACTIVATION OF THE CALLER OF MKLB$F. (OUTPUT)

3.4.9 GENERATE NONLOCAL GOTO

NAME: PL1$NL

PURPOSE:

THIS PRIMITIVE PERFORMS A FULL-FUNCTION NONLOCAL GOTO TO THE ACTIVATION AND STATEMENT IDENTIFIED BY A SUPPLIED LAPEL VALUE. LABEL VALUES CREATED BY CALLS TO MKLB$F ARE SUITABLE ARGUMENTS TO PL1$NL.

USAGE:

```
CALL PL1$NL (LABEL)
REAL*8 LABEL
```

LABEL IS A PL/I-COMPATIBLE LABEL VALUE (SUCH AS IS PRODUCED BY MKLB$F). PL1$NL WILL CAUSE A NONLOCAL GOTO TO THE STATEMENT AND ACTIVATION IDENTIFIED BY LABEL. (INPUT)

3.5 COMMAND ENVIRONMENT SUBROUTINES

IN THIS SECTION, ALL NEW DIRECT CALL SUBROUTINES THAT RELATE TO THE NEW COMMAND ENVIRONMENT AND AND ARE AVAILABLE TO THE USER ARE DOCUMENTED.

NOTE THAT NONE OF THESE DIRECT CALLS IS AVAILABLE TO R-MODE SOFTWARE.

THE SUBROUTINE CALLS DESCRIBED IN THIS SECTION ARE DOCUMENTED IN THE PL/I LANGUAGE, DUE TO THE LATTER'S GREATER PRECISION. FORTRAN AND COBOL USERS OF THESE INTERFACES HAVE ONLY TO MAKE A SIMPLE CONVERSION BETWEEN THE PL/I DATATYPES LISTED AND THEIR OWN DATATYPES. THE CONVERSION TABLE FOR THE PL/P VERSION OF THE PL/I LANGUAGE, AND FORTRAN IS AS FOLLOWS.

PL/P	FORTRAN
CHAR(N) VAR	INTEGER(((N+1)/2)+1) [1ST WORD=LENGTH IN CHARS]
CHAR(N)	INTEGER((N+1)/2)
FIXED BIN(15)	INTEGER
FIXED BIN(31)	INTEGER*4
LABEL	REAL*8
ENTRY VARIABLE	REAL*8
PTR	INTEGER*4
BIT(N)	INTEGER (1 <= N <= 16)

3.5.1 CL$GET - GET COMMAND LINE

NAME: CL$GET

PURPOSE:

THE PROCEDURE CL$GET IS CALLED TO READ A SINGLE LINE OF INPUT TEXT FROM THE CURRENTLY DEFINED COMMAND INPUT STREAM. THE LINE IS RETURNED AS A VARYING CHARACTER STRING WITHOUT THE NEWLINE CHARACTER AT THE END.

USAGE:

```
DCL CL$GET ENTRY (CHAR(*) VAR, FIXED BIN, FIXED BIN);

CALL CL$GET (COM_LINE, COM_LINE_SIZE, STATUS);
```

COM_LINE
: IS THE VARYING CHARACTER STRING INTO WHICH CL$GET WILL WRITE THE TEXT OF THE LINE READ FROM THE COMMAND INPUT STREAM. (OUTPUT)

COM_LINE_SIZE
: IS THE MAXIMUM LENGTH, IN CHARACTERS, OF COM_LINE. NOTE THAT BECAUSE COM_LINE IS A VARYING STRING, IT IS NOT BLANK PADDED TO THIS SIZE. (INPUT)

STATUS
: IS A STANDARD ERROR CODE. (OUTPUT)

NOTES:

AN EMPTY COMMAND LINE OR ONE CONSISTING OF ALL BLANKS WILL COMPARE EQUAL TO THE NULL STRING.

EXAMPLE:

```
DCL A CHAR(128) VAR;

CALL CL$GET (A, 128, STATUS);
```

3.5.2 COMLV$ - GET TO COMMAND LEVEL

NAMES: COMLV$, CMLV$E

PURPOSE:

THE ENTRIES COMLV$ OR CMLV$E ARE CALLED WHEN IT IS DESIRED TO INVOKE A NEW LISTENER LEVEL OF THE PRIMOS COMMAND LOOP. WHEN AND IF THAT INVOCATION OF THE COMMAND LOOP RETURNS (DUE TO A "START" COMMAND HAVING BEEN ENTERED), COMLV$ OR CMLV$E WILL RETURN TO ITS CALLER.

ENTRY: COMLV$

USAGE:

```
DCL COMLV$ ENTRY ();

CALL COMLV$;
```

THERE ARE NO ARGUMENTS.

THIS ENTRY IS USED WHEN THERE HAS BEEN NO "COMMAND ERROR" TO REPORT TO THE USER.

ENTRY: CMLV$E

USAGE:

```
DCL CMLV$E ENTRY ();

CALL CMLV$E;
```

THERE ARE NO ARGUMENTS.

THIS ENTRY IS USED WHEN "COMMAND ERROR" PROCESSING AT THE NEW COMMAND LEVEL IS DESIRED. SPECIFICALLY, THE COMMAND INPUT FILE (IF ANY) IS PAUSED, COMMAND OUTPUT TO THE TERMINAL IS FORCED ON, QUITS ARE ENABLED, AND "ER" IS USED FOR THE PROMPT MESSAGE.

NOTES:

COMLV$ AND CMLV$E SWITCH STACKS TO THE COMMAND PROCESSOR STACK, IF THE PROCESS WAS NOT ALREADY EXECUTING ON THAT STACK.

3.6 BLOCK_DEVICE_INTERFACE

THIS SECTION CONTAINS A SUMMARY OF THE BDI CALLS AND THEIR PURPOSE.

THE BDI FEATURES:

-- AUTOMATIC CHARACTER CONVERSION

-- ISOLATION OF CALLER FROM PROTOCOL TIME DEPENDENCE

-- PROTOCOL INDEPENDENCE

-- SIMPLE EVENT REPORTING

THE FIRST USERS OF THE BLOCK DEVICE INTERFACE ARE DPTX/TSF (ALSO KNOWN AS 3270 SUPPORT) AND DPTX/DSC (ALSO KNOWN AS 3270 EMULATION).

IN A SUBSYSTEM (E.G., DPTX) WHICH USES THE B.D.I., COMMUNICATIONS LIKE PROTOCOL HANDLING (WHICH INCLUDES DETAILS OF POLLING, ACKNOWLEDGING, TERMINAL ADDRESSING, DATA TRANSMISSION, MESSAGE

FRAMING, FIRST-LEVEL ERROR HANDLING, AND SO FORTH) IS DONE BY A SPECIFIC PROTOCOL HANDLING PROCESS. THIS PROTOCOL HANDLER IS A SEPARATE PROCESS (WRITTEN BY PRIME ENGINEERING) WHICH RUNS AS PART OF PRIMOS AND WHICH TAKES OVER MOST OF THE COMMUNCATIONS FUNCTIONS THAT HAD BEEN THE RESPONSIBILITY OF THE T$SLC USER. THE PROTOCOL HANDLER HAS NO RESPONSIBILITIES OTHER THAN KEEPING TRAFFIC MOVING ON THE COMMUNICATIONS LINE.

A USER PROGRAM'S ROLE IS TO CONCENTRATE ON THE CONTENTS OF ITS MESSAGES, NOT THE MECHANICS OF COMMUNICATIONS. THE USER PROGRAM NOW SIMPLY SETS UP A LOGICAL CONNECTION TO A DEVICE (POSSIBLY THE ONLY DEVICE, BUT POSSIBLY ONE OF MANY) ON A SYNCHRONOUS LINE. THE PROGRAMMER USES A SET OF CALLS TO PASS MESSAGES TO OR FETCH MESSAGES FROM THE PROTOCOL HANDLER WHEN IT WANTS TO COMMUNICATE WITH ITS DEVICE.

THE NEW CALLS ARE REFERRED TO, IN THIS DOCUMENT AND ELSEWHERE, AS THE BLOCK DEVICE INTERFACE.

3.6.1 OVERVIEW OF BDI FUNCTIONS

TO PASS MESSAGES OUT TO A DEVICE, OR FETCH MESSAGES IN FROM A DEVICE, THE USER MAKES CALLS TO A PART OF PRIMOS CALLED THE BLOCK DEVICE INTERFACE (BDI). THE BDI, LIKE THE USER-VISIBLE PART OF THE FILE SYSTEM, RUNS IN RING ZERO AS PART OF THE USER'S PROCESS, AND IS INVOKED THROUGH DIRECT-ENTRANCE CALLS. (THESE CALLS ARE THEREFORE ONLY AVAILABLE FROM V-MODE PROGRAMS.) THE BDI MAINTAINS ALL INFORMATION ABOUT DEVICE-TO-USER CONNECTIONS, AND TRANSLATES (IF NECESSARY) AND SCREENS ALL TERMINAL DATA COMING FROM OR GOING TO THE USER PROGRAM. IT ALSO MANAGES THE USER'S END OF THE IN-MEMORY QUEUES (ONE IN EACH DIRECTION FOR EACH DEVICE) WHICH PASS ALL INFORMATION TO OR FROM THE PROTOCOL HANDLER.

A USER PROGRAM CALLS THE BDI TO ATTACH TO AN AVAILABLE DEVICE (BD$ATT CALL) OR TO RELEASE A DEVICE IT NO LONGER NEEDS (BD$DET CALL). WHEN THESE CALLS ARE ISSUED FOR A USER TERMINAL DEVICE, THE BDI TAKES THE TERMINAL OUT OF OR RETURNS IT TO TELETYPE-EMULATION MODE AS APPROPRIATE. THE BDI WILL REFUSE TO HONOR BLOCK-MODE CALLS FOR A USER TERMINAL DEVICE THAT HASN'T FIRST BEEN ATTACHED IN BLOCK MODE. A USER PROGRAM MAY HAVE ANY NUMBER (UP TO 32, IN THIS IMPLEMENTATION) OF DEVICES OF THE SAME TYPE OR OF DIFFERENT TYPES ATTACHED AT ONCE.

THE "SET ATTRIBUTES" CALL BD$SET IS USED TO CHANGE SOME CHARACTERISTIC OF THE INTERFACE. THE CHARACTERISTICS DEFINED IN THIS IMPLEMENTATION AS BEING USER-SETTABLE ARE "DEVICE ENABLED", WHICH ALLOWS THE PROTOCOL HANDLER TO SOLICIT INPUT FROM THE DEVICE, AND "USER CHARACTER CODE", WHICH DEFINES THE CODE (ASCII OR EBCDIC) IN WHICH THE USER WISHES TO "SPEAK" TO THE DEVICE. THE "FETCH INFORMATION" CALL BD$INF IS USED TO PICK UP SOME INFORMATION ABOUT THE DEVICE'S CONFIGURATION AND CURRENT STATUS.

ON AN OUTPUT REQUEST (BD$OUT CALL) FROM THE USER, THE BDI PICKS UP THE OUTPUT TEXT, VERFIES THAT IT CONTAINS A VALID MESSAGE FOR

THAT DEVICE'S PROTOCOL (E.G. THAT A NON-TRANSPARENT MESSAGE DOESN'T CONTAIN DATA LINK CONTROL CHARACTERS), TRANSLATES IT INTO THE APPROPRIATE CHARACTER CODE (E.G. ASCII TO EBCDIC) IF NECESSARY, TRANSFORMS IT INTO THE FORMAT REQUIRED BY THE PROTOCOL HANDLER (AN INTERNAL STRUCTURE WHICH ISN'T OF CONCERN HERE), AND PUTS IT ONTO THAT DEVICE'S OUTPUT QUEUE FOR PASSAGE TO THE PROTOCOL HANDLER. THE CALL THEN RETURNS AND THE PROGRAM CAN CONTINUE TO EXECUTE WHILE ITS MESSAGE IS ON THE QUEUE OR IS BEING TRANSMITTED. ONCE THE MESSAGE HAS BEEN ENQUEUED, PRIMOS ASSUMES RESPONSIBILITY FOR GETTING IT OUT TO THE DEVICE; THE USER PROGRAM WILL ALWAYS BE NOTIFIED OF THE MESSAGE'S EVENTUAL FATE.

AN INPUT REQUEST (BD$INP CALL) FROM THE USER CAUSES THE BDI TO MOVE AN INPUT (IF AVAILABLE) FROM THE DEVICE'S INPUT QUEUE (WHICH IS FED BY THE PROTOCOL HANDLER) TO THE USER'S SPACE IN THE APPROPRIATE FORMAT. AN "INPUT" CAN BE EITHER A DATA MESSAGE CONTAINING CHARACTERS RECEIVED FROM THE DEVICE, OR A STATUS GENERATED BY THE PROTOCOL HANDLER IN RESPONSE TO A NOTEWORTHY EVENT (MESSAGE TRANSMITTED, DEVICE TIMED OUT, ETC.). INPUTS (WHETHER MESSAGES OR STATUSES) ARE PUT ON TO THE INPUT QUEUE, AND THEREFORE PULLED OFF BY AN BD$INP CALL, IN THE CHRONOLOGICAL ORDER IN WHICH THEY WERE ENCOUNTERED OR CREATED BY THE PROTOCOL HANDLER. THEREFORE, MESSAGES AND STATUSES MAY BE INTERSPERSED, AND THE PROGRAM SHOULD BE PREPARED TO HANDLE EITHER TYPE AT ANY BD$INP CALL. THE ORDER IN WHICH MESSAGES AND STATUSES ARRIVE MAY BE SIGNIFICANT, BECAUSE OF THE ASYNCHRONOUS NATURE OF INPUT AND OUTPUT; FOR EXAMPLE, IT IS POSSIBLE TO TELL WHETHER AN INPUT MESSAGE WAS RECEIVED BEFORE OR AFTER THE LATEST OUTPUT MESSAGE WAS TRANSMITTED, OR TO TELL WHETHER A GIVEN OUTPUT MESSAGE WAS TRANSMITTED BEFORE THE PROTOCOL HANDLER DISCOVERED THAT THE TERMINAL HAD BEEN POWERED OFF.

A BD$INP CALL MAY OPTIONALLY WAIT IF THE DEVICE'S INPUT QUEUE IS EMPTY AT THE TIME OF THE CALL. IF SEVERAL INPUTS ARE QUEUED, THE USER DOES SEVERAL BD$INP CALLS TO RETRIEVE THEM, ONE AT A TIME.

3.6.2 BDI PRIMITIVES

THIS SECTION DESCRIBES THE PROGRAMMER'S USE OF THE SIX BLOCK DEVICE INTERFACE CALLS.

BD$ATT: ATTACH A BLOCK DEVICE TO A USER

BD$SET: CHANGE AN ATTRIBUTE OF A BLOCK DEVICE

BD$OUT: ENQUEUE A DATA MESSAGE FOR TRANSMISSION TO A BLOCK DEVICE

BD$INP: FETCH THE NEXT ENQUEUED INPUT FOR A BLOCK DEVICE (MAY BE EITHER DATA OR STATUS)

BD$INF: FETCH SOME INFORMATION ABOUT BLOCK DEVICE STATUS

BD$DET: DETACH A BLOCK DEVICE FROM A USER

A PROGRAM USING THESE CALLS MUST HAVE SEG LOAD THE LIBRARY BDVLIB (IN UFD LIB). VALUES FOR THE KEYS AND TYPE/SUBTYPE CODES LISTED IN THIS SECTION ARE FOUND IN THE INSERT FILE SYSCOM>BD$KEYS. VALUES FOR THE ERROR CODES ARE IN SYSCOM>ERRD.F AND SYSCOM>ERRD.P.

A PROGRAM INITIALLY REFERS TO A DEVICE (THAT IS, ON A BD$ATT CALL) BY ITS "NAME". EACH DEVICE ON THE SYSTEM WILL HAVE BEEN ASSIGNED A NAME AT CONFIGURATION TIME. THE DEVICE NAME IS INTENDED TO BE RELATIVELY CONSTANT, THOUGH THE DEVICE BE MOVED TO A DIFFERENT LOCATION (E.G. LINE POSITION) OR SHUFFLED AROUND IN THE CONFIGURATION ORDER. A PROGRAM DOESN'T HAVE TO KNOW THE NAME OF ITS USER-TERMINAL DEVICE; THIS IS THE "DEFAULT DEVICE" ON AN BD$ATT CALL WHICH HAS NO NAME SPECIFIED. IF THE USER PROGRAM DOESN'T HAVE A BLOCK MODE USER TERMINAL DEVICE (BECAUSE ITS STANDARD USER TERMINAL ISN'T A 3277), IT WILL BE SO INFORMED BY AN ERROR CODE ON THE CALL.

WHEN THE ATTACHMENT BETWEEN THE PROGRAM AND A DEVICE HAS BEEN MADE, THE USER PROGRAM IS ASSIGNED A NUMBER (CALLED THE "DEVICE NUMBER" OR "LOGICAL STATION ID") WITH WHICH IT IS TO REFER TO THE DEVICE ON ALL FUTURE BD$ CALLS. THE NUMBER IS SIMPLY A SHORTHAND FOR USE IN REFERRING TO THAT DEVICE; IT REFLECTS ONLY THE POSITION OF THAT DEVICE IN AN INTERNAL TABLE, AND MAY CHANGE FROM DAY TO DAY AS THE SYSTEM CONFIGURATION CHANGES.

NONE OF THESE CALLS CAN BE USED ON A SYSTEM WHICH IS NOT RUNNING DPTX. IF THE DPTX SUBSYSTEM HASN'T BEEN PREVIOUSLY BEEN TURNED ON BY A "DPTX -ON" COMMAND AT THE SYSTEM CONSOLE, ANY BLOCK DEVICE CALL WILL RETURN AN ERROR OF E$DNC, "DPTX NOT CONFIGURED".

THERE ARE CURRENTLY NO RESTRICTIONS ON THE TYPE OR NUMBER OF DEVICES WHICH A PROGRAM CAN HAVE ATTACHED AT ONCE, EXCEPT THAT THE MAXIMUM NUMBER OF DEVICES SUPPORTED UNDER REV 17 IS 32, AND THERE IS AN OBVIOUS RESTRICTION THAT NO PROGRAM MAY ATTACH TO ITSELF A DEVICE WHICH IS THE STANDARD USER TERMINAL FOR ANOTHER PROCESS.

BD$ATT: ATTACHING A DEVICE.

```
CALL BD$ATT (NAME, LENGTH, DEVICE, CODE)
```

THIS CALL IS USED TO "ATTACH" A BLOCK DEVICE TO A USER SPACE. THE PROTOCOL HANDLER FOR THIS DEVICE WILL PERFORM ANY NECESSARY INITIALIZATION. INPUT AND OUTPUT QUEUES FOR THE DEVICE ARE FLUSHED, AND USER-SETTABLE ATTRIBUTES (SEE BD$SET) ARE INITIALIZED TO THEIR DEFAULTS.

NAME	AN ARRAY (UP TO 32 CHARACTERS) CONTAINING THE "NAME" OF THE DESIRED DEVICE.
LENGTH	THE LENGTH IN CHARACTERS OF THE NAME IN NAME.
	NOTE: IF NAME AND LENGTH ARE BOTH ZERO, THE CALLER'S USER TERMINAL WILL BE ATTACHED, IF IT IS A BLOCK DEVICE. (DPTX/TSF ONLY.)
DEVICE	A NUMBER (THE "LOGICAL STATION ID") ASSOCIATED WITH THIS DEVICE; TO BE USED ON ALL FUTURE BD$ CALLS. THIS IS RETURNED TO THE CALLER.
CODE	A RETURN VALUE INDICATING THE EFFECT OF THE CALL. A NON-ZERO VALUE MEANS THAT NO "ATTACH" WAS DONE.
	0: ATTACH WAS OK.
	E$DATT: DEVICE ALREADY ATTACHED IN BLOCK MODE; THIS CALL HAS NO EFFECT.
	E$BNAM: NO DEVICE FOUND WITH THAT NAME; CALL IGNORED.
	E$DNAV: DEVICE NOT AVAILABLE TO YOU (E.G., IT'S SOMEONE ELSE'S USER TERMINAL); CALL IGNORED.
	E$DNC: DPTX NOT CONFIGURED; CALL IGNORED.

EXAMPLES:

```
CALL BD$ATT (0,0,ZZZ,CODE)
```

THIS IS A "USER TERMINAL" ATTACH. THE NAME IS NOT NEEDED. THE DEVICE NUMBER IS RETURNED IN ZZZ IF THE CALLER'S USER TERMINAL HAS BEEN CONFIGURED AS A BLOCK DEVICE.

```
CALL BD$ATT ('VTERM.A',7,VNO,CODE)
```

THIS IS AN ATTACH BY NAME.

BD$SET: SET ATTRIBUTES FOR A DEVICE.

CALL BD$SET (DEVICE, KEY, CODE)

THIS CALL IS USED TO TELL THE BLOCK DEVICE INTERFACE TO CHANGE AN ATTRIBUTE OF A DEVICE.

DEVICE	THE LOGICAL STATION ID (RETURNED BY A PREVIOUS BD$ATT CALL) OF THE DEVICE FOR WHICH THIS CALL IS INTENDED.
KEY	SPECIFIES THE DESIRED ACTION:
K$ENBD:	(DEFAULT ATTRIBUTE) ENABLE DEVICE. THIS CAUSES THE PROTOCOL HANDLER TO RESUME NORMAL INPUT. A DEVICE IS AUTOMATICALLY ENABLED BY A "BD$ATT" CALL.
K$DSBD:	DISABLE DEVICE. THE PROTOCOL HANDLER WILL STOP SOLICITING AND/OR ACCEPTING INPUT FOR THIS DEVICE.
K$ASCD:	(DEFAULT ATTRIBUTE) TREAT ALL USER I/O AS ASCII. THE BDI WILL TRANSLATE EACH INPUT OR OUTPUT CHARACTER BETWEEN ASCII AND LINE CODE IF NECESSARY. USER CODE "ASCII" IS ASSUMED AS THE DEFAULT AFTER A BD$ATT CALL.
K$EBCD:	TREAT ALL USER I/O AS EBCDIC. THE BDI WILL TRANSLATE EACH INPUT OR OUTPUT CHARACTER BETWEEN EBCDIC AND LINE CODE IF NECESSARY. THIS SETTING OF THE USER CHARACTER CODE WILL REMAIN IN EFFECT UNTIL THE USER DOES A K$ASCD CALL OR DETACHES THE DEVICE.
CODE	A RETURN VALUE INDICATING THE EFFECT OF THE CALL. A NON-ZERO VALUE MEANS THAT NO ACTION WAS TAKEN.
0:	OK.
E$BDEV:	NOT YOUR DEVICE; CALL IGNORED.
E$BPAR:	BAD VALUE FOR KEY; CALL IGNORED.
E$DFD:	DEVICE HAS BEEN FORCIBLY DETACHED BY PRIMOS, MUST BE RE-ATTACHED (USER TERMINAL DEVICE ONLY).
E$DNC:	DPTX NOT CONFIGURED; CALL IGNORED.

EXAMPLES:

```
CALL BD$SET (VNO, K$EBCD, CODE)
```

THIS TELLS THE BDI TO TREAT ALL INPUT TO OR OUTPUT FROM THE USER PROGRAM AS EBCDIC, REGARDLESS OF WHETHER THE DEVICE VNO IS RUNNING EBCDIC OR ASCII CODE.

BD$OUT: OUTPUT TO A DEVICE.

CALL BD$OUT (DEVICE, KEY, DATA, LENGTH, OUTARR, CODE)

THIS CALL IS USED TO PASS OUTPUT DATA TO THE PROTOCOL HANDLER FOR EVENTUAL TRANSMISSION TO THE DEVICE. A RETURN CODE OF ZERO INDICATES THAT THE MESSAGE WAS SUCCESSFULLY QUEUED. SUCCESSFUL TRANSMISSION OF THE DATA IS INDICATED BY A T$TOK TYPE CODE RETURNED BY A LATER BD$INP CALL.

DEVICE — THE LOGICAL STATION ID (RETURNED BY A PREVIOUS BD$ATT CALL) OF THE DEVICE FOR WHICH THIS CALL IS INTENDED.

KEY — SPECIFIES THE DESIRED ACTION:

- K$XMTD: SEND THE DATA IN DATA (AND THE FIELDS IN OUTARR) TO THE PROTOCOL HANDLER FOR TRANSMISSION TO THIS DEVICE.
- K$MRK: INSERT OUTPUT MARKER INTO OUTPUT STREAM. THIS IS A NULL MESSAGE, WHICH CAUSES THE PROTOCOL HANDLER TO RETURN AN INPUT STATUS OF TYPE INPUT MARKER, WITH NO OTHER ACTION. THE FIELDS IN OUTARR(1-4) WILL BE FOUND RETURNED IN INPARR(7-10) ON THE INPUT MARKER. THIS IS PROVIDED FOR DIAGNOSTIC PURPOSES.

DATA — AN ARRAY CONTAINING THE DATA (IF ANY) TO BE TRANSMITTED. THIS FIELD IS IGNORED IF KEY IS K$MRK.

LENGTH — THE LENGTH IN CHARACTERS OF THE DATA TO BE TRANSMITTED. THE MAXIMUM ALLOWABLE LENGTH OF TRANSMITTED DATA IS DEPENDENT ON THE PROTOCOL TYPE OF THE DEVICE. THIS FIELD IS IGNORED IF KEY IS K$MRK.

OUTARR — A FOUR-WORD ARRAY CONTAINING (OPTIONAL) PROTOCOL-DEFINED PARAMETERS TO BE PASSED TO THE PROTOCOL HANDLER FOR THIS DEVICE. SECTIONS 4 AND 5 CONTAIN DESCRIPTIONS OF THE USES OF THIS ARRAY FOR PRIME-SUPPLIED PROTOCOLS. NEW PROTOCOLS MAY USE THIS FIELD FOR ANY PURPOSE, INCLUDING NEW "KEYS" OR FURTHER DESCRIPTION OF THE DATA IN DATA.

CODE — A RETURN VALUE INDICATING THE EFFECT OF THE CALL. A NON-ZERO VALUE MEANS THAT NO ACTION WAS TAKEN.

- 0: OK, DATA QUEUED FOR PROTOCOL HANDLER.
- E$BDEV: NOT YOUR DEVICE; CALL IGNORED.
- E$BPAR: BAD VALUE FOR KEY; CALL IGNORED.
- E$BLEN: BAD LENGTH SPECIFIER; LENGTH IS GREATER

	THAN THE MAXIMUM ALLOWED FOR THIS PROTOCOL, OR LESS THAN ZERO. CALL IGNORED.
E$BDAT:	THE DATA IN DATA CONTAINS COMMUNICATIONS CONTROL CHARACTERS, AND THE LINE PROTOCOL FOR THIS DEVICE IS NOT "TRANSPARENT". CHARACTERS WHICH ARE NOT ALLOWED IN THE USER'S DATA STREAM ARE SOH, STX, ETX, EOT, ENQ, DLE, NAK, SYN, ETB, AND ITB. THE CALL IS IGNORED.
E$QLEX:	OUTPUT QUEUE LENGTH EXCEEDED; CALL IGNORED, TRY AGAIN LATER.
F$NBUF:	NOT ENOUGH BUFFER SPACE TO ENQUEUE THIS MESSAGE; CALL IGNORED, TRY AGAIN LATER.
E$INWT:	INPUT WAITING. FOR SOME PROTOCOLS, THE USER MUST RETRIEVE ANY OUTSTANDING INPUT, WITH A BD$INP CALL, BEFORE AN OUTPUT DATA MESSAGE WILL BE HONORED BY THE BDI. THIS CONVENTION IS ENFORCED BOTH BY DPTX/DSC AND BY DPTX/TSF AT REVS 16 AND 17.
E$DFD:	DEVICE HAS BEEN FORCIBLY DETACHED BY PRIMOS, MUST BE RE-ATTACHED (USER TERMINAL DEVICE ONLY).
F$DNC:	DPTX NOT CONFIGURED; CALL IGNORED.

EXAMPLES:

```
ARR(1) = WRITE$
ARR(2) = 7
ARR(3) = 0
ARR(4) = 0
CALL BD$OUT (MYDEV, K$XMTD, BUFFER, 315, ARR, CODE)
```

THE FIRST 315 CHARACTERS OF BUFFER AND THE FOUR WORDS OF ARR WILL BE COPIED INTO THE QUEUE FOR PASSAGE TO THE PROTOCOL HANDLER WHICH CONTROLS DEVICE MYDEV.

BD$INP: INPUT FROM A DEVICE.

CALL BD$INP (DEVICE, KEY, DATA, LENGTH, INPARR, CODE)

THIS CALL IS USED TO FETCH INPUT IF AVAILABLE, OR TO WAIT UNTIL SOME INPUT APPEARS. THE INPUT MAY BE AN INPUT MESSAGE (DATA FROM THE DEVICE), OR A STATUS PASSED UP FROM THE PROTOCOL HANDLER.

DEVICE — THE LOGICAL STATION ID OF THE DEVICE FROM WHICH THE BDI IS TO LOOK FOR INPUT. IF DEVICE = -1, THE CALL WILL LOOK FOR THE FIRST (CHRONOLOGICAL) INPUT FROM ANY CURRENTLY ACTIVE DEVICE BELONGING TO THE CALLER. THE LOGICAL STATION ID OF THE DEVICE FROM WHICH INPUT HAS BEEN FOUND ON A SUCCESSFUL BD$INP CALL WILL BE RETURNED IN INPARR(4).

KEY — ACTION TO TAKE IF INPUT FROM THE SPECIFIED DEVICE (OR FROM ANY DEVICE, IF DEVICE = -1) IS NOT AVAILABLE AT THE TIME OF THE CALL:

- K$WAIT: WAIT UNTIL SOME INPUT IS AVAILABLE, AND RETURN WITH THE INPUT. THIS KEY CAUSES THE USER PROGRAM TO BE SUSPENDED FOR AN INDEFINITE PERIOD OF TIME IF INPUT IS NOT READILY AVAILABLE, MUCH AS TTIN OR A FORTRAN READ WOULD DO.
- K$NOWA: RETURN IMMEDIATELY, DON'T WAIT.

DATA — AN AREA INTO WHICH THE RECEIVED DATA (IF ANY) IS TO BE MOVED. VALID DATA WILL BE RETURNED ONLY IF CODE IS ZERO OR E$BFTS, AND INPARR(1) (SEE BELOW) IS T$RD.

LENGTH — MAXIMUM NUMBER OF CHARACTERS USER IS WILLING TO ACCEPT INTO THE DATA ARRAY. IF THE SUPPLIED LENGTH IS TOO SHORT FOR THE INPUT MESSAGE, AN E$BFTS ERROR CODE WILL BE GENERATED, LENGTH CHARACTERS WILL BE MOVED INTO DATA, AND THE REST OF THE INPUT WILL BE THROWN AWAY. THE ACTUAL LENGTH OF THE MESSAGE IN DATA WILL BE RETURNED IN INPARR(3).

INPARR — A TEN-WORD ARRAY TO WHICH IS RETURNED INFORMATION ABOUT THE INPUT.

- WORD 1: THE TYPE CODE OF THE INPUT. (SECTION 3.7)
- WORD 2: THE SUBTYPE CODE OF THE INPUT.
- 3: THE ACTUAL LENGTH IN CHARACTERS OF INPUT DATA IN DATA. (WILL BE 0 UNLESS TYPE = T$RD; WILL ALWAYS BE LESS THAN OR EQUAL TO LENGTH.)
- 4: THE DEVICE NUMBER FOR THIS INPUT.
- 5: INPUT STATUS BITS, AS FOLLOWS:

BITS 1-15: RESERVED
BIT 16: IF DEVICE = -1, THERE ARE MORE INPUTS QUEUED FOR SOME DEVICE ATTACHED TO THE CALLER, WHICH CAN BE RETRIEVED BY ANOTHER BD$INP CALL WITH DEVICE = -1. IF DEVICE IS THE NUMBER OF SOME DEVICE ATTACHED TO THE USER, THIS BIT INDICATES THAT THERE ARE MORE INPUTS QUEUED FOR THAT DEVICE, WHICH CAN BE RETRIEVED BY ANOTHER BD$INP CALL WITH THE SAME DEVICE NUMBER.

6: (RESERVED)

7-10: PROTOCOL-SPECIFIC PARAMETERS (SEE SECTIONS 4 AND 5 FOR DEFINITION OF THESE FIELDS FOR PRIME-SUPPLIED PROTOCOLS).

CODE A RETURN VALUE INDICATING THE EFFECT OF THE CALL. A NON-ZERO VALUE MEANS THAT NO INPUT WAS TRANSFERRED, OR THAT DATA WAS TRANSFERRED BUT SOME WAS LOST.

0: INPUT READY; CHECK INPARR(1) FOR TYPE.
E$BDEV: NOT YOUR DEVICE; CALL IGNORED.
E$BPAR: BAD VALUE OF KEY; CALL IGNORED.
E$BLEN: BAD LENGTH SPECIFIER; CALL IGNORED.
E$BFTS: BUFFER PROVIDED FOR INPUT DATA WAS TOO SMALL; THE DATA TRANSFERRED INTO DATA HAS BEEN TRUNCATED.
E$NINP: NO INPUT AVAILABLE (KEY = K$NOWA ONLY)
E$DFD: DEVICE HAS BEEN FORCIBLY DETACHED BY PRIMOS, MUST BE RE-ATTACHED (USER TERMINAL ONLY).
E$DNC: DPTX NOT CONFIGURED; CALL IGNORED.

EXAMPLES:

```
100   CALL BD$INP (MYDEV,K$WAIT,BUFFER,9999,ARR,CODE)
      IF (CODE.NE.0) GO TO 150
      IF (ARR(1).EQ.T$RD) CALL TNOU ('INPUT DATA',10)

WAIT FOR INPUT FROM DEVICE MYDEV AND CHECK FOR RECEIVED TEXT.

       CALL BD$OUT (DEV1,K$XMTD,TBUF,123,OARR,CODE)
       IF (CODE.NE.0) GO TO 200
100    CALL BD$INP (DEV1,K$WAIT,0,0,IARR,CODE)
       IF (CODE.EQ.0 .AND. IARR(1).EQ.T$TOK)  /* WAIT FOR T$T
OK
      X  CALL TNOU ('OUTPUT COMPLETE',15)
       GO TO 100        /* NOT T$TOK, TRY AGAIN

SEND A TEXT MESSAGE OUT WITH BD$OUT, AND WAIT FOR AN
ACKNOWLEDGEMENT THAT IT'S GONE OUT CORRECTLY.
```

BD$INF: INFORMATION ABOUT A DEVICE.

CALL BD$INF (DEVICE, KEY, DATA, LENGTH, INFARR, CODE)

THIS CALL IS USED TO FETCH SOME INFORMATION, WHICH IS MAINTAINED BY THE BLOCK DEVICE INTERFACE, ABOUT THE CURRENT STATE OF A DEVICE.

DEVICE — THE LOGICAL STATION ID OF THE INTENDED DEVICE.

KEY — INDICATES THE TYPE OF INFORMATION DESIRED:

K$INFN: THE NAME OF THE DEVICE WHOSE LOGICAL STATION ID IS IN DEVICE WILL BE RETURNED IN DATA, AND SIX WORDS OF DEVICE STATUS AND CONFIGURATION INFORMATION WILL BE RETURNED IN INFARR (1-6).

K$INFD: BD$INF WILL RETURN DEVICE STATUS "DATA" (HOWEVER DEFINED FOR THIS PROTOCOL) IN DATA INSTEAD OF THE DEVICE NAME, AND WILL RETURN UP TO FOUR WORDS OF PROTOCOL-SPECIFIC DEVICE STATUS INFORMATION (HOWEVER DEFINED FOR THIS PROTOCOL) IN INPARR (7-10), IN ADDITION TO THE STATUS AND CONFIGURATION INFORMATION IN INFARR(1-6). SEE SECTIONS 4 AND 5 FOR PROTOCOL-SPECIFIC DETAILS FOR PRIME-SUPPLIED PROTOCOLS.

DATA — ARRAY WHICH IS TO RECEIVE THE DEVICE NAME OR STATUS "DATA".

LENGTH — MAXIMUM NUMBER OF CHARACTERS OF NAME OR OF STATUS "DATA" WHICH USER IS WILLING TO ACCEPT INTO THE DATA ARRAY. IF THE SUPPLIED LENGTH IS TOO SHORT, LENGTH CHARACTERS WILL BE MOVED INTO DATA AND THE REST THROWN AWAY. ALTHOUGH AN E$BFTS ERROR CODE MAY BE GENERATED, THE ACTUAL LENGTH OF THE RETURNED DATA IS NOT GIVEN. (THE DEVICE NAME IS 32 CHARACTERS LONG, AND THE STATUS "DATA" IS OF A FIXED LENGTH AS DOCUMENTED FOR THE PARTICULAR PROTOCOL INVOLVED.)

INFARR — A TEN-WORD ARRAY WHICH WILL BE FILLED WITH INFORMATION ABOUT THE DEVICE, AS FOLLOWS:

WORD 1: PROTOCOL TYPE OF THIS DEVICE.

0: UNDEFINED

1: DPTX/TSF (3270 TERMINAL SUPPORT)

2: DPTX/DSC (3270 EMULATION PORT)

WORD 2: DEVICE STATUS AND CONFIGURATION BITS.

BIT 1-12: RESERVED.

BIT 13: CHARACTER CODE IN USE BY CALLER: FALSE = ASCII, TRUE = EBCDIC. THE SETTING OF THIS FLAG IS CONTROLLED BY THE USER PROGRAM VIA BD$SET CALLS, KEYS OF K$ASCD AND K$EBCD.

BIT 14: PHYSICAL DEVICE CODE: FALSE = ASCII, TRUE = EBCDIC.

BIT 15: NOT USED.

BIT 16: "ENABLE" STATUS OF DEVICE: FALSE = DISABLED, TRUE = ENABLED. THE SETTING OF THIS FLAG IS CONTROLLED BY THE USER PROGRAM VIA BD$SET CALLS, KEYS OF K$ENBD AND K$DSBD.

WORD 3: MAXIMUM INPUT AND OUTPUT BUFFER LENGTH DEFINED FOR THIS DEVICE. THIS IS THE MAXIMUM VALUE OF LENGTH WHICH IS ACCEPTABLE ON A BD$OUT CALL WITH KEY OF K$XMTD. IT IS ALSO THE LARGEST SIZE IN CHARACTERS WHICH ANY INPUT FROM THIS DEVICE CAN ATTAIN. THE USER PROGRAM SHOULD PROVIDE AN INPUT BUFFER OF AT LEAST THIS LENGTH ON BD$INP CALLS TO ENSURE ENOUGH ROOM FOR THE MAXIMUM SIZED MESSAGE.

4-6: (RESERVED)

7-10: PROTOCOL-SPECIFIC STATUS INFORMATION, IF ANY. SEE SECTIONS 4 AND 5 FOR PROTOCOL-SPECIFIC DETAILS FOR SOME PRIME-SUPPLIED PROTOCOLS.

CODE A RETURN VALUE INDICATING THE EFFECT OF THE CALL. A NON-ZERO VALUE MEANS THAT NO INFORMATION WAS TRANSFERRED.

0: OK, YOU HAVE THE REQUESTED INFORMATION.

E$BDEV: NOT YOUR DEVICE; CALL IGNORED.

E$BPAR: BAD VALUE OF KEY; CALL IGNORED.

E$PLEN: BAD LENGTH SPECIFIER. THE STATUS AND CONFIGURATION IN INFARR HAS BEEN SET UP, BUT THERE IS NOTHING IN DATA.

E$DFD: DEVICE HAS BEEN FORCIBLY DETACHED BY PRIMOS, MUST BE RE-ATTACHED (USER TERMINAL ONLY).

E$DNC: DPTX NOT CONFIGURED; CALL IGNORED.

EXAMPLES:

```
CALL BD$ATT (0,0,MYDEV,CODE)
CALL BD$INF (MYDEV,K$INFN,MYNAME,32,ARR,CODE)
```

THE NAME OF THE CALLER'S USER TERMINAL IS RETURNED IN MYNAME.

```
CALL BD$INF (VNO,K$INFD,INFBUF,3840,IARR,CODE)
MAXLEN = IARR(3)
IF (AND(IARR(2),:1).EQ.0) CALL BD$SET (VNO,K$ENBD,CODE)
```

PICK UP INFORMATION "DATA" FOR VNO, AND ENABLE VNO IF IT'S NOT ALREADY ENABLED.

BD$DET: DETACH FROM A DEVICE.

CALL BD$DET (DEVICE, CODE)

THIS CALL IS USED TO "DETACH" FROM A DEVICE. THE PROTOCOL HANDLER WILL PERFORM ANY NECESSARY CLEANUP. INPUT AND OUTPUT QUEUES FOR THE DEVICE ARE FLUSHED. ANY FURTHER BD$ CALLS BY THE USER (EXCEPT BD$ATT) WILL MEET WITH A E$BDEV ERROR.

DEVICE	THE LOGICAL STATION ID (RETURNED BY A PREVIOUS BD$ATT CALL) OF THE DEVICE TO BE RELEASED.
CODE	A RETURN VALUE INDICATING THE EFFECT OF THE CALL.
	0: DETACH WAS OK.
	E$BDEV: NOT YOUR DEVICE; CALL IGNORED.
	E$DNC: DPTX NOT CONFIURED; CALL IGNORED.

EXAMPLE:

```
CALL BD$DET (VNO,CODE)
IF (CODE.EQ.E$BDEV) CALL TNOU ('WAS NOT ATTACHED',16)
```

TYPE AND SUBTYPE CODES (BD$INP CALL)

THESE WILL BE RETURNED IN INPARR(1), INPARR(2) OF A SUCCESSFUL BD$INP CALL.

TYPE	MEANING
T$RD	DATA HAS BEEN RECEIVED FROM THE DEVICE. THE INPUT DATA IS IN THE ARRAY DATA; THE LENGTH IS IN INPARR(3).
T$TOK	A PREVIOUS DATA TRANSMISSION (DATA PROVIDED ON A PREVIOUS BD$OUT CALL WITH KEY OF K$XMTD) WAS SUCCESSFUL.
T$TF	A PREVIOUS TRANSMISSION (DATA PROVIDED ON A PREVIOUS BD$OUT CALL WITH KEY OF K$XMTD) FAILED. SUBTYPE CONTAINS FURTHER INFORMATION:

SUBTYPE	MEANING
T$REJ	MESSAGE REJECTED BY DEVICE (USUALLY MEANS NAK RECEIVED, RETRIES EXCEEDED)
T$MNA	MESSAGE WAS SENT, BUT NEVER ACKNOWLEDGED BY DEVICE.
T$MNS	MESSAGE NOT SENT. FOR EMULATION DEVICES, THIS MEANS THAT AN INPUT FROM THE HOST (WHICH CHANGED THE VIRTUAL BUFFER) WAS RECEIVED BEFORE THIS MESSAGE COULD BE SENT.

T$RF	DATA RECEPTION FAILED, RETRIES EXCEEDED. NO RECEIVED DATA IS PASSED TO THE USER. SUBTYPE CONTAINS FURTHER INFORMATION:

SUBTYPE	MEANING
T$MBAD	CHECKSUM OR FRAMING ERROR ON INPUT MESSAGE.
T$SIZ	INPUT MESSAGE TOO BIG (LARGER THAN DEFINED MAXIMUM ALLOWABLE MESSAGE).
T$ABRT	INPUT MESSAGE ABORTED BY THE SENDING DEVICE.
T$MINC	MESSAGE INCOMPLETE (ONE OR MORE BLOCKS MISSING).

T$STAT	DEVICE STATUS CHANGE. AN EVENT HAS OCCURRED WHICH THE PROTOCOL HANDLER CONSIDERS NOTEWORTHY, BUT WHICH IS NOT IMMEDIATELY RELATED TO MESSAGE TRANSMISSION OR RECEPTION. CURRENTLY IMPLEMENTED SUBTYPES ARE:

SUBTYPE	MEANING
T$DOWN	DEVICE HAS GONE DOWN AND IS UNAVAILABLE

		UNTIL FURTHER NOTICE. THIS IS AT THE PROTOCOL HANDLER'S DISCRETION AND IS THE RESULT OF LINE DISCONNECTION OR EXCESSIVE UNRECOVERABLE ERRORS.
	T$UP	DEVICE IS BACK UP AND IS NOW AVAILABLE AGAIN.
	T$DIAG	DIAGNOSTIC INFORMATION AVAILABLE. THIS IS INTENDED FOR USE BY DIAGNOSTIC PROGRAMS, DETAILS TO BE SPECIFIED LATER.
	T$TRI	TRANSACTION INITIATED. THIS INDICATES THAT THE REMOTE DEVICE HAS INITIATED A "TRANSACTION" (E.G. A FILE TRANSFER). NO DATA HAS YET BEEN TRANSFERRED. EACH SUPPORTED PROTOCOL HAS ITS OWN DEFINITION OF "TRANSACTION" (WHERE THIS IS A MEANINGFUL CONCEPT).
	T$TRC	TRANSACTION COMPLETE. THIS INDICATES THAT THE REMOTE DEVICE HAS ENDED THE "TRANSACTION" WHICH WAS IN PROGRESS, AND THAT NO FURTHER MESSAGES WILL BE RECEIVED WHICH BELONG TO THAT TRANSACTION.
T$BROK	DEVICE "BROKEN" BY PRIMOS, FOR A BULLETIN OR OTHER CHARACTER-MODE OUTPUT. THIS TYPE CAN BE GENERATED FOR USER TERMINAL DEVICES ONLY. USER SHOULD ASSUME THE "CURRENT" SCREEN FORMAT IS DESTROYED, AND REBUILD IT.	
T$MRK	INPUT MARKER. GENERATED BY PROTOCOL HANDLER IN RESPONSE TO AN OUTPUT MARKER (SEE BD$OUT CALL). INTENDED FOR DIAGNOSTIC USE.	

3.6.3 PROGRAMMER'S NOTES FOR 3270 EMULATION

(NOTE: THIS SECTION ASSUMES THAT THE READER HAS SOME FAMILIARITY WITH THE IBM 3270 DISPLAY SYSTEM. THE FOLLOWING IBM MANUALS ARE GOOD REFERENCES: "AN INTRODUCTION TO THE 3270 INFORMATION DISPLAY SYSTEM", GA27-2739, AND "IBM 3270 INFORMATION DISPLAY SYSTEM COMPONENT DESCRIPTION", GA27-2749.)

THE PROTOCOL HANDLER FOR 3277 EMULATION, WHICH IS CALLED THE VIRTUAL BUFFER EMULATOR (VBE) IN THIS DOCUMENT, IS DIFFERENT IN FUNCTION FROM THE PROTOCOL HANDLER FOR SUPPORT, OR MOST OTHER PROTOCOLS, IN THAT IT DOES NOT ITSELF HAVE "CONTROL" OVER ITS LINE IN TERMS OF SCHEDULING THE SENDING OF MESSAGES. THE VBE CAN SEND A MESSAGE TO THE HOST ONLY WHEN THE HOST REQUESTS IT (BY DOING A POLL). IF THE HOST CHOOSES NEVER TO REQUEST A PARTICULAR MESSAGE, THE VBE CAN NEVER SEND IT -- AN UNPLEASANT BUT UNAVOIDABLE PROBLEM FOR THIS PROTOCOL.

THE VBE MAINTAINS A "VIRTUAL BUFFER" ON BEHALF OF EACH VIRTUAL TERMINAL. THE GOAL OF THE VIRTUAL BUFFER DESIGN IS TO PROVIDE THE VBE AT ALL TIMES A COPY OF THE 3277 SCREEN WHICH IS CONSISTENT WITH WHAT THE HOST THINKS THE SCREEN MIGHT LOOK LIKE. IN GENERAL, ONE MIGHT THINK OF THE VBE AS A "VIRTUAL CONTROL UNIT", WHICH PERFORMS ALL COMMUNICATIONS FUNCTIONS AND CONTROLS THE BUFFERS FOR EACH ASSOCIATED DEVICE; AND OF THE USER PROGRAM AS THE "VIRTUAL OPERATOR", WHO EXAMINES MESSAGES DISPLAYED ON A "VIRTUAL TERMINAL", TYPES INTO FIELDS ON THIS VIRTUAL TERMINAL, AND REQUESTS THE TRANSFER OF HIS INPUT DATA BY HITTING A "VIRTUAL ATTENTION KEY."

THE VBE IS SOLELY RESPONSIBLE FOR THE MAINTENANCE OF THE VIRTUAL BUFFER. NO OTHER PROCESS IS ALLOWED TO MODIFY IT (ALTHOUGH THE USER PROCESS CAN READ IT IN THIS IMPLEMENTATION). THE VBE WILL MAINTAIN A LOGICALLY CONSISTENT IMAGE OF THE VIRTUAL BUFFER EVEN IF THE USER PROCESS MALFUNCTIONS OR FAILS TO PICK UP ITS INPUT MESSAGES.

SINCE THE VBE HAS A VIRTUAL BUFFER TO WORK WITH, IT CAN ALWAYS HANDLE A BUFFER-ORIENTED MESSAGE FROM THE HOST WITHOUT NEEDING TO CONSULT WITH ANOTHER PROCESS. THIS ALLOWS THE VBE TO MAKE A TIMELY AND ACCURATE RESPONSE TO THE HOST, WHILE THE USER FOR WHOM THE MESSAGE IS DESTINED MAY BE OFF DOING SOMETHING ELSE AND NOT ABLE TO RESPOND QUICKLY. WRITE-TYPE COMMANDS (WHICH MODIFY THE BUFFER) ARE PERFORMED DIRECTLY AND ACKNOWLEDGED IMMEDIATELY BY THE VBE IF CORRECT. THE BODY OF THE MESSAGE IS THEN FORWARDED TO THE USER PROCESS, WHICH WILL EXAMINE THE MESSAGE AT ITS LEISURE, INDEPENDENTLY OF ANY PROTOCOL TIMING REQUIREMENTS. READ-TYPE COMMANDS (WHICH FETCH DATA OUT OF THE BUFFER) ARE ALSO PERFORMED DIRECTLY BY THE VBE; THE USER PROCESS DOESN'T GET INVOLVED AT ALL IN THESE.

MOST PROTOCOLS HAVE A SIMPLE, "LINEAR" TRANSFER BETWEEN THE USER AND HIS DEVICE, WHICH WORKS AS FOLLOWS. ON A BD$OUT CALL, THE BLOCK DEVICE INTERFACE (BDI) WILL QUEUE AN OUTPUT MESSAGE FOR THE

PROTOCOL HANDLER, WHO WILL SEND IT WHEN THE LINE IS FREE AND REPORT THE RESULTS. THE PROTOCOL HANDLER WILL PICK UP MESSAGES FROM THE DEVICE IN SOME FASHION AND FORWARD THEM TO THE BDI WITHOUT EXAMINING THEM. ON A BD$INP CALL, THE BDI WILL FIND AN INPUT MESSAGE AND MOVE IT TO THE CALLER'S BUFFER.

BECAUSE OF THE VIRTUAL BUFFER INTERFACE, THE VBE AND BDI HAVE A SLIGHTLY DIFFERENT WAY OF COMMUNICATING. WHEN THE USER DOES A BD$OUT CALL TO PASS AN OUTPUT MESSAGE, THE BDI EXAMINES THE USER'S BUFFER AND COMPARES IT AGAINST THE CURRENT VIRTUAL BUFFER, TO MAKE SURE THAT THE USER PROGRAM IS NOT ATTEMPTING TO MAKE ANY CHANGES TO THE "SCREEN" WHICH COULD NOT BE MADE BY A VIRTUAL OPERATOR SITTING AT A VIRTUAL TERMINAL. AN ATTEMPT TO PUT A NEW CHARACTER IN A PROTECTED FIELD, FOR EXAMPLE, WILL CAUSE THE BDI TO REJECT THE USER'S BUFFER. IF THE USER'S BUFFER PASSES ALL TESTS, THE BDI BUILDS AN UPDATE TEMPLATE WHICH CONTAINS THE DESIRED VIRTUAL BUFFER CHANGES, AND PUTS IT ON A QUEUE FOR THE VBE. (NOTE THAT THE ENQUEUEING OF AN UPDATE IS NECESSARY BECAUSE THE BDI DOESN'T HAVE AUTHORITY TO MODIFY THE VIRTUAL BUFFER; ONLY THE VBE CAN DO THIS.)

THE VBE USES THIS UPDATE TEMPLATE TO CHANGE THE VIRTUAL BUFFER CONTENTS AND SET THE AID BYTE. WHEN THE HOST NEXT POLL'S THIS DEVICE, THE VBE WILL CONSTRUCT THE PROPER MESSAGE OUT OF ITS VIRTUAL BUFFER AND SEND IT TO THE HOST. AT THIS POINT WE CAN SAY THE USER'S MESSAGE HAS BEEN "SENT", AND THE VBE BUILDS A "TRANSMIT OK" MESSAGE FOR THE USER PROCESS. THE USER'S MESSAGE IS NOT CONSIDERED SENT IF THE HOST DOES A READ MODIFIED OR A READ BUFFER COMMAND.

"INPUT FROM A DEVICE" CONSISTS OF ANY TRANSMISSION FROM THE HOST WHICH CAUSES A CHANGE IN THAT DEVICE'S VIRTUAL BUFFER CONTENTS; THAT IS, A TRANSMISSION WHICH CONTAINS A WRITE, ERASE/WRITE, ERASE ALL UNPROTECTED, OR COPY COMMAND. THE VBE CAN'T JUST FORWARD THE MESSAGE TO THE USER PROCESS, BECAUSE THE VBE IS RESPONSIBLE FOR ACKNOWLEDGING OR REJECTING THE HOST'S MESSAGE BASED ON THE CORRECTNESS OF ITS INTERNAL STRUCTURE. THEREFORE, THE VBE CAREFULLY ANALYZES THE INPUT MESSAGE, AND ACKNOWLEDGES IT ONLY IF IT IS CORRECT. THE VBE THEN UPDATES THE VIRTUAL BUFFER TO REFLECT THE NEW STATE OF THE SCREEN (AS THE HOST EXPECTS IT TO BE), AND FORWARDS THE MESSAGE TO THE USER.

THE "MESSAGE" HERE IS A LITTLE MORE COMPLICATED, AS WELL. IN THE CURRENT IMPLEMENTATION, AN INPUT MESSAGE AS SEEN BY THE USER ALWAYS CONSISTS OF THE DATA SENT BY THE HOST PLUS A COMPLETE COPY OF THE NOW-UPDATED VIRTUAL BUFFER, WHICH REFLECTS THE EFFECTS OF THE ACCOMPANYING HOST DATA STREAM. WE HAVE DONE THIS FOR TWO REASONS:

-- THE USER PROGRAM ALWAYS HAS A CHOICE OF DEALING WITH EITHER THE RAW DATA STREAM FROM THE HOST OR WITH THE RESULTS OF APPLYING THAT DATA TO A KNOWN DATA BASE. THE USER MAY ONLY BE INTERESTED IN THE SCREEN IMAGE RESULTING FROM THE DATA STREAM, AND THE VBE HAS ALREADY DONE ALL THE WORK OF UPDATING THE SCREEN IMAGE. FOR A USER

PROGRAM DEALING WITH A 3277 REPLACEMENT TERMINAL, SAY, IT MAY BE FASTER TO WRITE OUT A COMPLETE NEW SCREEN IMAGE FROM THE VIRTUAL BUFFER COPY EACH TIME RATHER THAN TO CALCULATE THE UPDATE OF INDIVIDUAL FIELDS FROM THE LIST OF 3277 FORMATTING ORDERS IN THE INPUT DATA STREAM.

-- ON A COPY COMMAND, THE SCREEN IMAGE CAN CHANGE COMPLETELY BUT NONE OF THE NEW DATA IS CONTAINED IN THE INPUT DATA FROM THE HOST. (IT MAY HAVE BEEN COPIED FROM THE VIRTUAL BUFFER FOR A DIFFERENT DEVICE, WHICH MAY NOT EVEN BE "OWNED" BY THE USER GETTING THE NEW SCREEN.) IN THIS CASE, THE USER PROGRAM NEEDS A COMPLETE COPY OF HIS NEW DEVICE BUFFER.

BECAUSE OF THE ASYNCHRONOUS NATURE OF "INPUT" FROM THE HOST, "OUTPUT" FROM THE USER, AND "UPDATE" OF THE VIRTUAL BUFFER FROM THE USER'S SIDE, TWO PECULIAR SITUATIONS CAN ARISE WHICH CAUSE LOSS OF A USER'S OUTPUT MESSAGE.

-- AFTER THE BDI HAS BUILT AN UPDATE TEMPLATE FROM THE USER'S BD$OUT DATA, AND BEFORE THE VBE HAS PICKED UP THIS UPDATE AND CHANGED THE VIRTUAL BUFFER, THE HOST MAY WRITE TO THE DEVICE, CHANGING THE CONTENTS OF THE VIRTUAL BUFFER. BY THE TIME THE VBE PICKS UP THE USER'S UPDATE, IT NO LONGER "MATCHES" THE VIRTUAL BUFFER AND THE UPDATE CANNOT BE DONE.

-- AFTER THE VBE HAS DONE A VIRTUAL BUFFER UPDATE FROM A USER'S SUBMITTED UPDATE TEMPLATE, BUT BEFORE THE HOST HAS DONE A POLL TO RETRIEVE THE NEW MESSAGE, THE HOST MAY SELECT THE DEVICE; THIS CAUSES THE VBE TO FORGET (AS WOULD A REAL 3271 CONTROL UNIT) THAT AN "ATTENTION" WAS OUTSTANDING. THE HOST MAY NOW SEND A WRITE-TYPE COMMAND, OVERWRITING THE USER'S MESSAGE AND DESTROYING THE OUTSTANDING AID BYTE. THE MESSAGE HAS NOW DISAPPEARED, AND THERE IS NO WAY TO REBUILD IT SINCE THE VIRTUAL BUFFER FOR WHICH THE MESSAGE WAS ORIGINALLY BUILT NO LONGER EXISTS.

IN EITHER OF THESE CASES, THE VBE PUNTS BY THROWING AWAY THE USER'S OUTPUT MESSAGE AND BUILDING A RESULT FOR THE USER WHICH INDICATES "MESSAGE FAILED / COULDN'T BE SENT" (TYPE T$TF, SUBTYPE T$MNS). THE USER PROCESS WILL HAVE TO BE PREPARED TO HANDLE THIS TYPE OF ERROR, PROBABLY BY REBUILDING ITS MESSAGE AGAINST THE NEW (POST-WRITE) VIRTUAL BUFFER AND RESUBMITTING IT.

THE VBE WILL INFORM THE USER PROGRAM WHENEVER THE VIRTUAL DEVICE IS SELECTED OR DESELECTED BY THE HOST. THIS FEATURE WAS IMPLEMENTED BECAUSE, ON A 3277 TERMINAL, THE KEYBOARD BECOMES LOCKED (INPUT INHIBITED) WHENEVER THAT 3277 IS SELECTED, STAYS LOCKED FOR THE DURATION OF THE ENSUING COMMAND SEQUENCE, AND BECOMES UNLOCKED WHEN THAT DEVICE IS DESELECTED. WE FOUND THAT USER PROGRAMS WHOSE FUNCTION IS TO EMULATE A 3277 ON A DIFFERENT TERMINAL (E.G. THE OWL 1200) MIGHT NEED TO SIMULATE A 3277 VERY CLOSELY BY HAVING AN EXPLICIT "KEYBOARD LOCK" IN EFFECT WHILE THE

VIRTUAL TERMINAL IS "SELECTED". THEREFORE, WHENEVER THE VIRTUAL 3277 BECOMES "SELECTED", EITHER BECAUSE OF A SPECIFIC SELECTION ADDRESS SEQUENCE SENT BY THE HOST OR AS PART OF A POLL, THE VBE WILL ENQUEUE FOR THE USER A STATUS MESSAGE WITH TYPE T$STAT AND SUBTYPE T$TRI (TRANSACTION INITIATED). WHENEVER THE VIRTUAL 3277 BECOMES "DESELECTED", EITHER EXPLICITLY WHEN THE LINE RETURNS TO CONTROL MODE OR IMPLICITLY AS A GENERAL POLL STEPS TO THE NEXT DEVICE, THE VBE WILL ENQUEUE A STATUS MESSAGE WITH TYPE T$STAT AND SUBTYPE T$TRC (TRANSACTION COMPLETED).

THE VBE ASSUMES ALL RESPONSIBILITY FOR MAINTAINING THE LINK TO THE HOST ON BEHALF OF ALL (UP TO 32 DIFFERENT) USERS. THE VBE WILL NOT INFORM EACH USER WHEN THE HOST GOES DOWN; IT WILL INFORM ONLY THE SYSTEM OPERATOR BY PRINTING A MESSAGE AT THE SYSTEM CONSOLE. THE USER PROGRAM WILL NEVER RECEIVE STATUS MESSAGES FROM THE VBE OF TYPE T$STAT, SUBTYPE T$UP OR T$DOWN.

WHEN A VIRTUAL BUFFER IS NOT CONNECTED TO A USER PROGRAM (BECAUSE NOBODY HAS YET DONE A BD$ATT CALL TO ATTACH IT), THE VBE WILL PRETEND THAT THAT VIRTUAL TERMINAL IS "POWERED OFF"; IF THE HOST POLLS OR ATTEMPTS TO WRITE TO THAT TERMINAL, THE VBE WILL RETURN A STATUS WITH THE "IR" (INTERVENTION REQUIRED) BIT SET. THE VIRTUAL TERMINAL WILL ALSO BE MARKED AS "POWERED OFF" IF THE USER "DISABLES" IT VIA A BD$SET CALL WITH KEY OF K$DSBD.

IF THE USER PROGRAM, ONCE ATTACHED, SHOULD FAIL TO PICK UP ITS INPUT MESSAGES, AND THE INPUT QUEUE FOR THAT DEVICE BECOMES FULL, THE VBE WILL PRETEND THAT THE VIRTUAL TERMINAL IS "BUSY"; IT WILL SET A STATUS BIT OF "DB" (DEVICE BUSY) AND WILL REFUSE TO ACCEPT MORE MESSAGES FROM THE HOST, SINCE IT HAS NO WAY OF GETTING THE MESSAGES TO THE USER PROGRAM.

AS FOR ALL PROTOCOLS, INPUT TO AND OUTPUT FROM THE USER PROGRAM WILL NORMALLY BE IN ASCII. THIS INCLUDES ALL DISPLAYABLE CHARACTERS, AID BYTES, ADDRESS CHARACTERS, VIRTUAL BUFFER CONTENTS, AND SO FORTH. THE ACTUAL VIRTUAL BUFFER MANAGED BY THE VBE WILL BE IN THE APPROPRIATE LINE CODE (USUALLY EBCDIC), AS WILL BE ALL TRANSMISSION TO AND FROM THE HOST. THE BDI WILL DO ANY NECESSARY CHARACTER TRANSLATION. THE USER CAN TURN OFF TRANSLATION, BY USING A BD$SET CALL WITH KEY OF K$ERCD, SO THAT HE PROVIDES OR RECEIVES ALL MESSAGES IN EBCDIC. THIS FEATURE IS INTENDED PRIMARILY FOR THE USE OF DIAGNOSTIC ROUTINES AND THE PROGRAM WHICH IMPLEMENTS DPTX/TCF (THE 3270 PASS-THROUGH FACILITY), BUT IS GENERALLY AVAILABLE TO ANY USER.

THE FOLLOWING SECTIONS EXPLAIN THE PROTOCOL-SPECIFIC FEATURES WHICH A PROGRAMMER USING A 3270 EMULATION PORT NEEDS TO KNOW, IN ADDITION TO THE GENERAL BD$ CALLS DESCRIPTION IN SECTION 3.

3.6.3.1 BD$OUT: OUTPUT FROM USER TO VIRTUAL BUFFER EMULATOR

A USER PROGRAM MAKES THIS CALL TO SEND A 3277-LIKE MESSAGE TO THE HOST. THE USER'S DATA AREA SHOULD LOOK AS THOUGH IT HAD BEEN GENERATED BY A READ MODIFIED (WITH A SINGLE EXCEPTION, EXPLAINED IN THE NEXT SECTION). THE DATA IS NOT PASSED DIRECTLY TO THE HOST, BUT IS ANALYZED CAREFULLY BY BD$OUT TO ENSURE THAT IT IS A PROPER READ MODIFIED DATA STREAM AND IS THEN STORED IN THE VIRTUAL BUFFER FOR THE CORRECT DEVICE UNTIL THE HOST REQUESTS IT.

DATA FORMAT

THE OUTPUT DATA IN DATA (PROVIDED BY THE USER ON A BD$OUT CALL) MUST BE IN THE FOLLOWING FORMAT.

-- AID BYTE. THIS SINGLE BYTE MUST BE A VALID AID WHICH INDICATES OPERATOR ACTION: ONE OF ENTER, PROGRAM FUNCTION, PROGRAM ACCESS KEY, CLEAR KEY, OR TEST REQ KEY. AN AID OF "-" OR "Y", INDICATING NO ACTION BY THE OPERATOR, IS NOT CONSIDERED VALID. AID'S FOR THE OPERATOR IDENTIFICATION CARD READER AND SELECTOR PEN ARE NOT SUPPORTED AT PRESENT. THE AID SHOULD BE THE ONLY CHARACTER IN DATA IF THE AID INDICATES A "SHORT READ" TYPE (CLEAR OR PROGRAM ATTENTION KEY).

-- CURSOR ADDRESS. THIS IS TWO BYTES OF ADDRESS IN IBM 3270 ADDRESS FORMAT. IT MUST BE PRESENT ONLY IF THE AID INDICATES A "READ MODIFIED" TYPE (ENTER, PROGRAM FUNCTION, OR TST REQ KEY). NOTE THAT A CURSOR ADDRESS IS REQUIRED WITH A TEST REQUEST MESSAGE.

-- READ MODIFIED DATA. THIS CONSISTS OF MODIFIED FIELDS, OR THE ENTIRE SCREEN IF THE BUFFER IS UNFORMATTED, POSSIBLY WITH NULLS SUPPRESSED. DPTX/DSC ALLOWS THE USER PROGRAM TO INSERT NULLS IN THE DATA STREAM, FOR REASONS EXPLAINED BELOW. THE READ MODIFIED DATA PORTION OF THE OUTPUT DATA MAY BE EMPTY IF THERE ARE NO MODIFIED FIELDS.

ALL FIELDS SHOULD BE IN ASCII UNLESS THE PROGRAM HAS PREVIOUSLY DONE A BD$SET CALL WITH A KEY OF K$EBCD TO INDICATE THAT THE PROGRAM WILL BE "SPEAKING" IN EBCDIC. NOTE: THE ASCII TO EBCDIC CONVERSIONS USED BY DPTX ARE NOT THOSE USED BY IBM. A PROGRAM WHICH USES ASCII TO TALK WITH A DPTX DEVICE WHICH IS EBCDIC SHOULD FOLLOW THE PRIME ASCII CHARACTER CONVENTIONS DESCRIBED IN A SEPARATE DPTX DOCUMENT.

THE VIRTUAL BUFFER EMULATOR PROVIDES ALL FRAMING CHARACTERS. THE USER PROGRAM MAY NOT SUPPLY SOH, STX, ETB OR OTHER DATA LINK CONTROL CHARACTERS.

THE MAXIMUM ALLOWABLE DATA LENGTH (FIELD LENGTH ON THE BD$OUT CALL) AT REVS 16 AND 17 IS 2048 BYTES, WHICH INCLUDES AN AID, CURSOR ADDRESS, AND 2045 CHARACTERS OF 3277 DATA. THIS MAXIMUM LENGTH MAY BE ALTERED AT FUTURE RELEASES; ITS VALUE CAN ALWAYS

BE DETERMINED THROUGH A BD$INF CALL (Q.V.). THE MINIMUM ALLOWABLE LENGTH IS ONE BYTE (AID BYTE ONLY).

IF THE VIRTUAL BUFFER IS CURRENTLY FORMATTED (CONTAINS ATTRIBUTE CHARACTERS), THEN THE "READ MODIFIED" PORTION OF DATA CONSISTS OF MODIFIED FIELDS, EACH OF WHICH CONTAINS AN SBA, TWO-BYTE ADDRESS SEQUENCE, AND DATA (POSSIBLY WITH NULLS SUPPRESSED). IF THE VIRTUAL BUFFER IS CURRENTLY UNFORMATTED (CONTAINS NO ATTRIBUTE CHARACTERS), THEN THE "READ MODIFIED" PORTION OF DATA CONSISTS OF THE ENTIRE BUFFER CONTENTS (POSSIBLY WITH NULLS SUPPRESSED). DATA FOR AN UNFORMATTED BUFFER MUST NOT CONTAIN SBA CHARACTERS AND MUST BE LESS THAN OR EQUAL TO 1920 CHARACTERS (THE SCREEN SIZE) IN LENGTH.

THE ONLY EXCEPTION TO THE "READ MODIFIED" RULE ON DATA IS THAT NULLS ARE ALLOWED AS NORMAL CHARACTERS IN DATA, AND WILL BE INSERTED INTO THE VIRTUAL BUFFER AS IF THEY WERE NORMAL GRAPHIC CHARACTERS. THIS IS SO THAT USER PROGRAMS FOR WHOM THE POSITION OF CHARACTERS WITHIN NULLED FIELDS IS SIGNIFICANT CAN ENSURE THAT THE VIRTUAL BUFFER ACCURATELY REFLECTS THE POSITION OF THOSE CHARACTERS. (OTHERWISE, IT WOULD BE IMPOSSIBLE FOR A USER PROGRAM TO "TYPE" INTO THE MIDDLE OF A NULLED FIELD.) HOWEVER, NULLS ARE NEVER REQUIRED IN DATA, SINCE BD$OUT CORRECTLY INTERPRETS NULL SUPPRESSION.

A PROGRAM WHOSE FUNCTION IS SIMPLY TO MOVE DATA BETWEEN THE PRIME SYSTEM AND AN IBM SYSTEM, VIA THE 3271 LINE PROTOCOL, MIGHT ALWAYS USE AN UNFORMATTED BUFFER. EVEN IN THIS CASE, HOWEVER, THE FIRST THREE BYTES OF DATA MUST CONTAIN A VALID AID AND VALID CURSOR ADDRESS BYTES.

NOTE THAT THE DATA FOR A FORMATTED SCREEN MIGHT NOT CONTAIN ANY MODIFIED FIELDS. THIS REFLECTS A SITUATION IN WHICH AN OPERATOR HAS HIT A PROGRAM ATTENTION KEY WITHOUT TYPING ANYTHING, AND IS PERFECTLY LEGAL. SIMILARLY, THE DATA FOR AN UNFORMATTED SCREEN MAY BE EMPTY, REFLECTING A CASE IN WHICH THE "SCREEN" IS ALL NULLS.

PROTOCOL-SPECIFIC PARAMETERS (OUTARR(1-4))

NONE DEFINED FOR 3270 EMULATION PROTOCOL.

PROTOCOL-SPECIFIC ERRORS (CODE)

WHILE BD$OUT IS ATTEMPTING TO MATCH THE USER'S SUBMITTED 3277 DATA AGAINST THE EXISTING VIRTUAL BUFFER, IT MAY FIND ONE OF SEVERAL REASONS FOR REJECTING THE DATA. THEREFORE, THERE ARE SEVERAL ERROR CODES (SPECIFIC TO 3270 EMULATION) WHICH DEFINE "BAD DATA" MORE COMPLETELY. (THESE ARE IN ADDITION TO, NOT IN PLACE OF, THE CODE E$BDAT WHICH MEANS THAT INVALID DATA LINK CONTROL CHARACTERS (E.G. STX, ETX) WERE FOUND IN DATA.)

E$VIA: INVALID AID BYTE IN FIRST CHARACTER.

E$VICA: INVALID CURSOR ADDRESS IN SECOND AND THIRD

CHARACTERS.

E$VIF: IMPROPER FIELD. THIS INDICATES THAT AN SBA ORDER HAS BEEN FOLLOWED BY "ADDRESS CHARACTERS" WHICH ARE INVALID OR DO NOT INDICATE THE BEGINNING OF A FIELD.

E$VFR: FIELD REQUIRED. THE VIRTUAL BUFFER CONTAINS FIELDS, BUT THE USER IS ATTEMPTING TO WRITE AS IF IT WERE UNFORMATTED (NO SBA FOUND).

E$VFP: FIELD PROHIBITED. THE VIRTUAL BUFFER CONTAINS NO FIELDS, BUT THE USER IS ATTEMPTING TO WRITE AS IF IT WERE FORMATTED (BUFFER CONTAINS SBA CHARACTERS).

E$VPRO: PROTECTED FIELD CHECK. THIS INDICATES AN ATTEMPT TO PUT CHARACTERS INTO A PROTECTED FIELD.

E$VNUM: NUMERIC FIELD CHECK. THIS INDICATES AN ATTEMPT TO PUT NON-NUMERIC CHARACTERS INTO A NUMERIC FIELD (ONLY 0-9, PERIOD, MINUS SIGN AND DUP ALLOWED).

E$VPF: PAST END OF FIELD. THIS INDICATES AN ATTEMPT TO WRITE MORE CHARACTERS INTO A FIELD THAN WILL FIT, I.E. TO OVERWRITE THE ATTRIBUTE BYTE FOR A FOLLOWING FIELD, OR TO WRITE MORE THAN 1920 CHARACTERS INTO AN UNFORMATTED SCREEN.

E$VIRC: INVALID READ MODIFIED CHARACTER. THIS INDICATES THAT DATA CONTAINS A CHARACTER WHICH CANNOT EXIST IN READ MODIFIED DATA STREAM. INVALID CHARACTERS INCLUDE 3270 ORDERS SF, IC, PT, RA, AND EUA.

3.6.3.2 BD$INP: INPUT DATA FROM VIRTUAL BUFFER EMULATOR TO USER

AN INPUT FROM THE VBE WILL HAVE ONE OF THE FOLLOWING TYPE/SUBTYPE CODES.

T$RD: RECEIVED DATA FROM HOST (FORMAT DESCRIBED BELOW)

T$TOK: YOUR PREVIOUS OUTPUT MESSAGE WAS SENT TO AND ACCEPTED BY THE HOST.

TTF/TMNS: YOUR PREVIOUS OUTPUT WAS LOST BECAUSE THE HOST DID A WRITE-TYPE COMMAND BEFORE IT READ YOUR MESSAGE.

TTF/TREJ: YOUR PREVIOUS OUTPUT WAS LOST BECAUSE THE HOST CAUSED OUR TRANSMISSION TO BE ABORTED PART WAY THROUGH.

TTF/TMNA: YOUR PREVIOUS OUTPUT WAS LOST BECAUSE OF HOST FAILURE OR COMMUNICATIONS LINE DIFFICULTY.

T$STAT/T$TRI: YOUR VIRTUAL TERMINAL HAS BEEN SELECTED.

T$STAT/T$TRI: YOUR VIRTUAL TERMINAL HAS BEEN DESELECTED.

DATA FORMAT

INPUT DATA FROM THE VIRTUAL BUFFER EMULATOR WILL BE OF THE FOLLOWING FORMAT.

3840 BYTES: CURRENT VIRTUAL BUFFER COPY (FORMAT BELOW)

0-2048 BYTES: DATA STREAM WHICH ACCOMPANIED A WRITE-TYPE COMMAND FROM THE HOST; THIS CONSISTS OF ALL ORDERS AND DATA WHICH FOLLOWED A WRITE CONTROL CHARACTER IN THE INPUT STREAM, AND IS PRESENT ONLY IF THE COMMAND WAS A WRITE OR AN ERASE/WRITE.

A BD$INP CALL WHICH RETURNS A CODE OF ZERO (INPUT AVAILABLE) AND INPARR(1) OF T$RD (RECEIVED DATA FROM HOST) WILL ALWAYS LEAVE THE USER WITH THE ABOVE INFORMATION IN DATA. THE ACTUAL LENGTH IN CHARACTERS OF THE INPUT DATA IN DATA WILL BE RETURNED IN INPARR(3).

THE USER SHOULD PROVIDE A DATA BUFFER OF LENGTH 5888 BYTES (3840+2048) TO ENSURE THAT THERE IS ENOUGH ROOM FOR THE LARGEST INCOMING MESSAGE. THE SMALLEST DATA MESSAGE WILL BE 3840 BYTES (FOR A WRITE-TYPE COMMAND WITH NO DATA); THE VIRTUAL BUFFER COPY IS ALWAYS PASSED TO THE USER AS PART (POSSIBLY THE ENTIRETY) OF AN INPUT MESSAGE.

ALL CHARACTER FIELDS WILL BE IN ASCII UNLESS THE PROGRAM HAS PREVIOUSLY DONE A BD$SET CALL WITH A KEY OF K$EBCD TO INDICATE THAT THE PROGRAM WILL BE "SPEAKING" IN EBCDIC. NOTE: THE ASCII TO EBCDIC CONVERSIONS USED BY DPTX ARE NOT THOSE USED BY IBM. A PROGRAM WHICH USES ASCII TO TALK WITH A DPTX DEVICE WHICH IS EBCDIC SHOULD FOLLOW THE PRIME ASCII CHARACTER CONVENTIONS DESCRIBED IN A SEPARATE DPTX DOCUMENT.

THE VIRTUAL BUFFER COPY IS ALWAYS EXACTLY 1920 WORDS LONG (3840 CHARACTERS); THIS IS ONE WORD PER BUFFER POSITION. EACH WORD CONTAINS EIGHT BITS OF FLAGS AND ONE CHARACTER AS FOLLOWS:

BITS 1-2	ALWAYS 0.
3	THIS CHARACTER IS A "FF"
4	THIS CHARACTER IS A "NL"
5	THIS CHARACTER IS A "EM"
6	THIS CHARACTER IS A "DUP"
7	THIS CHARACTER IS A "FM"
8	THIS CHARACTER IS AN ATTRIBUTE
9-16	CHARACTER (WILL BE ASCII OR EBCDIC DEPENDING ON THE USER'S DEFINED INPUT/OUTPUT CHARACTER CODE)

THE HOST COMMAND WHICH ACCOMPANIED THE INPUT IS SPECIFIED IN INPARR(7), THE FIRST PROTOCOL-SPECIFIC PARAMETER (SEE NEXT SECTION). THE EXACT INTERPRETATION OF THE DATA STREAM PART OF DATA DEPENDS ON THE COMMAND TYPE.

WRITE: THE DATA STREAM CONSISTS OF 3270 SCREEN FORMATTING ORDERS AND/OR DATA CHARACTERS (0 TO 2048 BYTES). THE WRITE CONTROL CHARACTER IS SPECIFIED SEPARATELY (IN

INPARR(8)) AND IS NOT CONSIDERED PART OF THE DATA.

ERASE/WRITE: SAME AS FOR WRITE.

COPY: NO DATA STREAM IN DATA. THE COPY CONTROL CHARACTER AND FROM DEVICE SPECIFIER ARE IN INPARR(8); THEY ARE NOT CONSIDERED DATA.

ERASE ALL UNPROTECTED: NO DATA STREAM.

PROTOCOL-SPECIFIC PARAMETERS (INPARR(7-10))

THESE FOUR WORDS HAVE SPECIAL MEANINGS FOR 3270 EMULATION PROTOCOL, AS FOLLOWS. ALL FIELDS MARKED "CURRENT" REFLECT THE STATE OF THE VIRTUAL BUFFER AFTER THE PROCESSING OF THE COMMAND WHICH CAUSED THIS INPUT.

INPARR(7) HOST COMMAND AND NULL PROCESSING FLAGS:

BIT 1: NULL PROCESSING CAUSED BY THIS COMMAND (THIS FLAG IS FOR THE USE OF THE OWL INTERFACE PROGRAM, OWLDSC).

BIT 2: THIS COMMAND WAS CHAINED FROM A PREVIOUS READ OR WRITE COMMAND.

BITS 3-8: UNUSED.

BITS 9-16: A CODE INDICATING THE HOST COMMAND WHICH CAUSED THIS INPUT:

1: WRITE
2: ERASE/WRITE
3: COPY
4: ERASE ALL UNPROTECTED

INPARR(8) CURRENT WCC/CCC AND "FROM" ADDRESS.

BITS 1-8: THE WRITE CONTROL CHARACTER (IF COMMAND IS WRITE OR ERASE/WRITE) OR COPY CONTROL CHARACTER (IF COMMAND IS COPY) WHICH ACCOMPANIED THIS COMMAND. ONLY THE LOW 6 BITS (WHICH ARE THE WRITE OR COPY CONTROL FLAGS) ARE PROVIDED. IF THE COMMAND IS ERASE ALL UNPROTECTED, THIS FIELD WILL BE ZERO.

BITS 9-16: THE LOGICAL STATION ID OF THE DEVICE FROM WHICH THIS INPUT WAS COPIED. THIS FIELD WILL BE VALID ONLY IF THE COMMAND IS COPY. NOTE, PLEASE, THAT THE "FROM" DEVICE DOES NOT NECESSARILY BELONG TO THE USER PROGRAM.

INPARR(9) CURRENT CURSOR ADDRESS:

BITS 1-8: FIRST CURSOR ADDRESS BYTE
BITS 9-16: SECOND CURSOR ADDRESS BYTE

INPARR(10) CURRENT BUFFER ADDRESS:
BITS 1-8: FIRST CBA BYTE
BITS 9-16: SECOND CBA BYTE

NOTE THAT THESE FIELDS WILL BE VALID ONLY ON THE TRANSFER OF INPUT DATA (I.E., CODE = 0 AND INPARR(1) = T$RD).

PROTOCOL-SPECIFIC ERRORS (CODE)

NONE DEFINED FOR 3270 EMULATION PROTOCOL.

3.6.3.3 BD$INF: OBTAIN INFORMATION ABOUT EMULATION DEVICE

WHEN A USER PROGRAM CALLS BD$INF WITH A KEY OF K$INFD INSTEAD OF A KEY OF K$INFN, BD$INF RETURNS PROTOCOL SPECIFIC "STATUS DATA" IN DATA (INSTEAD OF THE DEVICE NAME), AND RETURNS PROTOCOL SPECIFIC FIELDS IN INFARR(7-10). THE PROTOCOL-SPECIFIC INFORMATION FOR 3270 EMULATION DEVICES IS DESCRIBED IN THIS SECTION.

THE "MAXIMUM MESSAGE SIZE" RETURNED IN INFARR(3) IS 2048. THIS INDICATES THE SIZE OF THE DATA STREAM ALLOWED ON I/O, AND DOES NOT INCLUDE ALLOWANCE FOR THE VIRTUAL BUFFER COPY (3840 BYTES) WHICH IS ALWAYS PASSED ALONG ON A T$RD INPUT.

DATA FORMAT

THE INPUT DATA RETURNED IN DATA ON A BD$INF CALL WITH A KEY OF K$INFD WILL BE A COPY OF THE CURRENT VIRTUAL BUFFER. THIS IS EXACTLY 1920 WORDS (3840 BYTES) LONG, AND HAS THE FORMAT DESCRIBED IN SECTION 4.2.1. THERE IS NO "INPUT DATA STREAM" RETURNED ON THIS CALL. IF LENGTH IS SHORTER THAN 3840 BYTES, THE VIRTUAL BUFFER COPY WILL BE TRUNCATED.

PROTOCOL-SPECIFIC PARAMETERS (INFARR(7-10))

THESE FOUR WORDS HAVE SPECIAL MEANINGS FOR 3270 EMULATION PROTOCOL, AS FOLLOWS.

INFARR(7) UNUSED.

INFARR(8) LAST WCC AND CURRENT AID.
BITS 1-8: LAST WCC SENT BY HOST (USEFUL FOR "KBD RESTORE" BIT)
BITS 9-16: CURRENT AID

INFARR(9) CURRENT CURSOR ADDRESS
BITS 1-8: FIRST CURSOR ADDRESS BYTE

BITS 9-16: SECOND CURSOR ADDRESS BYTE

INFARR(10) CURRENT BUFFER ADDRESS
BITS 1-8: FIRST CBA BYTE
BITS 9-16: SECOND CBA BYTE

PROTOCOL-SPECIFIC ERRORS (CODE)

NONE DEFINED FOR 3270 EMULATION PROTOCOL.

3.6.4 PROGRAMMER'S NOTES FOR 3270 SUPPORT

(NOTE: THIS SECTION ASSUMES THAT THE READER HAS SOME FAMILIARITY WITH THE IBM 3270 DISPLAY SYSTEM. THE FOLLOWING IBM MANUALS ARE GOOD REFERENCES: "AN INTRODUCTION TO THE 3270 INFORMATION DISPLAY SYSTEM", GA27-2739, AND "IBM 3270 INFORMATION DISPLAY SYSTEM COMPONENT DESCRIPTION", GA27-2749.)

THE PROTOCOL HANDLER FOR IBM 3271/3277 TERMINAL SUPPORT, CALLED THE SUPPORT TRAFFIC MANAGER (STM) IN THIS DOCUMENT, IS THE GATEWAY BETWEEN A USER PROCESS AND ITS 3277 TERMINAL WHETHER THAT TERMINAL IS IN BLOCK (NATIVE) MODE OR IN "STANDARD" (TELETYPE-EMULATION) MODE. THIS SECTION DESCRIBES WHAT THE USER PROGRAM SEES WHEN IT'S RUNNING A 3277 IN BLOCK MODE USING THE BD$ CALLS.

WHEN THE 3277 IS IN STANDARD MODE, THE PROGRAM IS UNAWARE THAT IT IS CONVERSING WITH ANYTHING OTHER THAN THE STANDARD COMMAND DEVICE. THE PROGRAM'S INTERFACE TO THE TERMINAL IS EXACTLY WHAT IT WOULD BE FOR A TELETYPE CONNECTED TO THE AMLC. THE STM AND ANOTHER MODULE WITHIN PRIMOS TAKE CARE OF ALL THE TRANSFORMATIONS THAT NEED TO OCCUR BETWEEN THE USER'S TERMINAL INPUT/OUTPUT BUFFERS AND THE 3277 SCREEN. (THIS IS DESCRIBED IN A SEPARATE DOCUMENT.) HOWEVER, WHEN THE PROGRAM DOES A BD$ATT CALL TO ATTACH ITS USER TERMINAL DEVICE IN BLOCK MODE (ASSUMING, OF COURSE, THAT THE USER TERMINAL IS A 3277), THE PROGRAM GAINS (ALMOST) COMPLETE CONTROL OVER THE FORMATTING OF THE 3277 SCREEN, AND BEGINS TO COMMUNICATE IN "MESSAGES" RATHER THAN IN "CHARACTERS" OR "LINES". (THE "ALMOST" IS FALLOUT FROM THE DUAL NATURE -- BLOCK AND STANDARD -- OF THE 3277 USER TERMINAL, AND IS EXPLAINED BELOW.)

WHEN THE PROGRAM DOES A BD$OUT CALL TO SEND A MESSAGE TO A 3270 SUPPORT DEVICE, THE BLOCK DEVICE INTERFACE (BDI) MOVES THAT MESSAGE ONTO A QUEUE FOR THE STM TO PICK UP. THE BDI DOESN'T EXAMINE THE MESSAGE EXCEPT TO CHECK THAT THE USER IS SPECIFYING A VALID 3270 COMMAND (SEE SECTION 5.1) AND THAT THE DATA DOESN'T CONTAIN INVALID CHARACTERS.

THE STM WILL FIND THE USER'S MESSAGE ON THE DEVICE'S OUTPUT QUEUE AND, AS SOON AS THE LINE IS FREE, WILL "SELECT" THAT DEVICE. IT WILL THEN SEND THE COMMAND AND ANY DATA THAT WERE IN THE USER'S MESSAGE. (IF THE DEVICE CAN'T BE SELECTED (FOR INSTANCE IF IT'S POWERED OFF), THE STM WILL THROW AWAY THE MESSAGE AND WILL RETURN A RESULT OF TTF/TMNS (MESSAGE TRANSMISSION FAILED, COULDN'T BE SENT)).

IF THE COMMAND JUST SENT IS A WRITE-TYPE -- ONE OF WRITE, ERASE/WRITE, COPY (TO THE TARGET DEVICE), OR ERASE ALL UNPROTECTED -- THE STM WILL WAIT FOR THE CONTROL UNIT TO ACKNOWLEDGE RECEIPT AND PROCESSING OF THE MESSAGE. IF THE ACKNOWLEDGEMENT IF GOOD, THE STM BUILDS A T$TOK (MESSAGE

TRANSMISSION OKAY) RESULT FOR THE USER AND ENQUEUES IT ON THE DEVICE'S INPUT QUEUE. IF THE ACKNOWLEDGEMENT IS NEGATIVE, THE STM DOES A SPECIFIC POLL TO RETRIEVE STATUS BITS WHICH DESCRIBE THE REASONS FOR FAILURE.

IF THE STATUS INDICATES THAT THE MESSAGE ITSELF WAS BAD (E.G. BECAUSE OF AN INVALID BUFFER ADDRESS), THE STM WILL SEND THE USER A TTF/TREJ MESSAGE (MESSAGE TRANSMISSION FAILED, REJECTED BY DEVICE). IF THE REASON FOR FAILURE IS TERMINAL / CONTROL UNIT / LINE MALFUNCTION, THE USER WILL GET A TTF/TMNA (MESSAGE TRANSMISSION FAILED, NEVER PROPERLY ACKNOWLEDGED) RESULT AFTER THE STM HAS RETRIED THE MESSAGE THE APPROPRIATE NUMBER OF TIMES. IN EITHER CASE, THE MESSAGE IS CONSIDERED LOST, AND THE USER MUST EXPLICITLY RESEND IT IF HE WANTS TO RETRY THE SAME MESSAGE.

AFTER SENDING A READ-TYPE COMMAND FROM THE USER -- ONE OF READ MODIFIED OR READ BUFFER -- THE STM WAITS FOR THE CONTROL UNIT TO SEND THE DATA BLOCKS WHICH CONTAIN THE READ RESPONSE, ACKNOWLEDGING EACH ONE (THERE MAY BE SEVERAL) AS THE BLOCK COMES IN. THE STM THEN GATHERS ALL DATA BLOCKS INTO A SINGLE MESSAGE, AFFIXES THE TYPE T$RD (RECEIVED DATA) TO IT, MARKS IT AS BEING INPUT FROM THE USER'S READ COMMAND, AND ENQUEUES IT ON THE DEVICE'S INPUT QUEUE. NOTE THAT THERE IS NO EXPLICIT ACKNOWLEDGEMENT FOR THE USER'S OUTPUT MESSAGE (WHICH WAS THE READ COMMAND); THE RECEIPT OF THE READ DATA IS AN IMPLICIT ACKNOWLEDGEMENT THAT THE COMMAND WAS DONE. IF THE READ COMMAND SHOULD FAIL, BECAUSE OF TERMINAL / CONTROL UNIT / LINE MALFUNCTION, THE USER WILL GET A TTF/TMNA RESULT WHICH INDICATES THAT THE READ WAS NOT COMPLETED CORRECTLY BY THE CONTROL UNIT.

THIS VERSION OF THE STM DOES NOT SUPPORT EXPLICIT "CHAINING" OF COMMANDS BY THE USER PROGRAM. IN IBM TERMINOLOGY, CHAINING OCCURS WHEN THE 3270 CONTROL UNIT PERFORMS A SEQUENCE OF SEVERAL COMMANDS FOR A SPECIFIC DEVICE WHILE THAT DEVICE REMAINS SELECTED. A CHAINED COMMAND -- ONE WHICH IS PERFORMED AFTER THE FIRST COMMAND IN A SEQUENCE AND BEFORE DEVICE DESELECTION -- MAY PRODUCE DIFFERENT RESULTS THAN AN IDENTICAL COMMAND WHICH IS NOT CHAINED. SINCE CHAINING IS NOT SUPPORTED, THE FOLLOWING TWO RESTRICTIONS HOLD: 1) EACH WRITE-TYPE COMMAND SHOULD PROVIDE AN EXPLICIT STARTING ADDRESS FOR THE FIRST POSITION TO BE WRITTEN, SINCE THE "CURRENT BUFFER ADDRESS" WON'T CARRY OVER FROM ONE COMMAND TO THE NEXT, AND 2) ALL READ MODIFIED COMMANDS WILL GENERATE DATA FROM THE BEGINNING OF THE BUFFER (AGAIN, SINCE THE CURRENT BUFFER ADDRESS DOESN'T CARRY OVER).

THE STM PERIODICALLY POLLS EACH DEVICE TO LOOK FOR INPUT INITIATED BY THE OPERATOR. IF A POLLED DEVICE HAS AN AID OUTSTANDING (BECAUSE THE OPERATOR HAS HIT THE ENTER OR SOME OTHER ATTENTION KEY) THE 3271 CONTROL UNIT WILL TRANSMIT A DATA MESSAGE, POSSIBLY IN SEVERAL BLOCKS. THE STM WILL GATHER THE BLOCKS INTO A SINGLE MESSAGE, TACK ON A TYPE CODE OF T$RD, MARK THE MESSAGE AS CONTAINING DATA FROM A POLL, AND WILL FORWARD THE MESSAGE TO THE USER PROGRAM. THE PROGRAM WILL PICK UP THIS MESSAGE AT SOME POINT IN THE FUTURE, WHEN IT MAKES A BD$INP CALL.

IF THE PROGRAM WISHES, IT CAN "DISABLE" THE DEVICE VIA A BD$SET CALL. WHILE A DEVICE IS DISABLED THE STM WILL NOT POLL IT FOR INPUT. THIS PROVIDES A WAY FOR THE PROGRAM TO PREVENT THE INPUT OF MESSAGES WHICH IT DOESN'T WANT, PERHAPS BECAUSE IT CAN'T HANDLE THEM AT THE MOMENT. WHEN THE TERMINAL IS RE-ENABLED, THE STM WILL RESUME POLLING THAT TERMINAL AS USUAL. DURING THE TIME THAT A DEVICE IS DISABLED, THE PROGRAM CAN STILL ISSUE COMMANDS, BOTH WRITE- AND READ-TYPE, WHICH THE STM WILL PERFORM.

A 3277 WHICH IS USED AS A PRIME STANDARD USER TERMINAL, AND WHICH IS CAPABLE OF BEING RUN IN TELETYPE EMULATION MODE, IS SUBJECT TO TWO UNUSUAL SITUATIONS, HERE CALLED "BULLETIN" AND "QUIT". THESE ARE SITUATIONS IN WHICH, EVEN THOUGH THE TERMINAL IS ATTACHED TO A USER PROGRAM IN BLOCK MODE, PRIMOS TEMPORARILY TAKES OVER ITS SCREEN FOR THE DISPLAY OF ITS OWN MESSAGES. ANY USER PROGRAM WHICH TALKS TO A BLOCK-MODE 3277 TERMINAL WHICH DOUBLES AS A USER TERMINAL MUST BE PREPARED TO HANDLE EITHER OF THESE SITUATIONS AT ANY TIME.

WHEN A 3277 IS IN BLOCK MODE, ITS USUAL TERMINAL INPUT AND OUTPUT BUFFERS ARE BYPASSED IN FAVOR OF THE DEVICE INPUT AND OUTPUT QUEUES WHICH CARRY BLOCK MESSAGES. HOWEVER, SINCE THE TERMINAL IS STILL A PRIME STANDARD USER TERMINAL, PRIMOS ASSUMES RESPONSIBILITY FOR GETTING SYSTEM MESSAGES ("BULLETINS") OUT TO THE TERMINAL, JUST AS IT WOULD FOR ANY OTHER USER TERMINAL. ANY STRING OF CHARACTERS FOUND IN THE TERMINAL'S CHARACTER OUTPUT BUFFER WHILE THE TERMINAL IS ATTACHED IN BLOCK MODE IS ASSUMED TO BE A HIGH-PRIORITY SYSTEM MESSAGE AND WILL BE TRANSMITTED TO THE 3277 USING THE FOLLOWING PROTOCOL.

-- THE SCREEN IS REFORMATTED AS FOR PRIME STANDARD MODE (24 LINES OF 79 CHARACTERS). IT IS NOT CLEARED OF DATA.

-- THE BULLETIN WILL BE DISPLAYED SOMEWHERE ON THE SCREEN.

-- THE STM WILL CREATE A STATUS RESULT OF T$BROK (SCREEN "BROKEN" BY PRIMOS) AND WILL ENQUEUE THIS ON THE BLOCK DEVICE INPUT QUEUE FOR THE USER PROGRAM.

THE PROGRAM WILL PICK UP THE T$BROK STATUS ON A FUTURE BD$INP CALL. THE PROGRAM IS RESPONSIBLE FOR REBUILDING THE SCREEN AS IT WANTS THE SCREEN TO APPEAR.

USE OF THE "TYPERS" (TNOU, FORTRAN WRITE, ETC.) WHILE THE USER TERMINAL IS IN BLOCK MODE WILL CAUSE THE SAME PHENOMENON TO OCCUR, AND IS NOT RECOMMENDED.

IF THE PROGRAM RUNNING A 3277 TERMINAL IN BLOCK MODE DOESN'T HAVE "BREAKS INHIBITED", THE OPERATOR CAN CAUSE A BREAK BY HITTING THE CANCEL (PA2) KEY. THIS IS ALWAYS TRUE FOR A 3277 WHICH IS A USER TERMINAL, WHETHER THE TERMINAL IS IN TELETYPE EMULATION MODE OR IN BLOCK MODE. NOTE THAT THE BD$SET "DISABLE" FUNCTIONS AS AN IMPLICIT "BREAKS INHIBIT", BECAUSE WHILE THE STM HAS SUSPENDED POLLING THE TERMINAL, IT WON'T SEE A QUIT REQUEST. THE "QUIT"

HAS THE FOLLOWING EFFECT:

-- THE NORMAL BREAK MESSAGE "QUIT," OR "QUIT." WILL APPEAR SOMEWHERE ON THE SCREEN. THE SCREEN IS NOT CLEARED OF DATA.

-- THE BLOCK INPUT AND OUTPUT QUEUES FOR THIS DEVICE ARE FLUSHED OF ALL MESSAGES, AND THE DEVICE IS MARKED IN AN INTERNAL TABLE AS "FORCIBLY DETACHED".

THE TERMINAL IS NOW BACK AT COMMAND LEVEL, AND THE OPERATOR CAN TYPE COMMANDS, RUN NEW PROGRAMS, AND SO FORTH, AS HE NORMALLY WOULD WHEN THE TERMINAL IS IN TELETYPE EMULATION MODE.

IF THE OPERATOR SHOULD TYPE "START" OR "S" BEFORE RUNNING ANOTHER PROGRAM, THE BLOCK MODE PROGRAM WILL BE RESTARTED WHERE IT WAS WHEN THE QUIT OCCURRED. THE PROGRAM DOESN'T KNOW THAT ANYTHING HAS HAPPENED; HOWEVER, THE STATE OF ITS TERMINAL MAY HAVE CHANGED SUBSTANTIALLY (QUEUED OUTPUTS LOST, SCREEN FORMAT DESTROYED). WE HAVE ARRANGED THAT THE USER WILL BE INFORMED ABOUT THE QUIT BY GETTING AN ERROR CODE OF E$DFD (DEVICE FORCIBLY DETACHED) WHENEVER THE PROGRAM DOES

-- A BD$OUT, BD$SET, BD$INF, OR BD$INP CALL FOR THIS (THE USER TERMINAL) DEVICE; OR

-- A BD$INP CALL WITH A KEY OF -1 (INPUT FROM ANY DEVICE).

IF THE PROGRAM WAS WAITING ON A BD$INP CALL WHEN THE QUIT OCCURRED, IT WILL STILL GET AN E$DFD CODE WHEN IT RETURNS FROM THAT CALL.

WHEN THE PROGRAM SEES AN E$DFD ERROR CODE IT SHOULD REBUILD THE 3277 SCREEN, AND (IF DESIRED) RECOVER ANY PREVIOUS OUTPUT MESSAGE FOR WHICH A RESULTANT STATUS OF T$TOK HAS NOT BEEN RECEIVED (SINCE ANY SUCH MESSAGES WERE FLUSHED FROM THE OUTPUT QUEUE).

3.6.4.1 BD$OUT: OUTPUT FROM USER TO SUPPORT TRAFFIC MANAGER

A USER PROGRAM INVOKES THIS CALL TO SEND A MESSAGE TO A 3270 TERMINAL WHICH IT HAS PREVIOUSLY ATTACHED. A "MESSAGE", IN 3270 PARLANCE, CONSISTS OF A COMMAND (ONE OF WRITE, ERASE/WRITE, COPY, ERASE ALL UNPROTECTED, READ MODIFIED, AND READ BUFFER), AND AN OPTIONAL DATA PORTION WHICH CONSISTS OF ORDERS (POSITIONING AND FORMATTING CHARACTERS) AND GRAPHIC CHARACTERS TO BE DISPLAYED ON THE SCREEN. EVERY MESSAGE WHICH THE USER SENDS TO ITS 327C TERMINAL (VIA BD$OUT AND THE SUPPORT TRAFFIC MANAGER) MUST INDICATE THE COMMAND TO BE INVOKED AS PART OF THE "CONTROL" INFORMATION WHICH ACCOMPANIES THE "DATA" (ORDERS DISPLAY DATA) PART OF THE MESSAGE.

DATA FORMAT

THE OUTPUT DATA IN DATA CONTAINS THE ORDERS AND DATA CHARACTERS TO BE WRITTEN TO THE DEVICE. THE COMMAND WHICH IS TO TRANSFER THE DATA, AND THE WRITE OR COPY CONTROL CHARACTER IF ANY, ARE SPECIFIED SEPARATELY IN OUTARR (SEE BELOW); THEY ARE NOT CONSIDERED PART OF THE DATA. BD$OUT EXAMINES THE DATA BUFFER TO ENSURE THAT IT DOESN'T CONTAIN DATA LINK CONTROL CHARACTERS (E.G. SOH, STX), BUT IT MAKES NO CHECKS FOR VALID 3270 DATA FORMAT.

OUTPUT DATA IS MEANINGFUL ONLY IF IT IS TO ACCOMPANY A WRITE OR ERASE/WRITE COMMAND. IF THE COMMAND IN OUTARR(1) IS ONE OF COPY, ERASE ALL UNPROTECTED, READ MODIFIED, OR READ BUFFER, THE USER'S DATA FIELD IS IGNORED.

THE MAXIMUM ALLOWABLE DATA LENGTH (FIELD LENGTH ON THE BD$OUT CALL) IS 2048 CHARACTERS. THE MINIMUM IS ZERO, SINCE NO WRITE DATA NEED ACCOMPANY A WRITE-TYPE COMMAND.

ALL FIELDS SHOULD BE IN ASCII UNLESS THE PROGRAM HAS PREVIOUSLY DONE A BD$SET CALL WITH A KEY OF K$EBCD TO INDICATE THAT THE PROGRAM WILL BE "SPEAKING" IN EBCDIC. NOTE: THE ASCII TO EBCDIC CONVERSIONS USED BY DPTX ARE NOT THOSE USED BY IBM. A PROGRAM WHICH USES ASCII TO TALK WITH A DPTX DEVICE WHICH IS EBCDIC SHOULD FOLLOW THE PRIME ASCII CHARACTER CONVENTIONS DESCRIBED IN A SEPARATE DPTX DOCUMENT.

PROTOCOL-SPECIFIC PARAMETERS (OUTARR(1-4))

THERE ARE THREE CONTROL FIELDS USED TO COMPLETE THE SPECIFICATION OF A 3270 MESSAGE. ONLY THE FIRST (THE 3270 COMMAND) IS REQUIRED, BUT THIS MUST BE SPECIFIED ON EACH BD$OUT CALL FOR A 3270 SUPPORT DEVICE.

OUTARR(1) A CODE FOR THE COMMAND TO BE TRANSMITTED AS PART OF THIS MESSAGE:
1: WRITE (WCC IS IN OUTARR(2))
2: ERASE/WRITE (WCC IS IN OUTARR(2))
3: COPY (CCC IS IN OUTARR(2); "FROM" DEVICE NUMBER IS IN OUTARR(3))

4: ERASE ALL UNPROTECTED
5: READ MODIFIED
6: READ BUFFER

OUTARR(2) THE WRITE CONTROL CHARACTER OR COPY CONTROL CHARACTER (IF ANY) WHICH IS TO ACCOMPANY THIS MESSAGE. THIS IS A ONE-BYTE FIELD (BITS 9-16 OF OUTARR(2)). IT IS IGNORED UNLESS THE COMMAND (OUTARR(1)) IS WRITE, ERASE/WRITE, OR COPY. ONLY THE LOW-ORDER SIX BITS (THE ACTUAL WRITE OR COPY CONTROL BITS) NEED BE SUPPLIED BY THE CALLER. SEE IBM DOCUMENTATION FOR THE MEANING OF BITS IN THIS CHARACTER.

OUTARR(3) THE LOGICAL STATION ID OF THE DEVICE TO BE COPIED "FROM", IF OUTARR(1) IS A COPY COMMAND. THE "FROM" DEVICE MUST HAVE BEEN ATTACHED TO THE USER PREVIOUS TO THIS CALL.

OUTARR(4) UNUSED.

PROTOCOL-SPECIFIC ERRORS (CODE)

THE FOLLOWING ERROR CODES, WHICH ARE SPECIFIC TO 3270 SUPPORT, MAY BE RETURNED BY BD$OUT IF THE FIELDS IN OUTARR ARE INCORRECT. NOTE THAT ALL BD$OUT ERROR CODES LISTED IN THE CALL DESCRIPTION IN SECTION 3 ARE STILL POSSIBLE.

E$SICM: INVALID 3270 COMMAND CODE IN OUTARR(1).

E$SBCF: BAD COPY "FROM" DEVICE NUMBER IN OUTARR(3). THE DEVICE SPECIFIED IN OUTARR(3) MUST BE ANOTHER 3270 DEVICE WHICH BELONGS TO THE CALLER.

3.6.4.2 BD$INP: INPUT FROM SUPPORT TRAFFIC MANAGER TO USER

AN INPUT MESSAGE FROM A 3270 DEVICE IS PICKED UP BY THE SUPPORT TRAFFIC MANAGER (STM) EITHER DURING A STM-INITIATED POLL (THE OPERATOR HAS PRESSED A PROGRAM ACCESS KEY) OR IN RESPONSE TO A PROGRAM-INITIATED READ MODIFIED OR READ BUFFER COMMAND. THIS INPUT MESSAGE IS RETURNED TO THE USER PROGRAM WHEN THE LATTER DOES A BD$INP CALL WHICH RETURNS AN INPARR(1), THE INPUT TYPE CODE, OF T$RD (RECEIVED DATA AVAILABLE). THE INFORMATION IN INPARR(7) INDICATES WHETHER A POLL OR AN EXPLICIT READ COMMAND BROUGHT IN THIS MESSAGE.

IN ADDITION TO PICKING UP DATA MESSAGES, BD$INP ALSO RETURNS THE RESULTS OF TRANSMISSIONS INITIATED BY THE PROGRAM (MESSAGE FAILED OR MESSAGE SUCCEEDED) AND INDICATIONS OF DEVICE STATUS CHANGES. THE TYPE CODE IN INPARR(1) IDENTIFIES THE CATEGORY OF THIS INPUT AS DATA, RESULT, OR STATUS. FOR TYPES OTHER THAN T$RD, DATA IS UNUSED BUT THE PROTOCOL-SPECIFIC AREA (INPARR(7-10)) CONTAINS POSSIBLE USEFUL INFORMATION.

A SUCCESSFUL BD$INP CALL WILL RETURN ONE OF THE FOLLOWING TYPE/SUBTYPE CODES:

T$RD:	DATA RECEIVED FROM TERMINAL.
T$TOK:	YOUR PREVIOUS MESSAGE SUCCESSFULLY TRANSMITTED.
TTF/TMNS:	YOUR PREVIOUS MESSAGE WAS DISCARDED WITHOUT BEING SENT BECAUSE THE TERMINAL CAN'T BE SELECTED.
TTF/TREJ:	YOUR PREVIOUS MESSAGE WAS REJECTED BY THE 3270 HARDWARE AS BEING AN ILL-FORMED MESSAGE.
TTF/TMNA:	YOUR PREVIOUS MESSAGE WAS LOST BECAUSE OF 3270 HARDWARE OR COMMUNICATIONS LINE FAILURE.

DATA_FORMAT

THE INPUT DATA IN DATA AFTER A BD$INP CALL (WHICH RETURNED AN INPARR(1) OF T$RD, RECEIVED DATA AVAILABLE) WILL CONSIST OF THE RECEIVED AID AND ALL FOLLOWING DATA AND FORMATTING CHARACTERS. THE EXACT FORMAT OF THE INPUT DATA DEPENDS ON THE TYPE OF AID BYTE AND THE METHOD BY WHICH THE DATA WAS RECEIVED.

THE DATA GENERATED IN RESPONSE TO A (STM-INITIATED) POLL OR IN RESPONSE TO A READ MODIFIED COMMAND INVOKED BY THE USER CONSISTS OF:

-- AID BYTE. THIS IS THE ONLY CHARACTER IN DATA IF THE AID WAS A "SHORT READ" TYPE (CLEAR OR PROGRAM ACCESS KEY).

-- CURSOR ADDRESS. THIS IS TWO BYTES OF ADDRESS IN IBM 3270 ADDRESS FORMAT. IT IS PRESENT IF THE AID WAS A "READ MODIFIED" TYPE (NO AID GENERATED, ENTER, OR PROGRAM FUNCTION KEY).

-- READ MODIFIED DATA. THIS CONSISTS OF MODIFIED FIELDS, OR THE ENTIRE SCREEN IF THE BUFFER IS UNFORMATTED, WITH NULLS SUPPRESSED. THIS MAY BE EMPTY IF THERE WERE NO MODIFIED FIELDS.

THE DATA GENERATED IN RESPONSE TO A READ BUFFER COMMAND INVOKED BY THE USER PROGRAM CONSISTS OF AN AID BYTE, TWO BYTE CURSOR ADDRESS, AND THE READ BUFFER DATA (ENTIRE BUFFER CONTENTS WITH ATTRIBUTES IDENTIFIED AND NULLS INCLUDED).

A READ MODIFIED INPUT WHICH IS A TEST REQ MESSAGE WILL APPEAR TO THE USER AS AN AID OF (CHARACTER) "0" AND A DUMMY CURSOR ADDRESS INDICATING THE POSITION AT ROW 1, COLUMN 1, FOLLOWED BY READ MODIFIED DATA.

ALL CHARACTER FIELDS WILL BE IN ASCII UNLESS THE PROGRAM HAS PREVIOUSLY DONE A BD$SET CALL WITH A KEY OF K$EBCD TO INDICATE THAT THE PROGRAM WILL BE "SPEAKING" IN EBCDIC. NOTE: THE ASCII TO EBCDIC CONVERSIONS USED BY DPTX ARE NOT THOSE USED BY IBM. A PROGRAM WHICH USES ASCII TO TALK WITH A DPTX DEVICE WHICH IS EBCDIC SHOULD FOLLOW THE PRIME ASCII CHARACTER CONVENTIONS DESCRIBED IN A SEPARATE DPTX DOCUMENT.

THE MAXIMUM INPUT DATA LENGTH HANDLED BY THE SUPPORT TRAFFIC MANAGER IS 2048 CHARACTERS. THE PROGRAMMER SHOULD TAKE CARE NOT TO CREATE ANY OUTPUT FORMATS WHICH, BECAUSE THEY CONTAIN MANY UNPROTECTED FIELDS, MIGHT GENERATE MORE THAN 2046 CHARACTERS OF READ MODIFIED DATA. THE MINIMUM INPUT DATA LENGTH (ON A BD$INP CALL WHICH RETURNS A TYPE OF T$RD) IS ONE, SINCE THERE WILL ALWAYS BE AT LEAST AN AID BYTE.

PROTOCOL-SPECIFIC PARAMETERS (INPARR(7-10))

THE FIELDS IN THESE FOUR WORDS HAVE DIFFERENT INTERPRETATIONS DEPENDING ON THE INPUT TYPE IN INPARR(1).

TYPE = I$RD (RECEIVED DATA)

INPARR(7) METHOD BY WHICH THIS DATA WAS RECEIVED:

- 5: READ MODIFIED COMMAND GENERATED BY THE USER PROGRAM
- 6: READ BUFFER COMMAND GENERATED BY THE USER PROGRAM
- 7: POLL (MESSAGE TRANSMISSION GENERATED BY OPERATOR'S PRESSING A PROGRAM ACCESS KEY, PICKED UP BY SUPPORT TRAFFIC MANAGER'S POLL)

INPARR(8) UNUSED.

INPARR(9) UNUSED.

INPARR(10) UNUSED.

TYPE = I$IOK (TRANSMISSION SUCCESSFUL)

INPARR(7) COMMAND WHICH WAS PERFORMED AS PART OF THE TRANSMITTED MESSAGE (WILL BE WRITE-TYPE OR COPY):

- 1: WRITE
- 2: ERASE/WRITE
- 3: COPY
- 4: ERASE ALL UNPROTECTED

INPARR(8) UNUSED.

INPARR(9) UNUSED.

INPARR(10) UNUSED.

TYPE = I$IF (TRANSMISSION FAILED). INPARR(2) (INPUT SUBTYPE CODE) CONTAINS MORE (PROTOCOL INDEPENDENT) INFORMATION ABOUT THE NATURE OF THE FAILURE.

INPARR(7) COMMAND WHICH WAS IN PROGRESS AT THE TIME OF THE FAILURE:

- 1: WRITE (INITIATED BY USER PROGRAM)

2: ERASE/WRITE (INITIATED BY USER PROGRAM)
3: COPY (INITIATED BY USER PROGRAM)
4: ERASE ALL UNPROTECTED (INITIATED BY USER PROGRAM)
5: READ MODIFIED (INITIATED BY USER PROGRAM)
6: READ BUFFER (INITIATED BY USER PROGRAM)
7: POLL (INITIATED BY SUPPORT TRAFFIC MANAGER)

INPARR(8) SENSE AND STATUS BYTES (IF ANY) RETRIEVED BY THE SUPPORT TRAFFIC MANAGER AFTER IT HAS DECIDED THE ERROR IS NON-RECOVERABLE. THESE TWO BYTES ARE ALWAYS PRESENTED TO THE USER PROGRAM IN EBCDIC, REGARDLESS OF THE ACTUAL LINE CODE OF THE DEVICE OR THE CHARACTER CODE SELECTED BY THE USER, BECAUSE AN EBCDIC SENSE AND STATUS REPRESENTATION CAN BE ANALYZED AS A BIT PATTERN WHEREIN THE DIFFERENT ERROR CONDITIONS ARE SPECIFIED BY SPECIFIC BITS. SEE IBM 3270 DOCUMENTATION FOR THE MEANING OF THE BITS IN THESE CHARACTERS.

INPARR(9) UNUSED.

INPARR(10) UNUSED. PROTOCOL-SPECIFIC ERRORS (CODE)

NONE DEFINED FOR 3270 SUPPORT PROTOCOL.

3.6.4.3 BD$INF: OBTAIN INFORMATION ABOUT A 3270 SUPPORT DEVICE

A CALL TO BD$INF WITH KEY = K$INFN WILL RETURN THE NAME OF THE 3270 SUPPORT DEVICE SPECIFIED BY THE NUMBER DEVICE AND SOME STATUS INFORMATION AS DESCRIBED IN SECTION 3 (GENERAL INFORMATION ON CALLS). THERE IS CURRENTLY NO PROTOCOL-DEFINED "STATUS DATA" OR PROTOCOL-DEFINED STATUS INFORMATION FOR 3270 SUPPORT, SO A CALL TO BD$INF WITH A KEY OF K$INFD WILL RETURN JUST THOSE FIELDS IN INFARR(1-6) WHICH WOULD HAVE BEEN RETURNED ON A CALL WITH A KEY OF K$INFN.

4 DPTX

4.1 TERMINAL SUPPORT FEATURES

PRIMOS NOW SUPPORTS A NEW TERMINAL TYPE - THE IBM 3271/3277 TERMINAL. THIS DEVICE IS A BLOCK-MODE TERMINAL WHICH USES SYNCHRONOUS COMMUNICATIONS PROTOCOLS. THE VERSION SUPPORTED BY PRIMOS USES THE BINARY SYNCHRONOUS COMMUNICATIONS PROTOCOL. TERMINAL SUPPORT COMES IN FLAVORS: ONE IS THE ABILITY TO USE THE IBM TERMINAL AS A TERMINAL INTERACTING WITH PRIMOS (USER OR ASSIGNED TERMINAL); THE OTHER TWO ARE THE ABILITY TO USE EITHER THE IBM TERMINAL OR THE OWL 1200 TERMINAL AS THOUGH CONNECTED TO AN IBM HOST MAINFRAME.

THESE FEATURES ARE SOLD SEPARATELY FROM PRIMOS. THE NAME CHOSEN FOR THE PRODUCTS IS "DPTX." THE THREE FLAVORS ARE:

DPTX/TSF	3271/3277 TERMINALS ARE PRIME TERMINALS
DPTX/TCF	3271/3277 TERMINALS ATTACHED TO PRIME ARE IBM HOST TERMINALS
DPTX/DSC	OWL 1200 TERMINALS ATTACHED TO PRIME ARE IBM HOST TERMINALS

WITH EITHER TCF OR DSC OPTIONS, THE USER MAY INTERACT WITH A PROCESS CONTAINED WITHIN THE HOST MAINFRAME. THE SOFTWARE LINK BETWEEN A USER-SPACE PROCEDURE AND THE HOST IS THE BLOCK DEVICE INTERFACE (SEE SECTION 3.8)

THE TSF OPTION IS REQUIRED FOR USE WITH REAL 3271/3277 TERMINALS (AS MENTIONED ABOVE). CURRENT PRIMOS PROGRAMS WILL STILL RUN PROPERLY EXCEPT THAT SINGLE-CHARACTER INPUT IS NOT POSSIBLE. IN ORDER TO DO BLOCK MODE I/O THE USER PROCESS MUST ALSO USE THE BLOCK DEVICE INTERFACE.

4.2 INFORMATION FOR THE SYSTEM ADMINISTRATOR

4.2.1 INSTALLING DPTX FROM THE MASTER DISK

DPTX IS SOLD IN THE THREE PACKAGES DISCUSSED ABOVE. NOTE THAT THE DPTX/TCF PACKAGE INCLUDES THE DSC AND TSF PACKAGES PLUS THE TCF FUNCTIONALITY.

IN EACH CASE, THE DPTX PACKAGE WHICH IS PURCHASED IS PROVIDED IN A UFD NAMED DPTX-XXX WHERE "XXX" IS THE NAME OF THE PACKAGE (DSC, TSF, OR TCF).

THE DPTX-XXX UFD CONTAINS THE COMMAND FILES TO BUILD THE DPTX SYSTEM. THE DPTX-XXX UFD ALSO HAS SEVERAL SUBUFDS:

-- SOURCE CONTAINS THE SOURCE MODULES FOR THE INTERFACE PROGRAMS, THE CONFIGURATION PROGRAM, AND THE LIBRARIES.

-- BINARY IS A SCRATCH-PAD SUBUFD WHERE BINARIES PRODUCED FROM BUILDING THE SYSTEM ARE PLACED.

-- CMDNCO CONTAINS THE COMMANDS USED TO INTERFACE TO DPTX, AND THE COMMAND TO INVOKE THE CONFIGURATION PROGRAM.

-- SYSTEM CONTAINS PROGRAMS AND COMMAND FILES USED TO START UP DPTX, AND THE SHARED SEGMENT IMAGES NECESSARY FOR RUNNING DPTX.

-- SYSCOM CONTAINS INSERT FILES USED FOR APPLICATION PROGRAMS WHICH ARE TO INTERFACE TO DPTX.

-- LIB CONTAINS THE OBJECT LIBRARIES USED FOR APPLICATION PROGRAMS WHICH ARE TO INTERFACE TO DPTX.

-- INFO CONTAINS DOCUMENTATION FOR DPTX.

DPTX IS INSTALLED BY COPYING ALL FILES IN SUBUFDS SYSCOM, CMDNCO, LIB, AND SYSTEM TO THEIR RESPECTIVE UFDS ON THE COMMAND PARTITION.

SHOULD IT BE NECESSARY TO REBUILD DPTX SOFTWARE FROM THE SOURCES, THE COMMAND FILE C_DPTX-XXX, WHERE "XXX" IS THE PRODUCT NAME (DSC, TSF, OR TCF), SHOULD BE RUN. THE COMMAND FILE WILL REQUEST THAT SUBUFD SYSCOM BE COPIED TO UFD SYSCOM. THIS CAN BE DONE VIA FUTIL, USING THE UFDCPY COMMAND.

WHEN THE COMMAND FILE HAS COMPLETED EXECUTING, SUBUFDS CMDNCO, SYSTEM, AND LIB SHOULD BE COPIED TO THEIR RESPECTIVE UFDS ON THE COMMAND PARTITION.

SEVERAL ADDITIONS MUST BE MADE TO THE PRIMOS COLD START CONFIGURATION FILES:

-- THE SYNCHRONOUS LINE(S) USED FOR DPTX MUST BE CONFIGURED VIA THE SMLC DIRECTIVE, IF THE DEFAULT DATA SET PARAMETERS DO NOT MATCH YOUR MODEM. THIS DIRECTIVE IS PLACED IN THE CONFIGURATION DATA FILE WHICH IS READ VIA THE COLD START CONFIG -DATA LINE IN THE C_PRMO FILE.

-- DPTX/DSC ONLY: EACH OWL-1200 TERMINAL TO BE USED AS A 3277 TERMINAL MUST HAVE ITS AMLC INPUT BUFFER SIZE INCREASED TO 2000 CHARACTERS. THIS IS DONE BY INCLUDING THE DIRECTIVE

AMLBUF XX 2000

WHERE "XX" IS THE LINE NUMBER FOR THE OWL-1200, IN THE CONFIGURATION DATA FILE. REMEMBER THAT THE AMLC LINE NUMBER IS IN OCTAL AND IS EQUAL TO (DECIMAL USER NUMBER) - 2.

-- DPTX/TSF AND DPTX/TCF ONLY: EACH 3277 TERMINAL MAY HAVE ASSOCIATED WITH IT A PRIMOS USER NUMBER, AND HENCE, AN AMLC LINE NUMBER. (THIS RELATIONSHIP IS SET UP BY USE OF DPTCFG.) EACH SUCH LINE MUST BE LOGICALLY DISCONNECTED FROM

PRIMOS SO THAT IT CAN BE USED BY DPTX. THIS IS ACCOMPLISHED BY ADDING THE STATEMENT

```
AMLC TTYNOP XX
```

WHERE "XX" IS THE NUMBER OF THE LINE BEING REPLACED BY A 3277 TERMINAL, TO THE C_PRMO FILE IN UFD CMDNCO. REMEMBER THAT THE AMLC LINE NUMBER IS IN OCTAL AND IS EQUAL TO (DECIMAL USER NUMBER) - 2.

-- DPTX/DSC AND DPTX/TCF ONLY: THE INTERFACE PROGRAMS TO DPTX/DSC AND DPTX/TCF PROVIDED BY PRIME ARE SHARED. THE FILE C_PRMO IN UFD CMDNCO MUST CONTAIN STATEMENTS TO INSTALL THE SHARED CODE. THE FILE C_DPTXSHR LOCATED IN UFD SYSTEM AFTER DPTX HAS BEEN INSTALLED, OR THE FOLLOWING STATEMENTS WILL ACCOMPLISH THIS

```
OPR 1
SHARE SYSTEM>02015A 2015
SHARE SYSTEM>02015B 2015
SHARE SYSTEM>T2015A 2015 /* INCLUDED ONLY FOR TCF
SHARE SYSTEM>T2015B 2015 /* INCLUDED ONLY FOR TCF
OPR 0
```

4.2.2 CONFIGURATION OF DPTX FOR YOUR SYSTEM

DPTCFG IS AN EXTERNAL COMMAND WHICH COMPILES A DPTX CONFIGURATION SOURCE FILE, AND WRITES A BINARY OUTPUT FILE WHICH IS READ AT DPTX INITIALIZATION. THIS ALLOWS ERRORS IN CONFIGURATION TO BE FOUND AND CORRECTED AT COMPILE TIME, INSTEAD OF DPTX INITIALIZATION. IT ALSO SIMPLIFIES CHANGES TO CONFIGURATION AND THE MAINTENANCE OF SEVERAL CONFIGURATIONS. REFER TO DOCUMENTATION ON THE MASTER DISK FOR DETAILS OF DPTCFG.

4.2.3 ENABLING DPTX

DPTX IS ENABLED BY USE OF THE "DPTX" COMMAND. (THIS COMMAND IS VALID ONLY FROM THE SYSTEM CONSOLE.) THE "DPTX" COMMAND HAS TWO OPTIONS: -ON AND -DATA.

DPTX -ON

THIS CAUSES THE DPTX SYSTEM TO BE CONFIGURED AND ENABLED. CONFIGURATION IS PERFORMED ACCORDING TO THE DPTCFG OUTPUT FILE DPTCON LOCATED IN UFD CMDNCO ON THE COMMAND PARTITION.

DPTX -DATA TREENAME

THIS CAUSES THE DPTX SYSTEM TO BE ENABLED AND CONFIGURED ACCORDING TO THE FILE SPECIFIED IN TREENAME. THIS FILE IS AN OUTPUT FILE PRODUCED BY DPTCFG.

NOTE THAT ONCE DPTX HAS BEEN ENABLED BY EITHER DPTX -ON OR DPTX -DATA TREENAME, ENTERING EITHER COMMAND AGAIN WILL BE IGNORED.

4.2.4 STARTING UP DPTX

THERE ARE FOUR POSSIBLE USER PROCESSES WHICH MUST RUN IN ORDER TO START UP DPTX. DPTX MUST HAVE BEEN ENABLED VIA THE DPTX COMMAND BEFORE STARTING THESE PROCESSES. THE PROCESSES ARE INVOKED BY EXECUTING PHANTOM COMMAND FILES, LOCATED IN UFD SYSTEM. THE FOUR PHANTOM COMMAND FILES, AND THE FUNCTION OF THE ASSOCIATED PROCESS, ARE

PH_BSC

THIS STARTS UP THE PROCESS TO MANAGE BINARY SYNCHRONOUS COMMUNICATIONS. DPTX MUST BE ENABLED BEFORE STARTING UP THIS PROCESS OR PRIMOS WILL HALT.

PH_EM

THIS STARTS UP THE 3270 EMULATOR PROCESS.

PH_TM

THIS STARTS UP THE PROCESS TO MANAGE TRAFFIC TO AND FROM IBM 3270-COMPATIBLE DEVICES.

PH_DH

THIS STARTS UP THE PROCESS WHICH HANDLES THE INTERFACE BETWEEN 3277 TERMINALS BEING USED AS PRIME USER TERMINALS AND PRIMOS.

EACH PART OF DPTX REQUIRES PHANTOMS TO BE RUN AS FOLLOWS:

DPTX/DSC: PH_BSC, PH_EM

DPTX/TSF: PH_BSC, PH_TM, PH_DH

DPTX/TCF: PH_BSC, PH_EM, PH_TM, PH_DH

EACH PHANTOM IS STARTED UP BY TYPING

```
PH PH_XX
```

ON THE SYSTEM CONSOLE, WHERE "PH_XX" IS ONE OF THE FOUR PHANTOM COMMAND FILE NAMES.

AFTER THE PHANTOMS HAVE LOGGED IN, THEIR PRIORITY MUST BE CHANGED. THIS IS DONE BY TYPING

```
CHAP -UU N
```

WHERE "UU" IS THE USER NUMBER , AND "N" IS THE NEW PRIORITY LEVEL FOR USER UU.

THE USER ASSIGNED TO PH_BSC MUST BE SET TO PRIORITY LEVEL 3.

THE USERS ASSIGNED TO PH_EM AND PH_TM MUST BE SET TO PRIORITY LEVEL 2.

THE USER ASSIGNED TO PH_DH MUST BE LEFT AT THE DEFAULT PRIORITY LEVEL.

4.3 DPTX/TCF OPERATION

DPTX/TCF (TRANSPARENT CONNECT FACILITY) ALLOWS THE USER OF A 3270 OR EQUIVALENT TERMINAL (MORE SPECIFICALLY -- AT PRESENT -- A 3277 TERMINAL WITH 3271 CONTROL UNIT UNDER BISYNC PROTOCOL USING THE EBCDIC INTERFACE CODE -- OR COMPATIBLE DEVICE) TO COMMUNICATE THROUGH A PRIME MACHINE TO AN IBM 360/370 MAINFRAME APPLICATIONS PROGRAM SPECIFICALLY DESIGNED FOR 3270 TERMINAL DEVICES. TCF OPERATES AS A USER PROCESS ON A PRIME SYSTEM, AND USES DPTX/DSC (IBM HOST TO PRIME SYSTEM) AND DPTX/TSF (PRIME SYSTEM TO 3270 TERMINAL) TO HANDLE THE ACTUAL COMMUNICATIONS BETWEEN THE IBM HOST, THE 3270 TERMINAL AND THE PRIME SYSTEM.

4.3.1 DIFFERENCES BETWEEN DPTX/TCF AND NORMAL 3270 OPERATION

DPTX/TCF IS ALMOST ENTIRELY TRANSPARENT TO THE TERMINAL USER IN NORMAL OPERATIONS. WITH THE FOLLOWING EXCEPTIONS, THE USER IS EFFECTIVELY USING A SIMILAR TERMINAL DIRECTLY CONNECTED TO THE IBM HOST:

- THE TCF INVOCATION ITSELF. THIS IS A FUNCTION OF PRIMOS, AND IF THE INVOCATION IS CORRECTLY CARRIED OUT, THE USER WILL NOT BE REQUIRED TO INTERACT WITH PRIMOS UNTIL TCF TERMINATES.

- WARNING AND ERROR MESSAGES SENT TO THE USER BY TCF TO NOTIFY HIM OF UNUSUAL CONDITIONS. IN THE CASE OF WARNING MESSAGES, THE USER MERELY WAITS (APPROXIMATELY 3 SECONDS) FOR A NEW SCREEN IMAGE FROM THE HOST TO APPEAR.

  ERROR MESSAGES, ON THE OTHER HAND, INDICATE A SERIOUS PROBLEM FROM WHICH TCF CANNOT RECOVER. THE USER MAY RE-INVOKE THE PROCESS, AND IF SUCCESSFUL, RESUME COMMUNICATIONS. SHOULD A SECOND INVOCATION FAIL, HE SHOULD CONTACT THE SYSTEM ADMINISTRATOR.

- THE UNAVAILABILITY OF THE PA2 (OR CANCEL) KEY. THIS KEY IS RESERVED UNDER PRIMOS FOR INTERRUPTING A USER PROCESS ATTACHED TO A PARTICULAR TERMINAL. THE EFFECT OF DEPRESSING THIS KEY (CALLED A 'BREAK' OR 'QUIT') MAY BE INHIBITED BY THE USER PROCESS UNDER EXECUTION. THIS IS THE CASE WITH TCF. WERE IT OTHERWISE, THE USER MIGHT INADVERTENTLY CAUSE TCF TO HALT, WHICH WOULD DISRUPT COMMUNICATIONS WITH THE IBM HOST, MAKING IT THINK THAT THE TERMINAL HAD 'DIED', POSSIBLY DAMAGING DATA (DEPENDING UPON THE REMOTE PROCESS) WITHOUT THE USER'S KNOWLEDGE OR BEYOND HIS CONTROL. THEREFORE, SHOULD THE USER DEPRESS

THE PA2 KEY, NOTHING WILL BE SENT TO THE HOST, ALTHOUGH FURTHER INPUT WILL BE INHIBITED. TO CONTINUE, MERELY PRESS THE 'RESET' KEY.

4.3.2 INVOCATION

DPTX/TCF IS INVOKED BY ENTERING:

TCF -H[OST] HNAME -T[ERMINAL] TNAME [-Q[UIT] 'Q_STRING']

ON A SYSTEM WHERE BOTH DPTX/TSF AND DPTX/DSC ARE EXISTENT AND RUNNING, AND WHERE:

HNAME — IS THE REQUIRED PRIMOS NAME FOR THE REMOTE IBM HOST MAINFRAME TO WHICH THE USER WISHES TO CONNECT. THIS NAME (AMONG OTHERS) IS DEFINED BY THE SYSTEM ADMINISTRATOR AT CONFIGURATION TIME.

TNAME — IS THE REQUIRED PRIMOS NAME OF THE IBM 3270-TYPE TERMINAL FROM WHICH THE USER WISHES TO COMMUNICATE. THIS MAY MOST EASILY BE IDENTIFIED AS THE TERMINAL FROM WHICH THE INVOCATION IS TAKING PLACE BY SPECIFYING THE NAME AS '*'. VALID TERMINAL NAMES ARE DEFINED BY THE SYSTEM ADMINISTRATOR AT CONFIGURATION TIME.

Q_STRING — IS THE QUIT STRING (MAXIMUM EIGHT CHARACTERS) THE USER WISHES TO USE TO FORCE THE PROGRAM TO RETURN TO PRIMOS COMMAND LEVEL. WHEN THIS STRING IS ENTERED AT THE TERMINAL, THE PROGRAM BREAKS THE CONNECTION WITH THE HOST, PRINTS OUT 'TCF HALT' AND RETURNS TO PRIMOS. TO THE HOST, THIS ACTION APPEARS AS IF THE TERMINAL WAS POWERED OFF. MOST TIME-SHARING SYSTEMS ARE VERY TOLERANT OF THIS, ALTHOUGH PARTICULAR APPLICATION PROGRAMS MAY NOT BE. THE USER IS ADVISED TO CONSULT THE INSTALLATION ADMINISTRATOR FOR GUIDANCE.

THE DEFAULT VALUE FOR THIS PARAMETER IS 'TCF$QUIT'. THE USER IS CAUTIONED THAT IN THIS SPECIFIC INSTANCE, UPPER AND LOWER CASE CHARACTERS ARE TREATED DISTINCTLY.

NOTE THAT THE KEYWORDS HOST, TERMINAL AND QUIT CAN BE SPECIFIED BY A MINIMUM OF ONE LETTER EACH.

EXAMPLES:

```
ICF -HOST IBM-370-HOST.1 -TERMINAL THIS-ONE
    -QUIT 'QUIT-NOW'
```

THIS COMMAND REQUESTS THAT THE USER AT THE TERMINAL NAMED 'THIS-ONE' BE CONNECTED TO THE HOST PORT NAMED 'IBM-370-HOST.1' IN BLOCK MODE, AND THAT WHENEVER THE USER ENTERS THE STRING 'QUIT-NOW' VERBATIM IN A CONTINUOUS STRING, TCF IS TO BREAK THAT CONNECTION AND RETURN THE USER TO PRIMOS COMMAND LEVEL.

WHEN THE COMMAND LINE IS ENTERED, TCF RESPONDS BY OUTPUTTING THE STRING:

```
'DPTX/ICF VERSION 1.0'
```

SHOULD THE USER INCORRECTLY ENTER THE COMMAND LINE, OR SHOULD THE NAMES OF THE TERMINAL AND/OR THE HOST AS GIVEN BY THE USER NOT MATCH THE INTERNAL TABLES OF AVAILABLE DEVICES MAINTAINED BY THE SYSTEM, THE USER IS INFORMED OF THE NATURE OF THE ERROR AND TCF RETURNS TO PRIMOS COMMAND LEVEL.

4.3.3 WARNING AND ERROR MESSAGES

ERROR MESSAGES PRODUCED BY DPTX/TCF INDICATE EITHER OF TWO CONDITIONS:

- AN INCORRECT INVOCATION OF THE PROGRAM.
- A FAILURE TO ESTABLISH OR RETAIN THE IBM HOST AND/OR THE 3270 TERMINAL BLOCK-MODE VIRTUAL LINKS (AS OPPOSED TO THE SERIAL-CHARACTER MODE LINK BETWEEN PRIMOS AND DPTX/TSF).

IN ANY OF THESE CASES, TCF RETURNS TO PRIMOS COMMAND LEVEL AFTER OUTPUTTING AN APPROPIATE ERROR MESSAGE.

ALL ERROR MESSAGES OUTPUT BY TCF (ACCOMPANIED BY APPROPIATE EXPLANATIONS) MAY BE FOUND IN APPENDIX A OF THE TCF DOCUMENT ON THE MASTER DISK.

TCF OUTPUTS A WARNING MESSAGE IN ANY OF THREE INSTANCES:

- A 'DATA COLLISION' WHERE THE USER HAS ATTEMPTED TO SEND SOMETHING FROM THE TERMINAL TO THE IBM HOST AT THE SAME TIME AS THE IBM HOST IS SENDING SOMETHING TO THE USER. IN THIS CASE, THE USER'S INPUT IS DISCARDED, AND THE HOST-UPDATED SCREEN IMAGE IS DISPLAYED AT THE USER TERMINAL FOLLOWING A WARNING MESSAGE. THE USER SHOULD WAIT UNTIL SUCH TIME AS THESE ACTIONS ARE COMPLETE BEFORE ATTEMPTING ANY REMEDIAL ACTION.

- A 'BROKEN SCREEN', WHERE A PRIME SYSTEM BULLETIN IS SENT TO THE USER'S TERMINAL. WHEN THIS OCCURS, A WARNING MESSAGE TELLING THE USER THAT THE TERMINAL CONNECTION WAS TEMPORARILY BROKEN IS DISPLAYED, FOLLOWED BY THE MOST RECENT SUCCESSFULLY TRANSMITTED SCREEN IMAGE FROM EITHER THE HOST OR THE TERMINAL.

- A TEMPORARY INTERRUPTION IN THE BLOCK-MODE CONNECTION(S) TO THE TERMINAL AND/OR HOST. THE USER RECEIVES A MESSGE TO THIS EFFECT, FOLLOWED BY A NEW SCREEN IMAGE.

4.4 DPTX/ISE OPERATION

WHAT THE SYSTEM LOOKS LIKE AT THE 3277 TERMINAL.

TELETYPE EMULATION MODE IS THE DEFAULT MODE FOR A 3270, AND IS IN EFFECT WHEN THE OPERATOR FIRST LOGS IN.

PRIME STANDARD VIDEO TERMINAL FEATURES WHICH AREN'T AVAILABLE ON A 3277, AND WHICH THEREFORE HAVE TO BE EMULATED, INCLUDE:

-- CHARACTER-AT-A-TIME TRANSMISSION. A PRIME STANDARD TERMINAL TRANSMITS CHARACTERS TO THE MAINFRAME AS SOON AS THEY ARE TYPED BY THE OPERATOR, AND DISPLAYS EACH CHARACTER AS IT IS RECEIVED FROM THE MAINFRAME. THE 3277 TRANSMITS A COMPLETE MESSAGE WHEN THE OPERATOR HITS THE ENTER KEY, AND DISPLAYS A COMPLETE MESSAGE AFTER THE WHOLE THING HAS BEEN RECEIVED FROM THE MAINFRAME.

-- FULL-DUPLEX OPERATION. A PRIME STANDARD TERMINAL CAN TRANSMIT AND RECEIVE CHARACTERS AT THE SAME TIME. THE 3277 CANNOT; IN PARTICULAR, THE OPERATOR'S KEYBOARD IS LOCKED (TYPING IS DISALLOWED) WHENEVER A MESSAGE IS BEING SENT TO OR RECEIVED FROM THE MAINFRAME, AND MAY IN FACT BE KEPT LOCKED FOR EXTENDED PERIODS AT THE DISCRETION OF THE HOST.

-- SCROLLING. A PRIME STANDARD TERMINAL AUTOMATICALLY MOVES OLD LINES UPWARDS AS NEEDED TO MAKE ROOM FOR NEW LINES AT THE BOTTOM. A 3277 WORKS WITH VARIABLE-LENGTH "FIELDS" AND HAS NO BUILT-IN CONCEPT OF A "LINE"; ALL SCREEN FORMATTING, INCLUDING SCROLLING OR AN ILLUSION THEREOF, MUST BE EXPLICITLY HANDLED BY THE MAINFRAME.

-- FULL ASCII CHARACTER SET. THIS RELEASE OF DPTX SUPPORTS EBCDIC TRANSMISSION CODE ONLY. THE RESULT IS THAT SOME ASCII GRAPHIC CHARACTERS AND CONTROL CHARACTERS ARE MISSING FROM THE KEYBOARD AND DISPLAY.

WE HAVE CHOSEN TO FORMAT THE 3277 SCREEN INITIALLY INTO 24 FIELDS, EACH COMPRISING 79 CHARACTERS. THESE FIELDS WILL BE TREATED AS "LINES". THE LAST CHARACTER POSITION (COLUMN 80) ON EACH LINE OF THE 3277 SCREEN IS RESERVED.

THE TOP "LINE" ON THE 3277 SCREEN IS RESERVED BY THE SUPPORT

PACKAGE FOR ITS OWN USE. THE RESERVED FIRST LINE WILL BE SET UP AS A PROTECTED FIELD. THE REST OF THE SCREEN (23 LINES) WILL BE LEFT UNPROTECTED; THIS MEANS THAT ALL CHARACTER POSITIONS ON THESE LINES, EXCEPT THOSE OCCUPIED BY ATTRIBUTE BYTES, CAN POTENTIALLY BE USED TO ENTER DATA. THE CURSOR WILL INITIALLY BE POSITIONED AT THE BEGINNING OF THE FIRST TYPEABLE LINE (LINE 2).

SINCE THE 3277 HAS NO SCROLLING CAPABILITY OF ITS OWN, WE HAVE DECIDED TO IMPLEMENT A "MODIFIED SCROLL" OR WRAPAROUND. THE FIRST NON-RESERVED LINE (LINE 2) ON THE SCREEN IS CONSIDERED TO BE THE LINE FOLLOWING THE LAST LINE ON THE SCREEN, AND THE SUPPORT PACKAGE WILL CAUSE OUTPUT LINES AUTOMATICALLY TO WRAP AROUND TO THE "TOP" WHEN THE SCREEN IS FULL. TO PROVIDE A "CLEAN" AREA FOR THE OPERATOR TO TYPE INPUT LINES INTO, THE SUPPORT PACKAGE WILL ALWAYS ENSURE THAT THERE ARE THREE BLANK LINES AT THE CURSOR POSITION WHEN THE 3277 IS READY TO ACCEPT OPERATOR INPUT. IF THE OPERATOR WISHES TO INPUT MORE THAN THREE LINES ON ONE TRANSMISSION, HE WILL HAVE TO CLEAR THE EXTRA LINES MANUALLY (WITH THE ERASE EOF KEY) BEFORE USING THEM.

SENDING AND RECEIVING MESSAGES WITH THE 3277

THE OPRATOR WILL SEND A MESSAGE TO THE PRIME SYSTEM BY TYPING IN ONE OR MORE INPUT LINES ON THE 3277 AND HITTING THE ENTER KEY. DATA WILL NOT BE PASSED TO THE PRIME SYSTEM UNTIL ENTER IS TYPED, AND AT THAT TIME ALL INPUT JUST TYPED IN WILL BE SENT.

NORMALLY, IF THE OPERATOR WISHES TO SEND A SINGLE LINE, HE WILL TYPE A STRING OF TEXT CHARACTERS AND THEN TYPE ENTER. IF HE WANTS TO SEND SEVERAL LINES, EACH LINE IS ENDED WITH THE NEW LINE KEY, AND THE LAST LINE MAY BE TERMINATED WITH A NEW LINE AND/OR ENTER KEY. THERE IS A DEFINITE AND IMPORTANT DIFFERENCE BETWEEN THE ACTIONS CAUSED BY THE NEW LNE AND ENTER KEYS. WHILE BOTH KEYS WILL RESULT IN AN ASCII NEW LINE CHARACTER BEING NSERTED IN THE MESSAGE, ONLY THE ENTER KEY WILL ACTUALLY CAUSE THE TEXT WHICH HAS BEEN TYPED IN TO BE SENT TO THE PRIME SYSTEM.

SOME EXAMPLES MAY HELP TO DEMONSTRATE THE USE OF THE NEW LINE AND ENTER KEYS. (THE CURSOR (_) IS SHOWN AFTER TYPING ENTER AND HAVING THE COMMAND PROCESSED BY PRIMOS.)

```
OK, DATE  <ENTER>                    "DATE" IS SENT

WEDNESDAY, MARCH 14, 1979   10:00 AM

_

OK,<ENTER>                           ONE NULL LINE SENT
_

OK, <NEW LINE>                       TWO NULL LINES SENT
<ENTER>
_
```

```
OK, DATE  <NEW LINE>                    "DATE" PLUS ONE NULL LINE S
ENT
<ENTER>

WEDNESDAY, MARCH 14, 1979   10:00 AM
```

ERASE AND KILL CHARACTERS ARE HONORED AS FOR A PRIME STANDARD TERMINAL; HOWEVER, THEY ARE NOT GENERALLY NECESSARY BECAUSE THE OPERATOR CAN USE THE CURSOR POSITIONING KEYS TO MAKE ANY CHANGES TO HIS INPUT LINES BEFORE HITTING ENTER. IF THE OPERATOR INTENDS ON USING THE ERASE AND KILL CHARACTERS, IT SHOULD BE NOTED THAT THOSE CHARACTERS MUST NOT BE ONES WHICH CANNOT BE GENERATED FROM THE 3277 KEYBOARD, E.G. CONTROL CHARACTERS.

THE SUPPORT PACKAGE WILL ANALYZE THE OPERATOR'S INPUT INTO LINES AS APPROPRIATE, AND PASS THESE ON ONE AT A TIME TO THE USER PROGRAM OR TO PRIMOS COMMAND LEVEL.

THE OPERATOR CAN ENTER AN EMPTY, OR "NULL", LINE (SUCH AS USED BY THE EDITOR TO SWAP BETWEEN EDIT AND INPUT MODES) BY SIMPLY PRESSING ENTER, WITHOUT HAVING TYPED IN ANY DATA CHARACTERS. SEVERAL EMPTY LINES IN SUCCESSION MAY BE ENTERED BY MOVING THE CURSOR DOWN SEVERAL LINES WITH THE NEW LINE KEY, THEN PRESSING ENTER.

OUTPUT FROM THE PRIME SYSTEM TO THE TERMINAL WILL BEGIN ON THE LINE FOLLOWING THE OPERATOR'S LAST INPUT LINE, AND MAY CONSIST OF ONE OR SEVERAL LINES. AT THE CONCLUSION OF OUTPUT FROM THE SYSTEM, THE CURSOR IS LEFT POSITIONED TO THE FIRST CHARACTER POSITION AVAILABLE FOR TYPING INPUT.

TYPING AHEAD ON A 3277.

THE HALF-DUPLEX 3277 CAN'T HANDLE SIMULTANEOUS INPUT AND OUTPUT.

THE SITUATION INVOLVING TYPE-AHEAD CAN BE QUITE CONFUSING TO AN OPERATOR ACCUSTOMED TO STANDARD FULL DUPLEX TERMINALS.

WE ANTICIPATE THAT OPERATORS WON'T USE TYPE-AHEAD VERY OFTEN, SINCE IT IS POSSIBLE TO ENTER SEVERAL LINES OF COMMANDS OR DATA ON THE 3277 AND TRANSMIT THEM ALL AT ONCE.

USE OF THE FUNCTION KEYS ON A 3277.

THE OPERATOR CAN CAUSE A QUIT (BREAK TO PRIMOS COMMAND LEVEL) BY PRESSING THE CANCEL (ALSO CALLED PA2) KEY AT THE 3277 TERMINAL. THIS HAS THE SAME EFFECT AS A BREAK OR CONTROL-P ON A PRIME STANDARD TERMINAL. THE SUPPORT PACKAGE WILL ENSURE THAT THE KEYBOARD IS NEVER LEFT LOCKED FOR MORE THAN A FEW SECONDS, SO THAT THE QUIT FUNCTION IS (ALMOST) ALWAYS AVAILABLE TO THE OPERATOR. (IF A RUNNING PROGRAM HAS SET "BREAKS INHIBITED", THE

QUIT WILL OF COURSE BE IGNORED.)

THE CLEAR KEY CLEARS THE ENTIRE SCREEN OF ALL DATA AND FORMATTING CHARACTERS. WHENEVER THE OPERATOR HITS THE CLEAR KEY, THE SUPPORT PACKAGE RE-INITIALIZES THE SCREEN TO THE BASIC FORMAT (24 LINES, 79 USEABLE CHARACTERS EACH, FIRST LINE PROTECTED). NO DATA IS TRANSFERRED TO THE USER PROGRAM OR TO PRIMOS COMMAND LEVEL.

IF THE OPERATOR PRESSES A PROGRAM FUNCTION (PF) KEY OTHER THAN ENTER, THE RESULTING DATA FROM THE TERMINAL WILL BE TREATED BY THE SUPPORT PACKAGE JUST AS IF THE OPERATOR HAD PRESSED ENTER. THE PROGRAM ACCESS KEYS OTHER THAN CANCEL AND CLEAR, AND THE TEST REQ KEY, WILL BE IGNORED COMPLETELY.
USERS ARE TO BE WARNED OF USE OF THE ERASE INPUT KEY. THIS KEY WILL ERASE THE SCREEN, AND REPOSITION THE CURSOR AT THE UPPER LEFT HAND CORNER, IMMEDIATELY BELOW THE TOP RESERVED LINE. THE PRIME SYSTEM CANNOT BE INFORMED OF THIS CHANGE IN CURSOR POSITION, DUE TO 3271/3277 HARDWARE DESIGN. THEREFORE, BE ADVISED THAT USE OF THE ERASE INPUT KEY MAY PRODUCE UNDESIRABLE RESULTS. IT IS SUGGESTED THAT THE CLEAR KEY BE USED IN CASES WHERE ERASE INPUT WOULD BE USED. THIS RESTRICTION MAY NOT HOLD WHEN THE 3277 IS BEING USED IN BLOCK MODE.

ALL CURRENT SOFTWARE WILL RUN WITHOUT CHANGE. THIS INCLUDES ALL USES OF THE FOLLOWING:

-- RAW DATA MOVERS: T11N (AND C11N), T10U, TNOU, TNOUA, TIDEC (AND SO FORTH);

-- THE I$ AND O$ IOCS SUBROUTINES FOR USER TERMINAL INPUT AND OUTPUT (WHICH CALL ON THE RAW DATA MOVERS);

-- "SKS 704" TO CHECK FOR THE PRESENCE OF INPUT CHARACTERS BEFORE USING THE RAW DATA MOVERS TO FETCH THEM.

BLOCK MODE.

THIS IS "NATIVE MODE" FOR THE 3277. THE SUPPORT PACKAGE PUTS THE TERMINAL INTO THIS MODE IN RESPONSE TO A REQUEST FROM A USER PROGRAM. ONCE THE PROGRAM MAKES THIS CALL, ALL FORMATTING OF THE SCREEN WILL BE EXPLICITLY UNDER CONTROL OF THE USER PROGRAM; PRIMOS JUST PASSES INFORMATION BACK AND FORTH. THE OPERATOR WILL CONTINUE TO USE THE ENTER KEY TO TRANSMIT A SCREEN OF INPUT DATA. THE PROGRAM FUNCTION AND PROGRAM ACCESS KEYS (OTHER THAN CANCEL, WHICH STILL MEANS QUIT) HAVE WHATEVER MEANING ARE ASSIGNED TO THEM BY THE USER PROGRAM. THE CLEAR KEY CLEARS THE SCREEN OF ALL DATA AND FORMATTING INFORMATION, BUT IT IS UP TO THE USER PROGRAM TO REBUILD THE SCREEN.

IF A BULLETIN OR SIMILAR PRIMOS-GENERATED MESSAGE IS SENT TO A 3277 IN BLOCK MODE, THE MESSAGE WILL BE DISPLAYED SOMEWHERE ON THE SCREEN, MIXED IN WITH THE FORMAT WHICH IS CURRENTLY BEING DISPLAYED. THE BLOCK MODE PROGRAM WILL CONTINUE TO RUN, HOWEVER.

THE TERMINAL IS RETURNED TO TELETYPE-EMULATION MODE WHEN THE OPERATOR HITS THE CANCEL KEY, CAUSING A QUIT, OR WHEN THE USER PROGRAM MAKES A CALL TO RETURN THE 3277 TERMINAL TO TELETYPE-EMULATION MODE. IN EITHER CASE, THE SCREEN WILL BE REFORMATTED INTO THE PRIMOS FORMAT OF 24 LINES, EACH 79 CHARACTERS.

4.5 DPTX/DSC OPERATION

THIS SECTION DESCRIBES THE USER PROGRAM AND THE EMULATION PROGRAM.

4.5.1 OWLDSC

OWLDSC IS A USER PROGRAM DESIGNED TO INTERFACE WITH THE DPTX/DSC PACKAGE. IT PROVIDES EMULATION OF AN IBM 3277 MODEL 2 DISPLAY STATION BY A PERKIN ELMER OWL-1200 TERMINAL. OWLDSC COMMUNICATES WITH THE VIRTUAL BUFFER EMULATOR VIA CALLS TO THE BLOCK DEVICE INTERFACE FOR DPTX.

OPERATION.

THE OWL INTERFACE PROGRAM IS INVOKED BY TYPING THE COMMAND "OWLDSC" FOLLOWED BY AN OPTIONAL ARGUMENT. THE COMMAND FORMAT IS

OWLDSC [-FAST]

THE "-FAST" OPTION IS DISCUSSED UNDER, "OWL HARDWARE DEFICIENCIES."

OWLDSC RESPONDS BY REQUESTING THE USER TO ENTER THE NAME OF A VALID PORT TO A MAINFRAME:

STATION NAME:

THIS PORT NAME IS ONE WHICH WAS DESCRIBED WHEN CONFIGURATION OF DPTX WAS PERFORMED. (SEE DPTX CONFIGURATION FOR FURTHER DETAILS.)

FUNCTION KEY MAPPING.

THE OWL-1200 TERMINAL PROVIDES EQUIVALENT FUNCTION KEYS FOR ALL ATTENTION KEYS ON A 3277 TERMINAL. (THE TERMS "ATTENTION ID" OR "AID", AND "FUNCTION KEY" ARE INTERCHANGEABLE.) THE MAPPING IS AS FOLLOWS:

IBM ATTENTION ID (AID)	OWL FUNCTION KEY EQUIVALENT	
	SHIFT	UNSHIFT
PF 1	F1	F1
PF 2	F2	F2
PF 3	F3	F3
PF 4	F4	F4
PF 5	F5	F5
PF 6	F6	F6
PF 7	F7	F7
PF 8	F8	F8
PF 9	F9	F9
PF 10	F10	F10
PF 11	F11	F11
PF 12	F12	F12
TST REQ	F13	F13
PA 1	F14	---
PA 2	F15	---
PA 3	F16	---
CLEAR	---	F14
ENTER	---	F16

TWO ADDITIONAL FUNCTION KEYS ARE PROVIDED ON THE OWL-1200.

-- RECOVER (UNSHIFT F15):

THIS KEY IS USED TO DISPLAY THE CURRENT COPY OF THE VIRTUAL BUFFER FOR THE PARTICULAR USER. A GOOD USE FOR THIS MIGHT BE IF THE USER MADE SOME CHANGES ON THE SCREEN AND WANTED TO

RESTORE THE ORIGINAL COPY OR IF THE USER ACCIDENTLY HIT THE LOCAL CLEAR ALL KEY. (NOTE THE DIFFERENCE BETWEEN CLEAR ALL AND THE IBM-EQUIVALENT CLEAR, UNSHIFT F14.)

-- SEND PAGE, SEND LINE, SEND MSG:

ANY OF THESE FUNCTION KEYS WILL CAUSE A "GRACEFUL" EXIT TO PRIMOS. USE OF CONTROL-P OR BREAK IS NOT RECOMMENDED SINCE OWLDSC IS EXECUTING IN A HALF-DUPLEX, BLOCK MODE ENVIRONMENT. THE SEND KEYS WILL RESTORE THE TERMINAL TO FULL-DUPLEX, CONVERSATIONAL MODE AND WILL ALSO DETACH THE TERMINAL FROM ITS EMULATION PORT.

THE OWL-1200 READ MODIFIED COMMAND HAS THREE DEFICIENCIES:

1. IT CANNOT READ CHARACTERS BEFORE THE FIRST ATTRIBUTE.

2. IT CANNOT READ CHARACTERS ON AN UNFORMATTED SCREEN.

3. IT CANNOT READ THE 1920'TH CHARACTER POSITION.

TO OVERCOME THESE PROBLEMS, THE DEFAULT MODE OF OWLDSC WILL ISSUE A 'READ ALL' COMMAND IN RESPONSE TO A USER ATTENTION, REGARDLESS OF WHETHER THE SCREEN IS FORMATTED OR NOT. THE VISUAL SCREEN AND THE VIRTUAL BUFFER WILL BE IDENTICAL IN THIS CASE.

A QUICKER VERSION OF OWLDSC CAN BE RUN BY APPENDING '-FAST' TO THE COMMAND LINE. THIS VERSION WILL ISSUE A 'READ MODIFIED' OPERATION IN RESPONSE TO A USER ATTENTION ON A FORMATTED SCREEN. NULLS ARE SUPPRESSED ON AN OWL READ MODIFIED COMMAND, THEREFORE, FIELDS CONTAINING LEADING NULLS FOLLOWED BY DATA CHARACTERS WILL APPEAR LEFT JUSTIFIED IN THE VIRTUAL BUFFER. ALSO, THE USER MUST BE AWARE OF APPLICATIONS WHERE CHARACTERS ARE INPUT ON THE SCREEN BEFORE AN ATTRIBUTE OR IN THE 1920'TH CHARACTER POSITION. FOR UNFORMATTED SCREENS, A 'READ ALL' WILL BE ISSUED TO SOLVE DEFICIENCY #2.

ERROR RECOVERY.

OWLDSC WILL ATTEMPT A READ OPERATION 'MAXRET' TIMES (CURRENTLY SET TO 3). AMONG THE REASONS FOR A READ RETRY ARE: CONTROL CHARACTER DETECTED IN READ DATA STREAM; EARLY ETX DETECTED; NO SOH CHARACTER TO BEGIN READ MOD OPERATION; ILLEGAL CURSOR ADDRESS FOUND; OR AN ILLEGAL CBA ADDRESS CORRESPONDING TO THE START OF A MODIFIED FIELD WAS FOUND IN A READ MOD DATA STREAM. IF EITHER THE READ OR READ MOD OPERATION IS UNSUCCESSFUL AFTER 'MAXRET' TIMES, OWLDSC WILL GO INTO RECOVER MODE AND DISPLAY THE CURRENT COPY OF THE USER'S VIRTUAL BUFFER ON THE OWL SCREEN (NOTE: THIS ACTION HAS THE SAME EFFECT AS THE USER TYPING THE RECOVER KEY - F15).

4.5.2 THE VBE

THE DPTX/DSC FUNCTION IS MANAGED BY A PROCESS CALLED THE PRIME VIRTUAL BUFFER EMULATOR (VBE). UP TO FOUR COPIES OF THE VBE MAY BE RUNNING AT ONCE, ONE FOR EACH SYNCHRONOUS LINE WHICH IS OPERATING DSC. THESE PROCESSES NORMALLY RUN AS PHANTOMS. THIS SECTION TELLS HOW TO CONFIGURE AND START UP A VBE.

THE VBE HAS THREE MAJOR FUNTIONS.

- IT KEEPS THE COMMUNICATIONS TO THE HOST GOING. THE VBE SPEAKS THE 3271 CONTROL UNIT DIALECT OF THE BINARY SYNCHRONOUS PROTOCOL; IT COMMUNICATES THROUGH THE BSCMAN PROCESS. SINCE THE HOST IS THE MASTER IN A MASTER-SLAVE RELATIONSHIP WITH THE VBE, THE VBE MUST ALWAYS BE RESPONSIVE TO THE HOST AND CANNOT SPEAK UNLESS SPOKEN TO.

- IT MAINTAINS THE "SCREEN IMAGE" AND CURRENT STATUS OF ALL VIRTUAL TERMINALS ON ITS VIRTUAL CONTROL UNITS. AT ALL TIMES THE SCREEN IMAGE STORED IN PRIMOS MEMORY IS KEPT CONSISTENT WITH WHAT THE HOST EXPECTS THE SCREEN TO LOOK LIKE. UPDATES TO THE SCREEN AND STATUS COME FROM BOTH THE HOST SIDE AND THE APPLICATION PROGRAM SIDE.

- IT TALKS TO EACH DPTX APPLICATION PROGRAM (THE "VIRTUAL OPERATOR" OF EACH VIRTUAL TERMINAL) TO SHOW IT NEW DATA SENT BY THE HOST AND TO READ THE SCREEN UPDATES "TYPED IN" BY THE APPLICATION PROGRAM. THE PROGRAMS RUNNING UNDER DSC WILL NORMALLY BE THE OWL EMULATOR PROGRAM OWLDSC OR THE 3270 PASS-THROUGH PROGRAM TCF, BUT COULD ALSO BE CUSTOMER-WRITTEN APPLICATIONS.

THE VBE ACCEPTS MESSAGES FROM THE HOST AS WOULD A 3271 CONTROL UNIT AND FORWARDS THEM TO THE APPROPRIATE APPLICATIONS PROGRAMS. IT ALSO TAKES MESSAGES GIVEN IT BY APPLICATION PROGRAMS AND SENDS THEM TO THE HOST WHEN THE HOST NEXT POLLS THE APPROPRIATE VIRTUAL TERMINALS. A VIRTUAL TERMINAL WHICH IS NOT CURRENTLY CLAIMED BY SOME APPLICATION PROGRAM IS CONSIDERED "POWERED OFF" BY THE VBE, AND CAN'T BE WRITTEN TO OR TYPED ONTO BY EITHER END.

HOST-VISIBLE DIFFERENCES

WE'VE ATTEMPTED TO MAKE THE VBE LOOK AND BEHAVE AS MUCH AS POSSIBLE LIKE A SET OF 3271 CONTROL UNITS AND 3277 TERMINALS. HOWEVER, A FEW DIFFERENCES WERE UNAVOIDABLE BECAUSE OF THE NATURE OF OUR IMPLEMENTATION.

"DEVICE BUSY" STATUS

A SITUATION CAN ARISE IN WHICH THE HOST WISHES TO SEND A COMMAND TO A VIRTUAL TERMINAL BUT THE VBE DOESN'T WANT TO ACCEPT IT BECAUSE IT CANNOT GET THE COMMAND TO THE APPLICATION PROGRAM RESPONSIBLE FOR THAT VIRTUAL TERMINAL. THIS PROBLEM CAN ARISE BECAUSE

- THE QUEUE WHICH THE VBE USES TO PASS MESSAGES TO THE APPLICATION PROGRAM IS FULL, BECAUSE THE PROGRAM ISN'T CONSUMING MESSAGES AS FAST AS THE HOST IS WRITING THEM. THIS CAN HAPPEN IF THE HOST IS DOING A LOT OF RAPID CHAINING OF COMMANDS, OR IF THE APPLICATION PROGRAM TAKES A LONG TIME TO PROCESS MESSAGES. "QUEUE FULL" IS USUALLY A TRANSIENT CONDITION WHICH CLEARS UP QUICKLY, UNLESS THE APPLICATION PROGRAM IS MALFUNCTIONING.

- THE VBE NEEDS SOME TEMPORARY BUFFERS TO STORE A MESSAGE TO BE SENT TO THE APPLICATION PROGRAM, BUT CAN'T GET ENOUGH BECAUSE THE SYSTEM'S POOL OF FREE BUFFERS IS EMPTY. BUFFERS ARE DYNAMICALLY SHARED AMONG ALL DPTX PROCESSES, AND AT TIMES OF PEAK LOAD THERE MAY NOT BE ENOUGH FOR INDIVIDUAL LARGE MESSAGES. THIS IS ALSO A TRANSIENT SITUATION WHICH WILL CLEAR UP QUICKLY AS THE LOAD FLUCTUATES.

IN EITHER OF THESE CASES, VBE WILL PRETEND THAT THE VIRTUAL TERMINAL BEING ADDRESSED IS "BUSY", AND WILL SET EITHER THE "DB" STATUS BIT (DEVICE BUSY ON SELECT) OR THE "DB" AND "US" STATUS BITS TOGETHER (COMMAND SENT TO A BUSY DEVICE).

ERROR RECOVERY

A REAL 3271 CONTROL UNIT IS STRICTLY A SLAVE TO THE HOST AND WILL WAIT PATIENTLY IN WHATEVER STATE IT'S IN WHEN THE HOST GOES DOWN. THE VBE, HOWEVER, TAKES SOME LIMITED RECOVERY ACTIONS ON ITS OWN WHEN IT DETERMINES THAT THE HOST IS DOWN; IT WILL RETURN THE CURRENT VIRTUAL CONTROL UNIT TO AN IDLE STATE AND PRINT OUT SOME DIAGNOSTIC INFORMATION.

DIAGNOSTIC AND ERROR MESSAGES

THE VBE PRINTS ON ITS USER TERMINAL A LOG OF UNUSUAL OCCURRENCES OR ERROR CONDITIONS. ALTHOUGH THE VBE NORMALLY RUNS AS A PHANTOM, THE LOG CAN BE ROUTED TO A COMMAND OUTPUT FILE OR THE VBE CAN BE RUN AT A TERMINAL FOR DIAGNOSTIC PURPOSES.

THIS SECTION LISTS ALL MESSAGES THAT MIGHT BE PRINTED BY THE VBE DURING EXECUTION. SOME OF THESE REFLECT COMMUNICATIONS ERRORS, SOME RECORD NORMAL BUT NOTEWORTHY OCCURRENCES, AND SOME PROVIDE DIAGNOSTIC INFORMATION FOR DEBUGGING THE VBE. (THE LATTER SHOULD NEVER BE SEEN IN NORMAL OPERATION.) THE VBE CONTINUES TO RUN AFTER ENCOUNTERING ALL ERRORS EXCEPT STARTUP ERRORS (E.G., BAD SMLC LINE NUMBER).

THE LIST SHOWS, IN THE LEFT MARGIN, THE NAME OF THE MODULE WITHIN THE VBE WHICH GENERATES THE MESSAGE.

EM3270 DPTX/DSC, VERSION 2.32, 04-FEB-79
THIS IS THE NORMAL STARTUP MESSAGE.

EM3270 DPTX NOT CONFIGURED

THE DPTX OPTION IS CURRENTLY OFF. THE VBE RETURNS TO PRIMOS COMMAND LEVEL.

EM3270 ILLEGAL SMLC LINE NUMBER: <N>.
THE LOGICAL SMLC LINE NUMBER <N> GIVEN TO #EMGO IS NOT WITHIN THE RANGE OF 0 THROUGH 3. THE VBE RETURNS TO PRIMOS COMMAND LEVEL.

EM3270 DPTX/DSC FOR LINE <N> ALREADY OWNED BY USER #<N>.
THERE IS ALREADY A COPY OF THE VBE RUNNING ON THE SMLC LINE #EMGO WAS TOLD TO USE. THE VBE RETURNS TO PRIMOS COMMAND LEVEL.

EM3270 NAK IN CONTROL MODE
THE HOST SENT A NAK CHARACTER WHILE THE VBE WAS IDLE. THIS IS A NORMAL OCCURRENCE, BUT WHICH MAY BE EVIDENCE OF LINE PROBLEMS OR RESPONSE TIME PROBLEMS ON THE DPTX SYSTEM.

IN THE FOLLOWING MESSAGES, <MODE> IS ONE OF (CTL, RSP, TXT, SEL) DEPENDING ON WHETHER THE CURRENT VIRTUAL CONTROL UNIT IS IDLE, SENDING MESSAGES TO THE HOST, RECEIVING COMMANDS FROM THE HOST, OR RESPONDING TO A SELECTION.

EM3270 <MODE>: DATA SET READY
DATA SET STATUS ON THE VBE'S LINE HAS CHANGED FROM "NOT READY" TO "READY". THIS HAPPENS WHEN THE LINE IS FIRST CONNECTED, OR WHEN A CONNECTION IS RE-ESTABLISHED.

EM3270 <MODE>: LOST DATA SET READY
DATA SET STATUS HAS CHANGED FROM "READY" TO "NOT READY". THIS INDICATES THAT THE LINE HAS BEEN DISCONNECTED.

EM3270 <MODE>: UNEXPECTED INPUT, COD1 :<N>, COD2 :<M>, IGNORE.
THE VBE RECEIVED A COMMUNICATION LINE INPUT (FROM THE COMMUNICATIONS PROCESS BSCMAN) WHICH IT WASN'T EXPECTING OR CAN'T HANDLE RIGHT NOW. THE INPUT IS IGNORED. A LIST OF THE POSSIBLE VALUES OF COD1 AND COD2, AND THEIR MEANINGS, IS GIVEN AT THE END OF THIS SECTION.

EM3270 <MODE>: HOST IS BROKEN, COD1 :<N>, COD2 :<M>, RETURN TO CONTROL MODE.
THE VBE RECEIVED A COMMUNICATION LINE INPUT (FROM THE COMMUNICATIONS PROCESS BSCMAN) WHICH INDICATES THAT THE HOST HAS FAILED (EITHER CRASHED OR FAILING TO FOLLOW BISYNC PROTOCOL CONVENTIONS). THE INPUT IS IGNORED, AND THE VBE RETURNS THE CURRENTLY-ACTIVE VIRTUAL CONTROL UNIT (IF ANY) TO AN IDLE STATE. A LIST OF THE POSSIBLE VALUES OF COD1 AND COD2, AND THEIR MEANINGS, IS GIVEN AT THE END OF THIS SECTION.

EM3270 <MODE>: CAN'T GET DB'S FOR VB COPY
THE VBE HAS REJECTED A COMMAND OR REFUSED A SELECTION BY THE HOST BECAUSE OF A (TRANSIENT) SHORTAGE OF FREE BUFFERS NEEDED TO FORWARD A MESSAGE TO THE APPLICATION PROGRAM. THIS IS A NORMAL OCCURRENCE; HOWEVER, IF IT PERSISTS IT MAY INDICATE A MALFUNCTION SOMEWHERE IN THE DPTX SYSTEM.

EM3270 BAD COMMAND, SEND EOT (4900).
THE VBE HAS REJECTED AN ILL-FORMATTED COMMAND FROM THE HOST.

EM3270 BAD TEXT, RTYPE IS :<N>, SEND NAK.
THE VBE HAS REJECTED AN ILL-FORMATTED COMMAND FROM THE HOST.

EM3270 UPDATE OK, DEVICE #<N>.
THE VBE HAS RECEIVED A VIRTUAL TERMINAL UPDATE MESSAGE FROM THE APPLICATION PROGRAM USING DEVICE #<N>. (<N> REFLECTS THE DPTCFG NUMBER OF A VIRTUAL TERMINAL.) THIS IS A NORMAL OCCURRENCE.

EM3270 UPDATE REJECTED, DEVICE #<N>.
THE VBE HAS DISCARDED A VIRTUAL TERMINAL UPDATE MESSAGE FROM AN APPLICATION PROGRAM; THE MESSAGE IS OUT OF DATE BECAUSE THE HOST HAS CHANGED THE STATE OF THE VIRTUAL TERMINAL AFTER THE MESSAGE WAS CONSTRUCTED. THE PROGRAM IS INFORMED THAT ITS MESSAGE WAS THROWN AWAY. THIS IS A NORMAL OCCURRENCE.

EM3270 DEVICE BUSY, SEND WACK.
THE VBE HAS SET THE "DEVICE BUSY" BIT FOR A VIRTUAL TERMINAL, FOR ONE OF THE REASONS LISTED IN THE PRECEDING SECTION, AND IS REFUSING A SELECTION OF THAT VIRTUAL TERMINAL. THIS IS A NORMAL OCCURRENCE; HOWEVER, IF IT PERSISTS IT MAY INDICATE A PROBLEM EITHER WITH THE APPLICATION PROGRAM USING THE DEVICE OR SOMEWHERE WITHIN THE DPTX SYSTEM.

EM3270 MISC. SENSE/STATUS, SEND RVI.
THE VBE HAS OTHER STATUS BITS SET THAN "DEVICE BUSY" ALONE FOR SOME VIRTUAL TERMINAL, AND IS REFUSING A SELECTION. THIS MAY INDICATE THE HOST TRYING TO SELECT A "POWERED-OFF" DEVICE, OR SOME OTHER STATUS CONDITION. THIS IS A NORMAL OCCURRENCE; HOWEVER, IF IT PERSISTS IT MAY INDICATE A PROBLEM EITHER WITH THE APPLICATION PROGRAM USING THE DEVICE OR SOMEWHERE WITHIN THE DPTX SYSTEM.

EM3270 SEND S/S MESSAGE.
THE VBE HAS SENT A STATUS MESSAGE TO THE HOST ON BEHALF OF THE VIRTUAL TERMINAL WHICH HAS JUST BEEN POLLED. THIS IS A NORMAL OCCURRENCE.

EM3270 CALL RDMODR "TST REQ" WAS DONE.

	THE VBE HAS JUST RESPONDED TO A READ MODIFIED OR POLL FROM THE HOST ON BEHALF OF A VIRTUAL TERMINAL WHICH HAD A "TEST REQUEST" ATTENTION OUTSTANDING. THIS IS A NORMAL OCCURRENCE.
EM3270	CALL COPYR WAS "DONE". THE VBE HAS JUST PERFORMED A COPY COMMAND FROM THE HOST. THIS IS A NORMAL OCCURRENCE.
CHKTAT	CHKTAT, USER Q FULL THE VBE HAS JUST DISCOVERED THAT THE USER'S QUEUE IS FULL, OR CLOSE ENOUGH TO FULL TO IMPEDE A FOLLOWING COMMAND SEQUENCE. THIS IS A NORMAL OCCURRENCE; HOWEVER, IF IT PERSISTS IT MAY INDICATE A PROBLEM WITH THE APPLICATION PROGRAM USING THE DEVICE.
CHKTAT	CHKTAT, SET "DB" BIT SAME AS ABOVE.
EMCFGB	EMCFGB, GETBK FAILED THE VBE'S FIRST ATTEMPT TO FETCH A FREE BUFFER HAS FAILED BECAUSE THE DYNAMIC BUFFER POOL IS COMPLETELY EMPTY. THIS INDICATES A MALFUNCTION WITHIN THE DPTX SYSTEM.
EMCFGB	EMCFGB, WAITING FOR BSCMAN THE VBE HAS BEEN STARTED UP BEFORE THE BSCMAN PROCESS WAS RUNNING. THE VBE WILL LOOP, DISPLAYING THIS MESSAGE EVERY FEW SECONDS, UNTIL BSCMAN IS STARTED UP. SEE THE SYSTEM ADMINISTRATOR'S GUIDE FOR DETAILS ON RUNNING BSCMAN.
HOLD	HOLD: KEY IS <N> THE VBE IS TRYING TO RECOVER FROM A "USER QUEUE FULL" CONDITION BY DOING SOME LIMITED LOCAL MESSAGE BUFFERING. THIS IS A NORMAL OCCURRENCE; THE MESSAGE IS FOR INTERNAL DEBUGGING ONLY.
HOLD	HOLD: OVERWRITE (<N>) THIS INDICATES A (NON-FATAL) BUG IN THE VBE.
LNKFLM	FIRST (OR) SECOND RBHA=0 (LNKFLM) THIS INDICATES A (NON-FATAL) BUG IN THE VBE.
LNKELM	NULL PTR, FIRST (OR) SECOND RB - FATAL (LNKELM) THIS INDICATES A SERIOUS FAILURE SOMEWHERE IN THE DPTX SYSTEM.
LNKELM	BAD CRBHA (LNKELM) THIS INDICATES A SERIOUS FAILURE SOMEWHERE IN THE DPTX SYSTEM.
SENBSC	BSCMAN Q FULL, WAITING THE QUEUE FROM THE VBE TO BSCMAN HAS BECOME FULL. THE VBE WILL LOOP WAITING FOR IT TO BECOME NOT FULL,

REPEATING THIS MESSAGE EVERY FEW SECONDS. THIS INDICATES A FAILURE EITHER IN THE VBE OR IN THE BSCMAN PROCESS.

VBGBK CGETBF FAILED

THE VBE HAS BEEN TEMPORARILY REBUFFED IN AN ATTEMPT TO GET SOME FREE BUFFERS. THIS IS A NORMAL OCCURRENCE; HOWEVER, IF IT PERSISTS IT MAY INDICATE A FAILURE SOMEWHERE WITHIN THE DPTX SYSTEM.

VBGBK VBGBK, WTG FREE STORE

SAME AS ABOVE, EXCEPT THAT THE VBE IS UNABLE TO CONTINUE PROCESSING UNTIL IT CAN GET THE NEEDED BUFFERS. THE VBE WILL REPEAT THIS MESSAGE EVERY FEW SECONDS UNTIL FREE BUFFERS ARE AVAILABLE.

VBVTAC VBVTAC, FATAL ERROR IN LOADQ1

THIS INDICATES A (NON-FATAL) ERROR IN THE VBE.

THE FOLLOWING IS A LIST OF THE POSSIBLE VALUES OF "COD1" AND "COD2" AND THEIR MEANINGS. THESE CODES ARE REFERENCED IN SOME OF THE EM3270 ERROR MESSAGES LISTED ABOVE, AND REFLECT ERROR OR UNUSUAL CONDITIONS DETECTED BY THE BSCMAN PROCESS WHICH FOR SOME REASON CANNOT BE HANDLED BY THE VBE. FOR MORE DETAIL ON ANY CODES, REFER TO BSCMAN DOCUMENTATION. VALUES FOR COD1, COD2 ARE IN OCTAL.

COD1	COD2	MEANING
1	--	RECEIVED "END OF TRANSMISSION" (EOT) CODE.
2	--	RECEIVED A POLL, SELECT, OR ENQUIRY (ENQ) MESSAGE.
3	--	RECEIVED TEXT MESSAGE FROM HOST.
4	--	RECEIVED POSITIVE ACKNOWLEDGEMENT (ACK0 OR ACK1) FROM HOST.
5	--	OUR TRANSMISSION HAS BEEN REJECTED, RETRIES EXCEEDED.
6	--	RECEIVED BLOCK WITH CHECKSUM ERROR, RETRIES EXCEEDED.
10	--	RECEIVED WAIT-ACKNOWLEDGE SEQUENCE (WACK) FROM HOST, RETRIES EXCEEDED.
11	--	RECEIVED REVERSE INTERRUPT SEQUENCE (RVI) FROM HOST.
12	1	THE PHYSICAL COMMUNICATIONS LINE IS CONNECTED.
12	2	THE PHYSICAL COMMUNICATIONS LINE IS DISCONNECTED.
12	3	THE MODEM "SIGNAL QUALITY DETECT" BIT HAS CHANGED TO "OFF" (VARIOUS MEANINGS DEPENDING ON MODEM).
12	4	THE MODEM "SIGNAL QUALITY DETECT" BIT HAS CHANGED TO "ON" (VARIOUS MEANINGS DEPENDING ON MODEM).
13	1	HOST HAS TIMED OUT WHEN IT SHOULD HAVE RESPONDED TO A TRANSMISSION, RETRIES EXCEEDED.
13	2	WE HAVE BEEN TOO SLOW IN RESPONDING TO THE HOST'S PREVIOUS TRANSMISSION (TWO-SECOND GRACE PERIOD EXPIRED).
13	3	A TRANSMISSION HAS FAILED (COULD BE SMLC, MODEM, CABLE OR CONFIGURATION PROBLEMS).
14	1	RECEIVED OUT-OF-SEQUENCE ACK FROM HOST, RETRIES EXCEEDED.
14	2	RECEIVED UNINTERPRETABLE BSC TEXT FROM HOST.

14	3	WE GAVE BSCMAN PROCESS A BAD CONFIGURATION ORDER.
14	4	RECEIVED UNINTERPRETABLE BSC TEXT FROM HOST.
14	6	RECEIVED A NEGATIVE ACKNOWLEDGEMENT (NAK) FROM HOST WHILE THE LINE WAS IDLE.
16	1	RECEIVE OVERRUN: RECEIVED LARGER TEXT MESSAGE FROM HOST THAN EXPECTED; MESSAGE LOST.
16	2	TRANSMIT UNDERRUN.
16	3	LOST SMLC STATUS.
16	4	TRANSMIT DATA NOT AVAILABLE.
16	5	INPUT TEXT FROM HOST LOST, LACK OF BUFFER SPACE.

5 PRIMOS INITIALIZATION ERROR MESSAGES

5.1 DISK BOOT ERROR MESSAGES

THE FOLLOWING LISTS ALL ERROR MESSAGES GENERATED BY THE DISK BOOT, THE PRIMOS PRELOADER ('PRIMOS') AND THE PRIMOS 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). DISK BOOT ERRORS 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.

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.

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
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

ONCE THE BOOT IS COMPLETELY LOADED AND ITS CONSISTENCY HAS BEEN VERIFIED, THEN ALL ERRORS (WITH THE EXCEPTION OF BAD DEVICE TYPE, 103) WILL CAUSE A MESSAGE TO BE PRINTED AT THE SYSTEM CONSOLE. THIS WILL BE FOLLOWED BY THE 'PHYSICAL DEVICE=' PROMPT.

5.1.1 PARITY ERROR AND MACHINE CHECK ERROR, 100, 101

PARITY AND MACHINE CHECK ERRORS ARE CAUGHT BY THE HARDWARE. ADDRESS INFORMATION IS NOT AVAILABLE ON THE P100, P200 OR P300. ADDRESS AND OTHER INFORMATION CAN BE FOUND IN THE DIAGNOSTIC STATUS WORD ON THE P400 OR P500.

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.

OTHERWISE, AFTER THE MEMORY TEST, THE ERROR MESSAGES, 'PARITY ERROR', OR, 'MACHINE CHECK', WILL BE PRINTED FOLLOWED BY THE 'PHYSICAL DEVICE=' PROMPT. IF THE ERRORS PERSIST, THE MESSAGES PERSIST.

5.1.2 BAD DEVICE TYPE, 103

THE BAD DEVICE TYPE CODE WILL APPEAR IN THE DATA LIGHTS IF A DEVICE TYPE OF 7 IS DETECTED.

5.1.3 BAD STATUS, 104

WHENEVER BAD STATUS IS DECTECTED DURING THE FIRST PHASE OF LOADING THE BOOT PROGRAM ITSELF, A HALT OCCURS WITH THE CODE 104 IN THE CONTROL PANEL DATA LIGHTS. THE STATUS IS STORED IN LOCATION 40 (OCTAL).

IF A BAD STATUS IS DETECTED WHILE LOADING DOS, THE MESSAGE 'BAD STATUS' IS PRINTED FOLLOWED BY THE STATUS WORD.

5.1.4 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 CONSOLE.

5.1.5 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.

5.1.6 'FILE' NOT FOUND

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

CONSOLE.

5.1.7 MEMORY TEST FAILURE

WHILE TESTING THE MEMORY, IF THE TEST PATTERN WRITTEN AND THEN SUBSEQUENTLY READ BACK DOES 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 CONSOLE.

5.1.8 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 ADDRESS LIGHTS ON THE CONTROL PANEL.

CODED HALT.

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

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

5.1.8.2 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).

5.2 PRELOADER ('PRIMOS') ERROR MESSAGES

5.2.1 <FILE-SYSTEM-ERROR-MESSAGE> CMDNCO (PRIMOS)

A FILE SYSTEM ERROR WAS ENCOUNTERED BY THE PRELOADER WHILE ATTEMPTING TO ATTACH TO CMDNCO.

5.2.2 <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.

5.2.3 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 COMMAND CONFIG.

5.2.4 <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>.

5.2.5 MISSING NTUSR, PAGDEV, OR COMDEV

THE CONFIGURATION DATA FILE DID NOT SPECIFY THESE REQUIRED PARAMETERS.

5.2.6 ILLEGAL PAGDEV

THE DEVICE SPECIFIED FOR PAGING IS NOT A LEGAL PAGING DEVICE.

5.2.7 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>.

5.2.8 ILLEGAL COMDEV

THE DEVICE SPECIFIED FOR THE COMMAND DEVICE IS NOT LEGAL.

5.2.9 ILLEGAL ALTDEV

THE DEVICE SPECIFIED AS THE ALTERNATE PAGING DEVICE IS NOT LEGAL.

5.2.10 <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.

5.2.11 END OF FILE. MISSING 'GO' CMND (PRIMOS)

THE CONFIGURATION DATA FILE DOES NOT INCLUDE A GO DIRECTIVE AS REQUIRED.

5.2.12 TRIOS ERROR

AN I/O ERROR OCCURRED WHILE PRELOADING THE PAGING DEVICE.

5.2.13 TOO MANY BAD SPOTS IN 'BADSPT' (BADSP$)

NUMBER OF BAD SPOTS ENTRIES IN THE FILE BADSPT EXCEEDS 16. PRIMOS SUPPORTS A MAXIMUM OF 16 BAD SPOTS IN BOTH PRIMARY AND ALTERNATE PAGING PARTITIONS.

5.2.14 INCORRECT 'BADSPT' FILE FORMAT (BADSP$)

THE FILE BADSPT HAS AN INCORRECT FILE FORMAT. FOR EXAMPLE: STARTING ADDRESS (SA) OF THE SAVED FORMAT FILE IS NOT EQUAL TO '1000.

5.2.15 BAD RECORD ADDRESS IS LESS THAN 16. (BADSP$)

A BAD RECORD IS FOUND TO HAVE AN ADDRESS LESS THAN OR EQUAL TO 16. THE FIRST 16 RECORDS OF A PARTITION CONTAINS THE MFD, BOOT, AND OTHER SPECIAL FILE.

5.2.16 SUM OF BAD SPOTS ON THE PRIMARY AND ALTERNATE PAGING DEVICE EXCEEDS 16

PRIMOS SUPPORTS A MAXIMUM OF 16 DEFECTIVE TRACKS (BAD SPOTS) ON BOTH PRIMARY AND ALTERNATE PAGING PARTITIONS.

5.3 PRIMOS INITIALIZATION ERROR MESSAGES

5.3.1 NTUSR+NPUSR+NRUSR TOO BIG (AINIT)

THE NUMBER OF TERMINAL PLUS PHANTOM PLUS REMOTE USERS EXCEEDS THE MAXIMUM NUMBER OF CONFIGURABLE USERS.

5.3.2 NRUSR INVALID (AINIT)

THE NUMBER OF REMOTE USERS SPECIFIED BY AN NRUSR DIRECTIVE EXCEEDS THE MAXIMUM NUMBER OF CONFIGURABLE REMOTE USERS (40, DECIMAL 32).

5.3.3 NTUSR, NPUSR, OR NRUSR INVALID (AINIT)

THE VALUE OF NTUSR, NPUSR, OR NRUSR IS INCORRECT.

5.3.4 SEEK FAILURE ON PAGDEV (AINIT)

THE INITIAL SEEK TO CYLINDER 0 ON THE PAGING DEVICE FAILED.

5.3.5 SEEK FAILURE ON ALTDEV (AINIT)

THE INITIAL SEEK TO CYLINDER 0 ON THE ALTERNATE PAGING DEVICE FAILED.

5.3.6 <FILE-SYSTEM-MESSAGE> CAN'T ATTACH TO CMDNC0 (AINIT)

A FILE SYSTEM ERROR WAS ENCOUNTERED WHILE ATTEMPTING TO ATTACH TO CMDNC0 FOR USER 1.

5.3.7 INVALID CONFIG COMMAND: <XXXXXX> (AINIT)

THE DIRECTIVE <XXXXXX> IN THE CONFIGURATION DIRECTIVE FILE IS NOT A RECOGNIZED CONFIGURATION DIRECTIVE.

5.3.8 BAD <CMND> PARAMETER (AINIT)

ONE OR MORE OF THE PARAMETERS SPECIFIED FOR THE CONFIGURATION DIRECTIVE <CMND> IS INVALID.

5.3.9 BAD LINE # IN AMLBUF CMND (AINIT)

AN AMLBUF DIRECTIVE SPECIFIES AN INVALID LINE NUMBER.

5.3.10 BAD DMQ AMLC CONFIGURATION (AINIT)

A DMQ BUFFER SIZE IN AN AMLBUF DIRECTIVE WAS TOO LARGE OR NOT EQUAL TO A POWER OF 2.

5.3.11 BAD LINE # IN ASRBUF CMND (AINIT)

AN ASRBUF DIRECTIVE SPECIFIED AN INVALID LINE NUMBER.

5.3.12 FILUNT INVALID (AINIT)

THE FILUNT DIRECTIVE SPECIFIES INCORRECT INFORMATION FOR PROPER CONFIGURATION.

5.3.13 TERMINAL I/O BUFFERS TOO LARGE (AINIT)

THE TOTAL SIZE OF THE TERMINAL I/O BUFFERS EXCEEDS 32K WORDS.

5.3.14 SMLC CTRLR # OUT OF RANGE (AINIT)

AN SMLC DIRECTIVE SPECIFIES AN INVALID CONTROLLER NUMBER.

5.3.15 SMLC LINE # OUT OF RANGE (AINIT)

AN SMLC DIRECTIVE SPECIFIES AN INVALID LINE NUMBER.

5.3.16 RESTART PLEASE

THIS MESSAGE APPEARS FOLLOWING ANY ERROR MESSAGE PRINTED BY THE PRIMOS 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.

5.4 NETWORK_INITIALIZATION_ERROR_MESSAGES

5.4.1 NETWORK NOT CONFIGURED (AINIT)

THIS MESSAGE IS NO LONGER ISSUED.

5.4.2 NETWORK_COLD_START_ERROR_MESSAGES

5.4.3 <FILE-SYSTEM-ERROR-MESSAGE> NETCON (AINIT)

A FILE SYSTEM ERROR HAS OCCURRED WHILE OPENING OR READING THE NETWORK CONFIGURATION FILE.

5.4.4 BAD NETCON FILE (AINIT)

THE NETWORK CONFIGURATION FILE HAS AN ILLEGAL FORMAT. RECREATE THE NETWORK CONFIGURATION FILE USING THE MOST RECENT VERSION OF NETCFG.

TOO MANY SMLC'S (AINIT)
TOO MANY RING NODES (AINIT)
TOO MANY IPC'S (AINIT)

THE NETWORK CONFIGURATION REQUIRES MORE TABLE SPACE THAN EXISTS IN THE OPERATING SYSTEM. THESE ERRORS CAN ONLY BE GENERATED IF

THE '-NOCHECK' OPTION WAS USED WITH THE EXTERNAL COMMAND NETCFG.

6 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.

6.1 ABBREV -- GLOBAL ENABLE/DISABLE OF ABBREV

ABBREV YES NO

THIS DIRECTIVE CONTROLS THE USAGE OF THE NEW INTERNAL COMMAND ABBREV. 'YES' IS THE DEFAULT, WHICH ALLOWS USERS TO USE ABBREV. SPECIFYING 'NO' WILL PREVENT ANY USERS FROM USING ABBREV. THIS DIRECTIVE IS USED TO SET THE FIGCOM VARIABLE ABBRSW.

6.2 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.

6.3 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).

6.4 AMLDIM -- SET AMLC EVENT TIMERS

THE AMLTIM DIRECTIVE IS EMPLOYED TO CONTROL SOME OF THE VARIABLE EVENT TIMERS WITHIN THE AMLC PROCESS. THE SYNTAX FOR THE DIRECTIVE IS:

```
AMLTIM [<TICKS>] [<DISCTIME>] [<GRACETIME>]
```

THE <TICKS> ARGUMENT IS THE NUMBER OF TENTHS OF A SECOND OF THE FREQUENCY FOR PERFORMING CARRIER CHECK OPERATIONS. AT THE END OF EACH PERIOD, THE AMLC PROCESS CHECKS FOR CARRIER LOSS ON LINES, AND, IF A LOSS HAS OCCURED, WILL FORCE THE LINE'S DATA TERMINAL READY (DTR) SIGNAL TO THE MODEM TO GO INACTIVE UNTIL THE NEXT SAMPLE PERIOD. THIS EFFECTIVELY CAUSES THE MODEM TO DISCONNECT THE INCOMING TELEPHONE LINE. THE PERIOD INDICATED FOR THIS TIME ALSO SERVES A BASE CLOCK FOR THE OTHER CARRIER OPERATIONS DESCRIBED BELOW. THE DEFAULT VALUE IS 2, I.E., 0.2 SECONDS. THE VALUE SPECIFIED MUST BE GREATER THAN ZERO.

THE <DISCTIME> ARGUMENT IS THE NUMBER OF TENTHS OF A SECOND OF THE FREQUENCY FOR FORCING DTR SIGNAL TO THE MODEM TO GO INACTIVE ON THOSE LINES WHICH DO NOT HAVE CARRIER PRESENT. THIS IS DONE FOR ROBUSTNESS OF THE MODEM AND THE INCOMING LINES. THE DEFAULT VALUE FOR THIS ARGUMENT IS 1800 (DECIMAL), I.E., THREE MINUTES. SPECIFYING A VALUE OF ZERO WILL DISABLE THIS FEATURE. THE VALUE SPECIFIED, IF NOT ZERO, MUST NOT BE LESS THAN THE <TICKS> ARGUMENT AND IS TRUNCATED TO THE NEAREST MULTIPLE OF THAT ARGUMENT.

THE <GRACETIME> ARGUMENT IS THE NUMBER OF TENTHS OF A SECOND OF THE GRACE PERIOD FOR TERMINAL LINES WHICH HAVE AN ACTIVE CARRIER, BUT ARE NOT CONNECTED TO LOGGED-IN PROCESSES. THIS ARGUMENT IS EFFECTIVELY THE MINIMUM TIME THAT THE LINE IS ALLOWED TO ESTABLISH ITSELF WITH A LOGGED-IN PROCESS. THE ACTUAL GRACE INTERVAL WILL VARY RANDOMLY FROM THE TIME SPECIFIED BY THIS ARGUMENT TO TWICE THE TIME OF THE ARGUMENT. THE DEFAULT VALUE FOR THIS ARGUMENT IS ZERO, I.E., IT IS DISABLED. SPECIFYING A

VALUE OF ZERO WILL DISABLE THIS FEATURE. THE VALUE SPECIFIED, IF NOT ZERO, MUST NOT BE LESS THAN THE <TICKS> ARGUMENT AND IS TRUNCATED TO THE NEAREST MULTIPLE OF THAT ARGUMENT.

6.5 AMLCLK -- SET AMLC BAUD RATE CLOCK

THE AMLCLK DIRECTIVE IS USED TO SET THE SOFTWARE PROGRAMABLE BAUD-RATE CLOCK IN THE AMLC HARDWARE. THE SYNTAX OF THE AMLCLK DIRECTIVE IS:

```
AMLCLK <BAUDRATE>
```

THE <BAUDRATE> ARGUMENT SPECIFIES THE DESIRED BAUD-RATE FOR THE PROGRAMABLE CLOCK IN THE AMLC HARDWARE. THE VALUE SPECIFIED MUST NOT BE LESS THAN 100 (DECIMAL) NOR GREATER THAN 9600 (DECIMAL). THIS OPERATION WAS PREVIOUSLY PERFORMED BY CHANGING THE VARIABLE "AMLCLK" IN THE AMINIT MODULE.

6.6 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.

6.7 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).

6.8 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>.

6.9 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/<PACDEV>	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

6.10 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.

6.11 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 '"').

6.12 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>	20	(16 DECIMAL)
<MAXUNT>	100	(128 DECIMAL)
<TOTUNT>	4000	(2048 DECIMAL)

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 NUMBER 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 128. OF THE 128 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 128 FILE UNITS OPEN SIMULTANEOUSLY (128*128=16384), 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.

6.13 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.

6.14 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
```

6.15 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).

6.16 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.

6.17 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.

6.18 LOUTOM -- SPECIFY INACTIVITY-LOGOUT QUANTUM

LOUTOM <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 LOUTIM. <MINS> MUST BE GREATER THAN ZERO.

6.19 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.)

6.20 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.

6.21 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.)

6.22 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.

6.23 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.

6.24 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 511 DECIMAL (777 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.)

6.25 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.

6.26 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 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).

6.27 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 (SEGMENTS NUMBERS 0-14 AND USER 1'S SEGMENT 6000 AND 6002).

6.28 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.

6.29 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 SEPARATE DOCUMENT DESCRIBING NETCFG.)

6.30 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.) ASSUME THAT THE SYSTEM DEFAULT DOES NOT PERMIT MULTIPLE WRITERS.

6.31 SMLC -- ENABLE AND CONFIGURE SMLC LINES

SMLC ON
SMLC CNTRLR <CTRLR-NUMBER> <DEVADR>
SMLC SMLCNN <CTRLR-NUMBER> <LINE-NUMBER>
SMLC DSC <LINE> <STRAP> <PROC> <RECV> N

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.

6.31.1 SMLC DATA SET CONTROL

NOTE: ONLY THE RSCMAN PROCESS CURRENTLY INTERPRETS THE DATA STRUCTURES INITIALIZED BY THE "DSC" OPTION.

DSC OPTION OF SMLC COMMAND

THE COLD START DIRECTIVE SMLC CAN BE USED TO SPECIFY DATA SET CONTROL CONFIGURATION FOR A SYNCHRONOUS LINE ON THE SMLC/HSSMLC/MDLC BOARD. THE FORM OF THE DIRECTIVE IS:

SMLC DSC <LINE> <STRAP> <PROC> <RECV>

EACH ARGUMENT IS SPECIFIED AS AN OCTAL NUMBER:

<LINE>

LOGICAL LINE NUMBER -- 0 THRU 3.

<STRAP>

BIT SET INDICATES SIGNALS WHICH ARE STRAPPED ON BY THE SOFTWARE.

:1 - DATA TERMINAL READY (DTR)

:2 - REQUEST TO SEND (RTS)

IN ADDITION, SPEED SELECT IN GERMANY IS SPECIFIED VIA THE :10 BIT:

SET - FAST

RESET - SLOW

<PROC>

INDICATES DATA SET CONTROL PROCEDURE (TO BE USED FOR TRANSMITTING DATA) AS FOLLOWS:

1 - NO DATA SET ORDERS. USUALLY USED WITH DTR AND RTS STRAPPED ON, WITH MODEMS USED FOR FOUR WIRE FULL DUPLEX SERVICE.

2 - USE DATA SET ORDERS AS FOLLOWS:
ISSUE RTS, WAIT FOR CLEAR TO SEND (CTS), SEND, DROP RTS. USUALLY USED WITH MOST HALF DUPLEX MODEMS.

3 - USE DATA SET ORDERS AS FOLLOWS:
WAIT FOR .NOT. CARRIER DETECT (CD), ISSUE RTS, WAIT FOR CTS, SEND, DROP RTS. RARELY USED, BUT MAY BE NECESSARY WITH 201 SERIES MODEMS ONLY IF LINES ARE VERY NOISY. TRY "2" FIRST.

<RECV>

INDICATES WHETHER THE RECEIVER IS TO BE TURNED ON BEFORE OR AFTER TRANSMITTING:

0 - TURN ON RECEIVER BEFORE TRANSMITTING. THIS PROVIDES THE FASTEST RESPONSE AND SHOULD BE USED IF POSSIBLE.

1 - TURN ON RECEIVER AFTER TRANSMITTING. THIS SETTING MUST BE USED WITH TWO WIRE 201 SERIES MODEMS. THIS SETTING MAY BE TRIED ON OTHER TWO WIRE SYSTEMS ONLY IF PROBLEMS APPEAR WHICH CANNOT BE SOLVED BY OTHER MEANS.

THE DEFAULT SETUP, IF NO SMLC DSC IS SPECIFIED, IS THE EQUIVALENT OF INCLUDING THE FOLLOWING LINE IN THE CONFIGURATION DATA FILE:

SMLC DSC <LINE> 1 2 0

6.32 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.)

6.33 UPS -- SET SYSTEM TO PERFORM RESTART AFTER POWER FAILURE

UPS <NUMBER>

<NUMBER> IS USED TO SET THE UPSSW VARIABLE IN FIGCOM -- THE SYSTEM UPS ACTION VARIABLE. THIS VARIABLE DETERMINES WHAT ACTIONS ARE TAKEN AFTER A POWER FAILURE. VALID VALUES OF <NUMBER> ARE:

177777 - NO UPS (DEFAULT)
0 - UPS, BUT HALT ON A WARM-START
>0 - NUMBER OF SECONDS TO DELAY AFTER WARM-START (NO OPERATOR START IS REQUIRED)

THIS DIRECTIVE ENABLES AN AUTOMATIC WARM START AFTER POWER IS RESTORED FOLLOWING A POWER FAILURE CHECK. IT IS DESIGNED TO BE USED WHEN AN UNINTERRUPTIBLE POWER SUPPLY (UPS) IS USED TO MAINTAIN POWER TO THE CPU AND MEMORY.

THE NUMBER OF SECONDS TO DELAY AFTER A WARM-START IS THE AMOUNT OF TIME IT TAKES FOR THE DISK(S) TO COME UP TO THE PROPER NUMBER OF REVOLUTIONS PER MINUTE. TYPICALLY, THIS IS ABOUT 1 MINUTE (ABOUT 100 OCTAL SECONDS) FOR A STORAGE MODULE.

6.34 WIRMEM -- PRINT SIZE OF WIRED MEMORY

THIS DIRECTIVE CAUSES THE SIZE OF WIRED MEMORY, IN PAGES, TO BE PRINTED ON THE SYSTEM CONSOLE DURING COLDSTART. NOTE THAT THIS VALUE CHANGES AS THE SYSTEM RUNS, BUT THAT IT DOES GIVE THE ADMINISTRATOR SOME IDEA OF THE RELATIVE MEMORY COST OF THE CONFIGURATION SELECTED.

7 MAPGEN - A TOOL FOR BUILDING PRIMOS

INTRODUCTION

THE MAPGEN TOOL IS A UTILITY PROGRAM EMPLOYED TO BUILD THE PRIMOS 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 SEGMENT-IMAGE 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.

7.1 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 SEGMENT-IMAGE FILE TEMPLATE WHICH IS TO BE INITIALLY LOADED INTO THE SPECIFIED

SEGMENT. IF A FILE IS NOT 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 SEGMENT-IMAGE FILE ARE DEFINED IN THE PAGE-MAP AS BEING REFERENCEABLE AND PRELOADED. ALL PAGES OUTSIDE OF THE SEGMENT-IMAGE FILE RANGE WILL BE UNREFERENCEABLE. THE USE OF AN ASTERISK IN PLACE OF THE <FILE> ARGUMENT WILL CAUSE ALL OF THE PAGES OF THE SEG TO BE UNREPLACEABLE. 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>  ::= <SYMBOL>  $LOW
<LADDR>  ::= <SYMBOL>  $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 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 BEGINNING AND THE END (PLUS ONE), RESPECTIVELY, OF THE SEGMENT-IMAGE 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 SEGMENT-IMAGE FILE.

WIRE -- SPECIFY RANGE LOCKED IN COLD-START MODULE

WIRE <RANGE> -OR- LOCK <RANGE>

THE SPECIFIED RANGE IS RESIDENT IN THE COLD-START SEGMENT-IMAGE 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 SEGMENT-IMAGE FILE AND IT IS PLACED AT SUCH A LOCATION IN THAT FILE SUCH THAT BOTH THE VIRTUAL- AND PHYSICAL-ADDRESSES ARE IDENTICAL.

CONTIG -- SPECIFY CONTIGUOUS PAGES IN PHYSICAL MEMORY

CONTIG <RANGE>

THE SPECIFIED RANGE IS RESIDENT IN THE COLD-START SEGMENT-IMAGE FILE AND IT'S PAGES ARE PLACED IN CONTINGUOUS PAGES OF PHYSICAL MEMORY.

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.

7.2 INFORMATION STORE DIRECTIVES

THE INFORMATION STORE DIRECTIVES CAUSE SOME INFORMATION GIVEN BY THE SEGMENT DESCRIPTION DIRECTIVES TO BE STORED WITHIN THE SEGMENT-IMAGE 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>

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 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 A SEGMENT-IMAGE FILE. THIS 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> <POINTER>

THE INITIAL PAGE-MAPS (HMAP, THE HARDWARE MAP, AND LMAP, THE LOGICAL ADDRESS MAP) FOR THE SYSTEM ARE STORED AT THE SPECIFIED

7.3 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 SEGMENT-IMAGE 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 COLD-START 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.

STAMP -- STORE DATE-TIME STAMP IN OBJECT

STAMP <LOCATION>

THIS DIRECTIVE CAUSES THE CURRENT DATE AND TIME TO BE STORED AT THE SPECIFIED LOCATION.

VERSION -- STORE LITERAL TEXT IN OBJECT

VERSION <LOCATION> <STRING>

THIS DIRECTIVE CAUSES THE SPECIFIED STRING TO BE STORED AT THE SPECIFIED LOCATION. THE STRING IS STORED IN THE FORMAT WHERE THE FIRST WORD IS THE STRING LENGTH IN CHARACTERS AND THE TEXT IMMEDIATELY FOLLOWS.

7.4 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.)

```
* C_COLD, PR1400, OS GROUP, 06/05/79.
*  BUILD PAGE MAPS AND CREATE *COLDS MEMORY IMAGE.
*
FILMEM ALL     /* TO ENSURE *COLDS IS PREDICTABLE
R *MAPGEN
*
*    THIS FILE CONTAINS THE DIRECTIVES FOR THE *MAPGEN PROGRAM
*    FOR GENERATING PRIMOS IV REV 17 - SINGLE 64 USER VERSION.
*
*
TABLE M_RING0
*
*    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, DISK CHANN
EL PROG.
    EMPTY      SEG0   AMLQCB
*
*    SEGMENT 01 - CONTAINS ASSOCIATIVE BUFFERS.
```

```
*
SEGMENT     1       *
    PAGE          0  177777              /* ALLOCATE NO PAGE SPACE
FOR BUFFERS.
    EMPTY         0  177777
*
*   SEGMENT 04 - CONTAINS PHANTOM INTS., SEMAPHORES, PCB, FAULT H
ANDLERS.
*
SEGMENT     4  PR0004
    EMPTY     SEG4  SEG4SZ              /* DO NOT NEED PAGING SPAC
E.
    WIRE      SEG4    VPSD              /* PHANTOM CODE, CHECKS, B
ASE, SEMCOM,
*                                       /* WARM/COLD START, ETC.
    WIRE      VPSD    VPSD   12000      /* VPSD WIRED ONLY FOR DEB
UGGING.
    WIRE     76000  U2PCRE              /* INTERRUPT HANDLERS, CHK
LOG, 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  WIRE6    ASSCOM             /* COLD-START RESIDENT UP
TO ASSCOM.
*
*   SEGMENT 07 - CONTAINS TFLIOB BUFFERS.
*
SEGMENT     7       *
    WIRE          0    2000             /* USER 1'S OUTPUT BUFFER
WIRED.
    WIRE      30000   32000             /* USER 1'S INPUT BUFFER W
IRED.
    EMPTY         0  177777              /* NO PACING SPACE.
*
*   SEGMENT 10 - CONTAINS USRCOM.
*
SEGMENT    10  PR0010
    RESIDE        0    2000             /* USER 1 COMMON COLD-STAR
T RESIDENT.
    PAGE          0  177777              /* REST IS UNINITIALIZED.
*
*   SEGMENT 11 - CONTAINS FILE SYSTEM PROCEDURE AND LINKAGE.
*
SEGMENT    11  PR0011
*
```

```
*    SEGMENT 12 - CONTAINS NETWORK SYSTEM PROCEDURE AND LINKAGE.
*
SEGMENT    12   PR0012
*
*    SEGMENT 13 - CONTAINS RING 3 STUFF.
*
SEGMENT     13   PR0013
*
*    SEGMENT 14 - CONTAIN ONE-TO-ONE STUFF LIKE PAGCOM, HDRBUF, SP
ECIAL CODE.
*
SEGMENT    14   PR0014
     EMPTY      SEG14     SDWE                /* NO PAGING SPACE.
     WIRE       SEG14     MMAP    10000       /* HDRBUF, CONFIG, RSAV, F
IGCOM,
*                                             /* MMAP.
     ONE        SEG14     MMAP    10000       /* TAPE-DUMP, MEMORY-SCAN
CODE, WARM-COLD
     WIRE       PAGCOM    SDWE                /* PAGCOM, PAGDEV, SDW0, S
DW1, SDW3 TABLES.
     ONE        PAGCOM    SDWE
*
*    SEGMENTS 15 THRU 20 - CONTAIN 3270 PROJECTS CODE AND LINKAGE
*
SEGMENT     15   PR0015
*
SEGMENT     16         *
     PAGE             0   177777
*
SEGMENT     17         *
     PAGE             0   177777
*
SEGMENT     20         *
     PAGE             0   177777
*
*    SEGMENT 22 - CONTAINS PAGE MAPS
*
SEGMENT   22       PR0022
     WIRE             0   HMAPE
     CONTIG           0   HMAPE
     PAGE         HMAPE   177777
*
*    SEGMENT 6002 - CONTAINS RING-3 STACK.
*
SEGMENT   6002   PR6002
     PAGE             0   177777
*
*    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, MMAP, PUTSEG, AND SDW'S.
HMAP    HMAP HMAPPA
MMAP    MMAP
PTUSEG PTUSEG
SDW           0 SDW0
SDW           3 SDW3
*
*    PRINT A MEMORY MAP.
*
MAP
*
*    DUMP OF PAGE MAPS FOR DEBUGGING ONLY.
*
*  DUMP
*
*    STAMP THE ORJECT TEXT BEFORE BUILDING.
*
STAMP STAMP
*
* SET VERSION NUMBER.
*
VERSION VERSIO 17.1.6
*
*    SAVE COLD-START IMAGE.
*
COLD  *COLDS  1/153777  4/1010
*
QUIT
*
CO  CONTINUE
CO  TTY
```

8 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.

8.1 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     . . .
```

8.2 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.

NOTE: THE FOLLOWING SCHEMES ARE PROVIDED AS ALTERNATIVES TO USING THE IOCS ROUTINES THAT FTN USES. THE ERROR RECOVERY PROVIDED IN THE IOCS ROUTINES CORRESPOND TO THAT DESCRIBED FOR SIMPLE WRITE ERROR RECOVERY.

8.2.1 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.

ONLY THE VERSION THREE CONTROLLER (MPC-3), WHICH SUPPORTS THE 6250 BPI TAPE DRIVES, HAS AN ERASE COMMAND. ON OTHER CONTROLLERS, 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.

PPOGRAM 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).

8.2.2 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.

8.3 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.

9 PRIMOS INTERNAL LOGIC

THE FOLLOWING DESCRIBES THE MAJOR MODIFICATIONS THAT HAVE BEEN MADE TO THE INTERNAL LOGIC OF PRIMOS. THIS INFORMATION IS REQUIRED NORMALLY ONLY BY THOSE INVOLVED IN THE MODIFICATION OR MAINTENANCE OF PRIMOS.

9.1 SEGMENT 4

ON THE PRIME V-MODE PROCESSORS 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.

9.2 SEGMENT 14

THE PRIMOS 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.

ALL CODE AND DATA WHICH HAS TO RESIDE ONE-TO-ONE WITH PHYSICAL MEMORY IS LOCATED IN SEGMENT NUMBER 14. THIS SEGMENT CONTAINS 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.

9.3 SEGMENT 22

ALL PAGE-MAPS FOR PRIMOS ARE LOCATED IN SEGMENT 22. THERE IS ROOM IN THIS SEGMENT FOR 511 PAGE MAPS. THE PAGES OF SEGMENT 22 NEED NOT BE ONE-TO-ONE WITH PHYSICAL MEMORY. HOWEVER, THE PAGES CONTAINING THE PAGE-MAPS FOR COLD-START PRIMOS SEGMENTS MUST BE CONTIGUOUS IN PHYSICAL MEMORY.

9.4 FILE SYSTEM DATA STRUCTURES

9.4.1 STRUCTURE OF USRCM$

USRCM$ HAS BEEN MODIFIED SO THAT BOTH THE UNIT TABLE ENTRIES AND ATTACH POINT ENTRIES CONTAIN INDICES TO UNIT TABLE ENTRIES IN A SEPARATE COMMON AREA, UTCOM$. THE LOGIN NAME REMAINS IN USRCM$ AS A 32 CHARACTER STRING. ONLY SIX CHARACTERS (3 WORDS) ARE USED; THE REMAINING WORDS ARE RESERVED FOR FUTURE USE.

AS IN PREVIOUS REVISIONS, VARIABLES IN USRCM$ SUCH AS UNITAB (START OF UNIT TABLE INDICES) ARE ARRAYS EQUIVALENCED THUS PROVIDING BASICALLY THE SAME FUNCTION AS PL/1 BASED OVERLAYS. THE EXPRESSION:

```
LUNIT = UNITAB ((USR-1)*USRSIZ)+UNITNR)
```

WHERE: USR IS THE PROCESS OR USER NUMBER
USRSIZ IS THE PER PROCESS LENGTH OF "USRCM$".
UNITNR IS THE FILE UNIT NUMBER.

THUS BECOMES AN INDEX TO THE UNIT TABLE ENTRY FOR THE UNIT. IF LUNIT 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:

```
LUNIT = CURATT ((USR-1)*USRSIZ)
LUNIT = HOMATT ((USR-1)*USRSIZ)
```

ARE USED TO OBTAIN THE INDICES 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. NOTE THAT PUDCOM IS IN THE PER-USER STACK SEGMENT SO THAT LUSR FOR AN PROCESS POINTS TO THAT PROCESS'S "SLICE" OF USRCM$.

9.4.2 STRUCTURE OF UTCOM$

THE COMMON AREA UTCOM$ CONTAINS UNIT TABLE ENTRIES. THE VARIABLES IN UTCOM$ SUCH AS VSTAT, VBRA, ETC. ARE EQUIVALENCED. 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 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.
VPOPRW	POSITION IN VPOPRA RECORD OF ENTRY CONTROL WORD (ECW) OF FILE ENTRY.
VTHRED	INDEX OF NEXT UNIT TABLE ENTRY ON THIS HASH THREAD.

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.

9.4.3 UNIT_TABLE_HASHING

ALL LOCAL UNIT TABLE ENTRIES ARE HAHED SO THAT RWLKCK FOR CROSS UNIT CHECKING IS FASTER. THE ARRAY UTHASH IN FSCOM IS THE HASH TABLE. EACH ENTRY IS ZERO FOR NO UNITS HASHED HERE OR THE INDEX OF THE FIRST UNIT TABLE ENTRY THAT HASHES HERE. EACH UNIT TABLE ENTRY HAS A LINK (VTHRED) TO THE NEXT ENTRY OR ZERO IF IT IS THE LAST ONE FOR THIS HASH INDEX.

THERE ARE THREE NEW ROUTINES TO HANDLE THE UNIT TABLE HASHING. THEY ARE: FSHASH, FSAHSH, AND FSUHSH. FSHASH IS A FUNCTION THAT TAKES THE BRA AND THE DVNO OF A FILE AND RETURNS A HASH INDEX INTO UTHASH. FSAHSH TAKES A UNIT TABLE ENTRY INDEX AND ADDS THAT ENTRY TO THE HASH THREAD. FSUHSH TAKES A UNIT TABLE ENTRY INDEX AND REMOVES IT FROM THE HASH THREAD.

9.4.4 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 SUBROUTINE 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.

9.5 TREENAME HANDLING IN DOSSUB

FOR THOSE COMMANDS WHICH ACCEPT TREFNAMES (SEE %TREENAMES%), A CALL IS MADE TO TA$ (A MODIFIED VERSION OF THE TA COMMAND) TO DO ANY TREENAME PROCESSING AND ATTACHING THAT IS NECESSARY.

TA$ DOES ALL OF THE ATTACH COMMAND'S PROCESSING. IN ALL OTHER CASES, TA$ RETURNS TO DOSSUB THE LAST ELEMENT OF THE TREENAME IN THE VARIABLE FNAME, AND IS ATTACHED TO THE APPROPRIATE PLACE SO THAT DOSSUB MAY SUCCESSFULLY PROCESS THE COMMAND. IT IS IMPORTANT TO NOTE THAT TA$ CALLS RDTK$P.

IF AN ATTACH WAS MADE ANYWHERE TO PROCESS THE TREENAME, THE LOGICAL VARIABLE ATSW IS SET TO .TRUE. IN TA$. ONCE DOSSUB HAS COMPLETED THE PROCESSING OF THE COMMAND, ATSW IS RESET TO .FALSE. AFTER RE-ATTACHING TO THE HOME UFD.

9.6 DISK ERROR RECOVERY DURING WARM STARTS

WHEN A WARM START IS PERFORMED, A CHECK IS NOW MADE TO SEE IF THE CURRENT DISK OPERATION HAS TAKEN LESS THAN (OR EXACTLY) 10 SECONDS. IF THE CURRENT OPERATION HAS TAKEN LESS THAN (OR EXACTLY) 10 SECONDS, A SINGLE ERROR WILL BE LOGGED, AND THE OPERATION WILL BE RETRIED. IF MORE THAN 10 RETRY OPERATIONS HAVE FAILED, IT IS ASSUMED THAT THE DISK OR CONTROLLER IS HUNG, AND NO MORE RETRIES WILL BE ATTEMPTED. THIS CORRECTS A PROBLEM WHEREBY A WARM START COULD DESTROY DATA ON A DISK.

9.7 NEW SEGMENT 0 WINDOW ALLOCATION ROUTINE: GTWNDO

OVERVIEW

ALL MODULES DOING MAPPED I/O THROUGH SEGMENT 0 (MTDIM, MPCDIM, FARDIM, ETC.) FORMERLY HAD SEGMENT 0 PAGE MAP ENTRIES STATICALLY ALLOCATED ON A PER-MODULE BASIS. CHANGING THIS ALLOCATION REQUIRED REASSEMBLY OF AT LEAST ONE MODULE (SEG14) AND RELOADING OF THE OPERATING SYSTEM. IN THE NEW SCHEME, SEGMENT 0 WINDOWS (PAGE MAP ENTRIES OR GROUPS OF PAGE MAP ENTRIES) ARE ALLOCATED ONLY WHEN NECESSARY (I.E., THE FIRST TIME A PARTICULAR DEVICE IS USED). THIS ALLOCATION IS DONE BY THE NEW ROUTINE GTWNDO DESCRIBED BELOW.

USAGE

GTWNDO

CALLING SEQUENCES:

```
   PMA:
               (CALL TO ERRRTN IF UNSUCCESSFUL)
               CALL      GTWNDO
               AP        <#PAGES>,S
               AP        WNDPTR,SL
*
               (RETURN ERROR CODE; CALLER WILL TAKE APPROPRIATE
               ACTION)
               CALL      GTWNDO
               AP        <#PAGES>,S
               AP        WNDPTR,S
               AP        CODE,SL
*
   PL1:
    DCL  NUM_PAGES          FIXED(15),
         WINDOW_PMNT_PTR    PTR,
        [CODE               FIXED(15),]
         GTWNDO             EXT ENTRY(FIXED, PTR [, FIXED]);
          .
          .
    CALL GTWNDO(NUM_PAGES, WINDOW_PMNT_PTR [, CODE]);
*
   FORTRAN:
      INTEGER*2  NUMPGS
      INTEGER*4  WNDPTR
     [INTEGER*2  CODE]
      .
      .
      CALL  GTWNDO(NUMPGS, WNDPTR [, CODE])
*
  RETURN: (IN SECOND ARGUMENT)
   IP TO PAGE MAP ENTRY IN HMAP (SEG14)
*
  ERROR ACTION:
   IF CODE IS PASSED, RETURN ERROR CODE E$ROOM
   ELSE CALL ERRRTN
*
```

THE FIRST ARGUMENT (NUM_PAGES) IS THE SIZE OF THE WINDOW IN PAGES. THE SECOND, RETURNED BY GTWNDO, IS A POINTER TO THE PAGE MAP ENTRY OF THE FIRST PAGE IN THE WINDOW. THE OPTIONAL THIRD ARGUMENT IS A RETURN CODE -- EITHER 0 OR E$ROOM.

THE FOLLOWING IS AN EXAMPLE OF GTWNDO USAGE (TAKEN FROM MTDIM):

```
*
*       INITIALIZE CONTROLLER, IF NEEDED
.*
        LDA     F$MTFLG2,*Y
        BEQ     T$MT2               0 => INITIALIZED
        LDL     PWINDS,1            WINDOW ALLOCATED?
        BLNE    MTIN1               YES, THE HMAP POINTER IS THERE
        CALL    GTWNDO              GET WINDOW
        AP      =6,S                6 PAGES LONG
```

```
        AP      WINDOW,SL       TO TEMPORARY
        LDX     SAVEX           RESTORE CONTROLLER INFO INDEX
        LDY     YSAVE
        LDL     WINDOW
        STL     PWINDS,1        STORE HMAP POINTER
MTIN1   LDA     DMADDR,1        GET DMA CHANNEL ADDR FOR THIS
                                CONTROLLER
        OTA     XDMXCH,Y        OUTPUT CHANNEL ADDRESS
        BCNE    NOMPC           IF NO "SKIP", THEN NO MPC
        .
        .
        .
```

9.8 KERNAL SUPPORT FOR BATCH

THE FOLLOWING DESCRIBES THE CHANGES TO PRIMOS EXECUTION LOGIC TO SUPPORT BATCH. MODULES AFFECTED (EITHER CHANGED OR NEW) ARE LISTED IN ALPHABETICAL ORDER, WITH THE EXCEPTION OF PRIMOS DATABASES WHICH ARE MENTIONED AS THEY APPLY TO THE CHANGE.

9.8.1 AINIT

PRIOR TO COPYING USER 1'S RING 3 STACK FOR LATER INITIALIZATION OF USERS, A NEW VALUE IN PUDCOM, PRVL, IS SET TO FALSE, SO THAT THIS VALUE WILL BE FALSE FOR ALL USERS. ONCE THIS COPY IS DONE, THE VALUE FOR PRVL FOR USER 1 IS SET TO TRUE.

9.8.2 BATCH$

THIS IS A NEW MODULE, AND CAN ONLY BE USED (SUCESSFULLY) BY A USER THAT HAS THE ABOVE DESCRIBED VALUE PRVL SET TO TRUE. THIS MODULE IS ESSENTIALLY THE SAME AS A CALL TO PHANT$, EXCEPT THAT THE USER WITH PRVL SET TO TRUE (TYPICALLY THE BATCH MONITOR), CAN START UP A PHANTOM WITH A DIFFERENT NAME.

9.8.3 DOSSUB

THE CHANGES MADE TO DOSSUB AS THEY RELATE TO BATCH IS THE ENHANCEMENT OF THE CHAP LOWER COMMAND, WHICH NOW ALLOWS A USER TO SPECIFY THAT THE LENGTH OF THE TIMESLICE CAN BE LOWERED IN TENTHS OF SECONDS. THIS ENHANCED COMMAND MUST BE SPECIFIED WITH A PRIORITY. THE SYNTAX IS

CHAP LOWER PRIORITY TIMESLICE

WHERE TIMESLICE IS SOME VALUE BETWEEN 1 AND 100. IF THE TIMESLICE CAN NOT BE LOWERED, OR IS GREATER THAN THE SYSTEM'S TIMESLICE, THE COMMAND IS EFFECTIVELY A NO-OP.

ALSO IN DOSSUB, THE CODE FOR THE LO COMMAND (LOGOUT), HAS BEEN MOVED TO A SEPARATE SUBROUTINE LOGO$$, WHICH IS ALSO SUPPORTIVE TO THE BATCH FACILITY. (A DESCRIPTION OF LOGO$$ APPEARS AFTER LOGIN.)

9.8.4 LOGIN

LOGIN ENSURES THAT THE VALUE FOR PRVL IS SET TO FALSE FOR A LOGGING IN USER. IF A PHANTOM JOB IS LOGGING IN, PHPRVL AS SET IN PHANT$ IS COPIED TO THE LOGGING IN PHANTOM'S RING 0 STACK AS ITS VALUE FOR PRVL. (PHANT$ CHANGES EXPLAIN THIS A BIT MORE.)

DURING LOGOUT, THE USER'S TIME SLICE VALUE IS RESET TO THE LAST TIMESLICE ISSUED AT THE SYSTEM CONSOLE, OR THE DEFAULT TIMESLICE.

9.8.5 LOGO$$

THIS IS A NEW MODULE THAT INCORPORATES MOST OF THE CODE THAT WAS IN DOSSUB, AND ALSO INCLUDES SOME NEW FUNCTIONALITY FOR BATCH. THIS MODULE ALLOWS A USER WITH PRVL SET IN THEIR RING 0 STACK TO LOG OUT ANY AND ALL USERS BY NUMBER OR NAME.

9.8.6 PABORT

THIS MODULE HAS BEEN CHANGED SO THAT PHANTOMS CAN NO LONGER TIME OUT FOR INACTIVITY, AND SO THAT CPU AND IO LIMITS ARE RECOGNIZED FOR PHANTOMS.

9.8.7 PHANT$

THE CHANGE TO PHANT$ IS THAT IT NOW HAS A NEW VARIABLE IN SUPCOM TO SET WHEN FIRING UP A PHANTOM (CALLED PHPRVL). IF USER 1 IS STARTING UP THE PHANTOM, THE PHANTOM IS SAID TO BE PRIVILEGED, AND PHPRVL IS SET TO TRUE; IN ALL OTHER CASES, PHPRVL IS SET TO FALSE.

9.8.8 IMAIN

THE ONLY CHANGES TO IMAIN RELATED TO BATCH ARE THE NEW VARIABLES PHPRVL AND SAVETS IN THE SUPCOM DATABASE.

9.9 PRIMOS SUPPORT FOR BAD SPOTS HANDLING ON PAGING PARTITIONS

PRMLD AND AINIT ARE THE ONLY TWO SYSTEM STARTUP AND INITIALIZATION MODULES WHICH WERE CHANGED TO IMPLEMENT THIS MECHANISM.

9.9.1 PRMLD - PRIMOS PRELOADER

IN ADDITION TO THE NORMAL FUNCTION PERFORMED BY PRMLD, THE PRELOADER WILL:

1. CHECK THE PACK NAME OF THE PRIMARY PAGING PARTITION (PAGDEV). IF THE PACK NAME IS 'PAGING', NUMBER OF FILE SYSTEM RECORDS IS READ FROM THE DISK RECORD AVAILABILITY TABLE (DSKRAT) AND THE PAGING RELOCATION CONSTANT (PAGREL) IS UPDATED ACCORDINGLY TO AVOID OVERWRITING THE FILE SYSTEM PORTION OF THE DISK. THIS PROCESS IS REPEATED FOR THE ALTERNATE PAGING DISK (IF ALTDEV IS SPECIFIED IN THE CONFIG COMMAND).

2. IF IT IS A SPLIT PARTITION (COMDEV=PAGDEV), OR THE PACK NAME OF THE PAGING DISK IS 'PAGING', AND A NON-EMPTY BADSPT FILE EXISTS, THE PRELOADER WILL READ THE INFORMATION FROM THE BADSPT FILE AND REASSIGN ALL THE LMAP PAGING RECORD INDEXES (WHICH ARE PREASSIGNED BY MAPGEN) TO AVOID USING BAD PAGING DISK TRACKS WHEN THE SYSTEM IS PRELOADED. THE BAD SPOTS INFORMATION IS ALSO PUT IN A COMMON BLOCK OF 16 WORDS (IT SUPPORTS A MAXIMUM OF 16 BAD SPOTS) WHICH GET PASSED TO AINIT.

9.9.2 AINIT

WHEN PAGING IS TURNED ON, THE PAGE MAP MUST CONTAIN THE CORRECT LMAP PAGING RECORD INDEXES. IF DEFECTIVE TRACKS EXIST IN THE PAGING DISK, THE PAGE MAPS IN THE COLD START MEMORY IMAGE *COLDS MAY NOT BE CORRECT. THEREFORE, AINIT NOW USES THE BAD SPOT INFORMATION PASSED ON BY THE PRELOADER TO REASSIGN THE LMAP PAGING RECORD INDEXES BEFORE PAGING IS TURNED ON. IT ALSO USES THE BAD SPOT INFORMATION WHEN ASSIGNING PAGING RECORD INDEXES FOR NON OPERATING SYSTEM PAGES TO AVOID DEFECTIVE TRACKS.

9.9.3 SEG14

AN ARRAY OF 16 WORDS NAMED BADREC IS ADDED FROM LOCATION '120 TO '137 WHICH IS USED BY AINIT TO ADDRESS THE BAD SPOT INFORMATION.

9.9.4 MAPGEN

THE COLD START MEMORY IMAGE *COLDS NOW STARTS AT '140 INSTEAD OF '120. THIS ENABLES THE PRELOADER TO PUT THE 16 WORDS BAD SPOTS INFORMATION AT LOCATION '120 TO '137 WHERE THEY DON'T GET OVERWRITTEN WHEN STARTUP AND INITIALIZATION PROCEDURE.

10 CORRECTED REVISION 16.2 PROBLEMS

THE FOLLOWING IS A LIST OF PROBLEMS, WHICH WERE CORRECTED AT REV 17 BUT NOT AT REV 16.2. WHERE APPLICABLE, TAR NUMBERS ARE INCLUDED.

10.1 PAGING DEVICE SIZE

FIXED CALCULATION OF PARTITION SIZE AND RECORD CAPACITY FOR DEVICES OF ALL TYPES. [TAR #22486]

10.2 OPEN, CLOSE COMMANDS

THESE COMMANDS NOW CHECK THAT THE UNIT NUMBER IS VALID.

10.3 COMOUTPUT COMMAND

FIXED SO THAT TREENAME PROCESSING REATTACHES TO THE HOME DIRECTORY. FIXED TO CHECK THAT FILE IS A SAM OR DAM FILE.

10.4 ABORT CONDITIONS

CERTAIN ABORT CONDITIONS NOW CANCEL THE USER'S WAIT FOR TERMINAL INPUT. [TAR #20004]

10.5 ATCH$$ PROBLEM

THE ATCH$$ CALL WOULD SOMETIMES RETURN FALSE 'NOT A UFD' ERROR CODES WHILE SEACHING ALL DISKS FOR A DIRECTORY WHOSE NAME WAS ALSO THE NAME OF A FILE IN AN MFD. [TAR #22461]

10.6 PHANT$ PROBLEM

IF THE UNIT ARGUMENT OF PHANT$ WAS SET TO 0, ON RETURN THE ARGUMENT WAS SET TO 6 BY PHANT$, EVEN IF THE ARGUMENT WAS A CONSTANT. [TAR #25758]

10.7 REMOTE LISTF OUTPUT

THE OUTPUT FROM A REMOTE LISTF WAS NOT BEING WRITTEN INTO THE USER'S COMOUTPUT FILE.

10.8 USER SEMAPHORE WAIT

A USER COULD MONOPOLIZE THE CPU BY CERTAIN USES OF THE SEM$WT CALL.

10.9 COMOUTPUT CONTROL ARGUMENT HANDLNG

"COMO TREENAME - CONTIN" WAS NOT BEING CORRECTLY HANDLED.

10.10 UII_PACKAGE

(THIS PACKAGE IS NOW PART OF THE OPERATING SYSTEM.) THE HANDLERS FOR ZMV AND ZMVD CAN NOW DO FULL-SEGMENT MOVES.

10.11 SHARE_COMMAND

THIS COMMAND COULD NOT HANDLE A FULL 64K-WORD FILE. [TAR #10555]

10.12 ERROR-CORRECTING_CODE_HANDLER

THE ECC HANDLER WOULD SOMETIMES PREMATURELY STOP THE RECORDING OF FURTHER ECC-CORRECTED MEMORY ERRORS.

10.13 DISK_PACK_NAME

WHEN ADDING A DISK WITH A PACK NAME OF A ODD LENGTH, AN UNINITIALIZED BYTE IN THE NAME SOMETIMES RESULTED. THE LOGPRT PROGRAM WOULD PUT A GARBAGE CHARACTER IN THE NAME. [TAR #24727]

10.14 COMINPUT_UNIT_NUMBER

THE COMINPUT COMMAND SOMETIMES DID NOT CORRECTLY OBTAIN ITS UNIT NUMBER ARGUMENT. [TAR #80697]

10.15 ASSIGNABLE_AMLC_LINES

IT IS NOW ASSURED THAT AN ATTEMPT TO UNASSIGN AN AMLC LINE WILL WAIT FOR ITS OUTPUT BUFFER TO EMPTY. [TAR #23415]

10.16 RING_0_GATES

IF THE NUMBER OF GATES IN SEG5 EXACTLY FILLED A PAGE, THE MODULE SEG5 FAILED TO ASSEMBLE. [TAR #25728]

10.17 SPLIT_DISK

LARGE-PARTITION SPLIT DISKS SOMETMES PRODUCED AN INVALID COMPUTATION OF AVAILABLE PAGING SPACE. [TAR #14541]

10.18 HIGH-PERFORMANCE_AMLC_CONFIGURATOR

MANY MISCELLANEOUS PROBLEMS FIXED. [TARS 24788, 23422]

10.19 AMLC_BUFFER_CHECK_DELAY

THE OUTPUT-BUFFER-FULL DELAY CYCLE IS NOW 0.5 SEC. [TAR #23421]

10.20 SYSTEM_TERMINAL_BAUD_RATE

THE BAUD RATE OF THE SYSTEM TERMINAL COULD BE IMPROPERLY SET IF NOT SPECIFIED EXPLICITLY VIA AN ASRATE CONFIG COMMAND. [TARS 80695, 14514]

11 CORRECTED REVISION 17.0 PROBLEMS

11.1 SRCH$$ PROBLEM

DO NOT ALLOW COMOUTPUT UNIT TO BE CLOSED BY SRCH$$ UNLESS CALLED FROM RING 0.

11.2 TP 2000 BUG

TO FIX FORMS BUG WHEN USING TELENET TP 2000 TIRS.

11.3 DPIX, SYNCHRONOUS COMMUNICATION

CODE CHANGED TO USE SEGMENT 17 POINTER INSTEAD OF SEG 0 POINTER. CODE REARRANGED TO SET FLAG CORRECTLY.

11.4 SMLC BUG FIX AND ENHANCEMENT

IMPROVEMENT TO DATA SET CONTROL HANDLING IN SYNCHRONOUS NETWORKS. FIX BUGS FOUND IN 16.6 FOR NCC. FIX BUGS FOUND IN SYNCHRONOUS NETWORKS OVER TELENET LINK.

11.5 COMINPUT

COMINPUT DOES NOT PICK UP THE UNIT NUMBER IF ITS POSITION IS GREATER THAN 3RD TOKEN.

11.6 DPIX

BUGS HAVE BEEN FIXED IN TM INVOLVING PROPER HANDLING OF 'TEST REQUEST' FUNCTION KEY.

11.7 NETWORKS

FIX BUGS IN RING STARTUP AND CLEANUP DIM CODE. ADD SUPPORT FOR 15 RING NODES. CHANGE THE NAME 'FARDIM' TO 'PNCDIM'. CHANGE LOAD SEQUENCE TO PUT UNWIRED NETWORK BUFFERS AT TOP OF SEGMENT.

11.8 MDLC

FIXED PROBLEMS IN TIMING SENSITIVE CODE IN SLCDIM. (1) SLCINI - WITH TWO CONTROLLERS AN INSTRUCTION MAY FAIL TO SKIP BECAUSE IT IS EXECUTED BEFORE THE CONTROLLER HAS FINISHED INITIALIZING FROM AN INSTRUCTION. (2) SLCOTP - AN OTP FUNCTION CODE INSTRUCTION MAY TAKE LONGER TO FINISH THAN 3 IRS INSTRUCTIONS.

11.9 PLPLIB AND TRIM

FIX PLP TRIM( ) BIF TO WORK CORRECTLY WHEN BIT CONTROL ARGUMET IS '00'B, SOULD RESULT IN INPUT STRING -- NOW RESULTS IN GARBAGE STRING.

11.10 I/O_WINDOWS

GTWNDO IS SUPPOSED TO BE WIRED (CONTAINS INH) BUT IS NOT.

11.11 NETWORKS

FIX BUG IN RLOGIN THAT DOESN T <QUIT> IF REMOTE TERMINAL BUFFER HAS >64 CHARACTERS IN IT.

11.12 DELETE_COMMAND

BUG HAS BEEN FIXED SO THAT DELETION OF A SEG DIRECTORY ON AN OLD PARTITION COULD
LEAD TO A RECURSIVE LOCK ATTEMPT ON TRNLOK (BETWEEN DELETE AND PRWF$$).MAKE SYSTEM HALT [TAP #11582]

11.13 RTNREC_PROBLEM

RTNREC WAS CALLING ERRPR$ WITH RATLO HELD = LOCK VIOLATION IF COMINPUT ON. MAKE SYSTEM HALT

11.14 WARM_START_CONFIGURATION

AMINIT WAS INCORRECTLY RESETTING THE AMLC INTERRUPT RATE DURING A WARM START. A FLAG WAS ADDED TO DETECT A COLD OR WARM START.

11.15 SMLC EMULATORS AND NETWORKS

CLEARER ERROR MESSAGE AT INITIALIZATION FOR RJE EMULATORS AND NETWORKS. PRORLEM HAS BEEN SOLVED SO THAT SAVING 2 CONTROLLERS CONFIGURED AND ONLY ONE PRESENT SO THAT THE ONE PRESENT IS STILL ENABLED.

11.16 GPASS$_PROBLEM

GPASS$ LEFT THE UNIT OPEN IF REQUESTED ENTRY AS NOT A UFD.

11.17 NETWORKS_PROBLEM

FIX BUG WHICH CHANGES THE SYSTEM ON WACK'S CORRECT COMMENTS.

11.18 DPIX,_3270

THE PROBLEM OF HANDLING A "DEVICE END" STATUS ON A GENERAL POLL

11.19 DPIX,_3270

THE INTERPRETATION OF IBM BUFFER ADDRESSES SO AS TO ADHERE EXACTLY TO IPM 3270 FUNCTIONALITY.

11.20 DAM_FILE,_DAM_INDEX

IF A DAM FILE IS OPENED BY MORE THAN ONE USER, AND ONE OF THE USER EPANDS THE FILE AND CAUSE THE DAM FILE TO EXTEND TO N+1 LEVEL INDEX FILE FROM THE PERVIOUS N LEVEL INDEX FILE THE INDEX

POINTER OF OTHER USERS TO THIS BECOMES INVALID.

11.21 SMLC_FOR_RJE

SPEED UP THE PASSING OF STATUS FROM THE SMLC DRIVER IN RING 0 TO THE RJE EMULATORS IN RING 3.

11.22 STACK_DUMP_UII,_ECC,_STOP$

1. DMSTK COMMAND SOMETIMES DISPLAYED A BAD OWNER LB.

2. COMMAND LEVEL'S STATIC MODE SWITCH IS NOW CLEAR AFTER STOP$ SIGNAL.

3. UII PACKAGE SAVES 1 REG. FOR 2 INSTRUCTIONS.

4. FCCU PAGE MAP OUT CODE USING BAD PARTITION TO PAGE MAPS. SEG4 HAS BEEN CHANGED TO USE PROPER PARTITION FOR THIS CASE.

11.23 NETWORKS

ADD SUPPORT FOR DATAPAC AND UNKNOWN PDN'S. MANY BUGS HAVE BEEN FIXED IN JPCF. PROBLEM FIXED WITH LONGER RING NFTS.

11.24 PHANTOM_PROBLEM

'PH LOCAL-FILE' WHILE HOME OR CURRENT UFD IS REMOTE GIVES SOMETIMES THE 'NO UFD ATTACHED' AND SOMTIMES CRASHES THE SYSTEM WHEN TRYING TO HASH REMOTELY.

11.25 ASSIGNING_AND_UNASSIGNING_DEVICES

IF AN ASSIGNED AMLC LINE WAS UNASSIGNED WITH A FULL OUTPUT BUFFER, DEVLOK WAS LOCKED FOR 20 SECS. PREVENTING ANY OTHER ASSIGN/UNASSIGNS OF DEVICES BY ANY USER.

11.26 MDLC_PROBLEM

INSTRUCTION TO INITIALIZE THE CONTROLLER INSERTED BEFORE INA DEVICE ID INSTRUCTION.

11.27 GSG$RA_BUG_FIX

SEGMENT DIRECTORY ENTRY NUMBER IS RETURNED INCORRECTLY AS: (CORRECT ENTRY #) + (INT (ENTRY #/(# ENTRIES/RECORD))). THIS CAN CAUSE BAD INFORMATION TO BE RETURNED BY GPATH$, AND THE 'STATUS UNITS' COMMAND IF TREENAMES CONTAIN SEGMENT ENTRY NUMBERS.

11.28 STACK_OVERFLOW_PROBLEM

FIX RING ZERO STACK OVERFLOW PROBLEM.

TABLE OF CONTENTS

```
*****************************************************************************
*****************************************************************************
*
*
*
*      JJJ    III    M    M   M    M   Y    Y
*        J      I    MM  MM   MM  MM   Y    Y
*        J      I    M M M    M M M     Y Y
*        J      I    M   M    M   M      Y
*   J    J      I    M   M    M   M      Y
*   J    J      I    M   M    M   M      Y
*    JJ       III   M   M    M   M      Y
*
*
*
*
*   RRRR    BBBB     AAA    TTTTT    CCC    H   H     1
*   R   R   B   B   A   A     T     C   C   H   H    11
*   R   R   B   B   A   A     T     C       H   H     1
*   RRRR    BBBB    AAAAA     T     C       HHHHH     1
*   R R     B   B   A   A     T     C       H   H     1
*   R  R    B   B   A   A     T     C   C   H   H     1
*   R   R   BBBB    A   A     T      CCC    H   H   111
*
*
*
*****************************************************************************
*****************************************************************************
```

REVISION 17 BATCH FOR THE CX USER

DATE: APRIL 12, 1979

TO:

FROM:

SUBJECT: REVISION 17 BATCH FOR THE CX USER

REFERENCE:

ABSTRACT

PRIME'S NEW BATCH OFFERING FOR REVISION 17 REPRESENTS A LARGE INCREASE IN FUNCTIONALITY, PERFORMANCE AND EASE OF USE OVER THE OLD CX SUBSYSTEM. THIS DOCUMENT WILL HELP THE CX USER CONVERT TO REVISION 17 BATCH RELATIVELY EASILY.

CX COMMAND FILES ARE UPWARDLY COMPATIBLE WITH BATCH COMMAND FILES - NO CHANGES NEED BE MADE TO OLD CX COMMAND FILES SO THAT THEY MAY BE USED FOR BATCH SUBMISSION. HOWEVER, THE COMMAND AND OPTION NAMES HAVE CHANGED AND THERE ARE NEW REQUIREMENTS AND FREEDOMS AVAILABLE IN THE NEW BATCH.

A. SUBMISSION OF JOBS

TO SUBMIT A JOB TO THE NEW BATCH SUBSYSTEM, THE JOB COMMAND IS INVOKED. THIS COMMAND IS FOLLOWED BY THE NAME OF THE COMMAND FILE TO BE SUBMITTED AND A LIST OF OPTIONS, AS FOLLOWS:

```
JOB TREENAME [-OPTIONS]
```

THE CX COMMAND WILL ACCEPT TWO OPTIONS WHILE SUBMITTING A JOB. THESE ARE -PRIORITY AND -CPULIMIT. THE -PRIORITY PARAMETER AFFECTS THE QUEUE PRIORITY AND THE EXECUTE-TIME PRIORITY OF THE JOB, AND THE -CPULIMIT PARAMETER BECOMES THE CPU TIME LIMIT ON THE JOB WHEN IT IS EXECUTED IN SECONDS. IF THE JOB EXCEEDS THAT LIMIT, IT WILL BE LOGGED OUT WITH THE MESSAGE CPU TIME LIMIT EXCEEDED.

THE JOB REPLACEMENTS FOR THOSE OPTIONS ARE -PRIORITY AND -CPTIME. THERE ARE NO ABBREVIATIONS FOR THESE OPTIONS. THE -CPTIME OPTION ACCEPTS THE EXACT SAME PARAMETER AS THE CX -CPULIMIT OPTION DOES, INCLUDING THE ABILITY TO SPECIFY NO LIMIT AS "NONE". THE MEANING AND RANGE OF -PRIORITY, HOWEVER, HAS CHANGED. WHILE IT STILL CONTROLS THE QUEUE PRIORITY OF THE JOB (I.E., THE HIGHER THE PRIORITY, THE SOONER IT IS SCHEDULED FOR EXECUTION), IT NO LONGER HAS ANY EFFECT ON THE EXECUTION-TIME PRIORITY. ALSO, THE RANGE OF THE PARAMETER IS FROM 0 TO 9 INSTEAD OF 0 TO 7.

NOTE: THE -CPTIME OPTION EXPECTS AN INTEGER*4 NUMBER AS DOES THE CX -CPULIMIT OPTION.

WHEN THE JOB PROGRAM SUBMITS A COMMAND FILE, IT FINDS OUT WHERE IT IS ATTACHED TO AS A HOME UFD AND WRITES THIS INTO THE QUEUE DATA ENTRY. IF THIS PATHNAME (WHICH IS STRIPPED OF PASSWORDS) CANNOT BE ATTACHED TO (BECAUSE ONE OF THE UFDS INVOLVED HAS TWO PASSWORDS), THEN A HOME UFD MUST BE EXPLICITLY SPECIFIED ON THE COMMAND LINE, USING THE -HOME OPTION. THE SPECIFICATION IS ESSENTIALLY A NO-OP BECAUSE ALL LEGAL CX FILES HAVE AN ATTACH COMMAND AS THEIR FIRST EXECUTABLE LINE; THEREFORE, ANY SUCCESSFUL ATTACH POINT MAY BE USED, EXAMPLE:

```
JOB C_COMPILE -HOME CMDNCO
```

HOWEVER, THE JOB COMMAND DOES MAKE SURE THAT THE HOME UFD SPECIFIED ISN'T A NULL SPECIFICATION OR A RELATIVE TREENAME (BEGINNING WITH '*>'), SO ANY LEGAL UFDNAME MUST BE PUT IN AS THE HOME UFD (REGARDLESS OF THE ACTUAL DESTINATION OF THE JOB). AS LONG AS THE ATTACH TO THAT UFD IS SUCCESFUL, WHETHER AS AN OWNER OR NON-OWNER, THE JOB SHOULD RUN SMOOTHLY BECAUSE AFTER THE ORIGINAL ATTACH, THE COMMAND FILE ITSELF WILL ATTACH TO THE REAL HOME UFD.

HOWEVER, THOSE USERS WHO HAD CX COMMAND FILES THAT WERE INTENDED TO RUN IN ANY UFD WILL FIND THAT IT IS MUCH EASIER TO SPECIFY THE HOME UFD ON THE COMMAND LINE RATHER THAN EDITING THE COMMAND FILE AND CHANGING THE ATTACH COMMAND EACH TIME. SIMPLY REMOVE THE FIRST ATTACH COMMAND FROM THE COMMAND FILE. IT WILL STILL BE A LEGAL COMMAND FILE FOR REV. 17 BATCH, ALTHOUGH CX WILL NO LONGER ACCEPT IT.

B. DISPLAYING STATUS OF QUEUES

THE FUNCTIONALITY AVAILABLE BY USING THE CX STATUS OPTIONS -A (ALL), -Q (QJEUE), -P (PERSONAL) AND -SNN (STATUS OF PARTICULAR JOB) IS PRESENT IN THE JOB COMMAND, WITH THE FOLLOWING EXCEPTIONS:

1. NO USER MAY DISPLAY THE STATUS OF ANY JOBS NOT BELONGING TO THAT USER, UNLESS THAT USER IS SYSTEM, IN WHICH CASE ALL JOBS ARE AVAILABLE FOR DISPLAYING.

2. THE "INTERNAL" NAME FORMAT OF JOBS HAS CHANGED. IN CX, THE INTERNAL NAME WAS OUTPUT TO THE USER AS CX##NN, BUT THE USER WOULD ONLY INPUT THE NN PART TO REFER TO THE JOB. IN REV. 17 BATCH, THE INTERNAL NAME IS OF THE FORMAT #SNNNN FOR BOTH INPUT AND OUTPUT, WHERE S IS A ONE-CHARACTER INTERNAL NAME FOR THE QUEUE AND NNNN IS A JOB ID WITHIN THE QUEUE.

3. REV. 17 BATCH ALLOWS USERS TO REFER TO JOBS USING AN EXTERNAL NAME. THIS EXTERNAL NAME IS THE LAST COMPONENT OF THE TREENAME USED TO SUBMIT THE JOB (I.E. FOR A JOB SUBMISSION OF ALPHA>BETA>GAMMA, THE EXTERNAL NAME WOULD BE GAMMA). ALL APPROPRIATE JOBS WITH THE SPECIFIED EXTERNAL NAME WILL BE DISPLAYED.

AN "ACTIVE" JOB IS A JOB THAT IS WAITING FOR EXECUTION, HAS BEEN HELD BY AN OPERATOR, OR IS EXECUTING.

THE TWO OPTIONS TO THE JOB COMMAND TO OBTAIN INFORMATION ON JOBS ARE -STATUS AND -DISPLAY. -STATUS RETURNS THE DISPOSITION OF THE SPECIFIED JOB(S), WHETHER WAITING, EXECUTING, HELD, COMPLETED, ABORTED, OR CANCELLED, AND THE QUEUE TO WHICH THE JOB(S) WAS(WERE) SUBMITTED. THE OUTPUT WILL BE TABULAR. -DISPLAY RETURNS ALL THE INFORMATION ABOUT THE JOB(S). THE INFORMATION DISPLAYED IS THE SUBMISSION DATE, THE INITIATION DATE, THE COMPLETION DATE AND THE NUMBER OF TIMES EXECUTED (IF APPLICABLE), THE FILE UNIT THAT THE COMMAND FILE WILL BE OPENED ON, THE QUEUE PRIORITY, AND THE CPU/ELAPSED TIME LIMITS. IF ANY ACCOUNTING INFORMATION WAS SPECIFIED, THIS TOO WILL BE PRINTED. THE OUTPUT FROM THIS COMMAND WILL BE CONSISTENT WHETHER OR NOT A SPECIFIC JOB NAME WAS SPECIFIED.

THE FORMATS OF THE COMMAND LINES TO USE THESE OPTIONS ARE AS FOLLOWS:

```
JOB JOBID -STATUS          /* STATUS OF ACTIVE JOBS NAMED "JOBID".
JOB JOBID -STATUS TODAY    /* STATUS OF TODAY'S JOBS NAMED "JOBID".
JOB JOBID -STATUS ALL      /* STATUS OF ALL JOBS NAMED "JOBID".

JOB -STATUS                /* STATUS OF ALL ACTIVE JOBS.
JOB -STATUS TODAY          /* STATUS OF TODAY'S JOBS.
JOB -STATUS ALL            /* STATUS OF ALL JOBS.
```

AS EXPLAINED ABOVE, AN "ACTIVE" JOB IS ANY JOB THAT IS EITHER WAITING, HELD, OR EXECUTING. THEREFORE, THE COMMAND:

JOB -STATUS

WILL NOT OUTPUT THE STATUS OF ANY COMPLETED, ABORTED OR CANCELLED JOBS. ALSO, THE COMMAND:

JOB C_TEST -STATUS

WILL ONLY REFERENCE ACTIVE JOBS WITH THE NAME "C_TEST".

FOR MOST USERS, THE OUTPUT IS LIMITED TO JOBS BELONGING TO THE USER MAKING THE REQUEST...IF THE USER IS LOGGED IN AS SYSTEM, ALL JOBS WILL BE DISPLAYED.

THE "JOBID" IS COMPARED TO BOTH THE INTERNAL AND EXTERNAL NAMES OF JOBS TO SEE IF THEY SHOULD BE OUTPUT, UNLESS THE USER IS LOGGED IN AS SYSTEM, IN WHICH CASE THE INTERNAL NAME IS ALWAYS COMPARED, BUT THE EXTERNAL NAME IS ONLY COMPARED IF THE JOB IS OWNED BY A USER LOGGED IN AS SYSTEM.

"TODAY" REFERS TO ANY JOB THAT WAS SUBMITTED, INITIATED, ABORTED, COMPLETED OR CANCELLED ON THE SAME DATE THAT THE REQUEST WAS MADE.

ALSO, ANY OCCURRENCE OF "-STATUS" MAY BE REPLACED BY "-DISPLAY" TO OBTAIN MORE INFORMATION ON THE DISPLAYED JOBS.

C. CONTROL_OVER_JOBS

THE CX OPTION -DNN (DROP A WAITING JOB) IS IMPLEMENTED IN REV. 17 BATCH AS FOLLOWS:

JOB JOBID -CANCEL

WHERE JOBID IS THE INTERNAL OR EXTERNAL NAME OF THE JOB. IF THE JOB IS EXECUTING WHEN THIS COMMAND IS ENTERED, THE JOB WILL NOT BE CANCELLED, BUT IT WILL BE FLAGGED AS BEING UNRESTARTABLE.

ABORTING THE JOB CAN BE DONE BY EITHER FORCE-LOGGING THE JOB OUT USING THE LOGOUT COMMAND (E.G., LOGOUT -58), OR USING THE -ABORT OPTION AS FOLLOWS:

JOB JOBID -ABORT

THIS WILL EITHER DO THE FORCE LOGOUT IF THE JOB IS EXECUTING, OR CANCEL THE JOB IF IT IS WAITING OR HELD (THE SAME WAY THAT -CANCEL DOES IT).

IF IT IS DESIRED TO RESTART AN EXECUTING JOB, USE THE -RESTART OPTION AS FOLLOWS:

```
JOB JOBID -RESTART
```

THIS WILL ONLY WORK ON EXECUTING JOBS. IT WILL UPDATE THE QUEUE INFORMATION ON THE PARTICULAR JOB TO INDICATE THAT IT IS READY FOR EXECUTION, THEN FORCE-LOGOUT THE CURRENTLY RUNNING JOB. NOTE THAT IF THAT JOB WAS CANCELLED WHILE IT WAS EXECUTING, IT WILL BE FLAGGED AS BEING UNRESTARTABLE, AND -RESTART WILL THEREFORE ONLY ABORT IT. THIS IS ALSO THE CASE IF THE JOB WAS SUBMITTED WITH A -RESTART NO OPTION (EXPLAINED IN PE-T-584, SECTION 5).

D. SUMMARY AND EXAMPLES

THE FOLLOWING EXAMPLE CX COMMAND LINES ARE SHOWN WITH THEIR JOB COUNTERPARTS:

CX C_COMPILE	JOB C_COMPILE
CX -S23	JOB C_COMPILE -DISPLAY ALL
CX -Q	JOB -STATUS
CX -P	JOB -DISPLAY ALL
CX C_GO -CPULIMIT 12	JOB C_GO -CPTIME 12
CX C_GO -PRIORITY 3	JOB C_GO -PRIORITY 3 -HOME CMDNCD
CX -D14	JOB C_GO -CANCEL

```
**********************************************************************************
**********************************************************************************
**
**
**
**      JJJ    III    M    M   M    M   Y    Y
**       J      I     MM  MM   MM  MM   Y    Y
**       J      I     M M M    M M M     Y Y
**       J      I     M M M    M M M      Y
**   J   J      I     M    M   M    M     Y
**   J   J      I     M    M   M    M     Y
**    JJ       III    M    M   M    M     Y
**
**
**
**   RRRR    BBBB     AAA    TTTTT    CCC    H    H   222
**   R   R   B   B   A   A     T     C   C   H    H  2   2
**   R   R   B   B   A   A     T     C       H    H      2
**   RRRR    BBBB    AAAAA     T     C       HHHHH     2
**   R R     B   B   A   A     T     C       H    H    2
**   R  R    B   B   A   A     T     C   C   H    H   2
**   R   R   BBBB    A   A     T      CCC    H    H  22222
**
**
**
**********************************************************************************
**********************************************************************************
```

AN OFF-LINE UTILITY FOR BATCH (FIXBAT)

DATE: APRIL 12, 1979

TO:

FROM:

SUBJECT: AN OFF-LINE UTILITY FOR BATCH (FIXBAT)

REFERENCE:

ABSTRACT

THE BATCH SUBSYSTEM FOR REVISION 17 DYNAMICALLY INCREASES QUEUE FILE SIZES AS NEW JOBS ARE SUBMITTED TO IT. HOWEVER, NO AUTOMATIC MECHANISM EXISTS IN THE SUBSYSTEM TO RECLAIM DISK SPACE BY DELETING VERY OLD BATCH JOB DEFINITIONS.

A PROGRAM NAMED FIXBAT (FOR FIX BATCH) HAS BEEN CREATED TO RELIEVE THIS PROBLEM. IT IS AN OFF-LINE UTILITY TO BE RUN BY THE SYSTEM ADMINISTRATOR WHEN THE BATCH DATABASE BEGINS TO TAKE UP MORE DISK SPACE THAN NECESSARY, AND IT IS ALSO AUTOMATICALLY RUN EVERY TIME THE BATCH MONITOR IS STARTED UP, ALTHOUGH AT THIS POINT IT WILL NOT RECLAIM DISK SPACE BY DELETING OLD BATCH JOBS, UNLESS OTHERWISE SPECIFIED.

WHILE THIS DOES NOT CAUSE BATCH TO DYNAMICALLY RECLAIM DISK SPACE WHILE THE BATCH MONITOR IS RUNNING, IT CAN BE SET UP SUCH THAT IT RECLAIMS IT AT EVERY SYSTEM COLD START.

THIS DOCUMENT EXPLAINS HOW TO USE FIXBAT AND IS INTENDED FOR USE BY THE SYSTEM ADMINISTRATOR AND OPERATORS ONLY, AS USERS ARE UNABLE TO RUN IT.

I. GENERAL DESCRIPTION OF FIXBAT

FIXBAT IS A RESUMABLE PROGRAM THAT RESIDES IN THE BATCHQ UFD UNDER THE NAME "*FIXBAT" (SEE PE-T-571). IT SERVES THREE PURPOSES:

1) TO DELETE OLD BATCH JOB ENTRIES THAT ARE AT LEAST A SPECIFIED AGE IN DAYS OLD.

2) TO FIX ANY AND ALL BROKEN POINTERS WITHIN THE QUEUE FILES, AND ALSO RECONSTRUCT THE QUEUE AND EXECUT FILES AS MUCH AS POSSIBLE FROM THE INFORMATION IN THE QUEUE FILES (SEE PE-T-571).

3) TO HANDLE THE START-UP PROTOCOL FOR THE BATCH MONITOR, MAKING SURE THAT THE DATABASE IS VALID BEFORE STARTING IT UP, AND, IF NECESSARY OR SPECIFIED BY THE ADMINISTRATOR, CLEANING UP THE DATABASE AND PERFORMING THE RECONSTRUCTION ACTIVITIES DESCRIBED IN PART 2 ABOVE.

A. STARTING UP FIXBAT

BEFORE FIXBAT CAN BE RUN, THE BATCH MONITOR MUST BE LOGGED OUT. IF IT IS NOT, AND FIXBAT IS RUN, IT WILL RETURN WITH THE ERROR MESSAGE "BATCH PROCESS STILL IN PROGRESS".

TO START UP FIXBAT, LOG INTO THE SYSTEM UFD, ATTACH TO THE BATCHQ UFD WITH THE OWNER PASSWORD, AND RESUME *FIXBAT. THE FOLLOWING ARE LEGAL OPTIONS:

```
-DAYS <N>
-QUIET
-STARTUP <ACTION>
```

FOR INTERACTIVE USE, THE -STARTUP OPTION SHOULD NOT BE SPECIFIED, AS IT TELLS FIXBAT TO HANDLE THE STARTING UP OF THE BATCH MONITOR. IF THIS OPEPATION IS ATTEMPTED, THE USER WILL RECEIVE THE ERROR MESSAGE "UNABLE TO PROCESS BATCH JOBS".

> NOTE: *FIXBAT DOES NOT NECESSARILY HAVE TO BE RUN WHILE ATTACHED TO BATCHQ AS HOME OR CURRENT. IF IT IS RESUMEABLE, IT WILL KNOW THE BATCHQ PASSWORD AND ATTACH TO BATCHQ ITSELF. IF THE -STARTUP OPTION WAS SPECIFIED, IT WILL ATTACH TO BATCHQ AS THE HOME UFD, OTHERWISE IT WILL NOT.
>
> THIS MEANS THAT THE USER WHO RUNS FIXBAT INTERACTIVELY WILL NOT LOSE HIS OR HER HOME ATTACH POINT. THE DRAWBACK TO THIS IS THAT FIXBAT TENDS BE A BIT SLOWER, SINCE IT HAS TO RE-ATTACH TO BATCHQ AT EACH OPERATION, WHEREAS IF IT HAD BATCHQ AS A HOME UFD, IT WOULD ONLY DO AN ATTACH TO HOME OPERATION.

THE -DAYS OPTION TAKES AN INTEGER ARGUMENT BETWEEN 1 AND 60, INCLUSIVE. THIS VALUE SPECIFIES THE NUMBER OF ABSOLUTE DAYS OLD THAT A JOB MUST BE TO BE REMOVED FROM THE QUEUE FILE. IF THIS OPTION IS NOT SPECIFIED, NO JOBS WILL BE REMOVED FROM THE QUEUE FILE.

THE -QUIET OPTION IS ONLY USEFUL WHEN SPECIFIED ALONG WITH THE -DAYS OPTION. NORMALLY, WHEN A JOB IS REMOVED FROM THE QUEUE FILE BECAUSE IT IS OLD ENOUGH, THE JOB INFORMATION WILL BE OUTPUT TO THE TERMINAL, IN THE SAME FORMAT AS THE JOB -DISPLAY COMMAND. IF -QUIET IS SPECIFIED, HOWEVER, THIS WILL NOT HAPPEN.

THE -STARTUP OPTION TELLS FIXBAT TO HANDLE THE STARTING UP OF THE BATCH MONITOR. IT REQUIRES AN ARGUMENT OF EITHER "SAVE", "SPOOL", "DELETE" OR "NOLOG". THIS ARGUMENT SPECIFIES WHAT FIXBAT SHOULD DO AS FAR AS RUNNING A COMOUTPUT FILE FOR THE BATCH MONITOR. IN PARTICULAR, IT SPECIFIES WHAT TO DO WITH THE OLD COMOUTPUT FILE.

B. WHEN DOES FIXBAT ACTUALLY FIX THE DATABASE?

FIXBAT DOES NOT ACTUALLY INDEPENDENTLY FIX ANY POINTERS, COUNTERS, ETC. AND RECLAIM DISK SPACE BY DELETING JOB ENTRIES. IT DOES THIS ALL AT ONCE. IF THE -DAYS ARGUMENT HAS NOT BEEN SPECIFIED, IT WILL NOT DELETE JOB ENTRIES, BUT IT MAY STILL MANAGE TO RECLAIM SOME DISK SPACE BY DELETING TEMPORARY FILES LEFT AROUND BY THE BATCH MONITOR.

WHEN FIXBAT STARTS SWEEPING THROUGH THE BATCHQ UFD AND THE CIFILE AND Q.CTRL SUB-UFDS (SEE PE-T-571), IT FIRST OUTPUTS THE MESSAGE "FIXING DATABASE". THIS MEANS THAT FIXBAT HAS DECIDED TO PERFORM THE CLEAN-UP OPERATION.

HOWEVER, IT IS POSSIBLE FOR FIXBAT TO DECIDE NOT TO PERFORM THE CLEANUP OPERATION. IT MAKES THIS DECISION IF IT THINKS THAT THE DATABASE IS PERFECTLY VALID (AND THEREFORE IT DOES NOT NEED TO FIX POINTERS, COUNTERS, ETC.), AND IT KNOWS IT DOES NOT HAVE TO DELETE OLD JOB ENTRIES.

IT MAKES THIS DECISION AS FOLLOWS:

IF FIXBAT WAS INVOKED WITH THE -STARTUP OPTION, THEN IT IS NOT BEING RUN INTERACTIVELY, AND THEREFORE IT MAY DECIDE NOT TO FIX THE DATABASE. IF IT HAD BEEN RUN INTERACTIVELY, THEN IT WOULD ALWAYS FIX THE DATABASE BECAUSE THAT WOULD BE THE ONLY PURPOSE IN RUNNING IT.

IF FIXBAT WAS NOT INVOKED WITH THE -DAYS OPTION, THEN IT DOES NOT HAVE TO DELETE ANY OLD JOB ENTRIES. THEREFORE, IT MAY DECIDE NOT TO FIX THE DATABASE. IF IT HAD BEEN INVOKED WITH -DAYS SPECIFIED, IT WOULD HAVE TO FIX THE DATABASE, SINCE IT WOULD HAVE NO OTHER WAY OF KNOWING WHETHER THERE WERE ANY

JOBS TO BE DELETED.

IF FIXBAT SEES THAT THE DATABASE IS STILL VALID, THEN IT MAY NOT HAVE TO FIX IT. IT CHECKS ON THE EXISTENCE OF THE "VALID." FILE IN BATCHQ TO DETERMINE THIS. IF THIS FILE IS MISSING, THEN THE DATABASE WAS EITHER INVALIDATED OR NEVER INITIALIZED, AND IN BOTH THOSE CASES, THE DATABASE SHOULD BE FIXED.

IF FIXBAT SEES THAT THE BATCH MONITOR WAS LOGGED OUT WITH THE "BATCH SYSTEM -STOP" COMMAND, THEN IT KNOWS THAT THE MONITOR LOGGED ITSELF OUT, GRACEFULLY. IN THAT CASE, IT MAY NOT HAVE TO FIX THE DATABASE. HOWEVER, IF THE FILE "MON.ST" DOES NOT EXIST IN BATCHQ, WHICH INDICATES THAT THE MONITOR WAS NOT GRACEFULLY LOGGED OUT, THEN EITHER THE MONITOR WAS FORCE-LOGGED OUT, ABORTED, OR THE SYSTEM WAS SHUT DOWN OR CRASHED. IN ANY OF THESE CASES, FIXBAT SHOULD FIX THE DATABASE TO GUARANTEE THAT NO POINTER MISMATCHES OCCUR AND TO FIX ANY COUNTERS THAT WERE BROKEN.

THEREFORE, THE BIG IF IS DECIDED AS FOLLOWS:

IF -STARTUP WAS SPECIFIED, AND
-DAYS WAS NOT, AND
THE DATABASE WAS VALID, AND
THE MONITOR DID NOT ABORT, THEN

THE DATABASE WILL NOT BE FIXED. OTHERWISE, IT WILL BE FIXED.

II. STARTING UP THE BATCH MONITOR WITH FIXBAT

THE -STARTUP OPTION SPECIFIES HOW THE BATCH MONITOR SHOULD BE STARTED UP RELATING TO LOG TRAILING (COMOUTPUT) OF ITS ACTIVITIES.

THIS OUTPUT IS GENERALLY ONLY USEFUL WHEN THE BATCH MONITOR ABORTS, ALTHOUGH SOME ADMINISTRATORS MAY FIND IT USEFUL FOR OTHER THINGS.

THE FILE BATCHQ>PH_GO IS WHAT IS USED TO START UP THE BATCH MONITOR (SEE PE-T-571). IT ACTUALLY RUNS *FIXBAT BEFORE IT RUNS *MONITR.

IN FACT, BECAUSE THE BATCHQ UFD MAY HAVE AN OWNER PASSWORD, AND BECAUSE THE COMMAND TO START UP THE BATCH MONITOR FROM THE SYSTEM CONSOLE IS "PHANTOM BATCHQ>PH_GO", BOTH THE PH_GO AND *FIXBAT PROGRAMS MUST HAVE READ ACCESS FOR NON-OWNERS (I.E. "7 1" PROTECTION), AND THE PH_GO FILE REFERENCES *FIXBAT USING THE TREENAME "BATCHQ>*FIXBAT", SINCE IT CANNOT BE GUARANTEED THAT IT IS ATTACHED TO BATCHQ AS CURRENT UFD OR HOME UFD. THIS IS BECAUSE THE PHANTOM COMMAND WOULD SET THE CURRENT UFD OF THE PHANTOM TO BATCHQ, BUT THE HOME UFD WOULD PROBABLY END UP AS CMDNCO. THEN, WHEN THE EXTERNAL LOGIN PROGRAM WAS RUN, IF IT REFERENCED ANY SYSTEM ACCOUNTING FILES, THE CURRENT ATTACH POINT WOULD BE LOST.

THE FIXBAT PROGRAM KNOWS THE BATCHQ PASSWORD, AND WHEN IT SEES THE -STARTUP OPTION, IT ATTACHES TO BATCHQ AS HOME UFD AND AS AN OWNER.

THE PH_GO FILE IS RELEASED ON THE MASTER DISK WITH THE RUNNING OF FIXBAT DONE AS FOLLOWS:

```
RESUME BATCHQ>*FIXBAT -STARTUP SAVE
```

THIS TELLS FIXBAT TO STARTUP THE BATCH MONITOR, AND SAVE THE PREVIOUS COMOUTPUT FILE BEFORE TURNING ON THE NEW ONE. IT ALSO TELLS FIXBAT NOT TO DELETE ANY JOB ENTRIES. FIXBAT MAY OR MAY NOT ACTUALLY FIX THE DATABASE IN THIS CASE. USUALLY, IT WON'T, IF THE BATCH MONITOR IS LOGGED OUT WITH "BATCH SYSTEM -STOP" BEFORE THE SYSTEM IS SHUT DOWN.

THE FOLLOWING ARE LEGITIMATE -STARTUP ARGUMENTS:

```
SAVE    /* SAVE OLD COMOUTPUT FILE.
SPOOL   /* SPOOL OLD COMOUTPUT FILE.
DELETE  /* DELETE OLD COMOUTPUT FILE.
NOLOG   /* DO NOT TURN ON COMOUTPUT.
```

THE SAVE, SPOOL, AND DELETE ARGUMENTS ALL PERFORM THE INDICATED REQUEST, THEN TURN ON COMOUTPUT TO A FILE NAMED O_LOG BEFORE EVEN MAKING THE DECISION TO FIX THE DATABASE. THE NOLOG OPTION CAUSES -STARTUP TO PERFORM AS IF NO -STARTUP WAS SPECIFIED; NO COMOUTPUT IS TURNED ON, AND THE OLD O_LOG FILE IS NOT TOUCHED.

A. THE SAVE SPECIFIER

IF -STARTUP SAVE IS SPECIFIED, A FILE NAMED OLDLOG IS DELETED, AND A "NOT FOUND" ERROR HERE IS IGNORED. THEN, THE FILE 0_LOG IS CNAME'D TO OLDLOG, AND AGAIN, A "NOT FOUND" ERROR IS IGNORED. THEN A COMO$$ CALL IS MADE TO TURN ON COMOUTPUT TO THE FILE 0_LOG, TRUNCATING THE FILE AFTER OPENING IT.

AS A RESULT, THE "STACK" OF LOG TRAILS IS 1 DEEP, I.E. THERE ARE USUALLY 2 LOG TRAILS AROUND, THE CURRENT ONE IN USE (0_LOG), AND THE LAST ONE GENERATED (OLDLOG).

B. THE SPOOL SPECIFIER

IF -STARTUP SPOOL IS SPECIFIED, A FILE NAMED 0_LOG IS SPOOLED UNDER THE FALSE NAME OF "BATCH.LOG" (SIMILAR TO THE -AS OPTION ON THE SPOOL COMMAND LINE) AND THEN DELETED, AND IN BOTH CASES, THE "NOT FOUND" EPROR IS IGNORED. THEN COMOUTPUT IS TURNED ON TO THE FILE 0_LOG, TRUNCATING THE FILE AFTER OPENING IT.

THEREFORE, THERE WILL GENERALLY ONLY BE ONE LOG TRAIL IN BATCHQ (THE ONE THAT IS IN USE, 0_LOG), AND ONE OR MORE WAITING IN THE SPOOLQ UFD (PRTXXX) SHOWING UP AS "BATCH.LOG" IN A SPOOL -LIST OUTPUT.

C. THE DELETE SPECIFIER

IF -STARTUP DELETE IS SPECIFIED, THE ONLY ACTION TAKEN IS TO TURN COMOUTPUT ON TO THE FILE 0_LOG, TRUNCATING THE FILE AFTER OPENING IT. THE ACTION OF TRUNCATING THE FILE (WHICH IS AN OPTIONAL BIT IN THE CALL TO COMO$$) CAUSES THE OLD FILE TO BE EFFECTIVELY "DELETED".

THEREFORE, THERE IS NEVER MORE THAN ONE LOG TRAIL IN BATCHQ, AND IT IS THE ONE THAT IS IN USE.

THIS METHOD IS NOT RECOMMENDED, SINCE IT PREVENTS ANALYZATION OF A PROBLEM THAT OCCURRED BEFORE THE LAST SYSTEM COLD-START. HOWEVER, IT IS PREFERRED TO -STARTUP NOLOG ON A SYSTEM WITH LOW DISK SPACE, SINCE IT AT LEAST GENERATES AN OUTPUT, USEFUL FOR ANALYZING THE MOST CURRENT CAUSE OF AN ERROR.

D. THE NOLOG SPECIFIER

IF -STARTUP NOLOG IS SPECIFIED, NO ACTION IS TAKEN RELATING TO COMOUTPUT FILES.

THIS SPECIFIER IS NOT RECOMMENDED SINCE IT PREVENTS ANALYZATION OF THE BATCH MONITOR'S BEHAVIOR AT APPROXIMATELY THE TIME OF A BATCH PROBLEM (SUCH AS A DATABASE BEING BROKEN, OR A JOB DISAPPEARING, ETC.).

IN SUMMARY, IT SHOULD BE NOTED THAT CHANGING THE PH_GO -STARTUP SPECIFIER CAN SOMETIMES RESULT IN FILES (SUCH AS OLDLOG OR O_LOG) BEING LEFT AROUND, OR RECENT LOG FILES BEING DELETED. IT IS RECOMMENDED THAT WHENEVER A CHANGE LIKE THIS IS MADE, THAT THE SYSTEM ADMINISTRATOR MAKE SURE THAT ANY EXISTING FILES ARE TAKEN CARE OF BEFORE THE BATCH MONITOR IS STARTED UP WITH THE NEW PH_GO FILE.

THE FILES TO LOOK FOR ARE O_LOG AND OLDLOG IN THE BATCHQ UFD, AND THEIR DISPOSITION IS UP TO THE ADMINISTRATOR (SPOOL, SAVE OR DELETE THEM).

THE O_LOG FILE, WHEN GENERATED, CONTAINS A VERBOSE FIRST LINE, SUITABLE AS A HEADER LINE FOR A SPOOL FILE. IT CONTAINS THE TIME OF DAY, THE DATE (AND THE DAY), AND THE FIXBAT REVISION NUMBER. AFTER THAT COMES TWO BLANK LINES, THEN COMES THE LOG TRAIL OF WHAT FIXBAT DID.

THIS MAY INCLUDE "COMMENTS" LIKE "FIXING DATABASE", "DELETED T$0001", "DELETED C00041", ETC., WHICH ARE NOT ERRORS, SIMPLY NOTIFICATIONS THAT CERTAIN FILES DEEMED USELESS BY FIXBAT WERE DELETED.

IT MAY ALSO INCLUDE THE INFORMATION ON ANY JOBS THAT WERE DELETED FROM THE QUEUES (UNLESS -QUIET WAS SPECIFIED). IN ADDITION, ANY STRANGE FILE FORMATS (SUCH AS PARTIAL QUEUE ENTRIES) ARE NOTED HERE.

IF FIXBAT ABORTS, THE REASON SHOULD BE FAIRLY EVIDENT BY LOOKING AT THE LOG FILE. USUALLY, DELETING THE OFFENDING FILE AND RESTARTING THE BATCH MONITOR (AND THEREFORE FIXBAT) IS THE FASTEST WAY TO FIX ANY PROBLEMS.

AFTER FIXBAT RUNS, THE PH_GO FILE WILL THEN RESUME *MONITR, AND THE MONITOR REV NUMBER WILL BE TYPED OUT, FOLLOWED BY A LOG TRAIL OF ITS ACTIVITIES. THERE IS NO PARTICULAR FORMAT ADHERED TO HERE, EXCEPT THAT THE INITIATION, AND COMPLETION OF JOBS IS READABLE HERE, IN ADDITION TO THE DELETION OF QUEUES.

ONE THING TO NOTE IS THAT MOST OUTPUT OF FIXBAT WHEN -STARTUP IS SPECIFIED, AND THE MONITOR, IS PRECEDED BY THE TIME OF DAY ON THE LINE. THIS IS USEFUL FOR DETERMINING THE TIMING OF CERTAIN OPERATIONS RELATED PERHAPS TO A USER COMPLAINT THAT "SOMETHING WENT WRONG". THEREFORE, RUNNING FIXBAT INTERACTIVELY MIGHT RESULT IN THE LINE:

```
DELETED T$0000
```

BEING TYPED OUT, WHEREAS IF IT HAD SEEN T$0000 WHILE RUNNING FROM PH_GO, THE LINE MIGHT HAVE LOOKED LIKE:

```
06:43:52 DELETED T$0000
```

III. DELETING THE OLD BATCH JOB ENTRIES

WHEN FIXBAT DECIDES TO FIX THE DATABASE, IT CAN ALSO DECIDE TO DELETE CERTAIN OLD BATCH JOB ENTRIES FROM THE QUEUE FILES. THE MECHANISM FOR DOING THIS IS TO ACTUALLY PHYSICALLY REMOVE THE JOB ENTRY FROM THE QUEUE, AND WRITE THE NEXT JOB ENTRY OVER THE DELETED ONE, AND REPEATING THIS PROCEDURE, IN EFFECT FILLING UP THE HOLE MADE UNTIL THE END OF THE QUEUE FILE IS REACHED.

IT WILL ONLY PERFORM THIS OPERATION, HOWEVER, IF A -DAYS ARGUMENT WAS SPECIFIED ON THE COMMAND LINE.

THE MECHANISM FOR DETERMINING WHETHER OR NOT A JOB SHOULD BE DELETED IS AS FOLLOWS:

1) THE JOB MUST NOT BE AN ACTIVE JOB, I.E. IT MUST BE IN A CANCELLED, ABORTED OR COMPLETED STATE.

2) THE JOB MUST HAVE REACHED ITS FINAL STATE IN THE SAME OR PREVIOUS YEAR AS THE CURRENT YEAR.

3) THE JOB MUST HAVE COMPLETED ON A DATE SUCH THAT THERE ARE AT LEAST <N> FULL DAYS BETWEEN THAT DATE AND THE CURRENT DATE, NON-INCLUSIVE. THIS MEANS THAT IF A JOB COMPLETED ON APRIL 10, 1979, AND THE CURRENT DATE IS APRIL 12, 1979, THE ONLY WAY THAT JOB CAN BE DELETED IS IF <N> IS 1. IF <N> IS 2, THE JOB WILL NOT BE DELETED UNTIL THE NEXT DAY.

THE <N> REFERRED TO IN ITEM 3 IS THE ARGUMENT SUPPLIED TO THE -DAYS OPTION.

WHEN FIXBAT DECIDES TO DELETE A JOB, IT WILL OUTPUT THE FINAL INFORMATION ON THAT JOB IN A SIMILAR FORMAT TO THE INFORMATION RETURNED BY A "JOB -DISPLAY" COMMAND, UNLESS THE -QUIET OPTION WAS SPECIFIED ON THE COMMAND LINE.

IV. CLEANING UP THE DATABASE

THE FIXBAT PROGAM WILL ALWAYS PERFORM SOME CLEANING UP OF THE BATCH DATABASE (BATCHQ). THIS CLEANING UP TAKES PLACE IN THREE STAGES:

1) DELETE TEMPORARY FILES OUT OF BATCHQ. TEMPORARY FILES ARE DEFINED AS A FILE WITH A 6-CHARACTER NAME BEGINNING WITH "T$" AND WITH THE LAST 4 CHARACTERS SUCCESSFULLY CONVERTING TO A DECIMAL NUMBER USING CNVA$A (SEE PE-T-315 FOR A DESCRIPTION OF THE APPLIB CNVA$A ROUTINE). THESE FILES ARE GENERATED BY THE BATCH MONITOR TO "BOOTSTRAP" UP BATCH JOBS, INCLUDING ATTACHING TO THE HOME UFD. IF THAT ATTACH FAILS, THE TEMPORARY FILE STAYS AROUND. USUALLY, HOWEVER, THE BATCH MONITOR DELETES THESE FILES ITSELF. WHEN A TEMPORARY FILE IS DELETED, THE MESSAGE "DELETED <NAME>" IS OUTPUT.

2) PROTECT ALL COMMAND FILES IN CIFILE TO "7 0" PROTECTION. A COMMAND FILE IS DEFINED AS A FILE WITH A 6-CHARACTER NAME BEGINNING WITH "C", WHERE THE SECOND CHARACTER IS BETWEEN "0" AND "9" OR "A" AND "V" (INCLUSIVE), AND WITH THE LAST 4 CHARACTERS SUCCESSFULLY CONVERTING TO A DECIMAL NUMBER USING CNVA$A. BETWEEN THIS STEP AND STEP 3, FIXBAT WILL GO THROUGH THE Q.CTRL QUEUE FILES, AND WHENEVER IT FINDS AN ACTIVE JOB, IT WILL ATTACH BACK TO BATCHQ>CIFILE AND PROTECT THAT JOB'S COMMAND FILE TO "1 0" PROTECTION.

3) DELETE ALL COMMAND FILES IN CIFILE. IGNORE "INSUFFICIENT ACCESS RIGHTS" ERRORS, BUT OUTPUT A "DELETED <NAME>" MESSAGE IF SUCCESSFUL. THIS RESULTS IN ALL COMMAND FILES THAT DO NOT BELONG TO ACTIVE JOBS BEING DELETED.

STEPS 2 AND 3 WILL NOT BE PERFORMED UNLESS FIXBAT DECIDES TO FIX THE DATABASE. BETWEEN STEPS 1 AND 2, THE MESSAGE "FIXING DATABASE" IS OUTPUT. THEREFORE, A SAMPLE LOG FILE OUTPUT MIGHT LOOK AS FOLLOWS:

```
06:43:52 DELETED T$0000
06:43:52 DELETED T$0001
06:43:52 FIXING DATABASE.
06:43:54 DELETED C10034
06:43:54 DELETED C20002
06:43:55 FIXBAT FINISHED.
```

V. FIXBAT ERROR MESSAGES AND RESPONSES

WHILE FIXBAT IS RUNNING, IT MAY OUTPUT CERTAIN MESSAGES DESCRIBING WHAT IT IS DOING, OR IT MAY ABORT WITH A PARTICULAR ERROR MESSAGE.

IN GENERAL, IF FIXBAT ABORTS, IT MEANS THAT CERTAIN PARTS OF THE DATABASE ARE IRRETRIEVABLY LOST. IT IS EXPECTED THAT THIS WILL USUALLY BE BATCH JOB DATA. WHILE DELETING THE OFFENDING FILE AND RE-RUNNING FIXBAT MAY HELP, IT DOES NOT GUARANTEE THAT FIXBAT WILL NOT ABORT ON A DIFFERENT FILE.

IF FIXBAT DOES NOT SEEM TO BE ABLE TO FIX THE DATABASE, THE COMMAND FILE C_RSET, OR, AS A LAST RESORT, C_BDIF SHOULD BE INVOKED. SEE PE-T-571 FOR A DESCRIPTION OF THESE COMMAND FILES.

DELETED <FILENAME>.
THIS MESSAGE MEANS THAT FIXBAT FOUND EITHER A TEMPORARY (T$XXXX) FILE, AN INACTIVE COMMAND FILE (CQNNNN), OR A QUEUE FILE (QCTRQP) THAT HELD ENTRIES THAT WERE ALL PAST THE -DAYS ARGUMENT, AND DELETED IT.

FIXING DATABASE.
THIS MESSAGE IS OUTPUT WHEN FIXBAT DECIDES TO ACTUALLY FIX THE ENTIRE BATCHQ DATABASE.

EXECUTE/DATA USERNAME MISMATCH. RE-INITIALIZE DATABASE.
FIXBAT HAS FOUND A JOB THAT IS SUPPOSEDLY EXECUTING, BUT THE CORRESPONDING JOB ID IN THE EXECUT FILE IS OWNED BY A DIFFERENT USER. USE C_RSET TO RE-INITIALIZE THE DATABASE, AND CONSIDER CHANGING THE BATCHQ PASSWORD, AS IT MAY INDICATE A SECURITY BREAKAGE.

REDUNDANT EXECUT ENTRY. RE-INITIALIZE DATABASE.
FIXBAT FOUND AN EXECUTING JOB THAT HAD MORE THAN ONE ENTRY IN THE EXECUT FILE, WHICH IS IMPOSSIBLE. USE C_RSET TO RE-INITIALIZE THE DATABASE.

EXECUTE DATA NOT FOUND (REINITIALIZE). <JOBID>
A JOB THAT HAD EXECUTION STATUS IN THE QUEUE FILE HAD NO CORRESPONDING ENTRY IN EXECUT. THIS CAN BE THE RESULT OF THE BATCH MONITOR BEING LOGGED OUT FORCIBLY IN BETWEEN UPDATING THE QUEUE FILE AND UPDATING EXECUT. USE C_RSET TO RE-INITIALIZE.

<FILENAME> LEFTOVER WORDS=NNNNN
THE INDICATED QUEUE FILE HAD NNNNN WORDS AT THE END OF IT, WHICH WAS NOT ENOUGH FOR A FULL QUEUE ENTRY. THIS IS NOT A FATAL ERROR, AND THE QUEUE FILE WILL MERELY BE TRUNCATED. IT COULD INDICATE A PROCESS SUBMITTING A JOB BEING FORCE-LOGGED OUT IN THE MIDDLE OF CREATING THE NEW QUEUE ENTRY.

FIXBAT FINISHED.
THE PROCESS OF FIXING THE BATCH DATABASE HAS BEEN SUCCESSFULLY

COMPLETED. FIXBAT WILL NOW EXIT TO PRIMOS.

BATCH PROCESS STILL IN PROGRESS.
MEANS THAT AN ATTEMPT WAS MADE TO RUN FIXBAT WHILE THE BATCH MONITOR WAS STILL LOGGED IN. USE "BATCH SYSTEM -STOP" TO LOG OUT THE BATCH MONITOR, THEN RUN FIXBAT.

ARGUMENT REQUIRED. <OPTION>
<OPTION> IS EITHER "-STARTUP" OR "-DAYS". THIS MESSAGE MEANS THAT THE OPTION WAS SPECIFIED WITH NO ARGUMENT. FOR "-STARTUP", LEGAL ARGUMENTS ARE "SPOOL", "SAVE", "DELETE", OR "NOLOG". FOR "-DAYS", A LEGAL ARGUMENT IS A DECIMAL NUMBER FROM 1 TO 60, INCLUSIVE.

OTHER ERROR MESSAGES SHOULD BE LOOKED UP IN PE-T-584, APPENDIX B.

```
**    J  J      I     M    M   M    M     Y
**     JJ      III    M    M   M    M     Y
**
**
**
**
**    RRRR    BBBB     AAA    TTTTT    CCC    H    H    333
**    R   R   B   B   A   A     T     C   C   H    H   3   3
**    R   R   B   B   A   A     T     C       H    H       3
**    RRRR    BBBB    AAAAA     T     C       HHHHH     33
**    R R     B   B   A   A     T     C       H    H       3
**    R  R    B   B   A   A     T     C   C   H    H   3   3
**    R   R   BBBB    A   A     T      CCC    H    H    333
**
**
**
**********************************************************************************************
**********************************************************************************************
```

ADMINISTRATOR'S GUIDE TO REVISION 17 BATCH

DATE: APRIL 12, 1979

TO:

FROM:

SUBJECT: ADMINISTRATOR'S GUIDE TO REVISION 17 BATCH

REFERENCE:

ABSTRACT

THIS DOCUMENT IS A GUIDE TO THE SYSTEM ADMINISTRATOR WHO WANTS TO INSTALL REVISION 17 BATCH. IT EXPLAINS THE FILE STRUCTURE OF THE SUBSYSTEM, EXPLAINS THE PROCEDURE FOR STARTING UP THE BATCH SUBSYSTEM, AND SHOWS HOW TO CLEAN UP AFTER BATCH.

I. FILE SYSTEM STRUCTURE

A. FILES IN THE BATCH SUBSYSTEM

THE BATCH SYSTEM DIRECTLY INVOLVES 3 TOP-LEVEL UFDS ON EACH SYSTEM. THESE ARE CMDNCO, BATCH, AND BATCHQ. CMDNCO IS USED TO HOLD ONLY THE COMMANDS ($$, BATCH, BATGEN AND JOB) THEMSELVES. BATCH CONTAINS THE SOURCE CODE (FORTRAN WITH SOME PMA) FOR THE BATCH SUBSYSTEM INCLUDING THE COMMAND FILES NECESSARY TO BUILD AND INSTALL ITSELF. BATCHQ CONTAINS THE COMMAND FILES USED TO INITIALIZE THE BATCH DATABASE (WHICH WILL ALSO RESIDE IN BATCHQ).

HERE IS A LIST OF ALL FILES INVOLVING THE BATCH SUBSYSTEM:

UFD CMDNCO:

*	$$	THE $$ COMMAND (SIMPLY EXITS WHEN INVOKED).
*	BATCH	THE BATCH COMMAND (STATUS AND MODIFICATION OF PROCESSES).
*	BATGEN	THE BATGEN COMMAND (STATUS AND MODIFICATION OF QUEUES).
*	JOB	THE JOB COMMAND (STATUS AND MODIFICATION OF JOBS).

UFD BATCH:

	$$	THE SOURCE CODE (FTN) FOR THE $$ COMMAND.
	BATCH	THE SOURCE CODE (FTN) FOR THE BATCH COMMAND.
	BATGEN	THE SOURCE CODE (FTN) FOR THE BATGEN COMMAND.
	FIXBAT	THE SOURCE CODE (FTN) FOR THE *FIXBAT PROGRAM.
	INIT	THE SOURCE CODE (FTN) FOR THE *INIT PROGRAM.
	JOB	THE SOURCE CODE (FTN) FOR THE JOB COMMAND.
	MONITR	THE SOURCE CODE (FTN) FOR THE *MONITR PROGRAM.
	B$LIBF	THE SOURCE CODE (FTN) FOR COMMON SUBROUTINES.
	B$LIBP	THE SOURCE CODE (PMA) FOR COMMON SUBROUTINES.
	B$COMN	FORTRAN $INSERT FILE FOR COMMON DATA.
	B$EXEC	FORTRAN $INSERT FILE FOR EXECUTION STATUS FILE.
	B$JCBS	FORTRAN $INSERT FILE FOR JOB QUEUE ENTRY DEFINITION.
	B$KEYS	FORTRAN $INSERT FILE FOR COMMON DECLARATIONS.
	B$QCOM	FORTRAN $INSERT FILE FOR COMMON QUEUE-HANDLING DATA.
	B$QSTA	FORTRAN $INSERT FILE FOR QUEUE STATUS FILE DEFINITION.
	B$QUEU	FORTRAN $INSERT FILE FOR BATDEF FILE DEFINITION.
	C_BATCH	COMMAND FILE TO BUILD AND INSTALL THE BATCH SUBSYSTEM.
	C_LIST	COMMAND FILE TO GENERATE A LISTING OF THE BATCH SUBSYSTEM.

UFD BATCHQ:

	C_BDIF	COMMAND FILE TO INITIALIZE ENTIRE BATCH DATABASE.
	C_RSET	COMMAND FILE TO INITIALIZE BATCH JOB QUEUES (NOT BATDEF).
	PH_GO	PHANTOM FILE RUN BY BATCH MONITOR.
*	*MONITR	RUN-FILE IMAGE OF BATCH MONITOR.
*	*FIXBAT	PROGRAM TO FIX BATCH DATABASE UP.
*	*INIT	PROGRAM TO INITIALIZE QUEUE AND EXECUT FILES.
$	VALID.	DETERMINES VALIDITY OF BATCH DATABASE.
$	CIFILE	(SUB-UFD) CONTAINS COMMAND FILES AWAITING EXECUTION.

```
$  Q.CTRL  (SUB-UFD) CONTAINS QUEUE CONTROL FILES WITH JOB DATA.
$  QUEUE   CONTAINS CURRENT QUEUE INFO (# JOBS WAITING, ETC.).
$  EXECUT  CONTAINS INFO ON CURRENTLY EXECUTING JOBS.
#  BATDEF  BATCH DEFINITION FILE (ENVIRONMENT).
```

THE SPECIAL SYMBOLS IN FRONT OF THE FILENAMES HAVE THE FOLLOWING MEANINGS:

```
* - GENERATED BY THE BATCH>C_BATCH COMMAND FILE.
$ - GENERATED BY THE C_RSET AND C_BDIF FILES.
# - GENERATED BY THE C_BDIF FILE ONLY (USING BATGEN).
```

B. BUILDING THE BATCH SUBSYSTEM

TO BUILD (OR RE-BUILD) THE BATCH SUBSYSTEM, ATTACH TO THE BATCH UFD AND INVOKE THE COMMAND FILE C_BATCH AS FOLLOWS:

```
OK, ATTACH BATCH
OK, COMINPUT C_BATCH
```

THIS COMMAND FILE WILL COMPILE, ASSEMBLE, LOAD AND INSTALL ALL OF THE FILES MARKED WITH AN ASTERISK ("*") IN THE ABOVE LIST, DELETING TEMPORARY FILES, BINARIES, RUN-FILES, ETC. THAT IT GENERATED IN THE BATCH UFD.

AFTER THE C_BATCH COMMAND FILE IS FINISHED, ATTACH TO THE BATCHQ UFD AND INVOKE THE C_BDIF FILE AS FOLLOWS:

```
OK, ATTACH BATCHQ <OWNER-PASSWORD>
OK, CLOSE ALL
OK, COMINPUT C_BDIF
```

NOTE: THIS COMMAND FILE CANNOT BE EXECUTED UNLESS THE SYSTEM DATE AND TIME HAVE BEEN SET VIA THE OPERATOR SETIME COMMAND.

THIS COMMAND FILE WILL FIRST MAKE SURE THAT THE BATCH MONITOR IS NOT RUNNING (IF IT IS, LOG IT OUT, AND THEN TRY THE C_BDIF FILE AGAIN), THEN USE FUTIL TO TREDELETE THE CIFILE AND Q.CTRL SUB-UFDS, THEN IT WILL RE-CREATE THEM, DELETE ALL T$XXXX FILES IN BATCHQ, DELETE THE BATDEF, QUEUE AND EXECUT FILES, RUN *INIT TO RE-CREATE EMPTY QUEUE AND EXECUT FILES, THEN RUN BATGEN TO GENERATE AN EMPTY BATDEF FILE.

THE TWO NON-FATAL ERROR MESSAGES "FILE IN USE. IN.USE" FROM FUTIL AND "NOT FOUND. BATDEF" FROM BATGEN ARE EXPECTED AND ARE TO BE IGNORED.

WHEN IT FINISHES WITH THIS, IT WILL CREATE A FILE CALLED "VALID." WHICH INDICATES THAT THE BATCH DATABASE IS VALID SIMPLY BY EXISTING. IF THIS FILE IS DELETED BY ANYONE OR ANY PROGRAM (INCLUDING BATCH PROGRAMS TO INDICATE AN INVALID DATABASE), THE C_BDIF OR C_RSET FILES SHOULD BE USED AGAIN TO RE-INITIALIZE THE DATABASE, IF *FIXBAT FAILS IN THE ATTEMPT (SEE PE-T-622).

IF THE BATDEF FILE ALREADY CONTAINS USEFUL INFORMATION, AND IT IS NOT DESIREABLE TO HAVE THAT INFORMATION DESTROYED BY C_BDIF, THE COMMAND FILE C_RSET (RESET JOB QUEUES) CAN BE USED INSTEAD. IT BEHAVES EXACTLY LIKE C_BDIF, EXCEPT THAT IT LEAVES BATDEF INTACT.

AFTER EITHER OF THESE FILES HAVE BEEN EXECUTED (ASSUMING BATDEF IS VALID IF C_RSET WAS USED), THE BATCH SUBSYSTEM IS NOW BUILT AND FULLY INSTALLED.

SECTION II IS A REFERENCE TO A DOCUMENT THAT EXPLAINS HOW TO DEFINE QUEUES USING THE BATGEN COMMAND. ONCE THIS IS ACCOMPLISHED, SECTION III IS THE NEXT ONE TO REFERENCE (RUNNING THE BATCH MONITOR).

C. CHANGING PASSWORDS

SOME INSTALLATIONS MAY CHOOSE TO INCREASE SECURITY WITHIN THE BATCH SUBSYSTEM, AS IT CAN BE USED FOR POTENTIALLY EVIL PURPOSES (SUCH AS LOGGING A PHANTOM IN UNDER SOMEONE ELSE'S LOGIN-NAME).

TO DO THIS, OWNER PASSWORDS SHOULD BE GIVEN TO BATCHQ, BATCH, AND CMDNCO.

HOWEVER, SINCE THE BATCH SUBSYSTEM NEEDS ACCESS TO BATCHQ AND ITS SUB-UFDS, IT MUST KNOW THE BATCHQ PASSWORD. TO GIVE IT THE NEW PASSWORD, DO THE FOLLOWING:

```
OK, ATTACH BATCH <OWNER-PASSWORD>
OK, ED B$LIBE
EDIT
FIND C_ATCH$B
C ATCH$B, BATCH, JCB, 03/29/79
LOCATE BATPAS;*2
      INTEGER BATPAS(3),OPASS(3),NPASS(3),LSTNAM(17)
      DATA BATPAS/'<OLD-PASSWORD>'/
CHANGE /<OLD-PASSWORD>/<NEW-PASSWORD>/
      DATA BATPAS/'<NEW-PASSWORD>'/
FILE

OK,
```

NOTE THAT THE NEW PASSWORD MUST BE EXACTLY SIX CHARACTERS LONG (INCLUDING TRAILING BLANKS).

AFTER THIS CHANGE HAS BEEN MADE, FOLLOW THE PROCEDURE FOR BUILDING THE BATCH SUBSYSTEM DESCRIBED ABOVE IN SECTION I.B.

AFTER BUILDING THE BATCH SUBSYSTEM, THE PASSWORD SHOULD THEN BE SET IN THE BATCHQ UFD TO <NEW-PASSWORD>. THEN, ATTACH DOWN TO THE CIFILE AND 0,CTRL SUB-UFDS AND SET ANY RANDOM PASSWORDS IN THEM (THESE SUB-UFDS ARE WHERE THE REAL SENSITIVE INFORMATION EXISTS), SO THAT USERS WILL

NOT BE ABLE TO ATTACH DOWN TO THEM AS OWNERS.

AT THIS POINT, PROTECT THE PH_GO AND *FIXBAT FILES IN BATCHQ TO "7 1" PROTECTION, BECAUSE PH_GO IS REFERENCED AS A NON-OWNER WHEN THE BATCH MONITOR IS STARTED UP, AND THE FIRST THING IT DOES IS RUN BATCHQ>*FIXBAT. THE *FIXBAT PROGRAM WILL ATTACH TO BATCHQ AS AN OWNER (SINCE IT KNOWS THE PASSWORD), BUT ONLY IF THE USER IS LOGGED IN AS "SYSTEM", AND WAS SPAWNED AS A PHANTOM FROM THE SYSTEM CONSOLE (I.E. HAS PRIVILEGES). SEE PE-T-622 FOR DETAILS.

THE BATCH SUBSYSTEM DOES NOT NEED TO KNOW THE PASSWORDS OF THESE SUB-UFDS, SINCE IT CAN GET THEM (AND ALWAYS DOES) WHEN IT IS ATTACHED TO BATCHQ AS AN OWNER.

NOTE THAT INVOKING THE C_RSET OR C_BDIF FILES WILL RE-CREATE THE Q.CTRL AND CIFILE SUB-UFDS, WHICH RESETS THEIR PASSWORDS TO BLANKS. WHENEVER THESE COMMAND FILES ARE RUN, THE SUB-UFDS SHOULD BE RE-PASSWORDED BY THE SYSTEM ADMINISTRATOR.

THE BATCH UFD SHOULD ALSO BE PASSWORDED SO THAT USERS WHO UNDERSTAND THE BATCH SUBSYSTEM (OR WHO HAVE READ THIS DOCUMENT) MAY NOT LOOK AT THE B$LIBF FILE TO DETERMINE THE PASSWORD. NO SOFTWARE NEEDS TO KNOW THE BATCH PASSWORD. CMDNCO SHOULD ALSO HAVE A PASSWORD, ALTHOUGH THIS IS NOT A REQUIREMENT.

ONCE BATCHQ IS PASSWORDED, THE C_BATCH COMMAND FILE WILL NO LONGER WORK. IF IT IS DESIRED TO RE-BUILD THE BATCH SUBSYSTEM, EITHER REMOVE THE BATCHQ PASSWORD OR ENCODE IT IN THE C_BATCH FILE WHENEVER IT REFERENCES BATCHQ (IT ONLY DOES SO ONCE, AFTER A FUTIL COMMAND).

II. DEFINING THE BATCH ENVIRONMENT - THE BATGEN COMMAND

PLEASE REFERENCE SECTION 3 OF PE-T-584, "A BATCH SUBSYSTEM FOR PRIMOS", FOR A DESCRIPTION OF THE BATGEN COMMAND AND IT USE.

III. RUNNING THE BATCH MONITOR

A. INITIATING THE BATCH MONITOR

TO SPAWN THE BATCH MONITOR, ENTER THE COMMAND:

OK, PHANTOM BATCHQ>PH_GO

FROM THE SYSTEM CONSOLE.

THIS FILE WILL RUN THE BATCHQ>*FIXBAT PROGRAM TO FIX ANY PROBLEMS WITH THE BATCH DATABASE (AND OPTIONALLY DELETE OLD JOB ENTRIES, SEE PE-T-622 FOR DETAILS), THEN WHEN IT FINISHES THAT, THE *MONITR PROGRAM WILL BE RUN, WHICH WILL SEND THE FOLLOWING MESSAGE TO THE SYSTEM CONSOLE:

BATCH WAITING FOR BATCH SYSTEM -START.

AT THIS POINT, THE BATCH MONITOR SHOULD BE CHAPED TO THE APPROPRIATE LEVELS.

THESE LEVELS REFER TO THE "RLEVEL" AND THE "TIMESLICE" THAT ARE VALUES RELATED TO QUEUE DEFINITION USING THE BATGEN COMMAND.

THE FORMAT OF THE CHAP COMMAND IS:

CHAP -<USER-NUMBER> <RLEVEL> <TIMESLICE>

WHERE <USER-NUMBER> IS THE DECIMAL USER-NUMBER OF THE PHANTOM SPAWNED, <RLEVEL> IS A NUMBER FROM 0 TO 3, WHERE 1 IS THE INITIAL VALUE, 0 IS THE LOWEST, AND 3 IS THE HIGHEST, AND <TIMESLICE> IS THE OCTAL TIMESLICE OF THE PROCESS IN TENTHS OF A SECOND.

WHEN A PROCESS IS SPAWNED FROM A PARTICULAR QUEUE, IT INITIALLY IS GIVEN THE SAME RLEVEL AND TIMESLICE THAT ITS SPAWNER, THE BATCH MONITOR, HAS AT THE TIME OF THE SPAWNING. IT WILL SHORTLY THEREAFTER LOWER ITS RLEVEL BY THE "DELTA-RLEVEL" VALUE SPECIFIED IN THE RLEVEL SUBCOMMAND OF BATGEN FOR THAT PARTICULAR QUEUE (SEE PE-T-584), AND SET ITS TIMESLICE TO EITHER THE VALUE GIVEN IN THE BATGEN TIMESLICE SUBCOMMAND FOR THAT QUEUE OR THE TIMESLICE IT ALREADY HAS, WHICHEVER IS SMALLER.

THEREFORE, THE RLEVEL OF THE BATCH MONITOR REPRESENTS THE HIGHEST RLEVEL THAT ANY BATCH JOBS ON THE SYSTEM WILL EVER RUN AT (ALTHOUGH THIS SHOULD BE AT LEAST 1 TO ALLOW SPEEDY DISPATCHING OF JOBS), AND THE TIMESLICE SHOULD ALSO BE THE HIGHEST TIMESLICE EVER USED BY ANY JOB (THE MINIMUM SHOULD BE 20 DECIMAL, OR 2 SECONDS, WHICH IS 24 OCTAL).

ALL DELTA-RLEVELS IN THE BATDEF FILE GENERATED BY BATGEN KEY OFF OF THE ACTUAL RLEVEL OF THE MONITOR; THEREFORE, THE MONITOR'S RLEVEL WHILE RUNNING SHOULD BE DECIDED BEFORE THE DELTA-RLEVELS ARE CHANGED, AS ONE SET WILL AFFECT THE OTHER.

AN EXAMPLE CONFIGURATION MIGHT BE:

```
QUEUE DEFAULT DELTA-RLEVEL=2, TIMESLICE=20
QUEUE BACKGROUND DELTA-RLEVEL=3, TIMESLICE=50
QUEUE EXPRESS DELTA-RLEVEL=1, TIMESLICE=1
```

NOTE THAT THESE TIMESLICE VALUES ARE ALL IN DECIMAL REPRESENTATION...THE ONLY PLACE THAT A TIMESLICE IS REPRESENTED AS AN OCTAL NUMBER IS IN THE CHAP COMMAND.

THE OPERATOR WOULD PROBABLY ENTER THE COMMAND:

OK, CHAP -60 3 62 /* OCTAL TIMESLICE.

FROM THE SYSTEM CONSOLE AFTER SPAWNING THE PHANTOM. JOBS THAT WERE RUN FROM THOSE QUEUES WOULD RUN WITH THE FOLLOWING VALUES:

QUEUE DEFAULT RLEVEL=1, TIMESLICE=20 (NORMAL USER)
QUEUE BACKGROUND RLEVEL=0, TIMESLICE=50 (LONG JOBS)
QUEUE EXPRESS RLEVEL=2, TIMESLICE=1 (SHORT IMPORTANT JOBS)

NOTE THAT THE BATCH MONITOR WAS CHAPED UP ONE RLEVEL HIGHER THAN NECESSARY...THIS HELPS ELIMINATE ANY PROBLEMS HAVING TO DO WITH A BATCH JOB TYING UP THE CPU SO THAT THE MONITOR CAN'T RUN FAST ENOUGH. IN GENERAL, THIS SHOULD NOT BE A PROBLEM, BUT IT COULD HAPPEN.

THE TIMESLICES WERE LOWERED FOR THE "FASTER" JOBS TO PREVENT THEM FROM TYING UP THE CPU FOR TOO LONG, AND THEY WERE RAISED FOR THE "SLOWER" JOB TO GUARANTEE THEM HAVING A REASONABLE AMOUNT OF CPU TIME.

B. TELLING THE BATCH MONITOR TO START UP

AFTER THE CHAP COMMAND HAS BEEN ISSUED, THE MONITOR MUST BE INFORMED THAT EVERYTHING IS "READY" FOR IT TO BEGIN PROCESSING. THIS IS DONE WITH THE BATCH COMMAND AS FOLLOWS:

BATCH SYSTEM -START

THE PROGRAM SHOULD RESPOND:

START-UP REQUEST ISSUED.

IF IT DOES NOT, THEN THE MONITOR IS EITHER NOT RUNNING, ALREADY STARTED-UP, OR JUST SPAWNED AND NOT READY FOR START-UP. IN THE THIRD CASE, WAIT FOR THE "*BATCH* WAITING FOR BATCH SYSTEM -START" MESSAGE TO BE SENT TO THE SYSTEM CONSOLE, AND TRY THE COMMAND AGAIN.

WHEN THE BATCH MONITOR RECEIVES THE START-UP REQUEST, IT WILL SEND A MESSAGE TO THE SYSTEM CONSOLE SAYING:

BATCH OPERATOR START-UP.

MEANING THAT IT HAS STARTED RUNNING. IF IT SENDS A MESSAGE SAYING:

BATCH START-UP REQUEST PREVIOUSLY PROCESSED.

THEN A BATCH SYSTEM -START COMMAND HAD ALREADY BEEN ISSUED TO IT.

C. MONITORING THE BATCH MONITOR

TO DETERMINE THE GENERAL STATUS OF THE BATCH SYSTEM (AND MONITOR), USE THE BATCH COMMAND AS FOLLOWS:

```
BATCH [SYSTEM] -DISPLAY
```

IT WILL OUTPUT THE NUMBER OF WAITING AND HELD JOBS PER QUEUE, AND THE CURRENTLY EXECUTING JOBS INCLUDING THE USER NAME, INTERNAL NAME AND QUEUE IN WHICH THE JOB IS RUNNING.

WHENEVER THE MONITOR SPAWNS A JOB TO PROCESS A USER'S COMMAND FILE, THE SPAWNED JOB WILL SEND A MESSAGE TO THE SYSTEM CONSOLE AS FOLLOWS:

```
*BATCH* EXECUTING EXTNAM FOR USER USRNAM(JOBIDN).
```

WHERE EXTNAM IS THE EXTERNAL NAME, USRNAM IS THE SUBMITTING USER, AND JOBIDN IS THE INTERNAL NAME OF THE JOB.

WHEN THE JOB ABORTS (OR COMPLETES), THE MONITOR WILL SEND A MESSAGE TO THE SYSTEM CONSOLE AS FOLLOWS:

```
*BATCH* JOB EXTNAM FOR USER USRNAM(JOBIDN) ABORTED.
```

WHERE EXTNAM, USRNAM, AND JOBIDN ARE EXPLAINED ABOVE, AND "ABORTED" MAY BE REPLACED BY "COMPLETED" IF APPROPRIATE.

THESE MESSAGES WILL HELP THE OPERATOR IN MONITORING BATCH USAGE AND LOAD SOMEWHAT WITHOUT HAVING TO MAKE TOO MANY INQUIRIES.

TWO JOB OPTIONS ARE ALSO USEFUL FOR OBTAINING THE STATUS OF THE BATCH SYSTEM: THEY ARE -STATUS AND -DISPLAY. THE -STATUS OPTION ONLY RETURNS THE JOB NAME, THE EXTERNAL AND INTERNAL NAMES, THE QUEUE AND THE STATUS OF SPECIFIED JOB(S); WHILE THE -DISPLAY RETURNS ALL OF THE INFORMATION ON THE SPECIFIED JOB(S).

WHEN THE COMMAND IS ISSUED FROM ANY PROCES LOGGED IN UNDER SYSTEM (INCLUDING THE SYSTEM CONSOLE), IT IS CAPABLE OF DISPLAYING ALL OF THE JOBS ON THE SYSTEM. NORMALLY, A USER CAN ONLY DISPLAY JOBS WITH THE SAME USER-NAME.

THE FORMAT OF THESE COMMANDS IS:

```
JOB -STATUS  [ALL]
    -DISPLAY [TODAY]
```

IF THE ALL OR TODAY PARAMETER IS OMITTED, ONLY "ACTIVE" JOBS (THOSE JOBS THAT ARE WAITING, HELD OR COMPLETED) ARE DISPLAYED. IF ALL IS SPECIFIED, ALL JOBS IN THE SYSTEM ARE DISPLAYED. IF TODAY IS SPECIFIED, ONLY THOSE JOBS THAT WERE INITIATED, SUBMITTED, COMPLETED, ABORTED OR CANCELLED ON THE SAME DATE THAT THE COMMAND WAS ISSUED ARE DISPLAYED.

THE FORMAT OF THE DISPLAY IS TABULAR IF -STATUS WAS USED, OTHERWISE IT COMES OUT ENTRY BY ENTRY.

TO REFER TO SPECIFIC JOBS IN THE BATCH SYSTEM, THE INTERNAL NAMES OF THE JOBS MUST BE USED SINCE EXTERNAL NAME REFERENCES ARE ONLY TO JOBS BELONGING TO THE USER MAKING THE REFERENCE (IN THIS CASE, SYSTEM).

THEREFORE, THE FORMAT OF SPECIFICALLY LOOKING AT JOBS IS:

```
JOB JOBID -STATUS [ALL]
          -DISPLAY [TODAY]
```

WHERE JOBID IS IN THE FORM #SNNNN (EXAMPLE: #10032). THESE INTERNAL NAMES ARE OUTPUT BY NEARLY ALL BATCH PROGRAMS WHEN THEY REFER TO JOBS (INCLUDING -STATUS AND -DISPLAY), SO THEY ARE RELATIVELY EASY TO ACCESS.

THE ALL AND TODAY SUFFIXES (AND THE ABSENCE THEREOF) HAVE THE SAME MEANING AS DESCRIBED ABOVE; THE DIFFERENCE IS THAT THERE IS A JOBID, WHICH LIMITS OUTPUT TO ONLY THOSE JOBS WITH THE SAME NAME (INTERNAL OR EXTERNAL) AS JOBID. EXTERNAL NAMES ARE ONLY COMPARED IF THE USER WHO SUBMITTED THE JOB IS THE SAME USER ISSUING THE JOB COMMAND (THEREFORE THE ONLY WAY FOR A USER LOGGED IN AS SYSTEM TO REFERENCE ANOTHER USER'S JOB IS TO USE THE INTERNAL NAME OF THAT JOB, OR DISPLAY IT BY NOT PROVIDING A JOBID AND THEREFORE DISPLAYING ALL JOBS).

IF IT IS DESIRED TO LOOK AT CURRENTLY DEFINED QUEUES, THE BATGEN -STATUS AND -DISPLAY COMMANDS SHOULD BE USED. THEY WILL OUTPUT SHORT AND LONG DESCRIPTIONS (RESPECTIVELY) OF THE CURRENTLY DEFINED QUEUES (-STATUS DOES IT IN TABULAR FORMAT) AS LONG AS THE BATDEF FILE IN BATCHQ IS READ-PERMITTED.

A TREENAME MAY BE SPECIFIED ON THE COMMAND LINE IF DESIRED, I.E.:

```
BATGEN [TREENAME] -STATUS
                  -DISPLAY
```

D. OPERATOR COMMANDS

THE OPERATOR HAS NEARLY FULL CONTROL OVER ALL JOBS IN THE BATCH SYSTEM.

HE CAN PERFORM ALL OF THE OPERATIONS THAT A USER CAN WHILE LOGGED IN UNDER SYSTEM UNDER THE FOLLOWING RESTRICTIONS:

1) HE MUST REFER TO ALL JOBS USING THEIR INTERNAL NAME.

2) HE CANNOT -ABORT OR -RESTART ANY JOBS UNLESS HE IS EITHER AT THE SYSTEM CONSOLE OR IS REFERENCING HIS OWN JOBS.

IF HE ATTEMPTS TO -ABORT A JOB FROM A TERMINAL THAT IS NOT THE SYSTEM CONSOLE WHILE LOGGED IN UNDER SYSTEM (ASSUMING THE JOB TO BE ABORTED IS

NOT LOGGED IN UNDER SYSTEM), THE ABORT WILL SIMPLY FAIL. IF HE ATTEMPTS TO -RESTART UNDER THE SAME CIRCUMSTANCES, THE JOB WILL SUCCESSFULLY BE FLAGGED AS BEING RESTARTABLE (UNLESS IT IS NOT RESTARTABLE), AND THEN IT WILL ALSO FAIL ON THE FORCE-LOGOUT ("INSUFFICIENT ACCESS RIGHTS").

EITHER WAY, THE INTEGRITY OF THE SYSTEM IS NOT DAMAGED; IF -RESTART WAS USED, THE JOB WILL BE RESTARTED WHEN IT COMPLETES (OR ABORTS); THE FORCE LOGOUT OF THE JOB IS ONLY TO SPEED UP THE PROCESS OF RESTARTING THAT JOB.

HERE IS A QUICK SUMMARY OF OPERATOR COMMANDS:

```
JOB JOBID -CANCEL    /* CANCEL A JOB.
          -ABORT     /* ABORT A JOB.
          -RESTART   /* RESTART A JOB.
          -HOLD      /* HOLD A JOB.
          -RELEASE   /* RELEASE A HELD JOB.
          -STATUS    /* STATUS OF A JOB.
          -DISPLAY   /* EXTENDED STATUS OF A JOB.
```

THE -HOLD AND -RELEASE OPTIONS ARE ONLY AVAILABLE TO THE OPERATORS. WHEN A JOB IS HELD, IT WILL STILL BE CONSIDERED AN "ACTIVE" JOB, AND IT WILL BE COUNTED IN THE LIST OF WAITING AND HELD JOBS GIVEN BY BATCH -DISPLAY (ALTHOUGH EXECUTING JOBS WILL NOT BE COUNTED HERE).

HOLDING A JOB IS USEFUL WHEN IT IS KNOWN THAT A RESOURCE THE JOB EXPECTS (MAGTAPE, DISK SPACE, LINE PRINTER, ETC.) IS NOT AVAILABLE.

WHEN THE RESOURCES ARE AVAILABLE, THE JOB CAN BE RELEASED WITH THE -RELEASE COMMAND.

E. STOPPING THE BATCH MONITOR

TO STOP THE BATCH MONITOR THE CLEANEST WAY POSSIBLE, USE THE BATCH COMMAND AS FOLLOWS:

```
BATCH SYSTEM -STOP
```

THIS WILL CAUSE THE MONITOR TO BE REQUESTED TO LOG ITSELF OUT. WHEN IT SEES THE REQUEST, IT WILL SEND A MESSAGE STATING THIS FACT TO THE SYSTEM CONSOLE ("OPERATOR STOP."), AND LOG ITSELF OUT.

IT IS DANGEROUS TO LOGOUT THE MONITOR (USING LOGOUT ALL OR LOGOUT -NN), SHUTDN ALL OR SHUTDN THE PARTITION ON WHICH BATCHQ RESIDES WHILE THE MONITOR IS RUNNING, SINCE IT MAY BE UPDATING IMPORTANT QUEUE INFORMATION, AND SUDDEN DEATH COULD INVALIDATE THE DATABASE.

IF THIS OCCURS, THE DATABASE WILL BE INACCESSIBLE BY USERS UNTIL THE MONITOR IS RESTARTED (WHICH RUNS *FIXBAT), *FIXBAT IS RUN "OFF-LINE", OR THE C_BDIF OR C_RSET FILES ARE INVOKED BY THE SYSTEM ADMINISTRATOR.

IV. CLEANING UP AFTER BATCH

AS JOBS ARE SUBMITTED TO THE BATCH MONITOR, THE JOB QUEUE FILES GROW LARGER AND LARGER. DEPENDING ON THE NUMBER OF JOBS SUBMITTED PER DAY, THE QUEUES COULD GET SO LARGE THAT A SIGNIFICANT AMOUNT OF DISK SPACE COULD BE WASTED ON THEM AS TIME PASSES.

THE QUICKEST WAY TO CLEAN UP THE BATCH DATABASE IS TO USE THE C_BDIF OR C_RSET FILES IN BATCHQ MENTIONED IN SECTION I.B.

HOWEVER, IT MAY SOMETIMES BE DESIREABLE TO CLEAN THINGS UP IN A SLOWER WAY. THIS CAN BE DONE ON A PER-QUEUE BASIS USING THE BATGEN DELETE COMMAND (DESCRIBED IN PE-T-584).

SIMPLY DELETE THE FULLEST QUEUE; THAT QUEUE WILL NO LONGER BE SUBMITTED TO BECAUSE IT WILL EFFECTIVELY BE DEAD AS FAR AS USERS ARE CONCERNED. HOWEVER, IT WILL NOT ACTUALLY CAUSE THE DELETION OF ALL QUEUE CONTROL FILES AND COMMAND FILES UNTIL THERE ARE NO WAITING OR EXECUTING JOBS IN THAT QUEUE.

THEREFORE, IF AT 7 P.M. THE QUEUE BACKGROUND WERE DELETED, AND IT HAD 9 HOUR-LONG JOBS WAITING IN IT, USERS WOULD NOT BE ABLE TO SUBMIT TO THE QUEUE AFTER 7 P.M. BUT THE 9 JOBS WOULD BE RUN, AND THEN, AT AROUND 3 A.M., THE QUEUE WOULD SIMPLY DISAPPEAR AFTER THE LAST JOB WAS RUN.

ANOTHER WAY TO CLEAN UP THE BATCH DATABASE IS TO RUN THE *FIXBAT PROGRAM, EITHER OFF-LINE OR BY MODIFYING BATCHQ>PH_GO (WHICH RUNS *FIXBAT) TO RUN IT WITH A -DAYS ARGUMENT. FOR DOCUMENTATION ON *FIXBAT, SEE PE-T-622.

A FAST WAY TO GET RID OF A SINGLE QUEUE IS TO CREATE A NEW BATDEF FILE THAT DOES NOT CONTAIN THE QUEUE; THIS CAN'T BE DONE WITH DELETE, SINCE, AS WAS STATED ABOVE, THE QUEUE DOES NOT ACTUALLY GO AWAY UNTIL THE MONITOR DELETES IT. HOWEVER, IF THERE WERE 3 QUEUES DEFINED, DEFAULT, EXPRESS, AND BACKGROUND, AND IT WAS DESIRED TO IMMEDIATELY CLEAN OUT THE BACKGROUND QUEUE, THEN THE BATGEN PROGRAM WOULD BE INVOKED USING SOME NON-EXISTENT TREENAME AS THE FILE (TO START WITH A NULL ENVIRONMENT).

THEN, THE QUEUES DEFAULT AND EXPRESS WOULD BE ADDED EXACTLY AS THEY APPEARED IN THE ACTUAL "LIVE" BATDEF FILE.

THEN THE FILE COMMAND WOULD BE USED, WITH THE DESTINATION BEING BATCHQ>BATDEF.

AS SOON AS THE MONITOR SEES THAT A QUEUE IS GONE FROM BATDEF, IT WILL IMMEDIATELY ABORT ANY JOB RUNNING IN THAT QUEUE, AND WHEN THOSE ABORTIONS ARE PROCESSED, OR IF THERE WERE NO EXECUTING JOBS IN THAT QUEUE, ALL QUEUE FILES AND COMMAND FILES RELATED TO THAT QUEUE WILL BE

DELETED WHETHER OR NOT THERE ARE ANY WAITING JOBS.

V. COMPATIBILITY WITH CX

THERE IS NO COMPATIBILITY WITH CX RELATED TO OPERATOR CONTROL OVER THE SYSTEM, SINCE CX HAD NONE, WITH THE EXCEPTION OF THE ABILITY TO FORCE-LOGOUT A JOB.

HOWEVER, IF CX IS RUN SIMULTANEOUSLY WITH BATCH ON THE SAME SYSTEM, A SERIOUS SECURITY ISSUE COULD RESULT:

THAT IS, ALL CX SLAVES (JOBS) ARE LOGGED IN AS SYSTEM; AND THE BATCH SUBSYSTEM RECOGNIZES ANY USER LOGGED IN AS SYSTEM AS AN OPERATOR.

THEREFORE, A MALICIOUS USER COULD CONCEIVABLY SUBMIT A JOB TO CX THAT OBTAINED STATUS OF ALL JOBS, CANCELLED OR ABORTED OTHER USERS JOBS, OR EVEN STOP THE MONITOR ITSELF USING CX.

THIS SAME PROBLEM EXISTS WITH THE SPOOLER PHANTOM, WHICH RECOGNIZES ANY PROCESS WITH THE SAME LOGIN NAME AS IT HAS AS A "PRIVILEGED" PROCESS, AND GENERALLY THIS LOGIN NAME IS SYSTEM.

```
**********************************************************************
**********************************************************************
*
*
*
*      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    Y
*    J   J      I    M     M  M     M   Y
*    J   J      I    M     M  M     M   Y
*     JJ      III   M     M  M     M   Y
*
*
*
*    RRRR    BBBB     AAA    TTTTT    CCC    H    H  77777
*    R   R   B   B   A   A     T     C   C   H    H      7
*    R   R   B   B   A   A     T     C       H    H     7
*    RRRR    BBBB    AAAAA     T     C       HHHHH    7
*    R R     B   B   A   A     T     C       H    H   7
*    R  R    B   B   A   A     T     C   C   H    H   7
*    R   R   BBBB    A   A     T      CCC    H    H   7
*
*
*
**********************************************************************
**********************************************************************
```

BATCH - REV.17.1 - CHANGES

C_RSET & C_BDIF NOW DO 'OPEN IN.USE 1 40003' INSTEAD OF 'OPEN IN.USE 1 40003' TO AVOID AN O/S BUG THAT LIMITS THE UNIT NUMBER FROM 1 TO <MAXUNT> EVEN THOJGH '40003' SAYS OBTAIN A FREE FILE UNIT FOR READING & WRITING (<$GETV+<$RDWR).

BATCH MONITOR DOES NOT WORK IF SYSTEM DATE & TIME SET BACKWARDS VIA SETIME WHILE IT IS RUNNING. BRING IT DOWN FIRST (BATCH SYSTEM -STOP), DO THE SETIME, AND THEN BRING IT BACK UP.

```
**********************************************************************************************
**********************************************************************************************
**
**
**
**      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     Y
**    J   J     I     M    M  M    M     Y
**    J   J     I     M    M  M    M     Y
**     JJ      III    M    M  M    M     Y
**
**
**
**    RRRR          SSS   BBBB    AAA   TTTTT   CCC   H   H  4
**    R   R    _   S   S  B   B  A   A    T    C   C  H   H  4   4
**    R   R   __   S      B   B  A   A    T    C      H   H  4   4
**    RRRR   _____  SSS   BBBB   AAAAA    T    C      HHHHH  44444
**    R R     __       S  B   B  A   A    T    C      H   H      4
**    R  R    _    S   S  B   B  A   A    T    C   C  H   H      4
**    R   R         SSS   BBBB   A   A    T     CCC   H   H      4
**
**
**
**********************************************************************************************
**********************************************************************************************
```

A BATCH SUBSYSTEM FOR PRIMOS

DATE: APRIL 12, 1979

TO:

FROM:

SUBJECT: A BATCH SUBSYSTEM FOR PRIMOS

REFERENCE:

ABSTRACT

THIS DOCUMENT IS THE FUNCTIONAL SPECIFICATION FOR THE NEW BATCH SUBSYSTEM AVAILABLE FOR REVISION 17 AS A REPLACEMENT FOR THE CURRENT CX FACILITY. INCLUDED ARE SPECIFICATIONS FOR THE BATCH PROCESS AND ADMINISTRATOR, OPERATOR, AND USER INTERFACES.

1 INTRODUCTION

THE DESIGN PRESENTED IN THIS DOCUMENT IS A SUBSET OF THE DESIGN PRESENTED IN PE-T-457, A BATCH SYSTEM FOR PRIMOS IV AND V. THE BATCH SUBSYSTEM DESCRIBED HERE WILL BE DISTRIBUTED AS STANDARD USER SOFTWARE, I.E. ON THE A1 MASTER DISK PARTITION.

IT WILL BE CONTAINED IN THE UFDS BATCH (SOURCE) AND BATCHQ (DATABASE, COMMAND FILES), AND INCLUDE 4 COMMANDS IN CMDNCO (JOB, BATCH, BATGEN, AND $$). FOR MORE INFORMATION ON THE UFD STRUCTURE AND THE FILES INVOLVED, SEE PE-T-571, ADMINISTRATOR'S GUIDE TO REVISION 17 BATCH.

1.1 OVERVIEW OF CAPABILITIES

THE FOLLOWING BRIEFLY OUTLINES THE MAJOR FUNCTIONAL CAPABILITIES PROVIDED BY THE BATCH SUBSYSTEM. THE ORGANIZATION USED IS PARALLEL TO THE EXTENDED DESCRIPTION IN SECTIONS 3-5:

ADMINISTRATIVE CAPABILITIES
OPERATIONAL CAPABILITIES
USER CAPABILITIES

ADMINISTRATIVE CAPABILITIES PROVIDE THE MECHANISM BY WHICH A SYSTEM ADMINISTRATOR DEFINES THE BATCH ENVIRONMENT FOR A PARTICULAR INSTALLATION -- NUMBER OF BATCH QUEUES, DEFAULT RESOURCE LIMITS, ETC. OPERATIONAL CAPABILITIES ARE THOSE EMPLOYED BY A SYSTEM OPERATOR TO START THE EXECUTION OF THE BATCH MONITOR, DISPLAY THE STATUS OF THE QUEUE(S), ABORT BATCH PROCESSES, ETC. USER CAPABILITIES IN THE GENERAL SENSE INCLUDE ALL THE CAPABILITIES OF PRIMOS. OF INTEREST IN THE CURRENT CONTEXT ARE THOSE OF JOB DEFINITION/SUBMISSION AND JOB STATUS DISPLAY/MODIFICATION.

1.1.1 ADMINISTRATIVE CAPABILITIES

DEFINITION OF THE BATCH ENVIRONMENT STARTS WITH THE SYSTEM ADMINISTRATOR. HE HAS THE ABILITY TO DEFINE FROM ONE TO SIX BATCH QUEUES. ASSOCIATED WITH EACH QUEUE ARE A NUMBER OF PARAMETERS THAT CHARACTERIZE EACH JOB IN THE QUEUE: MAXIMUM CPU TIME LIMIT, MAXIMUM ELAPSED TIME LIMIT, DEFAULT COMMAND-FILE UNIT, DEFAULT CPU TIME LIMIT, ETC.

1.1.2 OPERATIONAL CAPABILITIES

THE OPERATIONAL CAPABILITIES OF THE BATCH SYSTEM ALLOW THE INITIATION AND TERMINATION OF BATCH MONITORS. THE STATUS OF A MONITOR AND ANY OTHER RUNNING BATCH PROCESSES CAN BE DISPLAYED AT ANY TIME. INDIVIDUAL BATCH JOBS CAN BE ABORTED OR TERMINATED.

1.1.3 USER CAPABILITIES

JOBS CAN BE SUBMITTED TO THE BATCH SYSTEM BY THE JOB COMMAND. THE PHYSICAL FORM OF THE JOB ITSELF IS A COMMAND INPUT FILE, OPTIONALLY INCLUDING JOB INFORMATION. A USER CAN DETERMINE THE STATUS OF HIS JOB AND WHICH QUEUE IT IS IN, INCLUDING JOB PARAMETERS SUCH AS CPU TIME LIMIT, FILE UNIT TO BE USED FOR COMMAND INPUT, ACCOUNTING INFORMATION, ETC. HE CAN RESTART, CANCEL, OR ABORT ANY JOB SUBMITTED UNDER HIS LOGIN NAME.

1.2 SYSTEM REQUIREMENTS

THE BATCH SYSTEM SPECIFIED HEREIN IS INTENDED TO RUN UNDER PRIMOS IV OR V. THE SYSTEM CURRENTLY REQUIRES THE AVAILABILITY OF AT LEAST 2 PHANTOMS; ONE PROCESS WILL BE DEDICATED TO THE BATCH MONITOR, THE OTHER WILL BE USED TO RUN A USER'S BATCH JOB WHEN NEEDED.

1.3 A NOTE ON SYNTAX

THIS DOCUMENT CONTAINS DESCRIPTIONS FOR MANY COMMANDS, SUBCOMMANDS, AND OPTIONS. THE SYNTAX DESCRIBED HERE IS USED UNIFORMLY IN THESE DESCRIPTIONS. IT SHOULD BE UNDERSTOOD THAT THE COMMAND SYNTAX EMPLOYED BY THE IMPLEMENTED BATCH SYSTEM MAY BE MODIFIED OR ENHANCED AS A FUNCTION OF FUTURE CHANGES IN THE PRIMOS COMMAND ENVIRONMENT.

COMMAND LINE VARIABLES

LESS-THAN (<) AND GREATER-THAN (>) SIGNS SURROUNDING A STRING INDICATE A VARIABLE FOR WHICH SPECIFIC INFORMATION IS TO BE SUBSTITUTED, FOR EXAMPLE:

<FILENAME>

SHOULD BE REPLACED WITH A VALID FILENAME.

OPTIONAL PARAMETERS

BRACKETS ENCLOSE OPTIONAL PARAMETERS FOR A COMMAND OR OPTION, FOR EXAMPLE:

-DISPLAY [ALL]

THE -DISPLAY OPTION ACCEPTS THE OPTIONAL SPECIFIER 'ALL'.

ALTERNATIVE OPERAND SPECIFICATION, DEFAULTS

WHEN AN OPERAND HAS MORE THAN ONE POSSIBLE SPECIFICATION, CHOICES ARE STACKED. A DEFAULT OPTION, IF ANY, IS UNDERSCORED, FOR EXAMPLE:

DISPLAY <QUEUE-NAME>
<u>ALL</u>

THE DISPLAY COMMAND ACCEPTS A <QUEUE-NAME> OR 'ALL'. 'ALL' IS THE DEFAULT.

NUMBER BASE

ALL NUMERIC SPECIFICATIONS IN THIS DOCUMENT ARE EXPRESSED AS DECIMAL NUMBERS. THE BATCH SUBSYSTEM DEALS ONLY WITH DECIMAL NUMBERS.

1.4 DOCUMENT OVERVIEW

SECTION 2 DEFINES SOME BASIC TERMINOLOGY REQUIRED FOR PRESENTATION OF THE BATCH SYSTEM. SECTIONS 3-5 PRESENT SPECIFIC CAPABILITIES OF THE SYSTEM AS SEEN BY AN ADMINISTRATOR, OPERATOR, AND USER. APPENDIX A SUMMARIZES THE BATCH SYSTEM COMMANDS AND OPTIONS PRESENTED IN SECTIONS 3-5. APPENDIX B IS AN ALPHABETICALLY SORTED LIST OF MESSAGES DISPLAYED BY THE BATCH SYSTEM FOLLOWED BY THE SEVERITY LEVEL AND AN EXPLANATION OF THE MESSAGE.

2 CONCEPTS AND TERMINOLOGY

THE TOTALITY OF THE BATCH ENVIRONMENT IS COMPRISED OF OBJECTS OF SEVERAL DIFFERENT TYPES, SOME WHOSE CHARACTERISTICS ARE WELL UNDERSTOOD IN THE CONTEXT OF PRIMOS AND SOME THAT ARE PARTICULAR TO THE BATCH SYSTEM. TO ESTABLISH A COMMON FRAMEWORK IN WHICH THE BATCH SYSTEM CAN BE DESCRIBED, SOME PRELIMINARY DEFINITIONS ARE REQUIRED.

2.1 BATCH JOB

A BATCH JOB, OR SIMPLY JOB, IS THE BASIC UNIT OF WORK IN THE BATCH SYSTEM AND THE UNIT OF COMMUNICATION BETWEEN A BATCH USER AND THE BATCH SYSTEM.

2.1.1 COMPONENTS OF A JOB

THE PHYSICAL FORM OF A JOB IS A SEQUENTIAL FILE THAT CONTAINS PRIMOS COMMAND LINES AND AN OPTIONAL LINE OF BATCH-SPECIFIC INFORMATION AS THE FIRST NON-COMMENT LINE IN THE FILE.

NOTE: THE BATCH SYSTEM RESERVES THE USE OF '$$' IN THE FIRST TWO CHARACTER POSITIONS OF THE COMMAND LINE. THE ONLY CURRENTLY DEFINED BATCH COMMAND USING THIS CONSTRUCT IS $$ JOB. THIS IS DEFINED IN DETAIL IN SECTION 5.1.

THE PRIMOS COMMANDS IN A JOB ARE INDISTINGUISHABLE FROM THOSE AVAILABLE TO AN INTERACTIVE USER RUNNING A COMMAND FILE OR A PHANTOM JOB. A JOB CONTAINING ONLY PRIMOS COMMAND LINES IS EQUIVALENT TO A COMMAND INPUT FILE AND WILL BE PROCESSED IN THE SAME MANNER.

ANY STANDARD PRIMOS COMMAND FILE MAY BE SUBMITTED TO THE BATCH SYSTEM. THE REVERSE IS ALSO TRUE; ANY BATCH FILE, EVEN IF IT CONTAINS THE $$ JOB COMMAND AS THE FIRST NON-COMMENT LINE, WILL RUN SUCCESSFULLY AS A PRIMOS COMMAND FILE. THE $$ JOB LINE WILL BE IGNORED.

2.1.2 JOB NAMING AND IDENTIFICATION

ASSOCIATED WITH THE JOB IS AN INTERNAL NAME ASSIGNED BY THE BATCH SYSTEM. THE INTERNAL JOB NAME IS GENERATED AUTOMATICALLY BY THE BATCH SYSTEM AND IS OF THE FORM '#SNNNN', WHERE 'S' IS THE NUMBER OF THE QUEUE AND 'NNNN' IS A SEQUENCE NUMBER WITHIN THAT QUEUE.

ALSO ASSOCIATED WITH EACH JOB IS AN EXTERNAL NAME. THE NAME OF THE COMMAND FILE ITSELF (I.E., NOT INCLUDING ANY PATHNAME INFORMATION) BECOMES THIS EXTERNAL NAME. FOR EXAMPLE, IF THE COMMAND FILE COBCL>C_RUN WERE SUBMITTED, THE EXTERNAL NAME OF THE JOB WOULD BE C_RUN.

REFERENCE TO A JOB IN THE BATCH SYSTEM IS VIA INTERNAL OR EXTERNAL NAME.

IF THE USER REFERENCING HIS JOB HAS MORE THAN ONE ACTIVE JOB IN THE BATCH SYSTEM WITH THE SAME EXTERNAL NAME, HOWEVER, THE INTERNAL NAME MUST BE USED INSTEAD TO RESOLVE AMBIGUITIES, OR THE ERROR MESSAGE "MULTIPLE JOBS WITH THIS NAME (USE INTERNAL NAME)" WILL RESULT. THE EXCEPTIONS TO THIS RULE ARE THE COMMANDS THAT DISPLAY JOB INFORMATION, WHICH WILL DISPLAY ALL APPROPRIATE JOBS WITH THE SPECIFIED EXTERNAL NAME.

IN GENERAL, A USER CAN ONLY REFERENCE OR OBTAIN STATUS ON JOBS THAT WERE SUBMITTED UNDER THE SAME LOGIN NAME. THE EXCEPTION IS THE BATCH -DISPLAY COMMAND, WHICH WILL DISPLAY ALL EXECUTING JOBS.

ALSO, IF THE USER IS LOGGED IN AS SYSTEM, ALL JOBS CAN BE REFERENCED, NOT JUST THOSE BELONGING TO THAT USER. HOWEVER, ANY JOBS NOT BELONGING TO THAT USER CAN ONLY BE REFERENCED BY INTERNAL NAME.

2.2 BATCH QUEUE

A BATCH QUEUE IS A CHANNEL THROUGH WHICH JOBS FLOW FROM USERS MAKING JOB SUBMISSIONS TO BATCH PROCESSES THAT EXECUTE THE JOBS. A BATCH QUEUE IS A MULTI-WRITER/MULTI-READER QUEUE AND INCLUDES THREE DISTINCT SECTIONS OR TYPES OF INFORMATION.

THE BATCH QUEUE DEFINITION CONTAINS THE NAME OF THE QUEUE AND THE GENERIC CHARACTERISTICS OF THE JOBS IN THE QUEUE. THE NAME OF THE QUEUE IS A CHARACTER STRING BY WHICH THE QUEUE IS KNOWN TO THE SYSTEM ADMINISTRATOR, OPERATOR, AND USERS. THE CHARACTERISTICS OF THE QUEUE -- MAXIMUM CPU TIME, DEFAULT FILE-UNIT, ETC. -- PROVIDE A SET OF CRITERIA THAT IS USED TO DETERMINE THE ADMISSABILITY OF A JOB TO THE QUEUE.

THE BATCH QUEUE STATUS CONTAINS PARAMETERS AND FLAGS THAT DESCRIBE THE CURRENT STATUS OF THE QUEUE. EXAMPLES ARE NUMBER OF WAITING JOBS IN QUEUE, QUEUE STATUS (BLOCKED/UNBLOCKED/FLAGGED FOR DELETION), ETC.

THE JOB QUEUE ASSOCIATED WITH THE QUEUE IS A DATA STRUCTURE THAT CONTAINS THE PHYSICAL JOBS SUBMITTED TO THE QUEUE. WITH EACH JOB ARE PARAMETERS THAT DESCRIBE THE LIMITS OF THE JOB.

AN INSTALLATION CAN DEFINE FROM ONE TO SIX BATCH QUEUES.

2.3 BATCH SUBMISSION

A BATCH OR JOB SUBMISSION IS THE ADDITION OF A USER JOB TO THE JOB QUEUE OF A BATCH QUEUE. JOB SUBMISSION PROCEEDS IN TWO LOGICALLY DISTINCT PHASES. THE FIRST PHASE IS THE JOB COMMAND, INCLUDING DATA READ FROM THE (OPTIONAL) $$ JOB LINE IN THE COMMAND FILE.

CONFLICTING OPTIONS ON THE JOB AND $$ JOB LINE WILL BE RESOLVED WITH JOB OPTIONS TAKING PRECEDENCE. THE JOB COMMAND IS ALSO CAPABLE OF ABORTING, CANCELLING, RESTARTING, HOLDING, RELEASING, AND OBTAINING STATUS OF JOBS.

ONCE A JOB'S FINAL PARAMETERS HAVE BEEN DETERMINED, THE ACTUAL SUBMITTAL OF THE JOB TO THE BATCH QUEUE (POSSIBLY SPECIFIED BY THE JCB), HANDLING ERROR CONDITIONS SUCH AS CPU TIME GREATER THAN THE MAXIMUM DEFINED FOR THE QUEUE, ETC., IS PERFORMED.

2.4 BATCH MONITOR

THE BATCH MONITOR IS A PRIMOS PROCESS WHOSE SOLE TASK IT IS TO START OTHER PROCESSES TO EXECUTE BATCH JOBS. THESE BATCH PROCESSES RUN AS PHANTOMS.

WHEN A BATCH PROCESS IS INITIATED, IT IS GIVEN THE SAME LOGIN NAME AS THAT OF THE SUBMITTING USER. IT FIRST RUNS A PROGRAM THAT MAKES APPROPRIATE ENTRIES IN THE STATUS BLOCKS FOR THE QUEUE TO WHICH IT HAS BEEN ASSIGNED. IT THEN DETERMINES WHICH COMMAND FILE TO INVOKE, AND ON WHICH FILE UNIT IT IS TO BE INVOKED ON, AND EXECUTES IT.

2.5 BATCH ABORT HANDLING

A BATCH JOB CAN BE LOGGED OUT IN ONE OF TWO WAYS; VIA A FATAL ERROR (E.G., DISK FULL, INSUFFICIENT ACCESS RIGHT), OR WHEN IT REACHES THE END OF ITS COMMAND FILE.

IF IT LOGS OUT IN THE FIRST MANNER, THE STATUS OF THE JOB WILL BECOME "ABORTED". IF IT REACHES THE END OF ITS COMMAND FILE SUCCESSFULLY, IT WILL BE KNOWN AS "COMPLETED". THE ONLY DIFFERENCE IS THAT THE LAST THING THE COMMAND FILE WILL DO IS TO RUN THE BATCH PROGRAM, WHICH WILL CHANGE A STATUS CODE BEFORE LOGGING ITSELF OUT.

EITHER WAY, THE JOB'S TERMINATION MUST BE DISCOVERED BY THE BATCH ABORT HANDLER.

THE BATCH ABORT HANDLER IS CURRENTLY IMPLEMENTED IN THE BATCH MONITOR, WHICH WILL PERIODICALLY ATTEMPT TO OPEN THE COMMAND FILES OF ALL EXECUTING JOBS FOR WRITING. WHENEVER IT SUCCEEDS, IT ASSUMES THAT THE PARTICULAR BATCH JOB TO WHICH THE COMMAND FILE BELONGED HAS TERMINATED. IF THE BATCH PROGRAM WAS RUN TO FLAG SUCCESSFUL COMPLETION OF THE JOB,

THE STATUS OF THE JOB BECOMES "COMPLETED", OTHERWISE IT BECOMES "ABORTED".

AT THIS POINT, THE BATCH MONITOR WILL CLEAN UP AFTER THE JOB AS APPROPRIATE, FLAGGING ITS ABORT/COMPLETE STATUS, DELETING ITS TEMPORARY COMMAND FILE, ETC.

IF THE JOB HAS BEEN FLAGGED AS BEING RESTARTABLE, IT WILL LEAVE ITS STATUS AS "WAITING", AND WILL INDICATE IN THE JOB STATUS INFORMATION THAT THE JOB HAS ALREADY EXECUTED.

2.6 BATCH SYSTEM DATABASE

THE DATABASE FOR THE BATCH SYSTEM CONTAINS INFORMATION RELATING TO EACH BATCH QUEUE -- DEFINITION, STATUS, AND JOB QUEUE -- AND INFORMATION ABOUT EACH ACTIVE BATCH PROCESS.

IN THE FOLLOWING, THE TOTALITY OF THE BATCH SYSTEM DATABASE WILL BE REFERRED TO AS BATCHQ.

3 ADMINISTRATIVE FUNCTIONS

THE SYSTEM ADMINISTRATOP DEFINES THE BATCH ENVIRONMENT FOR AN INSTALLATION IN TERMS OF THE NUMBER OF BATCH QUEUES AND THEIR CHARACTERISTICS.

3.1 DEFINING A BATCH ENVIRONMENT -- THE BATGEN COMMAND

THE FIRST TASK OF THE SYSTEM ADMINISTRATOR IS TO DEFINE THE CHARACTERISTICS OF EACH BATCH QUEUE TO WHICH USERS CAN SUBMIT JOBS. THE VEHICLE FOR THIS SPECIFICATION IS THE BATGEN COMMAND, FOR BATCH GENERATION. BATGEN IS AN INTERACTIVE PROGRAM THAT ALLOWS THE ADMINISTRATOR TO DEFINE NEW BATCH QUEUES, MODIFY OR DISPLAY THE CHARACTERISTICS OF EXISTING QUEUES, OR UNDEFINE QUEUES.

3.1.1 INVOKING BATGEN

TO RUN BATGEN, THE ADMINISTRATOR LOGS INTO PRIMOS AND TYPES:

```
BATGEN <TREENAME>
```

TO OPERATE ON THE "LIVE" BATCH DEFINITION FILE, THE TREENAME BATCHQ>BATDEF SHOULD BE GIVEN. HOWEVER, THERE MAY BE A PASSWORD ON BATCHQ. IF SO, IT MUST BE INCLUDED IN THE TREENAME.

NOTE: THIS COMMAND CANNOT BE EXECUTED UNLESS THE SYSTEM DATE AND TIME HAVE BEEN SET VIA THE OPERATOR SETIME COMMAND.

THIS FILE WILL NOT BE MODIFIED UNTIL THE FILE COMMAND (DESCRIBED BELOW) IS GIVEN, AS ALL OTHER COMMANDS OPERATE ON A COPY OF THE FILE THAT EXISTS IN MEMORY ONLY.

AN ALTERNATE WAY OF INVOKING BATGEN IS AS FOLLOWS:

```
BATGEN [<TREENAME>] <OPTION>
                    -STATUS
                    -DISPLAY
```

IF <TREENAME> IS NOT SPECIFIED, THE DEFAULT "BATCHQ>BATDEF" IS USED. THIS FORMAT OF BATGEN IS A QUICK WAY OF DETERMINING THE CURRENT CONFIGURATION WITHOUT ENTERING BATGEN COMMAND MODE. THE STATUS OR DISPLAY COMMAND IS EXECUTED ON THE TREENAME SPECIFIED AND THE USER IS RETURNED TO PRIMOS COMMAND MODE.

THIS FORMAT IS PRIMARILY INTENDED FOR USERS WHO NEED TO KNOW THE NAMES OF AVAILABLE QUEUES AND THE STATUS OF EACH QUEUE (BATGEN -STATUS IS RECOMMENDED HERE), OR WHO NEED TO KNOW WHAT THE DEFAULT AND MAXIMUM PARAMETERS ON THE QUEUES ARE (BATGEN -DISPLAY WILL SUPPLY THIS INFORMATION). THE STATUS AND DISPLAY COMMANDS ARE

DESCRIBED BELOW.

NOTE THAT FOR THE COMMAND "BATGEN -STATUS" TO WORK, THE FILE BATCHQ>BATDEF MUST BE READ-PERMITTED (I.E., IF THERE IS A PASSWORD ON BATCHQ, THE PROTECTION OF BATDEF SHOULD BE 7 1).

3.1.2 BATGEN COMMANDS

ONCE <TREENAME> HAS BEEN READ IN AND VALIDATED, BATGEN TYPES A PROMPT CHARACTER AND WAITS FOR COMMANDS. VALID COMMANDS AT THIS LEVEL ARE ADD, MODIFY, DELETE, DISPLAY, STATUS, BLOCK, UNBLOCK, FILE, AND QUIT.

IF <TREENAME> DOES NOT EXIST, THE ERROR MESSAGE "NOT FOUND" WILL BE GENERATED, AND BATGEN WILL INITIALIZE ITS TABLES THAT ARE TO BE ASSOCIATED WITH THE FILE TO INDICATE THAT NO QUEUES ARE CONFIGURED, AND ENTER BATGEN COMMAND MODE. IF ANY OTHER ERROR OCCURS WHILE TRYING TO OPEN <TREENAME> FOR READING, A FATAL ERROR MESSAGE WILL BE PRINTED AND THE USER WILL BE RETURNED TO PRIMOS COMMAND MODE.

ADD COMMAND -- DEFINE NEW BATCH QUEUE

COMMAND FORMAT:

ADD <QUEUE-NAME>

<QUEUE-NAME> A NAME TO BE ASSOCIATED WITH THE QUEUE (32 CHARACTERS MAXIMUM).

COMMAND ACTION:

THE ADD COMMAND SPECIFIES THAT A NEW BATCH QUEUE IS TO BE DEFINED. <QUEUE-NAME> IS THE NAME TO BE ASSOCIATED WITH THE QUEUE. ALL FURTHER REFERENCES TO THE QUEUE BY ADMINISTRATOR, OPERATOR, OR USER USE <QUEUE-NAME> AS AN IDENTIFIER. IF <QUEUE-NAME> IS THE NAME OF A CURRENTLY DEFINED QUEUE, AN ERROR MESSAGE ("ALREADY EXISTS") IS PRINTED.

THIS NAME MUST CONFORM TO STANDARD PRIMOS FILENAME RULES. THE NAME "ALL" IS ILLEGAL, TO AVOID POSSIBLE AMBIGUITIES WITH COMMANDS SUCH AS "DELETE" THAT INTERPRET "ALL" AS REFERRING TO ALL QUEUES.

IN THE ABSENCE OF ANY ERROR CONDITIONS, THE ADD COMMAND THEN PROMPTS THE USER FOR SUBCOMMANDS SPECIFYING THE CHARACTERISTICS OF THE QUEUE:

ENTER QUEUE CHARACTERISTICS:

ANY OF THE SUBCOMMANDS DESCRIBED IN SECTION 3.1.3 BELOW CAN THEN

BE ENTERED.

MODIFY COMMAND -- MODIFY A QUEUE'S CHARACTERISTICS

COMMAND FORMAT:

MODIFY <QUEUE-NAME>

<QUEUE-NAME> THE NAME OF THE QUEUE WHOSE CHARACTERISTICS ARE TO BE MODIFIED.

COMMAND ACTION:

A CHECK IS MADE TO SEE IF THE QUEUE NAME EXISTS. IF IT DOESN'T, AN ERROR MESSAGE IS PRINTED. IN THE ABSENCE OF ERRORS, THE MODIFY COMMAND TYPES:

ENTER QUEUE CHARACTERISTICS:

AND WAITS FOR SUBCOMMANDS. THESE SUBCOMMANDS ARE THE SAME AS THOSE FOR THE ADD COMMAND -- SEE SECTION 3.1.3 BELOW.

DELETE COMMAND -- DELETE A QUEUE DEFINITION

COMMAND FORMAT:

DELETE <QUEUE-NAME>

<QUEUE-NAME> THE NAME OF THE QUEUE WHOSE DEFINITION IS TO BE DELETED.

COMMAND ACTION:

THE DEFINITION OF THE QUEUE <QUEUE-NAME> IS DELETED. AN ERROR MESSAGE IS GENERATED IF THE QUEUE DOES NOT EXIST.

THE QUEUE DOES NOT ACTUALLY "DISAPPEAR" - INSTEAD, ITS STATUS BECOMES "FLAGGED FOR DELETION", REPLACING THE STATUS THAT IT PREVIOUSLY HAD. WHEN THE BATCH MONITOR SEES THAT A CURRENTLY DEFINED QUEUE HAS BEEN "FLAGGED FOR DELETION", IT WILL DELETE ALL QUEUE CONTROL FILES AND COMMAND FILES RELATED TO THAT QUEUE ONLY WHEN THERE ARE NO ACTIVE (I.E., RUNNING, HELD OR WAITING) JOBS IN THAT QUEUE.

UNTIL THE QUEUE IS ACTUALLY DELETED, IT WILL SHOW UP IN STATUS AND DISPLAY COMMANDS AS BEING "FLAGGED FOR DELETION". WHEN IT IS DELETED, A MESSAGE WILL BE SENT TO THE SYSTEM CONSOLE. THE MESSAGE WILL BE EITHER "QUEUE <QUEUE-NAME> DELETED" OR "REMOVED <QUEUE-NAME> FROM BATDEF". THE FIRST MESSAGE INDICATES THAT SOME

JOB DATA WAS ACTUALLY DELETED, THE SECOND INDICATES THAT NO JOBS WERE EVER SUBMITTED TO THAT QUEUE, SO THAT ONLY THE BATDEF FILE WAS CHANGED.

DISPLAY COMMAND -- DISPLAY A QUEUE'S CHARACTERISTICS

COMMAND FORMAT:

```
DISPLAY [<QUEUE-NAME>]
        [    ALL     ]
```

<QUEUE-NAME> THE NAME OF THE QUEUE WHOSE CHARACTERISTICS ARE TO BE DISPLAYED. IF OMITTED, ALL QUEUES ARE DISPLAYED.

COMMAND ACTION:

THE CHARACTERISTICS OF THE SELECTED QUEUE ARE DISPLAYED. A SAMPLE DISPLAY IS SHOWN IN SECTION 3.1.4. THE CHARACTERISTICS DISPLAYED WILL REFLECT ANY CHANGES MADE DURING THE CURRENT BATGEN SESSION.

STATUS COMMAND - QUICK STATUS OF QUEUES

COMMAND FORMAT:

```
STATUS
```

COMMAND ACTION:

THE STATUS OF ALL QUEUES IS DISPLAYED IN TABULAR FORMAT.

BLOCK COMMAND -- DISALLOW SUBMISSIONS TO A QUEUE

COMMAND FORMAT:

```
BLOCK <QUEUE-NAME>
        ALL
```

<QUEUE-NAME> THE NAME OF THE QUEUE TO BE BLOCKED. 'ALL' APPLIES TO ALL QUEUES.

COMMAND ACTION:

A FLAG IS SET IN THE STATUS CONTROL BLOCK ASSOCIATED WITH <QUEUE-NAME> INDICATING THAT NO FURTHER JOBS ARE TO BE ADDED TO THE JOB QUEUE ASSOCIATED WITH <QUEUE-NAME>. AN ATTEMPT BY A USER TO ADD A JOB TO A BLOCKED QUEUE WILL RESULT IN AN ERROR MESSAGE.

(NOTE: BLOCK PREVENTS ADDITIONS TO A QUEUE. JOB INITIATIONS FROM A QUEUE WILL STILL OCCUR AS LONG AS THERE ARE ELIGIBLE JOBS IN THAT QUEUE.)

UNBLOCK COMMAND -- ALLOW SUBMISSIONS TO A QUEUE

COMMAND FORMAT:

UNBLOCK <QUEUE-NAME>
ALL

<QUEUE-NAME> THE NAME OF THE QUEUE TO BE UNBLOCKED. 'ALL' WILL UNBLOCK ALL CURRENTLY DEFINED QUEUES.

COMMAND ACTION:

THE BLOCKED FLAG IN THE STATUS CONTROL BLOCK OF THE SELECTED QUEUE(S) IS CLEARED. SUBMISSIONS TO THE QUEUE ARE ACCEPTED.

FILE COMMAND -- ESTABLISH NEW BATCH QUEUE DEFINITIONS

COMMAND FORMAT:

FILE [<TREENAME>]

<TREENAME> AN OPTIONAL TREENAME SPECIFYING THE NAME OF THE BATCH DEFINITION FILE TO BE CREATED. IF NOT SPECIFIED, THE TREENAME ENTERED WHEN THE BATGEN COMMAND WAS GIVEN IS USED.

COMMAND ACTION:

THE FILE COMMAND ESTABLISHES THE EFFECT OF ALL PRECEDING BATGEN COMMANDS: A NEW BATCH DEFINITION FILE IS WRITTEN TO DISK. IF BATGEN IS MODIFYING BATCH0>BATDEF, THE NEW DEFINITIONS BECOME EFFECTIVE IMMEDIATELY.

QUIT COMMAND -- EXIT BATGEN WITH NO BATCH MODIFICATIONS

COMMAND FORMAT:

QUIT

COMMAND ACTION:

BATGEN EXITS TO PRIMOS COMMAND LEVEL MAKING NO MODIFICATIONS TO THE BATCH DEFINITION FILE. A SUBSEQUENT 'START' COMMAND WILL REENTER THE BATGEN ENVIRONMENT WITH NO LOSS OF INFORMATION FROM THE SESSION QUITED. IF MODIFICATIONS HAVE BEEN MADE TO THE BATCH DEFINITION FILE, USER VERIFICATION IS REQUESTED BEFORE THE QUIT IS TAKEN (AS DONE BY ED).

3.1.3 SUBCOMMANDS FOR ADD AND MODIFY COMMANDS

THE BATGEN ADD AND MODIFY COMMANDS ENTER A SUBCOMMAND ENVIRONMENT IN WHICH THE CHARACTERISTICS, RESOURCE LIMITS, AND RESOURCE DEFAULTS OF A BATCH QUEUE ARE DEFINED OR REDEFINED. THE FOLLOWING DEFINES THESE SUBCOMMANDS AND THEIR ACTIONS. FOR MANY OF THE SUBCOMMANDS, THE INITIAL VALUES ARE ALSO GIVEN. THE NAMES OF THE SUBCOMMANDS ARE ALSO THE NAMES OF THE CHARACTERISTICS USED WITH THE DISPLAY COMMAND.

CPTIME -- SET CPU TIME LIMITS

COMMAND FORMAT:

```
CPTIME <DEFAULT> [<MAXIMUM>]
         5          5
```

COMMAND ACTION:

THE CPU TIME LIMITS ASSOCIATED WITH THE QUEUE ARE SET TO <DEFAULT> SECONDS (DEFAULT) AND <MAXIMUM> SECONDS (MAXIMUM). NO JOB WHOSE TIME LIMIT EXCEEDS <MAXIMUM> WILL BE ACCEPTED FOR THIS QUEUE. IF NOT SPECIFIED, THE DEFAULT CPU TIME LIMIT FOR ANY JOB ADDED TO THE QUEUE IS SET TO <DEFAULT> SECONDS.

IF A CPTIME SUBCOMMAND IS GIVEN, BUT NO <MAXIMUM> PARAMETER IS PRESENT, THEN THE MAXIMUM CPU TIME LIMIT WILL NOT CHANGE. SPECIFYING A <DEFAULT> OR <MAXIMUM> OF "NONE" WILL CAUSE NO LIMIT AT ALL TO BE IMPOSED.

NOTE THAT SPECIFYING A DEFAULT VALUE THAT IS HIGHER THAN THE MAXIMUM VALUE WILL CAUSE THE REJECTION OF ANY JOBS THAT ARE SUBMITTED TO THE QUEUE WITHOUT AN EXPLICIT CPU TIME LIMIT, SINCE THE JOB WILL ADOPT THE DEFAULT FOR THIS QUEUE, AND THEN THAT VALUE WILL BE CHECKED AGAINST THE MAXIMUM VALUE.

ETIME -- SET ELAPSED TIME LIMIT

COMMAND FORMAT:

```
ETIME <DEFAULT> [<MAXIMUM>]
        20          20
```

COMMAND ACTION:

THE ACTION OF ETIME IS COMPLETELY ANALOGOUS TO CPTIME, WITH THE VALUES BEING EXPRESSED AS MINUTES OF ELAPSED TIME.

RLEVEL -- SET R-LEVEL DIFFERENCE

COMMAND FORMAT:

```
RLEVEL <DELTA-VAL>
          0
```

COMMAND ACTION:

THE RLEVEL FOR A BATCH PROCESS EXECUTING JOBS FROM THIS QUEUE WILL BE LOWERED BY <DELTA-VAL>. IF UNSPECIFIED, THE RLEVEL FOR ANY PROCESSES EXECUTING FROM THIS QUEUE WILL BE LEFT UNCHANGED. THIS VALUE MAY RANGE FROM 0 TO 7.

THE RLEVEL OF THE PROCESS IS ALSO DIRECTLY CHANGABLE BY USING THE OPERATOR'S CHAP COMMAND. THIS RLEVEL CANNOT BE LOWERED BELOW LEVEL 0. THE SYSTEM DEFAULT IS 1, AND THE MAXIMUM LEVEL IS 3.

TIMESLICE -- SET TIMESLICE

COMMAND FORMAT:

```
TIMESLICE <VAL>
            20
```

COMMAND ACTION:

THE TIMESLICE FOR A BATCH PROCESS EXECUTING JOBS FROM THIS QUEUE WILL BE SET TO <VAL> UNITS. IF UNSPECIFIED, THE TIMESLICE FOR ANY PROCESSES EXECUTING FROM THIS QUEUE WILL BE LEFT UNCHANGED. IT WILL ALSO REMAIN UNCHANGED IF THE SPECIFIED VALUE IS LARGER THAN THE TIMESLICE OF THE BATCH MONITOR AT THE TIME A PARTICULAR JOB FROM THIS QUEUE IS SPAWNED, I.E. THE TIMESLICE OF A JOB CAN NOT BE RAISED, AND ANY ATTEMPT TO DO SO WILL BE IGNORED. THIS VALUE MAY RANGE FROM 1 TO 99.

FUNIT -- SET DEFAULT COMMAND INPUT UNIT

COMMAND FORMAT:

FUNIT <UNIT-NUMBER>
6

COMMAND ACTION:

THE DEFAULT FILE UNIT TO BE USED FOR COMMAND INPUT ASSOCIATED WITH JOBS IN THE QUEUE IS DEFINED. THE DEFAULT FILE UNIT CAN BE OVERRIDDEN BY THE USER'S JOB COMMAND WHEN A JOB IS SUBMITTED, OR BY THE (OPTIONAL) $$ JOB SPECIFIER IN THE COMMAND FILE.

THE RANGE OF THE FILE UNIT IS FROM 1 TO 126, OR WHATEVER THE HIGHEST LEGAL UNIT NUMBER ON THE SYSTEM IS (A COLD-START CONFIG DIRECTIVE).

PRIORITY -- SET DEFAULT JOB QUEUE PRIORITY

COMMAND FORMAT:

PRIORITY <VAL>
5

COMMAND ACTION:

THE PRIORITY COMMAND SETS THE DEFAULT JOB QUEUE PRIORITY FOR THE CURRENT QUEUE. ANY SUBMITTED JOB THAT DID NOT SPECIFY A -PRIORITY VALUE WILL DEFAULT TO <VAL>. THIS VALUE RANGES FROM 0 TO 9.

RETURN -- RETURN FROM ADD/MODIFY COMMAND

COMMAND FORMAT:

RETURN

COMMAND ACTION:

THE ADD OR MODIFY SUBCOMMAND ENVIRONMENT IS LEFT. ANY QUEUE ADDITIONS OR MODIFICATIONS ARE SAVED FOR POSSIBLE USE BY BATGEN'S FILE COMMAND.

QUIT -- ABORT ADD/MODIFY COMMAND

COMMAND FORMAT:

QUIT

COMMAND ACTION:

THE ADD OR MODIFY SUBCOMMAND ENVIRONMENT IS LEFT. ANY QUEUE ADDITIONS OR MODIFICATIONS ARE DISCARDED. BEFORE THE QUIT IS TAKEN, USER VERIFICATION IS REQUESTED (AS DONE BY ED).

3.1.4 BATGEN EXAMPLE

IN THE FOLLOWING SAMPLE BATGEN SESSION TWO BATCH QUEUES ARE DEFINED THAT MIGHT BE USEFUL IN A UNIVERSITY ENVIRONMENT. THE FIRST, EXPRESS, IS INTENDED FOR THE USE OF A LARGE NUMBER OF STUDENTS SUBMITTING SHORT JOBS. THE SECOND QUEUE, PAYROLL, IS INTENDED SOLELY FOR THE PROCESSING OF A PAYROLL. USER INPUT IS UNDERLINED; COMMENTS ARE OFFSET BY '/*'. EXCEPT FOR THE ILLUSTRATIVE ERROR, ALL INPUT COULD COME FROM A COMMAND FILE.

```
OK, BATGEN BATCHQ>BATDEF
GO
[BATGEN REV 17.0]
> ADD_EXPRESS
ENTER QUEUE CHARACTERISTICS:
$ CPTIME_2              /* 2 SECONDS MAX CPU TIME (DEFAULT)
$ ETIME_5               /* 5 MINUTES MAX ELAPSED TIME (DEFAULT)
$ PRIORITY_4            /* DEFAULT QUEUE PRIORITY IS 4
$ RETURN                /* SAVE QUEUE DEFINITION
> ADD_EXPRESS           /* WRONG QUEUE NAME
QUEUE EXPRESS ALREADY EXISTS.
> ADD_PAYROLL           /* START NEW QUEUE DEFINITION
ENTER QUEUE CHARACTERISTICS:
$ CPTIME_NONE_NONE      /* NO LIMITS
$ ETIME__NONE_NONE
$ FUNIT_126             /* UNIT 126 FOR COMMAND INPUT
$ PRIORITY_9            /* DEFAULT QUEUE PRIORITY IS 9
$ RLEVEL_2              /* LOWER RLEVEL BY 2 UNITS
$ RETURN
> DISPLAY               /* DISPLAY QUEUE DEFINITIONS

QUEUE NAME = EXPRESS, UNBLOCKED.
DEFAULT CPTIME=2, ETIME=5, PRIORITY=4;
MAXIMUM CPTIME=5, ETIME=20; FUNIT=6;
DELTA RLEVEL=0; TIMESLICE=20;

QUEUE NAME = PAYPOLL, UNBLOCKED.
DEFAULT CPTIME=NONE, ETIME=NONE, PRIORITY=9;
MAXIMUM CPTIME=NONE, ETIME=NONE; FUNIT=126;
```

```
DELTA RLEVEL=2; TIMESLICE=20;

> MODIFY_EXPRESS    /* MODIFY QUEUE DEFINITION
ENTER QUEUE CHARACTERISTICS:
$ CPTIME_3          /* SET CPU TIME LIMIT TO 3 SECONDS
$ RLEVEL_1          /* LOWER RLEVEL BY 1 UNIT
$ TIMESLICE_40      /* SET TIMESLICE TO 40 UNITS
$ RETURN
> DISPLAY_EXPRESS   /* CHECK MODIFICATION

QUEUE NAME = EXPRESS, UNBLOCKED.
DEFAULT CPTIME=3, ETIME=5, PRIORITY=4;
MAXIMUM CPTIME=5, ETIME=20; FUNIT=6;
DELTA RLEVEL=1; TIMESLICE=40;

> FILE              /* ESTABLISH BATCH QUEUE DEFINITIONS

OK.
```

4 OPERATIONAL FUNCTIONS

WHILE A SYSTEM ADMINISTRATOR HAS THE RESPONSIBILITY FOR DEFINING THE BATCH ENVIRONMENT IN TERMS OF BATCH QUEUES, AN OPERATOR HAS THE TASK OF INITIATING AND MONITORING THE EXECUTION OF BATCH JOBS BY THE BATCH MONITOR.

THE BOUNDARY BETWEEN THESE TWO FUNCTIONS IS NOT NECESSARILY CLEAR CUT. FOR EXAMPLE, AN ADMINISTRATOR MAY EFFECTIVELY INITIATE BATCH PROCESSING BY INCLUDING IN THE COLD START COMMAND FILE (C_PRMO) COMMANDS TO START THE BATCH MONITOR: AN OPERATOR MAY DECIDE TO BLOCK A PARTICULAR BATCH QUEUE USING THE BATGEN COMMAND. IN GENERAL, HOWEVER, THE OPERATOR IS CHIEFLY CONCERNED WITH THE DAY-TO-DAY OPERATION OF THE BATCH SYSTEM.

THE OPERATOR HAS TWO TOOLS WITH WHICH TO CONTROL THE BATCH SYSTEM -- THE BATCH COMMAND AND THE JOB COMMAND. THE BATCH COMMAND IS USED TO INITIATE AND TERMINATE THE BATCH MONITOR. THE BATCH QUEUES THAT ARE ACTIVE AND CURRENTLY EXECUTING JOBS CAN ALSO BE DISPLAYED.

THE JOB COMMAND (WHICH IS ALSO USED FOR USER JOB SUBMISSIONS) IS USED TO COMMUNICATE WITH AND CONTROL PARTICULAR JOBS. JOBS CAN BE HELD, RELEASED, ABORTED OR RESTARTED. THE CHARACTERISTICS OF JOBS CAN BE DISPLAYED AND MODIFIED.

4.1 CONTROLLING THE BATCH MONITOR -- THE BATCH COMMAND

THE BATCH COMMAND CONTROLS THE BATCH MONITOR (NOT JOBS). THE BATCH COMMAND MUST BE USED TO TELL THE MONITOR TO START UP. THE COMMANDS TO START UP THE MONITOR MAY BE IN THE C_PRMO FILE, AND ARE DESCRIBED IN RF-T-571, ADMINISTRATOR'S GUIDE TO REVISION 17 BATCH.

BATCH COMMANDS TO DISPLAY THE STATUS OF THE BATCH SYSTEM CAN BE ISSUED BY THE SYSTEM OPERATOR.

4.1.1 GENERAL FORM OF BATCH COMMAND

THE GENERAL FORM OF THE BATCH COMMAND IS:

BATCH SYSTEM [<OPTION>]

<OPTION> ONE OF THE OPTIONS DESCRIBED BELOW.

THE BATCH MONITOR MUST BE LOGGED IN AS SYSTEM. IN THE FUTURE, THE BATCH MONITOR(S) WILL BE LOGGED IN UNDER NAMES OTHER THAN SYSTEM, THUS THE REQUIREMENT TO INCLUDE THE MONITOR'S USER-NAME ON THE BATCH COMMAND LINE IS PRESENT TO ALLOW UPWARD COMPATIBILITY.

NOTE: THIS COMMAND CANNOT BE EXECUTED UNLESS THE SYSTEM DATE AND TIME HAVE BEEN SET VIA THE OPERATOR SETIME COMMAND.

4.1.2 BATCH COMMAND OPTIONS

IN THE COMMAND FORMATS THAT FOLLOW, 'BATCH SYSTEM' IS ASSUMED TO START THE COMMAND LINE, WITH THE EXCEPTION OF THE -DISPLAY OPTION, IN WHICH CASE THE SYSTEM SPECIFIER IS OPTIONAL AND IGNORED IF PRESENT.

-START -- START BATCH PROCESSING

COMMAND FORMAT:

-START

COMMAND ACTION:

THE -START OPTION IS USED TO INITIATE THE BATCH MONITOR. THE SEQUENCE OF ACTIONS TO START IT UP IS AS FOLLOWS:

THE COMMANDS TO INITIATE THE MONITOR MUST ALREADY HAVE BEEN ISSUED FROM THE SYSTEM CONSOLE. THE MONITOR WILL ENSURE THAT ANOTHER BATCH MONITOR IS NOT ALREADY RUNNING. IF ONE IS, AN ERROR MESSAGE WILL BE SENT TO THE SYSTEM CONSOLE. IF IT IS THE ONLY MONITOR, IT WILL WAIT FOR THE OPERATOR TO ISSUE THAT BATCH SYSTEM -START COMMAND BEFORE IT STARTS PROCESSING, AFTER SENDING A REMINDER MESSAGE ("WAITING FOR BATCH SYSTEM -START").

AT THIS POINT, THE MONITOR SHOULD BE CHAPED TO THE APPROPRIATE VALUES (SEE PE-T-571).

THEN THE BATCH SYSTEM -START COMMAND SHOULD BE ISSUED. THIS WILL CAUSE THE MONITOR TO START BATCH PROCESSING.

THIS OPTION CAN ONLY BE USED BY A USER LOGGED IN UNDER SYSTEM.

-STOP -- STOP BATCH PROCESSING

COMMAND FORMAT:

-STOP

COMMAND ACTION:

THE -STOP OPTION CAUSES THE BATCH MONITOR TO LOG OUT IN A GRACEFUL MANNER. ANY CURRENTLY RUNNING BATCH JOBS WILL RUN TO COMPLETION.

THIS OPTION CAN ONLY BE USED BY A USER LOGGED IN UNDER SYSTEM.

-DISPLAY -- DISPLAY STATUS OF BATCH PROCESS

COMMAND FORMAT:

-DISPLAY

COMMAND ACTION:

THE STATUS OF THE BATCH SUBSYSTEM IS DISPLAYED.

STATUS ABOUT THE SUBSYSTEM INCLUDES THE NUMBER OF WAITING JOBS IN EACH QUEUE AND WHICH JOB(S) ARE BEING EXECUTED.

4.2 CONTROLLING BATCH JOBS -- THE JOB COMMAND

WHILE THE BATCH COMMAND CONTROLS THE BATCH MONITOR, THE JOB COMMAND IS USED TO CONTROL INDIVIDUAL JOBS. THE JOB COMMAND IS ALSO THE INTERFACE BETWEEN INTERACTIVE USERS AND THE BATCH SYSTEM; THE USER'S VIEW OF THE JOB COMMAND IS DESCRIBED IN THE NEXT SECTION. THERE IS SOME OVERLAP BETWEEN JOB OPTIONS AVAILABLE TO NORMAL USERS AND THOSE RESTRICTED TO THE OPERATOR. IN GENERAL, A USER CAN REFER ONLY TO HIS OWN JOBS -- JOBS SUBMITTED UNDER HIS LOGIN NAME -- WHILE AN OPERATOR CAN REFER TO ANY JOB IN THE BATCH SYSTEM.

4.2.1 COMMON OPERATOR/USER JOB OPTIONS

THE GENERAL FORM OF THE OPERATOR JOB COMMAND IS:

JOB <JOBNAME> [<OPTION> ...]

<JOBNAME> THE NAME OF THE JOB. JOB NAMES ARE DESCRIBED IN SECTION 2.1.2.

NOTE: THIS COMMAND CANNOT BE EXECUTED UNLESS THE SYSTEM DATE AND TIME HAVE BEEN SET VIA THE OPERATOR SETIME COMMAND.

SEVERAL OF THE OPTIONS AVAILABLE TO THE OPERATOR ARE IDENTICAL TO USER FUNCTIONS, AND THEIR DEFINITION IS LEFT TO SECTION 5. THESE OPTIONS ARE:

-RESTART -- RESTART A CURRENTLY EXECUTING JOB
-CANCEL -- CANCEL A JOB WAITING FOR INITIATION
-ABORT -- CANCEL A WAITING OR EXECUTING JOB
-STATUS -- DISPLAY THE STATUS OF A JOB
-DISPLAY -- DISPLAY ALL OF THE CHARACTERISTICS OF A JOB

WHILE THE OPERATOR CAN REFER TO ANY JOB IN THE BATCH SYSTEM, HE CANNOT USE THE -ABORT COMMAND (ON A RUNNING JOB) OR THE -RESTART COMMAND, UNLESS HE IS ENTERING THOSE COMMANDS FROM THE SUPERVISOR

CONSOLE.

HOWEVER, IF ONE OF THESE OPERATIONS IS ATTEMPTED, THE -ABORT WILL FAIL WITHOUT RE-WRITING ANY INFORMATION ON THE JOB, WHEREAS THE -RESTART WILL FAIL AFTER SIGNALLING THE RESTARTABILITY OF THE JOB, SO THAT WHEN THE JOB DOES FINISH OF ITS OWN ACCORD, IT WILL BE RESTARTED LATER.

IN OTHER WORDS, AN -ABORT OR -RESTART THAT FAILS WILL NOT ENDANGER THE BATCH DATABASE, THE BATCH MONITOR OR THE JOB UPON WHICH THE OPERATION WAS ATTEMPTED (EXCEPT THAT -RESTART IT MAY CAUSE IT TO -RESTART IF IT IS NOT -CANCELLED).

4.2.2 OPERATOR-RESTRICTED JOB OPTIONS

(IN THE FOLLOWING OPTION DESCRIPTIONS, 'JOB <JOBNAME>' IS ASSUMED TO START EACH COMMAND LINE, UNLESS OTHERWISE SPECIFIED.)

-DISPLAY -- DISPLAY CURRENT JOB(S)

FORMAT:

```
-DISPLAY [ ALL ]
         [TODAY]
```

ACTION:

THIS OPTION BEHAVES SIMILARLY TO THE USER'S JOB -DISPLAY OPTION, EXCEPT THAT IF NO <JOBNAME> IS SPECIFIED, IT WILL NOT BE LIMITED TO ONLY THOSE JOBS THAT BELONG TO USER "SYSTEM". IF <JOBNAME> IS SPECIFIED, JOBS NOT BELONGING TO "SYSTEM" THAT HAVE AN INTERNAL NAME OF <JOBNAME> ARE DISPLAYED (IF THEY ARE SELECTED BY THE SUFFIX "TODAY", "ALL" OR ABSENCE INDICATING ACTIVE JOBS ONLY).

THEREFORE, IF A USER LOGGED IN AS SYSTEM ENTERS THE COMMAND "JOB C_TEST -DISPLAY ALL", ONLY JOBS WITH EXTERNAL NAME OF C_TEST BELONGING TO USER SYSTEM WILL BE DISPLAYED. HOWEVER, "JOB #10004 -DISPLAY ALL" WOULD DISPLAY THE JOB WITH INTERNAL NAME #10004 IN ADDITION TO ANY JOBS WITH AN EXTERNAL NAME OF #10004.

PLEASE NOTE THAT "JOB #10004 -DISPLAY" WOULD ONLY DISPLAY THE JOB WITH INTERNAL NAME #10004 IF IT WAS STILL ACTIVE, I.E. WAITING, HELD OR EXECUTING. ALSO, "JOB #10004 -DISPLAY TODAY" WOULD ONLY DISPLAY IT IF IT WERE SUBMITTED, INITIATED, COMPLETED, ABORTED, OR CANCELLED TODAY.

-STATUS -- OUTPUT STATUS JOB(S)

FORMAT:

-STATUS

ACTION:

THIS OPTION IS SIMILAR TO THE -DISPLAY OPTION, EXCEPT THAT ONLY THE STATUS OF THE JOB (IN ADDITION TO THE JOB IDENTIFICATION INFORMATION) IS DISPLAYED.

-HOLD -- DELAY JOB INITIATION

FORMAT:

-HOLD

ACTION:

THE SPECIFIED JOB IS MARKED AS BEING HELD BY THE OPERATOR. IT WILL NOT BE INITIATED UNTIL RELEASED BY THE OPERATOR. IF THE JOB IS ALREADY IN EXECUTION, AN ERROR MESSAGE IS RETURNED.

-RELEASE -- RELEASE A HELD OR SUSPENDED JOB

FORMAT:

-RELEASE

ACTION:

A PREVIOUSLY HELD JOB IS RELEASED. IF THE JOB WAS NOT HELD, AN ERROR MESSAGE IS RETURNED.

5 USER FUNCTIONS

USE OF THE BATCH SYSTEM ENTAILS JOB CREATION AND JOB SUBMISSION. DURING JOB CREATION A USER MUST CREATE A COMMAND FILE SPECIFYING PRIMOS COMMANDS, WITH AN OPTIONAL BATCH COMMAND AS THE FIRST NON-COMMENT LINE IN THE FILE. HE MUST UNDERSTAND THE FUNCTIONS PERFORMED BY THE BATCH COMMAND AND ITS VARIOUS OPTIONS. ONCE A JOB IS CREATED, IT MUST BE SUBMITTED TO THE BATCH SYSTEM. THE JOB COMMAND ALLOWS JOB SUBMISSION, MODIFICATION, AND MONITOPING.

THIS SECTION DESCRIBES THESE MAJOR COMPONENTS OF THE BATCH SYSTEM -- THE BATCH CONTROL COMMAND AND THE JOB COMMAND.

5.1 BATCH CONTROL COMMAND

THE BATCH CONTROL COMMAND $$ JOB CONTAINS DIRECTIVES TO THE BATCH SYSTEM AND ACCOUNTING INFORMATION. THIS BATCH CONTROL COMMAND IS FULLY PROCESSED ONLY BY THE BATCH SYSTEM.

5.1.1 THE $$ JOB COMMAND

THE $$ JOB COMMAND MARKS THE START OF A JOB, ALTHOUGH ITS USE IS OPTIONAL. IF IT IS PRESENT, IT WILL NOT BE COPIED WITH THE REST OF THE COMMAND FILE. THE <LOGNAM> PARAMETER IS ALWAYS REQUIRED ON THE $$ JOB COMMAND AND IT MUST MATCH THE LOGNAM OF THE SUBMITTING USER. OTHER OPTIONAL PARAMETERS SPECIFY VARIOUS CHARACTERISTICS OF THE JOB -- TIME LIMITS, QUEUE PRIORITY, ETC. THE FORMAT OF THE $$ JOB COMMAND IS:

$$ JOB <LOGNAM> [<OPTION> ...]

<LOGNAM> THE LOGIN NAME OF THE SUBMITTING USER.

<OPTION>... ONE OR MORE OPTIONS FROM THE FOLLOWING LIST.

REQUIRED INFORMATION ON THE $$ JOB COMMAND IS <LOGNAM>. OTHER INFORMATION IS OPTIONAL. OPTIONAL PARAMETERS ALL START WITH A '-'. '/*' (COMMENT DELIMITER) HAS ITS STANDARD PRIMOS MEANING. THE FOLLOWING DESCRIBES THE CURRENTLY DEFINED $$ JOB COMMAND OPTIONS.

-ACCT -- PROVIDE ACCOUNTING INFORMATION

FORMAT:

-ACCT <INFO>

ACTION:

THE SPECIFIED ACCOUNTING INFORMATION <INFO> IS PASSED TO THE BATCH SYSTEM AS PART OF THE JOB'S INFORMATION. THE BATCH SYSTEM ITSELF MAKES NO INTERNAL USE OF <INFO>.

THIS INFORMATION MUST NOT BE AN EXPLICIT REGISTER SETTING OR HAVE AN UNQUOTED MINUS SIGN ('-') IN FRONT OF IT (I.E. OPTIONS ARE ILLEGAL). ALSO, ITS LENGTH IS LIMITED TO 80 CHARACTERS.

-HOME -- DEFINE HOME UFD

FORMAT:

-HOME <TREENAME>

ACTION:

THE HOME UFD IN WHICH THE JOB IS TO RUN IS SET TO <TREENAME>. (THIS OPTION HAS THE SAME EFFECT AS INCLUDING AN 'ATTACH <TREENAME>' AS THE FIRST PRIMOS COMMAND IN THE JOB.)

NOTE: IF THIS OPTION IS NOT SPECIFIED ON EITHER THE JOB COMMAND LINE OR THE $$ JOB COMMAND LINE, AN ATTEMPT WILL BE MADE TO OBTAIN THE HOME UFD ATTACH POINT OF THE SUBMITTING USER. THE RESULTING PATHNAME WILL BE DISPLAYED AS PART OF THE ACKNOWLEDGEMENT MESSAGE.

HOWEVER, THE PATHNAME WILL NOT HAVE ANY PASSWORDS IN IT. ALSO, IF THE PATHNAME CANNOT BE OBTAINED FOR SOME REASON (E.G., ITS LENGTH BEING GREATER THAN 80 CHARACTERS), THE ERROR MESSAGE 'HOME UFD REQUIRED' WILL BE OUTPUT.

IF THE -HOME OPTION IS SPECIFIED ON EITHER THE COMMAND LINE OR ON THE OPTIONAL $$ JOB LINE, THEN NO ATTEMPT WILL BE MADE TO OBTAIN THE HOME UFD ATTACH POINT AS DESCRIBED ABOVE.

THE LENGTH OF THIS PARAMETER IS LIMITED TO 80 CHARACTERS.

-CPTIME -- SET CPU TIME LIMIT

FORMAT:

-CPTIME <SECONDS>

ACTION:

THE MAXIMUM NUMBER OF CPU SECONDS OF EXECUTION IS SPECIFIED FOR THE JOB. IF THE JOB EXCEEDS THIS LIMIT IT WILL BE TERMINATED WITH A 'CPU TIME LIMIT EXCEEDED' MESSAGE. IF THE -CPTIME OPTION IS NOT SPECIFIED, THE EXECUTION TIME LIMIT DEFAULTS TO THAT OF THE QUEUE TO WHICH THE JOB IS SUBMITTED.

TO SPECIFY NO LIMIT, TYPE "NONE". THE RANGE OF THE NUMERIC ARGUMENT IS FROM 1 TO WHATEVER THE MAXIMUM CPU TIME LIMIT ON THE PARTICULAR QUEUE IS.

-ETIME -- SET ELAPSED TIME LIMIT

FORMAT:

-ETIME <MINUTES>

ACTION:

THE -ETIME OPTION IS COMPLETELY ANALOGOUS TO THE -CPTIME OPTION, THE UNITS BEING EXPRESSED IN MINUTES OF ELAPSED TIME.

-PRIORITY -- SET JOB PRIORITY

FORMAT:

-PRIORITY <VAL>

ACTION:

THE JOB QUEUE PRIORITY OF THE JOB IS SET TO <VAL>. THIS VALUE RANGES FROM 0 TO 9.

THE QUEUE PRIORITY OF A JOB AFFECTS THE AMOUNT OF TIME THAT IT WILL WAIT IN THE QUEUE BEFORE IT IS INITIATED. THE HIGHER THE QUEUE PRIORITY, THE SOONER IT WILL BE INITIATED.

THIS QUEUE PRIORITY IS A LEVEL OF PRIORITY WITHIN A QUEUE, HOWEVER: IN OTHER WORDS, WHEN A SEARCH IS MADE FOR AN ELIGIBLE JOB, IT SEARCHES QUEUES IN THE ORDER THAT THEY ARE DEFINED, AND WITHIN THOSE QUEUES, IT SEARCHES THE PRIORITY LEVELS IN ORDER FOM

9 TO 0.

IF -PRIORITY IS NOT SPECIFIED ON EITHER THE JOB COMMAND LINE OR ON THE (OPTIONAL) $$ JOB LINE, THE DEFAULT QUEUE PRIORITY FOR THAT QUEUE WILL BE USED.

THIS OPTION IS NOT A LEGAL -CHANGE OPTION (DESCRIBED IN SECTION 5.2.2).

-QUEUE -- SPECIFY BATCH QUEUE

FORMAT:

-QUEUE <QUEUE-NAME>

ACTION:

<QUEUE-NAME> IS THE NAME OF THE BATCH QUEUE TO WHICH THE JOB IS TO BE SUBMITTED. THE JOB WILL BE REJECTED IF ANY OTHER SPECIFIED OPTIONS (E.G., -CPTIME, -ETIME) CONFLICT WITH THE MAXIMUM VALUES ASSOCIATED WITH <QUEUE-NAME>, OR IF <QUEUE-NAME> DOES NOT EXIST OR IS BLOCKED.

IF THIS OPTION IS NOT SPECIFIED, A SEARCH IS MADE FOR THE FIRST QUEUE THAT THE JOB WILL "FIT" INTO. THIS SEARCH WILL IGNORE ANY BLOCKED QUEUES, OR QUEUES THAT ARE FLAGGED FOR DELETION. IT WILL ALSO BYPASS ANY QUEUE WHOSE CPU AND ELAPSED TIME LIMITS ARE LESS THAN THOSE OF THE JOB BEING SUBMITTED, USING THE DEFAULT VALUES FOR EACH QUEUE IF THEY ARE NOT SPECIFIED.

THIS OPTION IS NOT A LEGAL -CHANGE OPTION (DESCRIBED IN SECTION 5.2.2).

-FUNIT -- DEFINE COMMAND INPUT FILE UNIT

FORMAT:

-FUNIT <UNIT-NUMBER>

ACTION:

THE -FUNIT OPTION CAN BE USED TO OVERRIDE THE QUEUE-DEFINED DEFAULT FILE UNIT THAT IS USED FOR COMMAND INPUT. THE RANGE OF THIS NUMBER IS FROM 1 TO 126.

-RESTART -- SET RESTARTABILITY OF JOB

FORMAT:

-RESTART YES
NO

ACTION:

THIS OPTION CONTROLS THE RESTARTABILITY OF A JOB. IF THIS OPTION IS NOT SPECIFIED, THE DEFAULT IS -RESTART YES, WHICH INDICATES THAT THE JOB IS RESTARTABLE VIA AN EXPLICIT "JOB -RESTART" COMMAND, OR AFTER A SYSTEM COLD-START IF THE JOB HAD BEEN EXECUTING WHEN THE SYSTEM WAS BROUGHT DOWN. THE -RESTART NO LSAGE CAUSES A "JOB -RESTART" TO CAUSE AN ABORT OF THE JOB (SIMILAR TO "JOB -ABORT"), AND AT SYSTEM COLD-START, IF THE JOB HAD BEEN RUNNING, IT WOULD BE TREATED JUST AS THOUGH IT HAD ABORTED FOR SOME OTHER REASON (E.G. DISK FULL, MISSPELLING OF OPTION NAME, ETC.).

NOTE THAT THIS OPTION IS ALSO AVAILABLE FOR USE BY THE -CHANGE OPTION. THEREFORE, A JOB'S RESTARTABILITY MAY BE CHANGED WHILE THE JOB IS STILL WAITING, HELD, OR EXECUTING.

IF A JOB HAS BEEN SIGNALLED TO BE RESTARTED, BUT HAS NOT YET BEEN DISCOVERED AS BEING ABORTED BY THE BATCH MONITOR, AND A JOB -CHANGE -RESTART NO IS PERFORMED ON IT, THE PENDING RESTART WILL BE CLEARED, AND THE JOB WILL BECOME INACTIVE WHEN THE BATCH MONITOR DISCOVERS THAT IT HAS ABORTED OR COMPLETED. AT THIS POINT, A JOB -CHANGE -RESTART YES WILL NOT BRING BACK THE JOB, BUT IF IT IS STILL ACTIVE AND THE MONITOR HAS STILL NOT DETECTED ITS ABORTING, A RESTART OF THE JOB WILL SUCCEED AFTER THE CHANGE.

ALSO, THE "YES" OR "NO" SUFFIX MUST BE SPECIFIED. THIS IS REQUIRED SO THAT THE JOB COMMAND CAN DISCERN BETWEEN SUBMITTING A JOB WITH THE -RESTART OPTION, AND RESTARTING A JOB.

5.1.2 SAMPLE JOBS

SIMPLE FORTRAN COMPILATION AND LOAD

```
$$ JOB FRODO -QUEUE EXPRESS -HOME FRODO>PLOT -CPTIME 5
FTN FOO -64V
SEG
LOAD #FOO
LOAD B_FOO
LIBRARY VAPPLB
LIBRARY
MAP M_FOO
MAP 3
MAP 2
SAVE
QUIT
```

THE JOB IS SUBMITTED TO QUEUE A. THE CPU TIME LIMIT FOR THIS JOB IS 5 SECONDS. THE FILE NAMED FOO IN THE HOME DIRECTORY FRODO>PLOT IS COMPILED IN 64V-MODE, AND THE RESULTING BINARY IMAGE IS LOADED WITH VAPPLB AND THE V-MODE FORTRAN LIBRARY, AND SAVED AFTER SEVERAL MAPS ARE PRODUCED.

5.2 THE JOB COMMAND

THE JOB COMMAND PROVIDES THE INTERACTIVE USER WITH AN INTERFACE TO THE BATCH SYSTEM. USING THE JOB COMMAND, BATCH CONTROL COMMANDS CAN BE SPECIFIED AND JOBS SUBMITTED; THE CHARACTERISTICS OF EXISTING JOBS CAN BE MODIFIED; AND JOBS CAN BE ABORTED OR CANCELED.

5.2.1 JOB SUBMISSION USING THE JOB COMMAND

TO SUBMIT A JOB USING THE JOB COMMAND, A USER TYPES:

```
JOB <TREENAME> [<OPTION> ...]
```

<TREENAME> A COMMAND FILE THAT IS TO BE SUBMITTED TO THE BATCH SYSTEM.

<OPTION>... ONE OR MORE OPTIONS IDENTICAL TO THE $$ JOB OPTIONS DESCRIBED ABOVE.

THE COMMAND FILE MAY OR MAY NOT CONTAIN A $$ JOB COMMAND. IF A $$ JOB COMMAND IS THE FIRST (NON-COMMENT) COMMAND IN THE FILE SPECIFIED BY <TREENAME>, THE JOB COMMAND WILL READ THE OPTIONS SPECIFIED ON ITS COMMAND LINE, THEN THROW THE LINE ITSELF AWAY (I.E., IT WILL NOT BE COPIED ALONG WITH THE REST OF THE COMMAND FILE).

NOTE: THIS COMMAND CANNOT BE EXECUTED UNLESS THE SYSTEM DATE AND TIME HAVE BEEN SET VIA THE OPERATOR SETIME COMMAND.

IF <TREENAME> DOES START WITH A $$ JOB COMMAND, JOB WILL MERGE ANY SPECIFIED OPTIONS, USING THE JOB OPTIONS TO OVERRIDE CORRESPONDING $$ JOB OPTIONS.

5.2.2 MODIFICATION OF A JOB'S CHARACTERISTICS

ONCE A JOB HAS BEEN SUBMITTED AND BEFORE IT HAS BEEN INITIATED, ANY OF ITS CHARACTERISTICS (EXCEPT INTERNAL JOB NAME, EXTERNAL JOB NAME, QUEUE NAME AND QUEUE PRIORITY) CAN BE MODIFIED BY A USER WHOSE LOGIN NAME MATCHES THAT OF THE USER WHO SUBMITTED THE JOB. MODIFICATIONS TO A JOB CAN BE PERFORMED INTERACTIVELY USING THE JOB COMMAND AS FOLLOWS:

```
JOB <JOBNAME> -CHANGE [<OPTION> ...]
```

<JOBNAME> THE NAME OF THE JOB WHOSE CHARACTERISTICS ARE TO BE MODIFIED.

<OPTION>... ANY OF THE OPTIONS SPECIFIED FOR THE $$ JOB COMMAND, EXCEPT FOR -QUEUE AND -PRIORITY.

MODIFYING A JOB'S CHARACTERISTICS CAN BE THOUGHT OF AS LOGICALLY CANCELLING A JOB (SEE BELOW) AND RESUBMITTING IT.

NOTE: IF THE JOB IS EXECUTING, THE CHANGES WILL BE MADE, BUT WILL ONLY TAKE EFFECT IF THE JOB IS RESTARTED AFTER IT ABORTS OR COMPLETES. THE EXCEPTION IS THE -RESTART YES/NO OPTION, WHICH WILL IMMEDIATELY AFFECT ITS RESTARTABILITY, AND POSSIBLY EVEN ANNULL A PENDING RESTART.

5.2.3 CANCELLING OR ABORTING A JOB

A JOB IN THE BATCH SYSTEM MAY BE CANCELLED OR ABORTED BY A USER WHOSE LOGIN NAME IS THE SAME AS THAT OF THE USER WHO SUBMITTED THE JOB. THESE FORMS OF THE JOB COMMAND ARE:

```
JOB <JOBNAME> -CANCEL
              -ABORT
              -RESTART
```

<JOBNAME> THE INTERNAL NAME OF THE JOB TO BE CANCELLED, ABORTED, OR RESTARTED.

THE -CANCEL OPTION IS USED TO DELETE A JOB FROM A BATCH QUEUE'S JOB QUEUE. IF THE JOB HAS ALREADY BEEN INITIATED, IT CANNOT BE CANCELLED. IN THIS CASE, HOWEVER, THE -CANCEL WILL DISALLOW -RESTARTS TO OCCUR ON THAT JOB (EFFECTIVELY CANCELLING FURTHER

EXECUTIONS OF THAT PARTICULAR JOB).

THE -ABORT OPTION IS USED TO DROP A JOB REGARDLESS OF ITS STATUS. IF THE JOB IS WAITING OR HELD, -ABORT IS EQUIVALENT TO -CANCEL. IF THE JOB HAS BEEN INITIATED, IT WILL BE IMMEDIATELY TERMINATED.

-RESTART IS LOGICALLY EQUIVALENT TO AN -ABORT AND RESUBMISSION OF THE JOB. IF THE JOB IS NOT EXECUTING, AN ERROR MESSAGE WILL BE OUTPUT.

HOWEVER, IF A -RESTART IS DONE TO AN EXECUTING JOB AND A -CANCEL WAS PERFORMED ON THAT JOB (WHILE EXECUTING) OR THE JOB WAS SUBMITTED WITH THE -RESTART NO OPTION, THEN THE JOB WILL BE LOGGED OUT BUT IT WILL NOT BE RESTARTED. IF THE -CANCEL IS DONE AFTER THE -RESTART, THEN THE JOB WILL NO LONGER BE RESTARTABLE.

5.2.4 DISPLAY OF JOB STATUS AND CHARACTERISTICS

THE CURRENT STATUS AND CHARACTERISTICS OF A JOB CAN BE DETERMINED BY A USER WHOSE LOGIN NAME IS THE SAME AS THAT OF THE USER WHO SUBMITTED THE JOB. THESE FORMS OF THE JOB COMMAND ARE:

```
JOB [<JOBNAME>] -STATUS  [ ALL ]
                -DISPLAY [TODAY]
```

<JOBNAME> THE NAME OF THE JOB(S) WHOSE STATUS/CHARACTERISTICS ARE TO BE DISPLAYED. OMITTING <JOBNAME> WILL RESULT IN A DISPLAY FOR JOBS SUBMITTED BY THE USER ISSUING THE JOB COMMAND ("ALL" WILL DISPLAY ALL JOBS, "TODAY" WILL DISPLAY TODAY'S JOBS, AND OMITTING THE SUFFIX WILL DISPLAY ONLY WAITING, HELD OR EXECUTING JOBS).

THE -STATUS OPTION IS USED TO OBTAIN INFORMATION ABOUT THE CURRENT STATUS OF THE JOB. THE INFORMATION RETURNED BY -STATUS IS THE QUEUE TO WHICH THE JOB BELONGS, EXECUTION STATUS -- RUNNING, COMPLETED, HELD, WAITING, CANCELLED OR ABORTED, AND THE INTERNAL AND EXTERNAL NAMES OF THE JOB.

THE -DISPLAY OPTION RETURNS MORE DETAILED INFORMATION ON THE JOB. THIS INCLUDES THE STATUS OF THE JOB IN ADDITION TO THE VALUES FOR ALL JOB AND $$ JOB COMMAND OPTIONS -- THOSE SPECIFIED EXPLICITLY AND THOSE ASSUMED FROM QUEUE-DEFINED DEFAULTS.

5.2.5 SAMPLE JOB COMMANDS

```
JOB C_ALL -CPTIME 1000 -ETIME 30 -QUEUE FAST -ACCT 'BE SEEING YOU'
```

THE COMMAND FILE C_ALL IS SUBMITTED AS A JOB TO RUN IN PR4.64 WITH 1000 SECONDS OF CPU TIME AND 30 MINUTES OF ELAPSED TIME. THE JOB IS GIVEN THE (UNUSED) ACCOUNTING INFORMATION 'BE SEEING YOU'. IT WILL RUN IN QUEUE FAST. THE INTERNAL NAME OF THE JOB MIGHT BE #10034.

```
JOB C_ALL -STATUS
```

THE STATUS OF ALL ACTIVE JOBS NAMED C_ALL BELONGING TO THE USER MAKING THE REQUEST ARE DISPLAYED.

```
JOB C_ALL -ABORT
```

THE JOB C_ALL IS IMMEDIATELY ABORTED.

```
JOB C_ALL -DISPLAY
```

THE INFORMATION ON ACTIVE JOBS BELONGING TO THE USER MAKING THE REQUEST WITH EXTERNAL NAMES OF C_ALL ARE DISPLAYED:

```
JOB C_ALL(#10034), USER NUMBR6 HELD (QUEUE FAST).
SUBMITTED TODAY AT 1:58:49 P.M.
(THIS JOB HAS ALREADY EXECUTED 2 TIMES).
FUNIT=6. PRIORITY=5, CPU LIMIT=1000, ELAPSED LIMIT=30.
ACCOUNTING: BE SEEING YOU
```

THE LINE INDICATING THAT THE JOB HAS ALREADY EXECUTED 2 TIMES INDICATES THAT TWICE, WHILE THE JOB WAS EXECUTING, A JOB -RESTART WAS PERFORMED ON IT, OR THE SYSTEM WAS COLD-STARTED TWICE, OR A COMBINATION THEREOF.

APPENDIX A: SUMMARY OF BATCH SYSTEM COMMANDS AND OPTIONS

BATGEN

```
BATGEN <TREENAME>
```

OR:

```
BATGEN [<TREENAME>] <OPTION>
                      -STATUS
                      -DISPLAY
```

BATGEN COMMANDS

```
ADD <QUEUE-NAME>

BLOCK <QUEUE-NAME>
          ALL

DELETE <QUEUE-NAME>

DISPLAY [<QUEUE-NAME>]
        [    ALL     ]

FILE [<TREENAME>]

MODIFY <QUEUE-NAME>

QUIT

STATUS

UNBLOCK <QUEUE-NAME>
            ALL
```

BATGEN SUBCOMMANDS FOR ADD AND MODIFY

```
CPTIME <DEFAULT> [<MAXIMUM>]
           5           5

ETIME <DEFAULT> [<MAXIMUM>]
          20          20

FUNIT <UNIT-NUMBER>
            6
```

PRIORITY <VAL>
5

QUIT

RETURN

RLEVEL <DELTA-VAL>
0

TIMESLICE <VAL>
20

BATCH_COMMAND

BATCH SYSTEM [<OPTION>]

-START

-STOP

-DISPLAY

BATCH_CONTROL_COMMANDS

$$ JOB <LOGNAM> [<OPTION> ...]

-ACCT <INFO>

-CPTIME <SECONDS>

-ETIME <MINUTES>

-FUNIT <FUNIT>

-HOME <TREENAME>

-QUEUE <QUEUE-NAME>

-PRIORITY <VAL>

-RESTART YES
NO

JOB COMMAND

JOB <TREENAME> [-CHANGE] [<$$ JOB-OPTIONS> ...]

OR:

JOB <JOBNAME> [<OPTIONS>]

-ABORT

-CANCEL

-DISPLAY

-HOLD

-RELEASE

-RESTART

-STATUS

APPENDIX B: BATCH ERROR MESSAGES AND RESPONSES

THE BATCH SYSTEM HAS SEVERAL CATEGORIES OF MESSAGES THAT IT WILL OUTPUT: FATAL, WARNING, AND RESPONSES ARE THE MESSAGES THAT USERS WILL RECEIVE IN GENERAL, MESSAGES THAT ARE SENT TO THE SYSTEM CONSOLE ARE TO BE SEEN BY OPERATORS, AND SEVERE ERROR MESSAGES, WHILE THEY CAN BE SEEN BY USERS (IF THEY OCCUR) ARE TO BE REPORTED TO THE SYSTEM ADMINISTRATOR.

A QUERY IS A MESSAGE THAT IS OUTPUT TO THE USER, AND EXPECTS AN ANSWER OF "YES", "NO", OR "OK". ENTERING RETURN (I.E., A BLANK LINE) IS THE SAME AS ANSWERING "YES".

A FATAL MESSAGE IS ONE THAT ENDS IN "ER!" BEING TYPED OUT. IT USUALLY MEANS THAT THE ACTION REQUESTED BY THE USER FAILED.

A WARNING MESSAGE USUALLY INDICATES A FAILURE ON THE PART OF THE USER, BUT WILL NOT RAISE THE "ER!" CONDITION, I.E., COMMAND FILES WILL NOT ABORT AS A RESULT OF THIS ERROR. THE USER IS USUALLY RETURNED TO THE MODE OF OPERATION THAT SHE WAS IN WHEN THE ERROR OCCURRED, WHICH CAN EITHER BE PRIMOS COMMAND MODE, BATGEN COMMAND MODE, OR BATGEN SUBCOMMAND MODE.

A RESPONSE IS SIMPLY THAT: A RESPONSE TO A COMMAND. THE RESPONSES DOCUMENTED HERE ARE GENERALLY RESPONSES GIVEN IN UNUSUAL CIRCUMSTANCES, I.E. ONE WOULD NOT EXPECT TO SEE THEM EVERY TIME ONE ISSUED THE COMMAND INVOLVED. MESSAGES THAT GENERALLY FALL INTO THIS CATEGORY ARE SUCH THINGS AS INFORMING THE USER THAT A JOB THAT IS RUNNING HAS ALREADY BEEN RUN ONE OR MORE TIMES, OR THAT A JOB IS NOT RESTARTABLE. ETC.

NOTE THAT A RESPONSE MAY IN ITSELF BE A WARNING, MEANING THAT THE OPERATION WILL BE ATTEMPTED BUT THAT AN ERROR MAY OCCUR, OR THAT UNDESIREABLE CONDITIONS EXIST.

A MESSAGE IS TEXT THAT IS SENT TO THE SYSTEM CONSOLE BY THE BATCH MONITOR. IN GENERAL, THESE MESSAGES ARE SIMPLY INFORMATIVE, BUT SOMETIMES THEY COULD INDICATE SEVERITY OF A HIGH LEVEL (SUCH AS "DATABASE INVALID"). THEY ARE ALWAYS PRECEDED BY THE TEXT "*BATCH*".

A SEVERE ERROR IS ONE THAT RESULTS WHEN THE BATCH DATABASE IS COMPROMISED., BREAKAGES IN SECURITY (I.E. A USER BREAKING INTO THE DATABASE AND CHANGING IT) AND DISK ERRORS WILL CAUSE THESE PROBLEMS. USUALLY THE ONLY WAY TO RESOLVE THE ERROR IS FOR THE SYSTEM ADMINISTRATOR TO RE-INITIALIZE THE DATABASE, OR RUN *FIXBAT (SEE PF-T-622).

THE MESSAGES THAT FOLLOW ARE SORTED IN ALPHABETICAL ORDER (FOR EASY LOOK-UP), FOLLOWED BY THE SEVERITY LEVEL OF THE MESSAGE (RESPONSE, WARNING, ETC.), AND THE EXPLANATION OF THE MESSAGE.

(CHANGES MADE)

(RESPONSE) THE CHANGES SPECIFIED IN A JOB -CHANGE OPERATION HAVE BEEN MADE. IF THE JOB IS INITIATED AFTER THE CHANGES ARE MADE, THEN IT WILL EXECUTE WITH THE SPECIFIED CHANGES IN PLACE. THE JOB STATUS WILL BE DISPLAYED AFTER THE ABOVE MESSAGE IS TYPED OUT.

(JOB NO LONGER RESTARTABLE)

(RESPONSE) A JOB -CANCEL WAS PERFORMED ON AN EXECUTING JOB. WHEREAS THE JOB ITSELF IS NOT CANCELLED, IT HAS BEEN FLAGGED AS BEING UNRESTARTABLE (I.E. A -RESTART WILL ABORT THE JOB BUT NOT RESTART IT), WHETHER OR NOT THE -RESTART IS DONE BEFORE THE -CANCEL OR AFTER.

(JOB NOT RESTARTABLE)

(WARNING) A JOB -RESTART WAS PERFORMED ON A JOB THAT HAD ALREADY BEEN FLAGGED AS BEING UNRESTARTABLE, AS A RESULT OF A -CANCEL BEING DONE ON THE JOB WHILE IT WAS EXECUTING, OR BEING SUBMITTED WITH THE -RESTART NO OPTION. AN ATTEMPT WILL BE MADE TO ABORT THE JOB.

(JOB RESTARTED)

(RESPONSE) A JOB -RESTART WAS PERFORMED ON A JOB, AND THE JOB HAS BEEN FLAGGED AS BEING RESTARTABLE. ALTHOUGH AN ERROR MESSAGE MAY APPEAR AFTER THIS MESSAGE, THE JOB WILL GENERALLY BE RESTARTED UNLESS A JOB -CANCEL OR JOB -CHANGE -RESTART NO IS DONE ON IT. POSSIBLE ERROPS AFTER THIS MESSAGE INCLUDE "INSUFFICIENT ACCESS RIGHTS" IF THE USER IS LOGGED IN AS SYSTEM, AND RESTARTED ANOTHER USER'S JOB FROM A USER TERMINAL (NOT THE SYSTEM CONSOLE), OR IF THE PROCESS RECENTLY LOGGED OUT. "NOT FOUND" MAY ALSO BE RETURNED IN THIS CASE.

(THIS JOB HAS ALREADY EXECUTED NN TIME(S)).

(RESPONSE) OUTPUT BY A JOB -DISPLAY COMMAND IF THE JOB BEING DISPLAYED IS EXECUTING AND HAS ALREADY BEEN EXECUTED. THIS IS THE RESULT OF A JOB -RESTART BEING DONE ON THAT JOB, OR A SYSTEM COLD-START AFTER BEING BROUGHT DOWN WHILE THE JOB WAS EXECUTING.

*** INVALID BATCH DATABASE, PLEASE CONTACT YOUR SYSTEM ADMINISTRATOR.

(SEVERE) THIS MESSAGE MEANS THAT THE RUNNING JOB DETECTED AN ERROR (SUCH AS DISK FAILURE, POINTER MISMATCH, OR MISPROTECTED FILE) IN THE BATCH SYSTEM DATABASE. IT WILL FLAG THE DATABASE AS INVALID. THE SYSTEM ADMINISTRATOR SHOULD BE NOTIFIED, AS SHE HAS THE

RESPONSIBILITY FOR RE-INITIALIZING THE DATABASE (OR RUNNING *FIXBAT OR FIXRAT AS THE CASE MAY BE). THE BATCH AND JOB COMMANDS WILL BE INOPERATIVE UNTIL THE SITUATION IS RESOLVED.

*** JOBS ARE NOT BEING PROCESSED AT THIS TIME.

(SEVERE) IF FOLLOWED BY "*** PLEASE CONTACT YOUR SYSTEM ADMINISTRATOR IMMEDIATELY", IT INDICATES THAT THE BATCH DATABASE HAS NOT BEEN INITIALIZED, OR THAT SOMETHING HAS HAPPENED TO IT (LIKE A DISK HEAD CRASH). IF FOLLOWED BY "*** PLEASE TRY AGAIN LATER", IT INDICATES THAT WHILE THE DATABASE IS STILL VALID, THE BATCH MONITOR WAS LOGGED OUT USING A METHOD OTHER THAN "BATCH SYSTEM -STOP", AND WILL VERIFY THE VALIDITY OF THE DATABASE WHEN IT IS STARTED UP. EITHER WAY, THE USER WILL BE IMMEDIATELY RETURNED TO COMMAND MODE, I.E., THE OPERATION THE USER ATTEMPTED WILL NOT BE PERFORMED. THIS CAN BE TYPED OUT BY THE BATCH OR THE JOB COMMANDS WHEN THEY START RUNNING.

<NN> IS OUT OF RANGE. <OPTION>

(FATAL) THE NUMBERS SUPPLIED AS PARAMETERS TO THE -FUNIT OR -PRIORITY OPTIONS DURING JOB SUBMISSION/CHANGING WERE OUT OF RANGE. THE RANGE FOR -FUNIT IS FROM 1 TO 126, AND -PRIORITY IS FROM 0 (ZERO) TO 9. THE JOB SHOULD BE RESUBMITTED OR CHANGED WITH LEGAL -FUNIT AND -PRIORITY VALUES. NOTE THAT THE SYSTEM MAY BE CONFIGURED TO HAVE FEWER THAN 126 UNITS PER USER AT COLD-START, AND THE -FUNIT ARGUMENT WILL BE LIMITED TO THE MAXIMUM CONFIGURED UNIT NUMBER.

<TEXT> SEEN WHEN END-OF-LINE EXPECTED.

(WARNING) THIS ERROR IS A WARNING, BUT IT MAY HAVE DIFFERING DEGREES OF FATALITY DEPENDING ON THE PROGRAM BEING RUN. IN GENERAL, IT CAUSES THE COMMAND LINE THAT WAS READ TO BE TOSSED OUT; IN BATGEN COMMAND/SUBCOMMAND MODE, THE USER WILL BE LEFT IN COMMAND/SUBCOMMAND MODE. IN THE CASE OF JOB, BATCH, AND ENTERING THE BATGEN COMMAND, THIS ERROR RESULTS IN THE USER BEING RETURNED TO PRIMOS, ALTHOUGH THE "ER!" CONDITION IS NOT RAISED. THIS MESSAGE MEANS THAT <TEXT> WAS SEEN WHEN THERE SHOULD HAVE BEEN NO MORE TEXT (END OF LINE).

?JOB <EXTNAM>(<INTNAM>) <STATUS>.

(WARNING) AN ATTEMPT WAS MADE TO PERFORM AN OPERATION ON A JOB USING THE JOB COMMAND THAT COULD NOT BE PERFORMED BECAUSE OF ITS STATUS. EXAMPLES ARE TRYING TO RESTART A COMPLETED JOB, OR RELEASE A JOB THAT IS NOT HELD.

BAD $$ COMMAND.

(FATAL) A COMMAND FILE WAS SUBMITTED USING THE JOB COMMAND THAT HAD A $$ LINE AS THE FIRST NON-COMMENT LINE, BUT THE $$ COMMAND WAS NOT A $$ JOB COMMAND. THE COMMAND FILE SHOULD BE CHANGED SO THAT THE "$$" LINE IS LEGAL (SEE SECTION 5.1.1). THE USE OF $$ IS RESERVED FOR FUTURE EXPANSION BY BATCH.

BAD QUEUE CONTROL FILE.

(SEVERE) ONE OF THE BATCH SUBSYSTEM DATABASE FILES IS INACCESSIBLE OR HAS A BAD FORMAT. THE BATCH SUBSYSTEM IS THEREFORE INOPERATIVE UNTIL IT IS FIXED.

BAD QUEUE DEFINITION FILE.

(FATAL) A FILE REFERENCED BY BATGEN DID NOT COMPLY TO FORMAT REQUIREMENTS. I.E. WAS NOT A LEGAL QUEUE DEFINITION FILE. IF THIS ERROR OCCURS IN OTHER THAN THE BATGEN PROGRAM, THEN THE SYSTEM BATCH DEFINITION FILE HAS BEEN OVERWRITTEN WITH ILLEGAL DATA, AND THE BATCH SUBSYSTEM IS INOPERATIVE.

BATDEF FILE IS MISSING.

(MESSAGE) THE QUEUE DEFINITION FILE, WHICH IS THE CRUX OF THE DATABASE, IS NOT PRESENT. THE MONITOR WILL LOG ITSELF OUT AFTER SENDING THIS MESSAGE. THE SYSTEM ADMINISTRATOR SHOULD USE BATGEN TO GENERATE A NEW BATDEF FILE, OR COPY A NEW BATDEF FROM A BACKUP COPY USING FUTIL.

CAN'T START BATCH JOB!

(MESSAGE) THE BATCH MONITOR WAS NOT SPAWNED FROM THE SYSTEM CONSOLE, AND THEREFORE CANNOT LOG IN PROCESSES UNDER DIFFERENT LOGIN NAMES OR LOG OUT OTHER PROCESSES. THE MONITOR WILL LOG ITSELF OUT GRACEFULLY AFTER SENDING THIS MESSAGE. SIMPLY RESPAWN THE BATCH MONITOR FROM THE SYSTEM CONSOLE IF THIS HAPPENS.

CANNOT RUN UNDER DOS.

(FATAL) NO PART OF THE BATCH SYSTEM CAN RUN UNDER PRIMOS II. IF IT IS NECESSARY TO CHANGE THE BATDEF FILE WHILE UNDER PRIMOS II, THE BEST SOLUTION IS TO DELETE IT. THE BATCH MONITOR WILL NOT BE RUNNABLE UNTIL IT IS RE-CREATED, BUT IT WILL NOT INVALIDATE THE DATABASE.

COMMAND FILE REQUIRED AS FIRST ARGUMENT ON SUBMISSION.

(FATAL) THE JOB COMMAND WAS GIVEN WITH JOB OPTIONS (SUCH AS -HOME, -PRIORITY, -CPTIME, ETC.) BUT NO COMMAND FILE WAS SEEN BEFORE THOSE OPTIONS. THE SYNTAX IS "JOB <IREENAME> [-OPTIONS]".

CPU LIMIT MUST BE SPECIFIED.

(FATAL) THE QUEUE REFERRED TO BY A -QUEUE OPTION DURING JOB SUBMISSION IS DEFINED SUCH THAT THE -CPTIME OPTION IS A REQUIRED PARAMETER (I.E. DEFAULT CPU LIMIT FOR THAT QUEUE IS GREATER THAN THE MAXIMUM CPU LIMIT FOR THAT QUEUE). THE JOB SHOULD BE RESUBMITTED WITH THE -CPTIME OPTION SPECIFIED. TO DETERMINE THE MAXIMUM LIMITS FOR QUEUES, USE BATGEN -DISPLAY.

DATABASE INVALID.

(MESSAGE) THIS IS A SEVERE ERROR. THE MONITOR WILL LOG ITSELF OUT AFTER SENDING THIS MESSAGE, AND THE BATCH SYSTEM WILL BE LEFT INOPERATIVE (USERS WILL RECEIVE ERROR MESSAGES IF THEY TRY TO INVOKE JOB OR BATCH). THE SYSTEM ADMINISTRATOR SHOULD DETERMINE WHAT THE ERROR WAS AND FIX IT IF POSSIBLE. IF THE BATCH MONITOR RUNS A COMOUTPUT FILE, THEN THAT SHOULD REVEAL THE SOURCE OF THE ERROR. IN GENERAL, IF THE EXACT CAUSE OF THE PROBLEM IS NOT KNOWN (SUCH AS A POINTER MISMATCH ERROR IN THE DATABASE, OR A DISK WRITE-PROTECTED ERROR), EITHER *FIXBAT SHOULD BE RUN (SEE PE-T-622), OR, IF THAT FAILS, THE COMMAND FILE C_BDIF SHOULD BE INVOKED IN BATCHQ TO REINITIALIZE THE ENTIRE DATABASE. IF IT DOESN'T WORK, THERE ARE PROBABLY DISK ERRORS. IF IT DOES, REDEFINE THE BATCH QUEUES USING BATGEN AND START THE BATCH MONITOR UP AGAIN. ALL JOB DATA WILL HAVE BEEN LOST.

ELAPSED TIME LIMIT MUST BE SPECIFIED.

(FATAL) SEE THE EXPLANATION FOR THE "CPU LIMIT MUST BE SPECIFIED" MESSAGE. REFERENCES TO CPU LIMITS AND THE -CPTIME OPTION SHOULD BE READ AS REFERENCES TO ELAPSED TIME LIMITS AND THE -ETIME OPTION.

END OF LINE.

(FATAL) ONE OF THE BATCH PROGRAMS WAS EXPECTING TO FIND MORE INFORMATION ON THE COMMAND LINE, BUT END-OF-LINE WAS FOUND INSTEAD. THE MESSAGE WILL GENERALLY CONTAIN MORE INFORMATION ON WHAT WAS EXPECTED. RE-ENTER THE COMMAND WITH THE ADDITIONAL REQUESTED INFORMATION.

END OF LINE. ILLEGAL <OPTION> ARGUMENT

(FATAL) ONE OF THE JOB PARAMETER OPTIONS WAS SPECIFIED ON THE JOB COMMAND LINE, BUT HAD NO ARGUMENT (END OF LINE). <OPTION> WILL BE REPLACED BY THE NAME OF THE OPTION. THE INFORMATION REQUIRED BY THAT OPTION SHOULD BE SUPPLIED WHEN THE COMMAND IS RE-ENTERED.

END OF LINE. QUEUE NAME REQUIRED

(WARNING) A COMMAND ENTERED WHILE IN BATGEN COMMAND MODE REQUIRED A QUEUE NAME (ADD, MODIFY, BLOCK, UNBLOCK, AND DELETE ALL REQUIRE QUEUE NAMES). RE-ENTER THE COMMAND WITH THE QUEUE NAME DESIRED.

END OF LINE. VALUE REQUIRED

(WARNING) WHILE IN BATGEN SUBCOMMAND MODE, A SUBCOMMAND WAS GIVEN WHICH REQUIRED AT LEAST ONE NUMERIC PARAMETER, BUT THERE WAS NONE. SUBCOMMANDS REQUIRING AT LEAST ONE NUMERIC PARAMETER ARE CPTIME, ETIME, FUNIT, PRIORITY, TIMESLICE AND RLEVEL. NOTE THAT THE CPTIME AND ETIME SUBCOMMANDS ACCEPT TWO PARAMETERS, BOTH OF WHICH MAY BE THE CHARACTER STRING "NONE" INDICATING NO LIMITS. RE-ENTER THE SUBCOMMAND WITH THE VALUE DESIRED (EXAMPLE: "FUNIT 13").

ENTER QUEUE CHARACTERISTICS:

(RESPONSE) THE ADD OR MODIFY COMMAND GIVEN WHILE IN BATGEN COMMAND MODE SUCCEEDED. THE USER IS NOW IN BATGEN SUBCOMMAND MODE, IDENTIFIED BY THE TYPING OF THE '$' PROMPT INSTEAD OF THE '>' PROMPT GIVEN WHEN IN BATGEN COMMAND MODE. TO RE-ENTER COMMAND MODE FROM SUBCOMMAND MODE, USE QUIT OR RETURN. RETURN SAVES THE INFORMATION CHANGED WHILE IN SUBCOMMAND MODE, QUIT DISCARDS IT (ASKING FOR VERIFICATION IF ANY OF IT WAS CHANGED).

ENVIRONMENT MODIFIED, OK TO QUIT?

(QUERY) A QUIT COMMAND WAS ISSUED WHILE IN BATGEN COMMAND MODE, AND THE ENVIRONMENT HAD BEEN MODIFIED. LEGAL ANSWERS TO THIS QUESTION ARE "YES", "NO", AND "OK". IF "YES" OR "OK" IS THE RESPONSE, A SUBSEQUENT START COMMAND WILL RE-ENTER BATGEN COMMAND MODE WITH NO LOSS OF INFORMATION ABOUT THE ENVIRONMENT.

HOME UFD REQUIRED.

(FATAL) THE -HOME OPTION WAS NOT PRESENT ON THE JOB OR THE (OPTIONAL) $$ JOB LINE DURING SUBMISSION, AND THE PROGRAM WAS UNABLE TO DETERMINE THE HOME ATTACH POINT OF THE SUBMITTING JOB. RESUBMIT THE JOB, AND INCLUDE THE -HOME OPTION FOLLOWED BY THE ABSOLUTE PATHNAME OF WHERE THE JOB IS TO EXECUTE. IF THE PATHNAME

CANNOT FIT, USE A SHORTER DESCRIPTION OF IT WHEN YOU RESUBMIT THE COMMAND FILE, AFTER EDITING THE FILE TO INCLUDE AN "ATTACH" COMMAND WITH A RELATIVE PATHNAME TO DESCEND THE REMAINING SUB-UFDS TO REACH THE DESTINATION.

HOME=<PATHNAME>

(RESPONSE) DURING JOB SUBMISSION, THE -HOME OPTION WAS NOT SPECIFIED ON THE COMMAND LINE OR IN THE COMMAND FILE ($$ JOB), SO THE JOB COMMAND DETERMINED THE HOME ATTACH POINT OF THE SUBMITTING JOB. THIS MESSAGE IS TYPED OUT (WHERE <PATHNAME> BECOMES THE HOME UFD FOR THE SUBMITTED JOB) TO REMIND THE USER THAT THE -HOME OPTION WAS NOT SPECIFIED. THE JOB DID SUCCESSFULLY SUBMIT, HOWEVER.

ILLEGAL -CHANGE OPTION.

(FATAL) THE OPTIONS -QUEUE AND -PRIORITY ARE ILLEGAL DURING A -CHANGE OPERATION USING THE JOB COMMAND, AS QUEUE AND QUEUE PRIORITY OF A JOB CANNOT BE CHANGED. CANCEL OR ABORT THE JOB AND RE-SUBMIT IT INTO THE APPROPRIATE QUEUE WITH THE DESIRED QUEUE PRIORITY.

ILLEGAL ANSWER.

(WARNING) OUTPUT WHEN THE ANSWER TO A QUESTION IS NOT "YES", "NO", OR "OK". WILL ASK THE QUESTION AGAIN. THESE QUESTIONS ARE ASKED WHEN A USER TRIES TO QUIT OUT OF BATGEN COMMAND OR SUBCOMMAND MODE AFTER MODIFYING THE ENVIRONMENT.

ILLEGAL COMBINATION. <OPTION>

(FATAL) A JOB PARAMETER (SUCH AS -ACCT, -HOME OR -QUEUE, ETC.) WAS SPECIFIED ON THE SAME JOB COMMAND LINE AS A OPTION TO PERFORM A CERTAIN ACTION (SUCH AS -CANCEL, -DISPLAY, -ABORT, ETC.). USE SEPARATE JOB COMMANDS TO PERFORM SEPARATE FUNCTIONS.

ILLEGAL LIMIT.

(FATAL) THE PARAMETERS SUPPLIED TO THE -CPTIME OR -ETIME OPTIONS DURING JOB SUBMISSION/CHANGING WERE NOT LEGAL LIMITS, I.E. THEY WERE LESS THAN OR EQUAL TO 0 (ZERO), OR WERE NOT LEGAL DECIMAL NUMBERS AND NOT THE STRING "NONE". RE-ENTER THE COMMAND WITH LEGAL LIMITS.

ILLEGAL NAME.

(FATAL) ONE OF THE BATCH PROGRAMS WAS EXPECTING A NAME OR COMMAND, BUT IT READ AN UNQUOTED TOKEN BEGINNING WITH A DASH ('-'), INDICATING THAT AN OPTION WAS PRESENT.

ILLEGAL NUMBER. <TEXT> (BATGEN)

(WARNING) WHILE IN BATGEN SUBCOMMAND MODE, A SUBCOMMAND WAS GIVEN WHICH EXPECTED A NUMERIC PARAMETER, BUT THE PARAMETER FOLLOWING THE SUBCOMMAND DID NOT COMPLY WITH RULES FOR NUMBERS. RE-ENTER THE LINE WITH A LEGAL DECIMAL NUMBER. ALL NUMBERS INPUT BY BATGEN (AND ALL OTHER BATCH SOFTWARE) ARE DECIMAL. SUBCOMMANDS THAT MAY RETURN THIS ERROR ARE CPTIME, ETIME, FUNIT, PRIORITY, TIMESLICE, AND RLEVEL. NOTE THAT THE CPTIME AND ETIME SUBCOMMANDS WILL ACCEPT THE CHARACTER STRING "NONE" INDICATING NO LIMITS, BUT WILL FLAG THE NUMBER 0 (ZERO) AS AN "ILLEGAL NUMBER". ALSO, THESE TWO SUBCOMMANDS INTERPRET THE NUMBERS AS FORTRAN INTEGER*4 NUMBERS, WHEREAS THE OTHER SUBCOMMANDS USE INTEGER*2.

ILLEGAL NUMBER. <TEXT> (JOB)

(FATAL) THE ARGUMENT FOR THE -FUNIT OR -PRIORITY OPTION DURING JOB SUBMISSION USING THE JOB COMMAND WAS NOT A LEGAL DECIMAL NUMBER. RE-ENTER THE COMMAND LINE WITH LEGAL NUMERIC PARAMETERS.

ILLEGAL OPTION.

(FATAL) ONE OF THE BATCH PROGRAMS WAS EXPECTING AN OPTION, I.E., A UNQUOTED TOKEN BEGINNING WITH A DASH ('-'). PERHAPS IT WAS ACCIDENTALLY LEFT OFF; IN ANY CASE, RE-ENTERING THE COMMAND LINE WITH A LEGAL FORMAT SHOULD PRODUCE MORE DESIREABLE RESULTS.

ILLEGAL QUEUE NAME. <TEXT> (BATGEN)

(WARNING) AN ATTEMPT WAS MADE TO ADD A QUEUE THAT HAD A NAME WHICH DID NOT COMPLY WITH FILENAME RULES (FIRST CHARACTER MUST NOT BE A DIGIT, CHARACTER SET LIMITED TO ALPHABETICS, DIGITS, AND SELECTED SPECIAL CHARACTERS). RE-ENTER THE COMMAND WITH A LEGAL QUEUE NAME. NOTE THAT A QUEUE NAME OF "ALL" IS ILLEGAL, SO THAT "DELETE ALL" WOULD NOT PRODUCE UNDESIREABLE RESULTS.

ILLEGAL QUEUE NAME. <TEXT> (JOB)

(FATAL) THE QUEUE NAME SPECIFIED AFTER A -QUEUE OPTION WHILE SUBMITTING OR CHANGING A JOB DID NOT COMPLY WITH QUEUE NAME FORMAT RULES. USE BATGEN -STATUS OR -DISPLAY TO DETERMINE THE NAMES OF LEGAL QUEUES.

IN.USE NOT OPEN.

(MESSAGE) THE FILE WHICH THE MONITOR KEEPS OPEN FOR WRITING WHILE IT IS RUNNING HAS BEEN MYSTERIOUSLY CLOSED. THE MONITOR WILL LOG ITSELF OUT AFTER SENDING THIS MESSAGE. THIS IS SOMETIMES THE RESULT OF AN ACCIDENTAL SHUTDOWN OF THE DISK THAT THE MONITOR USES (WHERE BATCHQ RESIDES) OR THE CLOSE BATCHQ>IN.USE COMMAND BEING GIVEN FROM THE SYSTEM CONSOLE. AFTER DETERMINING THAT THE BATCHQ UFD EXISTS, RE-SPAWN THE BATCH MONITOR.

INCORRECT USER-NAME.

(FATAL) A COMMAND FILE WAS SUBMITTED USING THE JOB COMMAND THAT HAD A $$ JOB LINE AS THE FIRST NON-COMMENT LINE, BUT THE USER-NAME SPECIFIED AFTER THE "JOB" SPECIFIER DID NOT MATCH THE USER-NAME OF THE SUBMITTING USER. EDIT THE COMMAND FILE AND CHANGE THE USERNAME IN THE $$ JOB LINE TO THE USERNAME OF THE SUBMITTER.

JOB <EXTNAM> FOR <USERNAME>(<INTNAM>) <STATUS>.

(MESSAGE) WILL BE OUTPUT BY THE BATCH MONITOR ANY TIME IT CHANGES THE STATUS OF A JOB (EXCEPT WHEN IT CHANGES A RESTARTED JOB BACK TO "WAITING"). <EXTNAM> IS THE EXTERNAL NAME OF THE JOB, <USERNAME> IS THE SUBMITTING USER. <INTNAM> IS THE INTERNAL NAME, AND <STATUS> IS EITHER "ABORTED" OR "COMPLETED".

JOB NAME REQUIRED.

(FATAL) THE OPTIONS -CHANGE, -CANCEL, -ABORT, -RESTART, -HOLD AND -RELEASE ALL REQUIRE A JOB IDENTIFIER (INTERNAL OR EXTERNAL NAME). RE-ENTER THE COMMAND WITH THE JOB ID (EXAMPLES: "JOB C.TOP -HOLD", "JOB #10032 -ABORT").

JOB NOT FOUND.

(FATAL) THE JOB REFERRED TO IN A JOB COMMAND SUCH AS -CHANGE, -CANCEL, -ABORT, -RESTART, -HOLD OR -RELEASE, COULD NOT BE FOUND BY SEARCHING THE ACTIVE JOBS LIST. THIS COULD MEAN ONE OF THREE THINGS: THAT NO JOB EXISTS WITH THAT NAME, THAT ALL JOBS THAT HAVE THAT NAME ARE NOT ACTIVE JOBS (I.E. HAVE COMPLETED, ABORTED OR BEEN CANCELLED), OR THAT A JOB EXISTS WITH THAT EXTERNAL NAME BUT THE USER MAKING THE REQUEST IS NOT THE SAME USER THAT ORIGINALLY SUBMITTED THE JOB.

JOB QUEUES INITIALIZED.

(RESPONSE) THIS IS OUTPUT BY THE *INIT PROGRAM WHEN IT IS RUN BY THE C_BDIF OR C_RSET FILES IN THE BATCHQ UFD. IT MEANS THAT THE QUEUE AND EXECUTION DATA HAS BEEN ZEROED.

JOB WILL BE RESTARTED.

(MESSAGE) THIS IS SENT TO THE SYSTEM CONSOLE AFTER A "JOB <EXTNAM> FOR <USERNAME>(<INTNAM>) ABORTED/COMPLETED" MESSAGE IS SENT, ONLY WHEN THE BATCH MONITOR IS FIRST STARTED UP. IT MEANS THAT THE JOB IS ELIGIBLE FOR RESTARTING, AND THAT IT IS THEREFORE BEING RESET TO THE WAITING STATE. IT GENERALLY INDICATES THAT THE JOB WILL BE RECOVERABLE FROM A SYSTEM SHUTDOWN.

MULTIPLE JOBS WITH THIS NAME (USE INTERNAL NAME).

(FATAL) A REFERENCE WAS MADE TO A JOB USING AN EXTERNAL NAME IN THE JOB COMMAND, AND THERE WERE AT LEAST 2 SUCH JOBS BELONGING TO THE USER MAKING THE REFERENCE THAT WERE ACTIVE. THE INTERNAL NAME MUST BE USED IN THIS CASE. USE JOB -STATUS ALL TO DETERMINE THE INTERNAL AND EXTERNAL NAMES OF ALL JOBS BELONGING TO THE USER ISSUING THE COMMAND IN THE DATABASE.

MULTIPLE OCCURANCE.

(FATAL) AN OPTION WAS SPECIFIED TWICE DURING JOB SUBMISSION OR JOB CHANGING (EXAMPLE: JOB C_TEST -HOME HERE -HOME THERE) ON EITHER THE JOB OR $$ JOB LINE, BUT NOT BOTH (I.E. IF IT IS SPECIFIED ONCE ON THE JOB LINE AND ONCE ON THE $$ JOB LINE, NO ERROR WILL RESULT AND THE PARAMETER ON THE JOB LINE WILL TAKE PRECEDENCE). RE-ENTER THE COMMAND, BUT SPECIFY EACH OPTION ONLY ONCE.

MULTIPLE PROCESSES ILLEGAL.

(MESSAGE) AN ATTEMPT WAS MADE TO START UP A SECOND BATCH MONITOR. THE MONITOR THAT SENT THIS MESSAGE WILL LOG OUT.

MUST BE FIRST OPTION.

(FATAL) THE OPTIONS -CHANGE, -CANCEL, -ABORT, -RESTART, -STATUS, -DISPLAY, -HOLD AND -RELEASE MUST BE THE FIRST OPTION ON THE JOB COMMAND LINE (AFTER A SOMETIMES OPTIONAL JOB IDENTIFIER). USE THE JOB COMMAND SEVERAL TIMES TO PERFORM SEVERAL OPERATIONS.

NO ACTIVE JOBS [NAMED "<JOBNAME>"]

(RESPONSE) WILL HAVE EITHER "FOR USER <USERNAME>" OR "IN SYSTEM" APPENDED TO IT, DEPENDING ON WHETHER OR NOT THE USER IS LOGGED IN AS SYSTEM. THIS MESSAGE IS TYPED OUT BY A JOB -DISPLAY OR -STATUS COMMAND, AND INDICATES THAT THERE ARE NO JOBS BELONGING TO THAT USER THAT ARE WAITING, HELD, OR EXECUTING. IF USER IS SYSTEM, THEN THERE ARE NO JOBS THAT ARE WAITING, HELD OR EXECUTING IN THE ENTIRE SYSTEM.

THE TEXT IN BRACKETS ('NAME "<JOBNAME>"') IS OUTPUT IF A JOBNAME WAS SPECIFIED FOR THE -DISPLAY OR -STATUS COMMAND, OTHERWISE IT IS OMITTED.

NO CONFIGURED QUEUES.

(RESPONSE) A BATGEN INVOCATION USING STATUS OR DISPLAY AS EITHER COMMANDS OR OPTIONS ON THE BATGEN COMMAND LINE FOUND THAT THERE WERE NO DEFINED QUEUES.

NO JOB CHANGES SPECIFIED.

(FATAL) THE -CHANGE OPTION WAS GIVEN TO THE JOB COMMAND, BUT NO ACTUAL CHANGES WERE SPECIFIED ON THE COMMAND LINE. SPECIFY CHANGES TO BE MADE AFTER THE -CHANGE OPTION.

NO JOBS [NAMED "<JOBNAME>"]

(RESPONSE) WILL HAVE EITHER "FOR USER <USERNAME>" OR "IN SYSTEM" APPENDED TO IT, DEPENDING ON WHETHER OR NOT THE USER IS LOGGED IN AS SYSTEM. THIS MESSAGE IS TYPED OUT BY A JOB -DISPLAY ALL OR -STATUS ALL COMMAND, AND INDICATES THAT THERE ARE NO JOBS BELONGING TO THAT USER (OR IN THE ENTIRE BATCH SYSTEM IF THE USER IS SYSTEM).

THE TEXT IN BRACKETS ('NAME "<JOBNAME>"') IS OUTPUT IF A JOBNAME WAS SPECIFIED FOR THE -DISPLAY OR -STATUS COMMAND, OTHERWISE IT IS OMITTED.

NO LONGER EXECUTING.

(FATAL) A JOB -ABORT OR JOB -RESTART WAS PERFORMED ON A JOB THAT HAD EXECUTION STATUS, BUT BY THE TIME THE EXECUTION FILE WAS READ IN TO DETERMINE THE USER NUMBER OF THE PROCESS, IT HAD DISAPPEARED. IF THE MESSAGE "(JOB RESTARTED)" HAD BEEN TYPED OUT, THEN THE JOB WOULD BE RESTARTED. ALTHOUGH THE OPERATION ITSELF WAS UNSUCCESSFUL, THE DESIRED RESULTS WERE ACHIEVED.

NO QUEUE AVAILABLE FOR JOB.

(FATAL) A JOB WAS SUBMITTED USING THE JOB COMMAND THAT DID NOT SPECIFY WHICH QUEUE THAT IT WAS TO BE SUBMITTED TO (NO -QUEUE OPTION), AND NO SUITABLE QUEUE COULD BE FOUND. SUITABILITY FOR A QUEUE INCLUDES CPU AND ELAPSED TIME LIMITS BEING WITHIN THE CONFINES OF THE QUEUE, QUEUE BEING BLOCKED, ETC. USE OF THE BATGEN -STATUS OR -DISPLAY COMMAND MAY YIELD A LIST OF LEGAL QUEUES AND THEIR STATUS IF THE FILE BATCHQ>BATDEF IS READ-PERMITTED.

NO QUEUES HAVE WAITING OR HELD JOBS.

(RESPONSE) A BATCH -DISPLAY COMMAND WAS ISSUED, AND THERE WERE NO QUEUES THAT HAD ANY WAITING OR HELD JOBS IN THEM. A QUEUE MAY HAVE ONE EXECUTING JOB IN IT, BUT AN EXECUTING JOB IS NOT CONSIDERED A WAITING OR HELD JOB.

NO RECENT JOBS [NAMED "<JOBNAME>"]

(RESPONSE) WILL HAVE EITHER "FOR USER <USERNAME>" OR "IN SYSTEM" APPENDED TO IT, DEPENDING ON WHETHER OR NOT THE USER IS LOGGED IN AS SYSTEM. THIS MESSAGE IS TYPED OUT BY A JOB -DISPLAY TODAY OR -STATUS TODAY COMMAND, AND INDICATES THAT THERE ARE NO JOBS BELONGING TO THAT USER (OR IN THE BATCH SYSTEM IF THE USER IS SYSTEM) THAT WERE SUBMITTED, INITIATED, ABORTED, COMPLETED OR CANCELLED TODAY.

THE TEXT IN BRACKETS ('NAME "<JOBNAME>"') IS OUTPUT IF A JOBNAME WAS SPECIFIED FOR THE -DISPLAY OR -STATUS COMMAND, OTHERWISE IT IS OMITTED.

NO RIGHT. MUST BE LOGGED IN AS SYSTEM

(FATAL) EITHER THE BATCH COMMAND WAS USED TO START OR STOP THE MONITOR, OR A -HOLD OR -RELEASE OPERATION WAS ATTEMPTED USING THE JOB COMMAND, AND THE USER WAS NOT LOGGED IN AS SYSTEM.

NO RUNNING JOBS.

(RESPONSE) A BATCH -DISPLAY COMMAND WAS ISSUED, AND THERE WERE NO JOBS THAT WERE CURRENTLY RUNNING. IT IS POSSIBLE FOR THERE TO BE NO RUNNING JOBS AND TO HAVE JOBS WAITING, HOWEVER, EVEN WHEN THE MONITOR IS RUNNING AND THERE ARE FREE PHANTOMS; THERE IS ALWAYS A SMALL AMOUNT OF TURNAROUND TIME BETWEEN THE SUBMITTAL OF A JOB AND THE EXECUTION OF A JOB. ON NEW PARTITIONS, THIS TIME IS ABOUT 20 SECONDS MAXIMUM, AND OLD PARTITIONS CAN TAKE UP TO A MINUTE AND A HALF.

NOT AN ABSOLUTE TREENAME.

(FATAL) THE HOME UFD SPECIFIED WITH THE -HOME OPTION DURING SUBMISSION USING THE JOB COMMAND (OR CHANGING OF JOB PARAMETERS) WAS A RELATIVE TREENAME, I.E. IT BEGAN WITH "*>". RE-SUBMIT THE JOB, GIVING AN ABSOLUTE PATHNAME AFTER THE -HOME OPTION.

NOT YOUR JOB.

(FATAL) A REFERENCE WAS MADE TO A JOB USING AN INTERNAL NAME IN THE JOB COMMAND, AND THE REFERENCED JOB DID NOT BELONG TO THE USER MAKING THE REFERENCE. USE "JOB -STATUS ALL" TO OBTAIN A LIST OF ALL JOBS BELONGING TO THE USER MAKING THE REQUEST.

NULL HOME UFD.

(FATAL) THE HOME UFD SPECIFIED WITH THE -HOME OPTION DURING SUBMISSION USING THE JOB COMMAND (OR CHANGING OF JOB PARAMETERS) WAS A NULL STRING. RE-SUBMIT THE JOB WITH AN ABSOLUTE PATHNAME AFTER THE -HOME OPTION.

OPERATOR START-UP.

(MESSAGE) THE MONITOR HAS SEEN THE REQUEST TO START VIA A BATCH SYSTEM -START COMMAND, AND IS NOW RUNNING.

OPERATOR STOP.

(MESSAGE) THE MONITOR RECEIVED A STOP REQUEST VIA A BATCH SYSTEM -STOP COMMAND. THE MONITOR WILL LOG OUT AFTER SENDING THIS MESSAGE. NOTE THAT THE MONITOR WILL RESPOND TO A STOP REQUEST EVEN IF IT HASN'T BEEN ISSUED THE BATCH SYSTEM -START REQUEST YET.

OUT OF RANGE.

(WARNING) WHILE IN BATGEN SUBCOMMAND MODE, A SUBCOMMAND WAS GIVEN WHICH EXPECTED A NUMERIC PARAMETER, BUT THE NUMBER WAS OUT OF RANGE FOR THAT SUBCOMMAND. THE RANGES ARE: 1 TO 126 FOR FUNIT, 0 TO 9 FOR PRIORITY, 1 TO 99 FOR TIMESLICE, AND 0 TO 7 FOR RLEVEL. RE-ENTER THE SUBCOMMAND WITH THE CORRECT PARAMETER. NOTE THAT THE FUNIT ARGUMENT, WHILE NORMALLY LIMITED TO 126, MAY HAVE A SMALLER UPPER LIMIT, DEPENDING ON THE COLD-START CONFIGURATION OF THE NUMBER OF AVAILABLE UNITS PER USER.

PLEASE FILE.

(WARNING) A QUIT COMMAND WAS ISSUED WHILE IN BATGEN COMMAND MODE, AND THE ENVIRONMENT HAD BEEN MODIFIED, SO THE QUESTION "ENVIRONMENT MODIFIED, OK TO QUIT?" WAS ASKED, AND THE ANSWER WAS "NO". THIS MESSAGE IS A REMINDER TO FILE OUT A MODIFIED ENVIRONMENT.

PLEASE RETURN.

(WARNING) A QUIT SUBCOMMAND WAS GIVEN WHILE IN BATGEN SUBCOMMAND MODE, AND BECAUSE THE QUEUE CHARACTERISTICS HAD BEEN MODIFIED, THE QUESTION "QUEUE DEFINITION MODIFIED, OK TO QUIT?" WAS ASKED, AND THE RESPONSE WAS "NO". THIS MESSAGE IS A REMINDER THAT THE PROPER WAY TO LEAVE A SUBCOMMAND SESSION IS TO USE THE RETURN SUBCOMMAND.

PLEASE STAND BY.

(RESPONSE) THIS MESSAGE AND OTHERS LIKE IT ("FILE IN USE, PLEASE STAND BY") WILL BE OUTPUT IF THE PROGRAM BEING RUN IS TRYING TO GAIN ACCESS TO A FILE THAT IS IN USE FOR MORE THAN 5 SECONDS. AFTER 20 SECONDS, THE "FILE IN USE..." MESSAGE WILL BE OUTPUT, AND AFTER 30 SECONDS, THE MESSAGE "TIMEOUT OF 30 SECONDS HAS OCCURRED" WILL BE OUTPUT AND THE PROGRAM WILL "GIVE UP". USUALLY THIS WILL RESULT IN A FATAL ERROR, AS IT COULD INDICATE THAT SYSTEM SECURITY IS BROKEN.

PLEASE WAIT.

(RESPONSE) THIS MESSAGE ASKS THAT THE USER BE PATIENT BECAUSE THE PROGRAM HE IS RUNNING HAS BEEN LOCKING UP THE BATCH DATABASE TOO LONG AND IS NOWING ALLOW OTHER PROCESSES TO HAVE ACCESS TO IT. IT IS NOT A FATAL ERROR. IT GENERALLY ONLY IS OUTPUT WHEN A SYSTEM IS HEAVILY LOADED, OR WHEN THE CURRENT PROCESS HAS A VERY LOW PRIORITY AND DOES NOT RUN FREQUENTLY.

PROCESS NOT STARTED.

(FATAL) A BATCH SYSTEM -STOP OR BATCH SYSTEM -START COMMAND WAS ISSUED, BUT THE BATCH MONITOR WAS NOT RUNNING. SEE PE-T-571 FOR A DESCRIPTION OF HOW TO START THE BATCH MONITOR UP.

QUEUE <NAME> ALREADY EXISTS (<STATUS>).

(WARNING) AN ATTEMPT WAS MADE TO ADD A QUEUE WHICH ALREADY EXISTED WHILE IN BATGEN COMMAND MODE. THE <STATUS> REFERRED TO IS EITHER "BLOCKED" OR "UNBLOCKED". TO CHANGE THE QUEUE DEFINITION, USE THE MODIFY SUBCOMMAND.

QUEUE <NAME> DELETED.

(MESSAGE) THE QUEUE REFERRED TO WAS FLAGGED FOR DELETION IN THE BATDEF FILE, AND HAS JUST BEEN DELETED BY THE BATCH MONITOR, BECAUSE THERE ARE NO LONGER ANY WAITING, HELD OR EXECUTING JOBS IN THAT QUEUE.

QUEUE <NAME> FLAGGED FOR DELETION.

(WARNING) AN ATTEMPT WAS MADE TO ADD OR DELETE A QUEUE WHICH HAD ALREADY BEEN DELETED, BUT WAS STILL FLAGGED FOR DELETION WHILE IN BATGEN COMMAND MODE. TO ALLOW THE QUEUE TO DISAPPEAR, FILE OUT THE BATDEF FILE, AND IT WILL DISAPPEAR WHEN THERE ARE NO MORE WAITING, HELD OR EXECUTING JOBS IN THAT QUEUE...THEN IT CAN BE ADDED AGAIN.

QUEUE BLOCKED.

(FATAL) THE QUEUE REFERRED TO BY A -QUEUE OPTION DURING JOB SUBMISSION IS CURRENTLY BLOCKED TO NEW SUBMISSIONS. TRY IT AGAIN LATER, OR USE ANOTHER QUEUE.

QUEUE DEFINITION MODIFIED, OK TO QUIT?

(QUERY) A QUIT SUBCOMMAND WAS GIVEN WHILE IN BATGEN SUBCOMMAND MODE, AND THE CHARACTERISTICS OF THE QUEUE BEING ADDED OR MODIFY'ED HAVE BEEN CHANGED. LEGAL ANSWERS TO THIS QUESTION ARE "YES", "NO" AND "OK". HITTING RETURN ALSO CAUSES THE QUIT TO BE TAKEN (I.E. "YES").

QUEUE DELETED.

(FATAL) THE QUEUE THAT THE JOB WAS BEING SUBMITTED TO WAS PRESENT WHEN IT WAS FIRST CHECKED OUT, BUT BY THE TIME THE COMMAND FILE HAD BEEN COPIED AND SOME OTHER ACTIVITIES HAD TAKEN PLACE, THE QUEUE HAD BEEN DELETED. THE JOB SHOULD BE RESUBMITTED TO A DIFFERENT QUEUE.

QUEUE DOES NOT EXIST.

(FATAL) THE -QUEUE OPTION ON THE JOB COMMAND LINE OR THE (OPTIONAL) $$ JOB LINE REFERRED TO A QUEUE THAT EITHER DID NOT EXIST OR WAS IN THE PROCESS OF BEING DELETED ("FLAGGED FOR DELETION"). THE BATCEN -STATUS OR -DISPLAY COMMAND SHOULD PROVIDE A LIST OF CURRENTLY AVAILABLE QUEUES AND THEIR STATUS, IF THE FILE THAT DEFINES QUEUES IS ACCESSIBLE BY USERS (I.E. READ-PERMITTED

FOR NON-OWNERS OF THE BATCHQ UFD).

QUEUE FULL.

(FATAL) THERE ARE ALREADY 10,000 JOBS (WHETHER ACTIVE OR INACTIVE) IN THE QUEUE THAT THE JOB IS BEING SUBMITTED TO BY THE JOB COMMAND. THE QUEUE MUST BE DELETED AND RE-CREATED BEFORE MORE JOBS CAN BE SUBMITTED TO IT. THE SYSTEM ADMINISTRATOR SHOULD BE ASKED TO DO THIS. MEANWHILE, IF ANY OTHER QUEUES ARE AVAILABLE, THEY CAN BE USED INSTEAD BY THE USER.

REGISTER SETTING.

(FATAL) REGISTER SETTINGS ARE ILLEGAL IN THE BATCH SUBSYSTEM (EXCEPT AS PART OF A SUBMITTED COMMAND FILE). RE-ENTER THE COMMAND LINE BUT WITHOUT THE REGISTER SETTING.

REMOVED <QUEUE-NAME> FROM BATDEF

(MESSAGE) THIS MESSAGE IS SENT TO THE SYSTEM CONSOLE WHEN THE BATCH MONITOR FINDS A QUEUE IN THE BATDEF FILE THAT IS FLAGGED FOR DELETION, BUT HAS NEVER HAD A JOB SUBMITTED TO IT. IT INDICATES THAT IT HAS DELETED THE QUEUE FROM BATDEF, BUT THAT NO JOB DATA WAS LOST AS A RESULT.

SEARCHING FOR FREE COMMAND FILE. PLEASE STAND BY.

(RESPONSE) THIS AND OTHER MESSAGES LIKE "QUEUE IS IN HEAVY USE...PLEASE STAND BY" MEAN THAT MANY USERS ARE SUBMITTING COMMAND FILES AT ONCE. THE SITUATION SHOULD RESOLVE ITSELF IN A SHORT AMOUNT OF TIME.

SPECIFIED VALUE IS OUT OF RANGE.

(FATAL) THE -CPTIME OR -ETIME OPTION SPECIFIED DURING JOB SUBMISSION OR A -CHANGE OPERATION IS GREATER THAN THE MAXIMUM ALLOWED BY THE QUEUE TO WHICH THE JOB WAS SUBMITTED. THIS MESSAGE WILL BE PRECEDED BY A MESSAGE INDICATING THE MAXIMUM LIMIT FOR THAT QUEUE ("CPU LIMIT IS XX" OR "ELAPSED TIME LIMIT IS XX"). IF THE LIMITS CANNOT BE LOWERED AND THE JOB SUCCESSFULLY RUN, THEN TRY A QUEUE WITH HIGHER LIMITS.

START-UP REQUEST ISSUED.

(RESPONSE) THE BATCH SYSTEM -START COMMAND HAS RESULTED IN THE BATCH MONITOR BEING REQUESTED TO START UP. THE MONITOR SHOULD EITHER RESPOND WITH "OPERATOR START-UP" WITHIN 10 SECONDS, OR IF

IT HAS ALREADY BEEN STARTED UP, A "START-UP REQUEST PREVIOUSLY PROCESSED." MESSAGE SHOULD BE SENT WITHIN A MINUTE AND A HALF.

START-UP REQUEST PREVIOUSLY PROCESSED.

(MESSAGE) THE MONITOR RECEIVED A START-UP REQUEST VIA A BATCH SYSTEM -START COMMAND, BUT IT HAD ALREADY BEEN STARTED. THIS MESSAGE IS JUST A REMINDER, NOT A FATAL ERROR.

STOP REQUEST ISSUED.

(RESPONSE) THE BATCH SYSTEM -STOP COMMAND HAS RESULTED IN THE BATCH MONITOR BEING REQUESTED TO STOP. WITHIN A MINUTE AND A HALF, THE MONITOR SHOULD SEND AN "OPERATOR STOP" MESSAGE TO THE SYSTEM CONSOLE AND LOG OUT.

SYNTAX ERROR. REGISTER SETTINGS ARE ILLEGAL

(WARNING) THIS MESSAGE IS OUTPUT IF END-OF-LINE IS EXPECTED AND A REGISTER SETTING IS FOUND INSTEAD. RE-ENTER THE COMMAND, BUT WITHOUT REGISTER SETTINGS.

SYSTEM TIME MUST BE SET FIRST.

(FATAL) IS OUTPUT WHEN A BATCH, BATGEN OR JOB COMMAND IS ISSUED FROM THE SYSTEM CONSOLE BEFORE THE SYSTEM DATE AND TIME ARE SET. NO PART OF THE BATCH SYSTEM CAN BE RUN UNTIL THE SYSTEM TIME IS SET USING THE SETIME COMMAND FROM THE SYSTEM CONSOLE. THE EXCEPTION IS THE BATCH MONITOR, WHICH WILL WAIT FOR THE SYSTEM TIME AND DATE TO BE SET BEFORE IT DOES ANYTHING ELSE.

THIS JOB CANNOT BE RESTARTED.

(RESPONSE) OUTPUT BY A JOB -DISPLAY COMMAND IF THE JOB BEING DISPLAYED HAS HAD A JOB -CANCEL DONE TO IT WHILE IT WAS EXECUTING, OR WAS SUBMITTED WITH THE -RESTART NO OPTION. ANY -RESTARTS DONE TO THE JOB WILL ABORT THE JOB (IF THEY SUCCEED), BUT THE JOB WILL NOT BE RESTARTED.

THIS JOB WILL BE RESTARTED.

(RESPONSE) OUTPUT BY A JOB -DISPLAY COMMAND IF THE JOB BEING DISPLAYED HAS HAD A JOB -RESTART DONE TO IT, BUT IT HAS NOT YET ABORTED OR COMPLETED AND IS STILL EXECUTING. WHEN THE MONITOR SEES THAT THE JOB HAS ABORTED/COMPLETED, IT WILL RETURN THE JOB TO THE "WAITING" STATE.

TOO MANY OPTIONS.

(FATAL) AT LEAST TWO OPTIONS WERE ENTERED THAT CONFLICTED WITH EACH OTHER, SUCH AS JOB -DISPLAY -CHANGE OR JOB C_TEST -ABORT -CANCEL. USE SEPARATE JOB COMMANDS TO PERFORM SEPARATE OPERATIONS.

TOO MANY QUEUES.

(WARNING) AN ATTEMPT WAS MADE TO ADD A QUEUE WHEN THERE WERE ALREADY SIX QUEUES (BLOCKED, UNBLOCKED OR FLAGGED FOR DELETION) DEFINED, USING THE ADD COMMAND IN BATGEN.

UNKNOWN COMMAND.

(WARNING) A COMMAND WAS ENTERED WHILE IN BATGEN COMMAND MODE THAT WAS UNRECOGNIZED. THE USER WILL BE LEFT IN BATGEN COMMAND MODE AND THE ERRONEOUS LINE WILL BE THROWN AWAY.

UNKNOWN OPTION.

(FATAL) AN OPTION WAS ENTERED TO THE BATCH OR JOB COMMAND THAT WAS NOT RECOGNIZED.

UNKNOWN QUEUE NAME.

(WARNING) A COMMAND ENTERED WHILE IN BATGEN COMMAND MODE REFERRED TO A QUEUE THAT EITHER DID NOT EXIST OR WAS "FLAGGED FOR DELETION" BY THE DELETE COMMAND.

UNKNOWN SUBCOMMAND.

(WARNING) WHILE IN BATGEN SUBCOMMAND MODE, A SUBCOMMAND WAS GIVEN WHICH WAS NOT RECOGNIZED. THE USER WILL BE LEFT IN SUBCOMMAND MODE.

UNRECOGNIZED OPTION.

(FATAL) BATGEN WAS INVOKED WITH AN OPTION ON THE COMMAND LINE THAT WAS NOT RECOGNIZED. THE ONLY LEGAL OPTIONS ARE -STATUS AND -DISPLAY.

WAITING FOR BATCH SYSTEM -START.

(MESSAGE) THIS MESSAGE IS OUTPUT WHEN THE BATCH MONITOR IS STARTED UP, AND AFTER IT HAS DETERMINED THAT THE SYSTEM DATE AND TIME HAVE

BEEN SET. IT SERVES AS A REMINDER FOR THE OPERATOR TO ISSUE THE BATCH SYSTEM -START COMMAND. IT IS ASSUMED THAT THE MONITOR HAS ALREADY BEEN CHAPED TO THE APPROPRIATE LEVELS WHEN THIS COMMAND IS ISSUED.

WARNING: JOBS ARE NOT BEING PROCESSED AT THIS TIME.

(PESPONSE) THIS MESSAGE MEANS THAT THE BATCH MONITOR IS NOT RUNNING, SO ANY SUBMITTED JOBS WILL NOT BE EXECUTED UNTIL IT IS START UP. THE OPERATION THAT THE USER REQUESTED WILL STILL BE PERFORMED. NOTE THAT IF THE MONITOR IS FORCED-LOGGED OUT, OR THE SYSTEM IS SHUT DOWN WITHOUT THE MONITOR LOGGING ITSELF OUT, THERE MAY BE A DATABASE PROBLEM AS A RESULT.

```
**********************************************************************************
**********************************************************************************
**
**
**
**      JJJ    III    M    M   M    M   Y    Y
**       J      I     MM  MM   MM  MM   Y    Y
**       J      I     M M M    M M M     Y Y
**       J      I     M M M    M M M      Y
**   J   J      I     M   M    M   M      Y
**   J   J      I     M   M    M   M      Y
**    JJ       III    M   M    M   M      Y
**
**
**
**   RRRR   BBBB    AAA   TTTTT   CCC   H   H  55555
**   R   R  B   B  A   A    T    C   C  H   H  5
**   R   R  B   B  A   A    T    C      H   H  5555
**   RRRR   BBBB   AAAAA    T    C      HHHHH      5
**   R R    B   B  A   A    T    C      H   H      5
**   R  R   B   B  A   A    T    C   C  H   H  5   5
**   R   R  BBBB   A   A    T     CCC   H   H   555
**
**
**
**********************************************************************************
**********************************************************************************
```

ADMINISTRATOR'S GUIDE TO REVISION 17 BATCH

DATE:

TO:

FROM:

SUBJECT: ADMINISTRATOR'S GUIDE TO REVISION 17 BATCH

REFERENCE:

ABSTRACT

THIS DOCUMENT IS A GUIDE TO THE SYSTEM ADMINISTRATOR WHO WANTS TO INSTALL REVISION 17 BATCH. IT EXPLAINS THE FILE STRUCTURE OF THE SUBSYSTEM, EXPLAINS THE PROCEDURE FOR STARTING UP THE BATCH SUBSYSTEM, AND SHOWS HOW TO CLEAN UP AFTER BATCH.

I. FILE SYSTEM STRUCTURE

A. FILES IN THE BATCH SUBSYSTEM

THE BATCH SYSTEM DIRECTLY INVOLVES 3 TOP-LEVEL UFDS ON EACH SYSTEM. THESE ARE CMDNCO, BATCH, AND BATCHQ. CMDNCO IS USED TO HOLD ONLY THE COMMANDS ($$, BATCH, BATGEN AND JOB) THEMSELVES. BATCH CONTAINS THE SOURCE CODE (FORTRAN WITH SOME PMA) FOR THE BATCH SUBSYSTEM INCLUDING THE COMMAND FILES NECESSARY TO BUILD AND INSTALL ITSELF. BATCHQ CONTAINS THE COMMAND FILES USED TO INITIALIZE THE BATCH DATABASE (WHICH WILL ALSO RESIDE IN BATCHQ).

HERE IS A LIST OF ALL FILES INVOLVING THE BATCH SUBSYSTEM:

UFD CMDNCO:

*	$$	THE $$ COMMAND (SIMPLY EXITS WHEN INVOKED).
*	BATCH	THE BATCH COMMAND (STATUS AND MODIFICATION OF PROCESSES).
*	BATGEN	THE BATGEN COMMAND (STATUS AND MODIFICATION OF QUEUES).
*	JOB	THE JOB COMMAND (STATUS AND MODIFICATION OF JOBS).

UFD BATCH:

	$$	THE SOURCE CODE (FTN) FOR THE $$ COMMAND.
	BATCH	THE SOURCE CODE (FTN) FOR THE BATCH COMMAND.
	BATGEN	THE SOURCE CODE (FTN) FOR THE BATGEN COMMAND.
	FIXBAT	THE SOURCE CODE (FTN) FOR THE *FIXBAT PROGRAM.
	INIT	THE SOURCE CODE (FTN) FOR THE *INIT PROGRAM.
	JOB	THE SOURCE CODE (FTN) FOR THE JOB COMMAND.
	MONITR	THE SOURCE CODE (FTN) FOR THE *MONITR PROGRAM.
	B$LIBF	THE SOURCE CODE (FTN) FOR COMMON SUBROUTINES.
	B$LIBP	THE SOURCE CODE (PMA) FOR COMMON SUBROUTINES.
	B$COMN	FORTRAN $INSERT FILE FOR COMMON DATA.
	B$EXEC	FORTRAN $INSERT FILE FOR EXECUTION STATUS FILE.
	B$JOBS	FORTRAN $INSERT FILE FOR JOB QUEUE ENTRY DEFINITION.
	B$KEYS	FORTRAN $INSERT FILE FOR COMMON DECLARATIONS.
	B$QCOM	FORTRAN $INSERT FILE FOR COMMON QUEUE-HANDLING DATA.
	B$QSTA	FORTRAN $INSERT FILE FOR QUEUE STATUS FILE DEFINITION.
	B$QUEU	FORTRAN $INSERT FILE FOR BATDEF FILE DEFINITION.
	C_BATCH	COMMAND FILE TO BUILD AND INSTALL THE BATCH SUBSYSTEM.
	C_LIST	COMMAND FILE TO GENERATE A LISTING OF THE BATCH SUBSYSTEM.

UFD BATCHQ:

	C_BDIF	COMMAND FILE TO INITIALIZE ENTIRE BATCH DATABASE.
	C_RSET	COMMAND FILE TO INITIALIZE BATCH JOB QUEUES (NOT BATDEF).
	PH_GO	PHANTOM FILE RUN BY BATCH MONITOR.
*	*MONITR	RUN-FILE IMAGE OF BATCH MONITOR.
*	*FIXBAT	PROGRAM TO FIX BATCH DATABASE UP.
*	*INIT	PROGRAM TO INITIALIZE QUEUE AND EXECUT FILES.
$	VALID.	DETERMINES VALIDITY OF BATCH DATABASE.
$	CIFILE	(SUB-UFD) CONTAINS COMMAND FILES AWAITING EXECUTION.

```
$  Q.CTRL   (SUB-UFD) CONTAINS QUEUE CONTROL FILES WITH JOB DATA.
$  QUEUE    CONTAINS CURRENT QUEUE INFO (# JOBS WAITING, ETC.).
$  EXECUT   CONTAINS INFO ON CURRENTLY EXECUTING JOBS.
#  BATDEF   BATCH DEFINITION FILE (ENVIRONMENT).
```

THE SPECIAL SYMBOLS IN FRONT OF THE FILENAMES HAVE THE FOLLOWING MEANINGS:

```
* - GENERATED BY THE BATCH>C_BATCH COMMAND FILE.
$ - GENERATED BY THE C_RSET AND C_BDIF FILES.
# - GENERATED BY THE C_BDIF FILE ONLY (USING BATGEN).
```

B. BUILDING THE BATCH SUBSYSTEM

TO BUILD (OR RE-BUILD) THE BATCH SUBSYSTEM, ATTACH TO THE BATCH UFD AND INVOKE THE COMMAND FILE C_BATCH AS FOLLOWS:

```
OK, ATTACH BATCH
OK, COMINPUT C_BATCH
```

THIS COMMAND FILE WILL COMPILE, ASSEMBLE, LOAD AND INSTALL ALL OF THE FILES MARKED WITH AN ASTERISK ("*") IN THE ABOVE LIST, DELETING TEMPORARY FILES, BINARIES, RUN-FILES, ETC. THAT IT GENERATED IN THE BATCH UFD.

AFTER THE C_BATCH COMMAND FILE IS FINISHED, ATTACH TO THE BATCHQ UFD AND INVOKE THE C_BDIF FILE AS FOLLOWS:

```
OK, ATTACH BATCHQ <OWNER-PASSWORD>
OK, CLOSE ALL
OK, COMINPUT C_BDIF
```

NOTE: THIS COMMAND FILE CANNOT BE EXECUTED UNLESS THE SYSTEM DATE AND TIME HAVE BEEN SET VIA THE OPERATOR SETIME COMMAND.

THIS COMMAND FILE WILL FIRST MAKE SURE THAT THE BATCH MONITOR IS NOT RUNNING (IF IT IS, LOG IT OUT, AND THEN TRY THE C_BDIF FILE AGAIN), THEN USE FUTIL TO TREDELETE THE CIFILE AND Q.CTRL SUB-UFDS, THEN IT WILL RE-CREATE THEM, DELETE ALL T$XXXX FILES IN BATCHQ, DELETE THE BATDEF, QUEUE AND EXECUT FILES, RUN *INIT TO RE-CREATE EMPTY QUEUE AND EXECUT FILES, THEN RUN BATGEN TO GENERATE AN EMPTY BATDEF FILE.

THE TWO NON-FATAL ERROR MESSAGES "FILE IN USE. IN.USE" FROM FUTIL AND "NOT FOUND. BATDEF" FROM BATGEN ARE EXPECTED AND ARE TO BE IGNORED.

WHEN IT FINISHES WITH THIS, IT WILL CREATE A FILE CALLED "VALID." WHICH INDICATES THAT THE BATCH DATABASE IS VALID SIMPLY BY EXISTING. IF THIS FILE IS DELETED BY ANYONE OR ANY PROGRAM (INCLUDING BATCH PROGPAMS TO INDICATE AN INVALID DATABASE), THE C_BDIF OR C_RSET FILES SHOULD BE USED AGAIN TO RE-INITIALIZE THE DATABASE, IF *FIXBAT FAILS IN THE ATTEMPT (SEE PE-T-622).

IF THE BATDEF FILE ALREADY CONTAINS USEFUL INFORMATION, AND IT IS NOT DESIREABLE TO HAVE THAT INFORMATION DESTROYED BY C_BDIF, THE COMMAND FILE C_RSET (RESET JOB QUEUES) CAN BE USED INSTEAD. IT BEHAVES EXACTLY LIKE C_BDIF, EXCEPT THAT IT LEAVES BATDEF INTACT.

AFTER EITHER OF THESE FILES HAVE BEEN EXECUTED (ASSUMING BATDEF IS VALID IF C_RSET WAS USED), THE BATCH SUBSYSTEM IS NOW BUILT AND FULLY INSTALLED.

SECTION II IS A REFERENCE TO A DOCUMENT THAT EXPLAINS HOW TO DEFINE QUEUES USING THE BATGEN COMMAND. ONCE THIS IS ACCOMPLISHED, SECTION III IS THE NEXT ONE TO REFERENCE (RUNNING THE BATCH MONITOR).

C. CHANGING PASSWORDS

SOME INSTALLATIONS MAY CHOOSE TO INCREASE SECURITY WITHIN THE BATCH SUBSYSTEM, AS IT CAN BE USED FOR POTENTIALLY EVIL PURPOSES (SUCH AS LOGGING A PHANTOM IN UNDER SOMEONE ELSE'S LOGIN-NAME).

TO DO THIS, OWNER PASSWORDS SHOULD BE GIVEN TO BATCHQ, BATCH, AND CMDNCO.

HOWEVER, SINCE THE BATCH SUBSYSTEM NEEDS ACCESS TO BATCHQ AND ITS SUB-UFDS, IT MUST KNOW THE BATCHQ PASSWORD. TO GIVE IT THE NEW PASSWORD, DO THE FOLLOWING:

```
OK, ATTACH BATCH <OWNER-PASSWORD>
OK, ED B$LIBF
EDIT
FIND C_ATCH$B
C ATCH$B, BATCH, JCB, 03/29/79
LOCATE BATPAS;*2
      INTEGER BATPAS(3),OPASS(3),NPASS(3),LSTNAM(17)
      DATA BATPAS/'<OLD-PASSWORD>'/
CHANGE /<OLD-PASSWORD>/<NEW-PASSWORD>/
      DATA BATPAS/'<NEW-PASSWORD>'/
FILE

OK,
```

NOTE THAT THE NEW PASSWORD MUST BE EXACTLY SIX CHARACTERS LONG (INCLUDING TRAILING BLANKS).

AFTER THIS CHANGE HAS BEEN MADE, FOLLOW THE PROCEDURE FOR BUILDING THE BATCH SUBSYSTEM DESCRIBED ABOVE IN SECTION I.B.

AFTER BUILDING THE BATCH SUBSYSTEM, THE PASSWORD SHOULD THEN BE SET IN THE BATCHQ UFD TO <NEW-PASSWORD>. THEN, ATTACH DOWN TO THE CIFILE AND Q.CTRL SUB-UFDS AND SET ANY RANDOM PASSWORDS IN THEM (THESE SUB-UFDS ARE WHERE THE REAL SENSITIVE INFORMATION EXISTS), SO THAT USERS WILL

NOT BE ABLE TO ATTACH DOWN TO THEM AS OWNERS.

AT THIS POINT, PROTECT THE PH_GO AND *FIXBAT FILES IN BATCHQ TO "7 1" PROTECTION, BECAUSE PH_GO IS REFERENCED AS A NON-OWNER WHEN THE BATCH MONITOR IS STARTED UP, AND THE FIRST THING IT DOES IS RUN BATCHQ>*FIXBAT. THE *FIXBAT PROGRAM WILL ATTACH TO BATCHQ AS AN OWNER (SINCE IT KNOWS THE PASSWORD), BUT ONLY IF THE USER IS LOGGED IN AS "SYSTEM", AND WAS SPAWNED AS A PHANTOM FROM THE SYSTEM CONSOLE (I.E. HAS PRIVILEGES). SEE PE-T-622 FOR DETAILS.

THE BATCH SUBSYSTEM DOES NOT NEED TO KNOW THE PASSWORDS OF THESE SUB-UFDS, SINCE IT CAN GET THEM (AND ALWAYS DOES) WHEN IT IS ATTACHED TO BATCHQ AS AN OWNER.

NOTE THAT INVOKING THE C_RSET OR C_BDIF FILES WILL RE-CREATE THE Q.CTRL AND CIFILE SUB-UFDS, WHICH RESETS THEIR PASSWORDS TO BLANKS. WHENEVER THESE COMMAND FILES ARE RUN, THE SUB-UFDS SHOULD BE RE-PASSWORDED BY THE SYSTEM ADMINISTRATOR.

THE BATCH UFD SHOULD ALSO BE PASSWORDED SO THAT USERS WHO UNDERSTAND THE BATCH SUBSYSTEM (OR WHO HAVE READ THIS DOCUMENT) MAY NOT LOOK AT THE B$LIBF FILE TO DETERMINE THE PASSWORD. NO SOFTWARE NEEDS TO KNOW THE BATCH PASSWORD. CMDNCO SHOULD ALSO HAVE A PASSWORD, ALTHOUGH THIS IS NOT A REQUIREMENT.

ONCE BATCHQ IS PASSWORDED, THE C_BATCH COMMAND FILE WILL NO LONGER WORK. IF IT IS DESIRED TO RE-BUILD THE BATCH SUBSYSTEM, EITHER REMOVE THE BATCHQ PASSWORD OR ENCODE IT IN THE C_BATCH FILE WHENEVER IT REFERENCES BATCHQ (IT ONLY DOES SO ONCE, AFTER A FUTIL COMMAND).

II. DEFINING THE BATCH ENVIRONMENT - THE BATGEN COMMAND

PLEASE REFERENCE SECTION 3 OF PE-T-584, "A BATCH SUBSYSTEM FOR PRIMOS", FOR A DESCRIPTION OF THE BATGEN COMMAND AND IT USE.

III. RUNNING THE BATCH MONITOR

A. INITIATING THE BATCH MONITOR

TO SPAWN THE BATCH MONITOR, ENTER THE COMMAND:

OK, PHANTOM BATCHQ>PH_GO

FROM THE SYSTEM CONSOLE.

THIS FILE WILL RUN THE BATCHQ>*FIXBAT PROGRAM TO FIX ANY PROBLEMS WITH THE BATCH DATABASE (AND OPTIONALLY DELETE OLD JOB ENTRIES, SEE PE-T-622 FOR DETAILS), THEN WHEN IT FINISHES THAT, THE *MONITR PROGRAM WILL BE RUN, WHICH WILL SEND THE FOLLOWING MESSAGE TO THE SYSTEM CONSOLE:

```
*BATCH* WAITING FOR BATCH SYSTEM -START.
```

AT THIS POINT, THE BATCH MONITOR SHOULD BE CHAPED TO THE APPROPRIATE LEVELS.

THESE LEVELS REFER TO THE "RLEVEL" AND THE "TIMESLICE" THAT ARE VALUES RELATED TO QUEUE DEFINITION USING THE BATGEN COMMAND.

THE FORMAT OF THE CHAP COMMAND IS:

```
CHAP -<USER-NUMBER> <RLEVEL> <TIMESLICE>
```

WHERE <USER-NUMBER> IS THE DECIMAL USER-NUMBER OF THE PHANTOM SPAWNED, <RLEVEL> IS A NUMBER FROM 0 TO 3, WHERE 1 IS THE INITIAL VALUE, 0 IS THE LOWEST, AND 3 IS THE HIGHEST, AND <TIMESLICE> IS THE OCTAL TIMESLICE OF THE PROCESS IN TENTHS OF A SECOND.

WHEN A PROCESS IS SPAWNED FROM A PARTICULAR QUEUE, IT INITIALLY IS GIVEN THE SAME RLEVEL AND TIMESLICE THAT ITS SPAWNER, THE BATCH MONITOR, HAS AT THE TIME OF THE SPAWNING. IT WILL SHORTLY THEREAFTER LOWER ITS RLEVEL BY THE "DELTA-RLEVEL" VALUE SPECIFIED IN THE RLEVEL SUBCOMMAND OF BATGEN FOR THAT PARTICULAR QUEUE (SEE PE-T-584), AND SET ITS TIMESLICE TO EITHER THE VALUE GIVEN IN THE BATGEN TIMESLICE SUBCOMMAND FOR THAT QUEUE OR THE TIMESLICE IT ALREADY HAS, WHICHEVER IS SMALLER.

THEREFORE, THE RLEVEL OF THE BATCH MONITOR REPRESENTS THE HIGHEST RLEVEL THAT ANY BATCH JOBS ON THE SYSTEM WILL EVER RUN AT (ALTHOUGH THIS SHOULD BE AT LEAST 1 TO ALLOW SPEEDY DISPATCHING OF JOBS), AND THE TIMESLICE SHOULD ALSO BE THE HIGHEST TIMESLICE EVER USED BY ANY JOB (THE MINIMUM SHOULD BE 20 DECIMAL, OR 2 SECONDS, WHICH IS 24 OCTAL).

ALL DELTA-RLEVELS IN THE BATDEF FILE GENERATED BY BATGEN KEY OFF OF THE ACTUAL RLEVEL OF THE MONITOR; THEREFORE, THE MONITOR'S RLEVEL WHILE RUNNING SHOULD BE DECIDED BEFORE THE DELTA-RLEVELS ARE CHANGED, AS ONE SET WILL AFFECT THE OTHER.

AN EXAMPLE CONFIGURATION MIGHT BE:

```
QUEUE DEFAULT DELTA-RLEVEL=2, TIMESLICE=20
QUEUE BACKGROUND DELTA-RLEVEL=3, TIMESLICE=50
QUEUE EXPRESS DELTA-RLEVEL=1, TIMESLICE=1
```

NOTE THAT THESE TIMESLICE VALUES ARE ALL IN DECIMAL REPRESENTATION...THE ONLY PLACE THAT A TIMESLICE IS REPRESENTED AS AN OCTAL NUMBER IS IN THE CHAP COMMAND.

THE OPERATOR WOULD PROBABLY ENTER THE COMMAND:

```
OK, CHAP -60 3 62  /* OCTAL TIMESLICE.
```

FROM THE SYSTEM CONSOLE AFTER SPAWNING THE PHANTOM. JOBS THAT WERE RUN FROM THOSE QUEUES WOULD RUN WITH THE FOLLOWING VALUES:

```
QUEUE DEFAULT RLEVEL=1, TIMESLICE=20 (NORMAL USER)
QUEUE BACKGROUND RLEVEL=0, TIMESLICE=50 (LONG JOBS)
QUEUE EXPRESS RLEVEL=2, TIMESLICE=1 (SHORT IMPORTANT JOBS)
```

NOTE THAT THE BATCH MONITOR WAS CHAPED UP ONE RLEVEL HIGHER THAN NECESSARY...THIS HELPS ELIMINATE ANY PROBLEMS HAVING TO DO WITH A BATCH JOB TYING UP THE CPU SO THAT THE MONITOR CAN'T RUN FAST ENOUGH. IN GENERAL, THIS SHOULD NOT BE A PROBLEM, BUT IT COULD HAPPEN.

THE TIMESLICES WERE LOWERED FOR THE "FASTER" JOBS TO PREVENT THEM FROM TYING UP THE CPU FOR TOO LONG, AND THEY WERE RAISED FOR THE "SLOWER" JOB TO GUARANTEE THEM HAVING A REASONABLE AMOUNT OF CPU TIME.

B. TELLING THE BATCH MONITOR TO START UP

AFTER THE CHAP COMMAND HAS BEEN ISSUED, THE MONITOR MUST BE INFORMED THAT EVERYTHING IS "READY" FOR IT TO BEGIN PROCESSING. THIS IS DONE WITH THE BATCH COMMAND AS FOLLOWS:

```
BATCH SYSTEM -START
```

THE PROGRAM SHOULD RESPOND:

```
START-UP REQUEST ISSUED.
```

IF IT DOES NOT, THEN THE MONITOR IS EITHER NOT RUNNING, ALREADY STARTED-UP, OR JUST SPAWNED AND NOT READY FOR START-UP. IN THE THIRD CASE, WAIT FOR THE "*BATCH* WAITING FOR BATCH SYSTEM -START" MESSAGE TO BE SENT TO THE SYSTEM CONSOLE, AND TRY THE COMMAND AGAIN.

WHEN THE BATCH MONITOR RECEIVES THE START-UP REQUEST, IT WILL SEND A MESSAGE TO THE SYSTEM CONSOLE SAYING:

```
*BATCH* OPERATOR START-UP.
```

MEANING THAT IT HAS STARTED RUNNING. IF IT SENDS A MESSAGE SAYING:

```
*BATCH* START-UP REQUEST PREVIOUSLY PROCESSED.
```

THEN A BATCH SYSTEM -START COMMAND HAD ALREADY BEEN ISSUED TO IT.

C. MONITORING THE BATCH MONITOR

TO DETERMINE THE GENERAL STATUS OF THE BATCH SYSTEM (AND MONITOR), USE THE BATCH COMMAND AS FOLLOWS:

```
BATCH [SYSTEM] -DISPLAY
```

IT WILL OUTPUT THE NUMBER OF WAITING AND HELD JOBS PER QUEUE, AND THE CURRENTLY EXECUTING JOBS INCLUDING THE USER NAME, INTERNAL NAME AND QUEUE IN WHICH THE JOB IS RUNNING.

WHENEVER THE MONITOR SPAWNS A JOB TO PROCESS A USER'S COMMAND FILE, THE SPAWNED JOB WILL SEND A MESSAGE TO THE SYSTEM CONSOLE AS FOLLOWS:

```
*BATCH* EXECUTING EXTNAM FOR USER USRNAM(JOBIDN).
```

WHERE EXTNAM IS THE EXTERNAL NAME, USRNAM IS THE SUBMITTING USER, AND JOBIDN IS THE INTERNAL NAME OF THE JOB.

WHEN THE JOB ABORTS (OR COMPLETES), THE MONITOR WILL SEND A MESSAGE TO THE SYSTEM CONSOLE AS FOLLOWS:

```
*BATCH* JOB EXTNAM FOR USER USRNAM(JOBIDN) ABORTED.
```

WHERE EXTNAM, USRNAM, AND JOBIDN ARE EXPLAINED ABOVE, AND "ABORTED" MAY BE REPLACED BY "COMPLETED" IF APPROPRIATE.

THESE MESSAGES WILL HELP THE OPERATOR IN MONITORING BATCH USAGE AND LOAD SOMEWHAT WITHOUT HAVING TO MAKE TOO MANY INQUIRIES.

TWO JOB OPTIONS ARE ALSO USEFUL FOR OBTAINING THE STATUS OF THE BATCH SYSTEM; THEY ARE -STATUS AND -DISPLAY. THE -STATUS OPTION ONLY RETURNS THE JOB NAME, THE EXTERNAL AND INTERNAL NAMES, THE QUEUE AND THE STATUS OF SPECIFIED JOB(S); WHILE THE -DISPLAY RETURNS ALL OF THE INFORMATION ON THE SPECIFIED JOB(S).

WHEN THE COMMAND IS ISSUED FROM ANY PROCES LOGGED IN UNDER SYSTEM (INCLUDING THE SYSTEM CONSOLE), IT IS CAPABLE OF DISPLAYING ALL OF THE JOBS ON THE SYSTEM. NORMALLY, A USER CAN ONLY DISPLAY JOBS WITH THE SAME USER-NAME.

THE FORMAT OF THESE COMMANDS IS:

```
JOB -STATUS  [ALL]
    -DISPLAY [TODAY]
```

IF THE ALL OR TODAY PARAMETER IS OMITTED, ONLY "ACTIVE" JOBS (THOSE JOBS THAT ARE WAITING, HELD OR COMPLETED) ARE DISPLAYED. IF ALL IS SPECIFIED, ALL JOBS IN THE SYSTEM ARE DISPLAYED. IF TODAY IS SPECIFIED, ONLY THOSE JOBS THAT WERE INITIATED, SUBMITTED, COMPLETED, ABORTED OR CANCELLED ON THE SAME DATE THAT THE COMMAND WAS ISSUED ARE DISPLAYED.

THE FORMAT OF THE DISPLAY IS TABULAR IF -STATUS WAS USED, OTHERWISE IT COMES OUT ENTRY BY ENTRY.

TO REFER TO SPECIFIC JOBS IN THE BATCH SYSTEM, THE INTERNAL NAMES OF THE JOBS MUST BE USED SINCE EXTERNAL NAME REFERENCES ARE ONLY TO JOBS BELONGING TO THE USER MAKING THE REFERENCE (IN THIS CASE, SYSTEM).

THEREFORE, THE FORMAT OF SPECIFICALLY LOOKING AT JOBS IS:

```
JOB JOBID -STATUS  [ALL]
          -DISPLAY [TODAY]
```

WHERE JOBID IS IN THE FORM #SNNNN (EXAMPLE: #10032). THESE INTERNAL NAMES ARE OUTPUT BY NEARLY ALL BATCH PROGRAMS WHEN THEY REFER TO JOBS (INCLUDING -STATUS AND -DISPLAY), SO THEY ARE RELATIVELY EASY TO ACCESS.

THE ALL AND TODAY SUFFIXES (AND THE ABSENCE THEREOF) HAVE THE SAME MEANING AS DESCRIBED ABOVE; THE DIFFERENCE IS THAT THERE IS A JOBID, WHICH LIMITS OUTPUT TO ONLY THOSE JOBS WITH THE SAME NAME (INTERNAL OR EXTERNAL) AS JOBID. EXTERNAL NAMES ARE ONLY COMPARED IF THE USER WHO SUBMITTED THE JOB IS THE SAME USER ISSUING THE JOB COMMAND (THEREFORE THE ONLY WAY FOR A USER LOGGED IN AS SYSTEM TO REFERENCE ANOTHER USER'S JOB IS TO USE THE INTERNAL NAME OF THAT JOB, OR DISPLAY IT BY NOT PROVIDING A JOBID AND THEREFORE DISPLAYING ALL JOBS).

IF IT IS DESIRED TO LOOK AT CURRENTLY DEFINED QUEUES, THE BATGEN -STATUS AND -DISPLAY COMMANDS SHOULD BE USED. THEY WILL OUTPUT SHORT AND LONG DESCRIPTIONS (RESPECTIVELY) OF THE CURRENTLY DEFINED QUEUES (-STATUS DOES IT IN TABULAR FORMAT) AS LONG AS THE BATDEF FILE IN BATCHQ IS READ-PERMITTED.

A TREENAME MAY BE SPECIFIED ON THE COMMAND LINE IF DESIRED, I.E.:

```
BATGEN [TREENAME] -STATUS
                  -DISPLAY
```

D. OPERATOR COMMANDS

THE OPERATOR HAS NEARLY FULL CONTROL OVER ALL JOBS IN THE BATCH SYSTEM.

HE CAN PERFORM ALL OF THE OPERATIONS THAT A USER CAN WHILE LOGGED IN UNDER SYSTEM UNDER THE FOLLOWING RESTRICTIONS:

1) HE MUST REFER TO ALL JOBS USING THEIR INTERNAL NAME.

2) HE CANNOT -ABORT OR -RESTART ANY JOBS UNLESS HE IS EITHER AT THE SYSTEM CONSOLE OR IS REFERENCING HIS OWN JOBS.

IF HE ATTEMPTS TO -ABORT A JOB FROM A TERMINAL THAT IS NOT THE SYSTEM CONSOLE WHILE LOGGED IN UNDER SYSTEM (ASSUMING THE JOB TO BE ABORTED IS

NOT LOGGED IN UNDER SYSTEM), THE ABORT WILL SIMPLY FAIL. IF HE ATTEMPTS TO -RESTART UNDER THE SAME CIRCUMSTANCES, THE JOB WILL SUCCESSFULLY BE FLAGGED AS BEING RESTARTABLE (UNLESS IT IS NOT RESTARTABLE), AND THEN IT WILL ALSO FAIL ON THE FORCE-LOGOUT ("INSUFFICIENT ACCESS RIGHTS").

EITHER WAY, THE INTEGRITY OF THE SYSTEM IS NOT DAMAGED; IF -RESTART WAS USED, THE JOB WILL BE RESTARTED WHEN IT COMPLETES (OR ABORTS); THE FORCE LOGOUT OF THE JOB IS ONLY TO SPEED UP THE PROCESS OF RESTARTING THAT JOB.

HERE IS A QUICK SUMMARY OF OPERATOR COMMANDS:

```
JOB JOBID -CANCEL      /* CANCEL A JOB.
          -ABORT       /* ABORT A JOB.
          -RESTART     /* RESTART A JOB.
          -HOLD        /* HOLD A JOB.
          -RELEASE     /* RELEASE A HELD JOB.
          -STATUS      /* STATUS OF A JOB.
          -DISPLAY     /* EXTENDED STATUS OF A JOB.
```

THE -HOLD AND -RELEASE OPTIONS ARE ONLY AVAILABLE TO THE OPERATORS. WHEN A JOB IS HELD, IT WILL STILL BE CONSIDERED AN "ACTIVE" JOB, AND IT WILL BE COUNTED IN THE LIST OF WAITING AND HELD JOBS GIVEN BY BATCH -DISPLAY (ALTHOUGH EXECUTING JOBS WILL NOT BE COUNTED HERE).

HOLDING A JOB IS USEFUL WHEN IT IS KNOWN THAT A RESOURCE THE JOB EXPECTS (MAGTAPE, DISK SPACE, LINE PRINTER, ETC.) IS NOT AVAILABLE.

WHEN THE RESOURCES ARE AVAILABLE, THE JOB CAN BE RELEASED WITH THE -RELEASE COMMAND.

E. STOPPING THE BATCH MONITOR

TO STOP THE BATCH MONITOR THE CLEANEST WAY POSSIBLE, USE THE BATCH COMMAND AS FOLLOWS:

```
BATCH SYSTEM -STOP
```

THIS WILL CAUSE THE MONITOR TO BE REQUESTED TO LOG ITSELF OUT. WHEN IT SEES THE REQUEST, IT WILL SEND A MESSAGE STATING THIS FACT TO THE SYSTEM CONSOLE ("OPERATOR STOP."), AND LOG ITSELF OUT.

IT IS DANGEROUS TO LOGOUT THE MONITOR (USING LOGOUT ALL OR LOGOUT -NN), SHUTDN ALL OR SHUTDN THE PARTITION ON WHICH BATCHQ RESIDES WHILE THE MONITOR IS RUNNING, SINCE IT MAY BE UPDATING IMPORTANT QUEUE INFORMATION, AND SUDDEN DEATH COULD INVALIDATE THE DATABASE.

IF THIS OCCURS, THE DATABASE WILL BE INACCESSIBLE BY USERS UNTIL THE MONITOR IS RESTARTED (WHICH RUNS *FIXBAT), *FIXBAT IS RUN "OFF-LINE", OR THE C_BDIF OR C_RSET FILES ARE INVOKED BY THE SYSTEM ADMINISTRATOR.

IV. CLEANING UP AFTER BATCH

AS JOBS ARE SUBMITTED TO THE BATCH MONITOR, THE JOB QUEUE FILES GROW LARGER AND LARGER. DEPENDING ON THE NUMBER OF JOBS SUBMITTED PER DAY, THE QUEUES COULD GET SO LARGE THAT A SIGNIFICANT AMOUNT OF DISK SPACE COULD BE WASTED ON THEM AS TIME PASSES.

THE QUICKEST WAY TO CLEAN UP THE BATCH DATABASE IS TO USE THE C_BDIF OR C_RSET FILES IN BATCHQ MENTIONED IN SECTION I.3.

HOWEVER, IT MAY SOMETIMES BE DESIREABLE TO CLEAN THINGS UP IN A SLOWER WAY. THIS CAN BE DONE ON A PER-QUEUE BASIS USING THE BATGEN DELETE COMMAND (DESCRIBED IN PE-T-584).

SIMPLY DELETE THE FULLEST QUEUE; THAT QUEUE WILL NO LONGER BE SUBMITTED TO BECAUSE IT WILL EFFECTIVELY BE DEAD AS FAR AS USERS ARE CONCERNED. HOWEVER, IT WILL NOT ACTUALLY CAUSE THE DELETION OF ALL QUEUE CONTROL FILES AND COMMAND FILES UNTIL THERE ARE NO WAITING OR EXECUTING JOBS IN THAT QUEUE.

THEREFORE, IF AT 7 P.M. THE QUEUE BACKGROUND WERE DELETED, AND IT HAD 9 HOUR-LONG JOBS WAITING IN IT, USERS WOULD NOT BE ABLE TO SUBMIT TO THE QUEUE AFTER 7 P.M. BUT THE 9 JOBS WOULD BE RUN, AND THEN, AT AROUND 3 A.M., THE QUEUE WOULD SIMPLY DISAPPEAR AFTER THE LAST JOB WAS RUN.

ANOTHER WAY TO CLEAN UP THE BATCH DATABASE IS TO RUN THE *FIXBAT PROGRAM, EITHER OFF-LINE OR BY MODIFYING BATCHQ>PH_GO (WHICH RUNS *FIXBAT) TO RUN IT WITH A -DAYS ARGUMENT. FOR DOCUMENTATION ON *FIXBAT, SEE PE-T-622.

A FAST WAY TO GET RID OF A SINGLE QUEUE IS TO CREATE A NEW BATDEF FILE THAT DOES NOT CONTAIN THE QUEUE; THIS CAN'T BE DONE WITH DELETE, SINCE, AS WAS STATED ABOVE, THE QUEUE DOES NOT ACTUALLY GO AWAY UNTIL THE MONITOR DELETES IT. HOWEVER, IF THERE WERE 3 QUEUES DEFINED, DEFAULT, EXPRESS, AND BACKGROUND, AND IT WAS DESIRED TO IMMEDIATELY CLEAN OUT THE BACKGROUND QUEUE, THEN THE BATGEN PROGRAM WOULD BE INVOKED USING SOME NON-EXISTENT TREENAME AS THE FILE (TO START WITH A NULL ENVIRONMENT).

THEN, THE QUEUES DEFAULT AND EXPRESS WOULD BE ADDED EXACTLY AS THEY APPEARED IN THE ACTUAL "LIVE" BATDEF FILE.

THEN THE FILE COMMAND WOULD BE USED, WITH THE DESTINATION BEING BATCHQ>BATDEF.

AS SOON AS THE MONITOR SEES THAT A QUEUE IS GONE FROM BATDEF, IT WILL IMMEDIATELY ABORT ANY JOB RUNNING IN THAT QUEUE, AND WHEN THOSE ABORTIONS ARE PROCESSED, OR IF THERE WERE NO EXECUTING JOBS IN THAT QUEUE, ALL QUEUE FILES AND COMMAND FILES RELATED TO THAT QUEUE WILL BE

DELETED WHETHER OR NOT THERE ARE ANY WAITING JOBS.

V. COMPATIBILITY WITH CX

THERE IS NO COMPATIBILITY WITH CX RELATED TO OPERATOR CONTROL OVER THE SYSTEM, SINCE CX HAD NONE, WITH THE EXCEPTION OF THE ABILITY TO FORCE-LOGOUT A JOB.

HOWEVER, IF CX IS RUN SIMULTANEOUSLY WITH BATCH ON THE SAME SYSTEM, A SERIOUS SECURITY ISSUE COULD RESULT:

THAT IS, ALL CX SLAVES (JOBS) ARE LOGGED IN AS SYSTEM; AND THE BATCH SUBSYSTEM RECOGNIZES ANY USER LOGGED IN AS SYSTEM AS AN OPERATOR.

THEREFORE, A MALICIOUS USER COULD CONCEIVABLY SUBMIT A JOB TO CX THAT OBTAINED STATUS OF ALL JOBS, CANCELLED OR ABORTED OTHER USERS JOBS, OR EVEN STOP THE MONITOR ITSELF USING CX.

THIS SAME PROBLEM EXISTS WITH THE SPOOLER PHANTOM, WHICH RECOGNIZES ANY PROCESS WITH THE SAME LOGIN NAME AS IT HAS AS A "PRIVILEGED" PROCESS, AND GENERALLY THIS LOGIN NAME IS SYSTEM.

```
**********************************************************************************
**********************************************************************************
**
**
**
**      JJJ    III    M    M   M    M   Y    Y
**       J      I     MM  MM   MM  MM   Y    Y
**       J      I     M M M    M M M     Y Y
**       J      I     M M M    M M M      Y
**    J  J      I     M    M   M    M     Y
**    J  J      I     M    M   M    M     Y
**     JJ      III    M    M   M    M     Y
**
**
**
**    RRRR    BBBB     AAA    TTTTT    CCC    H    H    666
**    R   R   B   B   A   A     T     C   C   H    H   6
**    R   R   B   B   A   A     T     C       H    H   6
**    RRRR    BBBB    AAAAA     T     C       HHHHH   6666
**    R R     B   B   A   A     T     C       H    H   6   6
**    R  R    B   B   A   A     T     C   C   H    H   6   6
**    R   R   BBBB    A   A     T      CCC    H    H    666
**
**
**
**********************************************************************************
**********************************************************************************
```

CHANGES TO BATCH FOR REVISION 17.1

DATE:

TO:

FROM:

SUBJECT: CHANGES TO BATCH FOR REVISION 17.1

REFERENCE:

ABSTRACT

BATCH HAS BEEN CHANGED AT REVISION 17.1 TO RUN ON A SYSTEM THAT HAS AT LEAST 16 (DECIMAL) FILE UNITS CONFIGURED PER USER (NOT COUNTING COMMAND OUTPUT).

THE CONFIGURATION DIRECTIVE FILUNT IN THE CONFIG FILE CAN BE USED BY THE SYSTEM ADMINISTRATOR TO LOWER THE MAXIMUM FILE UNIT NUMBER FROM 127. BATCH WILL SUCCESSFULLY RUN ON ANY REVISION 17 SYSTEM THAT HAS THIS PARAMETER AT LEAST AS HIGH AS 16 (DECIMAL).

THE ARGUMENTS TO THE JOB OPTION -FUNIT AND THE BATGEN SUBCOMMAND FUNIT ARE LIMITED TO THIS NUMBER, BUT AT REVISION 17.0, THIS LIMIT INCLUDED THE COMMAND OUTPUT UNIT, AND AT REVISION 17.1 THIS BUG HAS BEEN FIXED.

AN EXAMPLE ERROR IS:

```
JOB C FOO -FUNIT 17
[JOB REV 17.1]
17 IS OUT OF RANGE.  -FUNIT (JOB)
ER!
```

ALSO, A BUG IN FIXBAT HAS BEEN FIXED WHICH CAUSED IT TO ATTEMPT TO USE VERY HIGH UNIT NUMBERS, WHICH WOULD NOT BE AVAILABLE ON A 16-UNIT SYSTEM.

IT INTERNALLY ALLOCATES UNITS, BUT USED TO FORGET TO DE-ALLOCATE THEM ON CERTAIN TYPES OF ERRORS.

```
**********************************************************************************************
**********************************************************************************************
**
**
**
**      JJJ    III    M    M   M    M   Y    Y
**       J      I     MM  MM   MM  MM   Y    Y
**       J      I     M M M    M M M     Y Y
**       J      I     M M M    M M M      Y
**    J  J      I     M    M   M    M     Y
**    J  J      I     M    M   M    M     Y
**     JJ       III   M    M   M    M     Y
**
**
**
**    M    M    III   DDDD     AAA     SSS          DDDD     000     CCC
**    MM  MM     I    D   D   A   A   S   S         D   D   0   0   C   C
**    M M M      I    D   D   A   A   S             D   D   0   0   C
**    M M M      I    D   D   AAAAA    SSS          D   D   0   0   C
**    M   M      I    D   D   A   A       S        D   D   0   0   C
**    M   M      I    D   D   A   A   S   S   ..   D   D   0   0   C   C
**    M   M    III   DDDD    A   A    SSS    ..    DDDD     000     CCC
**
**
**
**********************************************************************************************
**********************************************************************************************
```

TABLE OF CONTENTS

MIDAS REV. 17

DATE: MAY 31, 1979

TO:

FROM:

SUBJECT: MIDAS REV. 17

REFERENCE: NONE

ABSTRACT

CONCURRENT PROCESS HANDLING AND THE DETECTION AND CORRECTION OF CONCURRENCY ERRORS ARE THE TWO MAJOR AREAS OF MODIFICATION IN MIDAS AT REV 17. DESIGNED TO PROVIDE A SUBSTANTIAL PERFORMANCE IMPROVEMENT, THE NEW CONCURRENT PROCESS HANDLING METHOD WILL REQUIRE SLIGHT MODIFICATION OF FORTRAN AND PMA MIDAS APPLICATION PROGRAMS. THE NEW METHOD WILL BE AVAILABLE TO COBOL, RPG II, AND BASICV USERS AT REV 17.1. USERS WHO WISH, MAY EASILY DISABLE THE NEW METHOD AND, AS A RESULT, EMPLOY THE CONCURRENT PROCESS HANDLING METHOD AVAILABLE IN PREVIOUS RELEASES.

THE SECOND REV 17 CHANGE ALLOWS MIDAS IN MOST CASES TO DETECT AND CORRECT CONCURRENCY ERRORS.

SECTION 1 DISCUSSES THE NEW CONCURRENT PROCESS HANDLING METHOD AND ITS IMPACT ON USER APPLICATIONS AND OPERATIONS. SECTION 2 DESCRIBES HOW MIDAS DETECTS AND CORRECTS CONCURRENCY ERRORS. INSTALLATION METHODS AND CONSIDERATIONS ARE DISCUSSED IN SECTION 3.

THIS PAGE RESERVED FOR THE TABLE OF CONTENTS.

1 HANDLING OF CONCURRENT MIDAS PROCESSES

1.1 OVERVIEW

IN PREVIOUS RELEASES, MIDAS COORDINATED CONCURRENT PROCESSES BY GATING PROCESSES AT THE SEGMENT SUBFILE LEVEL. (EG. A MIDAS FILE INDEX) THIS METHOD RELIED UPON FILE SYSTEM READ/WRITE LOCKS AND REQUIRED THAT SEGMENT SUBFILES BE OPENED AT THE START OF EACH MIDAS FILE OPERATION AND CLOSED UPON COMPLETION OF THE OPERATION. FOR EXAMPLE, TO RETRIEVE A RECORD, MIDAS OPENED THE INDEX SEGMENT SUBFILE(S) AND THE DATA SEGMENT SUBFILE. WHEN THE RETRIEVAL COMPLETED, MIDAS CLOSED THESE SEGMENT SUBFILES.

THE NEW CONCURRENT PROCESS HANDLING METHOD PROVIDES IMPROVED PERFORMANCE BY GREATLY REDUCING THE NUMBER OF FILE SYSTEM CALLS. THROUGH USE OF A SEMAPHORE AND A "LOCK" IN SHARED MEMORY, MIDAS SIMPLY ALLOWS ONLY ONE PROCESS AT A TIME TO EXECUTE A MIDAS FILE OPERATION. THEREFORE, MIDAS SEGMENT SUBFILES NEED NOT BE CLOSED AT THE END OF EACH OPERATION ONLY TO BE REOPENED AT THE START OF THE NEXT CALL. DETAILS OF THE NEW METHOD ARE DESCRIBED IN SECTION 1.2.

THE NEW METHOD OF HANDLING CONCURRENT PROCESSES REQUIRES THAT MIDAS BE NOTIFIED BOTH WHEN A PROCESS IS TO BEGIN USING A MIDAS FILE AND WHEN THE PROCESS HAS COMPLETED OPERATIONS ON THE FILE. FOR FORTRAN AND PMA USERS OF THE MIDAS CALL LEVEL INTERFACE, THIS REQUIREMENT MEANS THAT APPLICATION PROGRAMS MUST BE MODIFIED. SECTION 1.3 DESCRIBES METHODS OF MAKING THESE CHANGES. IMPORTANT INSTALLATION INSTRUCTIONS ARE DETAILED IN SECTION 3. IT SHOULD BE NOTED THAT USERS WHO DO NOT WISH TO MAKE APPLICATION PROGRAM CHANGES MAY DISABLE THE NEW METHOD OF HANDLING CONCURRENT PROCESSES AND THUS RETURN TO THE METHOD EMPLOYED BY PREVIOUS MIDAS RELEASES. THE PROCEDURE FOR DISABLING THE NEW METHOD IS DESCRIBED IN SECTION 3.4.

1.2 IMPLEMENTATION METHOD

TO MAINTAIN FILE INTEGRITY, MIDAS MUST SYCHRONIZE CONCURRENT PROCESSES. IN PREVIOUS RELEASES OF MIDAS, THIS SYNCHRONIZATION WAS ACCOMPLISHED BY OPENING FILE SEGMENTS FOR READING AND WRITING. SINCE FILE READ/WRITE LOCKS WERE SET TO 2 (N READERS AND ONE WRITER), ONLY ONE PROCESS COULD ACCESS A FILE SEGMENT AT A TIME. A SECOND PROCESS WAS ONLY ABLE TO PROCEED WHEN THE FIRST PROCESS FINISHED ITS MIDAS OPERATION AND THE FILE SEGMENTS WERE CLOSED. THIS METHOD OF SYNCHRONIZATION REQUIRED MANY CALLS TO THE FILE SYSTEM ROUTINE SRCH$$ TO OPEN AND CLOSE FILE SEGMENTS AND THUS IMPOSED A SIGNIFICANT PERFORMANCE PENALTY.

AT REV 17 MIDAS DOES NOT CLOSE FILE SEGMENTS BETWEEN MIDAS OPERATIONS. THIS, HOWEVER, REQUIRES THAT MIDAS FILE READ/WRITE LOCKS BE SET TO 3 (N READERS AND M WRITERS). OTERWISE, CONCUPRENT PROCESSES WOULD BE UNABLE TO OPEN A FILE SEGMENT WHICH HAD BEEN ALREADY OPENED BY ANOTHER PROCESS.

WITH FILE READ/WRITE LOCKS SET TO 3, FILE INTEGRITY COULD BE DESTROYED. THIS WOULD HAPPEN, FOR INSTANCE, IF TWO PROCESSES BOTH READ THE SAME RECORD AND THEN BOTH UPDATE THE RECORD. IN THIS CASE THE FIRST UPDATE WOULD BE LOST. TO PREVENT LOSS OF FILE INTEGRITY, MIDAS EMPLOYS A METHOD OF HANDLING CONCURRENT PROCESSES WHICH DOES NOT DEPEND ON OPENING AND CLOSING FILE UNITS.

IN THE NEW METHOD WHEN MIDAS IS CALLED, A CHECK IS DONE TO SEE IF ANY OTHER PROCESS IS USING MIDAS. TO DO THIS CHECK, MIDAS TESTS A "LOCK" LOCATED IN A SHARED MEMORY SEGMENT. A ZERO VALUE INDICATES THAT MIDAS IS AVAILABLE. IF NON-ZERO, THE LOW ORDER 15 BITS IS THE USER NUMBER OF THE PROCESS CURRENTLY ACCESSING MIDAS. (NOTE: BIT ONE IS ALWAYS SET WHEN MIDAS IS IN USE.) WHEN THE RESULT OF THE LOCK TEST IS ZERO, THE LOCK IS SET TO INDICATE THAT THE CURRENT PROCESS (DOING THE CHECK) NOW HAS SOLE ACCESS TO MIDAS. THIS "TEST AND SET" OPERATION IS NON-INTERRUPTIBLE. THEREFORE A PROCESS CANNOT MODIFY THE LOCK VALUE BETWEEN THE TIME THAT ANOTHER PROCESS HAS TESTED AND SET THE LOCK VALUE. IF THE TEST AND SET OPERATION IS SUCCESSFUL, THE PROCESS IS SAID TO HAVE "OBTAINED" THE LOCK.

IF WHEN TESTED, THE LOCK IS NON-ZERO, THE TESTING PROCESS MUST WAIT UNTIL MIDAS BECOMES AVAILABLE. TO ACCOMPLISH THIS, THE PROCESS IS SUSPENDED AND PUT ON A SEMAPHORE WAIT LIST. THE WAIT LIST FORMS A QUEUE OF PROCESSES WAITING TO BEGIN A MIDAS OPERATION. EACH TIME AN OPERATION COMPLETES, THE LOCK IS RELEASED, IE. THE LOCK VALUE IS SET TO ZERO. A PROCESS IS THEN REMOVED FROM THE WAIT LIST. THE RESTARTED PROCESS AGAIN MUST ATTEMPT TO OBTAIN THE LOCK.

1.3 APPLICATION IMPLICATIONS

1.3.1 APPLICATION PROGRAM MODIFICATIONS

WHEN REV 17 MIDAS IS INSTALLED, USERS MUST RELOAD ALL APPLICATION PROGRAMS WHICH USE AN UNSHARED MIDAS LIBRARY. IN ADDITION, TO OBTAIN THE POTENTIAL PREFORMANCE INCREASE, USERS MUST MODIFY FORTRAN AND PMA MIDAS APPLICATION PROGRAMS. THE MODIFICATIONS INVOLVE INSERTING SUBROUTINE CALLS TO NOTIFY MIDAS THAT FILE SEGMENTS ARE NOT TO BE CLOSED BETWEEN CALLS TO MIDAS. WHILE NOT ALL APPLICATION PROGRAMS NEED TO BE MODIFIED IMMEDIATELY, ALL PROGRAMS WHICH USE AN UNSHARED MIDAS LIBRARY MUST BE RELOADED WHEN REV 17 IS INSTALLED.

USERS MAY CHOOSE FROM TWO METHODS OF PROGRAM MODIFICATION. THE FIRST INVOLVES INSERTING CALLS TO SUBROUTINE NTFYM$. THE FIRST CALL SHOULD BE INSERTED FOLLOWING THE CALL TO OPEN THE MIDAS FILE BUT BEFORE THE FIRST MIDAS FILE OPERATION. THE OTHER CALL TO NTFYM$ SHOULD BE INSERTED JUST BEFORE THE CALL TO CLOSE THE MIDAS FILE. FOR FURTHER DETAILS REFER TO THE SECTION WHICH DESCRIBES SUBROUTINE NTFYM$.

THE SECOND METHOD IS TO REPLACE THE CALLS WHICH OPEN AND CLOSE A MIDAS FILE WITH CALLS TO OPENM$ AND CLOSM$ RESPECTIVELY. DETAILS

ARE PROVIDED IN THE SECTIONS WHICH DESCRIBE OPENM$ AND CLOSM$. SECTION 1.3.2 INCLUDES EXAMPLES TO ILLUSTRATE THE USE OF THESE ROUTINES.

MIDAS AT REV 17 SUPPORTS R MODE APPLICATIONS. HOWEVER, BECAUSE THE R MODE MIDAS LIBRARY ENTERS V MODE TO DO A PORTION OF THE CONCURRENT PROCESS HANDLING, REV 17 MIDAS WILL NOT WORK ON A PRIME P-300.

1.3.1.1 NTFYM$

****
*   *
* NTFYM$ *
*   *
****

FUNCTION

NOTIFY MIDAS THAT A MIDAS FILE (SEGMENT DIRECTORY) HAS BEEN OPENED OR IS ABOUT TO BE CLOSED BY THE USER.

CALLING SEQUENCE

```
CALL NTFYM$ (KEY, UNIT, STATUS)
```

KEY -- (INPUT) SPECIFIES WHETHER THE FILE HAS BEEN OPENED OR IS ABOUT TO BE CLOSED.
1 - FILE HAS BEEN OPENED
2 - FILE IS ABOUT TO BE CLOSED

UNIT -- (INPUT) FILE UNIT ON WHICH THE FILE IS OPEN

STATUS -- (OUTPUT) ERROR STATUS
0 - NO ERROR
10001 - BAD PARAMETER
10002 - TOO MANY MIDAS FILES OPEN SIMULTANEOUSLY MAY OCCUR ONLY IF KEY IS 1.

DISCUSSION

1. A CALL TO NTFYM$ AFTER A MIDAS FILE HAS BEEN OPENED NOTIFIES MIDAS THAT IT SHOULD LEAVE OPEN BETWEEN MIDAS CALLS ANY OF THE SPECIFIED FILE'S SEGMENT SUBFILES WHICH IT OPENS DURING SUBSEQUENT FILE ACCESS.

2. A CALL TO NTFYM$ BEFORE A MIDAS FILE IS CLOSED NOTIFIES MIDAS THAT IT SHOULD CLOSE ANY OF THE FILE'S SEGMENT SUBFILES THAT IT HAS LEFT OPEN.

3. IF THE MIDAS LIBRARY HAS BEEN CUSTOMIZED TO DISABLE INTERNAL LOCKING, A CALL TO NTFYM$ HAS NO EFFECT.

4. NTFYM$ IS MOST USEFUL IN THOSE APPLLICATIONS WHICH USE GENERAL PROCEDURES FOR FILE OPENING AND CLOSING.

1.3.1.2 OPENM$

```
**********
*        *
* OPENM$ *
*        *
**********
```

FUNCTION

OPENS A MIDAS FILE (SEGMENT DIRECTORY) AND, UNLESS THE MIDAS LIBRARY HAS BEEN CUSTOMIZED TO DISABLE INTERNAL LOCKING, CAUSES MIDAS TO LEAVE OPEN BETWEEN MIDAS CALLS ANY OF THE FILE'S SEGMENT SUBFILES WHICH IT OPENS DURING SUBSEQUENT FILE ACCESS. OPENM$ VERIFIES THAT THE SPECIFIED FILE EXISTS AND THAT IT IS OF THE APPROPRIATE TYPE, IE. SAM SEGMENT DIRECTORY.

CALLING SEQUENCE

```
CALL OPENM$ (KEY, TRENAM, NAMLEN, UNIT, STATUS)
```

KEY -- (INPUT) VALID SRCH$$ ACTION SUB-KEY (K$READ, K$WRIT, OR K$RDWR, OPTIONALLY TOGETHER WITH K$GETU)

TRENAM -- (INPUT) TREE NAME OF FILE TO BE OPENED

NAMLEN -- (INPUT) LENGTH OF TREE NAME IN CHARACTERS

UNIT -- (INPUT) IF K$GETU IS NOT SPECIFIED, THEN UNIT IS THE FILE UNIT ON WHICH THE FILE IS TO BE OPENED.
(OUTPUT) IF K$GETU IS SPECIFIED, UNIT IS THE FILE UNIT ON WHICH THE FILE WAS OPENED.

STATUS -- (OUTPUT) ERROR STATUS

0	- NO ERROR
< 10001	- FMS ERROR (SYSTEM DEFINED)
= 10001	- BAD KEY
= 10002	- TOO MANY MIDAS FILES OPEN SIMULTANEOUSLY
= 10003	- SPECIFIED FILE IS NOT A MIDAS SEGMENT DIRECTORY

1.3.1.7 CLOSM$

```
**********
*        *
* CLOSM$ *
*        *
**********
```

FUNCTION

CLOSES A MIDAS FILE (SEGMENT DIRECTORY) OPEN ON A SPECIFIED FILE UNIT AND, UNLESS THE MIDAS LIBRARY HAS BEEN CUSTOMIZED TO DISABLE INTERNAL LOCKING, CLOSES ANY OF THE FILE'S SEGMENT SUBFILES WHICH MIDAS HAS OPENED DURING THE COURSE OF FILE ACCESS.

CALLING SEQUENCE

```
CALL CLOSM$ (UNIT, CODE)
```

UNIT -- (INPUT) FILE UNIT ON WHICH THE MIDAS FILE IS OPEN

CODE -- (OUTPUT) ERROR STATUS
= 0 - NO ERROR
> 0 - FMS ERROR (SYSTEM DEFINED)

1.3.2 EXAMPLES

1.3.2.1 USE OF NTFYM$

IN THIS FORTRAN EXAMPLE THE PROGRAM OPENS FILE FNAME ON UNIT UNIT. VAPIABLE TYPE HAS PREVIOUSLY BEEN SET TO A VALUE WHICH DESCRIBES THE TYPE OF FILE OPENED. IF THE FILE IS OF TYPE "MIDAS", THE PROGRAM CALLS NTFYM$ TO NOTIFY MIDAS THAT IT IS READY TO BEGIN OPERATIONS ON THE FILE. AFTER PROCESSING HAS BEEN COMPLETED, THE PROGRAM NOTIFIES MIDAS OF THE FACT AND THEN CLOSES THE FILE.

```
C       OPEN THE FILE
        CALL SRCH$$(K$READ,FNAME,6,UNIT,FTYPE, CODE)
        IF (CODE .NE. 0) GO TO 9000
        IF (TYPE .NE. MIDAS) GO TO 200/* CHECK FILE TYPE
        CALL NTFYM$(1,UNIT,CODE) /* TELL MIDAS WE'RE READY
        IF (CODE .NE. 0) GO TO 9002
200     CONTINUE
        .
        .
        .
C       DO MIDAS FILE PROCESSING (EG. CALLS TO FIND$)
        .
        .
        .
        IF (TYPE .NE. MIDAS) GO TO 800
        CALL NTFYM$(2,UNIT,CODE)  /* TELL MIDAS PROCESSING IS D
ONE
800     CONTINUE
        CALL SRCH$$(K$CLOS,0,C,UNIT,TYPE,CODE) /* CLOSE FILE
        .
        .
        .
```

1.3.2.2 USE OF OPENM$ AND CLOSM$

THIS PROGRAM USES OPENM$ TO OPEN FILE FNAME ON UNIT UNIT AND AT THE SAME TIME NOTIFY MIDAS THAT PROCESSING IS ABOUT TO BEGIN. AFTER PROCESSING HAS BEEN COMPLETED, THE PROGRAM CALLS CLOSM$ TO NOTIFY MIDAS THAT PROCESSING HAS BEEN COMPLETED AND TO CLOSE THE FILE.

```
C       OPEN THE FILE AND NOTIFY MIDAS THAT WE'RE READY
C       TO USE THE FILE.
        CALL OPENM$(K$READ,FNAME,6,UNIT,CODE)
        IF (CODE .NE. 0) GO TO 9000
        .
        .
        .
C       DO MIDAS FILE PROCESSING (EG. CALLS TO FIND$)
        .
        .
        .
        CALL CLOSM$(UNIT,CODE) /* TELL MIDAS WE'RE DONE
C                                  AND CLOSE THE FILE
        .
        .
        .
```

1.3.3 ADMINISTRATION CHANGES

1.3.3.1 OVERVIEW

USERS MUST PERFORM TWO TYPES OF MIDAS INITIALIZATION PROCEDURES. WHEN DOING A COLD START, THE SEGMENT CONTAINING THE LOCK MUST BE SHARED, THE LOCK VALUE MUST BE SET TO ZERO AND THE SEMAPHORE DRAINED. INITIALIZATION OF THE SEMAPHORE AND SHARED LOCK IS HANDLED BY MIDAS UTILITY IMIDAS. FOR DETAILS REFER TO SECTION 1.3.3.2.

THE SECOND TYPE OF INITIALIZATION IS NECESSARY IF AN APPLICATION PROGRAM ABNORMALLY TERMINATES AND AS A CONSEQUENCE FAILS TO RELEASE THE SHARED LOCK. IF THE LOCK IS NOT RELEASED, ALL MIDAS PROCESSES WILL BE BLOCKED. TO RELEASE THE LOCK, MCLUP SHOULD BE EXECUTED. NOTE THAT A BLOCKED CONDITION MIGHT NOT BE IMMEDIATELY RECOGNIZED BY USERS. IF THIS CONDITION IS SUSPECTED, MCLUP MAY BE EXECUTED SIMPLY TO DETERMINE WHICH PROCESS HOLDS THE LOCK. MCLUP IS DESCRIBED IN MORE DETAIL IN SECTION 1.3.3.3.

1.3.3.2 MIDAS INITIALIZATION -- IMIDAS

```
**********
*        *
* IMIDAS *
*        *
**********
```

FUNCTION

INITIALIZES THE MIDAS SEMAPHORE AND SHARED LOCK.

DISCUSSION

1. IMIDAS MUST BE RUN AS PART OF THE COLD START SEQUENCE. IT IS THE RESPONSIBILITY OF THE PERSON INVOKING THIS UTILITY TO ENSURE THAT NO MIDAS APPLICATION PROGRAMS ARE EXECUTING WHEN THIS UTILITY IS RUN. COMMAND FILE C_MINIT MAY BE INSTALLED IN THE COLD START PROCEDURE TO SHARE THE SEGMENT CONTAINING THE LOCK AND TO EXECUTE IMIDAS.

2. IMIDAS HAS BEEN CODED AS A SUBROUTINE NAMED "MAIN" SO THAT IS CAN BE LOADED INTO SPLIT SEGMENT 4000. IMIDAS MAY THEN BE EXECUTED USING THE RESUME COMMAND.

3. COMMAND FILE C_IMIDAS IN UFD MIDAS>SOURCE MAY BE USED TO BUILD IMIDAS IN UFD MIDAS>CMDNCO.

4. IMIDAS MUST BE COMPILED WITH THE "-64V" AND "-BIG" FTN OPTIONS. DURING THE LOAD, THE COMMON BLOCK WITH THE NAME "LIST" MUST BE PLACED AT THE ADDRESS <0/1> WITH THE SEG COMMAND:

   SY LIST 0 1

1.3.3.3 MIDAS CLEANUP UTILITY -- MCLUP

```
**********
*        *
* MCLUP  *
*        *
**********
```

FUNCTION

AFTER ABNORMAL TERMINATION OF A MIDAS PROGRAM, MCLUP RE-INITIALIZES THE SHARED LOCK AND NOTIFIES THE SEMAPHORE TO AWAKEN ANY MIDAS PROCESS WAITING ON THE LOCK.

DISCUSSION

1. IF INVOKED WITH NO OPTIONS, MCLUP RE-INITIALIZES ONLY IF THE SHARED LOCK IS HELD BY THE TERMINAL USER, OTHERWISE MCLUP PRINTS THE USER NUMBER OF THE USER THAT HOLD THE LOCK.

2. IF INVOKED WITH AN OPTION OF THE FORM:

   -USER USERNUMBER

   THEN MCLUP WILL RE-INITIALIZE IF THE SHARED LOCK IS HELD BY THE SPECIFIED USER, OTHERWISE MCLUP PRINTS THE USER NUMBER OF THE USER THAT HOLDS THE LOCK.

3. MCLUP MAY BE BUILT IN UFD MIDAS>CMDNC0 BY COMMAND FILE C_MCLUP IN UFD MIDAS>SOURCE.

4. MCLUP MUST BE COMPILED WITH THE "-64V" AND "-BIG" FTN OPTIONS. DURING THE LOAD, THE COMMON BLOCK WITH THE NAME "LIST" MUST BE PLACED AT THE ADDRESS <0/1> WITH THE SEG COMMAND

   SY LIST C 1

2 RECOVERY FROM CONCURRENCY ERRORS

2.1 OVERVIEW

AT REV 17 MIDAS DETECTS AND CORRECTS MOST CONCURRENCY ERRORS. THESE ERRORS, ASSOCIATED WITH OPERATIONS INVOLVING THE CURRENT RECORD, OCCUR WHEN THE CURRENT INDEX ENTRY HAS BEEN DELETED OR PHYSICALLY MOVED SINCE THE TIME THE ENTRY BECAME CURRENT. IF MIDAS DISCOVERS THAT THE ENTRY HAS BEEN DELETED, THEN AN ERROR CODE OF 13 IS RETURNED. IN THE EVENT THAT THE ENTRY HAS BEEN MOVED, MIDAS AUTOMATICALLY LOCATES THE ENTRY AND CONTINUES NORMALLY.

2.2 IMPLEMENTATION OF CONCURRENCY ERROR DETECTION AND RECOVERY

AT THE FORTRAN CALL LEVEL INTERFACE, THE CONCEPT OF CURRENT RECORD AND CURRENT ENTRY IS IMPLEMENTED AS A FOURTEEN WORD COMMUNICATION ARRAY. THE COMMUNICATION ARRAY IS AN ARGUMENT IN MOST SUBROUTINE CALLS TO MIDAS. THE NEXT SECTION OUTLINES THE NEW COMMUNICATION ARRAY FORMAT FOR REV 17.

2.2.1 COMMUNICATION ARRAY FORMAT

WORD 1 (INPUT) IF -1 THEN MIDAS ARRAY CONTENTS ARE NOT USED.
(OUTPUT) ERROR STATUS

WORDS 2-4 CURRENT INDEX ENTRY ADDRESS
- WORD 2 BITS 1-8 -- ENTRY NUMBER
- WORD 2 BITS 9-16 -- SEGMENT FILE NUMBER
- WORDS 3 & 4 (32 BITS) -- WORD OFFSET OF INDEX BLOCK

WORD 5 HASH VALUE (BASED ON CURRENT KEY VALUE)

WORDS 6-9 CURPENT KEY VALUE (OR 1ST 4 WORDS OF KEY)

WORDS 10-12 CURRENT RECORD ADDRESS
- WORD 10 BIT 1 -- RECORD LOCKED FLAG
- WORD 10 BITS 9-16 -- SEGMENT FILE NUMBER
- WORDS 11 & 12 -- WORD OFFSET OF RECORD

WORD 13 DATA CONTROL WORD
- BITS 1-8 -- FLAG BITS
- BITS 9-16 -- PRIMARY KEY SIZE (BITS)

WORD 14 DATA RECORD LENGTH (WORDS)

NOTE THAT WORDS 2 THROUGH 9 OF THE COMMUNICATION ARRAY SPECIFY A CURRENT INDEX ENTRY AND WORDS 10 THROUGH 12 SPECIFY A CURRENT RECORD.

DURING OPERATIONS INVOLVING THE CURRENT ENTRY (EG. GET NEXT RECORD)

WORDS 2 THROUGH 4 ARE USED TO LOCATE THE EXPECTED POSITION OF THE ENTRY. TO VERIFY THAT THE POSITION CONTAINS THE CORRECT ENTRY, MIDAS COMPARES THE DATA POINTER IN THE ENTRY WITH THE DATA POINTER IN WORDS 10 THROUGH 12 OF THE COMMUNICATION ARRAY. IF THE POINTERS DON'T MATCH, THE THE ENTRY IS THE WRONG ONE.

EVEN IF THE POINTERS DO MATCH, MIDAS COMPARES THE KEY VALUE IN THE INDEX ENTRY TO THE KEY VALUE IN THE COMMUNICATION ARRAY. IF THEY DON'T MATCH, THEN THE ENTRY IS THE WRONG ONE. WHEN A WRONG ENTRY IS DETECTED, MIDAS SEARCHES FOR THE CORRECT ENTRY. IF NOT FOUND, MIDAS RETURNS AN ERROR CODE OF 13. NOTE THAT REV 16 VERSIONS EARLIER THAN REV 16.5 RETURNED AN ERROR CODE OF 13 WHEN A CONCURRENCY ERROR WAS DETECTED. USERS OF THESE EARLIER RELEASES MAY HAVE MODIFIED THEIR APPLICATIONS TO ATTEMPT TO RECOVER FROM AN ERROR 13. SINCE AT REV 17 AN ERROR 13 INDICATES THAT THE CURRENT INDEX ENTRY HAS BEEN DELETED, EXISTING APPLICATION ATTEMPTS TO HANDLE AN ERROR 13 MAY NEED MODIFICATION.

2.3 LIMITATIONS

FOR INDEXES WITH KEYS WHICH ARE LONGER THAN 8 BYTES, MIDAS MAY FAIL TO DETECT A CONCURRENCY ERROR. TO UNDERSTAND HOW THIS MAY OCCUR, NOTICE THAT IN THE COMMUNICATION ARRAY, AT MOST EIGHT BYTES OF A KEY MAY BE STORED. FOR KEYS LONGER THAN EIGHT BYTES, MIDAS STORES A HASH VALUE IN WORD 5 OF THE ARRAY. THE HASH VALUE IS BASED ON THE PORTION OF THE KEY BEYOND THE EIGHTH BYTE. NOW MIDAS WILL FAIL TO DETECT A CONCURRENCY ERROR ONLY IF:

A) THE DATA POINTERS MATCH,
B) THE KEY LONGER THAN 8 BYTES,
C) THE FIRST 8 BYTES OF THE KEY MATCH THE 8 BYTES STORED IN THE COMMUNICATION ARRAY, AND
D) THE HASH CODE, BASED ON THE REMAINING BYTES, IS THE SAME AS THE HASH CODE IN THE ARRAY.

3 INSTALLATION OF REV 17

3.1 BUILDING LIBRARIES AND UTILITIES

3.1.1 MIDAS UFD ORGANIZATION

AS SUPPLIED ON THE MASTER DISK, THE MIDAS UFD IS ORGANIZED INTO SEVERAL SUB-UFDS.

SOURCE -- CONTAINS ALL SOURCE AND COMMAND FILES.

LIB -- CONTAINS MIDAS LIBRARIES VKDALB, NVKDALB AND KIDALB.

SYSTEM -- CONTAINS FILES K4000, K2014A AND K2014B.

CMDNC0 -- CONTAINS MIDAS UTILITIES CREATK, KBUILD KIDDEL, IMIDAS, AND MCLUP.

3.1.2 COMMAND FILES

SEVERAL NEW COMMAND FILES HAVE BEEN ADDED.

C_MIDAS -- BUILDS MIDAS LIBRARIES AND UTILITIES.

C_VKDALB -- BUILDS THE SHARED V MODE LIBRARY, VKDALB. VKDALB IS PUT IN MIDAS>LIB. K4000, K2014A, AND K2014B ARE PLACED IN UFD MIDAS>SYSTEM.

C_NVKDALB -- BUILDS THE UNSHARED V MODE LIBRARY NVKDALB IN UFD MIDAS>LIB.

C_KIDALB -- BUILDS THE R MODE LIBRARY IN UFD MIDAS>LIB.

C_IMIDAS -- BUILDS UTILTY IMIDAS IN UFD MIDAS>SYSTEM.

C_MCLUP -- BUILDS UTILTY MCLUP IN UFD MIDAS>CMDNC0.

C_CREATK -- BUILDS CREATK IN MIDAS>CMDNC0.

C_KBUILD -- BUILDS KBUILD IN MIDAS>CMDNC0.

C_KIDDEL -- BUILDS KIDDEL IN MIDAS>CMDNC0.

3.2 INSTALLING THE SHARED LIBRARY

A. COPY VKDALB FROM MIDAS>LIB TO LIB,

B. COPY K4000, K2014A. AND K2014B FROM MIDAS>SYSTEM TO SYSTEM,

C. INSTALL COMMAND FILE C_MSHAR IN THE COLD START PROCEDURE.

3.3 MODIFYING THE SHARED LOCK AND SEMAPHORE VALUES

AS SUPPLIED AT REV 17.0, MIDAS USES SEMAPHORE NUMBER 64 AND WORD :177777 OF SEGMENT 2020 AS THE SHARED LOCK. THESE VALUES, DEFINED IN FILE KPARAM, MAY BE MODIFED BY USERS.

THE PARAMETERS ARE:

MSEMA1 -- SEMAPHORE NUMBER
SLSEG -- SEGMENT NUMBER OF THE SHARED LOCK
SLWORD -- WORD NUMBER OF THE SHARED LOCK

IF ANY OF THESE VALUES IS MODIFIED, THE USER MUST FOLLOW THE PROCEDURE DESCRIBED IN PARTS 2 AND 3 OF SECTION 3.4. MIDAS UTILITIES MCLUP AND IMIDAS MUST BE REBUILT AND INSTALLED. IN ADDITION, COMMAND FILE C_MINIT MUST BE MODIFIED SO THAT THE CORRECT SEGMENT GETS SHARED.

3.4 DISABLING THE REV 17 CONCURRENT PROCESS HANDLING

USERS MAY DISABLE THE REV 17 CONCURRENCY CONTROL METHOD AND THEREBY RETURN TO THE METHOD USED IN PREVIOUS RELEASES. NOTE THAT PROGRAMS WHICH USE NTFYM$, OPENM$, AND CLOSM$ WILL STILL WORK CORRECTLY.

PROCEDURE:

1) IN FILE KPARAM, CHANGE THE VALUE OF PARAMETER SHDSEG FROM .TRUE. TO .FALSE.,

2) FOR THE UNSHARED MIDAS LIBRARIES, KIDALB AND NVKDALB,

   A) COMPLILE SUBROUTINE LDPOOL. FOR V MODE LIBRARY NVKDALB USE FILE LONGPL. FOR THE R MODE LIBRARY KIDALB USE FILE LDPOOL.

   B) USE THE BINARY EDITOR, EDB, TO REPLACE THE OLD VERSION OF ROUTINE LDPOOL WITH THE NEW VERSION.

   C) RELOAD APPLICATION PROGRAMS WHICH USE THE UNSHARED LIBRARIES.

3) FOR THE SHARED V MODE LIBRARY VKDALB, REBUILD AND RE-INSTALL THE LIBRARY. APPLICATION PROGRAMS WHICH USE THE SHARED LIBRARY DO NOT NEED TO BE RE-LOADED.

3.5 NETWORK USERS

FOR NETWORK APPLICATIONS IN WHICH PROCESSES ACCESS REMOTE MIDAS FILES, THE REV 17 CONCURRENT PROCESS HANDLING METHOD MUST BE DISABLED TO PREVENT LOSS OF FILE INTEGRITY.

3.6 MIDAS FILE READ/WRITE LOCKS

WHEN REV 17 IS INSTALLED, THE READ/WRITE LOCK FOR EACH MIDAS FILE WHICH IS TO BE ACCESSED CONCURRENTLY, MUST BE SET TO 3. (N READERS AND M WRITERS)

3.7 RELOADING APPLICATION PROGRAMS

WHEN INSTALLING REV 17 MIDAS, ALL APPLICATION PROGRAMS WHICH USE AN UNSHARED MIDAS LIBRARY MUST BE RELOADED.

TABLE OF CONTENTS

```
**********************************************************************************************
**********************************************************************************************
**
**
**
**      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     SSS     000    RRRR    TTTTT
**   R   R   S   S   0   0   R   R     T
**   R   R   S       0   0   R   R     T
**   RRRR     SSS    0   0   RRRR      T
**   R R          S  0   0   R R       T
**   R  R    S   S   0   0   R  R      T
**   R   R    SSS     000    R   R     T
**
**
**
**********************************************************************************************
**********************************************************************************************
```

SORT COMMAND

DATE: MARCH 15, 1979

TO:

FROM:

SUBJECT: SORT COMMAND

REFERENCE: NONE

ABSTRACT

THE SORT COMMAND HAS SEVERAL ENHANCEMENTS FOR REV 17.

SORT NOW HANDLES THE FOLLOWING DATA TYPES AS SORT FIELDS (KEYS):

NUMERIC ASCII, LEADING SEPARATE SIGN
NUMERIC ASCII, TRAILING SEPARATE SIGN
NUMERIC ASCII, LEADING EMBEDDED SIGN
NUMERIC ASCII, TRAILING EMBEDDED SIGN
NUMERIC ASCII, UNSIGNED
PACKED DECIMAL.

TWO KEY TYPES FOR ASCII KEYS GIVE USERS THE OPTION OF HAVING LOWER CASE ALPHABETIC CHARACTERS SORT EQUAL TO UPPER CASE.

RECORD TYPE MAY BE EXPLICITLY SET, RATHER THAN DEFAULTING FROM THE SPECIFIED KEY TYPES. THIS GIVES FLEXIBILITY TO HANDLE NEW RECORD TYPES. FOR EXAMPLE, SORT WILL NOW HANDLE FIXED LENGTH RECORDS, AND BLANK COMPRESSED RECORDS WILL BE TREATED DIFFERENTLY THAN UNCOMPRESSED RECORDS.

THE MAXIMUM RECORD LENGTH MAY ALSO BE SPECIFIED BY THE USER.

MAXIMUM RECORD LENGTH IS NOW 32760 BYTES (CHARACTERS).

KEYS MAY BE AS LONG AS THE INPUT RECORDS.

THE MERGE OPTION IS NOW A TRUE MERGE RATHER THAN A SORT OF MULTIPLE INPUT FILES.

UP TO 20 FILES MAY BE SORTED INTO A SINGLE OUTPUT FILE.

THE USE OF KEYWORDS PROVIDES MORE FLEXIBILITY FOR SPECIFYING OPTIONAL INFORMATION.

****
* SORT *
****

****
* SORT *
****

PURPOSE:

THE SORT COMMAND IS USED TO SORT FROM ONE TO TWENTY FILES INTO A SINGLE OUTPUT FILE, OR TO MERGE FROM TWO TO ELEVEN PRESORTED FILES INTO A SINGLE FILE.

USAGE:

SORT IS INVOKED FROM PRIMOS COMMAND LEVEL. VARIOUS OPTIONS MAY BE SPECIFIED ON THE COMMAND LINE AS EXPLAINED BELOW. SORT REQUESTS A MINIMUM OF FILE AND KEY INFORMATION ON SUBSEQUENT LINES. ADDITIONAL INFORMATION, AS EXPLAINED BELOW, MAY BE SPECIFIED TO OVERRIDE DEFAULT RECORD LENGTH AND/OR RECORD TYPE.

INVOCATION:

SORT [-BRIEF] [-SPACE] [-MERGE]

THE MEANING OF THESE OPTIONS IS AS FOLLOWS:

OPTION	MEANING
BRIEF	SORT PROGRAM MESSAGES ARE NOT PRINTED AT THE USER'S TERMINAL.
SPACE	ANY BLANK LINES ARE DELETED FROM THE SORT OUTPUT FILE.
MERGE	A MERGE OF PRESORTED FILES IS REQUESTED.

FILE SPECIFICATION:

RESPOND TO THE FIRST INQUIRY WITH THE INPUT FILE NAME, OUTPUT FILE NAME, AND THE NUMBER OF PAIRS OF COLUMNS (DEFAULT IS 1). TO SPECIFY MULTIPLE INPUT FILES, RECORD TYPES, OR MAXIMUM RECORD LENGTHS THE FOLLOWING KEYWORDS MUST BE USED.

KEYWORD	USAGE
-INPUTFILE NAME	NAME IS AN INPUT FILE TREENAME UP TO 80 CHARACTERS LONG. THIS KEYWORD MAY BE USED REPEATEDLY, ONCE FOR EACH OF THE INPUT FILES.
-OUTPUTFILE NAME	NAME IS THE OUTPUT FILE TREENAME
-KEYS N	N IS THE NUMBER OF KEYS. DEFAULT IS 1.
-INTYPE TYPE	TYPE IS THE TYPE OF THE INPUT RECORDS: COMPRESSED UNCOMPRESSED FIXED OR VARIABLE. THESE RECORD TYPES ARE DESCRIBED BELOW.
-OUTTYPE TYPE	TYPE IS THE OUTPUT RECORD TYPE.
-INLENGTH N	N IS THE MAXIMUM LENGTH OF THE INPUT RECORDS. THIS LENGTH MUST BE SPECIFIED FOR FIXED LENGTH INPUT RECORDS.
-OUTLENGTH N	N IS THE MAXIMUM LENGTH OF THE OUTPUT RECORDS. THIS LENGTH MUST BE SPECIFIED FOR FIXED LENGTH OUTPUT RECORDS.

RECORD TYPES:

COMPRESSED SOURCE: BLANK COMPRESSED RECORD DELIMITED BY A NEWLINE CHARACTER (:212). SOURCE LINES CANNOT CONTAIN DATA WHICH MAY BE INTERPRETED AS A BLANK COMPRESSION INDICATOR (:221) OR A NEWLINE.

UNCOMPRESSED SOURCE: UNCOMPRESSED RECORD DELIMITED BY A NEWLINE CHARACTER (:212), AND THUS CANNOT CONTAIN DATA WHICH MAY BE INTERPRETED AS A NEWLINE.

VARIABLE LENGTH: RECORD STORED WITH LENGTH IN FIRST WORD.

FIXED LENGTH: RECORD CONTAINING DATA ONLY, NO LENGTH INFORMATION. THE LENGTH MUST BE SPECIFIED USING THE -INLENGTH OR -OUTLENGTH KEYWORDS. IF A NEWLINE CHARACTER IS APPENDED TO EACH RECORD SO THAT THE FILE CAN BE EDITED, IT MUST BE INCLUDED IN THE CHARACTER COUNT.

DEFAULT DEPENDS ON THE KEY TYPES SPECIFIED. INPUT TYPE DEFAULTS TO VARIABLE LENGTH IF ANY INTEGER, LONG INTEGER, SINGLE OR DOUBLE PRECISION REAL KEY IS SPECIFIED, OTHERWISE IT DEFAULTS TO ASCII. IF THE OUTPUT TYPE IS NOT GIVEN, IT IS ASSUMED TO BE THE SAME AS THE INPUT TYPE.

IF MULTIPLE INPUT FILES ARE USED, THEY MUST ALL CONTAIN THE SAME TYPE OF RECORDS.

KEYS SPECIFICATION:

RESPOND TO THE SUBSEQUENT INQUIRIES WITH THE APPROPRIATE STARTING AND ENDING COLUMN NUMBERS (CHARACTER POSITIONS). TO SORT IN DESCENDING ORDER, TYPE A SPACE AND THE LETTER R AFTER ENDING COLUMN OF THE FIELD TO BE REVERSE-SORTED. TO SPECIFY OTHER THAN ASCII KEYS, ENTER THE

PROPER CODE AT THE END OF THE LINE. ALTERNATIVELY, KEY INFORMATION MAY BE SPECIFIED BY USING CONTROL ARGUMENTS AS FOLLOWS:

CONTROL ARGUMENTS	USAGE
-START N	N IS STARTING COLUMN.
-END N	N IS ENDING COLUMN.
-DESCENDING	REQUESTS SORT IN DESCENDING ORDER.
-TYPE CODE	CODE CAN BE ONE OF:
	A - ASCII
	I - INTEGER
	J - LONG INTEGER
	F - SINGLE PRECISION REAL
	D - DOUBLE PRECISION REAL
	U - NUMERIC ASCII, UNSIGNED
	LS - LEADING SEPARATE
	TS - TRAILING SEPARATE
	LE - LEADING EMBEDDED
	TE - TRAILING EMBEDDED
	PD - PACKED DECIMAL
	AU - ASCII UPPERCASED WHEN COMPARED

WHEN THE LAST PAIR OF COLUMN NUMBERS IS ENTERED, SORTING BEGINS. WHEN THE SORT IS COMPLETE, SORT PRINTS THE NUMBER OF PASSES AND NUMBER OF ITEMS (LINES IN THE OUTPUT FILE), AND RETURNS TO PRIMOS.

MERGING:

AFTER ENTERING THE KEY INFORMATION, THE SORT PROGRAM ASKS FOR THE NUMBER OF ADDITIONAL MERGE FILES. THE USER THEN ENTERS THE MERGE FILENAMES, ONE PER LINE. WHEN THE LAST TREENAME IS ENTERED, MERGING BEGINS. WHEN THE MERGE IS COMPLETE, SORT PRINTS ONLY THE NUMBER OF PASSES BEFORE RETURNING TO PRIMOS.

EXAMPLES:

IN THE SAMPLE SORT RUNS BELOW, THE FOLLOWING INPUT FILES WILL BE USED.

```
OK, SLIST INPUT1
THIS
IS
A
SAMPLE
INPUT
FILE

OK, SLIST INPUT2
THIS            *
IS              *
A               *
SECOND          *
SAMPLE          *
INPUT           *
FILE            *

OK,
```

IN THE MERGE EXAMPLES, THE FOLLOWING PRESORTED COPIES OF THESE INPUT FILES WILL BE USED.

```
OK, SLIST MERGE1
A
FILE
INPUT
IS
SAMPLE
THIS

OK, SLIST MERGE2
A               *
FILE            *
INPUT           *
IS              *
SAMPLE          *
SECOND          *
THIS            *

OK,
```

THIS IS AN EXAMPLE OF SORTING A SINGLE INPUT FILE.

```
OK, SORT
SORT PROGRAM PARAMETERS ARE:
  INPUT TREE NAME -- OUTPUT TREE NAME FOLLOWED BY
  NUMBER OF PAIRS OF STARTING AND ENDING COLUMNS.
INPUT1_OUTPUT
  INPUT PAIRS OF STARTING AND ENDING COLUMNS
  ONE PAIR PER LINE--SEPARATED BY A SPACE.
  FOR REVERSE SORTING ENTER "R" AFTER DESIRED
  ENDING COLUMN--SEPARATED BY A SPACE.
  FOR A SPECIFIC DATA TYPE ENTER THE PROPER CODE
  AT THE END OF THE LINE--SEPARATED BY A SPACE.
     "A"  - ASCII
     "I"  - SINGLE PRECISION INTEGER
     "F"  - SINGLE PRECISION REAL
     "D"  - DOUBLE PRECISION REAL
     "J"  - DOUBLE PRECISION INTEGER
     "U"  - NUMERIC ASCII,UNSIGNED
     "LS" - NUMERIC ASCII,LEADING SEPARATE SIGN
     "TS" - NUMERIC ASCII,TRAILING SEPARATE SIGN
     "LE" - NUMERIC ASCII,LEADING EMBEDDED SIGN
     "TE" - NUMERIC ASCII,TRAILING EMBEDDED SIGN
     "PD" - PACKED DECIMAL
     "AL" - ASCII, UPPER & LOWER CASE SORT EQUAL
  DEFAULT IS ASCII.
1_5

BEGINNING SORT


 PASSES        2          ITEMS        6

[SORT-REV17.0]

OK, SLIST OUTPUT
A
FILE
INPUT
IS
SAMPLE
THIS

OK,
```

THIS IS AN EXAMPLE OF SORTING A SINGLE FILE IN DESCENDING ORDER AND THE USE OF THE BRIEF OPTION.

```
OK, SORT -BR
-INPUT INPUT1 -OUTPUT OUTFILE
-START 1 -END 5 -DESCENDING

BEGINNING SORT


 PASSES        2          ITEMS         6

[SORT-REV17.0]

OK, SLIST OUTFILE
THIS
SAMPLE
IS
INPUT
FILE
A

OK,
```

THIS IS AN EXAMPLE OF SORTING TWO FILES INTO A SINGLE OUTPUT FILE.
NOTE THAT THE ORDER OF INPUT IS MAINTAINED FOR RECORDS WITH EQUAL KEYS.

```
OK, SORT
SORT PROGRAM PARAMETERS ARE:
  INPUT TREE NAME -- OUTPUT TREE NAME FOLLOWED BY
  NUMBER OF PAIRS OF STARTING AND ENDING COLUMNS.
-INPUT INPUT1 -INPUT INPUT2 -OUTPUT OUTFILE
  INPUT PAIRS OF STARTING AND ENDING COLUMNS
  ONE PAIR PER LINE--SEPARATED BY A SPACE.
  FOR REVERSE SORTING ENTER "R" AFTER DESIRED
  ENDING COLUMN--SEPARATED BY A SPACE.
  FOR A SPECIFIC DATA TYPE ENTER THE PROPER CODE
  AT THE END OF THE LINE--SEPARATED BY A SPACE.
     "A"  - ASCII
     "I"  - SINGLE PRECISION INTEGER
     "F"  - SINGLE PRECISION REAL
     "D"  - DOUBLE PRECISION REAL
     "J"  - DOUBLE PRECISION INTEGER
     "U"  - NUMERIC ASCII,UNSIGNED
     "LS" - NUMERIC ASCII,LEADING SEPARATE SIGN
     "TS" - NUMERIC ASCII,TRAILING SEPARATE SIGN
     "LE" - NUMERIC ASCII,LEADING EMBEDDED SIGN
     "TE" - NUMERIC ASCII,TRAILING EMBEDDED SIGN
     "PD" - PACKED DECIMAL
     "AU" - ASCII, UPPER & LOWER CASE SORT EQUAL
  DEFAULT IS ASCII.
1 5 A

BEGINNING SORT


 PASSES        2          ITEMS        13

[SORT-REV17.0]

OK, SLIST OUTFILE
A
A               *
FILE
FILE            *
INPUT
INPUT           *
IS
IS              *
SAMPLE
SAMPLE          *
SECOND          *
THIS
THIS            *
```

THESE ARE TWO WAYS OF MERGING TWO PRESORTED FILES.
NOTE THAT THE ORDER OF INPUT IS MAINTAINED FOR RECORDS WITH EQUAL KEYS.

```
OK, SORT -MERGE
SORT PROGRAM PARAMETERS ARE:
  INPUT TREE NAME -- OUTPUT TREE NAME FOLLOWED BY
  NUMBER OF PAIRS OF STARTING AND ENDING COLUMNS.
-INPUT MERGE1 -INPUT MERGE2 -OUTPUT MERGEOUT -KEYS 1
  INPUT PAIRS OF STARTING AND ENDING COLUMNS
  ONE PAIR PER LINE--SEPARATED BY A SPACE.
  FOR REVERSE SORTING ENTER "R" AFTER DESIRED
  ENDING COLUMN--SEPARATED BY A SPACE.
  FOR A SPECIFIC DATA TYPE ENTER THE PROPER CODE
  AT THE END OF THE LINE--SEPARATED BY A SPACE.
     "A"  - ASCII
     "I"  - SINGLE PRECISION INTEGER
     "F"  - SINGLE PRECISION REAL
     "D"  - DOUBLE PRECISION REAL
     "J"  - DOUBLE PRECISION INTEGER
     "U"  - NUMERIC ASCII,UNSIGNED
     "LS" - NUMERIC ASCII,LEADING SEPARATE SIGN
     "TS" - NUMERIC ASCII,TRAILING SEPARATE SIGN
     "LE" - NUMERIC ASCII,LEADING EMBEDDED SIGN
     "TE" - NUMERIC ASCII,TRAILING EMBEDDED SIGN
     "PD" - PACKED DECIMAL
     "AU" - ASCII, UPPER & LOWER CASE SORT EQUAL
  DEFAULT IS ASCII.
1 5
INPUT THE NUMBER OF ADDITIONAL FILES TO BE MERGED. (MAX=     9): 0

BEGINNING MERGE


 PASSES       1
[SORT-REV17.0]

OK, SLIST MERGEOUT
A
A              *
FILE
FILE           *
INPUT
INPUT          *
IS
IS             *
SAMPLE
SAMPLE         *
SECOND         *
THIS
THIS           *
```

```
OK, SORT_-MERGE
SORT PROGRAM PARAMETERS ARE:
  INPUT TREE NAME -- OUTPUT TREE NAME FOLLOWED BY
  NUMBER OF PAIRS OF STARTING AND ENDING COLUMNS.
-OUTPUT_MERGEOUT
  INPUT PAIRS OF STARTING AND ENDING COLUMNS
  ONE PAIR PER LINE--SEPARATED BY A SPACE.
  FOR REVERSE SORTING ENTER "R" AFTER DESIRED
  ENDING COLUMN--SEPARATED BY A SPACE.
  FOR A SPECIFIC DATA TYPE ENTER THE PROPER CODE
  AT THE END OF THE LINE--SEPARATED BY A SPACE.
     "A"  - ASCII
     "I"  - SINGLE PRECISION INTEGER
     "F"  - SINGLE PRECISION REAL
     "D"  - DOUBLE PRECISION REAL
     "J"  - DOUBLE PRECISION INTEGER
     "U"  - NUMERIC ASCII,UNSIGNED
     "LS" - NUMERIC ASCII,LEADING SEPARATE SIGN
     "TS" - NUMERIC ASCII,TRAILING SEPARATE SIGN
     "LE" - NUMERIC ASCII,LEADING EMBEDDED SIGN
     "TE" - NUMERIC ASCII,TRAILING EMBEDDED SIGN
     "PD" - PACKED DECIMAL
     "AU" - ASCII, UPPER & LOWER CASE SORT EQUAL
  DEFALLT IS ASCII.
1_5
INPUT THE NUMBER OF ADDITIONAL FILES TO BE MERGED. (MAX=    11): 2
  INPUT FILES TO BE MERGED, ONLY ONE PER LINE.
MERGE1
MERGE2

BEGINNING MERGE



 PASSES       1
[SORT-REV17.0]

OK, SLIST MERGEOUT
A
A              *
FILE
FILE           *
INPUT
INPUT          *
IS
IS             *
SAMPLE
SAMPLE         *
SECOND         *
THIS
THIS           *
```

```
**********************************************************************************
**********************************************************************************
**
**
**
**      JJJ    III    M    M   M    M   Y    Y
**       J      I     MM  MM   MM  MM   Y    Y
**       J      I     M M M    M M M    Y Y
**       J      I     M M M    M M M     Y
**   J   J      I     M    M   M    M     Y
**   J   J      I     M    M   M    M     Y
**    JJ       III    M    M   M    M     Y
**
**
**
**     III    N    N  FFFFF    000                1     77777
**      I     NN   N  F       0   0              11         7
**      I     N N  N  F       0   0               1        7
**      I     N N  N  FFFF    0   0               1       7
**      I     N N  N  F       0   0               1      7
**      I     N  NN  F       0   0    ..         1     7
**     III    N    N  F        000     ..        111    7
**
**
**
**********************************************************************************
**********************************************************************************
```

BASICV FOR REV 17

DATE: MARCH 15, 1979

TO: R & D PERSONNEL

FROM: LARRY STABILE

SUBJECT: BASICV FOR REV 17

REFERENCE: NONE

ABSTRACT

IMPLEMENTED FOR REV 17 ARE THE FOLLOWING:

1. PERFORMANCE MEASUREMENT.
2. QUIT-HANDLING.
3. LOCAL VARIABLES ON USER-FUNCTIONS.
4. MISCELLANEOUS FUNCTIONALITY AND CHANGES.

1 QUIT-HANDLING

BASIC/VM NOW COMPLETELY SUPPORTS THE HANDLING OF QUITS (CONTROL-P'S) TYPED BY THE USER AT THE TERMINAL. DURING PROCESSING OF A COMMAND, A QUIT WILL RETURN THE USER TO BASIC COMMAND LEVEL. THIS IS USEFUL FOR ESCAPING FROM ACTIVITIES SUCH AS LONG PRINTOUTS, WITHOUT RESTARTING THE BASIC SESSION.

DURING PROGRAM EXECUTION, QUITS ARE HANDLED AS TRAPPABLE 'ERRORS', BY TWO NEW STATEMENTS:

ON QUIT GOTO STATEMENT-NUMBER

QUIT ERROR OFF

THESE TWO STATEMENTS ARE ANALOGOUS TO ON ERROR GOTO AND ERROR OFF, EXCEPT THAT ONLY QUITS ARE HANDLED BY THEM. IF QUIT ERROR OFF HAS BEEN EXECUTED, OR NO ON QUIT STATEMENT IS SET, THEN BASIC WILL RETURN TO COMMAND LEVEL WITH THE MESSAGE:

QUIT AT LINE STATEMENT-NUMBER

THE USUAL RULES OF ON ERROR GOTO REGARDING USER-DEFINED FUNCTIONS ALSO APPLY TO ON QUIT GOTO: TRANSFERS SHOULD ALWAYS BE MADE WITHIN THE FUNCTION DEFINITION.

A CAUTION TO USERS: BE CAREFUL OF WRITING YOURSELF INTO A LOOP SUCH THAT QUITS TRAP BACK TO THE SAME INFINITE LOOP FROM WHICH YOU WERE TRYING TO ESCAPE. THE ONLY WAY TO FREE YOUR TERMINAL THEN IS TO LOG YOURSELF OUT FROM ANOTHER TERMINAL VIA THE 'LOGOUT -USER_NUMBER' PRIMOS COMMAND.

EXAAMPLE

```
OK, BASICV

BASICV REV17.0
NEW OR OLD: OLD QUITTEST
>LIST
QUITTEST        THU, MAR 01 1979            11:00:10

10 ON QUIT GOTO 1000
20 PRINT 'HI!' WHILE 1
30 !
1000 PRINT 'QUIT!'
1005 QUIT ERROR OFF
1010 GOTO 20
1020 END
>RUN
QUITTEST        THU, MAR 01 1979            11:00:14

HI!
HI!
QUIT!  (A QUIT WAS TYPED HERE: THE ERROR TRAP WAS THEN TURNED OFF)
HI!
HI!    (A SECOND QUIT WAS TYPED JUST AFTER THIS POINT)
QUIT AT LINE 20
>QUIT

OK,
```

2 PERFORMANCE MEASUREMENT

THE PERFORMANCE MEASUREMENT PACKAGE IS FULLY DOCUMENTED IN INFO.PERF.17.

3 LOCAL VARIABLES

LOCAL VARIABLES IN USER-DEFINED FUNCTIONS ARE A MAJOR STEP REQUIRED FOR ANY FORM OF MODULAR PROGRAMMING IN BASIC. REV 17 BASICV NOW SUPPORTS LOCAL, STATIC VARIABLES AND ARRAYS IN A USER-FUNCTION DEFINITION. VARIABLES NOT DECLARED LOCAL ARE ALWAYS GLOBAL, EXCEPT FOR THE ARGUMENTS TO THE FUNCTION.

THE FORMAT IS EXEMPLIFIED BY:

```
DEF FNP$(X$,Y$,N) ! PADS X$ ON RIGHT WITH CHARACTER Y$
                  ! SUCH THAT TOTAL LENGTH IS N.
  LOCAL Z$
  Z$ = X$
  Z$ = Z$ + Y$ UNTIL LEN(Z$) = N
  FNP$ = Z$
FNEND
```

THE LOCAL STATEMENT ALWAYS APPEARS IMMEDIATELY FOLLOWING A DEF, AND INDICATES A SET OF NAMES TO BE TAKEN AS LOCAL TO THE FUNCTION DEFINED. THESE VARIABLES HAVE SIMILAR STATUS AS ARGUMENTS TO THE FUNCTION IN THAT THEY CANNOT BE PRINTED IN IMMEDIATE MODE DURING A PAUSE OR BREAK.

ARRAYS MAY ALSO BE DECLARED LOCAL, FOR EXAMPLE:

```
5 ! RETURNS DETERMINANT OF MATRIX FORMED USING VALUE OF X
10 DEF FNA(X)
20  LOCAL I,J
30  LOCAL DIM A(10,10)
40  A(I,J) = SIN(X*I*J) FOR I = 1 TO 10 FOR J = 1 TO 10
50  FNA = DET(A)
60 FNEND
```

NOTE THAT LOCAL VARIABLES AND ARRAYS ARE STATIC, THAT IS, THEIR VALUES ARE PRESERVED OVER MULTIPLE CALLS TO THE FUNCTION.

REFERRING TO THE ABOVE EXAMPLES, NOTE THAT USERS CAN KEEP STANDARD LIBRARIES OF FUNCTIONS WHICH NEED ONLY BE RESEQUENCED TO BE USED IN THEIR PROGRAMS. THE LOCAL STATEMENT HAS EFFECTIVELY REMOVED THE NEED TO SEARCH FOR VARIABLE-NAME CONFLICTS.

4 MISCELLANEOUS FUNCTIONAL ENHANCEMENTS AND CHANGES

A. MULTIPLE ASSIGNMENT A,B,C=<EXPR>

THIS FEATURE MAY ALSO BE REFERRED TO AS 'SIMULTANEOUS ASSIGNMENT', SINCE THE 'OLD' VALUES OF A VARIABLE ARE USED IN MAKING A CALCULATION FOR A TARGET VALUE. ARRAY-SUBSCRIPT CALCULATIONS ARE THE REASON THAT CLARIFICATION IS NEEDED:

```
I=5

I,A(I) = 10
```

ABOVE, A(5) WILL BE SET TO 10, NOT A(10); I IS NOT SET TO 10 BEFORE THE ARRAY-SUBSCRIPT CALCULATION IS PEFORMED.

B. DOUBLE QUOTE

DOUBLE-QUOTE (") HAS BEEN ADDED AS A LEGAL STRING-CONSTANT DELIMITER. NOTE THAT NOW BOTH " AND ' ARE PERMITTED, SO THAT EACH MAY BE USED AS PART OF THE STRING LITERAL.

C. BINARY FILES

LINE NUMBERS HAVE BEEN ADDED TO THE BINARY FILE, SO THAT PROGRAMS USING ERL WILL STILL WORK CORRECTLY WHEN COMPILED AND SAVED.

H. RANDOMIZE

RANDOMIZE STATEMENT. HAS SAME EFFECT AS THE RANDOM-NUMBER SEEDING THAT OCCURS UPON STARTING A BASIC PROGRAM. ALSO, RND MAY NOW BE USED SYNONOMOUSLY WITH RND(0), FOR COMPATIBILITY WITH MOST OTHER BASICS.

I. INVOLUTION *** CHANGE IN FUNCTIONALITY ***

THE ORDER OF PRIORITY IN EVALUATING INVOLUTION (A^B) HAS BEEN CHANGED TO BE ANSI-CONFORMING. NOW, A^B^C = (A^B)^C AS OPPOSED TO A^(B^C).

J. SIMPLE CURSOR CONTROL

@ IS NOW EQUIVALENT TO FNZ9$. THIS IS SO ONE MAY WRITE A CURSOR-POSITIONING FUNCTION OF THE FORM PRINT @(12,4);'ABCDE' . A TYPICAL FUNCTION FOR DOING THIS IS SHOWN BELOW.

```
100 ! THIS FUNCTION SERVES AS A CURSOR-POSITIONER FOR
101 !  THE FOX TERMINAL.
110 !     NOTE THAT IT IS WISE TO MARGIN OFF WHEN USING
111 !          THIS FUNCTION.
120 !
130 DEF @(X,Y)=CHAR(155)+'X'+CHAR(32+X)+CHAR(155)+'Y'+

     CHAR(32+Y)
140 !
150 PRINT @(12,30):'ABCDE'
```

K. INPUT QUOTES

QUOTES AS INPUT FOR INPUT VARIABLES. STRINGS GIVEN IN QUOTES FOR AN INPUT, INPUTLINE, OR MATINPUT STATEMENT WILL BE INPUT AS THE LITERAL STRING TYPED BUT WITH THE QUOTES REMOVED. ONE CAN THEN INPUT, FOR INSTANCE, LEADING AND TRAILING BLANKS.

L. ON.... GOTO [ELSE GOTO ...]

CHANGE TO ON...GOTO SN1,...,SNN . ONE CAN NOW USE A STATEMENT OF THE FORM ON...GOTO SN1,...,SNN ELSE [GOTO] SN . IF THE SELECTOR EXPRESSION IS OUT-OF-RANGE, THE ELSE LINE-NUMBER IS GIVEN CONTROL.

M. CVT$$

CVT$$ FUNCTION. UPPER-TO-LOWER CASE CONVERSION HAS BEEN ADDED TO COMPLEMENT THE LOWER-TO-UPPER CASE CONVERSION. ITS CODE IS 256. NOTE THAT IF BOTH THIS AND 32 (LOWER-TO-UPPER) ARE USED, THEN BOTH CONVERSIONS WILL BE PERFORMED, SO THAT:

IF X$ = 'ABCDE', THEN

CVT$$(X$,32) RETURNS 'ABCDE'

CVT$$(X$,256) RETURNS 'ABCDE'

CVT$$(X$,256+32) RETURNS 'ABCDE'

N. LIN#(UNIT) FUNCTION *** CHANGE IN FUNCTIONALITY ***

THE LIN# FUNCTION, WHICH RETURNS THE CURRENT POSITION IN A DIRECT-ACCESS FILE, HAS BEEN CHANGED TO RETURN ONE LESS THEN BEFORE. SO, WHEN A FILE IS OPENED, LIN# STARTS AT 0 (TAR 20150).

```
**********************************************************************
**********************************************************************
**
**
**
**      JJJ    III    M    M   M    M   Y     Y
**       J      I     MM  MM   MM  MM   Y     Y
**       J      I     M M M    M M M     Y Y
**       J      I     M  M     M  M      Y
**    J  J      I     M        M        Y
**    J  J      I     M        M        Y
**     JJ       III   M        M        Y
**
**
**
**    TTTTT   AAA    RRRR      1     77777          1
**      T    A   A   R   R    11         7         11
**      T    A   A   R   R     1        7          1
**      T    AAAAA   RRRR      1       7           1
**      T    A   A   R R       1      7            1
**      T    A   A   R  R      1     7       ..     1
**      T    A   A   R   R    111    7       ..    111
**
**
**
**********************************************************************
**********************************************************************
```

TARS FIXED FOR REV17.1

THE FOLLOWING BUG-FIXES FROM REV16.6 HAVE BEEN CARRIED OVER TO REV17.1:

TAR NO.	DESCRIPTION
14219	SELECT WITH ALTERNATE KEY, NOT DEFINED; GAVE A BAD ERROR MESSAGE.
14594	COMPILER DID NOT FLAG AS AN ERROR A DATA-NAME THAT WAS NOT DEFINED AS LEVEL 01/77 ON A USING STATEMENT.
21498	MIDAS 13 / COBOL 94 STATUS CODES ALONG WITH LEAVING RECORDS LOCKED WAS FIXED.
22304	COMPILER FAILED TO FLAG INDEX-NAMES THAT WERE THE SAME AS DATA-NAMES.
22307	BAD ERROR MESSAGE WAS ISSUED WHEN NO CORRESPONDENCE WAS FOUND.
22308	SOME COMPILER ERROR MESSAGES HAD SPACES IN THE MIDDLE OF A WORD.
22310	COMPILER WAS ABORTING WITH A TABLE GROUP ERROR, IF A LEVEL 88 STATMENT APPEARED DIRECTLY BEFORE THE LINKAGE SECTION.
22318	MISLEADING ERROR MESSAGE FROM COMPILER.
23360	MOVE CORRESPONDING WAS CAUSING THE FOLLOWING STATEMENT TO BE SKIPPED.
23677	COMPILER GENERATES FAULTY CODE TO EVALUATE A SUBSCRIPTED NUMERIC CONDITION NAME.
23679	THE GENERATED CODE CONVERTED INDEX ITEMS TO COMPUTATIONAL-3, INSTEAD OF EXTERNAL DECIMAL.
23683	DOCUMENTATION PROBLEM, THE MANUAL DID NOT STATE THAT THE SECONDARY KEY HAD TO BE CONTAINED WITHIN THE RECORD DESCRIPTION. THE COMPILER WILL NOW FLAG THIS CONDITION.
23684	THE COMPILER GENERATED BAD BRANCH WHEN TESTING NEGATED CONDITION WITH A COMPUTATIONAL ITEM.
80984	FIXED UNDER TAR 24806

SPECIAL INSTRUCTIONS:

THERE IS A BUG IN REV.1 COBOL INVOLVING THE FIGURATIVE CONSTANT ZERO IN A CONDITIONAL EXPRESSION. IF ZERO IS THE SUBJECT OF A RELATION CONDITION, THE PROGRAM WILL NOT COMPILE CORRECTLY. THE PROBLEM CAN BE TEMPORARILY CIRCUMVENTED BY REWRITING THE CONDITIONAL EXPRESSION SUCH THAT THE FIGURATIVE CONSTANT ZERO IS THE OBJECT OF THE RELATION CONDITION.

USING THE GENERAL FORMAT,

```
IF  ZERO  ;RELATIONAL OPERATOR= ;IDENTIFIER=
```

SHOULD BE CHANGED TO

```
IF ;IDENTIFIER= ;RELATIONAL OPERATOR= ZERO
                  (REVERSED WHEN
                       NECESSARY)
```

WE ARE CURRENTLY WORKING ON FIXING THIS BUG AND HOPE TO RELEASE THE FIX FOR THE NEXT RELEASE.

```
**********************************************************************************
**********************************************************************************
*
*
*
*      JJJ    III    M    M   M    M   Y    Y
*       J      I     MM  MM   MM  MM   Y    Y
*       J      I     M M M    M M M     Y Y
*       J      I     M M M    M M M      Y
*    J  J      I     M   M    M   M      Y
*    J  J      I     M   M    M   M      Y
*     JJ       III   M   M    M   M      Y
*
*
*
*    RRRR    PPPP    L        PPPP
*    R   R   P   P   L        P   P
*    R   R   P   P   L        P   P
*    RRRR    PPPP    L        PPPP
*    R R     P       L        P
*    R  R    P       L   L    P
*    R   R   P       LLLLL    P
*
*
*
**********************************************************************************
**********************************************************************************
```

SUBJECT: PL/P SPECIFICATION

INTRODUCTION

PL/P IS A VARIANT OF ANSI PL/I: ALTHOUGH A SUBSET IN MOST RESPECTS, IT CONTAINS SEVERAL SIGNIFICANT EXTENSIONS TO THE STANDARD LANGUAGE. SINCE THIS DOCUMENT DESCRIBES ONLY THE DIFFERENCES BETWEEN PL/P AND ANSI PL/I, READERS NOT FAMILIAR WITH THE FULL LANGUAGE SHOULD CONSULT ONE OF THE REFERENCES IN THE BIBLIOGRAPHY BEFORE PROCEEDING WITH THIS DOCUMENT. (APPENDIX A, "PL/P COURSE SYLLABUS," IS ALSO ATTACHED FOR THE READER'S CONVENIENCE, ALTHOUGH IT IS INTENDED TO BE USED IN CONNECTION WITH A COURSE WHEREIN THE TEACHER WOULD EXPAND UPON ITS CONTENTS WITH FULLER EXPLANATIONS AND EXAMPLES.)

ANSI FEATURES MISSING IN PL/P

THE FOLLOWING FEATURES OF ANSI PL/I ARE NOT INCLUDED IN PL/P:

1. ALL FORMS OF INPUT AND OUTPUT
2. THE CONDITION MECHANISM
3. THE ALLOCATE AND FREE STATEMENTS
4. THE STOP STATEMENT
5. THE DEFAULT STATEMENT
6. ALL ATTRIBUTES EXCEPT INTERNAL, EXTERNAL, STATIC, AUTOMATIC, BASED, ALIGNED, UNALIGNED, BIT, CHARACTER, VARYING, NONVARYING, ENTRY, RETURNS, LABEL, POINTER, BINARY, FIXED, LIKE, INITIAL, CONSTANT, VARIABLE, AND OPTIONS AND (IMPLICIT ONLY) DIMENSION, MEMBER, PRECISION, PARAMETER, REAL, AND STRUCTURE
7. ALL BUILTIN FUNCTIONS AND PSEUDOVARIABLES EXCEPT BINARY, CHARACTER, MOD, DIVIDE, AFTER, BEFORE, COPY, DATE, INDEX, LENGTH, REVERSE, SUBSTR, TIME, TRANSLATE, VERIFY, NULL, AND ADDR
8. AGGREGATE EXPRESSIONS AND PROMOTION, EXCEPT PROMOTION FROM SCALAR TO ARRAY IN SIMPLE ASSIGNMENT STATEMENTS
9. IMPLICIT DECLARATION OF USER-DEFINED NAMES
10. IMPLICIT CONVERSION, EXCEPT OF PRECISION OR SIZE, BETWEEN VARYING AND NONVARYING CHARACTER DATA, AND BETWEEN BIT AND BINARY WITH PRECISION <= 15

11. CONDITION PREFIXES

12. VARIABLE EXTENTS EXCEPT THE "*" STRING SIZE IN A CHARACTER PARAMETER DESCRIPTION IN THE DECLARATION OF AN EXTERNAL ENTRY

13. SUBSCRIPTED LABEL CONSTANTS

14. END STATEMENT CLOSURE LABELS

15. SCALED AND IMAGINARY ARITHMETIC CONSTANTS AND THE DEFAULT-SUPPRESSING CHARACTER P

16. THE %INCLUDE STATEMENT

17. OPTIONS ON THE BEGIN STATEMENT

18. UNCONNECTED ARRAY REFERENCES

19. MULTIPLE-TARGET ASSIGNMENT STATEMENTS

20. BY NAME ASSIGNMENT

21. BIT VARYING

22. THE REFER OPTION

23. THE ** AND / OPERATORS

IN ADDITION, THE FOLLOWING RESTRICTIONS APPLY:

1. NO REFERENCE MAY CONTAIN MORE THAN ONE PARENTHESIZED LIST, EXCEPT STRUCTURE REFERENCES IN WHICH THE SUBSCRIPTS ARE DISTRIBUTED OVER THE COMPONENTS.

2. DO INDICES MUST BE EITHER BINARY OR POINTER.

3. THE INITIAL ATTRIBUTE MAY ONLY CONTAIN STRING OR ARITHMETIC CONSTANTS, THE BUILTIN FUNCTION NULL(), OR "*".

4. ITERATION FACTORS IN THE INITIAL ATTRIBUTE MUST BE INTEGER CONSTANTS.

5. ONLY ITEMS OF STORAGE CLASS STATIC MAY BE INITIALIZED.

6. THE BASED ATTRIBUTE MAY NOT CONTAIN A POINTER REFERENCE; HENCE, ALL REFERENCES TO BASED VARIABLES MUST BE POINTER-QUALIFIED.

7. BITSTRINGS ARE ALIGNED SO THAT THEY WILL NOT CROSS WORD BOUNDARIES, AND MEMBERS OF BITSTRING ARRAYS ARE WORD-ALIGNED, REGARDLESS OF ALIGNMENT ATTRIBUTE.

8. BUILTIN FUNCTIONS WHICH DO NOT TAKE ARGUMENTS MUST BE CODED WITH AN EMPTY ARGUMENT LIST.

9. THE DATA TYPE MUST BE GIVEN FOR EACH PARAMETER IN AN ENTRY ATTRIBUTE AND FOR THE RETURN VALUE IN THE RETURNS OPTION OF THE PROCEDURE STATEMENT AND THE RETURNS ATTRIBUTE.

10. THE DECLARATION OF THE REFERENCE IN THE LIKE ATTRIBUTE MUST PHYSICALLY PRECEDE THE LIKE REFERENCE IN THE PROGRAM, AND MUST NOT CONTAIN A LIKE ATTRIBUTE.

11. THE FIRST ARGUMENT TO THE BIT BUILTIN FUNCTION MUST BE EITHER AN INTEGER CONSTANT OR A BINARY VARIABLE OF PRECISION <= 15.

12. SCALE FACTORS ARE NOT ALLOWED IN THE BINARY ATTRIBUTE.

13. THE FIRST ARGUMENT TO THE CHARACTER BUILTIN FUNCTION MUST BE EITHER AN INTEGER CONSTANT OR A BINARY VARIABLE.

14. THE SECOND ARGUMENT TO THE COPY BUILTIN FUNCTION MUST BE A CONSTANT.

15. THE THIRD ARGUMENT TO THE DIVIDE BUILTIN FUNCTION MUST BE PRESENT AND BE AN INTEGER CONSTANT AND THE FOURTH ARGUMENT MUST BE OMITTED.

16. THE LENGTH OF A SUBSTR BUILTIN FUNCTION OR PSEUDOVARIABLE WHOSE FIRST ARGUMENT IS A BITSTRING MUST BE CALCULATABLE AT COMPILE TIME.

17. CHARACTER STRING CONSTANTS MAY NOT CONTAIN THE NEWLINE CHARACTER.

18. UNALIGNED BITSTRINGS MAY BE PASSED AS ARGUMENTS ONLY VIA CALL-BY-VALUE.

19. FOR AGGREGATE PARAMETERS AND ARGUMENTS, ARRAY SIZE AND DIMENSIONALITY AND STRUCTURE SHAPE ARE NOT CONSIDERED IN ARGUMENT VALIDATION.

IMPLEMENTATION-SPECIFIC FEATURES

1. ALL PROCEDURES ARE IMPLICITLY RECURSIVE, EXCEPT THOSE INTERNAL PROCEDURES CODED WITH THE SHORTCALL OPTION.

2. THE MAXIMUM SIZE OF CHARACTER STRINGS IS 8192 CHARACTERS.

3. THE MAXIMUM SIZE OF BIT STRINGS IS 16 BITS.

4. THE UNALIGNED ATTRIBUTE IMPLIES CHARACTER ALIGNMENT FOR CHARACTER NONVARYING DATA, BIT ALIGNMENT FOR NON-ARRAY BITSTRING DATA, AND WORD ALIGNMENT FOR ALL OTHER CASES.

5. THE ALIGNED ATTRIBUTE IMPLIES WORD ALIGNMENT IN ALL CASES.

6. THE MAXIMUM LENGTH OF AN IDENTIFIER NAME IS 32 CHARACTERS.

7. THE MAXIMUM LENGTH OF A SOURCE LINE IS 256 CHARACTERS.

8. POINTER VARIABLES USE TWO WORDS (32 BITS) OF MEMORY AND HENCE MAY ONLY POINT TO WORD-ALIGNED DATA.

9. OPTIONS(GATE) MAY BE SPECIFIED ON THE EXTERNAL PROCEDURE STATEMENT, CAUSING RING WEAKENING TO OCCUR FOR ALL PARAMETER, BASED, AND EXTERNAL POINTERS UPON ASSIGNMENT.

10. OPTIONS(SAVE(REF)) MAY BE SPECIFIED ON A PROCEDURE STATEMENT TO CAUSE AN RSAV INTO THE VARIABLE "REF" TO BE GENERATED BEFORE ANY OTHER CODE IN THE PROCEDURE.

11. OPTIONS(SHORTCALL) MAY BE SPECIFIED ON AN INTERNAL PROCEDURE STATEMENT TO CAUSE IT TO BE ACCESSED BY JSY INSTEAD OF PCL; THE COMPILER DIAGNOSES INCORRECT USE OF THIS FEATURE.

12. OPTIONS(SHORTCALL[(INTEGER_CONSTANT)]) MAY BE SPECIFIED IN A DECLARATION WITH THE ENTRY ATTRIBUTE, SPECIFYING THAT THE EXTERNAL ROUTINE IS TO ACCESSED BY JSXB INSTEAD OF PCL. SEE APPENDIX A FOR MORE DETAILS.

13. EXTERNAL REFERENCES ARE RESOLVED ON THE BASIS OF THE FIRST EIGHT CHARACTERS OF THE NAME.

14. POINTER, BINARY, AND BIT RETURN VALUES ARE PASSED IN THE A OR L REGISTER IN ORDER TO CONFORM TO THE FORTRAN CONVENTION.

15. LABEL VARIABLES ARE STORED WITH THE FIRST TWO WORDS (I.E., THE POINTER TO THE EXECUTION ADDRESS) INTERCHANGED FOR COMPATIBILITY WITH FORTRAN ROUTINES EXPECTING AN ALTERNATE-RETURN ARGUMENT.

16. ARGUMENTS PASSED TO A CHARACTER(*)NONVARYING PARAMETER ARE ACTUALLY PASSED AS TWO ARGUMENTS, THE FIRST BEING THE STRING CONTENTS AND THE SECOND THE STRING LENGTH. THIS CONFORMS WITH A POPULAR FORTRAN CALLING SEQUENCE. (NO ADDITIONAL INFORMATION IS PASSED TO A CHARACTER(*)VARYING PARAMETER, SINCE THE STRING IS SELF-DEFINING WITH RESPECT TO LENGTH.)

NON-STANDARD EXTENSIONS IN PL/P

THE FOLLOWING EXTENSIONS TO ANSI PL/I ARE AVAILABLE IN PL/P (FOR MORE INFORMATION REGARDING THE FUNCTIONS OF THESE FEATURES, SEE APPENDIX A):

1. UPPERCASE AND LOWERCASE ARE COMPLETELY INTERCHANGEABLE, EXCEPT WITHIN CHARACTER STRING CONSTANTS.

2. "$INSERT TREENAME", IF IT BEGINS IN COLUMN 1, IS NOT FOLLOWED BY ANY OTHER TEXT ON THE LINE, AND THE "I" IS CAPITALIZED WILL BE LOGICALLY REPLACED IN THE COMPILER INPUT BY THE CONTENTS OF THE FILE REFERENCED BY "TREENAME".

3. THE "$" CHARACTER IS LEGAL IN IDENTIFIERS, EXCEPT THAT IT MAY NOT BE THE FIRST CHARACTER OF THE IDENTIFIER NAME.

4. ARGUMENT-PARAMETER TYPE AND NUMBER CHECKING IS DISABLED BY DECLARING THE ENTRY NAME WITH NO PARAMETER LIST.

5. THE SELECT STATEMENT

6. THE LEAVE STATEMENT

7. THE %NOLIST AND %LIST STATEMENTS

8. THE %REPLACE STATEMENT

9. THE UNTIL OPTION OF THE DO STATEMENT

10. THE CALL STATEMENT HAS BEEN EXTENDED TO RECOGNIZE THE FOLLOWING "BUILTIN SUBROUTINES": INHIBIT, ENABLE, WAIT, NOTIFYB, NOTIFYE.

11. THE SEARCH, LINKPTR, STACKPTR, BASEPTR, ADDREL, PTR (NONSTANDARD DEFINITION), REL, RING, SEGNO, BASEREL, STACKBASE, CSTORE, REGFILE, ADDQT, ADDQB, REMQT, REMQB, TESTQ, AND TRIM BUILTIN FUNCTIONS

12. THE REGFILE AND REGISTERS PSEUDOVARIABLES

BIBLIOGRAPHY

1. THE PL/I PROGRAMMING LANGUAGE, PAUL ABRAHAMS--C00-3077-151, COURANT MATHEMATICS AND COMPUTING LABORATORY, NEW YORK UNIVERSITY, MARCH 1978

2. MULTICS PL/I REFERENCE MANUAL--AM83 REV. 1, HONEYWELL INFORMATION SYSTEMS, JUNE 1976

APPENDIX A

PL/P COURSE SYLLABUS

I. OVERVIEW OF DIFFERENCES BETWEEN FORTRAN AND PL/I

A. NO DISTINCTION MADE BETWEEN UPPER AND LOWER CASE (PL/P ONLY)

B. FREER LINE STRUCTURE

1. NO COLUMN DEPENDENCIES

A. UP TO 256 COLUMNS (PL/P)

B. NO CONTINUATION COLUMN--STATEMENTS ARE COMPLETELY LINE-INDEPENDENT

C. NO COMMENT COLUMN--ALL COMMENTS BEGIN WITH "/*" AND MUST END WITH "*/"--MAY BEGIN ANYWHERE AND RUN FOR ANY NUMBER OF LINES

D. NO LABEL FIELD--LABELS ARE IDENTIFIERS, NOT NUMBERS, AND MAY BE IN ANY COLUMN; FOLLOWED BY ":" TO INDICATE LABEL

2. SEMICOLON REQUIRED TO END STATEMENTS--ALLOWS LINE INDEPENDENCE

3. SPACES ARE REQUIRED TO SEPARATE IDENTIFIERS (KEYWORDS, VARIABLES, LABELS, ETC.) AND NUMBERS FROM THEMSELVES AND EACH OTHER WHEN NOT SEPARATED BY OTHER NON-ALPHANUMERIC CHARACTERS; MAY BE USED FREELY ANYWHERE EXCEPT WITHIN LEXICAL ITEMS.

4. BLANK LINES ALLOWED

C. ARBITRARY COMPLEXITY OF EXPRESSIONS, EVEN IN SUBSCRIPTS

D. CALL BY VALUE INSTEAD OF CALL BY REFERENCE FOR CONSTANTS AND USER-SELECTED VARIABLES

E. IDENTIFIER NAMES ARE MORE FLEXIBLE

F. BLOCK-STRUCTURED--ALLOWS INTERNAL SUBROUTINES

* G. MULTIPLE ENTRY POINTS, WITH SAME OR DIFFERENT CALLING SEQUENCES

H. STORAGE CLASSES AVAILABLE ON A PER-VARIABLE BASIS FOR THINGS LIKE -DYNM, COMMON, AND RUNTIME STORAGE MANAGEMENT (NO PARALLEL IN FORTRAN)

I. EXTENDED DO-LOOP FUNCTIONALITY

1. LOGICAL CONDITIONS AS WELL AS ITERATIVE COUNTS

2. DOES NOT EXECUTE ONCE UNCONDITIONALLY

3. ABLE TO COUNT BACKWARDS

4. MULTIPLE SPECIFICATIONS FOR LOOPS

5. ABLE TO USE NON-INTEGER DATA FOR INDEX

J. IF-THEN-ELSE; NESTABLE; USABLE WITH COMPOUND STATEMENTS

K. WIDE RANGE OF DATA TYPES

1. MORE FLEXIBLE ARITHMETIC TYPES

2. ADDITIONAL TYPES FOR BITS, FIXED- AND VARYING-LENGTH CHARACTER STRINGS, POINTERS, LABELS, ETC.

L. ABLE TO GROUP DATA ITEMS LOGICALLY IN STORAGE, EVEN IF DIFFERENT DATA TYPES

M. ABLE TO COUNT CHARACTERS IN CALLS FOR THE PROGRAMMER

N. ABLE TO VALIDATE ARGUMENTS IN CALLS

O. ARRAYS ARE STORED IN ROW-MAJOR ORDER: RIGHTMOST SUBSCRIPT VARIES MOST RAPIDLY

P. KEYWORDS ARE NOT RESERVED AND MAY BE USED AS IDENTIFIERS

Q. DYNAMIC HANDLING OF ERRORS AND OTHER EXCEPTIONS (NOT IN PL/P)

R. FLEXIBLE I/O (NOT IN PL/P)

II. IDENTIFIERS

A. NAMING RESTRICTIONS

1. MUST BEGIN WITH AN ALPHABETIC CHARACTER; MAY CONTAIN DIGITS, "$", AND "_" ("_" IS GOOD FOR CLARITY BY SEPARATING WORDS IN A NAME)

2. INTERNAL NAMES MAY BE UP TO 32 CHARACTERS LONG; EXTERNAL NAMES MAY BE UP TO 8 CHARACTERS (PL/P AND SEG RESTRICTIONS)

B. EXCEPT FOR LABEL CONSTANTS AND BUILTIN FUNCTIONS (SEE BELOW), MUST BE EXPLICITLY DECLARED (PL/P RESTRICTION, BUT GOOD PROGRAMMING PRACTICE)

C. THE DECLARE STATEMENT

1. REPLACES THE TYPE STATEMENTS (REAL, LOGICAL, ETC.) OF FORTRAN; THE DATA TYPE OF AN IDENTIFIER IS GIVEN BY KEYWORDS FOLLOWING IT, RATHER THAN BY DIFFERENT TYPE

"*" => PLANNED BUT NOT YET AVAILABLE IN PL/P

STATEMENTS

2. MAY APPEAR ANYWHERE IN THE PROCEDURE, SUBJECT TO SCOPE RULES (SEE BELOW)

3. SYNTAX: DCL <VAR_SPEC>[, <VAR_SPEC>[, ...]];

A. TYPICAL SYNTAX OF <VAR_SPEC>:
[LEVEL] VARIABLE_NAME [(<DIMENSIONS>)] <ATTRIBUTES>

I. "LEVEL" IS USED ONLY TO DECLARE STRUCTURES (SEE BELOW)

II. <DIMENSIONS>, IF PRESENT, SPECIFIES THAT "VARIABLE_NAME" IS AN ARRAY. SYNTAX:
[LBOUND1:] HBOUND1 [, [LBOUND2:] HBOUND2 [, ...]]
IF "LBOUNDN" IS OMITTED, DEFAULT IS 1. NOTE THAT ARRAYS NEED NOT START AT INDEX=1. PL/P RESTRICTION: ALL BOUNDS MUST BE DECIMAL INTEGER CONSTANTS.

III. <ATTRIBUTES> SPECIFY STORAGE CLASS, SCOPE, INITIAL VALUE, AND DATA TYPE, SIZE, PRECISION, AND ALIGNMENT (SEE BELOW). IF MULTIPLE WORDS ARE USED FOR <ATTRIBUTES>, ORDER IS NOT IMPORTANT.

B. <VAR_SPEC> MAY BE "FACTORED" BY USE OF PARENTHESES SO THAT "LEVEL"S, <ATTRIBUTES>, AND <DIMENSIONS> APPLY TO MORE THAN IDENTIFIER.

D. ATTRIBUTES

1. STORAGE CLASSES

A. AUTOMATIC

I. DEFAULT STORAGE CLASS--MOST FREQUENTLY USED

II. LIKE -DYNM OPTION IN FORTRAN--STORAGE IS IN STACK, SO RECURSIVE INVOCATION GIVES A NEW "GENERATION" OF STORAGE, LEAVING VALUES IN PREVIOUS BUT STILL ACTIVE INVOCATIONS UNTOUCHED

III. DOES NOT NECESSARILY RETAIN VALUE FROM ONE CALL TO THE NEXT

IV. USED FOR MOST VARIABLES UNLESS THERE IS A REASON TO USE SOMETHING ELSE.

B. STATIC

I. LIKE SAVE IN FORTRAN

"*" => PLANNED BUT NOT YET AVAILABLE IN PL/P

II. ALL RECURSIVE INVOCATIONS REFERENCE THE SAME GENERATION OF STORAGE

III. RETAINS VALUE ACROSS INVOCATIONS

IV. USED FOR COMMUNICATION AMONG RECURSIVE INVOCATIONS OF A ROUTINE OR FOR VALUE RETENTION FROM ONE CALL TO THE NEXT

V. STORAGE IS IN LINKAGE SECTION

C. BASED

I. NO STORAGE ALLOCATED FOR BASED VARIABLES AT COMPILE OR LOAD TIME; EITHER REFERENCES STORAGE ALLOCATED FOR ANOTHER VARIABLE (SEE ADDR BUILTIN FUNCTION BELOW) OR STORAGE EXPLICITLY ALLOCATED BY THE PROGRAM AT RUNTIME (SEE ALLOCATE STATEMENT BELOW)

II. MUST BE REFERENCED THROUGH A POINTER (I.E., "PTR -> BASED_VARIABLE")

III. WHEN REFERENCING BASED STORAGE, THE POINTER PROVIDES THE ADDRESS; THE BASED VARIABLE PROVIDES ONLY THE DATA TYPE TEMPLATE TO BE USED IN ACCESSING THE STORAGE.

IV. USED FOR RUN-TIME STORAGE MANAGEMENT (SEE ALLOCATE AND FREE STATEMENTS BELOW), FOR LINKED LISTS OF DATA, FOR EASE OF PASSING STRUCTURES AS ARGUMENTS (BY PASSING A POINTER TO THE STRUCTURE), AND FOR ACCESSING THE SAME STORAGE AS DIFFERENT TYPES OF DATA (E.G., TREATING A WORD OF MEMORY AS 2 ASCII CHARACTERS BUT BEING ABLE TO ACCESS INDIVIDUAL BITS ALSO)

D. PARAMETER

I. APPLIES ONLY TO THE ARGUMENTS OF THE CURRENT PROCEDURE

II. PL/P SUPPORTS THE ADDRESS TYPE BUT NOT THE KEYWORD

E. STORAGE CLASSES ARE MUTUALLY EXCLUSIVE

2. SCOPES

A. INTERNAL--VARIABLE NAME IS KNOWN ONLY TO THE BLOCK WHICH DECLARES IT AND ANY CONTAINED BLOCKS (SEE "SCOPE RULES" BELOW); THIS IS THE DEFAULT SCOPE FOR VARIABLES

B. EXTERNAL

I. THE STORAGE ASSOCIATED WITH THE VARIABLE MAY BE ACCESSED BY ANY OTHER BLOCK WHICH ALSO DECLARES THE VARIABLE NAME AS EXT

II. CORRESPONDS TO FORTRAN NAMED COMMON

III. IMPLIES STATIC; NO OTHER STORAGE CLASS MAY BE GIVEN

3. DATA TYPES, SIZES, PRECISIONS

A. FIXED BIN [(PRECISION)]

I. CORRESPONDS TO FORTRAN INTEGER

II. 1 ≤ "PRECISION" ≤ 31 (FOR PL/P)--NUMBER OF BITS REQUIRED FOR THE ABSOLUTE VALUE OF THE RANGE OF INTEGERS THE VARIABLE REPRESENTS--E.G., FIXED BIN(15) IS THE SAME AS INTEGER*2. MUST BE A DECIMAL INTEGER CONSTANT

III. IF OMITTED, DEFAULT FOR "PRECISION" IS 15--SINGLE WORD INTEGER

IV. TWO ASSOCIATED FORMATS FOR CONSTANTS: BINARY CONSTANTS (";0 < 1=+B", WHERE ";<=" INDICATES CHOICE OF "0" OR "1" AND "+" INDICATES ONE OR MORE INSTANCES OF THE PRECEDING) OR DECIMAL INTEGER CONSTANTS

V. PL/P ALLOWS IMPLICIT CONVERSION TO AND FROM BIT, WHERE THE CORRESPONDING BIT PATTERN IS DEFINED AS THAT OF THE ABSOLUTE VALUE OF THE BINARY ITEM WITH THE SIGN BIT TRUNCATED; IT IS THEN ZERO-PADDED OR TRUNCATED ON THE RIGHT, AS NECESSARY TO MATCH LENGTHS

B. BIT [(SIZE)]

I. ALLOWS ACCESS BY NAMED VARIABLE TO INDIVIDUAL BITS OF MEMORY (INSTEAD OF SHIFTS, TRUNCATES, MASKS, ETC.)

II. 1 ≤ "SIZE" ≤ 16 (FOR PL/P)--NUMBER OF BITS IN THE DATA ITEM (MUST BE DECIMAL INTEGER CONSTANT IN PL/P)

III. BIT(1) MAY BE USED IN THE SAME WAY AS FORTRAN LOGICAL IN IF STATEMENTS (SEE BELOW FOR POSSIBLE VALUES--NOT ".TRUE." AND ".FALSE."!)

IV. IF OMITTED, DEFAULT FOR "SIZE" IS 1

"*" => PLANNED BUT NOT YET AVAILABLE IN PL/P

V. CONSTANTS ARE OF THE FORMAT

"[(FACTOR)] 'CHAR+'B[RADIX]"

WHERE:

"FACTOR" IS THE STRING REPLICATION FACTOR; IT MUST BE A DECIMAL INTEGER CONSTANT AND SPECIFIES THAT THE BIT CONSTANT IS ACTUALLY "FACTOR" CONCATENATED OCCURRENCES OF THE STRING GIVEN

"+" INDICATES ONE OR MORE OCCURRENCES OF "CHAR"

"RADIX" IS THE RADIX FACTOR, A DECIMAL INTEGER CONSTANT WHICH IS INTERPRETED AS THE NUMBER OF BITS REPRESENTED BY EACH "CHAR" AND IMPLIES WHICH CHARACTERS ARE LEGAL IN THE STRING. "B1" OR OMITTED IMPLIES BINARY REPRESENTATION; "B2" IMPLIES QUATERNARY; "B3" IMPLIES OCTAL; AND "B4" IMPLIES HEX.

VI. PL/P ALLOWS IMPLICIT CONVERSION TO FIXED BIN (SEE FIXED BIN FOR DEFINITION OF CONVERSION RESULT)

C. POINTER

I. A POINTER CONTAINS A MEMORY ADDRESS (E.G., THE RESULT OF AN EAL INSTRUCTION). ALL POINTERS IN PL/P ARE OF THE 2-WORD VARIETY--NO BIT OFFSETS.

II. THERE ARE NO POINTER CONSTANTS; VALUES FOR POINTERS ARE OBTAINED SOLELY THROUGH USE OF BUILTIN FUNCTIONS (SEE BELOW)

III. NO CONVERSIONS OF ANY SORT ARE DEFINED FOR POINTERS

D. CHAR [(SIZE)] [;VARYING < NONVARYING=]

I. REPRESENTS A STRING OF ASCII CHARACTERS

II. "SIZE" (MUST BE A DECIMAL INTEGER CONSTANT BETWEEN 1 AND 8192 FOR PL/P) SPECIFIES THE NUMBER OF CHARACTERS THE VARIABLE CAN CONTAIN. "NONVARYING" IMPLIES THAT CHARACTER STRINGS ASSIGNED TO THE VARIABLE WILL BE BLANK-PADDED ON THE RIGHT, IF NECESSARY, SO THAT THE LENGTH WILL ALWAYS BE EXACTLY "SIZE"; VARYING IMPLIES THAT THIS BLANK PADDING WILL NOT OCCUR AND THAT THE ACTUAL LENGTH, WHICH MAY BE ANYTHING FROM 0 TO "SIZE", WILL BE KEPT WITH THE VARIABLE (ACTUALLY, IN THE FIRST WORD OF STORAGE ALLOCATED FOR IT).

"*" => PLANNED BUT NOT YET AVAILABLE IN PL/P

III. "SIZE", IF OMITTED, DEFAULTS TO 1; IF NEITHER VAR NOR NONVARYING IS SPECIFIED, THE DEFAULT IS NONVARYING.

IV. CHARACTER CONSTANTS ARE OF THE FORM

[(FACTOR)] 'CHAR*'

WHERE

"FACTOR" IS THE REPLICATION FACTOR (SEE DESCRIPTION OF BITSTRING CONSTANTS ABOVE)

"*" INDICATES ZERO OR MORE OCCURRENCES OF "CHAR"

"CHAR" IS ANY VALID ASCII CHARACTER; IF A SINGLE QUOTE IS TO BE INCLUDED, IT MUST BE DOUBLED

CHARACTER CONSTANTS MUST NOT INCLUDE <NEWLINE> IN PL/P--I.E., THEY MUST NOT CROSS LINE BOUNDARIES

V. VARYING AND NONVARYING STRINGS MAY BE ASSIGNED TO EACH OTHER; NO OTHER IMPLICIT CONVERSIONS ARE ALLOWED BY PL/P FOR CHARACTER DATA

E. LABEL

I. LABEL VARIABLES IDENTIFY NOT ONLY A PARTICULAR EXECUTABLE STATEMENT BUT ALSO A PARTICULAR INVOCATION OF THE CONTAINING BLOCK

II. LABEL CONSTANTS ARE DEFINED BY THE LABEL NAME OCCURRING IMMEDIATELY BEFORE THE TEXT OF THE ASSOCIATED STATEMENT AND SEPARATED FROM IT BY A COLON. A SINGLE STATEMENT MAY HAVE MULTIPLE LABEL CONSTANTS

III. LABEL CONSTANTS OR VARIABLES MAY BE USED TO INTERFACE WITH FORTRAN ROUTINES WHICH EXPECT AN ALTERNATE-RETURN PARAMETER.

IV. NO CONVERSIONS ARE DEFINED FOR LABELS

F. ENTRY [([<ARG_LIST>])] [RETURNS (<ARG_DESC>)] [;CONSTANT < VARIABLE=] [OPTIONS(SHORTCALL[(SIZE)])]

I. DEFINES THE IDENTIFIER TO BE THE NAME BY WHICH A PROCEDURE OR ENTRY POINT IS TO BE REFERENCED. IF THE RETURNS ATTRIBUTE IS CODED, THE IDENTIFIER MUST BE USED ONLY LIKE A FORTRAN FUNCTION REFERENCE (I.E., THE PROCEDURE NAME MAY BE USED IN ANY

CIRCUMSTANCES WHERE A VARIABLE OF THE TYPE DESCRIBED IN <ARG_DESC> COULD BE); OTHERWISE, THE IDENTIFIER NAME MUST BE USED ONLY IN CALL STATEMENTS. IF "CONSTANT" IS CODED, THE IDENTIFIER IS THE NAME OF AN EXTERNAL PROCEDURE OR ENTRY POINT. IT MUST NOT BE USED OF INTERNAL PROCEDURES; IT MAY NOT BE A MEMBER OF A STRUCTURE, NOR MAY IT BE GIVEN A STORAGE CLASS OR DIMENSIONS. IF "VARIABLE" IS SPECIFIED, HOWEVER, THE NAME SIMPLY REFERENCES A VARIABLE WHOSE VALUE IS THAT OF A PROCEDURE OR ENTRY, EITHER INTERNAL OR EXTERNAL, AND THESE RESTRICTIONS DO NOT APPLY. THE ENTRY VALUE MAY BE ASSIGNED TO THE ENTRY VARIABLE IN ANY OF THE NORMAL WAYS (THE PROGRAMMER IS RESPONSIBLE FOR ASSURING CONSISTENCY OF PARAMETER DECLARATIONS.) IF NEITHER "CONSTANT" NOR "VARIABLE" IS GIVEN, THE DEFAULT IS "CONSTANT".

II. <ARG_LIST> HAS ONE <ARG_DESC> FOR EACH ARGUMENT TO BE PASSED IN THE CALL, SEPARATED BY COMMAS. AN <ARG_DESC> IS IDENTICAL IN FORMAT TO <VAR_SPEC> (SEE ABOVE), EVEN IF A STRUCTURE (SEE BELOW), EXCEPT WITHOUT THE VARIABLE NAME; IN ADDITION, ATTRIBUTES GIVING STORAGE CLASS, SCOPE, AND INITIAL VALUES MUST NOT BE SPECIFIED. EACH <ARG_DESC> DEFINES THE DATA TYPE EXPECTED BY THE CALLED PROCEDURE IN THE CORRESPONDING ARGUMENT POSITION OR RETURNED BY THE PROCEDURE. THE SUPPLIED ARGUMENTS MUST MATCH THE SPECIFIED DESCRIPTORS IN NUMBER AND IN TYPE. AN ERROR OCCURS IF THE WRONG NUMBER OF ARGUMENTS IS SUPPLIED. IF DATA TYPE, PRECISION, SIZE, OR ALIGNMENT MISMATCH OCCURS, THE ARGUMENT IS CONVERTED TO THE EXPECTED TYPE, ETC., IF POSSIBLE (AN ERROR OCCURS IF NOT), THE RESULT IS PLACED IN A COMPILER-GENERATED TEMPORARY, AND THE CALL IS MADE USING THE TEMPORARY IN PLACE OF THE SUPPLIED ARGUMENT (CALL BY VALUE).

III. IF ONLY THE PARENTHESES ARE GIVEN WITH NO <ARG_LIST>, THE CORRESPONDING FUNCTION REFERENCES/CALL STATEMENTS MUST HAVE NO ARGUMENTS. IF THE PARENTHESES ARE ALSO OMITTED, NO CHECKING WILL BE PERFORMED BY THE COMPILER ON ARGUMENT NUMBER OR TYPE.

IV. ONLY IN ENTRY <ARG_LIST>S, THE "SIZE" OF A CHAR <ARG_DESC> MAY BE SPECIFIED AS "*". IF <ARG_DESC> IS VARYING, ANY SIZE VARYING STRING OR CHARACTER CONSTANT MAY BE SUPPLIED AS AN ARGUMENT WITHOUT CAUSING MISMATCH. IF NONVARYING, ANY SIZE NONVARYING STRING OR CHARACTER CONSTANT MAY BE SUPPLIED WITHOUT MISMATCH AND, IN ADDITION, THE GENERATED CODE WILL HAVE 2 ARGUMENTS IN THIS

"*" => PLANNED BUT NOT YET AVAILABLE IN PL/P

POSITION: FIRST, THE STRING, AND SECOND, THE LENGTH OF THE STRING IN CHARACTERS. THIS FEATURE OF CHAR(*) NONVARYING IS USEFUL WHEN INTERFACING WITH MANY FORTRAN ROUTINES WHICH ACCEPT STRING-LENGTH ARGUMENT PAIRS. NOTE THAT MISMATCH BETWEEN VAR/NONVARYING CAUSES CALL BY VALUE.

V. IF CODED WITH "OPTIONS(SHORTCALL[(SIZE)])", THE ENTRY WILL BE ACCESSED VIA JSXB RATHER THAN PCL, AND ARGUMENTS, IF ANY, WILL BE PASSED VIA THE L-REG: IF 1 ARG, THE L-REG WILL POINT TO THE ARGUMENT; IF >1, THE L-REG WILL CONTAIN THE ADDRESS OF A LIST OF 3-WORD POINTERS TO THE ARGUMENTS. A MINIMUM OF 8 WORDS OF SPACE FOR USE BY THE CALLED ROUTINE WILL BE LEFT AT SB%+20 (DECIMAL); THE SIZE MAY BE INCREASED BY USING THE OPTIONAL "SIZE" ARGUMENT (WHICH MUST BE A CONSTANT).

4. INITIAL VALUES

A. SYNTAX: INIT (<VALUE_LIST>)

B. CAUSES THE VARIABLE TO GET THE GIVEN VALUE(S) AT THE TIME OF ALLOCATION-- AUTO, WHEN THE BLOCK IS ENTERED; STATIC, AT LOAD TIME; BASED, ON EXPLICIT ALLOCATION. ILLEGAL FOR PARAMETER.

C. <VALUE_LIST> IS A COMMA'ED LIST OF VALUES OF THE SAME TYPE AS THE VARIABLE BEING INITIALIZED (PL/P RESTRICTION: VALUES MUST BE CONSTANTS). IF AN ARRAY IS BEING INITIALIZED, <VALUE_LIST> MUST HAVE EXACTLY THE SAME NUMBER OF ELEMENTS AS THE ARRAY; IF THE VARIABLE IS SCALAR, THE LIST MUST HAVE ONE ELEMENT. IF SEVERAL CONSECUTIVE VALUES ARE IDENTICAL, THEY MAY BE REPLACED BY "(COUNT) VALUE", WHERE "COUNT" IS THE NUMBER OF VALUES REPLACED (NOTE AMBIGUITY WITH STRING REPLICATION FACTOR IN CASE OF BIT OR CHARACTER VALUES; MUST SPECIFY "(COUNT) (FACTOR) VALUE" TO ITERATE THESE DATA TYPES).

D. MUST NOT BE SPECIFIED FOR STRUCTURES (SEE BELOW); INDIVIDUAL VARIABLES WITHIN A STRUCTURE MAY BE INITIALIZED. ALLOWED BY PL/P ONLY FOR STATIC VARIABLES.

5. ALIGNMENT

A. SYNTAX: [;ALIGNED < UNALIGNED=]

B. "ALIGNED" OPTIMIZES EASE OF ACCESS; "UNAL" OPTIMIZES SPACE USED.

"*" => PLANNED BUT NOT YET AVAILABLE IN PL/P

C. IF OMITTED, "UNAL" IS THE DEFAULT FOR BIT AND CHAR NONVARYING; "ALIGNED" IS THE DEFAULT FOR ALL ELSE.

D. PL/P RESTRICTIONS AND FUNCTION

I. NO EFFECT FOR ANYTHING OTHER THAN BIT AND CHAR NONVARYING; EVERYTHING ELSE IS ALWAYS WORD-ALIGNED

II. IF UNALIGNED, CONTIGUOUS BITSTRINGS IN A STRUCTURE (SEE BELOW) WILL HAVE NO BREAKAGE BETWEEN THEM UNLESS NECESSARY TO AVOID A BITSTRING CROSSING A WORD BOUNDARY; OTHERWISE, EACH ALIGNED BITSTRING AND THE FOLLOWING ITEM WILL BEGIN ON A WORD BOUNDARY. IN ARRAYS OF BITSTRINGS, EACH ELEMENT BEGINS ON A WORD BOUNDARY REGARDLESS OF ALIGNMENT.

III. FOR CHAR DATA, EACH CHARACTER IS ALWAYS BYTE-ALIGNED, REGARDLESS OF ALIGNMENT. IF UNALIGNED, NO BREAKAGE WILL OCCUR BETWEEN ARRAY OR STRUCTURE ELEMENTS OF TYPE CHAR; OTHERWISE, EACH ALIGNED ELEMENT AND THE FOLLOWING ELEMENT WILL BEGIN ON A WORD BOUNDARY.

E. STRUCTURES

1. USED FOR GROUPING LOGICALLY-RELATED BUT DIFFERENT-TYPED DATA IN STORAGE AND FOR LINKED LISTS.

2. MEMBERS OF A STRUCTURE ARE DECLARED JUST LIKE OTHER VARIABLES, EXCEPT THAT "LEVEL" IS PRESENT AND GREATER THAN 1.

3. STRUCTURE MEMBERS MAY THEMSELVES BE STRUCTURES, IN WHICH CASE ONLY ALIGNMENT <ATTRIBUTES> MAY BE SPECIFIED AND THE "LEVEL" OF THE SUCCEEDING VARIABLE MUST BE GREATER THAN THAT OF THE CURRENT MEMBER.

4. THE TOP-LEVEL VARIABLE OF A STRUCTURE MUST HAVE A "LEVEL" OF 1. THIS VARIABLE AND ONLY THIS VARIABLE IN THE STRUCTURE MAY HAVE SCOPE AND STORAGE CLASS SPECIFIED.

5. IF ALIGNMENT IS EXPLICITLY SPECIFIED FOR A STRUCTURE, THAT ALIGNMENT IS THE DEFAULT FOR ALL MEMBERS OF THE STRUCTURE UNLESS OVERRIDEN BY A CONTAINED STRUCTURE FOR ITS OWN MEMBERS; OTHERWISE, ALIGNMENT FOR MEMBERS OF A STRUCTURE IS DETERMINED AS DESCRIBED ABOVE.

6. A STRUCTURE MEMBER NAME MUST BE UNIQUE WITHIN ITS IMMEDIATELY-CONTAINING STRUCTURE, BUT IT NEED NOT BE UNIQUE BEYOND THAT.

7. TO AVOID AMBIGUITY, STRUCTURE MEMBERS MAY BE "QUALIFIED" BY PRECEDING THE MEMBER NAME BY THE NAME OF A CONTAINING

STRUCTURE AND SEPARATING THEM BY A ".". AS MANY CONTAINING STRUCTURE NAMES AS ARE PRESENT MAY BE SPECIFIED USING THIS SYNTAX.

8. LIKE

A. SYNTAX: LIKE STRUCTURE_REFERENCE

B. "STRUCTURE_REFERENCE" IS THE NAME OF A STRUCTURE DECLARED EARLIER IN THE PROGRAM AND KNOWN TO THE BLOCK CONTAINING THE LIKE DECLARATION (SEE "SCOPE RULES" BELOW); IT NEED NOT BE A LEVEL-1 STRUCTURE.

C. THE EFFECT IS AS IF THE DECLARATIONS OF ALL MEMBERS OF "STRUCTURE_REFERENCE" HAD BEEN COPIED DIRECTLY FOLLOWING THE VARIABLE WITH THE LIKE ATTRIBUTE, EXCEPT THAT LEVEL NUMBERS ARE ADJUSTED UPWARD OR DOWNWARD AS NECESSARY TO BE COMPATIBLE WITH THEIR POSITION IN THE NEW STRUCTURE.

D. RESTRICTIONS: "STRUCTURE_REFERENCE" MUST HAVE BEEN DECLARED BEFORE ITS USAGE IN THE LIKE SPECIFICATION (PL/P ONLY); "STRUCTURE_REFERENCE" MUST NOT CONTAIN A LIKE ATTRIBUTE.

9. ANY SUBSCRIPTING REQUIRED IN THE "CHAIN" OF STRUCTURE REFERENCES MAY BE SPECIFIED AS IF THE MEMBER REFERENCED ITSELF HAD ALL THE DIMENSIONS REQUIRED.

10. IF A STRUCTURE IS BASED, A REFERENCE TO A MEMBER ASSUMES THE POINTER CONTAINS THE ADDRESS OF THE BASE, OR TOP LEVEL, OF THE STRUCTURE, AND THE OFFSET OF THE MEMBER WITHIN THE STRUCTURE IS ADDED TO THE VALUE OF THE POINTER IN EVALUATING THE REFERENCE.

11. IF A STRUCTURE IS EXTERNAL, ONLY THE TOP LEVEL NAME NEED MATCH DECLARATIONS IN OTHER BLOCKS; MEMBER NAMES ARE NOT CHECKED.

* 12. REFER

A. SYNTAX: <ALLOCATION_LENGTH> REFER REFERENCE_LENGTH

B. ALLOWS BASED STRUCTURES TO BE SELF-DEFINING IN THE AMOUNT OF STORAGE OCCUPIED BY A GENERATION OF THE STRUCTURE; REPLACES A BOUND OF AN ARRAY MEMBER OR THE SIZE OF A CHARACTER STRING MEMBER.

C. <ALLOCATION_LENGTH> IS AN EXPRESSION WHICH IS EVALUATED WHENEVER THE STRUCTURE IS ALLOCATED (SEE THA ALLOCATE STATEMENT BELOW); THE VALUE IS USED TO DETERMINE THE AMOUNT OF STORAGE TO ALLOCATE FOR THE STRUCTURE AND AFTER ALLOCATION THE VALUE IS ASSIGNED TO

"REFERENCE_LENGTH" IN THE NEW GENERATION. IF THE STRUCTURE IS SIMPLY USED TO OVERLAY ALREADY-ALLOCATED STORAGE, I.E., IT IS NEVER THE OBJECT OF AN ALLOCATE STATEMENT, <ALLOCATION_LENGTH> IS IGNORED AND MAY EVEN BE THE CONSTANT "0".

D. "REFERENCE_LENGTH" MUST BE AN ELEMENTARY (I.E., NON_STRUCTURE) SCALAR (INCLUDING INHERITED DIMENSIONS) MEMBER OF THE STRUCTURE; IT MUST PRECEDE THE MEMBER CONTAINING THE REFER OPTION WHICH REFERENCES IT, AND ITS "LEVEL" MUST BE LESS THAN OR EQUAL TO THAT OF THE MEMBER REFERENCING IT. WHENEVER THE STRUCTURE IS REFERENCED IN THE PROGRAM, "REFERENCE_LENGTH" IS USED AS THE ARRAY BOUND OR STRING SIZE OF THE MEMBER CONTAINING THE REFER OPTION.

E. PL/P RESTRICTIONS: ONLY 1 REFER OPTION PER LEVEL-1 STRUCTURE; THE MEMBER CONTAINING THE REFER MUST BE THE LAST MEMBER OF THE STRUCTURE; "REFERENCE_LENGTH" MUST BE "FIXED BIN(15)".

13. FORTRAN COMPATIBILITY

A. AN EXTERNAL STRUCTURE IS EQUIVALENT TO A COMMON BLOCK WITH THE NAME OF THE TOP LEVEL STRUCTURE AND WITH THE STRUCTURE MEMBERS HAVING DATA-TYPE <ATTRIBUTES> AS VARIABLES IN THE BLOCK.

B. A STRUCTURE WHOSE MEMBERS ARE UNALIGNED BIT STRINGS ALLOWS NAMED ACCESS TO INDIVIDUAL BITS, WHICH IN FORTRAN IS ACCOMPLISHED BY MEANS OF MASKS, SHIFTS, AND TRUNCATES.

F. SCOPE RULES

1. BLOCKS ARE DELIMITED BY BEGIN-END AND PROC-END PAIRS (SEE BELOW); THESE BLOCKS ARE INDISTINGUISHABLE IN THEIR EFFECTS ON IDENTIFIER SCOPES.

2. A INTERNAL VARIABLE IS KNOWN TO THE BLOCK IN WHICH THE VARIABLE IS DECLARED AND IN ALL CONTAINED BLOCKS, UNLESS THE CONTAINED BLOCK OR AN INTERVENING PARENT BLOCK HAS A DECLARATION OF THAT VARIABLE.

3. VARIABLES MAY BE REFERENCED ONLY IN BLOCKS IN WHICH THEY ARE KNOWN.

4. THE SCOPE OF AN INTERNAL PROCEDURE NAME IS THE SAME AS THAT OF AN INTERNAL VARIABLE DECLARED IN THE BLOCK CONTAINING THE PROCEDURE.

5. THE SCOPE OF AN ENTRY NAME IS THE SAME AS THAT OF THE PROCEDURE IMMEDIATELY CONTAINING THE ENTRY STATEMENT.

"*" => PLANNED BUT NOT YET AVAILABLE IN PL/P

III. EXPRESSIONS

A. TYPES OF EXPRESSIONS

1. OPERATORS--PERFORM AN OPERATION ON ARGUMENT(S), YIELDING THE SAME TYPE WITH POSSIBLY DIFFERENT PRECISION OR SIZE; PREFIX AND INFIX NOTATION.

2. BUILTIN FUNCTIONS--LIKE FORTRAN INTRINSICS, EXCEPT NAME IS NOT RESERVED. MUST NOT BE DECLARED IN PL/P.

3. COMPARISONS--RESULT IN BIT(1), VALUE DEPENDING ON WHETHER THE COMPARISON WAS TRUE OR FALSE; MAY BE USED ANYPLACE A BIT(1) VALUE IS NEEDED.

B. ARITHMETIC EXPRESSIONS

1. OPERATORS--STANDARD FORTRAN TYPES: "+", "-", "*", "/", "**" (LAST 2 NOT SUPPORTED BY PL/P; FOR DIVISION, SEE NEXT)

2. BUILTINS

   A. MOD (NUMBER, MODULUS) RETURNS (FIXED BIN)

   B. DIVIDE (DIVIDEND, DIVISOR, RESULT_PRECISION) RETURNS (FIXED BIN)

3. COMPARISONS: "=", "^=", ">", "^>", ">=", "<", "^<", "<="

C. CHARACTER EXPRESSIONS

1. OPERATORS: "<<" (CONCATENATION)

2. BUILTINS

   A. SEARCH (STRING1, STRING2) RETURNS (FIXED BIN)--RESULT IS CHARACTER POSITION WITHIN STRING 1 OF THE FIRST OCCURRENCE OF ANY CHARACTER IN STRING2, OR 0 IF NONE (NONSTANDARD)

   B. BIN (STRING [,RESULT_PRECISION]) RETURNS (FIXED BIN)--RESULT IS THE NUMERIC VALUE OF THE STRING WHEN INTERPRETED AS THE CHARACTER REPRESENTATION OF A DECIMAL INTEGER

   C. CHAR (INTEGER [, RESULT_SIZE_IN_CHARS]) RETURNS (CHAR(*))--RESULT IS THE CHARACTER REPRESENTATION OF THE INTEGER AS A DECIMAL NUMBER. THE SIZE WILL BE NO LESS THAN THAT REQUIRED TO REPRESENT THE MOST NEGATIVE VALUE AN INTEGER OF THE PRECISION OF THE FIRST ARGUMENT MAY TAKE, INCLUDING THE MINUS SIGN, BUT MAY BE INCREASED BY MEANS OF THE SECOND ARGUMENT. IF NOT ALL

"*" => PLANNED BUT NOT YET AVAILABLE IN PL/P

CHARACTER POSITIONS ARE USED IN REPRESENTING THE NUMBER, LEADING SPACES WILL BE ADDED TO COMPLETE THE SIZE; NO SPACE WILL OCCUR BETWEEN THE "-", IF PRESENT, AND THE FIRST DIGIT.

D. AFTER (STRING1, STRING2) RETURNS (CHAR(*))--RETURNS THE REMAINDER OF STRING1 WHICH FOLLOWS THE FIRST OCCURRENCE OF STRING2, OR '' (THE NULL STRING) IF NONE

E. BEFORE (STRING1, STRING2) RETURNS (CHAR(*))--RETURNS THE PORTION OF STRING1 WHICH PRECEDES THE FIRST OCCURRENCE OF STRING2, OR THE ENTIRE STRING1 IF STRING2 IS NOT FOUND

F. COPY (STRING1, COUNT) RETURNS (CHAR(*))--RESULT IS "COUNT" CONCATENATED OCCURRENCES OF STRING1 (PL/P RESTRICTS "COUNT" TO BE A CONSTANT)

G. DATE () RETURNS CHAR(6)--RESULT IS CURRENT DATE IN FORMAT YYMMDD

H. INDEX (STRING1, STRING2) RETURNS (FIXED BIN)--RESULT IS CHARACTER POSITION OF FIRST OCCURRENCE OF STRING2 IN STRING1, OR 0 IF NONE

I. LENGTH (STRING) RETURNS (FIXED BIN)--RESULT IS CURRENT LENGTH OF STRING (A CONSTANT FOR CHAR NONVARYING)

J. REVERSE (STRING) RETURNS (CHAR(*))--VALUE IS RESULT OF INTERCHANGING FIRST AND LAST CHARACTERS, SECOND AND NEXT-TO-LAST, ETC.

K. SUBSTR (STRING, START_POSITION [, RESULT_LENGTH]) RETURNS ((CHAR(*))--RESULT IS SUBSTRING OF "STRING" STARTING AT "START_POSITION" AND RUNNING FOR "RESULT_LENGTH" CHARACTERS, IF SPECIFIED, OR UNTIL THE END OF "STRING" IF NOT

L. TIME () RETURNS (CHAR(9))--RESULT IS CURRENT TIME IN FORMAT HHMMSSMMM

M. TRANSLATE (STRING1, OUTPUT_CHARS [, INPUT_CHARS]) RETURNS (CHAR(*))--RESULT IS COMPUTED ACCORDING TO THE FOLLOWING ALGORITHM:

```
FOR EACH CHARACTER OF STRING1:
  POSN = INDEX (INPUT_CHARS, STR1_CHAR)
  IF POSN = 0
      THEN RESULT_CHAR = STR1_CHAR
  ELSE RESULT_CHAR = SUBSTR (OUTPUT_CHARS, POSN, 1)
```

IF "INPUT_CHARS" IS OMITTED, THE ASCII COLLATING SEQUENCE IS USED.

"*" => PLANNED BUT NOT YET AVAILABLE IN PL/P

N. VERIFY (STRING1, STRING2) RETURNS (FIXED BIN)--RESULT IS CHARACTER POSITION OF FIRST CHARACTER OF STRING1 WHICH IS NOT IN STRING2, OR 0 IF NONE

O. TRIM (STRING, CONTROL_BITS [, TRIM_CHAR]) RETURNS (CHAR(*))--RESULT IS WHAT IS LEFT OF "STRING" AFTER REMOVING LEADING AND/OR TRAILING "TRIM_CHAR"S FROM IT. "CONTROL_BITS" IS A BITSTRING OF LENGTH 2 WHOSE FIRST BIT SPECIFIES REMOVAL OF LEADING CHARACTERS AND WHOSE SECOND BIT IMPLIES REMOVAL OF TRAILING CHARACTERS. IF "TRIM_CHAR" IS OMITTED, ASCII "SPACE" IS THE DEFAULT. (NONSTANDARD)

3. COMPARISONS: SAME AS ARITHMETIC; ORDINAL COMPARISONS ARE BASED ON ASCII COLLATING SEQUENCE WITH SHORTER ITEM BLANK-PADDED ON THE RIGHT TO THE LENGTH OF THE LONGER.

D. POINTER EXPRESSIONS

1. THERE ARE NO POINTER OPERATORS.

2. BUILTINS

A. STANDARD

I. NULL() RETURNS (PTR)--RESULT IS A POINTER WHICH WILL GENERATE A FAULT IF USED.

II. ADDR (VARIABLE) RETURNS (PTR)--RESULT IS A POINTER WITH THE VALUE OF THE ADDRESS OF THE VARIABLE.

B. NONSTANDARD (DEPEND ON PRIME REPRESENTATION OF POINTERS)

I. SEGNO (PTR1) RETURNS (BIT(12))--RESULT IS SEGMENT NUMBER OF GIVEN POINTER

II. RING (PTR2) RETURNS (BIT(2))--RESULT IS RING NUMBER OF GIVEN POINTER

III. REL (PTR1) RETURNS (FIXED BIN(15))--RESULT IS WORD OFFSET OF GIVEN POINTER

IV. PTR (SEGNO, WORDNO [, RINGNO]) RETURNS (PTR)--RESULT IS POINTER WHOSE VALUE IS GIVEN BY THE ARGUMENTS; SEGNO IS BIT(12), WORDNO IS FIXED BIN(15), AND RING IS BIT(2)

V. ADDREL (PTR1, REL_OFFSET) RETURNS (PTR)--RESULT IS A POINTER WHOSE SEGMENT NUMBER IS THAT OF "PTR1" AND WHOSE WORD NUMBER IS THAT OF "PTR1" PLUS "REL_OFFSET"

VI. BASEPTR (PTR1) RETURNS (PTR)--RESULT IS A POINTER WHOSE SEGMENT NUMBER IS THAT OF "PTR1" AND WHOSE WORD NUMBER IS 0

VII. BASEREL (PTR1, WORD_OFFSET) RETURNS (PTR)--RESULT IS A POINTER WHOSE SEGMENT NUMBER IS THAT OF "PTR1" AND WHOSE WORD NJMBER IS GIVEN BY "WORD_OFFSET"

VIII. STACKPTR () RETURNS (PTR)--RESULT IS A POINTER WHOSE VALUE IS THE ADDRESS OF THE CURRENT STACK FRAME

IX. STACKBASE () RETURNS (PTR)--RESULT IS A POINTER WHOSE VALUE IS THE STACK BASE

X. LINKPTR () RETURNS (PTR)--RESULT IS A POINTER WHOSE VALUE IS LB%+256

3. COMPARISONS--ONLY TESTS FOR EQUALITY ("=", "^=") ARE ALLOWED

E. BIT EXPRESSIONS

1. OPERATORS--'^' (COMPLEMENT), '&' (LOGICAL AND), '|' (LOGICAL OR), '||' (CONCATENATION; NOT YET SUPPORTED BY PL/P)

2. BUILTINS

A. SUBSTR--SAME AS FOR CHARACTER EXCEPT FIRST ARGUMENT IS A BITSTRING AND POSITIONS AND LENGTHS REFER TO BITS INSTEAD OF CHARACTERS. PL/P RESTRICTION: LENGTH MUST BE COMPUTABLE AT COMPILE TIME.

B. BIN--SAME AS FOR CHARACTER, EXCEPT FIRST ARGUMENT IS A BITSTRING AND THE RESULT IS DERIVED BY STANDARD BIT TO BINARY CONVERSION DESCRIBED UNDER FIXED BIN ABOVE.

C. BIT (INTEGER [, RESULT_SIZE_IN_BITS]) RETURNS (BIT(*))--RESULT IS DERIVED BY STANDARD BINARY TO BIT CONVERSION DESCRIBED ABOVE.

* D. SOME (BITSTRING) RETURNS (BIT(1))--RESULT IS '1'B IF ANY BIT IN "BITSTRING" IS '1'B, '0'B OTHERWISE.

* E. EVERY (BITSTRING) RETURNS (BIT(1))--RESULT IS '1'B IF EVERY BIT IN "BITSTRING" IS '1'B, '0'B OTHERWISE.

3. COMPARISONS--SAME AS CHARACTER

4. COMPARISONS AND INFIX OPERATORS ARE DEFINED TO WORK ONLY ON BITSTRINGS OF THE SAME LENGTH; THE SHORTER OPERAND IS RIGHT PADDED WITH '0'B TO THE LENGTH OF THE LONGER.

"*" => PLANNED BUT NOT YET AVAILABLE IN PL/P

F. OPERATING SYSTEM BUILTIN FUNCTIONS (NONSTANDARD)

1. CSTORE--GENERATES "STAC" OR "STLC", DEPENDING ON SIZE OF OPERANDS. FIRST ARG IS TARGET, SECOND IS OLD VALUE, THIRD IS NEW VALUE. RESULT = '1'B IF OPERATION FAILED, '0'B OTHERWISE.

2. REGFILE (OFFSET) RETURNS(FIXED BIN(31))--GENERATES "LDLR OFFSET"

3. ADDQT (QUEUE, ITEM) RETURNS(BIT(1))--ADDS "ITEM" TO THE TOP OF "QUEUE" VIA "ATQ"; RESULT = '1'B IF SUCCESSFUL, '0'B IF QUEUE IS FULL.

4. ADDQB--SAME AS ADDQT, WITH "ABQ"

5. REMQT (QUEUE, ITEM) RETURNS(BIT(1))--REMOVES TOP ITEM OF "QUEUE", STORING IT IN "ITEM", VIA "RTQ"; RESULT = '1'B IF SUCCESSFUL, '0'B IF QUEUE IS EMPTY.

6. REMQB--SAME AS REMQT WITH "RBQ"

7. TESTQ (QUEUE) RETURNS (FIXED BIN)--RESULT IS THE NUMBER OF ITEMS IN "QUEUE"

IV. STATEMENTS

A. PROC_NAME: PROCEDURE [(<PARAMETER_LIST>)] [RETURNS (<ARG_DESC>)] [OPTIONS (<OPTION_LIST>)];

1. "PROC_NAME" IS THE NAME USED IN CALL STATEMENTS AND ENTRY DECLARATIONS.

2. <PARAMETER_LIST> IS THE COMMA'ED LIST OF VARIABLE NAMES BY WHICH THE PROCEDURE WILL REFER TO ITS PARAMETERS.

3. RETURNS MUST BE CODED IFF THE PROCEDURE IS REFERENCED AS A FUNCTION; <ARG_DESC> HAS THE SAME SYNTAX AS THE CORRESPONDING ITEM IN THE ENTRY DECLARATION (SEE ABOVE).

4. MUST BE THE FIRST STATEMENT IN ANY COMPILATION; A "MAIN" ROUTINE, IN THE FORTRAN SENSE, DOES NOT EXIST.

5. EACH PROC STATEMENT MUST HAVE A MATCHING END STATEMENT, WHICH TERMINATES THE PROCEDURE.

6. MAY BE USED INSIDE ANOTHER PROCEDURE FOR INTERNAL SUBROUTINES; SEE "SCOPE RULES" ABOVE FOR EFFECT ON SCOPES OF IDENTIFIERS. IF NORMAL FLOW OF CONTROL REACHES AN INTERNAL PROC STATEMENT, THE ENTIRE INTERNAL PROCEDURE WILL BE SKIPPED. AN INTERNAL SUBROUTINE MAY NOT BE CALLED UNDER ANY CIRCUMSTANCES UNLESS THE IMMEDIATELY CONTAINING BLOCK IS ACTIVE, I.E., HAS BEEN INVOKED AND HAS NOT EXITED.

"*" => PLANNED BUT NOT YET AVAILABLE IN PL/P

7. IN PL/P, ALL PROCEDURES ARE RECURSIBLE.

8. <OPTION_LIST> IS A COMMA'ED LIST OF OPTIONS.

A. NOCOPY--MAY BE SPECIFIED FOR EXTERNAL PROCS TO SUPPRESS CALL-BY-VALUE FOR CONSTANTS.

B. GATE--CAUSES RING WEAKENING TO OCCUR FOR ALL BASED, PARAMETER, AND EXTERNAL POINTER ASSIGNMENTS.

C. SAVE (REF)--CAUSES AN RSAV INTO "REF" BEFORE ANY OTHER CODE IN THE PROCEDURE IS EXECUTED.

D. SHORTCALL--MAY BE SPECIFIED FOR INTERNAL PROCS TO SPEED UP CALLING AND RETURNING BY USING JSY INSTEAD OF PCL.

* B. ENTRY_NAME: ENTRY [(<PARAMETER_LIST>)] [RETURNS (<ARG_DESC>)] [OPTIONS (<OPTION_LIST>)];

1. DEFINES ANOTHER ENTRY POINT TO THE PROCEDURE CONTAINING THE ENTRY STATEMENT BESIDES THE MAIN ONE (I.E., THE PROC STATEMENT).

2. HAS NO EFFECT ON SCOPES AND TAKES NO MATCHING END STATEMENT; OTHERWISE, FUNCTION AND SYNTAX IS EXACTLY THE SAME AS A PROC STATEMENT.

3. "INVISIBLE" TO NORMAL FLOW OF CONTROL.

4. WHEN AN ENTRY STATEMENT IS CALLED, EXECUTION BEGINS IMMEDIATELY FOLLOWING THE ENTRY STATEMENT INSTEAD OF AT THE BEGINNING OF THE CONTAINING PROCEDURE.

C. IF <EXPRESSION> THEN [<EXECUTABLE_STATEMENT>]; [ELSE [<EXECUTABLE_STATEMENT>];]

1. <EXPRESSION> MUST BE CAPABLE OF BEING CONVERTED TO BIT; IF ANY BIT IN THE RESULTING STRING IS NON-ZERO, THE CONDITION IS TRUE.

2. IF STATEMENTS MAY BE NESTED; ELSE PARTS MATCH ON A LIFO BASIS.

D. BEGIN;

1. TAKES A MATCHING END STATEMENT; THE INTERVENING STATEMENTS ARE KNOWN AS A BEGIN BLOCK.

2. AN ENTIRE BEGIN BLOCK IS CONSIDERED AS AN EXECUTABLE STATEMENT FOR THE PURPOSES OF THEN AND ELSE (SEE ABOVE), ALLOWING MULTIPLE STATEMENTS TO BE CONTROLLED BY AN IF WITHOUT USE OF GOTOS.

"*" => PLANNED BUT NOT YET AVAILABLE IN PL/P

3. FOR EFFECT ON IDENTIFIER SCOPES, SEE "SCOPE RULES" ABOVE.

F. TWO FLAVORS OF "DO"

1. ALL DOS TAKE A MATCHING END STATEMENT; INTERVENING STATEMENTS ARE KNOWN AS A DO GROUP.

2. ANY DO GROUP MAY BE USED LIKE A BEGIN BLOCK AS AN EXECUTABLE STATEMENT AFTER A THEN OR ELSE.

3. SIMPLE DO - "DO;" - LIKE BEGIN EXCEPT WITHOUT EFFECT ON IDENTIFIER SCOPES AND HENCE CHEAPER

4. COMPLEX DOS

A. ALL COMPLEX DOS ARE ITERATIVE, I.E., THEY LOOP UNTIL SOME TEST HAS BEEN FAILED. THIS CONDITION IS TESTED AT THE BEGINNING OF THE LOOP, SO THAT IF THE TEST IS FAILED UPON ENTERING THE GROUP, IT WILL NOT BE EXECUTED AT ALL.

B. DO WHILE (<EXPRESSION>);

I. <EXPRESSION> IS EVALUATED EXACTLY LIKE THE <EXPRESSION> IN AN IF STATEMENT.

II. THE STATEMENTS IN THE DO GROUP ARE REPETITIVELY EXECUTED UNTIL THE CONDITION IS FALSE AT THE BEGINNING OF AN ITERATION.

C. DO INDEX = <SPEC_LIST>;

I. FOR PL/P, "INDEX" IS RESTRICTED TO BE EITHER A FIXED BIN(15) OR PTR.

II. <SPEC_LIST> IS A COMMA'ED LIST OF <SPEC>S; IF MORE THAN ONE <SPEC> IS SUPPLIED, THE GROUP IS EXECUTED UNDER CONTROL OF THE FIRST <SPEC> UNTIL THE TEST IN THAT <SPEC> FAILS; THE GROUP IS THEN EXECUTED UNDER CONTROL OF THE SECOND <SPEC> UNTIL THAT TEST FAILS; ETC.

III. <SPEC> SYNTAX: INIT_VALUE [;[TO FINAL_VALUE] [BY INCREMENT] < REPEAT <EXPRESSION1>=] [WHILE (<EXPRESSION2>)]

IV. "INIT_VALUE" IS ASSIGNED TO "INDEX" BEFORE EXECUTING THE LOOP THE FIRST TIME. THE TO/BY OPTION MAY ONLY BE USED IF "INDEX" IS A FIXED BIN(15); "INCREMENT" MAY BE NEGATIVE. IF THIS OPTION IS CHOSEN AND TO IS SPECIFIED, THE LOOP IS EXECUTED, INCREMENTING THE VALUE OF "INDEX" BY "INCREMENT" AFTER EACH LOOP, UNTIL THE VALUE OF

(FINAL_VALUE - INDEX) * INCREMENT IS LESS THAN 0. IF THE TO OPTION IS OMITTED BUT THE BY OPTION IS SPECIFIED, THE CHECK AT THE BEGINNING OF EACH LOOP IS OMITTED, BUT INCREMENTATION CONTINUES AS DESCRIBED. IF THE BY OPTION AND "INCREMENT" ARE NOT SPECIFIED, THE DEFAULT IS 1. IF TO AND BY ARE BOTH OMITTED, THE GROUP IS EXECUTED AT MOST ONCE, SUBJECT TO THE WHILE CLAUSE (SEE BELOW).

VI. IF THE REPEAT OPTION IS SPECIFIED, AT THE END OF EACH LOOP <EXPRESSION1> IS EVALUATED AND THE VALUE ASSIGNED TO "INDEX" FOR THE NEXT ITERATION. THIS IS PARTICULARLY USEFUL FOR TRAVERSING LINKED LISTS.

VII. IF THE WHILE OPTION IS CODED, THE LOOP WILL BE TERMINATED IF <EXPRESSION2> YIELDS A FALSE VALUE WHEN THE CONDITIONS ARE BEING EVALUATED AT THE BEGINNING OF EACH ITERATION. IF CODED WITH THE TO/BY OPTION, THE WHILE CLAUSE CAN ONLY SHORTEN, NOT EXTEND, THE LOOP DURATION. NOTE: THE WHILE OPTION USED IN THIS WAY APPLIES ONLY TO THE CURRENT <SPEC>.

D. UNTIL (NONSTANDARD) MAY BE USED IN PLACE OF OR IN CONJUNCTION WITH WHILE. THE SYNTAX IS THE SAME, BUT THE SENSE OF THE TEST IS INVERTED AND IS PERFORMED AT THE END OF THE LOOP INSTEAD OF THE BEGINNING.

E. END; - TERMINATES CURRENT BLOCK/GROUP/PROCEDURE. MUST HAVE EXACT NUMBER TO TERMINATE ALL OUTSTANDING BLOCKS/GROUPS/PROCEDURES.

G. CALL NAME [<ARG_LIST>];

1. <ARG_LIST> MUST MATCH THE PARAMETER LIST IN THE ENTRY DECLARATION FOR "NAME" IN NUMBER, DATA TYPE, ALIGNMENT, ETC. (SEE ENTRY ATTRIBUTE ABOVE).

2. CALL BY VALUE CAN BE SELECTED FOR ANY ARGUMENT(S) BY ENCLOSING THE VARIABLE NAME IN PARENTHESES (CONSTANTS AND EXPRESSIONS ARE AUTOMATICALLY DONE AS CALL BY VALUE). IN ADDITION, THIS TECHNIQUE CAN BE USE TO SUPPRESS COMPILER WARNING MESSAGES ABOUT ARGUMENT/PARAMETER MISMATCH.

3. THE FOLLOWING UNDECLARED NAMES ("BUILTIN SUBROUTINES") MAY BE CALLED, BUT INSTEAD OF INVOKING A PROCEDURE THEY GENERATE A PARTICULAR MACHINE INSTRUCTION. (NONSTANDARD FEATURE)

A. INHIBIT()--GENERATES "INH"

B. ENABLE()--GENERATES "ENB"

"*" => PLANNED BUT NOT YET AVAILABLE IN PL/P

C. WAIT(SEMAPHORE)--GENERATES "WAIT"

D. NOTIFYB(SEMAPHORE)--GENERATES "NFYB"

E. NOTIFYE(SEMAPHORE)--GENERATES "NFYE"

H. GO TO LABEL;

1. "LABEL" MAY BE EITHER A LABEL CONSTANT OR VARIABLE; IF THE LABEL IS NOT IN THE CURRENT BLOCK (PROCEDURE OR BEGIN BLOCK), AN IMPLICIT RETURN IS DONE TO THE INVOCATION OF THE BLOCK CONTAINING THE LABEL--I.E., ALL INTERVENING STACK FRAMES ARE POPPED BY A NON-LOCAL GOTO.

2. MAY BE ABBREVIATED "GOTO" (NO SPACE).

I. RETURN [(<EXPRESSION>)];

1. <EXPRESSION> MUST BE PROVIDED IFF THE ASSOCIATED PROC OR ENTRY STATEMENT WAS CODED WITH THE RETURNS OPTION (SEE ABOVE); THIS MECHANISM REPLACES THE FORTRAN PRACTICE OF ASSIGNING THE RETURN VALUE TO THE FUNCTION NAME.

2. THE RETURN STATEMENT WITH NO EXPRESSION IS OPTIONAL; IF THE FLOW OF CONTROL ENCOUNTERS THE END STATEMENT OF A PROCEDURE, AN IMPLICIT RETURN IS DONE.

J. ASSIGNMENT STATEMENT

1. SYNTAX: ;<VARIABLE_REFERENCE> < <PSEUDOVARIABLE>= = <EXPRESSION>;

2. <VARIABLE_REFERENCE> MAY BE STRUCTURE OR POINTER QUALIFIED OR SUBSCRIPTED.

3. A PSEUDOVARIABLE IS LIKE A BUILTIN FUNCTION EXCEPT THAT IT RECEIVES A VALUE.

A. SUBSTR--THE PORTION OF THE FIRST ARGUMENT SPECIFIED BY THE SECOND AND OPTIONAL THIRD ARGUMENT IS REPLACED BY THE EXPRESSION ON THE RIGHT HAND SIDE OF THE "=". VALID FOR BIT AND CHAR. NOTE-- THE SUBSTR PSEUDOVARIABLE DOES NOT CHANGE THE LENGTH OF A CHAR VAR ARGUMENT.

B. REGISTERS()--NO ARGUMENTS; CAUSES AN "RRST" FROM THE VALUE OF THE RIGHT HAND SIDE. (NONSTANDARD)

C. REGFILE (OFFSET)--CAUSES "STLR OFFSET" USING THE VALUE OF THE RIGHT HAND SIDE. (NONSTANDARD)

4. ANY PL/I DATA TYPE MAY BE ASSIGNED.

"*" => PLANNED BUT NOT YET AVAILABLE IN PL/P

* K. ALLOCATE BASED_NAME SET (PTR_REFERENCE);

1. CAUSES STORAGE TO BE ALLOCATED FROM A SYSTEM-MANAGED POOL OF FREE STORAGE, PERFORMS VALUE INITIALIZATION AS REQUIRED BY INIT AND REFER CLAUSES IN THE DECLARATION OF "BASED_NAME", AND SETS THE VALUE OF A POINTER TO THE ADDRESS OF THE BEGINNING OF THE STORAGE ALLOCATED.

2. THE AMOUNT OF STORAGE ALLOCATED IS CALCULATED BY THE COMPILER FROM THE DECLARATION OF "BASED_NAME" RATHER THAN BEING SPECIFIED IN THE STATEMENT.

3. "BASED_NAME" IS THE NAME OF A VARIABLE, SIMPLE OR AGGREGATE, WHICH WAS DECLARED WITH THE BASED ATTRIBUTE AND IS KNOWN TO THE BLOCK CONTAINING THE ALLOCATE STATEMENT; IT MUST NOT BE POINTER-QUALIFIED.

4. "PTR_REFERENCE" RECEIVES THE ADDRESS OF THE ALLOCATED STORAGE.

* L. FREE <BASED_REFERENCE>;

1. RETURNS TO THE SYSTEM-MANAGED POOL OF FREE STORAGE A SINGLE GENERATION OF STORAGE WHICH HAS BEEN EXPLICITLY ALLOCATED (I.E., NO STORAGE MAY BE FREED WHICH HAS NOT BEEN THE OBJECT OF AN ALLOCATE STATEMENT).

2. THE AMOUNT OF STORAGE FREED IS DETERMINED BY THE COMPILER FROM THE DECLARATION OF THE VARIABLE IN <BASED_REFERENCE> RATHER THAN BEING SPECIFIED IN THE STATEMENT.

3. THE GENERATION OF STORAGE TO BE FREED IS DETERMINED BY THE ADDRESS VALUE OF A POINTER; THEREFORE, <BASED_REFERENCE> MUST BE EXPLICITLY POINTER-QUALIFIED. WITH A POINTER.

* M. SELECT [<SELECT_EXPRESSION>];

1. A "CASE" STATEMENT (NOT IN STANDARD PL/I); TERMINATED BY AN END STATEMENT; MAY BE USED AS THE <EXECUTABLE_STATEMENT> OF A THEN OR ELSE CLAUSE.

2. THE CASES ARE SPECIFIED AS

    WHEN <EXPRESSION_LIST> <EXECUTABLE_STATEMENT>;

    WHERE <EXPRESSION_LIST> IS A COMMA'ED LIST OF <EXPRESSION>S; THE LAST CASE MAY BE

    OTHERWISE <EXECUTABLE_STATEMENT>;

3. IF <SELECT_EXPRESSION> IS SPECIFIED, THE <EXECUTABLE_STATEMENT> OF THE FIRST WHEN CLAUSE, ANY OF WHOSE <EXPRESSION>S COMPARES EQUAL TO <SELECT_EXPRESSION>,

IS EXECUTED; IF <SELECT_EXPRESSION> IS OMITTED, WHEN <EXPRESSION>S ARE EVALUATED LIKE THE <EXPRESSION> IN AN IF STATEMENT, AND THE <EXECUTABLE_STATEMENT> OF THE FIRST WHEN CLAUSE, ANY OF WHOSE CONDITIONS IS TRUE, IS EXECUTED. IF NO WHEN CLAUSE IS SELECTED, THE <EXECUTABLE_STATEMENT> OF THE OTHERWISE CLAUSE, IF ANY, IS EXECUTED. AFTER THE SELECTED <EXECUTABLE_STATEMENT>, IF ANY, IS EXECUTED, CONTROL IS TRANSFERRED TO THE END STATEMENT ASSOCIATED WITH THE SELECT; I.E., UNDER NO CIRCUMSTANCES WILL MORE THAN ONE CASE BE SELECTED.

4. THE <EXECUTABLE_STATEMENT>S MAY BE DO OR SELECT GROUPS OR BEGIN BLOCKS, AS WELL AS SIMPLE STATEMENTS.

5. FORTRAN COMPATIBILITY: THE SELECT STATEMENT CAN BE USED TO SIMULATE A COMPUTED-GOTO AND WILL GENERATE EQUALLY GOOD CODE WHEN USED THIS WAY.

N. $INSERT FILENAME--JUST LIKE FORTRAN (NONSTANDARD)

O. LEAVE [LABEL]; (NONSTANDARD)

1. CAUSES CONTROL TO BE TRANSFERRED TO THE STATEMENT FOLLOWING THE END OF THE SELECTED DO-GROUP--THE INNERMOST, IF "LABEL" IS OMITTED, OR THE ONE WHOSE "DO" IS LABELLED WITH "LABEL", IF SPECIFIED.

2. "LABEL" MUST BE A LABEL CONSTANT; CONTROL CANNOT LEAVE A BEGIN OR PROC BLOCK.

P. %NOLIST; AND %LIST; (NONSTANDARD)

1. %NOLIST TURNS OFF LISTING GENERATION FOR SUBSEQUENT LINES.

2. %LIST RESTORES THE LISTING, IF ONE IS BEING GENERATED. AS MANY %LIST STATEMENTS MUST BE CODED AS NEEDED TO MATCH THE UNTERMINATED %NOLIST STATEMENTS FOR LISTING TO RESUME.

Q. %REPLACE ID BY LEXEME [, ID BY LEXEME [, ...]]; (NONSTANDARD)

1. SUBSEQUENT REFERENCES TO "ID" ARE TREATED AS IF "LEXEME" HAD APPEARED IN THE SOURCE AT THAT SPOT INSTEAD.

2. "LEXEME" IS A SINGLE LEXICAL ITEM (E.G., IDENTIFIER, STRING CONSTANT, DECIMAL CONSTANT, ETC.).

3. "ID" MAY NOT BE REPLACED IN A SUBSEQUENT %REPLACE STATEMENT.

"*" => PLANNED BUT NOT YET AVAILABLE IN PL/P

```
**********************************************************************************************
**********************************************************************************************
**
**
**
**      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     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     CCC     000    BBBB     000    L
**    R   R   C   C   0   0   B   B   0   0   L
**    R   R   C       0   0   B   B   0   0   L
**    RRRR    C       0   0   BBBB    0   0   L
**    R R     C       0   0   B   B   0   0   L
**    R  R    C   C   0   0   B   B   0   0   L    L
**    R   R    CCC     000    BBBB     000    LLLLL
**
**
**
**********************************************************************************************
**********************************************************************************************
```

COBOL FOR REV. 17

DATE: MAY 3, 1979

TO:

FROM:

SUBJECT: COBOL FOR REV. 17

REFERENCE: NONE

ABSTRACT

THIS DOCUMENT DESCRIBES THE NEW FEATURES AND ERROR MESSAGE FOR REV. 17 COBOL.

SORT VERB, PERFORM AFTER AND SEARCH ALL ARE THE NEW VERBS ADDED TO THE COBOL COMPILER FOR REV 17. ALSO THE CROSS-REFERENCE IS NOW AVAILABLE. THERE IS A PROGRAM THAT DEMONSTRATES THIS FEATURE IN THE SAMPLE PROGRAM SECTION.

PERFORM (AFTER)

FORMAT 3

```
                                        <THRU    <
PERFORM  PROCEDURE-NAME-1  [ <           <       PROCEDURE-NAME-2 ]
                             <           <
                             <THROUGH    <

                                                    <IDENTIFIER-2<
                        <IDENTIFIER-1<              <            <
          VARYING       <            <  FROM       <INDEX-NAME-2<
                        <INDEX-NAME-1<              <            <
                                                    <LITERAL-1   <

                <IDENTIFIER-3<
          BY    <            <     UNTIL  CONDITION-1
                <LITERAL-2   <

                                                    <IDENTIFIER-5<
                        <IDENTIFIER-4<              <            <
          AFTER         <            <  FROM       <INDEX-NAME-4<
                        <INDEX-NAME-3<              <LITERAL-3   <

                <IDENTIFIER-6<
          BY    <            <     UNTIL  CONDITION-2] . . .
                <LITERAL-4   <
```

SYNTAX RULES

1. EACH IDENTIFIER REPRESENTS A NUMERIC ELEMENTARY ITEM DESCRIBED IN THE DATA DIVISION.

2. EACH LITERAL REPRESENTS A NUMERIC LITERAL.

3. THE WORDS THRU AND THROUGH ARE EQUIVALENT.

4. IF AN INDEX-NAME IS SPECIFIED IN THE VARYING OR AFTER PHRASE THEN:

   A. THE IDENTIFIER IN THE ASSOCIATED FROM AND BY PHRASES MUST BE AN INTEGER DATA ITEM.

   B. THE LITERAL IN THE ASSOCIATED FROM PHRASE MUST BE A POSITIVE INTEGER.

C. THE LITERAL IN THE ASSOCIATED BY PHRASE MUST BE A NON-ZERO INTEGER.

5. IF AN INDEX-NAME IS SPECIFIED IN THE FROM PHRASE, THEN:

A. THE IDENTIFIER IN THE ASSOCIATED VARYING OR AFTER PHRASE MUST BE AN INTEGER DATA ITEM.

B. THE IDENTIFIER IN THE ASSOCIATED BY PHRASE MUST BE AN INTEGER DATA ITEM.

C. THE LITERAL IN THE ASSOCIATED BY PHRASE MUST BE AN INTEGER.

6. LITERAL IN THE BY PHRASE MUST NOT BE ZERO.

GENERAL

FORMAT 3 IS THE PERFORM...VARYING. THIS VARIATION OF THE PERFORM STATEMENT IS USED TO AUGMENT THE VALUES REFERENCED BY ONE OR MORE IDENTIFIERS OR INDEX-NAMES IN AN ORDERLY FASHION DURING THE EXECUTION OF A PERFORM STATEMENT. IN THE FOLLOWING DISCUSSION, EVERY REFERENCE TO IDENTIFIER AS THE OBJECT OF THE VARYING, AFTER, AND FROM (CURRENT VALUE) PHRASES ALSO REFERS TO INDEX-NAMES. IF INDEX-NAME-1 OR INDEX-NAME-3 IS SPECIFIED, THE VALUE OF THE ASSOCIATED INDEX AT THE BEGINNING OF THE PERFORM STATEMENT MUST BE SET TO AN OCCURRENCE NUMBER OF AN ELEMENT IN THE TABLE. IF INDEX-NAME-2 OR INDEX-NAME-4 IS SPECIFIED, THE VALUE OF THE DATA ITEM REFERENCED BY IDENTIFIER-1 OR IDENTIFIER-4 AT THE BEGINNING OF THE PERFORM STATEMENT MUST BE EQUAL TO AN OCCURRENCE NUMBER OF AN ELEMENT IN A TABLE ASSOCIATED WITH INDEX-NAME-2 OR INDEX-NAME-4. INDEX-NAME-1 OR INDEX-NAME-3 MUST NOT RESULT IN THE ASSOCIATED INDEX BEING SET TO A VALUE OUTSIDE THE RANGE OF THE TABLE ASSOCIATED WITH INDEX-NAME-1 OR INDEX-NAME-3; EXCEPT THAT, AT THE COMPLETION OF THE PERFORM STATEMENT, THE INDEX ASSOCIATED WITH INDEX-NAME-1 MAY CONTAIN A VALUE THAT IS OUTSIDE THE RANGE OF THE ASSOCIATED TABLE BY ONE INCREMENT OR DECREMENT VALUE.

IN FORMAT 3, WHEN ONE IDENTIFIER IS VARIED, IDENTIFIER-1 IS SET TO THE VALUE OF LITERAL-1 OR THE CURRENT VALUE OF IDENTIFIER-2 AT THE POINT OF INITIAL EXECUTION OF THE PERFORM STATEMENT: THEN, IF THE CONDITION OF THE UNTIL CLAUSE IS FALSE, THE SEQUENCE OF PROCEDURES, PROCEDURE-NAME-1 THROUGH PROCEDURE-NAME-2, IS EXECUTED ONCE. THE VALUE OF IDENTIFIER-1 IS AUGMENTED BY THE SPECIFIED INCREMENT OR DECREMENT VALUE (THE VALUE OF IDENTIFIER-3 OR LITERAL-2) AND CONDITION-1 IS EVALUATED AGAIN. THE CYCLE CONTINUES UNTIL THIS CONDITION IS TRUE, AT WHICH POINT CONTROL IS TRANSFERRED TO THE NEXT EXECUTABLE STATEMENT FOLLOWING THE PERFORM STATEMENT. IF CONDITION-1 IS TRUE AT THE BEGINNING OF EXECUTION OF THE PERFORM STATEMENT, CONTROL IS TRANSFERRED TO THE NEXT EXECUTABLE STATEMENT FOLLOWING THE PERFORM STATEMENT.

IN FORMAT 3, WHEN TWO IDENTIFIERS ARE VARIED, IDENTIFIER-1 AND IDENTIFIER-4 ARE SET TO THE CURRENT VALUE OF IDENTIFIER-2 AND IDENTIFIER-5, RESPECTIVELY. AFTER THE IDENTIFIERS HAVE BEEN SET, CONDITION-1 IS EVALUATED: IF TRUE, CONTROL IS TRANSFERRED TO THE NEXT EXECUTABLE STATEMENT; IF FALSE, CONDITION-2 IS EVALUATED. IF CONDITION-2 IS FALSE, PROCEDURE-NAME-1 THROUGH PROCEDURE-NAME-2 IS EXECUTED ONCE, THEN IDENTIFIER-4 IS AUGMENTED BY IDENTIFIER-6 OR LITERAL-4 AND CONDITION-2 IS EVALUATED AGAIN. THIS CYCLE OF EVALUATION AND AUGMENTATION CONTINUES UNTIL THIS CONDITION IS TRUE. WHEN CONDITION-2 IS TRUE, IDENTIFIER-4 IS SET TO THE VALUE OF LITERAL-3 OR THE CURRENT VALUE OF IDENTIFIER-5, IDENTIFIER-1 IS AUGMENTED BY IDENTIFIER-3 AND CONDITION-1 IS REEVALUATED. THE PERFORM STATEMENT IS COMPLETED IF CONDITION-1 IS TRUE; IF NOT, THE CYCLES CONTINUE UNTIL CONDITION-1 IS TRUE.

DURING THE EXECUTION OF THE PROCEDURES ASSOCIATED WITH THE PERFORM STATEMENT, ANY CHANGE TO THE VARYING VARIABLE (IDENTIFIER-1 AND INDEX-NAME-1), THE BY VARIABLE (IDENTIFIER-3), THE AFTER VARIABLE (IDENTIFIER-4 AND INDEX-NAME-3), OR THE FROM VARIABLE (IDENTIFIER-2 AND INDEX-NAME-2) WILL BE TAKEN INTO CONSIDERATION AND WILL AFFECT THE OPERATION OF THE PERFORM STATEMENT.

AT THE TERMINATION OF THE PERFORM STATEMENT, IDENTIFIER-4 CONTAINS THE CURRENT VALUE OF IDENTIFIER-5. IDENTIFIER-1 HAS A VALUE THAT EXCEEDS THE LAST USED SETTING BY AN INCREMENT OR DECREMENT VALUE, UNLESS CONDITION-1 WAS TRUE WHEN THE PERFORM STATEMENT WAS ENTERED, IN WHICH CASE IDENTIFIER-1 CONTAINS THE CURRENT VALUE OF IDENTIFIER-2.

WHEN TWO IDENTIFIERS ARE VARIED, IDENTIFER-4 GOES THROUGH A COMPLETE CYCLE (FROM,BY,UNTIL) EACH TIME IDENTIFIER-1 IS VARIED.

WHEN THE CONTENTS OF THREE OR MORE DATA ITEMS REFERENCED BY IDENTIFIERS ARE VARIED, THE MECHANISM IS THE SAME AS FOR TWO IDENTIFIERS EXCEPT THAT THE DATA ITEM BEING VARIED BY EACH AFTER PHRASE GOES THROUGH A COMPLETE CYCLE EACH TIME THE DATA ITEM BEING VARIED BY THE PRECEDING AFTER PHRASE IS AUGMENTED.

AFTER THE COMPLETION OF A FORMAT 3 PERFORM STATEMENT, EACH DATA ITEM VARIED BY AN AFTER PHRASE CONTAINS THE CURRENT VALUE OF THE DATA ITEM REFERENCED BY THE IDENTIFIER IN THE ASSOCIATED FROM PHRASE. THE DATA ITEM REFERENCED BY IDENTIFIER-1 HAS A VALUE THAT EXCEEDS ITS LAST USED SETTING BY ONE INCREMENT OR DECREMENT VALUE, UNLESS CONDITION-1 IS TRUE WHEN THE PERFORM STATEMENT IS ENTERED, IN WHICH CASE THE DATA ITEM REFERENCED BY IDENTIFIER-1 CONTAINS THE CURRENT VALUE OF THE DATA ITEM REFERENCED BY IDENTIFIER-2.

THE RANGE OF A PERFORM STATEMENT CONSISTS LOGICALLY OF ALL THOSE STATEMENTS THAT ARE EXECUTED AS A RESULT OF EXECUTING THE PERFORM STATEMENT THROUGH EXECUTION OF THE IMPLICIT TRANSFER OF CONTROL TO THE NEXT EXECUTABLE STATEMENT FOLLOWING THE PERFORM STATEMENT. THE RANGE INCLUDES ALL STATEMENTS THAT ARE EXECUTED AS THE RESULT OF A TRANSFER OF CONTROL BY GO TO, PERFORM AND CALL STATEMENTS IN THE RANGE OF THE PERFORM STATEMENT, AS WELL AS ALL STATEMENTS IN DECLARATIVE PROCEDURES

THAT ARE EXECUTED AS A RESULT OF THE EXECUTION OF STATEMENTS IN THE RANGE OF THE PERFORM STATEMENT. THERE IS NO REQUIREMENT FOR THE STATEMENTS IN THE RANGE OF A PERFORM TO APPEAR CONSECUTIVELY IN THE SOURCE PROGRAM.

IF A SEQUENCE OF STATEMENTS REFERRED TO BY A PERFORM STATEMENT INCLUDES ANOTHER PERFORM STATEMENT, THE SEQUENCE OF PROCEDURES ASSOCIATED WITH THE INCLUDED PERFORM MUST ITSELF EITHER BE TOTALLY INCLUDED IN, OR TOTALLY EXCLUDED FROM, THE LOGICAL SEQUENCE REFERRED TO BY THE FIRST PERFORM. THUS, AN ACTIVE PERFORM STATEMENT, WHOSE EXECUTION POINT BEGINS WITHIN THE RANGE OF ANOTHER ACTIVE PERFORM STATEMENT, MUST NOT ALLOW CONTROL TO PASS TO THE EXIT OF THE OTHER ACTIVE PERFORM STATEMENT: FURTHERMORE, TWO OR MORE SUCH ACTIVE PERFORM STATEMENTS MAY NOT HAVE A COMMON EXIT.

*REFER TO SAMPLE PROGRAM LINES 75-85 FOR AN EXAMPLE.

SORT FUNCTION

THE SORT STATEMENT CREATES A SORT FILE BY EXECUTING INPUT PROCEDURES OR BY TRANSFERRING RECORDS FROM ANOTHER FILE, SORTS THE RECORDS IN THE SORT FILE ON A SET OF SPECIFIED KEYS, AND, IN THE FINAL PHASE OF THE SORT OPERATION, MAKES AVAILABLE EACH RECORD FROM THE SORT FILE, IN SORTED ORDER, TO SOME OUTPUT PROCEDURES OR TO AN OUTPUT FILE.

DATA DIVISION FOR THE SORT

FILE SECTION

AN SD FILE DESCRIPTION GIVES INFORMATION ABOUT THE SIZES AND THE NAMES OF THE DATA RECORDS ASSOCIATED WITH THE FILE TO BE SORTED. THERE ARE NO LABEL PROCEDURES WHICH THE USER CAN CONTROL, AND THE RULES FOR BLOCKING AND INTERNAL STORAGE ARE PECULIAR TO THE SORT STATEMENT.

THE SORT FILE DESCRIPTION

THE SORT FILE DESCRIPTION FURNISHES INFORMATION CONCERNING THE PHYSICAL STRUCTURE, IDENTIFICATION, AND RECORD NAMES OF THE FILE TO BE SORTED .

GENERAL FORMAT

```
SD FILE-NAME

     [RECORD CONTAINS [INTEGER-1 TO] INTEGER-2 CHARACTERS]

          < RECORD IS    <
     [DATA < RECORDS ARE < DATA-NAME-1 [, DATA-NAME-2] ...].
```

SYNTAX RULES

1. THE LEVEL INDICATOR SD IDENTIFIES THE BEGINNING OF THE SORT FILE DESCRIPTION AND MUST PRECEDE THE FILE-NAME.

2. THE CLAUSES WHICH FOLLOW THE NAME OF THE FILE ARE OPTIONAL, AND THEIR ORDER OF APPEARANCE IS IMMATERIAL.

3. ONE OR MORE RECORD DESCRIPTION ENTRIES MUST FOLLOW THE SORT FILE DESCRIPTION ENTRY; HOWEVER, NO READ, WRITE, OPEN OR CLOSE STATEMENTS MAY BE EXECUTED FOR THIS FILE.

4. THE FILE MUST BE SPECIFIED IN A SELECT STATEMENT.

PROCEDURE DIVISION SORT STATEMENT

GENERAL FORMAT

```
                        < DESCENDING <
SORT FILE-NAME-1 ON     <            < KEY DATA-NAME-1 ...
                        < ASCENDING  <

<                                       < THRU    <                  <
<                                       <         <                  <
< INPUT PROCEDURE IS SECTION-NAME-1 [   <         < SECTION-NAME-2 ] <
<                                       <         <                  <
<                                       < THROUGH <                  <
< USING FILE-NAME-2 ...                                              <

<                                       < THRU    <                  <
<                                       <         <                  <
< OUTPUT PROCEDURE IS SECTION-NAME-3 [  <         < SECTION-NAME-4 ] <
<                                       <         <                  <
<                                       < THROUGH <                  <
< GIVING FILE-NAME-3 ....                                            <
```

SYNTAX RULES

1. SORT STATEMENTS MAY APPEAR ANYWHERE EXCEPT IN THE DECLARATIVES PORTION OF THE PROCEDURE DIVISION OR IN AN INPUT OR OUTPUT PROCEDURE ASSOCIATED WITH A SORT STATEMENT.

2. FILE-NAME-1 MUST BE DESCRIBED IN A SORT FILE DESCRIPTION ENTRY IN THE DATA DIVISION.

3. IF THE USING PHRASE IS SPECIFIED AND THE FILE REFERENCED BY FILE-NAME-1 CONTAINS VARIABLE-LENGTH RECORDS, THE SIZE OF THE RECORDS CONTAINED IN THE FILE REFERENCED BY FILE-NAME-2 MUST NOT BE LESS THAN THE SMALLEST RECORD NOR LARGER THAN THE LARGEST RECORD DESCRIBED FOR FILE-NAME-1. IF THE FILE REFERENCED BY FILE-NAME-1 CONTAINS FIXED-LENGTH RECORDS, THE SIZE OF THE RECORDS CONTAINED IN THE FILE REFERENCED BY FILE-NAME-2 MUST NOT BE LARGER THAN THE LARGEST RECORD DESCRIBED FOR THE FILE REFERENCED BY FILE-NAME-1.

4. DATA-NAME-1 IS A KEY DATA-NAME. KEY DATA-NAMES ARE SUBJECT TO THE FOLLOWING RULES:

   A. THE DATA ITEMS IDENTIFIED BY KEY DATA-NAMES MUST BE DESCRIBED IN RECORDS ASSOCIATED WITH FILE-NAME-1.

   B. KEY DATA-NAMES MAY BE QUALIFIED.

   C. THE DATA ITEMS IDENTIFIED BY KEY DATA-NAMES MUST NOT BE VARIABLE-LENGTH DATA ITEMS, NOR MAY THEY NAME GROUP ITEMS WHICH CONTAIN VARIABLE-OCCURRENCE DATA ITEMS.

   D. IF FILE-NAME-1 HAS MORE THAN ONE RECORD DESCRIPTION, THEN THE DATA ITEMS IDENTIFIED BY KEY DATA-NAMES NEED BE DESCRIBED IN ONLY ONE OF THE RECORD DESCRIPTIONS. THE SAME CHARACTER POSITIONS WHICH ARE REFERENCED BY A KEY DATA-NAME IN ONE RECORD DESCRIPTION ENTRY ARE TAKEN AS THE KEY IN ALL RECORDS OF THE FILE.

   E. NONE OF THE DATA ITEMS IDENTIFIED BY KEY DATA-NAMES CAN BE DESCRIBED BY AN ENTRY WHICH EITHER CONTAINS AN OCCURS CLAUSE OR IS SUBORDINATE TO AN ENTRY WHICH CONTAINS AN OCCURS CLAUSE.

5. SECTION-NAME-1 REPRESENTS THE NAME OF AN INPUT PROCEDURE. SECTION-NAME-3 REPPESENTS THE NAME OF AN OUTPUT PROCEDURE.

6. THE WORDS THRU AND THROUGH ARE EQUIVALENT.

7. IN THE DATA DIVISION, FILE-NAME-2 AND FILE-NAME-3 MUST BE DESCRIBED IN A FILE DESCRIPTION ENTRY, NOT IN A SORT FILE DESCRIPTION ENTRY.

8. IF THE GIVING PHRASE IS SPECIFIED AND THE FILE REFERENCED BY FILE-NAME-3 CONTAINS VARIABLE-LENGTH RECORDS, THE SIZE OF THE RECORDS CONTANED IN THE FILE REFERENCED BY FILE-NAME-1 MUST NOT BE LESS THAN THE SMALLEST RECORD NOR LARGER THAN THE LARGEST RECORD DESCRIBED FOR FILE-NAME-3. IF THE FILE REFERENCED BY FILE-NAME-3 CONTAINS FIXED-LENGTH RECORDS, THE SIZE OF THE RECORDS CONTAINED IN THE FILE REFERECED BY FILE-NAME-1 MUST NOT BE LARGER THAN THE LARGEST RECORD DESCRIBED FOR THE FILE REFERENCED BY FILE-NAME-3.

GENERAL RULES

1. IF THE FILE REFERENCED BY FILE-NAME-1 CONTAINS ONLY FIXED-LENGTH RECORDS, ANY RECORD IN THE FILE REFERENCED BY FILE-NAME-2 CONTAINING FEWER CHARACTER POSITIONS THAN THAT FIXED LENGTH IS SPACE-FILLED ON THE RIGHT BEGINNING WITH THE FIRST CHARACTER POSITION AFTER THE LAST CHARACTER IN THE RECORD WHEN THAT RECORD IS RELEASED TO THE FILE REFERENCED BY FILE-NAME-1.

2. THE DATA-NAMES FOLLOWING THE WORK KEY ARE LISTED FROM LEFT TO RIGHT IN THE SORT STATEMENT IN ORDER OF DECREASING SIGNIFICANCE WITHOUT REGARD TO HOW THEY ARE DIVIDED INTO KEY PHRASES. THE LEFTMOST DATA-NAME IS THE MAJOR KEY, THE NEXT DATA-NAME IS THE NEXT MOST SIGNIFICANT KEY, ETC.

   A. WHEN THE ASCENDING PHRASE IS SPECIFIED, THE SORTED SEQUENCE WILL BE FROM THE LOWEST VALUE OF THE CONTENTS OF THE DATA ITEMS IDENTIFIED BY THE KEY DATA-NAMES TO THE HIGHEST VALUE, ACCORDING TO THE RULES FOR COMPARISON OF OPERANDS IN A RELATION CONDITION.

   B. WHEN THE DESCENDING PHRASE IS SPECIFIED, THE SORTED SEQUENCE WILL BE FROM THE HIGHEST VALUE OF THE CONTENTS OF THE DATA ITEMS IDENTIFIED BY THE KEY DATA-NAMES TO THE LOWEST VALUE, ACCORDING TO THE RULES FOR COMPARISON OF OPERANDS IN A RELATION CONDITION.

3. IF THE CONTENTS OF ALL THE KEY DATA ITEMS ASSOCIATED WITH ONE DATA RECORD ARE EQUAL TO THE CONTENTS OF THE CORRESPONDING KEY DATA ITEMS ASSOCIATED WITH ONE OR MORE OTHER DATA RECORDS, THEN THE ORDER OF RETURN OF THESE RECORDS IS UNDEFINED.

4. THE INPUT PROCEDURE MUST CONSIST OF ONE OR MORE SECTIONS THAT APPEAR CONTIGUOUSLY IN A SOURCE PROGRAM AND DO NOT FORM A PART OF ANY OUTPUT PROCEDURE. IN ORDER TO TRANSFER RECORDS TO THE FILE REFERENCED BY FILE-NAME-1, THE INPUT PROCEDURE MUST INCLUDE THE EXECUTION OF AT LEAST ONE RELEASE STATEMENT. CONTROL MUST NOT BE PASSED TO THE INPUT PROCEDURE EXCEPT WHEN A RELATED SORT STATEMENT IS BEING EXECUTED. THE INPUT PROCEDURE CAN INCLUDE ANY PROCEDURES NEEDED TO SELECT, CREATE, OR MODIFY RECORDS. THE RESTRICTIONS ON THE PROCEDURAL STATEMENTS WITHIN THE INPUT PROCEDURE ARE AS FOLLOWS:

   A. THE INPUT PROCEDURE MUST NOT CONTAIN ANY SORT STATEMENTS.

   B. THE INPUT PROCEDURE MUST NOT CONTAIN ANY EXPLICIT TRANSFERS OF CONTROL TO POINTS OUTSIDE THE INPUT PROCEDURE; GO TO, AND PERFORM STATEMENTS IN THE INPUT PROCEDURE ARE NOT PERMITTED TO REFER TO PROCEDURE-NAMES OUTSIDE THE INPUT PROCEDURE. COBOL STATEMENTS ARE ALLOWED THAT WILL CAUSE AN IMPLIED TRANSFER OF CONTROL TO DECLARATIVES.

   C. THE REMAINDER OF THE PROCEDURE DIVISION MUST NOT CONTAIN ANY TRANSFERS OF CONTROL TO POINTS INSIDE THE INPUT PROCEDURE; GO TO AND PERFORM STATEMENTS IN THE REMAINDER OF THE PROCEDURE DIVISION MUST NOT REFER TO PROCEDURE-NAMES WITHIN THE INPUT PROCEDURE.

5. IF AN INPUT PROCEDURE IS SPECIFIED, CONTROL IS PASSED TO THE INPUT PROCEDURE BEFORE THE FILE REFERENCED BY FILE-NAME-1 IS SEQUENCED BY THE SORT STATEMENT. BEFORE CONTROL PASSES THE LAST STATEMENT IN THE INPUT PROCEDURE, THE FILE REFERENCED BY FILE-NAME-3 MUST NOT BE OPEN. THE COMPILER INSERTS A RETURN MECHANISM AT THE END OF THE LAST SECTION IN THE INPUT PROCEDURE AND WHEN CONTROL PASSES THE LAST STATEMENT IN THE INPUT PROCEDURE, THE RECORDS THAT HAVE BEEN RELEASED TO THE FILE REFERENCED BY FILE-NAME-1 ARE SORTED.

6. DURING THE EXECUTION OF THE INPUT PROCEDURE, THE OUTPUT PROCEDURE OR ANY USE AFTER EXCEPTION PROCEDURE, NO STATEMENT MANIPULATING THE FILES REFERENCED BY, OR ACCESSING THE RECORD AREAS ASSOCIATED WITH FILE-NAME-2 OR FILE-NAME-3 MAY BE EXECUTED.

7. IF THE USING PHRASE IS SPECIFIED, ALL THE RECORDS IN THE FILE(S) REFERENCED BY FILE-NAME-2 ARE TRANSFERRED TO THE FILE REFERENCED BY FILE-NAME-1. AT THE TIME OF EXECUTION OF THE SORT STATEMENT, THE FILE REFERENCED BY FILE-NAME-2 MUST NOT BE IN THE OPEN MODE. FOR EACH OF THE FILES REFERENCED BY FILE-NAME-2 THE EXECUTION OF THE SORT STATEMENT CAUSES THE FOLLOWING ACTIONS TO BE TAKEN:

   A. THE PROCESSING OF THE FILE IS INITIATED. THE INITIATION IS PERFORMED AS IF AN OPEN STATEMENT WITH THE INPUT PHRASE HAD BEEN EXECUTED.

   B. THE LOGICAL RECORDS ARE OBTAINED AND RELEASED TO THE SORT OPERATION. EACH RECORD IS OBTAINED AS IF A READ STATEMENT WITH THE NEXT AND THE AT END PHRASES HAD BEEN EXECUTED.

   C. THE PROCESSING OF THE FILE IS TERMINATED. THE TERMINATION IS PERFORMED AS IF A CLOSE STATEMENT WITHOUT OPTIONAL PHRASES HAD BEEN EXECUTED.

   THESE IMPLICIT FUNCTIONS ARE PERFORMED SUCH THAT ANY ASSOCIATED USE PROCEDURES ARE EXECUTED.

8. THE OUTPUT PROCEDURE MUST CONSIST OF ONE OR MORE SECTIONS THAT APPEAR CONTIGUOUSLY IN A SOURCE PROGRAM AND DO NOT FORM PART OF ANY INPUT PROCEDURE. IN ORDER TO MAKE SORTED RECORDS AVAILABLE FOR PROCESSING, THE OUTPUT PROCEDURE MUST INCLUDE THE EXECUTION OF AT LEAST ONE RETURN STATEMENT. CONTROL MUST NOT BE PASSED TO THE OUTPUT PROCEDURE EXCEPT WHEN A RELATED SORT STATEMENT IS BEING EXECUTED. THE OUTPUT PROCEDURE MAY CONSIST OF ANY PROCEDURES NEEDED TO SELECT, MODIFY OR COPY THE RECORDS THAT ARE BEING RETURNED, ONE AT A TIME IN SORTED ORDER, FROM THE SORT FILE. THE RESTRICTIONS ON THE PROCEDURAL STATEMENTS WITHIN THE OUTPUT PROCEDURE ARE AS FOLLOWS:

   A. THE OUTPUT PROCEDURE MUST NOT CONTAIN ANY SORT STATEMENTS.

B. THE OUTPUT PROCEDURE MUST NOT CONTAIN ANY EXPLICIT TRANSFERS OF CONTROL TO POINTS OUTSIDE THE OUTPUT PROCEDURE: GO TO, AND PERFORM STATEMENTS IN THE OUTPUT PROCEDURE ARE NOT PERMITTED TO REFER TO PROCEDURE-NAMES OUTSIDE THE OUTPUT PROCEDURE. COBOL STATEMENTS ARE ALLOWED THAT WILL CAUSE AN IMPLIED TRANSFER OF CONTROL TO DECLARATIVES.

C. THE REMAINDER OF THE PROCEDURE DIVISION MUST NOT CONTAIN ANY TRANSFERS OF CONTROL TO POINTS INSIDE THE OUTPUT PROCEDURE; GO TO AND PERFORM STATEMENTS IN THE REMAINDER OF THE PROCEDURE DIVISION ARE NOT PERMITTED TO REFER TO PROCEDURE-NAMES WITHIN THE OUTPUT PROCEDURE.

9. IF AN OUTPUT PROCEDURE IS SPECIFIED, CONTROL PASSES TO IT AFTER THE FILE REFERENCED BY FILE-NAME-1 HAS BEEN SEQUENCED BY THE SORT STATEMENT. THE FILE REFERENCED BY FILE-NAME-2 IS NOT IN THE OPEN MODE. THE COMPILER INSERTS A RETURN MECHANISM AT THE END OF THE LAST SECTION IN THE OUTPUT PROCEDURE AND WHEN CONTROL PASSES THE LAST STATEMENT IN THE OUTPUT PROCEDURE, THE RETURN MECHANISM PROVIDES FOR TERMINATION OF THE SORT AND THEN PASSES CONTROL TO THE NEXT EXECUTABLE STATEMENT AFTER THE SORT STATEMENT. BEFORE ENTERING THE OUTPUT PROCEDURE, THE SORT PROCEDURE REACHES A POINT AT WHICH IT CAN SELECT THE NEXT RECORD IN SORTED ORDER WHEN REQUESTED. THE RETURN STATEMENTS IN THE OUTPUT PROCEDURE ARE THE REQUESTS FOR THE NEXT RECORD.

10. IF THE GIVING PHRASE IS SPECIFIED, ALL THE SORTED RECORDS ARE WRITTEN ON THE FILE REFERENCED BY FILE-NAME-3 AS THE IMPLIED OUTPUT PROCEDURE FOR THE SORT STATEMENT. AT THE TIME OF THE EXECUTION OF THE SORT STATEMENT, THE FILE REFERENCED BY FILE-NAME-3 MUST NOT BE OPEN. FOR EACH OF THE FILES REFERENCED BY FILE-NAME-3, THE EXECUTION OF THE SORT STATEMENT CAUSES THE FOLLOWING ACTIONS TO BE TAKEN:

A. THE PROCESSING OF THE FILE IS INITIATED. THE INITIATION IS PERFORMED AS IF AN OPEN STATEMENT WITH THE OUTPUT PHRASE HAD BEEN EXECUTED.

B. THE SORTED LOGICAL RECORDS ARE RETURNED AND WRITTEN ONTO THE FILE. THE RECORDS ARE WRITTEN AS IF A WRITE STATEMENT WITHOUT ANY OPTIONAL PHRASES HAD BEEN EXECUTED.

C. THE PROCESSING OF THE FILE IS TERMINATED. THE TERMINATION IS PERFORMED AS IF A CLOSE STATEMENT WITHOUT OPTIONAL PHRASES HAD BEEN EXECUTED.

11. IF THE FILE REFERENCED BY FILE-NAME-3 CONTAINS ONLY FIXED-LENGTH RECORDS, ANY RECORD IN THE FILE REFERENCED BY FILE-NAME-1 CONTAINING FEWER CHARACTER POSITIONS THAN THAT FIXED LENGTH IS SPACE-FILLED ON THE RIGHT BEGINNING WITH THE FIRST CHARACTER POSITION AFTER THE LAST CHARACTER IN THE RECORD WHEN THAT RECORD IS RETURNED FROM THE FILE REFERENCED BY FILE-NAME-3.

*REFER TO SAMPLE PROGRAM SECTION FOR AN EXAMPLE OF A SORT PROGRAM.

THE OCCURS CLAUSE

FUNCTION

THE OCCURS CLAUSE ELIMINATES THE NEED FOR SEPARATE ENTRIES FOR REPEATED DATA ITEMS AND SUPPLIES INFORMATION REQUIRED FOR THE APPLICATION OF SUBSCRIPTS OR INDICES.

GENERAL FORMAT

```
OCCURS INTEGER-2 TIMES

      < ASCENDING  <
  [   <            <   KEY IS DATA-NAME-2 [ , DATA-NAME-3 ] ... ]
      < DESCENDING <

  [ INDEXED BY INDEX-NAME-1 [ , INDEX-NAME-2 ] ... ]
```

SYNTAX RULES

1. INTEGER-1 MUST BE GREATER THAN ONE.

2. DATA-NAME-1 MUST DESCRIBE AN ELEMENTARY UNSIGNED INTEGER.

3. DATA-NAME-1, DATA-NAME-2, DATA-NAME-3,...MAY BE QUALIFIED.

4. DATA-NAME-2 MUST EITHER BE THE NAME OF THE ENTRY CONTAINING THE OCCURS CLAUSE OR THE NAME OF AN ENTRY SUBORDINATE TO THE ENTRY CONTAINING THE OCCURS CLAUSE.

5. DATA-NAME-3,...MUST BE THE NAME OF AN ENTRY SUBORDINATE TO THE GROUP ITEM WHICH IS THE SUBJECT OF THIS ENTRY.

6. ITEMS IDENTIFIED BY THE DATA-NAMES IN THE KEY IS PHRASE MUST NOT CONTAIN AN OCCURS CLAUSE EXCEPT WHERE DATA-NAME-2 IS THE SUBJECT OF THE ENTRY.

7. THERE MUST NOT BE AN ENTRY THAT CONTAINS AN OCCURS CLAUSE BETWEEN THE ITEMS IDENTIFIED BY THE DATA-NAMES IN THE KEY IS PHRASE AND THE SUBJECT OF THE ENTRY.

8. DATA-NAME-2, DATA-NAME-3,...MUST BE SPECIFIED WITHOUT THE SUBSCIPTING OR INDEXING NORMALLY REQUIRED.

9. AN INDEXED BY PHRASE IS REQUIRED IF THE SUBJECT OF THIS ENTRY, OR AN ENTRY SUBORDINATE TO THIS ENTRY, IS TO BE REFERENCED BY INDEXING. THE INDEX-NAME IDENTIFIED BY THIS CLAUSE IS NOT DEFINED ELSEWHERE SINCE ITS ALLOCATION AND FORMAT ARE DEPENDENT ON THE HARDWARE AND, NOT BEING DATA, CANNOT BE ASSOCIATED WITH ANY DATA HIERARCY.

10. THE OCCURS CLAUSE MUST NOT BE SPECIFIED IN A DATA DESCRIPTION ENTRY THAT HAS A LEVEL-NUMBER OF 01, 66 OR 88.

11. INDEX-NAME-1, INDEX-NAME-2,...MUST BE UNIQUE WORDS WITHIN THE PROGRAM.

GENERAL RULES

1. THE OCCURS CLAUSE IS USED IN DEFINING TABLES AND OTHER HOMOGENEOUS SETS OF REPEATED DATA ITEMS.

2. EXCEPT FOR THE OCCURS CLAUSE ITSELF, ALL DATA DESCRIPTION CLAUSES ASSOCIATED WITH AN ITEM WHOSE DESCRIPTION INCLUDES AN OCCURS CLAUSE APPLY TO EACH OCCURRENCE OF THE ITEM DESCRIBED.

3. THE NUMBER OF OCCURRENCES OF THE SUBJECT ENTRY IS DEFINED AS FOLLOWS:

   THE VALUE OF INTEGER-2 REPRESENTS THE EXACT NUMBER OF OCCURRENCES.

4. THE KEY IS PHRASE IS USED TO INDICATE THAT THE REPEATED DATA IS ARRRANGED IN ASCENDING OR DESCENDING ORDER ACCORDING TO THE VALUES CONTAINED IN DATA-NAME-2, DATA-NAME-3,.... THE ASCENDING OR DESCENDING ORDER IS DETERMINED ACCORDING TO THE RULES FOR THE COMPARISON OF OPERANDS.

THE SEARCH ALL STATEMENT

FUNCTION

THE SEARCH STATEMENT IS USED TO SEARCH A TABLE FOR A TABLE ELEMENT THAT SATISFIES THE SPECIFIED CONDITION AND TO ADJUST THE ASSOCIATED INDEX-NAM TO INDICATE THAT TABLE ELEMENT.

GENERAL FORMAT

```
SEARCH ALL IDENTIFIER-1          [ AT END IMPERATIVE-STATEMENT-1 ]

          <                    < EQUALS          <     <IDENTIFIER-2  < <
WHEN      < DATA-NAME-1        < IS EQUAL TO     <     <LITERAL-1     < <
          <                    < IS =            <     <              < <
          <                                                           <
          < CONDITION-NAME-1

          <                    < EQUALS          <  <IDENTIFIER-3  < <
[ AND     < DATA-NAME-2        < IS EQUAL TO     <  <LITERAL-2     < < ]
          <                    < IS =            <  <              < <
          <                                                        <
          < CONDITION-NAME-2                                       <

     < IMPERATIVE-STATEMENT-2 <
     < NEXT SENTENCE          <
```

SYNTAX RULES

1. IDENTIFIER-1 MUST NOT BE SUBSCRIPTED OR INDEXED BUT ITS DESCRIPTION MUST CONTAIN AN OCCURS CLAUSE AND AN INDEXED BY CLAUSE. THE DESCRIPTION OF IDENTIFIER-1 MUST ALSO CONTAIN THE KEY IS PHRASE IN ITS OCCURS CLAUSE.

2. IDENTIFIER-2, WHEN SPECIFIED, MUST BE DESCRIBED AS USAGE IS INDEX OR AS A NUMERIC ELEMENTARY DATA ITEM WITHOUT ANY POSITIONS TO THE RIGHT OF THE ASSUMED DECIMAL POINT.

3. ALL REFERENCED CONDITION-NAMES MUST BE DEFINED AS HAVING ONLY A SINGLE VALUE. THE DATA-NAME ASSOCIATED WITH A CONDITION-NAME MUST APPEAR IN THE KEY CLAUSE OF IDENTIFIER-1. EACH DATA-NAME-1, DATA-NAME-2 MAY BE QUALIFIED. EACH DATA-NAME-1, DATA-NAME-2 MUST BE INDEXED BY THE FIRST INDEX-NAME ASSOCIATED WITH IDENTIFIER-1 ALONG WITH OTHER INDICES OR LITERALS AS REQUIRED.

4. WHEN A DATA-NAME IN THE KEY CLAUSE OF IDENTIFIER-1 IS REFERENCED, OR WHEN A CONDITION-NAME ASSOCIATED WITH A DATA-NAME IN THE KEY CLAUSE OF IDENTIFIER-1 IS REFERENCED, ALL PRECEDING DATA-NAMES IN THE KEY CLAUSE OF IDENTIFIER-1 OR THEIR ASSOCIATED CONDITION-NAMES MUST ALSO BE REFERENCED.

GENERAL RULES

1. IN A SEARCH STATEMENT, THE RESULTS OF THE SEARCH ALL OPERATION ARE PREDICTABLE ONLY WHEN:

   A. THE DATA IN THE TABLE IS ORDERED IN THE SAME MANNER AS DESCRIBED IN THE ASCENDING/DESCENDING KEY CLAUSE ASSOCIATED WITH THE DESCRIPTION OF IDENTIFIER-1, AND

   B. THE CONTENTS OF THE KEY(S) REFERENCED IN THE WHEN CLAUSE ARE SUFFICIENT TO IDENTIFY A UNIQUE TABLE ELEMENT.

2. WHEN THE SEARCH ALL IS USED, A NONSERIAL TYPE OF SEARCH OPERATION MAY TAKE PLACE, THE INITIAL SETTING OF THE INDEX-NAME FOR IDENTIFIER-1 IS IGNORED AND ITS SETTING IS VARIED DURING THE SEARCH OPERATION, WITH THE RESTRICTION THAT AT NO TIME IS IT SET TO A VALUE THAT EXCEEDS THE VALUE WHICH CORRESPONDS TO THE LAST ELEMENT OF THE TABLE, OR THAT IS LESS THAN THE VALUE THAT CORRESPONDS TO THE FIRST ELEMENT OF THE TABLE. THE LENGTH OF THE TABLE IS DISCUSSED IN THE OCCURS CLAUSE. (THE OCCURS CLAUSE.) IF ANY OF THE CONDITIONS SPECIFIED IN THE WHEN CLAUSE CANNOT BE SATISFIED FOR ANY SETTING OF THE INDEX WITHIN THE PERMITTED RANGE, CONTROL IS PASSED TO IMPERATIVE- STATEMENT-1 OF THE AT END PHRASE, WHEN SPECIFIED, OR TO THE NEXT EXECUTABLE SENTENCE WHEN THIS PHRASE IS NOT SPECIFIED, IN EITHER CASE THE FINAL SETTING OF THE INDEX IS NOT PREDICTABLE. IF ALL THE CONDITIONS CAN BE SATISFIED, THE INDEX INDICATES AN OCCURRENCE THAT ALLOWS THE CONDITIONS TO BE SATISFIED, AND CONTROL PASSES TO IMPERATIVE-STATEMENT-2.

3. AFTER EXECUTION OF IMPERATIVE-STATEMENT-1, IMPERATIVE-STATEMENT-2, OR IMPERATIVE-STATEMENT-3, THAT DOES NOT TERMINATE WITH A GO TO STATEMENT, CONTROL PASSES TO THE NEXT EXECUTABLE SENTENCE.

4. THE INDEX-NAME THAT IS USED FOR THE SEARCH OPERATION IS THE FIRST (OR ONLY) INDEX-NAME THAT APPEARS IN THE INDEXED BY CLAUSE OF IDENTIFIER-1. ANY OTHER INDEX-NAMES FOR IDENTIFIER-1 REMAIN UNCHANGED.

5. IF IDENTIFIER-1 IS A DATA-ITEM SUBORDINATE TO A DATA ITEM THAT CONTAINS AN OCCURS CLAUSE, ONLY THE SETTING OF THE INDEX-NAME ASSOCIATED WITH IDENTIFIER-1 (AND THE DATA ITEM, IDENTIFIER-2 OR INDEX-NAME-1, IF PRESENT) IS MODIFIED BY THE EXECUTION OF THE SEARCH STATEMENT. TO SEARCH A MULTI-DIMENSIONAL TABLE IT IS NECESSARY TO EXECUTE A SEARCH STATEMENT SEVERAL TIMES. PRIOR TO EACH EXECUTION OF A SEARCH STATEMENT, SET STATEMENTS MUST BE EXECUTED WHENEVER INDEX-NAMES MUST BE ADJUSTED TO APPROPRIATE SETTINGS.

*REFER TO SAMPLE PROGRAM LINES 57-61 AND 143-146 FOR EXAMPLE.

ERROR MESSAGE

THE NEW ERROR MESSAGE FOR THIS REVISION OF THE COMPILER.

: ERROR IN USING SORT, RELEASE, RETURN

THIS ERROR IS GENERATED FOR THE FOLLOWING REASONS:

- NO SD DEFINED FOR THE SORT-FILE
- NO SELECT CLAUSE FOR THE SORT-FILE
- SORT KEYS NOT IN SD DESCRIPTION
- RELEASE NOT USED IN THE INPUT PROCEDURE
- RETURN NOT USED IN THE OUTPUT PROCEDURE

CROSS REFERENCE

THE CROSS REFERENCE HAS TWO COMPILE-TIME OPTIONS -XREF OR -NOXREF. THE DEFAULT IS NO CROSS REFERENCE, (I.E. -NOXREF), IF THIS DEFAULT MUST BE CHANGED, USE THE FOLLOWING PROCEDURE:

```
REST COBOL
SAVE COBOL 3/4077
```

THIS WILL SET THE CROSS REFERENCE FLAG TO ON.

SAMPLE PROGRAM

THE SAMPLE PROGRAM WHICH FOLLOWS HAS AN EXAMPLE OF A CROSS REFERENCE. THE 'D' FOLLOWING THE LINE NUMBER IN THE CROSS REFERENCE LISTING AT THE END OF THE PROGRAM SPECIFIES THAT THIS IS WHERE A PARAGRAPH OR SECTION NAME IS DEFINED IN THE PROCEDURE DIVISION.

```
OK, COBOL SORTIT  -XREF

REV 17.0  COBOL      SOURCE FILE:  SAMPLE.SORT                    04/23/79  11:33
(0001)          IDENTIFICATION DIVISION.
(0002)          PROGRAM-ID.  SORTIT.
(0003)          ENVIRONMENT DIVISION.
(0004)          CONFIGURATION SECTION.
(0005)          SOURCE-COMPUTER. PRIME.
(0006)          OBJECT-COMPUTER. PRIME
(0007)          INPUT-OUTPUT SECTION.
(0008)          FILE-CONTROL.
(0009)              SELECT NET-FILE-IN ASSIGN TO PFMS.
(0010)              SELECT NET-FILE-OUT ASSIGN TO PFMS.
(0011)              SELECT NET-FILE-WORK ASSIGN TO PFMS.
(0012)          DATA DIVISION.
(0013)          FILE SECTION.
(0014)          SD  NET-FILE-WORK.
(0015)          01  SALES-RECORDS.
(0016)              05  EMPL-NO              PIC 9(6).
(0017)              05  DEPT                 PIC 99.
(0018)              05  NET-SALES            PIC 9(7)V99.
(0019)              05  NAME-ADR             PIC X(61).
(0020)              05  MONTH                PIC XX.
(0021)          FD  NET-FILE-IN
(0022)              LABEL RECORDS ARE STANDARD
(0023)              VALUE OF FILE-ID 'FILEIN'.
(0024)          01  NET-CARD-IN.
(0025)              05  EMPL-NO-IN           PIC 9(6).
(0026)              05  DEPT-IN              PIC 99.
(0027)                  88  OFF-SITE-LOCATION   VALUE 7, 6.
(0028)              05  NET-SALES-IN         PIC 9(7)V99.
(0029)              05  NAME-ADDR-IN         PIC X(61).
(0030)              05  MONTH-IN             PIC 99.
(0031)          FD  NET-FILE-OUT
(0032)              LABEL RECORDS ARE STANDARD
(0033)              VALUE OF FILE-ID 'FILEOUT'.
(0034)          01  NET-CARD-OUT.
(0035)              05  EMPL-NO-OUT          PIC 9(6).
(0036)              05  DEPT-OUT             PIC 99.
(0037)              05  NET-SALES-OUT        PIC 9(7)V99.
(0038)              05  NAME-ADDR-OUT        PIC X(61).
(0039)              05  MONTH-OUT            PIC 99.
(0040)          WORKING-STORAGE SECTION.
(0041)          77  SUM-DEPT             PIC S9(14)V99 VALUE ZEROS.
(0042)          01  MONTH-ACCEPT         PIC 99 VALUE ZERO.
(0043)              88  VALID-MONTH     VALUE 01 THRU 12.
(0044)          01  TABLE-VALUE.
(0045)              02  FILLER           PIC X(5) VALUE '01JAN'.
(0046)              02  FILLER           PIC X(5) VALUE '02FEB'.
(0047)              02  FILLER           PIC X(5) VALUE '03MAR'.
(0048)              02  FILLER           PIC X(5) VALUE '04APR'.
```

```
(0049)              02  FILLER              PIC X(5) VALUE '05MAY'.
(0050)              02  FILLER              PIC X(5) VALUE '06JUN'.
(0051)              02  FILLER              PIC X(5) VALUE '07JUL'.
(0052)              02  FILLER              PIC X(5) VALUE '08AUG'.
(0053)              02  FILLER              PIC X(5) VALUE '09SEP'.
(0054)              02  FILLER              PIC X(5) VALUE '10OCT'.
(0055)              02  FILLER              PIC X(5) VALUE '11NOV'.
(0056)              02  FILLER              PIC X(5) VALUE '12DEC'.
(0057)          01  MONTH-TABLE  REDEFINES  TABLE-VALUE.
(0058)              02  MON-TAB  OCCURS 12 TIMES INDEXED BY INDX
(0059)                  ASCENDING KEY MONTH-NO.
(0060)                  03  MONTH-NO    PIC 99.
(0061)                  03  MONTH-VALUE PIC XXX.
(0062)          01  TABLE-AREA.
(0063)              03  SITE    OCCURS 2 TIMES INDEXED BY INDX1.
(0064)                05  MONTHS    OCCURS 12 TIMES INDEXED BY INDX2.
(0065)                  07  DEPT-TOTAL    OCCURS 7 TIMES INDEXED BY INDX3
(0066)                                     PIC S9(14)V99 COMP-3.
(0067)          *
(0068)          01  DISPLAY-TOTALS.
(0069)              02  FILLER              PIC XX VALUE SPACE.
(0070)              02  PRINT-MONTH         PIC XXX VALUE SPACE.
(0071)              02  FILLER              PIC X(16) VALUE '  TOTAL SALES = '.
(0072)              02  PRINT-SUM           PIC ZZ,ZZZ,ZZZ,ZZZ,ZZZ.99-.
(0073)          PROCEDURE DIVISION.
(0074)          START-PARA.
(0075)              PERFORM INT-PARA
(0076)                  VARYING INDX1 FROM 1 BY 1
(0077)                    UNTIL INDX1 > 2
(0078)                  AFTER INDX2 FROM 1 BY 1
(0079)                    UNTIL INDX2 > 12
(0080)                  AFTER INDX3 FROM 1 BY 1
(0081)                    UNTIL INDX3 > 7.
(0082)              GO TO SORT-PARA.
(0083)          *
(0084)           INT-PARA.
(0085)              MOVE ZEROS TO DEPT-TOTAL(INDX1, INDX2, INDX3).
(0086)          *
(0087)           SORT-PARA.
(0088)              SORT NET-FILE-WORK
(0089)                  ASCENDING KEY DEPT
(0090)                  DESCENDING KEY NET-SALES
(0091)                  INPUT PROCEDURE SCREEN-DEPT
(0092)                  GIVING NET-FILE-OUT.
(0093)              GO TO GET-TOTALS.
(0094)
(0095)          SCREEN-DEPT SECTION.
(0096)          S-DEPT1.
(0097)              OPEN INPUT NET-FILE-IN.
(0098)          S-DEPT2.
(0099)              READ NET-FILE-IN
(0100)                  AT END GO TO S-DEPT-FINAL.
(0101)              SET INDX1 TO 2.
```

```
(0102)              IF NOT OFF-SITE-LOCATION
(0103)                 MOVE NET-CARD-IN TO SALES-RECORDS
(0104)                 RELEASE SALES-RECORDS
(0105)                 SET INDX1 TO 1.
(0106)              SET INDX2 TO MONTH-IN.
(0107)              SET INDX3 TO DEPT-IN.
(0108)              ADD NET-SALES-IN TO DEPT-TOTAL (INDX1, INDX2, INDX3).
(0109)              GO TO S-DEPT2.
(0110)          S-DEPT-FINAL.
(0111)              CLOSE NET-FILE-IN.
(0112)          S-DEPT-END.
(0113)              EXIT.
(0114)          *
(0115)          GET-TOTALS SECTION.
(0116)          GET-TOTAL.
(0117)              DISPLAY 'ENTER MONTH XX (01-12) OR ENTER 99 TO QUIT'.
(0118)              ACCEPT MONTH-ACCEPT.
(0119)              IF MONTH-ACCEPT = 99
(0120)                  GO TO DONE-PARA.
(0121)              IF  NOT VALID-MONTH
(0122)                  GO TO GET-TOTAL.
(0123)              PERFORM FIND-MONTH.
(0124)              SET INDX1 TO 1.
(0125)              SET INDX2 TO MONTH-ACCEPT.
(0126)              DISPLAY 'IN STATE'.
(0127)          GET-NEXT.
(0128)              SET INDX3 TO 1.
(0129)              PERFORM ADD-TOTALS 7 TIMES.
(0130)              MOVE SUM-DEPT TO PRINT-SUM.
(0131)              DISPLAY DISPLAY-TOTALS.
(0132)              MOVE ZEROS TO SUM-DEPT.
(0133)              SET INDX1 UP BY 1.
(0134)              IF INDX1 > 2
(0135)                  GO TO GET-TOTAL.
(0136)              DISPLAY 'OUT OF STATE'.
(0137)              GO TO GET-NEXT.
(0138)          *
(0139)          ADD-TOTALS.
(0140)              ADD DEPT-TOTAL (INDX1, INDX2, INDX3) TO SUM-DEPT.
(0141)              SET INDX3 UP BY 1.
(0142)          *
(0143)          FIND-MONTH.
(0144)              SEARCH ALL MON-TAB
(0145)                  WHEN MONTH-NO(INDX) = MONTH-ACCEPT
(0146)                  MOVE MONTH-VALUE (INDX) TO PRINT-MONTH.
(0147)          *
(0148)          DONE-PARA.
(0149)              STOP RUN.
```

SUM-DEPT	0041	0130	0132	0140
TABLE-AREA	0062			
TABLE-VALUE	0044	0057		
VALID-MONTH	0043	0121		

P R O G R A M S T A T I S T I C S

EXECUTABLE CODE SIZE: 546 WORDS.
CONSTANT POOL SIZE: 98 WORDS.
TOTAL PURE PROCEDURE SIZE: 644 WORDS.

WORKING-STORAGE SIZE: 1634 BYTES.
TOTAL LINKFRAME SIZE: 1320 WORDS.

STACK SIZE: 56 WORDS.

TRACE MODE: OFF.

NO ARGUMENTS EXPECTED.

149 SOURCE LINES.

NO ERRORS, NO WARNINGS, PRIME V-MODE COBOL, REV 17.00.11 <SORTIT>

OK, SEG
[SEG REV 17.0]
LOAD #SORTIT
$ LO B_SORTIT (LOAD BINARY FILE)
$ LI VCOBLB (LOAD COBOL LIBRARY)
$ LI VSRTLI (LOAD SORT LIBRARY)
$ LI (LOAD GENERAL LIBRARY)
 LOAD COMPLETE
$ SA (SAVE SEG FILE)
$ Q (QUIT)

OK,

```
**********************************************************************
**********************************************************************
**
**
**
**      JJJ    III    M   M   M   M   Y   Y
**       J      I     MM MM   MM MM   Y   Y
**       J      I     M M M   M M M    Y Y
**       J      I     M   M   M   M     Y
**   J   J      I     M   M   M   M     Y
**   J   J      I     M   M   M   M     Y
**    JJ       III    M   M   M   M     Y
**
**
**
**   H   H  EEEEE  L      PPPP          SSS   EEEEE  EEEEE
**   H   H  E      L      P   P        S   S  E      E
**   H   H  E      L      P   P        S      E      E
**   HHHHH  EEEE   L      PPPP          SSS   EEEE   EEEE
**   H   H  E      L      P                S  E      E
**   H   H  E      L   L  P        ..   S   S  E      E
**   H   H  EEEEE  LLLLL  P        ..    SSS   EEEEE  EEEEE
**
**
**
**********************************************************************
**********************************************************************
```

F77 IS THE FORTRAN 77 COMPILER IT IS INVOKED BY THE COMMAND:

```
F77 NAME [OPTIONS]
```

)PTIONS ARE PRECEDED BY A '-'. THE NAME MAY BE A PATH NAME,
3UT NEITHER IT NOR ANY FILE NAME MADE FROM IT MAY EXCEED 32 CHARACTERS.
THE COMMAND LINE SYNTAX IS THE SAME AS OTHER PRIME COMPILERS: THE -S, -B, AND
-L OPTIONS ARE ALL SUPPORTED.

THE FOLLOWING OPTIONS ARE SUPPORTED:

-XREF - PRODUCE A CROSS-REFERENCE LISTING. (IMPLIES L)
-NOXREF - DON'T HAVE -XREF

-OFFSET - PRODUCE A OFFSET MAP IN L_NAME
-EXPLIST - PRODUCE A PSEUDO-ASSEMBLY LISTING OF THE GENERATED CODE
    IN L_NAME (IMPLIES L)

-OPTIMIZE - EXECUTE THE OPTIMIZER PHASE
-NOOPTIMIZE - DON'T USE THE OPTIMIZER
-STATISTICS - PRINT OUT STATISTICS ABOUT THE COMPILATION

-RANGE - COMPILE CODE TO CHECK SUBSCRIPT AND SUBSTR RANGES
-UPCASE - MAP LOWER CASE TO UPPER CASE IN IDENTIFIERS
-LCASE - UPPER AND LOWER CASE ARE DISTINCT IN IDENTIFIERS

-SILENT - SUPPRESS LEVEL 1 ERROR MESSAGES
-DEBUG - PRODUCE A DEBUGGER SYMBOL TABLE
-DYNM - PUT LOCAL VARIABLES ON STACK

-SAVE - PUT LOCAL VARIABLES IN STATIC SPACE
-INTS - ASSUME INTEGERS ARE INTEGER*2
-INTL - ASSUME INTEGERS ARE INTEGER*4

-LOGS - ASSUME LOGICALS ARE LOGICAL*2
-LOGL - ASSUME LOGICALS ARE LOGICAL*4
-BIG - HANDLE ARRAYS LARGER THAN A SEGMENT

-NOBIG - NO ARRAYS LARGER THAN 1 SEGMENT
-64V - GENERATE V-MODE CODE
-32I - GENERATE I-MODE CODE (NOT YET SUPPORTED)

-D01 - MAKE ALL DO LOOPS AT LEAST ONE TRIP

THE DEFAULT OPTIONS ARE: '-64V -OPTIMIZE -INTL -UPCASE -NOBIG'.

EXAMPLE:

```
F77 FOO -L YES -NESTING
```

WILL COMPILE FOO TO PRODUCE AN OBJECT FILE NAMED B_FOO AND A LISTING
FILE NAMED L_FOO. THE LISTING WILL CONTAIN A NESTING LEVEL NUMBER AND
THE CODE WILL BE OPTIMIZED.

COMPILATION STATISTICS ARE WRITTEN TO THE TERMINAL AFTER EACH PHASE
IS COMPLETE IF THE -STATISTICS OPTION IS SPECIFIED.

ERROR MESSAGES ARE WRITTEN TO THE TERMINAL AND TO NAME.ERROR. FOUR
LEVELS OF ERRORS ARE REPORTED: LEVEL 1 IS A WARNING, LEVEL 2 IS AN

ERROR THAT HAS BEEN FIXED, LEVEL 3 IS AN ERROR THAT HAS NOT BEEN FIXED, AND LEVEL 4 IS AN ERROR THAT PREVENTS CONTINUED COMPILATION. ANY ERROR OF LEVEL 3 PREVENTS OPTIMIZATION AND CODE GENERATION IF DETECTED PRIOR TO THOSE PHASES.

COMMENTS:

1. THE CROSS-REFERENCE OPTION MAY CAUSE THE COMPILER'S VIRTUAL SYMBOL SPACE TO OVERFLOW FOR VERY LARGE SOURCE PROGRAMS.

2. THE LINE NUMBERS GIVEN IN DBG REFER TO THE LINE NUMBERS FOUND IN THE LISTING FILE, SO IT IS RECOMMENDED THAT A LISTING FILE BE MADE FOR USE WITH DBG. THE SOURCE FILE DBG WILL ATTEMPT TO OPEN IS THE FILE DBG.SOURCE IN THE CURRENT UFD.

```
**********************************************************************
**********************************************************************
**
**
**
**      JJJ    III    M    M   M    M   Y    Y
**       J      I     MM  MM   MM  MM   Y    Y
**       J      I     M M M    M M M     Y Y
**       J      I     M  M  M  M  M      Y
**   J   J      I     M     M  M     M   Y
**   J   J      I     M     M  M     M   Y
**    JJ       III    M     M  M     M   Y
**
**
**
**   RRRR    M    M   III    DDDD     AAA     SSS      1
**   R   R   MM  MM    I     D   D   A   A   S   S    11
**   R   R   M M M M   I     D   D   A   A   S        1
**   RRRR    M  M  M   I     D   D   AAAAA    SSS      1
**   R R     M     M   I     D   D   A   A        S    1
**   R  R    M     M   I     D   D   A   A   S   S    1
**   R   R   M     M  III    DDDD    A   A    SSS    111
**
**
**
**********************************************************************
**********************************************************************
```

MIDAS 17.1 MODIFICATIONS

DATE:

TO:

FROM:

SUBJECT: MIDAS 17.1 MODIFICATIONS

REFERENCE:

MINOR ENHANCEMENTS

MIDAS UTILITIES CREATK, KBUILD, KIDDEL, REVERT, AND MPACK NOW DISPLAY THE NAME OF THE UTILITY AND THE CURRENT REV STAMP.

SUBROUTINE MSGCTL

SUBROUTINE MSGCTL ALLOWS A PROCESS TO DISABLE OR ENABLE THE PRINTING OF ERROR MESSAGES AT A TERMINAL.

CALLING SEQUENCE: CALL MSGCTL (FUNCTION)

FUNCTION -- EM$DSB, DISABLE ERROR MESSAGE DISPLAY
EM$ENB, ENABLE ERROR MESSAGE DISPLAY

1. THE DEFAULT CONDITION RESULTS IN ERROR MESSAGES BEING DISPLAYED.

2. PARAMETERS EM$DSB AND EM$ENB MAY BE FOUND IN INSERT FILE SYSCOM>PARM.K AND MIDAS INSERT FILE KPARAM.

3. MSGCTL IS USED BY THE COBOL AND BASICV RUN TIME PACKAGES.

4. THE ROUTINE IS INTENDED FOR THOSE USERS OF FORMS OR SOME OTHER SPECIAL SCREEN FORMATTING PROCEDURES. TO AVOID POSSIBLE PROBLEMS, FORTRAN APPLICATION PROGRAMMERS SHOULD BE CERTAIN THAT ALL ERROR CONDITIONS ARE CHECKED UPON RETURN FROM CALLS TO MIDAS.

SEMAPHORE CHANGE

MIDAS NOW USES SEMAPHORE -16 RATHER THAN NUMBER 64. THE VALUE OF THE

SEMAPHORE IS DETERMINED BY PARAMETER MSEMA1 IN FILE KPARAM. NOTE THAT NEGATIVE SEMAPHORE NUMBERS ARE FOR USE BY PRIME SUPPORTED SOFTWARE ONLY. THIS CHANGE WAS MADE TO AVOID CONFLICT WITH CUSTOMER USE OF SEMAPHORES.

```
**********************************************************************************************
**********************************************************************************************
**
**
**
**      JJJ    III    M   M   M   M   Y   Y
**       J      I     MM MM   MM MM   Y   Y
**       J      I     M M M   M M M    Y Y
**       J      I     M M M   M M M     Y
**    J  J      I     M   M   M   M     Y
**    J  J      I     M   M   M   M     Y
**     JJ      III    M   M   M   M     Y
**
**
**
**    RRRR    V   V   SSS    RRRR   TTTTT  L        III     222
**    R   R   V   V  S   S   R   R    T    L         I     2   2
**    R   R   V   V  S       R   R    T    L         I        2
**    RRRR    V   V   SSS    RRRR     T    L         I       2
**    R R     V V        S   R R      T    L         I      2
**    R  R    VV     S   S   R  R     T    L   L     I     2
**    R   R   V       SSS    R   R    T    LLLLL   III   22222
**
**
**
**********************************************************************************************
**********************************************************************************************
```

MODIFICATIONS TO SORT AND VSRTLI

DATE:

TO:

FROM:

SUBJECT: MODIFICATIONS TO SORT AND VSRTLI

REFERENCE:

ABSTRACT

THIS DOCUMENT LISTS THE CHANGES THAT HAVE BEEN MADE TO VSRTLI FOR REV 17.1. SINCE SORT USES VSRTLI ROUTINES, CHANGES MADE TO THE LIBRARY AFFECT THE SORT COMMAND.

THE FOLLOWING BUGS HAVE BEEN FIXED:

1. WHEN THE INPUT RECORD TYPE IS UNCOMPRESSED SOURCE, VARIABLE, OR FIXED LENGTH, AND THE OUTPUT RECORD TYPE IS COMPRESSED SOURCE, THEN TRAILING BLANKS IN THE OUTPUT RECORDS ARE COMPRESSED BUT NOT DELETED.

   FIX: TRAILING BLANKS ARE NOW REMOVED.

2. SORTING AN INPUT FILE CONTAINING COMPRESSED SOURCE RECORDS INTO AN OUTPUT FILE TO CONTAIN UNCOMPRESSED SOURCE, VARIABLE, OR FIXED LENGTH RECORDS WILL NOT WORK CORRECTLY.

   FIX: COMPRESSED SOURCE RECORDS WILL NOW BE CONVERTED CORRECTLY.

3. WHEN PASSING SOURCE RECORDS TO SORT FROM AN INPUT PROCEDURE, THEY MUST END WITH A NEWLINE CHARACTER (:212). IF NOT, SORT WILL NOT ISSUE AN ERROR MESSAGE, AND WILL RETURN THE RECORDS WITH A NEWLINE APPENDED WHICH MAY OVERFLOW YOUR BUFFER IF USING AN OUTPUT PROCEDURE, CAUSING UNPREDICTABLE RESULTS.

   FIX: IF A SOURCE RECORD IS PASSED WITHOUT A NEWLINE ON THE END, A "WARNING - LINE TRUNCATED" MESSAGE WILL BE ISSUED AND THE LAST CHARACTER WILL BE OVERWRITTEN BY A NEWLINE.

4. WHEN SORTING RECORDS WITH EQUAL KEYS, THE ORDER THEY ARE SORTED IS UNDEFINED.

   FIX: THE ORDER OF INPUT IS MAINTAINED FOR RECORDS WITH EQUAL KEYS.

5. THE LIBRARY ROUTINES DO NOT ALLOW A MIXTURE OF INPUT/OUTPUT PROCEDURES AND FILES WHEN SORTING OR MERGING.

   FIX: A MIXTURE OF INPUT/OUTPUT PROCEDURES AND FILES IS NOW SUPPORTED WHEN SORTING, BUT ONLY FILE I/O CAN BE USED WHEN MERGING.

6. "WARNING - LINE TRUNCATED" MESSAGES ARE PRINTED INCONSISTENTLY FOR THE VARIOUS RECORD TYPES.

   FIX: RECORD TRUNCATION WARNING MESSAGES ARE PRINTED WHENEVER THE DATA (NOT INCLUDING RECORD DELIMITERS) IN THE INPUT OR OUTPUT RECORDS EXCEEDS THE MAXIMUM RECORD LENGTH, AND THE EXCESS DATA IS IGNORED.

```
**********************************************************************
**********************************************************************
**
**
**
**      JJJ    III    M    M   M    M   Y    Y
**       J      I     MM  MM   MM  MM   Y    Y
**       J      I     M M M    M M M     Y Y
**       J      I     M M M    M M M      Y
**   J   J      I     M   M    M   M      Y
**   J   J      I     M   M    M   M      Y
**    JJ       III    M   M    M   M      Y
**
**
**
**   RRRR    PPPP    L          1      GGG
**   R   R   P   P   L         11     G   G
**   R   R   P   P   L          1     G
**   RRRR    PPPP    L          1     G
**   R R     P       L          1     G  GG
**   R  R    P       L    L     1     G   G
**   R   R   P       LLLLL     111     GGGG
**
**
**
**********************************************************************
**********************************************************************
```

PL1G COMPILER FOR REV. 17.1

DATE: AUGUST 13, 1979

TO:

FROM:

SUBJECT: PL1G COMPILER FOR REV. 17.1

REFERENCE: NONE

ABSTRACT

THIS MEMO DESCRIBES THE OPERATING PROCEDURES OF THE NEW PL1 SUBSET-G COMPILER. IN PARTICULAR, THE COMMAND LINE OPTIONS OF THE NEW COMPILER ARE LISTED AND NOTES ON THE USE OF THE COMPILER ARE FURNISHED. THIS INFORMATION WILL ALLOW USERS ALREADY FAMILIAR WITH SUBSET PL1 TO COMPILE, LOAD, AND EXECUTE THEIR PROGRAMS. USERS NOT FAMILIAR WITH THE PL1 LANGUAGE SHOULD READ THE FORTHCOMING USER'S MANUAL.

1 PL1G COMPILER OPTIONS

PL1G IS THE PL/I SUBSET G COMPILER IT IS INVOKED BY THE COMMAND:

PL1G NAME [OPTIONS]

OPTIONS ARE PRECEDED BY A '-'. THE NAME MAY BE A PATH NAME, BUT NEITHER IT NOR ANY FILE NAME MADE FROM IT MAY EXCEED 32 CHARACTERS. THE COMMAND LINE SYNTAX IS THE SAME AS OTHER PRIME COMPILERS: THE -S, -B, AND -L OPTIONS ARE ALL SUPPORTED.

THE FOLLOWING OPTIONS ARE SUPPORTED:

-XREF	-- PRODUCE A CROSS-REFERENCE LISTING. (IMPLIES L)
-OFFSET	-- PRODUCE AN OFFSET MAP IN L_NAME (IMPLIES L)
-EXPLIST	-- PRODUCE A PSEUDO-ASSEMBLY LISTING OF THE GENERATED CODE IN L_NAME (IMPLIES L)
-OPTIMIZE	-- EXECUTE THE OPTIMIZER PHASE
-NOOPTIMIZE	-- DON'T USE THE OPTIMIZER
-STATISTICS	-- PRINT OUT STATISTICS ABOUT THE COMPILATION
-RANGE	-- COMPILE CODE TO CHECK SUBSCRIPT AND SUBSTR RANGES
-UPCASE	-- MAP LOWER CASE TO UPPER CASE IN IDENTIFIERS
-LCASE	-- UPPER AND LOWER CASE ARE DISTINCT IN IDENTIFIERS
-NESTING	-- PUT A NESTING LEVEL NUMBER IN THE LISTING (IMPLIES L)
-SILENT	-- SUPPRESS LEVEL 1 (WARNING) ERROR MESSAGES
-DEBUG	-- PRODUCE A FULL DEBUGGER (D3G) SYMBOL TABLE AND DEBUGGER LISTING (DEBUGGER LISTING IS CALLED "FILENAME.DBG")
-64V	-- PRODUCE V-MODE CODE
-BIG	-- DENOTES THAT ARRAYS MAY BE LARGER THAN 1 SEGMENT
-PRODUCTION	-- PRODUCE "PRODUCTION" DEBUGGER SYMBOL TABLE

THE DEFAULT OPTIONS AS DISTRIBUTED ARE '-B YES -L NO -64V -OPTIMIZE -UPCASE'. THE DEFAULT OPTIONS MAY BE CHANGED BY USE OF THE PROGRAM PL1GDF, WHICH IS FOUND IN THE TOOLS UFD.

EXAMPLE:

```
PL1G FOO -L YES -NESTING
```

WILL COMPILE FOO TO PRODUCE AN OBJECT FILE NAMED B_FOO AND A LISTING FILE NAMED L_FOO. THE LISTING WILL CONTAIN A NESTING LEVEL NUMBER AND THE CODE WILL BE OPTIMIZED.

EACH COMPILATION PRODUCES TEMPORARY FILES (NAMED "T$XXXX") IN THE CURRENT WORKING DIRECTORY. THESE FILES ARE NORMALLY DELETED AT THE END OF THE COMPILATION.

2 ERROR MESSAGES

ERROR MESSAGES ARE WRITTEN TO THE TERMINAL AND TO FILENAME.ERROR. FOUR LEVELS OF ERRORS ARE REPORTED: LEVEL 1 IS A WARNING, LEVEL 2 IS AN ERROR THAT HAS BEEN FIXED, LEVEL 3 IS AN ERROR THAT HAS NOT BEEN FIXED, AND LEVEL 4 IS AN ERROR THAT PREVENTS CONTINUED COMPILATION. ANY ERROR OF LEVEL 3 PREVENTS OPTIMIZATION AND CODE GENERATION IF DETECTED PRIOR TO THOSE PHASES.

3 PROGRAM LOADING

AT REV. 17.1 THE PL1G COMPILER OUTPUTS V-MODE CODE EXCLUSIVELY. THUS, THE SEGMENTED LOADER (SEG) MUST BE USED TO LOAD THE OBJECT MODULES PRODUCED BY THE COMPILER. ALSO, A NEW LIBRARY MUST BE LOADED PRIOR TO LOADING THE STANDARD LIBRARY. THIS LIBRARY IS NAMED PL1GLB AND IS LOCATED IN UFD LIB. THUS, THE FOLLOWING COMMANDS ISSUED TO SEG'S VIRTUAL LOADER SHOULD BE USED FOR PL1G PROGRAMS (AFTER ALL USER MODULES HAVE BEEN LOADED):

```
$LI PL1GLB
$LI
```

4 MISCELLANEOUS NOTES

4.1 CROSS-REFERENCE OPTION

THE CROSS-REFERENCE OPTION MAY CAUSE THE COMPILER'S VIRTUAL SYMBOL SPACE TO OVERFLOW FOR VERY LARGE SOURCE PROGRAMS.

4.2 DBG INTERFACE

THE LINE NUMBERS GIVEN IN DBG REFER TO THE LINE NUMBERS FOUND IN THE DEBUGGER LISTING FILE (NAMED "FILENAME.DBG"). THESE NUMBERS WILL DIFFER FROM THOSE OF THE SOURCE IF " APPEAR IN THE DEBUGGER LISTING.

SINCE COMPILATION IN DEBUG MODE PRODUCES EXTRA INFORMATION IN THE BINARY AND SEG FILES IN ADDITION TO PRODUCING THE DEBUGGER LISTING FILE, USE OF THE "-DEBUG" OPTION FOR LARGE PROGRAMS MAY REQUIRE A SIGNIFICANT AMOUNT OF DISK SPACE.

AT REV 17.1, USE OF DBG ON PROGRAMS IN WHICH MULTIPLE EXTERNAL PROCEDURES EXIST IN A SINGLE SOURCE FILE IS NOT SUPPORTED.

4.3 SEGMENT USAGE

COMPILATION OF PL1G PROGRAMS USES SEGMENTS 4004-4007 AND 4027. IN USER PROGRAMS SEGMENTS 4027 THROUGH 4010 ARE USED - IN DESCENDING ORDER - AS THE SYSTEM FREE STORAGE POOL (IN WHICH ALLOCATE AND FREE REQUESTS OPERATE AND IN WHICH SOME COMPILER-GENERATED TEMPORARIES ARE ALLOCATED).

4.4 ONCODE BUILTIN FUNCTION

VALUES RETURNED BY THE ONCODE BUILTIN FUNCTION ARE DIVIDED INTO TWO CLASSES ACCORDING TO WHETHER OR NOT THEY REPRESENT AN INPUT-OUTPUT ERROR. VALUES WHICH ARE LESS THAN THE VALUE OF THE SYMBOL "ONCODE_BASE" ARE INPUT-OUTPUT ERRORS AND VALUES GREATER THAN OR EQUAL TO "ONCODE_BASE" REPRESENT ALL OTHER RUNTIME ERRORS. THIS SYMBOL IS DEFINED IN THE FILE SYSCOM>ONCODES.PL1. SINCE THE VALUES RETURNED BY THIS FUNCTION ARE SUBJECT TO CHANGE, IT IS RECOMMENDED THAT THIS FILE BE INCLUDED IN THE SOURCE FILE (%INCLUDE 'SYSCOM>ONCODES.PL1') AND THE SYMBOLIC KEYS IN THE FILE REFERENCED INSTEAD OF THE NUMERIC VALUES THEMSELVES.

THE SYMBOLS DEFINED IN THIS FILE REPRESENT ALL ERRORS WHICH ARE NOT RELATED TO INPUT-OUTPUT. THUS, THESE SYMBOLS SHOULD HAVE THE VALUE OF THE SYMBOL "ONCODE_BASE" ADDED TO THEM BEFORE THEY ARE USED IN CALLING "SIGNL$", SO THAT THEY ARE NOT CONFUSED WITH THE INPUT-OUTPUT RELATED ERRORS.

TWO ONE-DIMENSIONAL ARRAYS OF CHARACTER STRINGS ARE ALSO DEFINED IN THIS FILE. THEY CONTAIN THE TEXT OF THE ERROR MESSAGES OUTPUT BY THE DEFAULT ONUNIT HANDLER. THE ARRAY "IO_ONCODE_MESSAGE" - WHICH CONTAINS STRINGS DECLARED AS "CHAR(68) VARYING" - CONTAINS THE TEXT OF THE INPUT-OUTPUT RELATED ERROR MESSAGES, AND THE ARRAY "ONCODE_MESSAGE" - WHICH CONTAINS STRINGS DECLARED AS "CHAR(46) VARYING" - CONTAINS THE TEXT OF ALL THE OTHER POSSIBLE ERROR MESSAGES. TO ACCESS THE MESSAGE CORRESPONDING TO A GIVEN ONCODE VALUE, THE FOLLOWING CONSTRUCTS SHOULD BE USED:

```
ONCODE_VAL = ONCODE();
IF (ONCODE_VAL < ONCODE_BASE)
    THEN MSG = IO_ONCODE_MESSAGE(ONCODE_VAL);
    ELSE MSG = ONCODE_MESSAGE(ONCODE_VAL-ONCODE_BASE);
```

```
**********************************************************************************************
**********************************************************************************************
**
**
**
**      JJJ    III    M    M    M    M    Y    Y
**        J      I     MM  MM    MM  MM    Y    Y
**        J      I     M M M    M M M      Y Y
**        J      I     M M M    M M M       Y
**    J   J      I     M    M    M    M      Y
**    J   J      I     M    M    M    M      Y
**     JJ       III    M    M    M    M      Y
**
**
**
**    RRRR     CCC     000    PPPP     Y    Y     1
**    R   R   C   C   0   0   P   P    Y    Y    11
**    R   R   C       0   0   P   P     Y Y      1
**    RRRR    C       0   0   PPPP       Y       1
**    R R     C       0   0   P          Y       1
**    R  R    C   C   0   0   P          Y       1
**    R   R    CCC     000    P          Y      111
**
**
**
**********************************************************************************************
**********************************************************************************************
```

THE COPY UTILITY HAS BEEN CHANGED TO ALLOW A MULTI-HEAD PARTITION TO BE COPIED TO A SET OF REMOVABLE CMD CARTRIDGES, FOR BACKUP PURPOSES. THE PARTITION IS COPIED A DISK SURFACE AT A TIME.

NOTE THAT THE CONTENTS OF EACH CARTRIDGE ARE MEANINGLESS ON THEIR OWN; THEY MUST SUBSEQUENTLY BE RESTORED IN THE SAME ORDER TO AN EQUAL SIZED PARTITION (ALTHOUGH NOT NECESSARILY THE SAME ONE). NOTE ALSO THAT ON BOTH THE DUMP AND THE RESTORE, THE DISK BEING COPIED TO MUST BE FREE FROM BADSPOTS, AND IF THE ORIGINAL PARTITION BEING BACKED UP HAD BADSPOTS, THESE WILL BE TRANSFERRED TO THE FINAL PARTITION.

1. OPERATION UNDER PRIMOS IV

THE CMD MUST BE POWERED DOWN PRIOR TO CHANGING THE REMOVABLE CARTRIDGE. THEREFORE, ALTHOUGH ONLY THE TO AND FROM PARTITIONS NEED TO BE ASSIGNED, ANY REMAINING PARTITIONS OF THE CMD MUST BE SHUTDOWN. IF PART OF THE CMD IS BEING USED AS A PAGING DISC, THE COPY UTILITY SHOULD BE RUN ONLY UNDER PRIMOS II.

THE USER IS PROMPTED TO CHANGE THE CARTRIDGE BY THE MESSAGE:

```
SURFACE N READY?
```

THE CMD SHOULD THEN BE POWERED DOWN, THE NEXT CARTRIDGE INSERTED AND POWERED UP AGAIN, WHEN THE RESPONSE YES WILL CONTINUE THE COPY. IF THE RESPONSE IS GIVEN BEFORE THE DISC IS READY, THE PROGRAM WILL ABORT WITH THE MESSAGE:

```
SRWREC NOT READY
ER!
```

THE PROGRAM CAN BE RESTARTED AT THE NEXT SURFACE BY TYPING:

```
S
```

IT IS NOT NECESSARY TO RE-COPY THE PREVIOUS SURFACES.

IF, WHEN RESTORING A PARTITION, A SURFACE IS MOUNTED IN THE WRONG ORDER, THE PROGRAM WILL INFORM THE USER WHICH SURFACE HE HAS MOUNTED IN ERROR, AND THEN RE-REQUEST THE CORRECT SURFACE.

2. OPERATION UNDER PRIMOS II

AS FOR PRIMOS IV, THE USER IS PROMPTED WHEN TO CHANGE THE REMOVABLE CARTRIDGE. IF THE DISC IS NOT READY, THE PROGRAM WILL WAIT UNTIL THE DISC BECOMES READY.

4. HARDWARE STATUS

IN ORDER TO DRIVE A CMD, THE 4004 CONTROLLER MUST CONTAIN THE FOLLOWING ECRS (AS KNOWN ON 12TH JULY 79).

WIRE WRAP BOARDS	ECR	3749
ETCHED BOARDS	ECRS	3178, 3550IA.

THESE ECRS:

1. ALLOW ERRORS TO BE DETECTED IN CMD DRIVES.
2. CHANGE WRITE PROTECT STATUS BIT FROM BIT 12 TO BIT 3.
3. FIX A PROBLEM WITH THE STALL CHANNEL ORDER.
4. PREVENT CORRECTABLE AND READ ERRORS WHEN ALTERNATING BETWEEN REMOVABLE AND NON-REMOVABLE SURFACES OF A CMD.

```
**********************************************************************************
**********************************************************************************
**
**
**
**      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     SSS    L         III     SSS    TTTTT
**    R   R   S   S   L          I     S   S     T
**    R   R   S       L          I     S         T
**    RRRR     SSS    L          I      SSS      T
**    R R          S  L          I         S     T
**    R  R    S   S   L    L     I     S   S     T
**    R   R    SSS    LLLLL    III     SSS      T
**
**
**
**********************************************************************************
**********************************************************************************
```

SLIST FOR REV. 17

SLIST HAS BEEN MODIFIED AS FOLLOWS

1. ENHANCED ERROR DETECTION AND REPROTING
2. ADDITION OF K$GETU FUNCTIONALITY (IGNORED UNDER PRIMOS 2.)
3. ALWAYS CLOSES FILE WHEN ERROR OCCURS
4. PRINTS REV. #.
5. PRINTS USAGE INFO WHEN INVOKED WITHOUT FILENAME
6. OBSOLETE FILE SYSTEM CALLS REPLACED
7. RESTART PROBLEM CORRECTED.

```
**********************************************************************
**********************************************************************
**
**
**
**      JJJ    III    M    M   M    M   Y    Y
**       J      I     MM  MM   MM  MM   Y    Y
**       J      I     M M M    M M M     Y Y
**       J      I     M M M    M M M      Y
**    J  J      I     M   M    M   M      Y
**    J  J      I     M   M    M   M      Y
**     JJ      III    M   M    M   M      Y
**
**
**
**    RRRR    BBBB     AAA     SSS     III     CCC
**    R   R   B   B   A   A   S   S     I     C   C
**    R   R   B   B   A   A   S         I     C
**    RRRR    BBBB    AAAAA    SSS      I     C
**    R R     B   B   A   A        S    I     C
**    R  R    B   B   A   A   S   S     I     C   C
**    R   R   BBBB    A   A    SSS     III     CCC
**
**
**
**********************************************************************
**********************************************************************
```

STATUS OF TARS FOR THIS, REV 16.4 BASIC:

STATUS OF TARS FOR THIS, REV 16.4 BASIC:

12546,80582 - PRINT USING JUXTAPOSED ITEMS WHEN THE FIRST NUMERIC ITEM OVERFLOWED. FIXED.

13717 - .NL. DID NOT RESET COLUMN COUNT IN ENTER STATEMENT. FIXED.

24728 - STATEMENT NUMBER 0 WAS NOT SENSED AS AN ERROR. FIXED.

15819 - PRINT USING ROUNDING IS NOT CONSISTENT. MACHINE FLOATING ACCURACY IS THE PROBLEM HERE, BUT NOTE THAT THE ACTUAL COMPUTATION ACCURACY IS NOT AFFECTED BY THIS PROBLEM, WHICH IS DUE TO THE INPUT CONVERSION OF ASCII DIGITS TO FLOATING NUMBERS. A BETTER METHOD IS USED BY BASIC/VM AND FORTRAN, SO THESE PROBLEMS WILL NOT SHOW UP.

80236,80469 - HALTS ARE ENCOUNTERED WHEN STRINGS ARE PASSED TO A FORTRAN PROGRAM. THE DOCUMENTATION IS WRONG, AND INDEED STRINGS ARE NOT ALLOWED TO BE PASSED TO A FORTRAN PROGRAM.

22783 - A 'FOR-NEXT UNMATCHING' ERROR WAS GENERATED WHEN IN FACT NO MISMATCH EXISTED. FIXED.

```
**********************************************************************
**********************************************************************
*
*
*
*      JJJ    III    M    M   M    M   Y    Y
*       J      I     MM  MM   MM  MM   Y    Y
*       J      I     M M M    M M M     Y Y
*       J      I     M M M    M M M      Y
*   J   J      I     M   M    M   M      Y
*   J   J      I     M   M    M   M      Y
*    JJ       III    M   M    M   M      Y
*
*
*
*   RRRR     AAA    V   V    AAA    III   L
*   R   R   A   A   V   V   A   A    I    L
*   R   R   A   A   V   V   A   A    I    L
*   RRRR    AAAAA   V   V   AAAAA    I    L
*   R R     A   A    V V    A   A    I    L
*   R  R    A   A    VV     A   A    I    L    L
*   R   R   A   A    V      A   A   III   LLLLL
*
*
*
**********************************************************************
**********************************************************************
```

\VAIL FOR REV. 17

\VAIL FOR REV. 17

\VAIL NOW SUPPORTS 62 LOGICAL DISKS.
\VAIL USED TO PRINT GARBAGE WHEN 'REPORT MODE' IS SPECIFIED
\ND ONLY VOLUME ID IS IN THE DISCS FILE.

```
*****************************************************************************
*****************************************************************************
*
*
*
*      JJJ    III    M    M   M    M   Y    Y
*       J      I     MM  MM   MM  MM   Y    Y
*       J      I     M MM M   M MM M    Y Y
*       J      I     M    M   M    M     Y
*   J   J      I     M    M   M    M     Y
*   J   J      I     M    M   M    M     Y
*    JJ       III    M    M   M    M     Y
*
*
*
*   RRRR     AAA    PPPP    PPPP    L        III    BBBB     222
*   R   R   A   A   P   P   P   P   L         I     B   B   2   2
*   R   R   A   A   P   P   P   P   L         I     B   B       2
*   RRRR    AAAAA   PPPP    PPPP    L         I     BBBB      2
*   R R     A   A   P       P       L         I     B   B    2
*   R  R    A   A   P       P       L    L    I     B   B    2
*   R   R   A   A   P       P       LLLLL   III    BBBB    22222
*
*
*
*****************************************************************************
*****************************************************************************
```

SUBJECT- APPLIB AND VAPPLB BUG FIXES FOR 17.0

SUBJECT- APPLIB AND VAPPLB BUG FIXES FOR 17.0

THE FOLLOWING BUGS WERE FIXED.

. MSUB$A PERFORMED INCORRECTLY IN SOME CASES WHEN THE FIRST
CHARACTER OF THE SUBSTRING TO BE MOVED OCCUPIED THE RIGHT HALF
OF A WORD (I.E. WHEN AFC WAS EVEN) THIS PROBLEM ALSO AFFECTED
ISTR$A

. UNIT$A WOULD RETURN .FALSE. IF PASSED A BAD UNIT NUMBER.
T NOW RETURNS .TRUE.

. THE VARIOUS OPEN ROUTINES SOMETIMES RETURNED A GARBAGE CHARACTER
T THE END OF A TREENAME IF THE ERROR 'FILE NOT FOUND' OCCURRED.
HIS AFFECTED THE ERROR MEASAGE ONLY. IT HAS BEEN CORRECTED

```
**********************************************************************
**********************************************************************
**
**
**
**      JJJ    III    M    M   M    M   Y    Y
**       J      I     MM  MM   MM  MM   Y    Y
**       J      I     M M M    M M M     Y Y
**       J      I     M   M    M   M      Y
**   J   J      I     M   M    M   M      Y
**   J   J      I     M   M    M   M      Y
**    JJ       III    M   M    M   M      Y
**
**
**
**   RRRR     AAA    PPPP    PPPP    L        III    BBBB      1
**   R   R   A   A   P   P   P   P   L         I     B   B    11
**   R   R   A   A   P   P   P   P   L         I     B   B     1
**   RRRR    AAAAA   PPPP    PPPP    L         I     BBBB      1
**   R R     A   A   P       P       L         I     B   B     1
**   R  R    A   A   P       P       L    L    I     B   B     1
**   R   R   A   A   P       P       LLLLL   III    BBBB    111
**
**
**
**********************************************************************
**********************************************************************
```

APPLICATIONS LIBRARY FORREV 17.1

DATE:

TO:

FROM:

SUBJECT: APPLICATIONS LIBRARY FORREV 17.1

REFERENCE:

THE FOLLOWING BUG FIXES AND CHANGES WERE MADE TO THE APPLICATIONS LIBRARY (APPLIB AND VAPPLB) FOR REV 17.1:

1) TAR #13361 - THE ROUTINE TIME$A WAS RETURNING AN INCORRECT FLOATING POINT VALUE AT THE MINUTE MARK BECAUSE OF MISPLACED CODE.

2) THE ROUTINE RNUM$A WAS NOT RESETTING ITS' NEGATIVE NUMBER FLAG WHEN IT ENCOUNTERED AN ILLEGAL NEGATIVE NUMBER. CONSEQUENTLY, WHEN THE PROMPT WAS REPEATED AND A NEW NUMBER ENTERED, IT ALWAYS CAME BACK NEGATIVE.

3) THE MAXIMUM RDTK$$ BUFFER SIZE IN THE ROUTINE RNAM$A HAS BEEN INCREASED TO 80 WORDS (160 CHARACTERS), AND THE CALCULATION OF THIS BUFFER SIZE IN WORDS HAS BEEN CORRECTED TO ALLOW FOR ODD BUFFER LENGTHS (IN CHARACTERS) SPECIFIED BY THE USER.

4) THE ROUTINES EXST$A AND DELE$A HAVE BEEN MODIFIED SO THAT UNDER PRIMOS II THE SEARCH RANGE FOR AN AVAILABLE FILE UNIT IS 1 TO 16.

5) ALL ROUTINES IN VAPPLB HAVE BEEN COMPILED WITH THE "-PBECB" OPTION.

```
**********************************************************************
**********************************************************************
**
**
**
**      JJJ    III    M    M   M    M   Y     Y
**       J      I     MM  MM   MM  MM   Y     Y
**       J      I     M M M    M M M     Y Y
**       J      I     M  M     M  M      Y
**   J   J      I     M        M        Y
**   J   J      I     M        M        Y
**    JJ       III    M        M        Y
**
**
**
**   RRRR      CCC    M    M   PPPP     FFFFF
**   R   R    C   C   MM  MM   P   P    F
**   R   R    C       M M M    P   P    F
**   RRRR     C       M  M     PPPP     FFFF
**   R R      C       M        P        F
**   R  R     C   C   M        P        F
**   R   R     CCC    M        P        F
**
**
**
**********************************************************************
**********************************************************************
```

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 JANUARY 5, 1978

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 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 CMPF 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.

EXAMPLE:

CONSIDER THE FOLLOWING TWO FILES:

FILEA	FILEB
THE	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.
```

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] -OUTF 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:

-OUTF IS REQUIRED BEFORE THE NAME OF THE OUTPUT FILE TREE NAME.

-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 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 MRGF COMMAND CAN BE INVOKED FROM A COMINPUT FILE. IF EDIT MODE IS ENTERED, HOWEVER, INPUT WILL BE ACCEPTED ONLY FROM THE USER'S TERMINAL. THE COMINPUT FILE WILL CONTINUE TO BE USED AS INPUT FOLLOWING THE COMPLETION OF THE MRGF. THIS FEATURE ALLOWS A USER TO PERFORM MRGF OPERATIONS FROM A COMINPUT FILE WITHOUT HAVING TO KNOW IN ADVANCE THE PROPER RESPONSES TO EDIT REQUESTS.

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
```

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.

```
**********************************************************************************
**********************************************************************************
**
**
**
**      JJJ    III    M    M   M    M   Y    Y
**       J      I     MM  MM   MM  MM   Y    Y
**       J      I     M M M    M M M     Y Y
**       J      I     M M M    M M M      Y
**   J   J      I     M   M    M   M      Y
**   J   J      I     M   M    M   M      Y
**    JJ       III    M   M    M   M      Y
**
**
**
**   RRRR     CCC    H   H    AAA    N   N   GGG    EEEEE   SSS
**   R   R   C   C   H   H   A   A   NN  N  G   G   E      S   S
**   R   R   C       H   H   A   A   N N N  G       E      S
**   RRRR    C       HHHHH   AAAAA   N N N  G       EEEE    SSS
**   R R     C       H   H   A   A   N N N  G   GG  E          S
**   R  R    C   C   H   H   A   A   N  NN  G   G   E      S   S
**   R   R    CCC    H   H   A   A   N   N   GGGG   EEEEE   SSS
**
**
**
**********************************************************************************
**********************************************************************************
```

SUBJECT: MAJOR CHANGES AND INCOMPATIBILITIES BETWEEN REV 16 AND REV 17

A. COMMANDS DROPPED

THE FOLLOWING COMMANDS ARE OBSOLETE AND HAVE BEEN DROPPED FROM THE REV 17 MASTER DISK:

EXPAND, CMPRES, PUSS, HILOAD, FTNOPT, AND CVVTMA

B. NEW COMMANDS

THE FOLLOWING NEW COMMANDS HAVE BEEN ADDED TO THE NON-CHARGEABLE MASTER DISK:

BATGEN, JOB, $$, BATCH, CONCAT, AND PROP

C. CHARGEABLE PRODUCTS DROPPED

THE FOLLOWING PRODUCTS HAVE BEEN DROPPED. ALL OF THEM HAVE BEEN REPLACED BY SIMILAR PRODUCTS AS LISTED IN THE NEXT SECTION. LIST:

HASP 300 8400, RJ2780, RJ1004, RJ2780, AND RJCDC.

D. CHARGEABLE PRODUCTS ADDED

THE FOLLOWING CHARGEABLE PRODUCTS HAVE BEEN ADDED:

DBG, DPTX-DSC, DPTX-TSF, DPTX-TCF, RJE2780, RJE1004, RJE200VT, RJEHASP, RJEGRTS, AND RJE702C

E. PRODUCT STATUS CHANGES

THE FOLLOWING PRODUCTS HAVE BEEN MOVED FROM THE NON-CHARGEABLE MASTER DISK TO THE CHARGEABLE MASTER DISK AND HAVE BEEN REORGANIZED APPROPRIATELY:

FTN, MIDAS, BASIC, AND DBASIC.

NOTE THAT BOTH BASIC AND DBASIC ARE NOW ORGANIZED UNDER THE BASIC UFD.

F. MAJOR INCOMPATIBILITIES

MAJOR CHANGES AND INCOMPATIBILITIES BETWEEN REV 16 AND 17

1. THE REV 17 SHARED LIBRARIES CANNOT BE RUN UNDER REV 16 PRIMOS AND VISA VERSA.

2. THE BINARY FILES GENERATED BY REV 17 COMPILES CANNOT BE LOADED BY REV 16 SEG, BUT BY REV 16 BINARY FILES CAN BE LOADED BY REV 17 SEG.

3. THE REV 16 NETWORK CONFIGURATION FILE NETCON WILL NOT RUN WITH A REV 17 PRIMOS.

4. THE OPERATOR INTEFACE TO THE LINE PRINTER SPOOLER IS INCOMPATIBLE WITH REV 16. THE STARTUP PROCEDURE FOR SPOOLER PHANTOMS HAS CHANGED.

5. REV 16 COMPILED BASIC V CODE CANNOT BE RUN USING REV 17 BASIC V. PROGRAMS MUST BE RECOMPILED.

6. THE V-MODE FORTRAN LIBRARY USES SEGMENT 2050 INSTEAD OF 2014. THIS CHANGE REQUIRES CHANGES TO THE C_PRMO FILE THAT BRINGS UP PRIMOS. THE NEW COMMANDS ARE GIVEN IN C_PRMO.TEMPLATE IN UFD PRIRUN.

```
**********************************************************************************************
**********************************************************************************************
**
**
**
**      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    RRRR   U   U   N   N   000   FFFFF   FFFFF     1
**    R   R   R   R  U   U   NN  N  0   0  F       F        11
**    R   R   R   R  U   U   N N N  0   0  F       F         1
**    RRRR    RRRR   U   U   N N N  0   0  FFFF    FFFF      1
**    R R     R R    U   U   N N N  0   0  F       F         1
**    R  R    R  R   U   U   N  NN  0   0  F       F         1
**    R   R   R   R   UUU    N   N   000   F       F       111
**
**
**
**********************************************************************************************
**********************************************************************************************
```

MODIFICATIONS TO RUNOFF

DATE:

TO:

FROM:

SUBJECT: MODIFICATIONS TO RUNOFF

REFERENCE:

ABSTRACT

THIS DOCUMENT LISTS THE CHANGES THAT HAVE BEEN MADE TO RUNOFF FOR REV 15.6. ALL OF THESE BUG FIXES HAVE BEEN MADE TO RUNOFF REV 16.4 OR 16.6, AND ARE ALSO DOCUMENTED IN EARLIER UPDATES OF THIS PET.

THE FOLLOWING CHANGES HAVE BEEN MADE IN RUNOFF FOR REV 15.6:

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) RUNOFF DID NOT ALLOW REVISION BARS TO BE PRINTED ON BLANK LINES IN THE MIDDLE OF A REVISED BLOCK OF TEXT. BY TURNING THE REVISION BAR ON WITH ".RBAR ALL", BLANK LINES WILL NOW GET THE RBAR.

2) ADJUST MODE DID NOT ALWAYS ADJUST QUITE RIGHT WHEN DEALING WITH UNDERLINED TEXT THAT ALSO INCLUDED PHANTOM HYPHENS. THIS HAS BEEN CORRECTED.

3) USING A PHANTOM HYPHEN IN A DECIMAL HEADING WHEN GENERATING A TABLE OF CONTENTS NO LONGER SAVES A SPACE FOR THE HYPHEN WHEN FILLING THE TABLE OF CONTENTS LINE WITH PERIODS BEFORE THE NUMBER. THIS MEANS THE PAGE NUMBERS WILL BE CORRECTLY LINED UP.

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) WHEN A .SM COMMAND HAPPENED TO COINCIDE WITH A PAGE EJECT OCCURRING BECAUSE OF A PREVIOUS COMMAND, THE NEW SIDE MARGINS DID NOT ACTUALLY TAKE EFFECT UNTIL THE NEXT PAGE. THIS HAS BEEN CORRECTED. OTHER COMMANDS THAT CAUSE A POSSIBLE BREAK AND EJECT TO SET UP A NEW PAGE ENVIRONMENT WILL ALSO NOW TAKE EFFECT ON THE APPROPRIATE PAGE RATHER THAN POSSIBLY WAITING AN EXTRA PAGE. (TAR 24980)

6) 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.

7) 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.

8) RUNOFF ALWAYS USED THE DEFAULT ERASE AND KILL CHARACTERS OR " AND ? EVEN IF THE USER HAD RESET THESE TO BE OTHER CHARACTERS. RUNOFF WILL NOW USE THE USER'S ERASE AND KILL CHARACTERS IN COMMAND MODE. (TARS 20194, 23668)

9) IF A DECIMAL HEADER GOT PUSHED TO THE TOP OF A PAGE BECAUSE OF NOT BEING ALLOWED TO LEAVE WIDOWS ON THE BOTTOM OF THE PREVIOUS PAGE AND A FLOAT FILE HAD BEEN ENTERED SOMETIME PREVIOUSLY, THERE WAS THE POSSIBILITY OF GETTING PART OR ALL OF THE HEADER FOLLOWED BY THE FLOAT FILE BEFORE GETTING THE TEXT THAT BELONGED WITH THE HEADER. THIS HAS BEEN CORRECTED BY CHECKING IF A FLOAT SHOULD START AFTER THE NEED, BEFORE PUTTING THE HEADER IN THE OUTPUT FILE. IF FSTART IS TRUE WE BACK UP THE INPUT FILE ONE LINE, BY THE SUBROUTINE BACKUP, DO THE FLOAT FILE AND COME BACK AND RE-READ THE DECIMAL HEADER COMMAND THAT CAUSED THE PROBLEM AND RE-DO IT. (TAR 23222)

10) .DL DID NOT WORK IF YOU HAPPENED TO DO A DLEVEL TO THE FIRST TIME

AT THAT LEVEL. .DL LEAVES THE USER AT THE TEXT MARGIN FOR THE SPECIFIED LEVEL. THE .DN COMMAND CHECKS IF THIS IS THE FIRST TIME AT THAT LEVEL AND IF IT IS, DOES NOT DO AN UNDENT TO THE HEADING MARGIN. SINCE FUTURE DECIMALIZATION COMMANDS ONLY DO RELATIVE AND NOT ABSOLUTE INDENTS THE MARGINS ARE OFF THEREAFTER BY THE AMOUNT OF THAT LEVEL'S TEXT INDENT. ADDING A FLAG TDL TO SAY WHEN IT WAS A .DL THAT GOT YOU TO THE LEVEL THE UNDENT WILL HAPPEN ANYWAY IF YOU ARE AT THE TOP NUMBER FOR THE LEVEL AND GOT THERE BY A .DL. (TAR 23221)

11) RUNOFF HAS BEEN MODIFIED TO ACCEPT TREENAMES UP TO 80 CHARACTERS IN LENGTH FOR .INSERT FILES. (TAR 12603)

12) WHEN PROCESSING THE .//// COMMAND FOR APPORTIONING TEXT ACROSS A LINE, RUNOFF OCCASIONALLY BLANKED OUT ANYTHING THAT MIGHT HAVE BEEN IN THE FIRST COLUMN IF THE APPORTIONED TEXT BELONGED IN A LATER COLUMN WHEN WORKING WITH MULTIPLE COLUMNS. (TAR 22802)

13) USING .+ OR .> WITHOUT FOLLOWING TEXT TO CREATE A BLANK LINE CAUSED RUNOFF TO PRINT TWO LINE FEEDS WHEN SENDING OUTPUT TO THE TERMINAL. BECAUSE THE TWO NEWLINES WERE PUT IN THE SAME WORD, THIS DID NOT SHOW UP WHEN THE OUTPUT WAS SENT TO A FILE AND SPOOLED. THE EXTRA LINES WERE NOT COUNTED AS FAR AS RUNOFF PAGES WERE CONCERNED EITHER WHICH CAUSED THOSE PAGES TO APPEAR LONGER THAN THEY ACTUALLY WERE. (TAR 20967)

14) THE BLANK CHARACTER DID NOT GET TRANSLATED INTO A SPACE WHEN IT WAS USED IN HEADERS AND FOOTERS.

15) DEFINED SYMBOLS COULD HAVE AT MOST 30 CHARACTERS BUT THE LOOP TO TRANSLATE THEM WENT UP TO 60 SO IF THE DEFINITION WAS LONGER THAN 30 THERE WAS GARBAGE ON THE END.

A RESTRICTION THAT RUNOFF USERS SHOULD BE AWARE OF IS, WHEN USING THE TWO COMMANDS .BREAK AND .INDENT N IN CONJUNCTION THE .BREAK SHOULD PRECEED THE .INDENT OR IT MAY NOT WORK AS EXPECTED.

```
**********************************************************************
**********************************************************************
**
**
**
**      JJJ    III    M   M   M   M   Y   Y
**       J      I     MM MM   MM MM   Y   Y
**       J      I     M M M   M M M    Y Y
**       J      I     M M M   M M M     Y
**    J  J      I     M   M   M   M     Y
**    J  J      I     M   M   M   M     Y
**     JJ      III    M   M   M   M     Y
**
**
**
**    RRRR     CCC     OOO    PPPP    Y   Y
**    R   R   C   C   O   O   P   P   Y   Y
**    R   R   C       O   O   P   P    Y Y
**    RRRR    C       O   O   PPPP     Y
**    R R     C       O   O   P        Y
**    R  R    C   C   O   O   P        Y
**    R   R    CCC     OOO    P        Y
**
**
**
**********************************************************************
**********************************************************************
```

SOFTWARE ASPECTS OF THE CARTRIDGE MODULE DISC (CMD)

DATE: 13TH JULY, 1979

SUBJECT: SOFTWARE ASPECTS OF THE CARTRIDGE MODULE DISC (CMD)

REFERENCE:

ABSTRACT

THIS DOCUMENT EXPLAINS THE FORMATTING, PARTITIONING AND ADDRESSING OF THE CARTRIDGE MODULE DISC, AND THE UPDATED UTILITIES NECESSARY FOR ITS USE.

1. DISK FORMAT

THE CMD IS A RACK-MOUNTED DRIVE OF 32, 64 OR 96 MB OF TOTAL STORAGE CAPACITY; THIS IS DIVIDED INTO 1 DISK SURFACE OF REMOVABLE MEDIA (16 MB) AND 1, 3 OR 5 SURFACES OF NON-REMOVABLE MEDIA (16, 48 OR 80 MB). THESE FIGURES DO NOT INCLUDE THE EXTRA DISC SURFACES NEEDED FOR TIMING AND POSITIONING INFORMATION. EACH SURFACE OF THE CMD IS FORMATTED IDENTICALLY TO A STORAGE MODULE DISK SURFACE; THERE ARE 823 TRACKS WITH NINE 1040-WORD RECORDS PER TRACK.

2. PARTITIONING AND ADDRESSING THE CMD

2.1 REMOVABLE SURFACE

A PRIMOS PARTITION MAY NOT SPAN BOTH THE REMOVABLE AND THE NON-REMOVABLE SURFACES OF A CMD. THUS THE REMOVABLE SURFACE WILL ALWAYS BE A SINGLE PARTITION. THE PHYSICAL DEVICE NUMBER IS SET UP AS FOR THE STORAGE MODULE (SEE SECTION 2.3).

2.2 NON-REMOVABLE SURFACES

THE PARTITIONS OF THE NON-REMOVABLE SECTION MUST START ON AN EVEN HEAD NUMBER; THE HEAD OFFSET IS TAKEN FROM THE TOP OF THE NON-REMOVABLE SECTION OF THE CMD, BUT THE HIGH ORDER BIT (BIT 1) IS TURNED ON IN ORDER TO TELL THE CMD CONTROLLER TO ACCESS THE NON-REMOVABLE SURFACES.

2.3 PHYSICAL DEVICE NUMBERS OF THE POSSIBLE CMD PARTITIONS

THE PHYSICAL DEVICE NUMBER FOR CMD PARTITIONS IS SET UP AS FOLLOWS:-

BITS 1-4	HEAD OFFSET FROM TOP OF REMOVABLE OR NON-REMOVABLE SECTION/2 BIT 1 SET ON FOR NON-REMOVABLE SECTION.
BITS 5-8 + BIT 16	NO OF HEADS
BIT 9	CONTROLLER NO.
BITS 10-13	0110 (DEVICE TYPE, AS FOR STORAGE MODULE)
BITS 14-15	UNIT NUMBER
BIT 16	(SEE BITS 5-8).

ASSUME THAT THE CMD IS DEVICE 0 ON EITHER DISC CONTROLLER (FOR OTHER DEVICES, BITS 14-15 MUST BE ALTERED ACCORDINGLY).

		FIRST CONTROLLER	SECOND CONTROLLER
32 MB	REMOVABLE	61 (16 MB)	261 (16 MB)
	NON-REMOVABLE	100061 (16 MB)	100261 (16 MB)

64 MB	REMOVABLE	61 (16 MB)	261 (16 MB)
	NON-REMOVABLE	100460 (32 MB) 110061 (16 MB) OR 100461 (48 MB)	100660 (32 MB) 110261 (16 MB) OR 100661 (48 MB)
96 MB	REMOVABLE	61 (16 MB)	261 (16 MB)
	NON-REMOVABLE	100460 (32 MB) 110460 (32 MB) 120061 (16 MB) OR 101060 (64 MB) 120061 (16 MB) OR 101061 (80 MB) OR 100460 (32 MB) 110461 (48 MB)	100660 (32 MB) 110660 (32 MB) 120261 (16 MB) OR 101260 (64 MB) 120261 (16 MB) OR 101261 (80 MB) OR 100660 (32 MB) 110661 (48 MB)

3. UTILITIES: FIXRAT, MAKE, COPY

3.1 FIXRAT

FIXRAT REV 16.4 SUPPORTS CMD DEVICES.

3.2 MAKE

MAKE REV 16.8 SUPPORTS CMD DEVICES.

3.3 COPY

REV 16.2 COPY WILL SUPPORT THE COPYING OF EQUAL SIZED PARTITIONS, WHERE EITHER ONE OR BOTH OF THE PARTITIONS ARE ON A CMD.

3.4 COPY REV 16.8 AND 17.1

THE COPY UTILITIES AT REV 16.8 AND 17.1 HAVE BEEN CHANGED TO ALLOW A MULTI-HEAD PARTITION TO BE COPIED TO A SET OF REMOVABLE CMD CARTRIDGES, FOR BACKUP PURPOSES. THE PARTITION IS COPIED A DISK SURFACE AT A TIME.

NOTE THAT THE CONTENTS OF EACH CARTRIDGE ARE MEANINGLESS ON THEIR OWN; THEY MUST SUBSEQUENTLY BE RESTORED IN THE SAME ORDER TO AN EQUAL SIZED PARTITION (ALTHOUGH NOT NECESSARILY THE SAME ONE). NOTE ALSO THAT ON BOTH THE DUMP AND THE RESTORE, THE DISK BEING COPIED TO MUST BE FREE FROM BADSPOTS, AND IF THE ORIGINAL PARTITION BEING BACKED UP HAD BADSPOTS, THESE WILL BE TRANSFERRED TO THE FINAL PARTITION.

3.4.1. OPERATION UNDER PRIMOS IV

THE CMD MUST BE POWERED DOWN PRIOR TO CHANGING THE REMOVABLE CARTRIDGE. THEREFORE, ALTHOUGH ONLY THE TO AND FROM PARTITIONS NEED TO BE ASSIGNED, ANY REMAINING PARTITIONS OF THE CMD MUST BE SHUTDOWN. IF PART OF THE CMD IS BEING USED AS A PAGING DISC, THE COPY UTILITY SHOULD BE RUN ONLY UNDER PRIMOS II.

THE USER IS PROMPTED TO CHANGE THE CARTRIDGE BY THE MESSAGE:

```
SURFACE N READY?
```

THE CMD SHOULD THEN BE POWERED DOWN, THE NEXT CARTRIDGE INSERTED AND POWERED UP AGAIN, WHEN THE RESPONSE YES WILL CONTINUE THE COPY. IF THE RESPONSE IS GIVEN BEFORE THE DISC IS READY, THE PROGRAM WILL ABORT WITH THE MESSAGE:

```
SRWREC NOT READY
ER!
```

THE PROGRAM CAN BE RESTARTED AT THE NEXT SURFACE BY TYPING:

```
S
```

IT IS NOT NECESSARY TO RE-COPY THE PREVIOUS SURFACES.

IF, WHEN RESTORING A PARTITION, A SURFACE IS MOUNTED IN THE WRONG ORDER, THE PROGRAM WILL INFORM THE USER WHICH SURFACE HE HAS MOUNTED IN ERROR, AND THEN RE-REQUEST THE CORRECT SURFACE.

3.4.2. OPERATION UNDER PRIMOS II

AS FOR PRIMOS IV, THE USER IS PROMPTED WHEN TO CHANGE THE REMOVABLE CARTRIDGE. IF THE DISC IS NOT READY, THE PROGRAM WILL WAIT UNTIL THE DISC BECOMES READY.

4. HARDWARE STATUS

IN ORDER TO DRIVE A CMD, THE 4004 CONTROLLER MUST CONTAIN THE FOLLOWING ECRS (AS KNOWN ON 12TH JULY 79).

WIRE WRAP BOARDS	ECR	3749
ETCHED BOARDS	ECRS	3178, 3550IA.

THESE ECRS:

1. ALLOW ERRORS TO BE DETECTED IN CMD DRIVES.
2. CHANGE WRITE PROTECT STATUS BIT FROM BIT 12 TO BIT 3.
3. FIX A PROBLEM WITH THE STALL CHANNEL ORDER.
4. PREVENT CORRECTABLE AND READ ERRORS WHEN ALTERNATING BETWEEN REMOVABLE AND NON-REMOVABLE SURFACES OF A CMD.

```
**********************************************************************************************
**********************************************************************************************
**
**
**
**      JJJ    III    M    M   M    M   Y    Y
**       J      I     MM  MM   MM  MM   Y    Y
**       J      I     M M M    M M M     Y Y
**       J      I     M M M    M M M      Y
**    J  J      I     M    M   M    M     Y
**    J  J      I     M    M   M    M     Y
**     JJ       III   M    M   M    M     Y
**
**
**
**    RRRR    EEEEE  DDDD      1
**    R   R   E      D   D    11
**    R   R   E      D   D     1
**    RRRR    EEEE   D   D     1
**    R R     E      D   D     1
**    R  R    E      D   D     1
**    R   R   EEEEE  DDDD    111
**
**
**
**********************************************************************************************
**********************************************************************************************
```

MODIFICATIONS TO THE EDITOR

DATE:

TO:

FROM:

SUBJECT: MODIFICATIONS TO THE EDITOR

REFERENCE:

ABSTRACT

THIS DOCUMENT DESCRIBES THE CHANGES THAT HAVE BEEN MADE TO THE EDITOR FOR REV 17.1.

CHANGES MADE TO THE EDITOR FOR REV 17.1:

1) A PAIR OF NEW MODES CKPAR AND NCKPAR WAS ADDED. THEY SPECIFY WHETHER TO PRINT CHARACTERS WITH PARITY OFF AS IF THEY HAD PARITY ON OR TO PRINT THEM AS ^NNN, WHERE N IS AN OCTAL NUMBER. CKPAR MEANS CHECK THE PARITY AND ONLY PRINT AS A REAL CHARACTER IF THE PARITY IS ON. NCKPAR MEANS DON'T CHECK THE PARITY -- PRINT ALL CHARACTERS SAME AS THEY WERE PRINTED IN REV 17.0. DEFAULT IS NCKPAR. (TAR 25039)

2) THE EDITOR NOW CHECKS IF BOXMODE IS ON BEFORE DECIDING TO PRINT THE CHARACTERS USED FOR DIRECTION IN BOXMODE AS ^E, ^N, ^W, AND ^S. (TAR 20619)

3) IF A LINE HAPPENED TO HAVE OVER 127 BLANKS ON THE END, THE FIRST 127 TRAILING BLANKS DID NOT GET TRUNCATED WHEN THE LINE WAS WRITTEN BACK TO THE FILE.

4) THE WORKING-SET SIZE OF THE SHARED EDITOR WAS REDUCED FROM JUST OVER 6 NONSHARED PAGES PER USER TO UNDER 3 NONSHARED PAGES PER USER BY REWRITING THE COMMAND FILE TO BUILD THE EDITOR. THE FORTRAN MODULES ARE COMPILED WITH THE DYNM OPTION TO REDUCE THE SIZE OF THEIR LINK FRAMES, AND WITH THE PRECB OPTION SO THAT THE ECB'S CAN BE SHARED, PURE LINK FRAMES ARE ALSO PUT IN THE SHARED SEGMENT AND THE LOAD ORDER WAS REARRANGED TO PUT SELDOM USED MODULES AT THE END.

```
**********************************************************************************************
**********************************************************************************************
**
**
**
**      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     SSS     000    RRRR    TTTTT
**    R   R   S   S   0   0   R   R     T
**    R   R   S       0   0   R   R     T
**    RRRR     SSS    0   0   RRRR      T
**    R R          S  0   0   R R       T
**    R  R    S   S   0   0   R  R      T
**    R   R    SSS     000    R   R     T
**
**
**
**********************************************************************************************
**********************************************************************************************
```

SORT COMMAND

DATE: MARCH 15, 1979

TO:

FROM:

SUBJECT: SORT COMMAND

REFERENCE: NONE

ABSTRACT

THE SORT COMMAND HAS SEVERAL ENHANCEMENTS FOR REV 17.

SORT NOW HANDLES THE FOLLOWING DATA TYPES AS SORT FIELDS (KEYS):

NUMERIC ASCII, LEADING SEPARATE SIGN
NUMERIC ASCII, TRAILING SEPARATE SIGN
NUMERIC ASCII, LEADING EMBEDDED SIGN
NUMERIC ASCII, TRAILING EMBEDDED SIGN
NUMERIC ASCII, UNSIGNED
PACKED DECIMAL.

TWO KEY TYPES FOR ASCII KEYS GIVE USERS THE OPTION OF HAVING LOWER CASE ALPHABETIC CHARACTERS SORT EQUAL TO UPPER CASE.

RECORD TYPE MAY BE EXPLICITLY SET, RATHER THAN DEFAULTING FROM THE SPECIFIED KEY TYPES. THIS GIVES FLEXIBILITY TO HANDLE NEW RECORD TYPES. FOR EXAMPLE, SORT WILL NOW HANDLE FIXED LENGTH RECORDS, AND BLANK COMPRESSED RECORDS WILL BE TREATED DIFFERENTLY THAN UNCOMPRESSED RECORDS.

THE MAXIMUM RECORD LENGTH MAY ALSO BE SPECIFIED BY THE USER.

MAXIMUM RECORD LENGTH IS NOW 32760 BYTES (CHARACTERS).

KEYS MAY BE AS LONG AS THE INPUT RECORDS.

THE MERGE OPTION IS NOW A TRUE MERGE RATHER THAN A SORT OF MULTIPLE INPUT FILES.

UP TO 20 FILES MAY BE SORTED INTO A SINGLE OUTPUT FILE.

THE USE OF KEYWORDS PROVIDES MORE FLEXIBILITY FOR SPECIFYING OPTIONAL INFORMATION.

```
********                                        ********
* SORT *                                        * SORT *
********                                        ********
```

PURPOSE:

THE SORT COMMAND IS USED TO SORT FROM ONE TO TWENTY FILES INTO A SINGLE OUTPUT FILE, OR TO MERGE FROM TWO TO ELEVEN PRESORTED FILES INTO A SINGLE FILE.

USAGE:

SORT IS INVOKED FROM PRIMOS COMMAND LEVEL. VARIOUS OPTIONS MAY BE SPECIFIED ON THE COMMAND LINE AS EXPLAINED BELOW. SORT REQUESTS A MINIMUM OF FILE AND KEY INFORMATION ON SUBSEQUENT LINES. ADDITIONAL INFORMATION, AS EXPLAINED BELOW, MAY BE SPECIFIED TO OVERRIDE DEFAULT RECORD LENGTH AND/OR RECORD TYPE.

INVOCATION:

SORT [-BRIEF] [-SPACE] [-MERGE]

THE MEANING OF THESE OPTIONS IS AS FOLLOWS:

OPTION	MEANING
BRIEF	SORT PROGRAM MESSAGES ARE NOT PRINTED AT THE USER'S TERMINAL.
SPACE	ANY BLANK LINES ARE DELETED FROM THE SORT OUTPUT FILE.
MERGE	A MERGE OF PRESORTED FILES IS REQUESTED.

FILE SPECIFICATION:

RESPOND TO THE FIRST INQUIRY WITH THE INPUT FILE NAME, OUTPUT FILE NAME, AND THE NUMBER OF PAIRS OF COLUMNS (DEFAULT IS 1). TO SPECIFY MULTIPLE INPUT FILES, RECORD TYPES, OR MAXIMUM RECORD LENGTHS THE FOLLOWING KEYWORDS MUST BE USED.

KEYWORD	USAGE
-INPUTFILE NAME	NAME IS AN INPUT FILE TREENAME UP TO 80 CHARACTERS LONG. THIS KEYWORD MAY BE USED REPEATEDLY, ONCE FOR EACH OF THE INPUT FILES.
-OUTPUTFILE NAME	NAME IS THE OUTPUT FILE TREENAME
-KEYS N	N IS THE NUMBER OF KEYS. DEFAULT IS 1.
-INTYPE TYPE	TYPE IS THE TYPE OF THE INPUT RECORDS: COMPRESSED UNCOMPRESSED FIXED OR VARIABLE. THESE RECORD TYPES ARE DESCRIBED BELOW.
-OUTTYPE TYPE	TYPE IS THE OUTPUT RECORD TYPE.
-INLENGTH N	N IS THE MAXIMUM LENGTH OF THE INPUT RECORDS. THIS LENGTH MUST BE SPECIFIED FOR FIXED LENGTH INPUT RECORDS.
-OUTLENGTH N	N IS THE MAXIMUM LENGTH OF THE OUTPUT RECORDS. THIS LENGTH MUST BE SPECIFIED FOR FIXED LENGTH OUTPUT RECORDS.

RECORD TYPES:

COMPRESSED SOURCE: BLANK COMPRESSED RECORD DELIMITED BY A NEWLINE CHARACTER (:212). SOURCE LINES CANNOT CONTAIN DATA WHICH MAY BE INTERPRETED AS A BLANK COMPRESSION INDICATOR (:221) OR A NEWLINE.

UNCOMPRESSED SOURCE: UNCOMPRESSED RECORD DELIMITED BY A NEWLINE CHARACTER (:212), AND THUS CANNOT CONTAIN DATA WHICH MAY BE INTERPRETED AS A NEWLINE.

VARIABLE LENGTH: RECORD STORED WITH LENGTH IN FIRST WORD.

FIXED LENGTH: RECORD CONTAINING DATA ONLY, NO LENGTH INFORMATION. THE LENGTH MUST BE SPECIFIED USING THE -INLENGTH OR -OUTLENGTH KEYWORDS. IF A NEWLINE CHARACTER IS APPENDED TO EACH RECORD SO THAT THE FILE CAN BE EDITED, IT MUST BE INCLUDED IN THE CHARACTER COUNT.

DEFAULT DEPENDS ON THE KEY TYPES SPECIFIED. INPUT TYPE DEFAULTS TO VARIABLE LENGTH IF ANY INTEGER, LONG INTEGER, SINGLE OR DOUBLE PRECISION REAL KEY IS SPECIFIED, OTHERWISE IT DEFAULTS TO ASCII. IF THE OUTPUT TYPE IS NOT GIVEN, IT IS ASSUMED TO BE THE SAME AS THE INPUT TYPE.

IF MULTIPLE INPUT FILES ARE USED, THEY MUST ALL CONTAIN THE SAME TYPE OF RECORDS.

KEYS SPECIFICATION:

RESPOND TO THE SUBSEQUENT INQUIRIES WITH THE APPROPRIATE STARTING AND ENDING COLUMN NUMBERS (CHARACTER POSITIONS). TO SORT IN DESCENDING ORDER, TYPE A SPACE AND THE LETTER R AFTER ENDING COLUMN OF THE FIELD TO BE REVERSE-SORTED. TO SPECIFY OTHER THAN ASCII KEYS, ENTER THE

PROPER CODE AT THE END OF THE LINE. ALTERNATIVELY, KEY INFORMATION MAY BE SPECIFIED BY USING CONTROL ARGUMENTS AS FOLLOWS:

CONTROL ARGUMENTS	USAGE
-START N	N IS STARTING COLUMN.
-END N	N IS ENDING COLUMN.
-DESCENDING	REQUESTS SORT IN DESCENDING ORDER.
-TYPE CODE	CODE CAN BE ONE OF:
	A - ASCII
	I - INTEGER
	J - LONG INTEGER
	F - SINGLE PRECISION REAL
	D - DOUBLE PRECISION REAL
	U - NUMERIC ASCII, UNSIGNED
	LS - LEADING SEPARATE
	TS - TRAILING SEPARATE
	LE - LEADING EMBEDDED
	TE - TRAILING EMBEDDED
	PD - PACKED DECIMAL
	AU - ASCII UPPERCASED WHEN COMPARED

WHEN THE LAST PAIR OF COLUMN NUMBERS IS ENTERED, SORTING BEGINS. WHEN THE SORT IS COMPLETE, SORT PRINTS THE NUMBER OF PASSES AND NUMBER OF ITEMS (LINES IN THE OUTPUT FILE), AND RETURNS TO PRIMOS.

MERGING:

AFTER ENTERING THE KEY INFORMATION, THE SORT PROGRAM ASKS FOR THE NUMBER OF ADDITIONAL MERGE FILES. THE USER THEN ENTERS THE MERGE FILENAMES, ONE PER LINE. WHEN THE LAST TREENAME IS ENTERED, MERGING BEGINS. WHEN THE MERGE IS COMPLETE, SORT PRINTS ONLY THE NUMBER OF PASSES BEFORE RETURNING TO PRIMOS.

EXAMPLES:

IN THE SAMPLE SORT RUNS BELOW, THE FOLLOWING INPUT FILES WILL BE USED.

```
OK, SLIST INPUT1
THIS
IS
A
SAMPLE
INPUT
FILE

OK, SLIST INPUT2
THIS            *
IS              *
A               *
SECOND          *
SAMPLE          *
INPUT           *
FILE            *

OK,
```

IN THE MERGE EXAMPLES, THE FOLLOWING PRESORTED COPIES OF THESE INPUT FILES WILL BE USED.

```
OK, SLIST MERGE1
A
FILE
INPUT
IS
SAMPLE
THIS

OK, SLIST MERGE2
A               *
FILE            *
INPUT           *
IS              *
SAMPLE          *
SECOND          *
THIS            *

OK,
```

THIS IS AN EXAMPLE OF SORTING A SINGLE INPUT FILE.

```
OK, SORT
SORT PROGRAM PARAMETERS ARE:
  INPUT TREE NAME -- OUTPUT TREE NAME FOLLOWED BY
  NUMBER OF PAIRS OF STARTING AND ENDING COLUMNS.
INPUT1_OUTPUT
  INPUT PAIRS OF STARTING AND ENDING COLUMNS
  ONE PAIR PER LINE--SEPARATED BY A SPACE.
  FOR REVERSE SORTING ENTER "R" AFTER DESIRED
  ENDING COLUMN--SEPARATED BY A SPACE.
  FOR A SPECIFIC DATA TYPE ENTER THE PROPER CODE
  AT THE END OF THE LINE--SEPARATED BY A SPACE.
     "A"  - ASCII
     "I"  - SINGLE PRECISION INTEGER
     "F"  - SINGLE PRECISION REAL
     "D"  - DOUBLE PRECISION REAL
     "J"  - DOUBLE PRECISION INTEGER
     "U"  - NUMERIC ASCII,UNSIGNED
     "LS" - NUMERIC ASCII,LEADING SEPARATE SIGN
     "TS" - NUMERIC ASCII,TRAILING SEPARATE SIGN
     "LE" - NUMERIC ASCII,LEADING EMBEDDED SIGN
     "TE" - NUMERIC ASCII,TRAILING EMBEDDED SIGN
     "PD" - PACKED DECIMAL
     "AU" - ASCII, UPPER & LOWER CASE SORT EQUAL
  DEFAULT IS ASCII.
1_5

BEGINNING SORT


 PASSES       2          ITEMS    .     6

[SORT-REV17.0]

OK, SLIST OUTPUT
A
FILE
INPUT
IS
SAMPLE
THIS

OK,
```

THIS IS AN EXAMPLE OF SORTING A SINGLE FILE IN DESCENDING ORDER AND THE USE OF THE BRIEF OPTION.

```
OK, SORT -BR
-INPUT INPUT1 -OUTPUT OUTFILE
-START 1 -END 5 -DESCENDING

BEGINNING SORT


 PASSES        2          ITEMS         6

[SORT-REV17.0]

OK, SLIST OUTFILE
THIS
SAMPLE
IS
INPUT
FILE
A

OK,
```

THIS IS AN EXAMPLE OF SORTING TWO FILES INTO A SINGLE OUTPUT FILE.
NOTE THAT THE ORDER OF INPUT IS MAINTAINED FOR RECORDS WITH EQUAL KEYS.

```
OK, SORT
SORT PROGRAM PARAMETERS ARE:
  INPUT TREE NAME -- OUTPUT TREE NAME FOLLOWED BY
  NUMBER OF PAIRS OF STARTING AND ENDING COLUMNS.
-INPUT INPUT1 -INPUT INPUT2 -OUTPUT OUTFILE
  INPUT PAIRS OF STARTING AND ENDING COLUMNS
  ONE PAIR PER LINE--SEPARATED BY A SPACE.
  FOR REVERSE SORTING ENTER "R" AFTER DESIRED
  ENDING COLUMN--SEPARATED BY A SPACE.
  FOR A SPECIFIC DATA TYPE ENTER THE PROPER CODE
  AT THE END OF THE LINE--SEPARATED BY A SPACE.
     "A"  - ASCII
     "I"  - SINGLE PRECISION INTEGER
     "F"  - SINGLE PRECISION REAL
     "D"  - DOUBLE PRECISION REAL
     "J"  - DOUBLE PRECISION INTEGER
     "U"  - NUMERIC ASCII,UNSIGNED
     "LS" - NUMERIC ASCII,LEADING SEPARATE SIGN
     "TS" - NUMERIC ASCII,TRAILING SEPARATE SIGN
     "LE" - NUMERIC ASCII,LEADING EMBEDDED SIGN
     "TE" - NUMERIC ASCII,TRAILING EMBEDDED SIGN
     "PD" - PACKED DECIMAL
     "AU" - ASCII, UPPER & LOWER CASE SORT EQUAL
  DEFAULT IS ASCII.
1 5 A

BEGINNING SORT


 PASSES       2          ITEMS       13

[SORT-REV17.0]

OK, SLIST OUTFILE
A
A              *
FILE
FILE           *
INPUT
INPUT          *
IS
IS             *
SAMPLE
SAMPLE         *
SECOND         *
THIS
THIS           *
```

THESE ARE TWO WAYS OF MERGING TWO PRESORTED FILES.
NOTE THAT THE ORDER OF INPUT IS MAINTAINED FOR RECORDS WITH EQUAL KEYS.

```
OK, SORT_-MERGE
SORT PROGRAM PARAMETERS ARE:
  INPUT TREE NAME -- OUTPUT TREE NAME FOLLOWED BY
  NUMBER OF PAIRS OF STARTING AND ENDING COLUMNS.
-INPUT_MERGE1_-INPUT_MERGE2_-OUTPUT_MERGEOUT_-KEYS_1
  INPUT PAIRS OF STARTING AND ENDING COLUMNS
  ONE PAIR PER LINE--SEPARATED BY A SPACE.
  FOR REVERSE SORTING ENTER "R" AFTER DESIRED
  ENDING COLUMN--SEPARATED BY A SPACE.
  FOR A SPECIFIC DATA TYPE ENTER THE PROPER CODE
  AT THE END OF THE LINE--SEPARATED BY A SPACE.
     "A"  - ASCII
     "I"  - SINGLE PRECISION INTEGER
     "F"  - SINGLE PRECISION REAL
     "D"  - DOUBLE PRECISION REAL
     "J"  - DOUBLE PRECISION INTEGER
     "U"  - NUMERIC ASCII,UNSIGNED
     "LS" - NUMERIC ASCII,LEADING SEPARATE SIGN
     "TS" - NUMERIC ASCII,TRAILING SEPARATE SIGN
     "LE" - NUMERIC ASCII,LEADING EMBEDDED SIGN
     "TE" - NUMERIC ASCII,TRAILING EMBEDDED SIGN
     "PD" - PACKED DECIMAL
     "AU" - ASCII, UPPER & LOWER CASE SORT EQUAL
  DEFAULT IS ASCII.
1_5
INPUT THE NUMBER OF ADDITIONAL FILES TO BE MERGED. (MAX=     9): 0

BEGINNING MERGE



 PASSES        1
[SORT-REV17.0]

OK, SLIST MERGEOUT
A
A               *
FILE
FILE            *
INPUT
INPUT           *
IS
IS              *
SAMPLE
SAMPLE          *
SECOND          *
THIS
THIS            *
```

```
OK, SORT -MERGE
SORT PROGRAM PARAMETERS ARE:
  INPUT TREE NAME -- OUTPUT TREE NAME FOLLOWED BY
  NUMBER OF PAIRS OF STARTING AND ENDING COLUMNS.
-OUTPUT MERGEOUT
  INPUT PAIRS OF STARTING AND ENDING COLUMNS
  ONE PAIR PER LINE--SEPARATED BY A SPACE.
  FOR REVERSE SORTING ENTER "R" AFTER DESIRED
  ENDING COLUMN--SEPARATED BY A SPACE.
  FOR A SPECIFIC DATA TYPE ENTER THE PROPER CODE
  AT THE END OF THE LINE--SEPARATED BY A SPACE.
     "A"  - ASCII
     "I"  - SINGLE PRECISION INTEGER
     "F"  - SINGLE PRECISION REAL
     "D"  - DOUBLE PRECISION REAL
     "J"  - DOUBLE PRECISION INTEGER
     "U"  - NUMERIC ASCII,UNSIGNED
     "LS" - NUMERIC ASCII,LEADING SEPARATE SIGN
     "TS" - NUMERIC ASCII,TRAILING SEPARATE SIGN
     "LE" - NUMERIC ASCII,LEADING EMBEDDED SIGN
     "TE" - NUMERIC ASCII,TRAILING EMBEDDED SIGN
     "PD" - PACKED DECIMAL
     "AU" - ASCII, UPPER & LOWER CASE SORT EQUAL
  DEFAULT IS ASCII.
1,5
INPUT THE NUMBER OF ADDITIONAL FILES TO BE MERGED. (MAX=    11): 2
  INPUT FILES TO BE MERGED, ONLY ONE PER LINE.
MERGE1
MERGE2

BEGINNING MERGE



 PASSES       1
[SORT-REV17.0]

OK, SLIST MERGEOUT
A
A              *
FILE
FILE           *
INPUT
INPUT          *
IS
IS             *
SAMPLE
SAMPLE         *
SECOND         *
THIS
THIS           *
```

```
**********************************************************************************
**********************************************************************************
**
**
**
**      JJJ    III    M    M   M    M   Y    Y
**       J      I     MM  MM   MM  MM   Y    Y
**       J      I     M M M    M M M     Y Y
**       J      I     M M M    M M M      Y
**    J  J      I     M    M   M    M     Y
**    J  J      I     M    M   M    M     Y
**     JJ      III    M    M   M    M     Y
**
**
**
**    RRRR    V    V   SSS    RRRR    TTTTT  L         III
**    R   R   V    V  S   S   R   R     T    L          I
**    R   R   V    V  S       R   R     T    L          I
**    RRRR    V  V     SSS    RRRR      T    L          I
**    R R     V V          S  R R       T    L          I
**    R  R    VV       S   S  R  R      T    L    L     I
**    R   R   V        SSS    R   R     T    LLLLL   III
**
**
**
**********************************************************************************
**********************************************************************************
```

V-MODE SORT LIBRARY (VSRTLI)

DATE: MARCH 29, 1979

TO:

FROM:

SUBJECT: V-MODE SORT LIBRARY (VSRTLI)

REFERENCE: NONE

ABSTRACT

THE V-MODE SORT LIBRARY HAS SEVERAL ENHANCEMENTS FOR REV 17.

IT NOW HANDLES THE FOLLOWING DATA TYPES AS SORT FIELDS (KEYS):

- NUMERIC ASCII, LEADING SEPARATE SIGN
- NUMERIC ASCII, TRAILING SEPARATE SIGN
- NUMERIC ASCII, LEADING EMBEDDED SIGN
- NUMERIC ASCII, TRAILING EMBEDDED SIGN
- NUMERIC ASCII, UNSIGNED
- PACKED DECIMAL.

TWO KEY TYPES FOR ASCII KEYS GIVE USERS THE OPTION OF HAVING LOWER CASE ALPHABETIC CHARACTERS SORT EQUAL TO UPPER CASE.

MAXIMUM RECORD LENGTH HAS BEEN INCREASED FROM 14998 TO 32760 BYTES.

KEYS MAY BE AS LONG AS THE INPUT RECORDS (THE PREVIOUS LIMIT WAS 316 BYTES). THE MAXIMUM NUMBER OF KEYS HAS BEEN CHANGED FROM 20 TO 50.

NEW SORT-ONLY AND MERGE-ONLY ROUTINES HAVE BEEN ADDED WITH SOME ADDITIONAL FUNCTIONALITY:

RECORD TYPE MAY BE EXPLICITLY SET, RATHER THAN DEFAULTING FROM THE SPECIFIED KEY TYPES. THIS GIVES FLEXIBILITY TO HANDLE NEW RECORD TYPES. FOR EXAMPLE, THE NEW LIBRARY ROUTINES WILL NOW HANDLE FIXED LENGTH RECORDS, AND BLANK COMPRESSED RECORDS WILL BE TREATED DIFFERENTLY THAN UNCOMPRESSED RECORDS.

THE MERGE OPTION IS NOW A TRUE MERGE RATHER THAN A SORT OF MULTIPLE INPUT FILES.

UP TO 20 FILES MAY BE SORTED INTO A SINGLE OUTPUT FILE.

SPECIFICATION OF OPEN UNITS ALLOWS I/O TO AND FROM SEGMENT DIRECTORIES.

******** ********
VSRTLI *VSRTLI*
******** ********

INTRODUCTION

VSRTLI IS A V-MODE SORT LIBRARY WHICH CONTAINS A SET OF SUBROUTINES TO PERFORM SORT AND MERGE OPERATIONS ON FILES.

WHEN SORTING OR MERGING, RECORDS WITH EQUAL KEYS WILL RETAIN THEIR ORIGINAL RELATIVE ORDER.

NAMING CONVENTIONS

VSRTLI ROUTINES FOLLOW A CONSISTENT NAMING CONVENTION DESIGNED TO AVOID THE POSSIBILITY OF CONFLICT BOTH WITH USER WRITTEN ROUTINES AND SYSTEM ROUTINES. FOR COMPATIBILITY WITH EARLIER VERSIONS OF THE LIBRARY, THE EXISTING ENTRY POINTS: SUBSRT, ASCS$$, AND ASCSRT WILL NOT BE CHANGED. ALL OTHER NAMES WILL END WITH THE SUFFIX "$S".

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

THE USE OF COMMON BLOCKS HAS CHANGED AS OF REV 17. TO AVOID PROBLEMS DUE TO USER DECLARATIONS OF THE COMMON BLOCKS EB$1, EB$2, EB$3, EB$4, AND EB$5, THESE NAMES ARE NO LONGER USED.

RECORDS

SINCE THE TERMS 'ASCII FILE' AND 'BINARY FILE' HAVE DIFFERENT MEANINGS TO VARIOUS USERS, THE FOLLOWING (HOPEFULLY) UNAMBIGUOUS TERMS ARE DEFINED HERE AND WILL BE USED BY THE SORT PROGRAM AND LIBRARY.

RECORD TYPES

THE SORT LIBRARY ROUTINES WILL HANDLE THE FOLLOWING RECORD TYPES:

COMPRESSED SOURCE - BLANK COMPRESSED RECORD DELIMITED BY A NEWLINE CHARACTER (:212). COMPRESSED SOURCE LINES CANNOT CONTAIN DATA WHICH MAY BE INTERPRETED AS A BLANK COMPRESSION INDICATOR (:221) OR A NEWLINE.

UNCOMPRESSED SOURCE - RECORD WITH NO BLANK COMPRESSION AND DELIMITED BY A NEWLINE CHARACTER (:212), AND THUS CANNOT CONTAIN DATA WHICH MAY BE INTERPRETED AS A NEWLINE.

VARIABLE LENGTH - RECORD STORED WITH LENGTH IN FIRST WORD.
FIXED LENGTH - RECORD CONTAINING DATA ONLY, NO LENGTH INFORMATION. THE LENGTH MUST BE PASSED AS THE MAXIMUM LINE SIZE. IF A NEWLINE CHARACTER IS APPENDED TO EACH RECORD SO THAT THE FILE CAN BE EDITED, IT MUST BE INCLUDED IN THE CHARACTER COUNT.

DEFAULT DEPENDS ON THE KEY TYPES SPECIFIED. INPUT TYPE DEFAULTS TO VARIABLE LENGTH IF AN INTEGER, LONG INTEGER, SINGLE OR DOUBLE PRECISION REAL KEY IS SPECIFIED; OTHERWISE IT DEFAULTS TO COMPRESSED SOURCE. IF THE OUTPUT TYPE IS NOT SPECIFIED, IT IS ASSUMED TO BE THE SAME AS THE INPUT TYPE. IF MULTIPLE INPUT FILES ARE USED, THEY MUST ALL CONTAIN RECORDS OF THE SAME TYPE.

RECORD LENGTH
THE MAXIMUM RECORD LENGTH ALLOWED IS 32760 BYTES (CHARACTERS).

KEYS

UP TO 50 KEY FIELDS MAY BE SPECIFIED, WITH A TOTAL LENGTH NOT TO EXCEED THE MAXIMUM RECORD LENGTH. EACH KEY MUST START AND END ON A BYTE BOUNDARY.

KEY TYPES
THE FOLLOWING KEY TYPES ARE SUPPORTED:

ASCII
INTEGER(SHORT)
INTEGER(LONG)
REAL
DOUBLE PRECISION
NUMERIC ASCII, LEADING SEPARATE SIGN
NUMERIC ASCII, TRAILING SEPARATE SIGN
NUMERIC ASCII, LEADING EMBEDDED SIGN
NUMERIC ASCII, TRAILING EMBEDDED SIGN
NUMERIC ASCII, UNSIGNED
PACKED DECIMAL
ASCII, LOWER CASE SORTS EQUAL TO UPPER CASE

DESCRIPTIONS OF THESE TYPES FOLLOW.

ASCII KEYS ARE CHARACTER STRINGS. THEY ARE STORED ONE CHARACTER PER BYTE, AND ARE LIMITED ONLY BY THE LENGTH OF THE RECORD.

THE INTEGER AND REAL DATA TYPES ARE SUMMARIZED IN THE TABLE BELOW.

MODE	BYTE LENGTH	RANGE
INTEGER(SHORT)	2	-32767 TO +32767
INTEGER(LONG)	4	-2**31 TO +2**31-1
REAL	4	±(10**-38 TO 10**38)
DOUBLE PRECISION	8	±(10**-9902 TO 10**9825)

UNSIGNED NUMERIC ASCII KEYS ARE STORED ONE DIGIT PER BYTE, AND ARE LIMITED ONLY BY THE LENGTH OF THE RECORD.

THE SIGNED NUMERIC ASCII KEYS REQUIRE ONE BYTE PER DIGIT AND DIFFER ONLY IN THE PLACEMENT OF THE SIGN. THE SIGN CAN EITHER BE LEADING OR TRAILING THE NUMERIC FIELD, AND EITHER IN A SEPARATE CHARACTER POSITION OR EMBEDDED IN A DIGIT. IF THE SIGN IS SEPARATE, A PLUS SIGN (+) IS USED TO REPRESENT A POSITIVE NUMBER AND A MINUS SIGN (-) IS USED TO REPRESENT A NEGATIVE NUMBER. A SPACE WILL BE TREATED AS A POSITIVE SIGN. IF THE SIGN IS EMBEDDED, A SINGLE CHARACTER IS USED TO REPRESENT A DIGIT AND THE SIGN OF THE FIELD. EMBEDDED SIGN CHARACTERS ARE AS FOLLOWS:

DIGIT	POSITIVE	NEGATIVE
0	0,-,+,:	=,-
1	1 A	J
2	2 B	K
3	3 C	L
4	4 D	M
5	5 E	N
6	6 F	O
7	7 G	P
8	8 H	Q
9	9 I	R

SIGNED NUMERIC ASCII KEYS MAY BE UP TO 63 DIGITS PLUS SIGN.

PACKED NUMERIC FIELDS USE A FOUR BIT NIBBLE TO REPRESENT EACH DIGIT FOLLOWED BY A SIGN NIBBLE. A POSITIVE SIGN IS REPRESENTED BY HEX C IN THE SIGN NIBBLE, AND A NEGATIVE FIELD HAS A HEX D IN THE SIGN NIBBLE. A PACKED FIELD MUST HAVE AN ODD NUMBER OF DIGITS PLUS THE SIGN; SINCE THEY ARE STORED TWO NIBBLES (DIGIT OR SIGN) PER BYTE, THIS COMES OUT TO A FULL NUMBER OF BYTES. PACKED DECIMAL KEYS MAY BE UP TO 63 DIGITS PLUS SIGN.

NONEXISTENT KEYS

NONEXISTENT KEYS WOULD OCCUR WHEN THE ACTUAL LENGTH OF A RECORD IS LESS THAN THE ENDING BYTE NUMBER OF A KEY (I.E. THE RECORD IS TOO SHORT). NONEXISTENT KEYS ARE HANDLED DIFFERENTLY FOR THE DIFFERENT RECORD TYPES. THE KEY IS UNDEFINED FOR UNCOMPRESSED SOURCE OR VARIABLE LENGTH RECORDS (AT PRESENT NO ERROR MESSAGE IS GIVEN). COMPRESSED SOURCE RECORDS PAD THE KEY WITH SPACES. THIS CONDITION CANNOT OCCUR WITH FIXED LENGTH RECORDS SINCE THE KEY FIELDS ARE CHECKED TO BE WITHIN THE RECORD LENGTH.

THE ROUTINES

BELOW ARE DETAILED DESCRIPTIONS OF THE ROUTINES IN VSRTLI. NO ARGUMENTS PASSED TO A SORT ROUTINE WILL BE CHANGED EXCEPT THOSE LISTED AS "RETURNED" ARGUMENTS.

SUBSRT

SUBSRT IS THE EARLIEST VERSION OF A DISK ORIENTED SORT ROUTINE. IT HAS A SIMPLER CALLING SEQUENCE THAN ASCS$$, AND THUS LESS FUNCTIONALITY. SUBSRT CAN BE USED TO SORT A SINGLE INPUT FILE, CONTAINING COMPRESSED SOURCE RECORDS, ON ASCII KEYS IN ASCENDING ORDER.

CALLING SEQUENCE:

```
CALL SUBSRT(TREE1,LEN1,TREE2,LEN2,NUMKEY,NSTART,NEND,NPASS,NITEM)
```

PARAMETERS:

SUBSRT PARAMETERS ARE DEFINED AS IN ASCS$$.

ASCS$$ (ALSO CALLABLE AS ASCSRT)

ASCS$$ HAS THE ABILITY TO SORT OR MERGE COMPRESSED SOURCE OR VARIABLE LENGTH RECORDS FROM AND TO DISK FILES. ANY OF THE SUPPORTED KEY TYPES MAY BE USED, AND THERE MAY BE ASCENDING AND DESCENDING KEYS WITHIN THE SAME SORT (MERGE).

CALLING SEQUENCE:

```
CALL ASCS$$(TREE1,LEN1,TREE2,LEN2,NUMKEY,NSTART,NEND,NPASS,
            NITEM,NREV,ISPCE,MGCNT,MGBUFF,LEN,LOC(PBUFF),
            MSIZE,NTYPE,LINSIZ,NUNITS,UNITS)
```

PARAMETERS:

TREE1	INPUT FILE TREENAME.
LEN1	LENGTH OF INPUT TREENAME IN CHARACTERS, UP TO 80.
TREE2	OUTPUT FILE TREENAME.
LEN2	LENGTH OF OUTPUT TREENAME IN CHARACTERS, UP TO 80.
NUMKEY	NUMBER OF PAIRS OF STARTING AND ENDING COLUMNS (STARTING AND ENDING BYTES IF BINARY). UP TO 20. DEFAULT = 1.
NSTART	VECTOR CONTAINING STARTING COLUMNS/BYTES. EACH STARTING COLUMN MUST BE ≥ 1.
NEND	VECTOR CONTAINING ENDING COLUMNS/BYTES. EACH ENDING COLUMN MUST BE $\leq$ LINSIZ.
NPASS	NUMBER OF PASSES (RETURNED).

NITEM	NUMBER OF ITEMS (RETURNED). (INTEGER*4).
NREV	VECTOR CONTAINING SORT ORDER FOR EACH KEY 0 = ASCENDING 1 = DESCENDING. DEFAULT = 0 = ASCENDING.
ISPCE	SPACE OPTION 0 = INCLUDE BLANK LINES IN SORT 1 = DELETE BLANK LINES. DEFAULT = 0 = INCLUDE BLANK LINES.
MGCNT	NUMBER OF MERGE FILES (UP TO 10).
MGBUFF	ARRAY DIMENSIONED(40,MGCNT) CONTAINING MERGE FILENAMES.
LEN	VECTOR CONTAINING LENGTHS OF MERGE TREENAMES IN CHARACTERS, UP TO 80.
PBUFF	OBSOLETE.
MSIZE	SIZE OF PRESORT BUFFER IN WORDS, <65535. THIS MAY BE A FULL 16-BIT UNSIGNED INTEGER. IF NONZERO, MSIZE MUST BE AT LEAST 1024 (ONE PAGE). DEFAULT IS ONE SEGMENT (65536 WORDS).
NTYPE	VECTOR CONTAINING TYPE OF EACH KEY 1 = ASCII 2 = SINGLE PRECISION INTEGER 3 = SINGLE PRECISION REAL 4 = DOUBLE PRECISION REAL 5 = DOUBLE PRECISION INTEGER 6 = NUMERIC ASCII,LEADING SEPARATE SIGN 7 = NUMERIC ASCII,TRAILING SEPARATE SIGN 8 = PACKED DECIMAL 9 = NUMERIC ASCII,LEADING EMBEDDED SIGN 10 = NUMERIC ASCII,TRAILING EMBEDDED SIGN 11 = NUMERIC ASCII,UNSIGNED 12 = ASCII, LOWER CASE SORTS EQUAL TO UPPER CASE. DEFAULT = ALL ASCII KEYS.
LINSIZ	MAXIMUM LINE SIZE IN CHARACTERS (BYTES). DEFAULT = 32760.
NUNITS	OBSOLETE.
UNITS	OBSOLETE.

NOTES:

1. PARAMETERS ARE ALL INTEGER*2 EXCEPT NITEM WHICH IS INTEGER*4. MSIZE MAY BE A FULL 16-BIT UNSIGNED INTEGER, BUT IT CANNOT BE INTEGER*4.
2. THE LAST FOUR PARAMETERS ARE OPTIONAL.
3. PRESORT BUFFER IS NOW A COMMON BLOCK, PSRT$S. THUS, THE PARAMETER PBUFF IS OBSOLETE AND THE PARAMETER MSIZE CORRESPONDS TO THE SIZE OF THIS NEW COMMON BLOCK, NOT TO THE SIZE OF THE BUFFER IDENTIFIED BY PBUFF.
4. FILE UNITS ARE SUPPLIED DYNAMICALLY USING THE SRCH$$ KEY K$GETU. THUS. THE PARAMETERS NUNITS AND UNITS ARE NOW OBSOLETE.

SRTF$S

THIS ROUTINE WILL SORT ONE TO TWENTY INPUT FILES INTO A SINGLE OUTPUT FILE.

CALLING SEQUENCE:

```
CALL SRTF$S(INBUFF,INLEN,INUNTS,INCNT,TREE2,LEN2,OUTUNT,
            NUMKEY,NSTART,NEND,NREV,NTYPE,
            ERCODE,INREC,OUTREC,SPCLS,MSIZE)
```

PARAMETERS:

INBUFF	ARRAY DIMENSIONED(40,INCNT) CONTAINING INPUT FILENAMES.
INLEN	VECTOR CONTAINING LENGTHS OF INPUT TREENAMES IN CHARACTERS, UP TO 80.
INUNTS	VECTOR CONTAINING INPUT FILE UNITS(IF OPEN UNITS ARE USED).
INCNT	NUMBER OF INPUT FILES (UP TO 20).
TREE2	OUTPUT FILE TREENAME.
LEN2	LENGTH OF OUTPUT TREENAME IN CHARACTERS, UP TO 80.
OUTUNT	OUTPUT FILE UNIT (IF AN OPEN UNIT IS USED).
NUMKEY	NUMBER OF PAIRS OF STARTING AND ENDING COLUMNS (STARTING AND ENDING BYTES IF BINARY), UP TO 20. DEFAULT = 1.
NSTART	VECTOR CONTAINING STARTING COLUMNS/BYTES. EACH STARTING COLUMN MUST BE ≥1.
NEND	VECTOR CONTAINING ENDING COLUMNS/BYTES. EACH ENDING COLUMN MUST BE ≤INREC(2).
NREV	VECTOR CONTAINING SORT ORDER FOR EACH KEY 0 = ASCENDING 1 = DESCENDING. DEFAULT = 0 = ASCENDING.
NTYPE	VECTOR CONTAINING TYPE OF EACH KEY 1 = ASCII 2 = SINGLE PRECISION INTEGER 3 = SINGLE PRECISION REAL 4 = DOUBLE PRECISION REAL 5 = DOUBLE PRECISION INTEGER 6 = NUMERIC ASCII,LEADING SEPARATE SIGN 7 = NUMERIC ASCII,TRAILING SEPARATE SIGN 8 = PACKED DECIMAL 9 = NUMERIC ASCII,LEADING EMBEDDED SIGN 10 = NUMERIC ASCII,TRAILING EMBEDDED SIGN 11 = NUMERIC ASCII,UNSIGNED 12 = ASCII, LOWER CASE SORTS EQUAL TO UPPER CASE. DEFAULT = ALL ASCII KEYS.
ERCODE	ERROR CODE (RETURNED).
INREC	FIVE WORD ARRAY CONTAINING INPUT RECORD INFORMATION: INREC(1) = INPUT RECORD TYPE 1 = COMPRESSED SOURCE (BLANKS COMPRESSED) 2 = VARIABLE LENGTH 3 = FIXED LENGTH (INREC(2) MUST BE SPECIFIED) 4 = UNCOMPRESSED SOURCE (NO BLANK COMPRESSION). DEFAULT DEPENDS ON THE KEY TYPES SPECIFIED IN ARGUMENT

	NTYPE (AS DESCRIBED ON PAGE 3).
	INREC(2) = MAXIMUM INPUT LINE SIZE IN CHARACTERS (BYTES).
	DEFAULT = 32760.
	REQUIRED FOR SORTING FIXED LENGTH RECORDS.
	IF LINSIZ IS LARGER THAN DEFAULT VALUE, SEE THE SECTION
	ON VARIABLE RECORD SIZE.
	INREC(3-5) MUST BE ZERO, AND ARE RESERVED FOR FUTURE USE.
OUTREC	FIVE WORD ARRAY CONTAINING OUTPUT RECORD INFORMATION:
	OUTREC(1) = OUTPUT RECORD TYPE (SEE INREC)
	OUTREC(2) = MAXIMUM OUTPUT LINE SIZE IN CHARACTERS(BYTES).
	OUTREC(3-5) MUST BE ZERO, AND ARE RESERVED FOR FUTURE USE.
SPCLS	FIVE WORD ARRAY CONTAINING:
	SPCLS(1) = SPACE OPTION
	0 = INCLUDE BLANK LINES IN SORT
	1 = DELETE BLANK LINES.
	DEFAULT = 0 = INCLUDE BLANK LINES.
	SPCLS(2-5) MUST BE ZERO, AND ARE RESERVED FOR FUTURE USE.
MSIZE	SIZE OF PRESORT BUFFER IN PAGES(UNITS OF 1024 WORDS), $\leq$64.
	DEFAULT IS ONE SEGMENT (64 PAGES).

THE FOLLOWING FIVE ROUTINES ALLOW FOR USER'S OWN INPUT AND OUTPUT PROCEDURES. THESE ROUTINES MUST ALL BE CALLED, AND IN THE ORDER GIVEN, TO ASSURE THAT THE SORT IS DONE CORRECTLY.

SETU$S

THIS ROUTINE CHECKS THE PARAMETERS SUPPLIED BY THE USER AND SETS UP ALL TABLES RELEVANT TO THE PARTICULAR APPLICATION.

CALLING SEQUENCE:

```
CALL SETU$S(INBUFF,INLEN,INUNTS,INCNT,TREE2,LEN2,OUTUNT,
            NUMKEY,NSTART,NEND,NREV,NTYPE,ERCODE,
            INREC,OUTREC,SPCLS,MSIZE,IPROC,OPROC)
```

PARAMETERS:

INBUFF	ARRAY DIMENSIONED(40,INCNT) CONTAINING INPUT FILENAMES.
INLEN	VECTOR CONTAINING LENGTHS OF INPUT TREENAMES IN CHARACTERS, UP TO 80.
INUNTS	VECTOR CONTAINING INPUT FILE UNITS(IF OPEN UNITS ARE USED).
INCNT	NUMBER OF INPUT FILES (UP TO 20).
TREE2	OUTPUT FILE TREENAME.
LEN2	LENGTH OF OUTPUT TREENAME IN CHARACTERS, UP TO 80.
OUTUNT	OUTPUT FILE UNIT (IF AN OPEN UNIT IS USED).
NUMKEY	NUMBER OF PAIRS OF STARTING AND ENDING COLUMNS (STARTING AND ENDING BYTES IF BINARY), UP TO 20. DEFAULT = 1.
NSTART	VECTOR CONTAINING STARTING COLUMNS/BYTES. EACH STARTING COLUMN MUST BE ≥1.
NEND	VECTOR CONTAINING ENDING COLUMNS/BYTES. EACH ENDING COLUMN MUST BE ≤LINSIZ.
NREV	VECTOR CONTAINING SORT ORDER FOR EACH KEY 0 = ASCENDING 1 = DESCENDING. DEFAULT = 0 = ASCENDING.
NTYPE	VECTOR CONTAINING TYPE OF EACH KEY 1 = ASCII 2 = SINGLE PRECISION INTEGER 3 = SINGLE PRECISION REAL 4 = DOUBLE PRECISION REAL 5 = DOUBLE PRECISION INTEGER 6 = NUMERIC ASCII,LEADING SEPARATE SIGN 7 = NUMERIC ASCII,TRAILING SEPARATE SIGN 8 = PACKED DECIMAL 9 = NUMERIC ASCII,LEADING EMBEDDED SIGN 10 = NUMERIC ASCII,TRAILING EMBEDDED SIGN 11 = NUMERIC ASCII,UNSIGNED 12 = ASCII, LOWER CASE SORTS EQUAL TO UPPER CASE. DEFAULT = ALL ASCII KEYS.

ERCODE	ERROR CODE (RETURNED).
INREC	FIVE WORD ARRAY CONTAINING INPUT RECORD INFORMATION: INREC(1) = INPUT RECORD TYPE 1 = COMPRESSED SOURCE (BLANKS COMPRESSED) 2 = VARIABLE LENGTH 3 = FIXED LENGTH (LINSIZ MUST BE SPECIFIED) 4 = UNCOMPRESSED SOURCE (NO BLANK COMPRESSION). DEFAULT DEPENDS ON THE KEY TYPES SPECIFIED IN ARGUMENT NTYPE (AS DESCRIBED ON PAGE 3). INREC(2) = MAXIMUM INPUT LINE SIZE IN CHARACTERS (BYTES). DEFAULT = 32760. REQUIRED FOR SORTING FIXED LENGTH RECORDS. INREC(3-5) MUST BE ZERO, AND ARE RESERVED FOR FUTURE USE.
OUTREC	FIVE WORD ARRAY CONTAINING OUTPUT RECORD INFORMATION: OUTREC(1) = OUTPUT RECORD TYPE (SEE INREC) OUTREC(2) = MAXIMUM OUTPUT LINE SIZE IN CHARACTERS(BYTES). OUTREC(3-5) MUST BE ZERO, AND ARE RESERVED FOR FUTURE USE.
SPCLS	FIVE WORD ARRAY CONTAINING: SPCLS(1) = SPACE OPTION 0 = INCLUDE BLANK LINES IN SORT 1 = DELETE BLANK LINES. DEFAULT = 0 = INCLUDE BLANK LINES. SPCLS(2-5) MUST BE ZERO, AND ARE RESERVED FOR FUTURE USE.
MSIZE	SIZE OF PRESORT BUFFER IN PAGES(UNITS OF 1024 WORDS), $\leq$64. DEFAULT IS ONE SEGMENT (64 PAGES).
IPROC	INPUT DATA SOURCE 0 = INPUT FILE 1 = INPUT PROCEDURE.
OPROC	OUTPUT DATA DESTINATION 0 = OUTPUT FILE 1 = OUTPUT PROCEDURE.

RLSE$S

THIS ROUTINE TRANSFERS RECORDS TO THE INITIAL PHASE OF THE SORT. IF AN INPUT PROCEDURE IS USED, RLSE$S IS CALLED ONCE FOR EACH LINE RELEASED TO SORT. IF AN INPUT FILE IS USED, RLSE$S SHOULD BE CALLED ONLY ONCE.

CALLING SEQUENCE:

```
CALL RLSE$S(RLBUFF,LENGTH)
```

PARAMETERS:

RLBUFF	BUFFER CONTAINING NEXT RECORD FOR SORT.
LENGTH	LENGTH OF RECORD IN CHARACTERS OR BYTES. THIS IS NOT NECESSARILY THE FULL LENGTH OF RLBUFF.

CMBN$S

THIS ROUTINE PERFORMS THE FINAL INTERNAL SORT. IF THE SORT COULD NOT BE DONE IN MEMORY ALL AT ONCE, CMBN$S MERGES THE STRINGS PREVIOUSLY ORDERED.

CALLING SEQUENCE:

```
CALL CMBN$S
```

RTRN$S

THIS ROUTINE RETURNS THE SORTED RECORDS FROM THE FINAL PHASE OF THE SORT. IF AN OUTPUT FILE IS SPECIFIED, RTRN$S SHOULD BE CALLED ONCE. IF AN OUTPUT PROCEDURE IS SPECIFIED, EACH CALL TO RTRN$S OBTAINS THE NEXT SORTED RECORD.

CALLING SEQUENCE:

```
CALL RTRN$S(RTBUFF,LENGTH)
```

PARAMETERS:

RTBUFF	BUFFER CONTAINING NEXT SORTED RECORD (RETURNED). SHOULD BE LARGE ENOUGH TO HOLD THE LONGEST RECORD SORTED.
LENGTH	LENGTH OF RECORD IN CHARACTERS OR BYTES (RETURNED). WHEN ALL RECORDS HAVE BEEN RETURNED, CALLS TO RLSE$S RETURN A RECORD LENGTH OF 0.

CLNU$S

CLNU$S IS CALLED TO CLOSE ALL UNITS OPENED BY THE SORT ROUTINES AND TO DELETE ANY TEMPORARY FILES CREATED.

CALLING SEQUENCE:

```
CALL CLNU$S
```

NOTES:

1. ALL PARAMETERS ARE INTEGER*2.
2. IF INPUT IS FROM FILE(S), RLSE$S ARGUMENTS ARE NOT USED.
3. IF INPUT IS FROM FILE(S), MULTIPLE CALLS TO RLSE$S RESULTS IN MULTIPLE OCCURRENCES OF EACH RECORD WHEN SORTED.
4. IF OUTPUT IS TO A FILE, RTRN$S ARGUMENTS ARE NOT USED.

5. IF OUTPUT IS TO A FILE, MULTIPLE CALLS TO RTRN$S RETURN A RECORD LENGTH OF 0, WITH NO EFFECT ON THE OUTPUT FILE.

THE FOLLOWING THREE ROUTINES ARE FOR MERGING ONLY. AT PRESENT, NO OUTPUT PROCEDURES ARE ALLOWED, BUT THE ROUTINES NEEDED ARE DESCRIBED BELOW TO BE READILY AVAILABLE WHEN THAT FUNCTIONALITY IS ADDED.
WHEN OUTPUT IS TO FILES, ONLY THE ROUTINE MRG1$S SHOULD BE CALLED.
WHEN OUTPUT PROCEDURES ARE USED, THESE ROUTINES MUST ALL BE CALLED, AND IN THE ORDER GIVEN, TO ASSURE THAT THE MERGE IS DONE CORRECTLY. NO INPUT PROCEDURES ARE ALLOWED WHEN MERGING.

MRG1$S

THIS RCUTINE WILL MERGE ONE TO ELEVEN PREVIOUSLY SORTED FILES INTO A SINGLE OUTPUT FILE.

CALLING SEQUENCE:

```
CALL MRG1$S(INBUFF,INLEN,INUNTS,INCNT,TREE2,LEN2,OUTUNT,
            NUMKEY,NSTART,NEND,NREV,NTYPE,
            ERCODE,INREC,OUTREC,SPCLS,OPROC)
```

PARAMETERS:

INBUFF	ARRAY DIMENSIONED(40,INCNT) CONTAINING INPUT FILENAMES.
INLEN	VECTOR CONTAINING LENGTHS OF INPUT TREENAMES IN CHARACTERS, UP TO 80.
INUNTS	VECTOR CONTAINING INPUT FILE UNITS(IF OPEN UNITS ARE USED).
INCNT	NUMBER OF INPUT FILES (UP TO 20).
TREE2	OUTPUT FILE TREENAME.
LEN2	LENGTH OF OUTPUT TREENAME IN CHARACTERS, UP TO 80.
OUTUNT	OUTPUT FILE UNIT (IF AN OPEN UNIT IS USED).
NUMKEY	NUMBER OF PAIRS OF STARTING AND ENDING COLUMNS (STARTING AND ENDING BYTES IF BINARY), UP TO 20. DEFAULT = 1.
NSTART	VECTOR CONTAINING STARTING COLUMNS/BYTES. EACH STARTING COLUMN MUST BE ≥1.
NEND	VECTOR CONTAINING ENDING COLUMNS/BYTES. EACH ENDING COLUMN MUST BE ≤INREC(2).
NREV	VECTOR CONTAINING SORT ORDER FOR EACH KEY 0 = ASCENDING 1 = DESCENDING. DEFAULT = 0 = ASCENDING.
NTYPE	VECTOR CONTAINING TYPE OF EACH KEY 1 = ASCII 2 = SINGLE PRECISION INTEGER 3 = SINGLE PRECISION REAL 4 = DOUBLE PRECISION REAL 5 = DOUBLE PRECISION INTEGER 6 = NUMERIC ASCII,LEADING SEPARATE SIGN 7 = NUMERIC ASCII,TRAILING SEPARATE SIGN 8 = PACKED DECIMAL 9 = NUMERIC ASCII,LEADING EMBEDDED SIGN

	10 = NUMERIC ASCII,TRAILING EMBEDDED SIGN
	11 = NUMERIC ASCII,UNSIGNED
	12 = ASCII, LOWER CASE SORTS EQUAL TO UPPER CASE.
	DEFAULT = ALL ASCII KEYS.
ERCODE	ERROR CODE (RETURNED).
INREC	FIVE WORD ARRAY CONTAINING INPUT RECORD INFORMATION:
	INREC(1) = INPUT RECORD TYPE
	1 = COMPRESSED SOURCE (BLANKS COMPRESSED)
	2 = VARIABLE LENGTH
	3 = FIXED LENGTH (INREC(2) MUST BE SPECIFIED)
	4 = UNCOMPRESSED SOURCE (NO BLANK COMPRESSION).
	DEFAULT DEPENDS ON THE KEY TYPES SPECIFIED IN ARGUMENT NTYPE (AS DESCRIBED ON PAGE 3).
	INREC(2) = MAXIMUM INPUT LINE SIZE IN CHARACTERS (BYTES).
	DEFAULT = 32760.
	REQUIRED FOR SORTING FIXED LENGTH RECORDS.
	INREC(3-5) MUST BE ZERO, AND ARE RESERVED FOR FUTURE USE.
OUTREC	FIVE WORD ARRAY CONTAINING OUTPUT RECORD INFORMATION:
	OUTREC(1) = OUTPUT RECORD TYPE (SEE INREC)
	OUTREC(2) = MAXIMUM OUTPUT LINE SIZE IN CHARACTERS(BYTES).
	OUTREC(3-5) MUST BE ZERO, AND ARE RESERVED FOR FUTURE USE.
SPCLS	FIVE WORD ARRAY CONTAINING:
	SPCLS(1) = SPACE OPTION
	0 = INCLUDE BLANK LINES IN SORT
	1 = DELETE BLANK LINES.
	DEFAULT = 0 = INCLUDE BLANK LINES.
	SPCLS(2-5) MUST BE ZERO, AND ARE RESERVED FOR FUTURE USE.
OPROC	OUTPUT DATA DESTINATION
	0 = OUTPUT FILE
	1 = OUTPUT PROCEDURE.

MRG2$S

EACH CALL TO MRG2$S RETURNS THE NEXT MERGED RECORD. IF AN OUTPUT FILE IS SPECIFIED, MRG2$S SHOULD NOT BE CALLED.

CALLING SEQUENCE:

```
CALL MRG2$S(RTBUFF,LENGTH)
```

PARAMETERS:

RTBUFF	BUFFER CONTAINING NEXT MERGED RECORD (RETURNED). SHOULD BE LARGE ENOUGH TO HOLD THE LONGEST RECORD MERGED.
LENGTH	LENGTH OF RECORD IN CHARACTERS OR BYTES (RETURNED). WHEN ALL RECORDS HAVE BEEN RETURNED, CALLS TO RLSE$S RETURN A RECORD LENGTH OF 0.

MRG3$S

MRG3$S IS CALLED TO CLOSE ALL UNITS OPENED BY THE MERGE ROUTINES. IF AN OUTPUT FILE IS SPECIFIED, MRG3$S SHOULD NOT BE CALLED.

CALLING SEQUENCE:

```
CALL MRG3$S
```

NOTES:

1. ALL PARAMETERS ARE INTEGER*2.
2. IF OUTPUT IS TO A FILE, MRG2$S AND MRG3$S SHOULD NOT BE CALLED.
3. IT IS RECOMMENDED THAT THE MAXIMUM LINE SIZE (INREC(2)) BE SPECIFIED WHENEVER POSSIBLE. THIS IS NOT REQUIRED, BUT ITS SPECIFICATION WILL RESULT IN A MORE EFFICIENT USE OF THE LARGE COMMON BLOCKS.

```
**********************************************************************************************
**********************************************************************************************
**
**
**
**      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     SSS    PPPP     000     000    L
**    R   R   S   S   P   P   0   0   0   0   L
**    R   R   S       P   P   0   0   0   0   L
**    RRRR     SSS    PPPP    0   0   0   0   L
**    P P          S  P       0   0   0   0   L
**    R  R    S   S   P       0   0   0   0   L    L
**    R   R    SSS    P        000     000    LLLLL
**
**
**
**********************************************************************************************
**********************************************************************************************
```

REV. 17 CHANGES TO THE PRIMOS SPOOL SUBSYSTEM

DATE:

TO:

FROM:

SUBJECT: REV. 17 CHANGES TO THE PRIMOS SPOOL SUBSYSTEM

REFERENCE: NONE

ABSTRACT

WHILE THERE ARE MANY VALID SUGGESTIONS WAITING TO BE IMPLEMENTED IN THE SPOOL SUBSYSTEM, THIS DOCUMENT IS ADDRESSING ITSELF ONLY TO THE CURRENT NEED FOR MORE ADVANCED FORMS HANDLING AND MORE FLEXIBLE METHODS FOR ADDRESSING PRINTERS, TO MAKE THE SPOOL SUBSYSTEM MORE USABLE ON NETWORKS ON A SHORT TERM BASIS.

THE CHANGES TO THE USER INTERFACE ARE NOT EXTENSIVE. HOWEVER, THE INTERFACE BETWEEN THE COMPUTER OPERATOR AND THE RUNNING SPOOL PHANTOM HAS UNDERGONE MANY CHANGES TO SUPPORT THE FUNCTIONALITY DESCRIBED IN THIS DOCUMENT. THIS REVISION OF THE DOCUMENT INCLUDES CHANGES MADE BETWEEN SOFTWARE REVISIONS 17.0 AND 17.1.

REV. 17 CHANGES TO THE PRIMOS SPOOL SUBSYSTEM

I. USER VISIBLE CHANGES

A. FORMS HANDLING

THE FORM NAME IS STORED AS PART OF A QUEUE ENTRY ALONG WITH THE REST OF THE DATA ON THAT JOB (SIZE OF FILE, NAME OF USER, ETC.). PREVIOUSLY, WHEN THE PHANTOM SAW THAT REQUEST, IT DECIDED WHETHER OR NOT IT COULD PRINT IT BASED ON A DIRECT COMPARISON BETWEEN THE FORM NAME IN THE QUEUE ENTRY AND THE FORM TYPE IT HAD BEEN TOLD WAS MOUNTED.

THE USER MAY NOW SPECIFY ANY ONE OF A NUMBER OF SYNONYMS FOR A CERTAIN FORM TYPE. HE IS TOLD IF THE FORM TYPE IS UNRECOGNIZED. IF NOT DESIRED, THIS CHECKING CAN BE DISABLED BY THE PARTICULAR SYSTEM'S OPERATIONS STAFF. SYNONYMS ARE DESCIBED MORE FULLY IN SECTION II.F.

B. DESTINATION PRINTERS

PREVIOLSLY, THE ONLY WAY THE USER COULD EXERCISE CONTROL OVER WHERE HIS OR HER OUTPUT WENT WAS TO USE THE -HOME OPTION WHICH GENERALLY FORCED OUTPUT TO A PRINTER WHICH WAS LOCAL TO THE SYSTEM UPON WHICH THE REQUEST WAS MADE. WITHOUT USING THAT OPTION, A REQUEST COULD CONCEIVABLY BE PRINTED ON ANY PRINTER WITHIN A NETWORK, EVEN IF THE DISTANCE BETWEEN THE PERSON MAKING THE REQUEST AND THE LINE PRINTER SATISFYING THE REQUEST WERE MANY MILES APART.

ANOTHER PROBLEM INVOLVED THE COMPUTER OPERATOR, WHO DIDN'T KNOW WHERE TO DELIVER OUTPUT UNLESS INFORMATION EXISTED THAT ASSOCIATED THE LOGIN NAME ON THE LISTING WITH THAT PERSON'S LOCATION.

THE FOLLOWING SOLUTIONS HAVE BEEN IMPLEMENTED:

1. CREATION OF A NEW SPOOL OPTION -AT, WHICH IS FOLLOWED BY A DESTINATION NAME (16 CHARACTERS OR LESS) THAT IS USED TO DETERMINE THE PRINTABILITY OF THE FILE, AND ALSO IS PRINTED ON THE HEADER PAGE. THE NAME IS PART OF THE REQUEST ENTRY IN THE QUEUE CONTROL FILE. IF THE -AT OPTION IS NOT USED, THE DEFAULT FOR THAT SYSTEM WILL BE USED. IF THE OPERATIONS STAFF HAS NOT SET UP A DEFAULT, THE DESTINATION FIELD WILL BE BLANK AND THE FILE CAN BE PRINTED ANYWHERE.
2. LEAVE THE -HOME OPTION TO DO EXACTLY WHAT IT USED TO DO, BUT STRESS THAT GIVEN THE NEW AND MORE DYNAMIC WAY TO DO THINGS, IT WILL SOON BE OBSOLETE.

C. THE OUTPUT

MORE FLEXIBILITY WAS NEEDED FOR THE USER. THE FOLLOWING FUNCTIONALITY HAS BEEN ADDED:

1. THE -AS OPTION IS FOLLOWED BY A NAME (1-16 CHARACTERS) THAT REPLACES THE CONTENTS OF THE FILENAME FIELD IN THE QUEUE ENTRY. THEREFORE, SPOOL ALPHA -AS BETA WOULD CAUSE THE FILE ALPHA TO BE PRINTED ON THE LINE PRINTER WITH THE COVER NAME BETA (I.E. SPOOL -LIST AND THE HEADER PAGE WOULD SHOW BETA). THE OPTION IS PARTICULARLY USEFUL IF ALPHA IS A TREENAME WITH PASSWORDS.
2. THE -COPIES OPTION IS FOLLOWED BY THE NUMBER OF COPIES (1-99) OF THE FILE TO BE PRINTED. THE DEFAULT IS 1. ALL OF THE COPIES OF A FILE WILL COME OUT UNDER ONE SET OF HEADER AND TRAILER PAGES, UNLESS THE PHANTOM HANDLING THE REQUEST IS INTERRUPTED.
3. THE -NOHEAD OPTION SPECIFIES THAT THE FILE SHOULD BE PRINTED WITHOUT ANY HEADER (OR TRAILER) PAGES. THIS ALLOWS BETTER USE OF PRE-PRINTED FORMS. CARE MUST BE TAKEN THAT FILES PRINTED WITH THIS OPTION CAN BE INDENTIFIED IN A MULTI-USER ENVIRONMENT.

D. CLEANING UP THE COMMAND LINE

THE OLD SYNTAX OF THE SPOOL COMMAND LINE WAS VERY CONFUSING. IT HAD A MIXTURE OF USER AND OPERATOR COMMANDS, MULTIPLE COMMANDS WERE ALLOWED, AND MUCH OBSOLETE NOTATION WAS SUPPORTED. CLEANING UP THE COMMAND LINE INVOLVED THE FOLLOWING STEPS:

1. REMOVEL OF THE SUPPORT OF OBSOLETE NOTATION. THIS NOTATION USED SPECIAL CHARACTERS TO INDENTIFY KEYWORDS. FOR EXAMPLE, A COLON PRECEDING A KEYWORD INDICATED A COMMAND, AND A LEFT PARENTHESIS INDICATED AN OPTION.
2. ALLOWING ONLY ONE COMMAND PER COMMAND LINE. THIS PREVENTS THE AMBIGUITIES THAT COULD ARISE WITH THE OLD SYNTAX. OPERATORS MADE EXTENSIVE USE OF THE MULTIPLE COMMAND FEATURE. STEP 3 DEALS DEALT WITH THAT ISSUE. THE USER MAY STILL SPECIFY A -LIST COMMAND AT THE END OF A COMMAND LINE WHICH IS SPOOLING A FILE.
3. REMOVING THE SUPPORT OF OPERATOR COMMANDS. SECTION II DESCRIBES THE TRANSFERAL OF THIS FUNCTIONALITY TO A NEW PROGRAM.
4. EXTENSIONS TO THE CANCEL COMMAND. THE USER MAY NOW GIVE A DECIMAL NUMBER AS AN ARGUMENT. THIS NUMBER IS THE REQUEST THAT IS TO BE CANCELLED. E.G. 'SPOOL -CANCEL 4' INSTEAD OF 'SPOOL -CANCEL PRT004'. MULTIPLE ARGUMENTS CAN BE SPECIFIED. E.G. 'SPOOL -CANCEL 2 4 PRT006 7'. A USER NO LONGER NEEDS TO CLOSE AN OPEN SPOOL FILE BEFORE HE CANCELS IT.

F. NEW SPOOLQ FILES

THE NEW USER FUNCTIONALITY THAT HAS BEEN DISCUSSED REQUIRES ADDITIONAL FILES TO BE MAINTAINED IN THE SPOOLQ UFD. EACH OF THESE FILES IS

OPTIONAL (SPOOL CAN BE RUN WITHOUT THEM). THESE FILES ARE MAINTAINED USING ED.

1. L.DFLT

   THIS FILE CONSISTS OF ONE ASCII LINE OF FROM 1 TO 16 CHARACTERS. IF THE USER DOES NOT USE THE -AT OPTION ON THE SPOOL COMMAND LINE, THE STRING IN THIS FILE WILL BE USED AS A DEFAULT. IF THIS FILE IS NOT PRESENT, THE DEFAULT IS A BLANK FIELD.

2. L.FORM

   A FILE OF ASCII LINES WHERE EACH LINE IS A 1-6 CHARACTER NAME. IF THE USER USES THE -FORM OPTION ON THE SPOOL COMMAND LINE, THIS FILE IS SEARCHED FOR A MATCH ON THE USER'S FORM NAME. HE GETS AN ERROR MESSAGE IF NO MATCH OCCURS. IF THIS FILE DOES NOT EXIST, NO CHECKING IS DONE.

3. L.DEST

   A FILE OF ASCII LINES WHERE EACH LINE IS A 1-16 CHARACTER NAME. ITS FUNCTION IS THE SAME AS L.FORM, EXCEPT THAT IT IS SEARCHED FOR A MATCH WITH THE USER'S -AT NAME.

II. OPERATOR USE OF THE NEW SPOOL SUBSYSTEM

A. GENERAL MODIFICATIONS

ALL OPERATOR COMMANDS HAVE BEEN REMOVED FROM SPOOL, (ABORT, DROP, BACK, RESTART, LOGOUT, PAPER, HANG, GO, LENGTH, TIME, USER, QUIT, FINISH) AND A NEW PROGRAM INVOKED AS A COMMAND MAINLY FOR USE BY OPERATORS CALLED PROP (PRINTER OPERATOR), HAS BEEN WRITTEN.

THE INTERFACE IN THE PHANTOM FOR COMMUNICATION WITH THE OPERATOR HAVE BEEN REWRITTEN TO COMMUNICATE WITH PROP ON A MORE RELIABLE AND EFFICIENT BASIS.

THE OPERATOR HAS BEEN GIVEN MORE FLEXIBILITY IN CONTROLLING THE ACTIONS OF THE PHANTOM; HE CAN ASK THE PHANTOM TO LOG OUT AFTER FINISHING THE CURRENT JOB (VS. LOGGING OUT IMMEDIATELY); HE CAN ASK THE PHANTOM FOR ITS STATUS; HE CAN SET A LIMIT THAT SPECIFIES THE BIGGEST FILE A PHANTOM SHOULD PRINT; AND HE CAN SET DIFFERENT PARAMETERS THAT DEFINE A NETWORK ENVIRONMENT FOR THE PHANTOM.

B. DEFINING WHICH PHANTOM TO COMMUNICATE WITH

IF PROP IS GIVEN A COMMAND THAT OPERATES ON A SPECIFIC PHANTOM, PROP MUST BE ABLE TO KNOW WHICH PHANTOM TO COMMUNICATE WITH.

THE NEW METHOD IS TO ASSIGN EACH PHANTOM A UNIQUE LABEL (1-16 CHARACTERS, 1-4 ON OLD PARTITIONS). THE SPOOL SUBSYSTEM WILL NOW REVOLVE AROUND THESE PRINTER/PHANTOM LABELS RATHER THAN THEIR PROCESS NUMBERS. FOR INSTANCE, IF A PHANTOM THAT WAS STARTED UP WAS GIVEN THE LABEL 'ENBPRO', THEN THAT PHANTOM WOULD HOLD THE FILE O.ENBPRO OPEN WHILE IT WAS RUNNING, LOOK FOR OPERATOR REQUESTS IN R.ENBPRO, AND USE M.ENBPRO TO SEND MESSAGES. E.ENBPRO WOULD HOLD DATA BETWEEN RUNS OF THE PHANTOM, A.ENBPRO WOULD BE A TEMPORARY FILE, AND O_ENBPRO WOULD CONTAIN COMOUTPUT.

C. PROP OVERVIEW

AS HAS BEEN MENTIONED, EACH REV 17 PHANTOM RUNS WITH AN ASSOCIATED LABEL. THIS LABEL IS ALSO THE NAME OF THE 'ENVIRONMENT' WHICH CONTROLS THE ACTIONS OF THE PHANTOM.

A PROP COMMAND LINE IS OF THE FORM:

```
PROP <LABEL> -<COMMAND>
```

THE ONLY EXCEPTION TO THIS RULE IS THE

```
PROP -STATUS
```

COMMAND LINE. THE STATUS COMMAND IS DESCRIBED IN SECTION II.E. PROP COMMANDS CAN BE SEPERATED INTO TWO CLASSES. SECTION II.D DECRIBES THOSE COMMANDS WHICH FALL INTO THE 'OPERATIONS' CLASS. SECTION II.E DEALS WITH THE 'ENVIRONMENT' CLASS COMMANDS.

D. OPERATIONS COMMANDS

THESE COMMANDS ARE MEANT TO REPLACE THE OPTIONS THAT ARE BEING REMOVED FROM SPOOL. THE FOLLOWING OLD SPOOL OPTIONS ARE PRESENT IN PROP AS COMMANDS AND DO EXACTLY THE SAME THINGS: ABORT, DROP, BACK, AND RESTART.

THE HANG COMMANDS IS STILL PRESENT BUT WITH ADDED ARGUMENTS TO MAKE IT MORE POWERFUL. THE FINISH OPTION IS GONE AS ITS FUNCTIONALITY HAS BEEN TAKEN OVER BY THE NEW HANG COMMAND. THE FUNCTIONALITY OF THE LOGOUT COMMAND NOW FALLS UNDER THE STOP COMMAND. STOP TAKES THE SAME ARGUMENTS AS HANG AND IS THUS MORE POWERFUL THAN LOGOUT WAS.

THE HANG AND STOP COMMANDS TAKE THE FOLLOWING ARGUMENTS:

```
NOW      - IMMEDIATELY.
FINISH   - AFTER CURRENTLY PRINTING FILE IS FINISHED.
IDLE     - WHEN THE PHANTOM HAS NO MORE WORK TO DO.
```

THE GO OPTION USED TO TAKE THE SPOOLER OUT OF A HANG STATE. IF IT WAS PRINTING A FILE AND FINISH HAD BEEN SENT TO IT, THE OPERATOR WOULD HAVE TO WAIT FOR THE FILE TO FINISH TO RESUME IT WITH GO, UNTIL THEN, GO WOULD BE A NO-OP. THE NEW SPOOLER TREATS CONTINUE TO MEAN CLEAR ALL FLAGS THAT INDICATE IF IT IS IN HANG MODE. CONTINUE REPLACES GO AS A

COMMAND.

PHANTOMS USED TO BE STARTED UP BY TYPING A COMMAND LINE OF THE FORM:

PH SPOOLQ>PH_XXX

THE NEW WAY TO START UP A PHANTOM IS TO USE THE START COMMAND.

PROP <LABEL> -START

E. ENVIRONMENT COMMANDS

THE STATUS COMMAND WILL GIVE A LIST OF THE CURRENTLY DEFINED ENVIRONMENTS. IT WILL ALSO INDICATE WHICH ONES ARE BEING USED BY A PHANTOM. IF A DETAILED DESCRIPTION OF A PARTICULAR ENVIRONMENT IS DESIRED, THE DISPLAY COMMAND IS USED. THESE TWO COMMANDS ARE THE ONLY PROP COMMANDS THAT AN ORDINARY USER IS ALLOWED TO USE. TO MAKE USE OF THE OTHER PROP COMMANDS, A USER'S LOGIN NAME MUST MATCH THE LOGIN NAME OF THE USER THAT CREATED THE ENVIRONMENT.

A CURRENTLY DEFINED ENVIRONMENT CAN BE DELETED BY USING THE DELETE COMMAND. THE REMAINING PROP COMMANDS (CREATE, MODIFY) PUT PROP INTO ENVIRONMENT DEFINITION MODE.

ENVIRONMENT DEFINITION MODE IS USED TO SPECIFY THE VALUES OF THE PARAMETERS THAT MAKE UP AN ENVIRONMENT.

THE CREATE COMMAND IS USED SET UP AN ENVIRONMENT THAT IS NOT CURRENTLY DEFINED. THE MODIFY COMMAND IS USED IF THE ENVIRONMENT IS DEFINED. IF THE MODIFY COMMAND IS USED ON A STARTED PHANTOM, IT MAY BE FOLLOWED BY ONE OF THE ARGUMENTS THAT HANG AND STOP TAKE. THE SPECIFIED ARGUMENT WILL DETERMINE WHEN THE ENVIRONMENT CHANGES WILL BE MADE.

WHEN ENVIRONMENT DEFINITION MODE IS ENTERED, PROP PROMPTS WITH '> '. SUB-COMMANDS MAY THEN BE GIVEN, ONE PER LINE. THE FORMAT OF A SUB-COMMAND IS: <PARAMETER NAME> <VALUE>

THE FOLLOWING IS A LIST OF PARAMETERS THAT ARE LEGAL SUB-COMMANDS.

PLOT ON<OFF
IF 'ON', SCAN THE QUEUE FOR FILES WHICH ARE DESIGNATED AS PLOT FILES. IF 'OFF', IGNORE PLOT FILES THAT ARE PRESENT IN THE QUEUE. THE DEFAULT IS 'OFF'.

PRINT ON<OFF
IF 'ON', SCAN THE QUEUE FOR PRINT FILES. IF 'OFF' IGNORE THEM. THE DEFAULT IS 'ON'.

DEVICE PR0<PR1<PR2<PR3<CENPR<CE2PR<PLOT<AMLC
OUTPUT TO THIS DEVICE. THE DEFAULT IS PR0. IF AMLC IS SELECTED, IT MUST BE FOLLOWED BY AN OCTAL LINE NUMBER. THE AMLC LINE WILL BE USED AS CONFIGURED ON THE SYSTEM CONSOLE.

UPCASE ON<OFF
IF 'ON', CONVERT ALL LOWER CASE CHARACTERS TO UPPER CASE BEFORE PRINTING. IF 'OFF', DO NO CONVERSION. THIS IS THE DEFAULT.

HEADER 0<1<2
SET THE NUMBER OF HEADER PAGES. A SETTING OF 2 WILL ALSO GIVE A TRAILER PAGE. THE DEFAULT IS 1.

PAPER <NAME>
<NAME> FORMS ARE MOUNTED. <NAME> IS FROM 1-6 CHARACTERS. THE DEFAULT IS ' '. ALL FORM SYNONYMS ARE DELETED.

LENGTH <N>
PRINT <N> LINES PER PAGE. THE DEFAULT IS 38 LINES.

LINES <N>
THERE ARE <N> PHYSICAL LINES PER PAGE. THE DEFAULT IS LENGTH+13.

WIDTH <N>
THERE ARE <N> PHYSICAL COLUMNS ON A PAGE. USED FOR FORMATTING HEADER AND/OR TRAILER PAGES. THE DEFAULT IS 108 COLUMNS.

LIMIT <N>
DO NOT PRINT ANY FILES BIGGER THAN <N> DISK RECORDS. THE DEFAULT IS 30000.

UPPER <N>
DO NOT LOOK FOR SPOOLQ'S ON ANY LOGICAL DISK WITH A NUMBER HIGHER THAN <N>. THE DEFAULT IS 63. <N> IS A DECIMAL NUMBER.

LOWER <N>
LOOK FOR SPOOLQ'S STARTING AT LOGICAL DISK <N>. THE DEFAULT IS 0. <N> IS A DECIMAL NUMBER.

LARGE <N>
GIVE PRIORITY TO FILES THAT HAVE A LENGTH IN RECORDS THAT IS LESS THAN <N>. THE DEFAULT IS 30.

COMOUT ON<OFF

IF 'ON', KEEP A COMOUTPUT FILE OF ALL PHANTOM ACTIONS. THE NAME OF THE FILE IS SPOOLQ>O_<LABEL>. IF 'OFF', TURN COMOUTPUT OFF. APPEND TO THE FILE IF IT ALREADY EXISTS. THE DEAULT IS 'OFF'.

MESSAGE <TEXT>
PRINT <TEXT> ON EVERY HEADER PAGE. <TEXT> IS ONE LINE OF UP TO 80 CHARACTERS. IF <TEXT> IS OMITTED, A NULL MESSAGE WILL RESULT.

THE TWO SUB-COMMANDS THAT DO NOT TAKE AN ARGUMENT ARE USED TO EXIT FROM PROP'S ENVIRONMENT DEFINITION MODE. QUIT WILL EXIT AND ANY PARAMETER CHANGES THAT HAVE BEEN MADE WILL BE IGNORED. THE FILE COMMAND WILL EXIT AND TAKE ACTION APPROPRIATE TO THE WAY IN WHICH ENVIRONMENT DEFINITION MODE WAS ENTERED. IF THE CREATE COMMAND WAS GIVEN, AN ENVIRONMENT WILL BE CREATED AND DEFAULTS USED WHEREVER THE USER DID NOT SPECIFY A VALUE. IF THE MODIFY COMMAND WAS GIVEN, ANY PARAMETERS THAT THE USER CHANGED WILL BE UPDATED IN THE ENVIRONMENT. CHANGES THAT PROP MAKES TO AN ENVIRONMENT ARE PHYSICALLY REPRESENTED BY CHANGING THE FILE E.<LABEL>. THIS FILE HOLDS ALL DATA DEFINING THE ENVIRONMENT FOR THE PHANTOM WITH NAME <LABEL>.

F. SYNONYMS

THE PHANTOM ATTEMPTS TO MATCH A FILE'S FORM TYPE WITH THE PAPER NAME IN ITS ENVIRONMENT. IT ATTEMPTS TO MATCH A FILE'S AT FIELD WITH ITS LABEL. BOTH THE PAPER AND LABEL NAMES CAN HAVE A NUMBER OF SYNONYMS THAT ESSENTIALLY MAP TO THE SAME NAME AT THE PHANTOM LEVEL. FOR EXAMPLE, WHILE FORM TYPE 'WHITE' MIGHT BE MOUNTED, IF IT HAS SYNONYMS 'PE-T', 'NARROW' AND '8_X_11', THEN ANY REQUEST WITH ANY OF THOSE NAMES AFTER THE -FORM OPTION IS ELIGIBLE FOR PRINTING. EXAMPLE:

SPOOL RPE-T-442 -FORM 8_X_11

WILL PRINT ON THE PRINTER WITH WHITE FORMS MOUNTED, AND THE ONLY NOTICEABLE DIFFERENCE BETWEEN DOING THE ABOVE AND SPECIFYING FORM WHITE IS THAT BY INSPECTING THE HEADER PAGE THE WORD 8_X_11 WILL BE FOUND SOMEWHERE ON IT.

THIS APPLIES TO THE PRINTER NAME AS WELL. A GOOD EXAMPLE IS RIGHT HERE IN NEWTON WHERE WE ONLY HAVE THREE RUNNING LINE PRINTERS ON THE FIRST FLOOR. IF USERS COULD GET IN THE HABIT OF SPOOLING FILES WITH THE OPTION -AT NEWTON-1, -AT NEWTON-2 OR -AT NEWTON-3 DEPENDING ON WHICH FLOOR THEY'RE ON, THEN, IF WE GOT MORE PRINTERS AND PUT THEM ON DIFFERENT FLOORS, LISTINGS WOULD BE EASIER TO DELIVER AND WE WOULD YIELD FASTER THROUGHPUT. AN EXAMPLE TABLE OF SYNONYMS FOR OUR ENB PRINTER WOULD BE:

'NEWTON', 'NEWTON-1', 'NEWTON-2', 'NEWTON-3', 'BLDG. 6', 'PRIME-6', '1ST FLOOR', ETC.

IF WE HAD A SECOND FLOOR PRINTER, IT MIGHT HAVE A SET OF SYNONYMS THAT

LOOK LIKE:

'NEWTON', 'NEWTON-2', 'BLDG. 6', 'PRIME-6', '2ND FLOOR', ETC.

WHILE THERE IS A LIMIT ON THE NUMBER OF SYNONYMS FOR FORM NAMES AND PRINTER NAMES, IT IS CONFIGURABLE. THE DEFAULT IS 8.

THE ENVIRONMENT DEFINITION MODE SUB-COMMANDS TO DEFINE SYNONYM NAMES ARE:

```
DEST      FOR DEFINING SYNONYMS OF THE PRINTER NAME.
FORM      FOR DEFINING SYNONYMS OF THE FORM NAME.
```

SIMPLY FOLLOW THE COMMAND WITH THE SYNONYM, AND IT WILL BE ENTERED INTO THE ARRAY. EXAMPLES:

```
DEST NEWTON-3RD
DEST 3RD.FLOOR
FORM L.CASE
```

THE SUB-COMMANDS UNDEST AND UNFORM HAVE THE OPPOSITE EFFECT, RESPECTFULLY. THEY TAKE A NAME AND DELETE THAT NAME AS A SYNONYM IF IT EXISTS.

G. COMPATIBILITY

THIS DESIGN IS BEING AFFECTED BY THE USUAL AMOUNT OF CONCERN THAT PRIME PROGRAMMERS DEVOTE TO COMPATIBILITY; IT IS FOR THIS REASON THAT MANY "NICE" FUNCTIONS THAT COULD BE PUT INTO PROP WILL NOT BE, AND THEY WON'T UNTIL IT IS DETERMINED EXACTLY WHAT IS DESIRED IN SPOOLERS IN THE FUTURE, SO THAT REVISION 17 SPOOL WILL BE COMPATIBLE WITH NEWER VERSIONS IN THE FUTURE.

AND, CONCERNING COMPATIBILITY, EXCEPTING THE DESCRIBED CHANGES, THE NEW SPOOL SHOULD BE COMPATIBLE WITH REVISION 16 SPOOL ON THE USER LEVEL. IF A REV. 16 SPOOL COMMAND IS GIVEN AND A NEW SPOOLER PHANTOM SEES THE REQUEST, IT WILL SEE ONLY A 6-CHARACTER LOGIN NAME, A BLANK DESTINATION (-AT) NAME, A REQUEST FOR 1 COPY, ETC. JUST AS IF A REVISION 16 PHANTOM HAD SEEN IT.

ON THE OTHER HAND, IF A NEW SPOOL COMMAND PRODUCES A REQUEST THAT IS SEEN BY A REVISION 16 PHANTOM, THEN THAT PHANTOM WILL IGNORE ALL OF THE NEW DATA, BUT STILL RECEIVE A SIX-CHARACTER LOGIN NAME, IT WILL NEVER PRINT MORE THAN ONE COPY ANYWAY, AND IF THE FORM NAME MATCHES ITS FORM NAME, IT WILL PRINT THE REQUEST NO MATTER HOW FAR AWAY THE PRINTER THAT IT IS RUNNING ON IS AWAY FROM THE USER THAT SUBMITTED THE REQUEST.

HOWEVER, THERE WILL BE NO COMPATIBILITY BETWEEN REVISION 16 SPOOL AND THE NEW SPOOL SUBSYSTEM AT THE OPERATOR COMMAND LEVEL - IF SPOOL -PAPER WHITE IS GIVEN TO A PHANTOM RUNNING THE NEW SPOOLER, THE REVISION 16 SPOOLER WILL NOT EVEN SEE IT, RETURNING NO SPOOLER AS AN ERROR MESSAGE.

THIS WORKS BOTH WAYS (I.E. PROP <LABEL> -ABORT TO A REV. 16 SPOOLER WILL RETURN THE NO SPOOLER ERROR MESSAGE TOO).

```
**********************************************************************************************
**********************************************************************************************
**
**
**
**      JJJ    III    M    M   M    M   Y    Y
**        J      I    MM  MM   MM  MM   Y    Y
**        J      I    M M M    M M M     Y Y
**        J      I    M M M    M M M      Y
**    J   J      I    M    M   M    M     Y
**    J   J      I    M    M   M    M     Y
**     JJ      III    M    M   M    M     Y
**
**
**
**    RRRR    U    U   SSS     AAA     GGG    EEEEE
**    R   R   U    U   S   S   A   A   G   G   E
**    R   R   U    U   S       A   A   G       E
**    RRRR    U    U    SSS    AAAAA   G       EEEE
**    R R     U    U       S   A   A   G  GG   E
**    R  R    U    U   S   S   A   A   G   G   E
**    R   R    UUU     SSS    A   A    GGGG   EEEEE
**
**
**
**********************************************************************************************
**********************************************************************************************
```

USAGE: A SYSTEM METERING TOOL

DATE:

TO:

FROM:

SUBJECT: USAGE: A SYSTEM METERING TOOL

REFERENCE: NONE

ABSTRACT

THE COMMAND "USAGE" (TEMPORARILY EXPERIMENTAL AND KNOWN AS "X.USAGE") ALLOWS USERS AND SYSTEM MAINTENANCE PERSONNEL TO MONITOR THE PERFORMANCE OF A RUNNING PRIMOS SYSTEM IN A NUMBER OF KEY AREAS. THIS TOOL FORMERLY DEPENDED FOR ITS OPERATION ON DETAILED KNOWLEDGE OF CERTAIN DATABASE LOCATIONS WITHIN PRIMOS; THIS DEPENENCY HAS BEEN REMOVED.

--------- PRINT SYSTEM PERFORMANCE METERS ---------
< USAGE < < USAGE <
--------- EXTERNAL COMMAND 03-19-79 ---------

NAME: USAGE

PURPOSE:

THE COMMAND "USAGE" PERFORMS A DIFFERENTIAL SAMPLING ON CERTAIN PERFORMANCE METERS MAINTAINED BY A RUNNING PRIMOS SYSTEM. ON EACH PASS THROUGH THE PRINTING LOOP, THE DIFFERENTIAL VALUES OF THE VARIOUS SYSTEM AND PER-USER METERS ARE DISPLAYED IN A REASONABLE VISUAL FORMAT. OPTIONS EXIST TO CHOOSE BETWEEN A SHORT AND LONG FORM OF OUTPUT, AND TO CONTROL THE INTERVAL BETWEEN SAMPLES.

USAGE:

USAGE [CONTROL_ARG1 ... CONTROL_ARGN]

THE CONTROL_ARGI ARE OPTIONAL CONTROL ARGUMENTS, SELECTED IN ANY ORDER FROM THE LIST BELOW.

-FREQ N
SELECTS AUTOMATIC SAMPLING EVERY N SECONDS (N MUST BE AN INTEGER IN THE RANGE [1, 32767]). IT IS NOT RECOMMENDED THAT N BE LESS THAN 30, SINCE THE RESULTING DIFFERENTIAL VALUES MAY BE UNDULY INFLUENCED BY STATISTICAL ARTIFACT. IF -FREQ IS NOT GIVEN, MANUAL SAMPLING IS SELECTED (SEE BELOW).

-TIMES N
IF AUTOMATIC SAMPLING IS IN EFFECT, SPECIFIES THE TOTAL NUMBER OF SAMPLES TO BE TAKEN. THE COMMAND WILL TERMINATE AFTER N SETS OF DATA HAVE BEEN PRINTED. N MUST BE AN INTEGER IN THE RANGE [1, 32767]. IF -TIMES IS NOT SPECIFIED, SAMPLING CONTINUES INDEFINITELY.

-BRIEF
SPECIFIES THAT A SHORT FORM OF OUTPUT IS TO BE PRODUCED. THIS FORM PRESENTS AN OVERVIEW OF WHAT PROCESSES AND USERS ARE CONSUMING SYSTEM RESOURCES; ADDITIONAL DETAILS ARE TO BE HAD FROM THE LONG FORM. THE DEFAULT IS TO PRODUCE THE LONG FORM.

MANUAL_SAMPLING

IF MANUAL SAMPLING IS IN EFFECT, USAGE WILL PAUSE TO COMMAND LEVEL (VIA A CALL TO EXIT) WHEN IT IS FIRST INVOKED, AFTER TAKING THE FIRST SAMPLE. SUBSEQUENT "START" COMMANDS (NO ARGUMENTS PERMITTED) WILL CAUSE USAGE TO TAKE ANOTHER SAMPLE, AND PRINT THE MOST RECENT DIFFERENTIAL VALUES. IT IS NOT RECOMMENDED THAT THE MANUAL SAMPLING TIME BE LESS THAN 30 REAL SECONDS.

THE_DATA_DISPLAYED

THE FOLLOWING DATA ARE DISPLAYED IN THE LONG FORM OF OUTPUT:

DTIME
IS THE NUMBER OF REAL SECONDS FROM THE PREVIOUS SAMPLE TO THE ONE JUST TAKEN; THE DIFFERENTIAL VALUES PRINTED COVER THIS AMOUNT OF REAL TIME.

CPTOT
IS THE NUMBER OF CPU SECONDS CHARGED TO ALL USER PROCESSES SINCE COLDSTART.

IOTOT
IS THE NUMBER OF I/O (DISK) SECONDS CHARGED TO ALL USER PROCESSES SINCE COLDSTART.

DCPTOT
IS THE NUMBER OF CPU SECONDS CHARGED TO ALL USER PROCESSES IN THE CURRENT SAMPLING INTERVAL.

%CP
IS THE PERCENT OF DTIME DURING WHICH CPU TIME WAS CHARGED TO USER PROCESSES. THIS CAN BE LOOSELY INTERPRETED AS THE PERCENT USEFUL UTILIZATION OF THE CPU.

DPFCN
IS THE NUMBER OF PAGE FAULTS ENCOUNTERED BY ALL PROCESSES DURING THE LAST SAMPLING INTERVAL.

PF/SEC
IS THE PAGE FAULT FREQUENCY IN FAULTS PER SECOND, OVER THE LAST SAMPLING INTERVAL.

DIOTOT
IS THE NUMBER OF I/O (DISK) SECONDS CHARGED TO ALL USER PROCESSES IN THE LAST SAMPLING INTERVAL.

%IO
IS THE PERCENT OF DTIME DURING WHICH I/O (DISK) WAS CHARGED TO USER PROCESSES. THIS CAN BE LOOSELY INTERPRETED AS THE PERCENT OF TIME DISK I/O WAS IN PROGRESS.

DIOCN
IS THE NUMBER OF I/O (DISK) OPERATIONS EXECUTED IN THE LAST SAMPLING INTERVAL.

IO/SEC
IS THE I/O (DISK) REQUEST RATE IN OPERATIONS PER SECOND, OVER THE LAST SAMPLING INTERVAL.

%OVLAP
IS AN ESTIMATE OF THE AMOUNT OF I/O (DISK) TRAFFIC WHICH HAS BEEN OVERLAPPED WITH OTHER COMPUTATION DURING THE LAST SAMPLING INTERVAL. IT IS COMPUTED AS:

$$\%OVLAP = \frac{MAX\ (0,\ (DIOTOT - IDLE)) * 100}{DIOTOT}$$

DLOCNT
IS THE TOTAL NUMBER OF CALLS MADE IN THE LAST SAMPLING INTERVAL TO THE FILE SYSTEM ASSOCIATIVE BUFFER MANAGER, LOCATE.

LC/SEC
IS THE LOCATE USE RATE IN CALLS PER SECOND, OVER THE LAST SAMPLING INTERVAL.

DLOFCT
IS THE NUMBER OF TIMES DURING THE LAST SAMPLING INTERVAL THAT A CALL TO LOCATE FOUND THE RECORD ALREADY IN THE ASSOCIATE BUFFERS.

DLOSCT
IS THE NUMBER OF TIMES DURING THE LAST SAMPLING INTERVAL THAT A CALL TO LOCATE WAS MADE ON THE SAME RECORD THE PROCESS HAD JUST PREVIOUSLY LOCATE'D.

DLOUCT
IS THE NUMBER OF TIMES DURING THE LAST SAMPLING INTERVAL THAT A CALL TO LOCATE WAS MADE FOR A RECORD THAT WAS ALREADY IN THE ASSOCIATE BUFFERS AND IN USE BY ANOTHER PROCESS.

DLOCCT
IS THE NUMBER OF TIMES DURING THE LAST SAMPLING INTERVAL THAT A CALL TO LOCATE WAS MADE AND A DISK READ HAD TO BE PERFORMED (I.E. THE NUMBER OF LOCATE MISSES).

LM/SEC
IS THE RATE OF LOCATE DISK READS (MISSES) IN READS PER SECOND, OVER THE LAST SAMPLING INTERVAL.

%MISS
IS THE PERCENT OF ALL LOCATE CALLS DURING THE LAST SAMPLING INTERVAL THAT CAUSED A LOCATE MISS (DISK READ).

%XCP OR %ERR
IS THE PERCENT OF CPU UTILIZATION NOT OTHERWISE ACCOUNTED FOR, AND PRESUMED TAKEN BY INTERRUPTS, SCHEDULER OVERHEAD, PROCESS EXCHANGE, AND SIMILAR OPERATIONS. THIS VALUE CAN BE NEGATIVE IF ONE OR MORE PROCESSES HAS BEEN OVERCHARGED WITH RESPECT TO CPU TIME.

%CLK
IS THE PERCENT OF CPU TIME USED BY THE CLOCK SERVICE PROCESS DURING THE LAST SAMPLING INTERVAL.

%AML
IS THE PERCENT OF CPU TIME USED BY THE ASYNCHRONOUS MULTIPLE LINE CONTROLLER DIM DURING THE LAST SAMPLING INTERVAL.

%MPC

IS THE PERCENT OF CPU TIME USED BY THE MICROPROGRAMMED PERIPHERAL CONTROLLER DIM (LINE PRINTERS, ETC.) DURING THE LAST SAMPLING INTERVAL.

%IPC

IS THE PERCENT OF CPU TIME USED BY THE INTERPROCESSOR COMMUNICATION CONTROLLER DIM DURING THE LAST SAMPLING INTERVAL.

%FAR

IS THE PERCENT OF CPU TIME USED BY THE RING NETWORK CONTROLLER DIM DURING THE LAST SAMPLING INTERVAL.

%SLC

IS THE PERCENT OF CPU TIME USED BY THE SYNCHRONOUS MULTIPLE LINE CONTROLLER DIM DURING THE LAST SAMPLING INTERVAL.

%BAK OR %IDLE

IS THE PERCENT OF CPU TIME USED BY THE BACKSTOP PROCESS DURING THE LAST SAMPLING INTERVAL. THIS VALUE CAN BE (ROUGHLY) INTERPRETED AS THE PERCENT OF TOTAL CPU TIME WASTED BECAUSE EITHER THERE WAS NO WORK TO BE DONE, OR ALL OTHER PROCESSES WERE WAITING FOR I/O OR OTHER EXTERNAL EVENTS.

%DSK

IS THE PERCENT OF CPU TIME USED BY THE TWO DISK DIM PROCESSES DURING THE LAST SAMPLING INTERVAL.

METERING INFORMATION SUPPLIED FOR EACH USER

MEM

IS THE TOTAL NUMBER OF PHYSICAL PAGES RESIDENT IN MEMORY (AT THE TIME THE PAGE CONTROL DATABASES WERE EXAMINED) THAT BELONG TO THE USER'S SEGMENTS (SEGMENT NUMBERS 0 THROUGH 3777 ARE CHARGED TO USER 1). THIS VALUE CAN BE TAKEN AS A ROUGH ESTIMATE OF THE DEMAND THE USER IS PLACING ON VIRTUAL MEMORY MANAGEMENT. IF THE SYSTEM IS PAGING AT A REASONABLY HIGH RATE, THIS VALUE CAN ALSO APPROXIMATE THE SIZE OF THE USER'S AVERAGE WORKING SET OVER REASONABLY SHORT INTERVALS. THIS VALUE IS NOT ALWAYS A GOOD WORKING SET ESTIMATOR, HOWEVER.

CPTIME

IS THE CPU TIME, IN SECONDS, USED BY THIS USER SINCE LOGIN.

DCP

IS THE CPU TIME, IN SECONDS, USED BY THIS USER DURING THE LAST SAMPLING INTERVAL.

%CP

IS THE PERCENT OF TOTAL CPU TIME USED BY THIS USER DURING THE LAST SAMPLING INTERVAL.

ICTIME

IS THE I/O (DISK) TIME, IN SECONDS, USED BY THIS USER SINCE LOGIN.

DIO
IS THE I/O (DISK) TIME, IN SECONDS, USED BY THIS USER DURING THE LAST SAMPLING INTERVAL.

%IO
IS THE PERCENT OF REAL TIME (OVER THE LAST SAMPLING INTERVAL) DURING WHICH I/O (DISK) WAS IN PROGRESS FOR THIS USER.

NOTES

IF A USER LOGS IN OR LOGS OUT DURING A SAMPLING INTERVAL, THE METER VALUES READ MAY BE SUCH THAT INCORRECT (EVEN NEGATIVE) VALUES RESULT. SOME CAUTION MUST THEREFORE BE USED IN INTERPRETING THE PER-USER METERING DATA.

EXAMPLE:

```
USAGE -FREQ 1800 -TIMES 10
```

WILL METER THE SYSTEM FOR 5 HOURS, WITH A SAMPLING TIME OF 30 MINUTES.

```
**********************************************************************************
**********************************************************************************
**
**
**
**      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  DDDD
**    P   R   E      D   D
**    R   R   E      D   D
**    RRRR    EEEE   D   D
**    R R     E      D   D
**    R  R    E      D   D
**    R   R   EEEEE  DDDD
**
**
**
**
**********************************************************************************
**********************************************************************************
```

MODIFICATIONS TO THE EDITOR

DATE: APRIL 12, 1979

TO:

FROM:

SUBJECT: MODIFICATIONS TO THE EDITOR

REFERENCE: NONE

ABSTRACT

THIS DOCUMENT DESCRIBES TWO ENHANCEMENTS AND ONE BUG FIX THAT HAVE BEEN MADE TO THE REV. 17 VERSION OF THE EDITOR.

THE FOLLOWING CHANGES HAVE BEEN MADE TO THE EDITOR:

1) THE FUNCTIONALITY OF THE GENERAL ESCAPE SYMBOL HAS BEEN EXTENDED SO THAT IT CAN NOW BE USED TO ESCAPE ANY OF THE SPECIAL SYMBOLS THE EDITOR RECOGNIZES. ENTERING THE ESCAPE SYMBOL IN FRONT OF ANY CHARACTER IN EITHER EDIT OR INPUT MODE CAUSES THAT SYMBOL TO BE TREATED AS A LITERAL CHARACTER. TO USE THE ESCAPE CHARACTER ITSELF AS A LITERAL CHARACTER TWO ESCAPES MUST BE TYPED.

2) THE EDITOR IS NOW ABLE TO HANDLE FILES LARGER THAN 32K LINES. THE PROBLEM BEFORE WAS THAT THE LINE POINTERS WERE INTEGER*2 AND THEREFORE WHENEVER THE FILE BECAME LARGER THAN THAT THE EDITOR WAS NOT SURE WHERE IT WAS. THESE HAVE BEEN CHANGED TO INTEGER*4. AS A SIDE EFFECT THE COUNTER IN MODE COUNT CAN ALSO GET LARGER THAN 5 DIGITS. DEFAULT REMAINS THE SAME, HOWEVER, 5 DIGITS WITH PRINTED LEADING ZEROES. A SECOND SIDE EFFECT WILL BE NOTICED IN MODE NUMBER, LINE NUMBERS SMALLER THAN 5 DIGITS WILL BE PRINTED AS 5 DIGITS WITH LEADING ZEROES JUST AS THEY WERE BEFORE. LINE NUMBERS LARGER THAN 5 DIGITS WILL SUDDENLY JUMP TO 10 DIGITS IN LENGTH. THIS CORRECTION DEALS WITH TARS: 80440, 13712, 15392, 12700, AND 24931.

3) THE LINESZ COMMAND DID NOT HAVE A LOWER LIMIT WHICH MEANT THE USER COULD CHANGE THE EDITOR'S MAXIMUM LINE SIZE TO 1, SINCE THIS CAUSES THE COMMAND LINE TO ALSO BE LIMITED TO ONE, ANY COMMANDS LONGER THAN THAT COULD NO LONGER BE USED. THIS RESULTED IN THE USER NOT BEING ABLE TO DO MUCH OF ANYTHING SINCE IT WAS IMPOSSIBLE TO RESET THE LINESIZE WITH ONLY ONE LETTER. THE MINIMUM LINESZ IS NOW 10.

```
**********************************************************************************************
**********************************************************************************************
**
**
**
**      JJJ    III    M    M   M    M   Y    Y
**       J      I     MM  MM   MM  MM   Y    Y
**       J      I     M M M    M M M     Y Y
**       J      I     M M M    M M M      Y
**    J  J      I     M    M   M    M     Y
**    J  J      I     M    M   M    M     Y
**     JJ       III   M    M   M    M     Y
**
**
**
**    RRRR     SSS     III    ZZZZZ   EEEEE
**    R   R   S   S     I         Z    E
**    R   R   S         I        Z     E
**    RRRR     SSS      I       Z      EEEE
**    R R          S    I      Z       E
**    R  R    S   S     I     Z        E
**    R   R    SSS     III    ZZZZZ   EEEEE
**
**
**
**********************************************************************************************
**********************************************************************************************
```

CHANGES TO SIZE FOR REVISION 17.1

DATE:

TO:

FROM:

SUBJECT: CHANGES TO SIZE FOR REVISION 17.1

REFERENCE:

ABSTRACT

THIS IS A DOCUMENT OF THE MINOR CHANGES TO SIZE FOR REVISION 17.1. THE CHANGES INCLUDE:

FILES THAT ARE LONGER THAN 32,767 NORMALIZED (440-WORD) RECORDS ARE NOW ACCURATELY REPRESENTED INSTEAD OF HAVING NEGATIVE SIZES.

SIZE NO LONGER PLURALIZES A 1-RECORD FILE WHILE REPORTING ITS SIZE.

SIZE HAS BEEN SPEEDED UP FOR SAM AND DAM FILES.

SIZE NOW DYNAMICALLY OBTAINS FILE UNITS FROM THE OPERATING SYSTEM INSTEAD OF ALWAYS USING UNIT 1.

WHEN ERRORS OCCUR WHILE READING A FILE, THE ERROR MESSAGES NOW CONTAIN THE FILE NAME.

THE ALGORITHM FOR CALCULATING NORMALIZED RECORDS FROM THE NUMBER OF WORDS IN THE FILE HAS BEEN FIXED.

WHEN SIZE WAS INVOKED ON FILES LARGER THAN 32,767 RECORDS (=14,417,480 WORDS), IT WOULD OUTPUT AN INCORRECT NUMBER TO REPRESENT ITS SIZE, SINCE IT USED ONLY INTEGER*2 FIELDS FOR THE NUMBER OF RECORDS. THE NUMBER WOULD EITHER BE NEGATIVE, OR IF THE FILE WAS VERY LARGE IT MIGHT BE POSITIVE.

NOW, SIZE USES INTEGER*4 NUMBERS TO CALCULATE THE NUMBER OF RECORDS. IF THE FILE TURNS OUT TO BE LESS THAN 32,768 RECORDS LONG, THEN SIZE WILL STILL USE TODEC TO OUTPUT THE FILE SIZE, THEREFORE RESEMBLING THE EARLIER REVISIONS OF SIZE. HOWEVER, IF THE FILE IS LARGER THAN 32,767 RECORDS IN LENGTH, IT WILL USE A 10-DIGIT FIELD FOR OUTPUT, USING THE APPLIB ROUTINE CNVB$A TO PERFORM THE CONVERSION. EXAMPLES:

```
"     34 RECORDS IN FILE"     VS.
"     437987 RECORDS IN FILE"
```

WHEN A ONE-RECORD FILE IS SIZED, THE OUTPUT WILL NOW BE SINGULAR, I.E.:

```
"      1 RECORD IN FILE"  INSTEAD OF
"      1 RECORDS IN FILE"
```

SIZE NOW WORKS FASTER ON SAM AND DAM FILES. PREVIOUSLY, SIZE CALLED PRWF$$ REPEATEDLY TO STEP FORWARD 4000 WORDS IN THE FILE UNTIL AN EOF WAS ENCOUNTERED, AT WHICH POINT SIZE WOULD CALL PRWF$$ AGAIN TO READ THE POSITION IN THE FILE (REPRESENTING THE NUMBER OF WORDS IN THE FILE).

NOW, SIZE STEPS FORWARD EITHER 16,384 (2^14) WORDS FOR SAM FILES, OR 2,147,483,647 (2^31-1) WORDS FOR DAM FILES PER CALL TO PRWF$$. THIS RESULTS IN FEWER CALLS INTO PRIMOS FOR SAM FILES, AND EXACTLY 2 CALLS FOR DAM FILES. SIZE BECOMES MUCH FASTER FOR LARGE DAM FILES, SOMEWHAT FASTER FOR ALL FILES MORE THAN 4000 WORDS LONG, BUT STILL QUITABLE.

BECAUSE QUIT DOES NOT WORK WHILE CERTAIN FILE SYSTEM OPERATIONS (SUCH AS PRWF$$ POSITIONING) ARE TAKING PLACE, SIZE WILL NOT ATTEMPT TO POSITION FORWARD VERY FAR IN SAM FILES, SO THAT VERY LONG FILES WOULD NOT PREVENT THE USER FROM QUITING. LARGE DAM FILES, ON THE OTHER HAND, CAN BE SIZED VERY QUICKLY.

SIZE NOW USES THE NEW K$GETU FUNCTIONALITY AVAILABLE AT REV. 16 IN PRIMOS TO OPEN THE FILE, MEANING THAT IT WILL NO LONGER CLOSE AND USE UNIT 1. THE EXCEPTION IS WHEN SIZE RUNS UNDER PRIMOS II, OR ANY OPERATING SYSTEM THAT DOES NOT SUPPORT K$GETU, IN WHICH CASE IT WILL STILL USE UNIT 1 (CLOSING IT IF NECESSARY).

IF SIZE ENCOUNTERS AN ERROR ON ANY OPERATION OTHER THAN THE OPENING OF THE FILE (WHICH HAS SPECIAL ERROR REPORTING OF ITS OWN), IT WILL NOW REPORT THE FILENAME ALONG WITH THE ERROR MESSAGE. EXAMPLE: "OPERATION ILLEGAL ON DIRECTORY. UFD>SUBUFD (SIZE)".

SIZE NOW CALCULATES THE NUMBER OF RECORDS USED BY A FILE THE SAME WAY THAT FLTIL DOES. THE OLD METHOD SIZE USED WAS TO SIMPLY DIVIDE THE NUMBER OF WORDS IN THE FILE BY 440 AND THEN ADD 1. THIS WOULD RESULT IN A FILE THAT WAS EXACTLY 440 WORDS IN LENGTH TO BE REPORTED AS BEING 2 RECORDS LONG, WHEN IN FACT IT WOULD ONLY TAKE UP 1 RECORD. THE NEW METHOD IS TO SET THE NUMBER OF WORDS TO 1 IF IT WAS ZERO (ALLOWING FOR THE SPECIAL CASE WHICH STATES THAT EMPTY FILES ALWAYS HAVE 1 RECORD ALLOCATED), ADD 43$_8$ TO THAT NUMBER, AND THEN DIVIDE IT BY 440. EXAMPLE: A FILE WHICH IS 880 WORDS LONG IS (880+439)/440 RECORDS LONG, WHICH IS 1319/440 OR 2 RECORDS LONG. ALL REMAINDERS ARE THROWN OUT SINCE THIS IS INTEGER DIVISION.

```
**********************************************************************************************
**********************************************************************************************
**
**
**
**      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      Y
**    J  J      I     M   M   M   M      Y
**    J  J      I     M   M   M   M      Y
**     JJ      III    M   M   M   M      Y
**
**
**
**    RRRR     SSS    EEEEE    GGG      1
**    R   R   S   S   E       G   G    11
**    R   R   S       E       G        1
**    RRRR     SSS    EEEE    G        1
**    R R          S  E       G  GG     1
**    R  R    S   S   E       G   G     1
**    R   R    SSS    EEEEE    GGGG    111
**
**
**
**********************************************************************************************
**********************************************************************************************
```

SEG - 17.1 - CHANGES

A NEW WILL NOW COPY ALL ENTRIES IN THE SEGDIR CORRECTLY (BUG FIX).

WILL RECOGNIZE PL1G DEBUG GROUP FOR PICTURES.

CORRECTLY ASSIGNS SYMBOL FOR BLANK COMMON (BUG FIX).

S 1000 DOES NOT RE-INITIALIZE SHARED LIBRARIES (BUG FIX).

CMDSEG IS NO LONGER ATTACH POINT DEPENDANT

```
**********************************************************************************************
**********************************************************************************************
**
**
**
**      JJJ    III    M    M   M    M   Y    Y
**        J      I    MM  MM   MM  MM   Y    Y
**        J      I    M M M    M M M     Y Y
**        J      I    M M M    M M M      Y
**    J   J      I    M   M    M   M      Y
**    J   J      I    M   M    M   M      Y
**     JJ      III    M   M    M   M      Y
**
**
**
**    RRRR     SSS    EEEEE    GGG
**    R   R   S   S   E       G   G
**    R   R   S       E       G
**    RRRR     SSS    EEEE    G
**    R R          S  E       G  GG
**    R  R    S   S   E       G   G
**    R   R    SSS    EEEEE    GGGG
**
**
**
**********************************************************************************************
**********************************************************************************************
```

CHANGES TO LOAD & SEG FOR REV. 17

DATE: APRIL 5, 1979

TO:

FROM:

SUBJECT: CHANGES TO LOAD & SEG FOR REV. 17

REFERENCE: NONE

ABSTRACT

SEG FOR REV. 17

THE FOLLOWING ARE THE CHANGES MADE TO SEG FOR REV 17:

1) SEG WILL NOW HANDLE DBG OBJECT TEXT.

2) SEG NOW SUPPORTS AN EXPANDED SYMBOL COMMAND

3) SEG ALSO HAS TWO NEW COMMANDS:

SZ FOR RESTRICTING THE PLACEMENT OF LINKS IN SECTOR ZERO BASE AREAS

AU FOR AUTOMATICALLY PLACING BASE AREAS BETWEEN PROCEDURES

4) SEG WILL NOW SET BIT 1 OF THE FIRST WORD OF THE FIRST SB TO INDICATE THAT SEG IS KICKING OFF A MAIN PROGRAM FOR COBOL (AND ANYONE ELSE WHO WANTS) TO CHECK.

5) SEG WILL ACCEPT TREENAMES BEGINNING WITH "*>" (THE USER WILL BE ATTACHED TO HIS HOME UFD BEFORE THE SEARCH IS DONE [TSRC$S IS USED].

6) SEG WILL NO LONGER HANDLE FLOATING POINT EXCEPTIONS. AT REV. 17, THESE ARE HANDLED BY PRIMOS.

LOAD FOR REV. 17

THERE IS NO NEW FUNCTIONALITY FOR LOAD AT REV. 17

1) DBG TEXT

SEG WILL NOW HANDLE THE DEBUGGER OBJECT GROUP TYPES PUT OUT BY THE REV17 COMPILERS. THESE ARE GROUP TYPES: 53,54,55,56,57,58,59,63. IN ADDITION, SEG WILL CREATE AND OUTPUT TO DBG GROUP TYPES 61 & 62. TO HANDLE THESE GROUPS, SEG WILL DEDICATE ENTRY #1 IN THE SEGMENT DIRECTORY FOR THEM AND WHEN THEY ARE ENCOUNTERED DURING A LOAD, SEG WILL COPY THEM DIRECTLY INTO ENTRY #1 FOR DBG TO READ. SEG WILL AUTOMATICALLY PUT IN THE AIG (ADDRESS INFORMATION GROUP) AND THE BLOCK TERMINATOR GROUPS TO THIS FILE ONLY WHEN OTHER DBG INFORMATION HAS BEEN ENCOUNTERED FOR THE CURRENT PROCEDURE. WHEN RELOADING INTO AN OLD RUNFILE, THAT DOES NOT HAVE THE SPACE FOR THE DBG GROUPS, SEG WILL TREAT THE LOAD JUST LIKE AN OLD LOAD, AND WILL NOT UPDATE THE RUNFILE TO MAKE THE SPACE FOR THEM. TO USE DBG WITH AN OLD RUNFILE, THE RUNFILE MUST BE REBUILT WITH SEG FROM SCRATCH.

2) SYMBOL COMMAND

THE SYMBOL COMMAND IN SEG HAS BEEN EXPANDED TO ALLOW ASSIGNING THE VALUE OF ALREADY DEFINED SYMBOLS TO NEW SYMBOLS AND/OR CONSTANTS WITH OPTIONALLY ASSIGNING CONSTANT OFFSETS IN THE SAME MANNER AS LOAD NOW OFFERS.

SY <NEW_SYMBOL_NAME> <VALUE> [<PARS>]

WHERE <NEW_SYMBOL_NAME> ::= A SYMBOL NAME TO WHICH THE VALUE WILL BE ASSIGNED

<VALUE> ::= <OLD_SYMBOL_NAME> < <SEGNO WORDNO> < *

<OLD_SYMBOL_NAME> ::= AN ALREADY DEFINED & SATISFIED SYMBOL

<SEGNO WORDNO> ::= AN ADDRESS IN MEMORY

<PARS> ::= AN OCTAL NUMBER < -

3) SZ & AU

3.1) THE SZ COMMAND WILL RESTRICT THE USE OF SECTOR ZERO BASE AREAS IN PROCEDURE SEGMENTS. THE COMMAND TAKES THE FORM:

SZ <SEGNO> [YES < NO]

WHERE <SEGNO> ::= THE SEGMENT OF THE SECTOR ZERO BASE AREA TO STOP/START USING

NO IS THE DEFAULT, AND WILL CAUSE THE GIVEN BASE AREA NOT TO BE USED AND AN ERROR GENERATED IF A LINK IS NEEDED.

YES WILL ALLOW THE USE OF THAT BASE AREA AGAIN.

3.2) THE AU COMMAND WILL PLACE BASE AREAS BETWEEN YOUR PROCEDURES AUTOMATICALLY FOR YOU. THIS COMMAND LOOKS LIKE:

AU <NUMBER>

WHERE <NUMBER> IS THE SIZE OF THE BASE AREA. THE DEFAULT IS 0, AND TURNS THIS FEATURE OFF. PROCEDURES GREATER THAN '341 WORDS IN LENGTH WILL HAVE A BASE AREA ALLOCATED BEFORE AND AFTER THE PROCEDURE CODE.

4) MAIN PROGRAMS

RUNIT AND THE SHARE4 LIBRARY WILL NOW SET A BIT IN THE FIRST STACK FRAME THAT SEG CREATES, INDICATING THAT THEY HAVE STARTED OF THIS PROGRAM, AND THAT THE PROGRAM THAT THEY ARE PCLING IS THE "MAIN PROGRAM". WORD 0 OF THIS STACK FRAME WILL NOW BE '100000 INSTEAD OF 0. THE ADDRESS OF THIS WORD IS STORED IN THE CURRENT STACK FRAME AT SB 'EXIT PROGRAM' INSTRUCTION.

5) TREENAMES

FORMERLY, WHEN A TREENAME WAS SPECIFIED AS AN OBJECT FILE DURING A LOAD, THE USER WAS NOT REATTACHED TO HIS HOME UFD UNLESS HE SPECIFICALLY REATTACHED TO IT, OR ISSUED A LIBRARY COMMAND (LI,PL,IL). FROM NOW ON, THE USER WILL BE ATTACHED TO HIS HOME UFD BEFORE, AND AGAIN AFTER THE TREENAME IS USED BY SEG.

6) FLEX

FORMERLY, WHEN A FLOATING POINT EXCEPTION OCCURRED, TRANSFER OF CONTROL WENT TO SEG TO TRY TO RECOVER AND CONTINUE. NOW FLEX IS A PART OF THE CONDITION MECHANISM, LEAVING ITS HANDLING TO EITHER THE USER OR PRIMOS.

7) LOAD

NO NEW FUNCTIONALITY EXISTS FOR LOAD AT REV. 17. THE ONLY REASON THERE IS A LOAD REV. 17 IS BECAUSE IT SHARES CODE WITH SEG THAT HAS CHANGED.

```
**********************************************************************************************
**********************************************************************************************
**
**
**
**      JJJ    III    M    M   M    M   Y    Y
**       J      I     MM  MM   MM  MM   Y    Y
**       J      I     M M M    M M M     Y Y
**       J      I     M M M    M M M      Y
**   J   J      I     M    M   M    M     Y
**   J   J      I     M    M   M    M     Y
**    JJ       III    M    M   M    M     Y
**
**
**
**   RRRR    RRRR    U    U   N    N   000    FFFFF   FFFFF    222
**   R   R   R   R   U    U   NN   N  0   0   F       F       2   2
**   R   R   R   R   U    U   N N  N  0   0   F       F           2
**   RRRR    RRRR    U    U   N N  N  0   0   FFFF    FFFF        2
**   R R     R R     U    U   N N  N  0   0   F       F         2
**   R  R    R  R    U    U   N  NN  0   0   F       F        2
**   R   R   R   R    UUU    N    N   000    F       F       22222
**
**
**
**********************************************************************************************
**********************************************************************************************
```

SUBJECT- RUNOFF BUGS FIXED IN REV. 17

SUBJECT- RUNOFF BUGS FIXED IN REV. 17

1. TAP 23221 AND 23222 - PROBLEMS WITH DECIMALIZATION COMMANDS HAVE BEEN FIXED

2. THE .SM COMMAND NOW TAKES EFFECT ON THE NEXT PAGE RATHER THAN WAITING FOR AN EXTRA PAGE

3. RUNOFF NO LONGER SAVES A PLACE FOR PHANTOM HYPHENS IN THE TABLE OF CONTENTS SO ALL PAGES SHOULD BE LINED UP CORRECTLY.

4. RUNOFF STACKS FILE NAMES FOR ERROR MESSAGES CORRECTLY.

```
**********************************************************************************************
**********************************************************************************************
**
**
**
**      JJJ    III    M    M   M    M   Y    Y
**       J      I     MM  MM   MM  MM   Y    Y
**       J      I     M M M    M M M     Y Y
**       J      I     M M M    M M M      Y
**    J  J      I     M    M   M    M     Y
**    J  J      I     M    M   M    M     Y
**     JJ      III    M    M   M    M     Y
**
**
**
**    RRRR    RRRR    U    U   N    N   000    FFFFF   FFFFF
**    R   R   R   R   U    U   NN   N  0   0   F       F
**    R   R   R   R   U    U   N N  N  0   0   F       F
**    RRRR    RRRR    U    U   N N  N  0   0   FFFF    FFFF
**    R R     R R     U    U   N N  N  0   0   F       F
**    R  R    R  R    U    U   N  N N  0   0   F       F
**    R   R   R   R    UUU    N   NN   000    F       F
**
**
**
**********************************************************************************************
**********************************************************************************************
```

MODIFICATIONS TO RUNOFF

DATE: APRIL 12, 1979

TO:

FROM:

SUBJECT: MODIFICATIONS TO RUNOFF

REFERENCE:

ABSTRACT

THIS DOCUMENT DESCRIBES THE BUG FIXES THAT HAVE BEEN MADE TO RUNOFF FOR RELEASE AT REV. 16.4 AND 17.0.

CHANGES IN RUNOFF FOR REV 16.4 AND 17.0:

1) IF A DECIMAL HEADER GOT PUSHED TO THE TOP OF A PAGE BECAUSE OF NOT BEING ALLOWED TO LEAVE WIDOWS ON THE BOTTOM OF THE PREVIOUS PAGE AND A FLOAT FILE HAD BEEN ENTERED SOMETIME PREVIOUSLY, THERE WAS THE POSSIBILITY OF GETTING PART OR ALL OF THE HEADER FOLLOWED BY THE FLOAT FILE BEFORE GETTING THE TEXT THAT BELONGED WITH THE HEADER. THIS HAS BEEN CORRECTED BY CHECKING IF A FLOAT SHOULD START AFTER THE NEED, BEFORE PUTTING THE HEADER IN THE OUTPUT FILE. IF FSTART IS TRUE WE BACK UP THE INPUT FILE ONE LINE, BY THE SUBROUTINE BACKUP, DO THE FLOAT FILE AND COME BACK AND RE-READ THE DECIMAL HEADER COMMAND THAT CAUSED THE PROBLEM AND RE-DO IT.
TAR 23222

2) .DL DID NOT WORK IF YOU HAPPENED TO DO A DLEVEL TO THE FIRST TIME AT THAT LEVEL. .DL LEAVES THE USER AT THE TEXT MARGIN FOR THE SPECIFIED LEVEL. THE .DN COMMAND CHECKS IF THIS IS THE FIRST TIME AT THAT LEVEL AND IF IT IS, DOES NOT DO AN UNDENT TO THE HEADING MARGIN. SINCE FUTURE DECIMALIZATION COMMANDS ONLY DO RELATIVE AND NOT ABSOLUTE INDENTS THE MARGINS ARE OFF THEREAFTER BY THE AMOUNT OF THAT LEVEL'S TEXT INDENT. ADDING A FLAG TDL TO SAY WHEN IT WAS A .DL THAT GOT YOU TO THE LEVEL THE UNDENT WILL HAPPEN ANYWAY IF YOU ARE AT THE TOP NUMBER FOR THE LEVEL AND GOT THERE BY A .DL.
TAR 23221

3) WHEN A .SM COMMAND HAPPENED TO COINCIDE WITH A PAGE EJECT OCCURRING BECAUSE OF A PREVIOUS COMMAND, THE NEW SIDE MARGINS DID NOT ACTUALLY TAKE EFFECT UNTIL THE NEXT PAGE. THIS HAS BEEN CORRECTED. OTHER COMMANDS THAT CAUSE A POSSIBLE BREAK AND EJECT TO SET UP A NEW PAGE ENVIRONMENT WILL ALSO NOW TAKE EFFECT ON THE APPROPRIATE PAGE RATHER THAN POSSIBLY WAITING AN EXTRA PAGE.

4) THE NAMES OF FILES WERE NOT STACKED CORRECTLY FOR PRINTING WITH ERROR MESSAGES, THIS HAS BEEN CORRECTED.

5) USING A PHANTOM HYPHEN IN A DECIMAL HEADING WHEN GENERATING A TABLE OF CONTENTS NO LONGER SAVES A SPACE FOR THE HYPHEN WHEN FILLING THE TABLE OF CONTENTS LINE WITH PERIODS BEFORE THE NUMBER. THIS MEANS THE PAGE NUMBERS WILL BE CORRECTLY LINED UP.

```
**********************************************************************************
**********************************************************************************
**
**
**
**      JJJ    III    M    M   M    M   Y    Y
**       J      I     MM  MM   MM  MM   Y    Y
**       J      I     M M M    M M M     Y Y
**       J      I     M M M    M M M      Y
**   J   J      I     M   M    M   M      Y
**   J   J      I     M   M    M   M      Y
**    JJ      III    M   M    M   M      Y
**
**
**
**   PPPP   L        OOO    GGG   PPPP   RRRR   TTTTT
**   P   P  L       O   O  G   G  P   P  R   R    T
**   P   P  L       O   O  G       P   P  R   R    T
**   PPPP   L       O   O  G       PPPP   RRRR     T
**   P      L       O   O  G  GG    P      R R      T
**   P      L   L   O   O  G   G   P      R  R     T
**   P      LLLLL    OOO    GGGG   P      R   R    T
**
**
**
**********************************************************************************
**********************************************************************************
```

LOGPRT REV 17

DATE: MARCH 29, 1979

TO:

FROM:

SUBJECT: LOGPRT REV 17

REFERENCE: NONE

ABSTRACT

THE LOGPRT COMMAND FOR PRIMOS REV 17 SUPPORTS TWO NEW EVENT TYPES, QUIET AND REMARK, AND AN EXTENSION TO THE COLD START ENTRY THAT CONTAINS A PRIMOS REVISION IDENTIFICATION. A NEW COMMAND LINE OPTION, -CENSUS, CAN BE USED TO OBTAIN A COUNT OF THE VARIOUS TYPES OF ENTRIES IN LOGREC. THE -FROM OPTION WILL ACCEPT 'TODAY' AS A DATE AND ACCEPTS AN OPTIONAL TIME SPECIFICATION.

A NEW LOGPRT OPTION, -REMARK, ALLOWS AN OPERATOR TO ENTER A TEXTUAL MESSAGE DIRECTLY INTO THE LOGREC FILE.

USE OF THE -INPUT OPTION IS NOW REQUIRED TO OVERRIDE THE DEFAULT LOGREC TREENAME SPECIFICATION: LOGPRT WILL NO LONGER PROMPT FOR THE INPUT TREENAME. (USERS WHO RUN LOGPRT FROM A COMMAND FILE TAKE NOTE!)

THIS UPDATE DESCRIBES THESE AND OTHER MODIFICATIONS MADE TO LOGPRT FOR REV 17. AS ALWAYS, THE NEW VERSION OF LOGPRT SHOULD BE INSTALLED AT THE SAME TIME AS THE NEW VERSION OF PRIMOS IS INSTALLED. (THE NEW LOGPRT, WILL, HOWEVER, FUNCTION CORRECTLY ON LOGREC FILES GENERATED BY PREVIOUS VERSIONS OF PRIMOS.)

1 NEW LOGREC ENTRIES

1.1 EXTENDED COLD START ENTRY

A COLD START ENTRY (TYPE=0) HAS BEEN EXTENDED WITH AN (OPTIONAL) ASCII VERSION IDENTIFICATION STRING FOLLOWING THE 8-WORD CPU ID FIELD. IF THIS FIELD IS PRESENT, LOGPRT INCLUDES IT IN THE FORMATTING OF THE COLD START ENTRY.

1.2 MACHINE CHECK QUIET MODE ENTRY

AFTER THE OCCURENCE OF 1024 ECCC ERRORS, PRIMOS SETS THE MACHINE CHECK MODE TO QUIET, WHICH CAUSES SUBSEQUENT ECCC ERRORS TO GO UNREPORTED. THIS EVENT IS NOW RECORDED IN THE LOGREC FILE WITH AN ENTRY OF TYPE=19. THE EVENT NAME (FOR THE -TYPES OPTION) IS QUIET.

1.3 REMARK ENTRY

THE REMARK EVENT (TYPE=20, NAME=REMARK) IS PROVIDED TO ALLOW ENTERING AN ARBITRARY ASCII STRING DIRECTLY INTO THE LOGREC FILE. LOGPRT SURROUNDS THE STRING WITH QUOTATION MARKS AND PLACES IT VERBATIM IN THE OUTPUT FILE. (SEE ALSO -REMARK OPTION BELOW.)

2 EVENT TYPE CENSUS DISPLAY

LOGPRT NOW MAINTAINS COUNTERS OF ALL EVENT TYPES PROCESSED AND DISPLAYS THESE COUNTERS WHEN THE END OF THE LOGREC FILE IS REACHED. ONLY SELECTED TYPES ARE COUNTED, AND ONLY NON-ZERO COUNTERS ARE DISPLAYED. (THE NUMBER OF DATE/TIME STAMPS IS DISPLAYED, ALTHOUGH DATE/TIME STAMP ENTRIES ARE NOT INCLUDED IN THE END-OF-FILE TOTALS MESSAGE.) FOR THE OVERFLOW ENTRY TYPE, THE TOTAL NUMBER OF OVERFLOWS IS ALSO DISPLAYED. A TYPICAL CENSUS DISPLAY APPEARS AS:

```
TYPE    NUMBER
COLD    20
WARM    5
TIMDAT  1019
CHECKS  2813
DISKER  589
OVERFL  513 (TOTAL=37455)
SHUTDN  11
DSKNAM  130
```

THE -CENSUS OPTION (DESCRIBED BELOW) CAN BE USED TO LIMIT LOGPRT'S PROCESSING TO JUST THE CENSUS DISPLAY.

3 NEW COMMAND LINE FUNCTIONS

3.1 CHANGE TO -INPUT HANDLING

USE OF THE -I OPTION IS NOW THE ONLY WAY OF OVERRIDING THE DEFAULT INPUT TREENAME (<0>CMDNC0>LOGREC). LOGPRT WILL NO LONGER PROMPT FOR THE INPUT TREENAME IF -I IS OMITTED FROM THE COMMAND LINE.

3.2 THE -CENSUS OPTION

THE -CENSUS OPTION (-C IS MINIMAL ABBREVIATION) INSTRUCTS LOGPRT TO MERELY COUNT THE ENTRIES IN THE INPUT FILE AND DISPLAY THE TOTALS ON THE TERMINAL. THE ENTRIES THAT ARE COUNTED ARE CONTROLLED BY THE USUAL SELECTION CRITERIA (-FROM, -TYPES), AND ONLY NON-ZERO TOTALS ARE DISPLAYED. (NOTE THAT THERE IS NO WAY TO REQUEST CENSUS-ONLY PROCESSING TO A DISK FILE; COMMAND OUTPUT CAN BE USED IF THIS IS REQUIRED.)

3.3 THE -REMARK OPTION

THE -REMARK OPTION ENTERS AN EVENT OF TYPE REMARK (SEE ABOVE) DIRECTLY INTO THE LOGREC FILE. THIS CAN BE USED, FOR EXAMPLE, BY AN OPERATOR WHO WISHES TO RECORD AN OBSERVATION ON SOME EVENT THAT MIGHT AFFECT THE SUBSEQUENT OPERATION OF THE SYSTEM. USE OF THE -REMARK OPTION IS AS FOLLOWS:

```
LOGPRT [-I <TREENAME>] -R <TEXT OF MESSAGE>
```

THE -I OPTION IS REQUIRED ONLY IF THE LOGREC FILE IS NOT <0>CMDNC0>LOGREC. ALL OTHER LOGPRT OPTIONS ARE IGNORED IF -REMARK IS SPECIFIED. THE -REMARK OPTION MUST BE THE LAST OPTION SPECIFIED ON THE COMMAND LINE. ALL TEXT AFTER THE -REMARK OPTION IS TAKEN AS THE TEXT TO BE ENTERED INTO LOGREC. THE MESSAGE CAN BE UP TO 160 CHARACTERS IN LENGTH AND NEED NOT BE SURROUNDED BY APOSTROPHES. WRITE-ACCESS IS REQUIRED ON THE LOGREC FILE.

3.4 NEW -FROM HANDLING

THE -FROM OPTION WILL NOW ACCEPT 'TODAY' (WHICH CAN BE ABBREVIATED TO 'T') AS AN ALTERNATIVE TO THE MMDDYY DATE SPECIFICATION. FOLLOWING THE DATE SPECIFICATION, -FROM WILL ACCEPT AN OPTIONAL TIME SPECIFICATION OF THE FORM HHMM (HOURS, MINUTES) BETWEEN 0000 AND 2359. OMITTING THE TIME SPECIFICATION IS EQUIVALENT TO SPECIFYING '0000'.

AN ADDITIONAL MODIFICATION HAS BEEN MADE IN THE WAY -FROM OPERATES. FORMERLY, ENTRY FORMATTING WAS TRIGGERED BY THE FIRST ENTRY WITH A DATE/TIME STAMP AFTER THAT SPECIFIED WITH THE -FROM OPTION. THIS

MEANT THAT AN OUT-OF-SEQUENCE ENTRY (E.G., THE WRONG DATE ENTERED BY THE OPERATOR) COULD TURN ON ENTRY FORMATTING PREMATURELY. LOGPRT NOW CHECKS EACH ENTRY INDIVIDUALLY TO SEE IF ITS DATE/TIME STAMP INDICATES THAT IT SHOULD BE FORMATTED.

3.5 NEW ABBREVIATION FOR THE -CONTINUE OPTION

THE MINIMAL ABBREVIATION FOR -CONTINUE IS NOW -CO, SINCE IT IS ENVISIONED THAT THE -CENSUS OPTION WILL BE USED MUCH MORE FREQUENTLY.

3.6 THE -DUMP OPTION

AS AN ADDITIONAL AID TO THOSE WHO DEFINE THEIR OWN EVENT TYPES, THE -DUMP OPTION IS PROVIDED. SPECIFYING -DU ON THE COMMAND LINE WILL CAUSE LOGPRT, IN ADDITION TO ITS NORMAL FORMATTING, TO DUMP EACH ENTRY PROCESSED IN OCTAL.

3.7 RESPECIFICATION OF OUTPUT TREENAME

TO RESPECIFY THE OUTPUT TREENAME (AFTER REPLYING NO TO THE "OK TO DELETE..." MESSAGE) IT USED TO BE NECESSARY TO REENTER ALL THE COMMAND LINE OPTIONS. IT IS NOW SUFFICIENT TO ENTER JUST THE NEW TREENAME. 'TTY' WILL, AS ON THE COMMAND LINE, CAUSE OUTPUT TO THE TERMINAL.

4 INTERNAL MODIFICATIONS

[THE FOLLOWING IS OF INTEREST ONLY TO THOSE USERS WHO DEFINE THEIR OWN EVENT TYPES AND/OR PROVIDE SPECIAL FORMATTING OF ENTRIES.]

BESIDES THE CHANGES NECESSARY TO SUPPORT THE ABOVE MODIFICATIONS, SOME MINOR MODIFICATIONS HAVE BEEN MADE TO THE ENTRY FORMATTING ROUTINES. THE STORE ROUTINE WILL NO LONGER TRUNCATE MESSAGES TOO LONG FOR ONE LINE. A LONG LINE WILL BE SPLIT AT A LINE BOUNDARY, SUBSEQUENT LINES BEING INDENTED. THE FNDEC ROUTINE HAS AN ADDED ARGUMENT -- SPECIFY 0 FOR INTEGER*2, 1 FOR INTEGER*4.

A NEW ROUTINE, DMPLIN, PERFORMS ALL LINE OUTPUT. IT EXPECTS A SINGLE INTEGER*2 ARGUMENT AS FOLLOWS:

```
-1 => JUST RESET LINE POINTERS (FORMERLY CALL F10U(0))
 0 => DUMP LINE, DON'T INDENT NEXT LINE (FORMERLY CALL F10U(:212))
 1 => DUMP LINE, INDENT NEXT LINE
```

ANOTHER NEW ROUTINE, USCORE(NCHAR), CAN BE USED TO UNDERSCORE AND DUMP THE PREVIOUS LINE.

```
**********************************************************************************************
**********************************************************************************************
**
**
**
**      JJJ    III    M    M   M    M   Y    Y
**       J      I     MM  MM   MM  MM   Y    Y
**       J      I     M M M    M M M     Y Y
**       J      I     M   M    M   M      Y
**   J   J      I     M    M   M    M     Y
**   J   J      I     M    M   M    M     Y
**    JJ       III    M    M   M    M     Y
**
**
**
**   RRRR    FFFFF   U    U   TTTTT    III    L          1
**   R   R   F       U    U     T       I     L         11
**   R   R   F       U    U     T       I     L          1
**   RRRR    FFFF    U    U     T       I     L          1
**   R R     F       U    U     T       I     L          1
**   R  R    F       U    U     T       I     L    L     1
**   R   R   F        UUU      T      III    LLLLL    111
**
**
**
**********************************************************************************************
**********************************************************************************************
```

CHANGES TO FUTIL FOR REVISION 17.1

DATE:

TO:

FROM:

SUBJECT: CHANGES TO FUTIL FOR REVISION 17.1

REFERENCE:

ABSTRACT

THIS IS A DOCUMENT OF THE MINOR CHANGES TO FUTIL FOR REVISION 17.1. THE CHANGES INCLUDE:

FIXING THE "TREDELETE A>B" BUG, WHERE FUTIL WOULD TREDELETE A BEFORE RETURNING A SYNTAX ERROR.

THE UFDCPY COMMAND CAN NO LONGER BE ABBREVIATED TO "U", "UF", OR "UFD".

FUTIL NOW READS IN THE COMMAND LINE USING COMANL INSTEAD OF C1IN (VIA LIBRARY ROUTINE RDCOM), FIXING A BUG THAT CAUSED FUTIL TO BEGIN EXECUTION OF A COMMAND BEFORE THE RETURN KEY WAS HIT.

COMMAND LINES CAN NOW BE UP TO 160 CHARACTERS IN LENGTH UNDER PRIMOS REV. 17. UNDER PREVIOUS OPERATING SYSTEMS (INCLUDING DOS), THE LIMIT IS STILL 80.

THE METHOD OF SIZING SAM AND DAM FILES HAS BEEN SPEEDED UP.

CERTAIN ERROR MESSAGES HAVE BECOME MORE EXPLANATIVE.

A BUG CAUSING "?NO MORE UNITS" TO BE OUTPUT AFTER MANY "INSUFFICIENT ACCESS RIGHTS" ERRORS OCCURRED HAS BEEN FIXED.

THE "TREDELETE A>B" BUG (TAR #10439) HAS BEEN FIXED. THE PROBLEM WAS THAT FUTIL WOULD PARSE THE "TREDEL" COMMAND, THEN PARSE THE "A" TOKEN, DO THE "TREDELETE A", AND THEN TRY TO PARSE FURTHER TO DETERMINE IF THERE WERE MORE ITEMS TO BE TREDELETED. AT THIS POINT, FUTIL WOULD DETERMINE THAT THERE WAS A SYNTAX ERROR, AND REPORT IT.

FUTIL NOW PARSES THE COMMAND AND THEN, IF THE COMMAND IS NOT A "FROM", "TO", "ATTACH", OR "*" (COMMENT) COMMAND, IT SCANS THE ENTIRE COMMAND LINE, LOOKING FOR A ">" SYMBOL, DENOTING A TREENAME. IF ONE IS FOUND, FUTIL IMMEDIATELY RETURNS WITH A SYNTAX ERROR.

THE "UFDCPY" COMMAND USED TO BE ABBREVIATED TO "U" (TAR #80855). SINCE THE COMMAND BECAME MORE POWERFUL AT REV. 14 DUE TO THE INTRODUCTION OF THE MERGE CAPABILITY (WHICH CAN CAUSE FILES TO BE LOST IF MISHANDLED), IT WAS SUGGESTED IN THE TAR THAT "U" NO LONGER BE ACCEPTED AS AN ABBREVIATION. MOST NEW DOCUMENTATION (NOT MANXXXX DOCUMENTS) SPECIFIED THE MINIMUM ABBREVATION TO BE "UFDC", ALTHOUGH AT LEAST ONE SPECIFIED IT AS "UFD".

FUTIL NO LONGER ACCEPTS "U", "UF", OR "UFD" AS AN ACCEPTABLE ABBREVIATION FOR "UFDCPY" OR FOR ANY OTHER COMMAND. IF ONE OF THESE ABBREVIATIONS ARE GIVEN, FUTIL WILL RESPOND WITH AN ERROR.

DURING INPUT, FUTIL USED TO CALL A LIBRARY SUBROUTINE NAMED RDCOM. THIS SUBROUTINE CALLED C1IN TO OBTAIN INPUT FROM THE TERMINAL, AND DID ITS OWN ERASE/KILL PROCESSING. IT WOULD ALSO RETURN IMMEDIATELY AS SOON AS 80 CHARACTERS WERE READ OR THE NEW-LINE CHARACTER WAS SEEN, WITHOUT GUARANTEEING THAT A NEW-LINE CHARACTER WAS PUT IN THE COMMAND LINE ARRAY.

THIS MEANT THAT A) THE USER, WHILE TYPING IN A LONG COMMAND LINE, COULD SUDDENLY FIND THE COMMAND HAD EXECUTED BEFORE THE RETURN KEY HAD BEEN HIT, B) ANY COMMAND FILES THAT RAN FUTIL COULD NOT USE ERASE AND KILL CHARACTERS SUCCESSFULLY, WHICH IS INCONSISTENT WITH PRIMOS AND SUBSYSTEMS THAT USE COMANL AND RDTK$$, AND C) WHILE REPORTING SYNTAX ERRORS ON LONG COMMAND LINES, FUTIL COULD "GO WEST" OUTPUTTING GARBAGE.

TO FIX THIS, THE CALL TO RDCOM HAS BEEN REPLACED WITH A CALL TO COMANL FOLLOWED BY A CALL TO RDTK$$ TO READ RAW_TEXT INTO AN 80-WORD BUFFER (160 CHARACTERS) AND A DO-LOOP TO UNPACK THE BUFFER TO RESEMBLE THE RESULTS OF CALLING RDCOM. THIS MEANS THAT PRIMOS NOW DOES ERASE/KILL PROCESSING (AND IT WILL NOT DO IT IF INPUT IS COMING FROM A COMMAND FILE). IT ALSO GUARANTEES THAT THE USER HAS HIT RETURN BEFORE THE COMMAND IS EXECUTED (SINCE COMANL WILL NOT RETURN UNTIL THEN). ALSO, IT GUARANTEES THAT A NEW-LINE CHARACTER IS IN THE LAST WORD OF THE UNPACKED ARRAY, MEANING ALL SYNTAX ERRORS WILL BE REPORTED ACCURATELY.

THIS ALSO ALLOWS FUTIL TO MAKE USE OF THE 160-CHARACTER COMMAND LINE BUFFER PRESENT IN PRIMOS REV. 17, WHILE STILL RUNNING UNDER PRIMOS II AND EARLIER REVISIONS OF PRIMOS IV & V. IN OTHER WORDS, FUTIL NOW ACCEPTS COMMAND LINES UP TO 160 CHARACTERS IN LENGTH UNDER PRIMOS REV. 17. UNDER PRIMOS II, IT STILL ACCEPTS ONLY 80.

FUTIL HAS BEEN MADE SLIGHTY FASTER WHEN SIZING SAM AND DAM FILES, EITHER IN UFDS OR IN SEGDIRS. FUTIL USED TO CALL PRWF$$ REPEATEDLY TO POSITION FORWARD 4096 WORDS UNTIL AN EOF WAS FOUND, AND THEN CALL PRWF$$ TO READ THE CURRENT POSITION (WHICH REPRESENTED THE FILE SIZE IN WORDS).

NOW, FUTIL CALLS PRWF$$ REPEATEDLY, POSITIONING FORWARD 16,384 (2^14) WORDS IF THE FILE IS A SAM FILE, OR 2,147,483,647 (2^31-1) WORDS IF THE FILE IS A DAM FILE BEFORE READING THE POSITION. THIS RESULTS IN FEWER CALLS TO PRIMOS FOR SAM FILES (WHILE STILL BEING QUITABLE), AND EXACTLY 2 CALLS PER FILE TO PRIMOS FOR DAM FILES (WHICH BY THEIR NATURE CAN BE SIZED SO RAPIDLY THAT QUITS RESPOND SPEEDILY). THESE SPEED-UPS WILL OCCUR IN ALL FILES OVER 4096 WORDS LONG.

ERROR MESSAGES IN FUTIL HAVE BECOME MORE EXPLANATIVE; IN PARTICULAR, THE ERROR MESSAGE "?" FOLLOWED BY A REPROMPT (">") INDICATING ONE OF THREE CONDITIONS: A) THAT THE COMMAND WAS UNRECOGNIZABLE, B) THAT THE COMMAND WAS AN ABBREVIATION OF ANOTHER COMMAND WHICH IS TOO DANGEROUS TO BE ABBREVIATED, AND C) THAT A PROTECT- OR SRWLOC-CLASS COMMAND WAS ATTEMPTED WHILE THE FROM-DIR WAS INSIDE A SEGMENT DIRECTORY.

THE ERROR MESSAGES HAVE NOW BECOME A) "?UNKNOWN COMMAND - XXXXXX", B) "?CAN'T ABBREVIATE XXXXXX COMMAND.", AND C) "?OPERATION ILLEGAL INSIDE SEGDIRS.".

ALSO, A BUG IN FUTIL EXISTED WHICH CAUSED FUTIL TO PRODUCE THE ERROR MESSAGE "?NO MORE UNITS" AFTER ENCOUNTERING MANY (APPROX. 14) ERRORS DURING A LISTF COMMAND, EVEN THOUGH MANY UNITS WERE STILL AVAILABLE. THIS WAS DUE TO FUTIL NOT INTERNALLY RETURNING THE UNIT THAT IT HAD INTERNALLY ALLOCATED AFTER ENCOUNTERING AN ERROR WHILE TRYING TO OPEN THAT UNIT. (IT REMEMBERED TO RETURN IT ONLY IF THE ERROR WAS E$FIUS - "FILE IN USE").

FUTIL NOW REMEMBERS TO RETURN THE UNIT INTERNALLY.

```
**********************************************************************************
**********************************************************************************
**
**
**
**      JJJ    III    M    M   M    M   Y    Y
**       J      I     MM  MM   MM  MM   Y    Y
**       J      I     M M M    M M M     Y Y
**       J      I     M  M  M  M  M  M     Y
**    J  J      I     M     M  M     M     Y
**    J  J      I     M     M  M     M     Y
**     JJ       III   M     M  M     M     Y
**
**
**
**    RRRR    FFFFF   TTTTT   N    N
**    R   R   F         T     NN   N
**    R   R   F         T     N N  N
**    RRRR    FFFF      T     N N  N
**    R R     F         T     N N  N
**    R  R    F         T     N  NN
**    R   R   F         T     N    N
**
**
**
**********************************************************************************
**********************************************************************************
```

REV. 17 - FTN

DATE: MARCH 29, 1979

TO:

FROM:

SUBJECT: REV. 17 - FTN

REFERENCE:

ABSTRACT

THIS MEMO DESCRIBES THE ENHANCEMENTS AND CHANGES TO FTN FOR REV. 17. SIGNIFICANT NEW FEATURES INCLUDE COMPATIBILITY WITH THE HIGHER LEVEL LANGUAGE DEBUGGER, WARNING MESSAGES, AND INCREASED SHORTCALL FUNCTIONALITY. THE DO LOOP OPTIMIZATION OF REV. 16 FTNOPT IS ALSO INCLUDED, AND A FEW BUGS ARE FIXED.

1 HIGHER LEVEL LANGUAGE DEBUGGER CAPABILITY

THE COMPILER IN 64V MODE WILL BE ABLE TO PRODUCE BINARY FILES COMPATIBLE WITH HIGHER LEVEL LANGUAGE DEBUGGER USE. THIS REQUIRES THREE NEW COMMAND LINE OPTIONS:

-DEBUG PRODUCE SPECIAL DEBUGGABLE CODE
-PROD "PRODUCTION MODE"
-NODEBUG PRODUCE BINARY FILE WITHOUT DEBUGGER INFORMATION

THE DEBUG OPTION OR MODE ALLOWS THE USER FULL USE OF THE DEBUGGER FACILITIES. MODULES COMPILED THIS WAY WILL ACCEPT STATEMENT BREAKPOINTS FROM THE DEBUGGER, AND THE DEBUGGER RECOGNIZES THEIR STATEMENT NUMBERS AND SOURCE LINE NUMBERS. TO ACHIEVE THIS THE GENERATED CODE MUST BECOME SLOWER AND MORE SPACE-CONSUMING. PART OF THIS IS DUE TO INTERSTATEMENT OPTIMIZATION BEING TURNED OFF.

PRODUCTION MODE ALLOWS THE DEBUGGER USER TO SET BREAKPOINTS AT PROCEDURE ENTRIES AND EXITS, BUT DOES NOT PERMIT BREAKPOINTS AT INDIVIDUAL STATEMENTS. VARIABLES ARE ACCESSIBLE JUST AS IN DEBUG MODE. THIS MODE IS IDEAL FOR MOST PROGRAMS WHICH HAVE PASSED THE DEBUGGING STAGE SINCE IT YIELDS EXECUTABLE CODE JUST AS EFFICIENT AS THE REV. 16 COMPILER'S CODE. THE ONLY RUN-TIME DIFFERENCE IS THAT FOR EACH COMMON BLOCK A NEW CONSTANT RESIDES IN MEMORY AS AN INDIRECT POINTER TO THAT BLOCK. ALSO, THE EXTRA INFORMATION WHICH PRODUCTION AND DEBUG MODES PLACE IN THE BINARY FILE CAN TRIPLE THE SIZE OF THAT FILE AND WILL INCREASE THE SIZE OF SEG RUNFILES.

-NODEBUG IS THE DEFAULT OPTION.

A FEW DETAILS OF DEBUGGER FUNCTIONALITY ARE DETERMINED BY THE COMPILER. THE CONDITIONAL PART OF A LOGICAL IF STATEMENT IS SEEN AS A STATEMENT ITSELF. THUS EXECUTION MAY BREAK BETWEEN THE LOGICAL EXPRESSION AND THE CONDITIONAL PART OF THE STATEMENT. ALSO STATEMENT LABELS ARE KNOWN TO THE DEBUGGER WITH A $ PREFIXED. THUS TO SET A BREAKPOINT AT THE STATEMENT LABELLED 99 THE COMMAND IS BRK $99. STATEMENT 0010 IS REFERRED TO AS $10. (IN FTN, STATEMENT LABELS PASSED AS ARGUMENTS ARE ALREADY REFERRED TO IN THIS WAY. FOR THE SAKE OF CONSISTENCY THE CROSS REFERENCE AND EXPANDED LISTINGS OF THE COMPILER ARE CHANGED TO DISPLAY $ PREFIXING A LABEL RATHER THAN _. $ WILL APPEAR EVEN WHEN -NODEBUG IS SELECTED.)

BREAKPOINTS AT STATEMENT FUNCTIONS ARE NOT ALLOWED. ALSO, THE DEBUGGER WILL NOT RECEIVE INFORMATION ABOUT $INSERT FILES, SO THAT THE SOURCE COMMAND WILL NOT DO ALL THAT MIGHT BE DESIRED.

THE COMMAND LINE OPTIONS AFFECT THE SETTING OF B REGISTER BITS 4 AND 7 AS FOLLOWS:

	4	7
-DEBUG	1	1
-PROD	0	1
-NODEBUG	0	0

2 WARNINGS AND ERRORS

THE COMPILER DISTINGUISHES BETWEEN ERRORS AND WARNINGS, AND REPORTS EACH TOTAL WITH THE PROGRAM UNIT NAME AND THE COMPILER VERSION. IF THERE ARE WARNINGS, THE WARNING TOTAL APPEARS TO THE RIGHT OF THE COMPILER VERSION, IN THE FORM

N WARNING[S]

WHERE N IS THE TOTAL WARNINGS AND THE PLURAL IS OPTIONAL.

EACH WARNING IS REPORTED IN THE LISTING FILE IN A FORMAT SIMILAR TO AN ERROR, EXCEPT THAT WARNING - IS INSERTED FOLLOWING THE CONTEXT.

NEW MESSAGES ARE:

DEBUG TURNS OFF OPT
BOTH THE -DEBUG AND -OPT (OR -UNC) OPTIONS WERE SELECTED. COMPILATION PROCEEDS AS IF THE OPTIMIZATION OPTION DID NOT APPEAR.

NO DEBUG IN R MODE (WARNING)
THE DEBUG OPTION WAS SELECTED FOR A COMPILATION IN A MODE OTHER THAN 64V. COMPILATION PROCEEDS AS IF NO -DEBUG OCCURRED.

NAME - VARIABLE NOT USED (WARNING)
THE COMPILER'S ANALYSIS MAY UNCOVER INSTANCES OF A NAME DECLARED BUT NOT USED. SUCH VARIABLES ARE NOT ACCESSIBLE WHEN USING THE HIGHER LEVEL LANGUAGE DEBUGGER.

NAME - NEVER GIVEN A VALUE (WARNING)
LOCAL VARIABLES ARE CHECKED TO ENSURE THEY HAVE HAD A VALUE ASSIGNED TO THEM AT SOME POINT IN THE PROGRAM.

NAME - PARAMETER IS BETTER (WARNING)
THE USER IS REMINDED WHEN A VARIABLE INITIALIZED IN A DATA STATEMENT AND REMAINING CONSTANT COULD BE HANDLED MORE EFFICIENTLY WITH THE PARAMETER STATEMENT.

THE LATTER THREE WARNINGS WILL BE ISSUED ONLY WHEN -DEBUG IS SELECTED, AND EVEN THEN MAY NOT APPEAR IN EVERY APPLICABLE CIRCUMSTANCE. IF A VARAIPLE IS DECLARED IN A $INSERT FILE BUT NEVER USED, FOR EXAMPLE, THERE IS NO WARNING.

FINALLY, THE COMPILER RECOVERS BETTER FROM THE ERROR CONDITION "NO PATH TO STATEMENT". IF THE PROGRAM IS OTHERWISE CORRECT, THE GENERATED CODE WILL BE EXECUTABLE.

3 STACK FRAME HEADER

THE DEFAULT STACK FRAME HEADER SIZE IS 20 RATHER THAN THE OLD 10. THERE IS A NEW FORTRAN SPECIFICATION STATEMENT AVAILABLE WHEN -SPO IS SELECTED, TO OVERRIDE THE DEFAULT. ITS FORM IS:

```
STACK HEADER <CONSTANT EXPRESSION>
```

THE VALUE OF THE EXPRESSION MUST BE AN INTEGER BETWEEN 0 AND 32767. THE STACK HEADER STATEMENT IS USEFUL ONLY FOR SPECIALIZED OPERATING SYSTEM PROGRAMS, TO CONTROL THE SIZE OF THE STACK FRAME.

4 SHORTCALL CAPABILITY

THE SYNTAX OF THE SHORTCALL SPECIFICATION STATEMENT IS CHANGED TO:

```
SHORTCALL <REF>, <REF> ..., <REF>
```

WHERE EACH <REF> IS

```
<EXTERNAL SYMBOL> [ ( <CONSTANT EXPRESSION> ) ].
```

THE VALUE OF THE EXPRESSION MUST BE AN INTEGER BETWEEN 0 AND 32767. IF (<CONSTANT EXPRESSION>) IS NOT SUPPLIED, THE DEFAULT VALUE IS 8. THE COMPILER RESERVES WORDS OF MEMORY FOR SHORTCALL TEMPORARIES EQUAL TO THE MAXIMUM OF THESE VALUES OVER ALL <REF>S IN SHORTCALL STATEMENTS. THIS SPACE IS RESERVED IMMEDIATELY AFTER THE STACK FRAME HEADER OF THE CURRENT PROGRAM UNIT, AND BEFORE ALL LOCAL STORAGE.

MULTIPLE ARGUMENTS TO SHORTCALLED ENTRIES GENERATE CODE AS FOLLOWS:

NO ARGUMENTS

```
JSXB  LB%+X,*
```

1 ARGUMENT

```
EAL   ARG
JSXB  LB%+X,*
```

N ARGUMENTS (N > 1)

```
EAL   ARG1
STL   SB%+T
EAL   ARG2
STL   SB%+T+3
 <
 <
EAL   ARGN
STL   SB%+T+(N-1)*3
EAL   SB%+T
JSXB  LB%+X,*
```

WHERE T IS THE OFFSET OF A TEMPORARY BLOCK IN THE STACK FRAME.

5 FTNOPT FUNCTIONALITY

FTN FOR REV. 17 WILL INCLUDE THE FEATURES OF REV. 16 FTNOPT, NAMELY OPTIONS FOR DO LOOP OPTIMIZATION AND MODIFICATIONS TO GENERATE MORE EFFICIENT CODE FOR THE P550.

THERE ARE THREE OPTIMIZATION COMMAND LINE OPTIONS:

-STDOPT	STANDARD PRE-REV.16 OPTIMIZATION (DEFAULT)
-UNCOPT	PERFORM DO-LOOP OPTIMIZATION UNCONDITIONALLY
-OPT	PERFORM DO-LOOP OPTIMIZATION WHENEVER THE LOOP CONTAINS NO GO TO STATEMENTS

THESE AFFECT B REGISTER BITS 5 AND 6 AS FOLLOWS:

	5	6
-STDOPT	0	0
-UNCOPT	1	1
-OPT	1	0

-STDOPT AND -OPT ARE SAFE OPTIONS.

6 TARS BEING HANDLED

THE CONCORDANCE WILL BE ABLE TO LIST SOURCE LINE NUMBERS ABOVE 8191. (13971)

THE INTRINSICS LS AND RS WHEN GIVEN A NEGATIVE SHIFT COUNT SHOULD HAVE NO EFFECT. THE COMPILER HAS BEEN INCONSISTENT IN THIS MATTER, PRODUCING IN-LINE CODE INCORRECTLY. THE DOCUMENTATION HAS BEEN IN ERROR. (25264)

A STATEMENT FUNCTION NAME PASSED AS AN ACTUAL ARGUMENT WILL NO LONGER CAUSE THE COMPILER TO HANG IN V MODE COMPILATIONS. (25561)

GENERALIZED SUBSCRIPTS: OFFSET IN AN ARRAY WAS SOMETIMES CALCULATED INCORRECTLY WHEN A CONSTANT IS FOLLOWED BY A - OPERATOR, E.G. ARRAY(5-I-J). (23673)

WHEN THE DCLVAR OPTION IS USED, ONLY THE FIRST UNDECLARED VARIABLE IN A STATEMENT WAS FLAGGED AND OTHERS WERE IGNORED. (15361)

7 SHORTCALL FOR CONVERSION FUNCTIONS

THE MEANS EXIST FOR V-MODE LIBRARY CONVERSION FUNCTIONS (WHICH ARE NORMALLY PROCEDURE CALLED) TO BE INVOKED VIA SHORT CALL.

```
**********************************************************************************
**********************************************************************************
**
**
**
**      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 M    Y Y
**       J      I     M  M  M  M  M  M     Y
**    J  J      I     M     M  M     M     Y
**    J  J      I     M     M  M     M     Y
**     JJ      III    M     M  M     M     Y
**
**
**
**    RRRR     CCC    X    X     1
**    R   R   C   C   X    X    11
**    R   R   C         X X      1
**    RRRR    C          X       1
**    R R     C         X X      1
**    R  R    C   C     X  X     1
**    R   R    CCC      X   X   111
**
**
**
**********************************************************************************
**********************************************************************************
```

CHANGES TO CX FOR REVISIONS 17.1 & 16.7

DATE:

TO:

FROM:

SUBJECT: CHANGES TO CX FOR REVISIONS 17.1 & 16.7

REFERENCE:

ABSTRACT

THE FOLLOWING CHANGES WERE MADE TO THE CX SEQUENTIAL JOB PROCESSOR FOR REVISIONS 16.7 & 17.1:

CX CAN NOW RUN ON A SYSTEM THAT HAS AT LEAST 16 (DECIMAL) FILE UNITS AVAILABLE FOR USE (USING THE FILUNT CONFIGURATION DIRECTIVE).

THE CX DROP COMMAND (CX -DN) HAS BEEN FIXED SO THAT WHEN IT IS ONLY GIVEN ONE DIGIT (EX.: "CX -D6"), IT WILL NO LONGER RETURN THE ERROR "?CAN'T - FILE DELETED FROM COMMAND MODE".

ON A SYSTEM WHICH HAS BEEN CONFIGURED SO THAT THE HIGHEST AVAILABLE UNIT NUMBER (NOT INCLUDING COMMAND OUTPUT) IS 16 DECIMAL OR MORE, CX WILL NOW SUCCESSFULLY RUN. PREVIOUSLY, IT WOULD ASSUME THAT IT COULD USE UNIT 62 (IAR #20546).

WHEN DROPPING A FILE (CX -DN), CX COULD DELETE THE ENTRY AND THEN NOT FIND THE COMMAND FILE, DUE TO A FIX AT REV 16.1 THAT ALLOWED THE DROP AND STATUS COMMANDS TO ACCEPT A SINGLE DIGIT JOB NUMBER ("CX -D6" OR "CX -S5"). THE PROBLEM WAS THAT CX USED THE TEXT IMMEDIATELY FOLLOWING THE "-D" AS THE LAST TWO CHARACTERS OF THE FILE NAME "CX##NN" TO DELETE, RESULTING IN "CX##6 " IN SINGLE-DIGIT CASES.

WHEN THIS OCCURRED, CX WOULD REPORT A DROP ERROR (ALTHOUGH THE DROP ITSELF WOULD BE SUCCESSFUL), AND WOULD NOT DELETE THE COMMAND FILE.

```
**********************************************************************************
**********************************************************************************
**
**
**
**      JJJ    III    M    M   M    M   Y    Y
**       J      I     MM  MM   MM  MM   Y    Y
**       J      I     M M M    M M M     Y Y
**       J      I     M M M    M M M      Y
**    J  J      I     M   M    M   M      Y
**    J  J      I     M   M    M   M      Y
**     JJ       III   M   M    M   M      Y
**
**
**
**    RRRR     CCC    X   X
**    R   R   C   C   X   X
**    R   R   C         X X
**    RRRR    C           X
**    R R     C         X X
**    R  R    C   C     X   X
**    R   R    CCC      X   X
**
**
**
**********************************************************************************
**********************************************************************************
```

CHANGES TO CX FOR REV. 17.0

DATE: APRIL 5, 1979

TO:

FROM:

SUBJECT: CHANGES TO CX FOR REV. 17.0

REFERENCE:

ABSTRACT

CX HAS UNDERGONE MINOR CHANGES FOR REVISION 17. THIS DOCUMENT DESCRIBES THOSE CHANGES AND IN ADDITION WARNS AGAINST RUNNING CX AND EITHER THE BATCH SUBSYSTEM OR THE SPOOL SUBSYSTEM SIMULTANEOUSLY FOR SECURITY REASONS.

THE CHANGE FOR CX REVISION 17 IS THAT THE REVISION NUMBER IS TYPED OUT WHEN A CX PROGRAM IS FIRST RUN, NOT WHEN IT EXITS. EXAMPLE:

```
OK, CX -A
[CX REV 17.0]

?CAN'T - JOB FILE EMPTY

OK,
```

AT REVISION 16.1, THIS WOULD HAVE LOOKED AS FOLLOWS:

```
OK, CX -A

?CAN'T - JOB FILE EMPTY

[CX REV 16.1]
OK,
```

THERE IS ALSO A CHANGE IN THE INSTALLATION OF CX. AT REVISION 16, THE FORMAT OF THE CX DATABASE FILES WAS CHANGED, AND THE FILE NAMES WERE ALSO CHANGED TO MINIMIZE CONFLICTS.

THE NAMES AT REVISION 15 OF THE CX DATABASE FILES WERE "JOBS*" AND "USER#". AT REVISION 16, THESE BECAME "JOBS*T" AND "USER#S" RESPECTIVELY.

BECAUSE THIS CHANGE WAS MADE, THE BUILD PROCEDURE FOR CX WOULD DELETE, FROM THE "CX***" UFD, THE OLD "JOBS*" AND "USER#" FILES, IN ADDITION TO ANOTHER OBSOLETE FILE NAMED "C_SCAN" (WHICH WAS USED TO START UP THE SLAVE PHANTOM, AND WAS REPLACED BY THE USE OF THE OPERATING SYSTEM SUBROUTINE PHANT$).

THEN THE BUILD PROCEDURE RAN THE *INIT PROGRAM, WHICH CREATES NULL "JOBS*T" AND "USER#S" FILES.

AT REVISION 17, HOWEVER, THESE PROCEDURES ARE NOT DONE, BECAUSE THE FORMAT AND NAMES OF THE CX DATABASE FILES HAVE NOT CHANGED. THEREFORE, IT SHOULD BE POSSIBLE TO INSTALL CX REVISION 17 OVER CX REVISION 16.1 WITHOUT CLEARING OUT THE JOB DATA BASE. IN FACT, AS LONG AS THE CX MONITOR IS NOT ITSELF RUNNING, OTHER RUNNING CX JOBS SHOULD NOTICE NO DIFFERENCE AND FINISH JUST AS IF A NEW INSTALL WAS NOT DONE.

THIS, OF COURSE, DEPENDS ON WHETHER OR NOT THE ACTUAL "JOBS*T" AND "USER#S" FILES ARE LEFT UNTOUCHED.

HOWEVER, IF, FOR ANY REASON, AN INSTALLATION INSTALLS REVISION 17 DIRECTLY FROM A REVISION 15 SYSTEM, THEN THE FOLLOWING PROCEDURE SHOULD BE PERFORMED:

```
OK, ATTACH_CX***
OK, DELETE_C_SCAN   /* IGNORE "NOT FOUND" ERROR.
OK, DELETE_JOBS*    /* IGNORE "NOT FOUND" ERROR.
OK, DELETE_USER#    /* IGNORE "NOT FOUND" ERROR.
OK, RESUME_*INIT
[CX REV 17.0]
JOBS*T INITIALIZED.
USER#S INITIALIZED.

OK.
```

AT THIS POINT, THE CX SUBSYSTEM IS READY FOR OPERATION VIA "PHANTOM CX***>PH_GO" FROM A USER TERMINAL OR THE SYSTEM CONSOLE.

THERE IS A SECURITY ISSUE WITH RUNNING CX, HOWEVER; AT PREVIOUS REVISIONS, THE SECURITY ISSUE INVOLVED RUNNING CX AND THE SPOOL SUBSYSTEM AT THE SAME TIME. AT REVISION 17, THIS CONFLICT IS EXTENDED TO THE NEW BATCH SUBSYSTEM.

THE PROBLEM MANIFESTS ITSELF AS FOLLOWS: BOTH SPOOL AND BATCH ALLOW OPERATOR-TO-PHANTOM COMMUNICATION, WHICH FOR SPOOL REFERS TO COMMUNICATION BETWEEN THE OPERATOR AND SPOOLER PHANTOMS, AND FOR BATCH THIS REFERS TO COMMUNICATION BETWEEN THE OPERATOR AND THE RUNNING BATCH MONITOR.

THE SPOOL PROGRAM DETERMINES WHETHER OR NOT THE USER MAKING THE OPERATOR REQUEST HAS THE RIGHT TO DO THIS BY COMPARING THE USER NAMES OF THE USER AND THE RUNNING PHANTOM. GENERALLY, THIS USER NAME IS SYSTEM, BUT THIS IS NOT HARDWIRED.

HOWEVER, BATCH DETERMINES THE RIGHT OF THE USER MAKING THE REQUEST BY MAKING SURE THAT THE USER IS LOGGED IN AS SYSTEM. THIS IS DUE TO THE FACT THAT THE BATCH MONITOR MUST BE SPAWNED FROM THE SYSTEM CONSOLE, WHICH AUTOMATICALLY CAUSES IT TO BE LOGGED IN AS SYSTEM. IN ADDITION, BATCH ALLOWS THE "OPERATOR" (USER LOGGED IN AS SYSTEM) TO PERFORM ALL OPERATIONS ON USER'S JOBS THAT USERS CAN, INCLUDING DISPLAYING STATUS, CANCELLING, ABORTING, AND RESTARTING JOBS. THE PRIVILEDGE TO "HOLD" AND "RELEASE" JOBS IS ONLY EXTENDED TO THE USER LOGGED IN AS SYSTEM.

THE SECURITY PROBLEM THEREFORE ARISES WITH THE CX SUBSYSTEM, SINCE IT USUALLY RUNS UNDER THE LOGIN NAME OF SYSTEM. USERS CAN SUBMIT CX JOBS THAT PERFORM PRIVILEDGED OPERATIONS ON EITHER THE BATCH OR SPOOL SUBSYSTEM, BECAUSE USER JOBS IN CX RUN UNDER THE LOGIN NAME OF THE CX MONITOR (USUALLY "SYSTEM"), NOT THE USER'S LOGIN NAME.

THEREFORE, UNLESS THE CX MONITOR IS LOGGED IN UNDER A USER-NAME OTHER THAN "SYSTEM", RUNNING CX IN TANDEM WITH BATCH PRESENTS A SECURITY ISSUE.

IT IS SUGGESTED THAT THE ADMINISTRATOR WHO DESIRES TO USE THE NEW BATCH SUBSYSTEM NOT RUN CX AT REVISION 17, AS CONVERSION HAS BEEN MADE AS EASY AS POSSIBLE FOR THE USERS (THEY HAVE NO NEED TO CHANGE THEIR COMMAND FILES TO RUN UNDER BATCH).

IF, HOWEVER, AN INSTALLATION REQUIRES THE SIMULTANEOUS RUNNING OF BOTH CX AND BATCH OR SPOOL, A POSSIBLE SOLUTION WOULD BE TO RUN THE CX PHANTOM UNDER A DIFFERENT LOGIN NAME (SUCH AS "CXUSER"). THIS REQUIRES THE OPERATOR TO LOG INTO A TERMINAL UNDER THAT LOGIN NAME AND SPAWN THE CX PHANTOM BY HAND WHENEVER THE SYSTEM IS COLD-STARTED.

```
**********************************************************************************
**********************************************************************************
**
**
**
**      JJJ   III   M   M   M   M  Y   Y
**       J     I    MM MM  MM MM  Y   Y
**       J     I    M M M  M M M   Y Y
**       J     I    M M M  M M M    Y
**   J   J     I    M   M  M   M    Y
**   J   J     I    M   M  M   M    Y
**    JJ     III   M   M  M   M    Y
**
**
**
**   RRRR   DDD   BBBB    AAA    SSS    III    CCC
**   R   R  D  D  B   B  A   A  S   S    I    C   C
**   R   R  D   D B   B  A   A  S        I    C
**   RRRR   D   D BBBB   AAAAA   SSS     I    C
**   R R    D   D B   B  A   A      S    I    C
**   R  R   D   D B   B  A   A  S   S    I    C   C
**   R   R  DDD  BBBB   A   A   SSS    III    CCC
**
**
**
**********************************************************************************
**********************************************************************************
```

STATUS OF TARS FOR THIS, REV 16.4 DBASIC:

STATUS OF TARS FOR THIS, REV 16.4 DBASIC:

12546,80582 - PRINT USING JUXTAPOSED ITEMS WHEN THE FIRST NUMERIC ITEM OVERFLOWED. FIXED.

13717 - .NL. DID NOT RESET COLUMN COUNT IN ENTER STATEMENT. FIXED.

24728 - STATEMENT NUMBER 0 WAS NOT SENSED AS AN ERROR. FIXED.

15819 - PRINT USING ROUNDING IS NOT CONSISTENT. MACHINE FLOATING ACCURACY IS THE PROBLEM HERE, BUT NOTE THAT THE ACTUAL COMPUTATION ACCURACY IS NOT AFFECTED BY THIS PROBLEM, WHICH IS DUE TO THE INPUT CONVERSION OF ASCII DIGITS TO FLOATING NUMBERS. A BETTER METHOD IS USED BY BASIC/VM AND FORTRAN, SO THESE PROBLEMS WILL NOT SHOW UP.

80236,80469 - HALTS ARE ENCOUNTERED WHEN STRINGS ARE PASSED TO A FORTRAN PROGRAM. THE DOCUMENTATION IS WRONG, AND INDEED STRINGS ARE NOT ALLOWED TO BE PASSED TO A FORTRAN PROGRAM.

22783 - A 'FOR-NEXT UNMATCHING' ERROR WAS GENERATED WHEN IN FACT NO MISMATCH EXISTED. FIXED.

```
**********************************************************************************
**********************************************************************************
**
**
**
**      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     Y
**    J  J      I     M     M  M     M     Y
**    J  J      I     M     M  M     M     Y
**     JJ      III    M     M  M     M     Y
**
**
**
**    PRRR    DDDD     000     SSS
**    R   R   D   D   0   0   S   S
**    R   R   D   D   0   0   S
**    PRRR    D   D   0   0    SSS
**    R R     D   D   0   0       S
**    R  R    D   D   0   0   S   S
**    P   R   DDDD     000     SSS
**
**
**
**********************************************************************************
**********************************************************************************
```

DOS CHANGES FOR REV. 16.8

DATE:

TO:

FROM:

SUBJECT: DOS CHANGES FOR REV. 16.8

REFERENCE:

ABSTRACT

PE-T-666 DESCRIBES DOS CHANGES FOR REVISION 16.8

1 AUTOMATIC STARTUP OF BOOT DISK

DOS HAS BEEN MODIFIED TO PERFORM AN AUTOMATIC STARTUP OF THE DISK FROM WHICH DOS WAS BOOTED. THE MESSAGE WHICH DOS PRINTS WHEN IT IS FIRST BOOTED HAS BEEN MODIFIED TO APPEAR IN THE FORM SHOWN IN THE FOLLOWING EXAMPLE:

```
PRIMOS II REV. 16.8 07/02/79 (AT 170000)
STARTING UP DISK 000460
```

THIS CHANGE ELIMINATES THE NEED TO EXPLICITLY STARTUP THE BOOT DISK BEFORE ISSUING OTHER COMMANDS TO DOS. IF THE USER WISHES TO STARTUP A DIFFERENT PARTITION THAN THE BOOT DISK (E.G., IF HE BOOTS DISK 460 BUT WANTS 10460 STARTED UP), HE CAN STILL ISSUE THE APPROPRIATE STARTUP COMMAND TO DOS. THIS CHANGE WILL WORK FOR ALL DISK TYPES CURRENTLY SUPPORTED BY DOS.

2 ADDISK SVC

A NEW SVC HAS BEEN ADDED TO DOS TO ALLOW A PROGRAM RUNNING UNDER DOS TO ADD ADDITIONAL FILE SYSTEM PARTITIONS. THE CALLING SEQUENCE IS AS SHOWN:

```
INTEGER*2   PDEV        /*PHYSICAL DISK NUMBER
INTEGER*2   LDEV        /*LOGICAL DISK NUMBER
INTEGER*2   CODE        /*ERROR CODE

CALL ADDISK (PDEV, LDEV, CODE)
```

THE REQUESTED PDEV IS STARTED UP, AND ITS LDEV IS RETURNED. IF THE PDEV IS NOT A VALID PARTITION, A NON-ZERO CODE IS RETURNED AND THE PDEV IS NOT ADDED. THE SVC CODE FOR THE ADDISK CALL IS '1527.

NOTE THAT THE ADDISK SVC IS NOT SUPPORTED BY A FORTRAN LIBRARY INTERLUDE. IN ADDITION, ITS FUNCTIONALITY IS NOT SUPPORTED UNDER PRIMOS. IT IS INTENDED FOR USE ONLY BY THE PRIMOS PRELOADER.

```
**********************************************************************************
**********************************************************************************
**
**
**
**      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     Y
**    J  J      I     M   M   M   M      Y
**    J  J      I     M   M   M   M      Y
**     JJ       III   M   M   M   M      Y
**
**
**
**    RRRR     CCC     OOO    N   N   CCC     AAA    TTTTT
**    R   R   C   C   O   O   NN  N  C   C   A   A     T
**    R   R   C       O   O   N N N  C       A   A     T
**    RRRR    C       O   O   N N N  C       AAAAA     T
**    R R     C       O   O   N N N  C       A   A     T
**    R  R    C   C   O   O   N  NN  C   C   A   A     T
**    P   R    CCC     OOO    N   N   CCC    A   A     T
**
**
**
**********************************************************************************
**********************************************************************************
```

DATE: FEBRUARY 15, 1979

SUBJECT: "CONCAT" COMMAND - CONCATENATE FILES FOR SPOOLING

REFERENCE: NONE

ABSTRACT

THE FOLLOWING COMMAND WILL BE ADDED TO THE STANDARD COMMAND LIBRARY, CMDNCO, AT REV. 17. THIS COMMAND, CONCAT, PERFORMS THE FUNCTION OF THE EXPERIMENTAL X.CONCAT COMMAND, BUT HAS SIGNIFICANT FUNCTIONAL AND HUMAN-FACTOR IMPROVEMENTS. CONCAT AND X.CONCAT ARE NOT COMPATIBLE.

THE COMPLETE DOCUMENTATION FOR CONCAT IS ATTACHED. PLEASE DIRECT COMMENTS ON THE SPECIFICATION OF THE COMMAND OR ITS DOCUMENTATION TO E. STANMYER OR G. KESSLER AS SOON AS POSSIBLE.

THIS REVISION CORRECTS AN ERROR IN THE PREVIOUS DOCUMENT AND REFLECTS A CHANGE IN ERROR HANDLING FOR THE INSERT COMMAND.

NAME: CONCAT

PURPOSE:

THE COMMAND "CONCAT" IS USED TO COMBINE A NUMBER OF INPUT FILES INTO AN OUTPUT FILE SUITABLE FOR SPOOLING. VARIOUS OPTIONS ALLOW THE USER TO SPECIFY INPUT AND OUTPUT FILE UNITS, TITLE AND BANNER GENERATION AND SUPPRESSION, AND SO ON. IN ADDITION, CONCAT OPERATES IN A COMMAND MODE, READING ITS COMMANDS FROM THE TERMINAL OR A COMMAND FILE.

USAGE:

CONCAT IS INVOKED FROM PRIMOS COMMAND LEVEL. VARIOUS OPTIONS CAN BE SPECIFIED ON THE COMMAND LINE AS EXPLAINED BELOW UNDER "INVOCATION". AFTER PROCESSING THE COMMAND LINE, CONCAT ENTERS ONE OF TWO MODES OF EXECUTION. IF -COMMAND IS SPECIFIED ON THE COMMAND LINE, COMMAND MODE IS ENTERED. IN THIS MODE COMMANDS MAY BE GIVEN TO CHANGE THE OPERATING ENVIRONMENT, FOR EXAMPLE RESETTING HEADERS, BANNERS AND PAGE NUMBERS. IF -COMMAND IS NOT SPECIFIED ON THE COMMAND LINE, INSERT MODE IS ENTERED AND CONCAT ACCEPTS A LIST OF INPUT FILES. SEE "MODES OF EXECUTION" BELOW FOR FURTHER EXPLAINATION.

INVOCATION:

CONCAT [OUTFILE] [OPTIONS] ...

OUTFILE IS THE OPTIONAL TREENAME SPECIFYING THE PATH TO THE OUTPUT FILE. IF OUTFILE IS OMITTED, THE FILE OPEN ON THE OUTPUT UNIT IS USED INSTEAD. (THE DEFAULT OUTPUT UNIT IS UNIT 2 BUT CAN BE CHANGED WITH THE -OUNIT OPTION.)

OPTIONS EXCEPT -BANNER, WHICH MUST BE LAST, ARE SELECTED IN ANY ORDER (FOLLOWING OUTFILE IF SPECIFIED) FROM THE FOLLOWING SET OF CONTROL ARGUMENTS. ALL OPTIONS CAN BE ABBREVIATED TO THREE LETTERS EXCEPT FOR -DELETE WHICH HAS NO ABBREVIATION. THEY FALL INTO VARIOUS CLASSES:

FILE UNITS (-IUNIT, -OUNIT)

OUTFILE VERIFICATION (-VERIFY, -OVERWRITE, -APPEND)

OUTFILE FORMATTING (-HEADER, -BANNER, -NHEADER, -EJECT. -RESETP, NRESETP)

OUTFILE DISPOSITION (-CLOSE, -OPEN, -TRUNCATE)

INPUT FILE DELETION (-DELETE, -NDELETE)

ON THE COMMAND LINE THE FOLLOWING COMMAND ARGUMENTS ARE AVAILABLE:

-IUNIT N

SPECIFIES THE UNIT ON WHICH THE INPUT FILES WILL BE OPENED. THE DEFAULT UNIT IS 1.

-OUNIT N

SPECIFIES THE UNIT ON WHICH THE OUTPUT FILE WILL BE OPENED. THE DEFAULT UNIT IS 2. IF OUTFILE IS OMITTED FROM THE COMMAND LINE, THE FILE OPEN ON UNIT N WILL BE USED FOR OUTPUT.

-VERIFY

IF OUTFILE ALREADY EXISTS, CAUSES AN "OK TO MODIFY OLD" QUERY FOLLOWED BY AN "OVERWRITE OR APPEND" QUERY. THIS IS THE DEFAULT MODE.

-OVERWRITE

CAUSES OUTFILE TO BE OVERWRITTEN IF IT ALREADY EXISTS.

-APPEND

CAUSES OUTPUT TO BE APPENDED TO OUTFILE IF IT ALREADY EXISTS.

-HEADER

SPECIFIES TITLE GENERATION AND BANNER PAGE SUPPRESSION. THIS IS THE DEFAULT. THE FIRST LINE OF EACH INPUT FILE IS READ. IF IT IS A TITLE (BEGINS WITH OCTAL 1 IN THE LEFT BYTE), THEN THE LINE ONLY APPEARS AS THE TITLE FOR THE FILE. OTHERWISE THE LINE APPEARS BOTH AS THE TITLE LINE AND AS THE FIRST LINE OF THE FILE.

-BANNER [BANNER_LINE]

SPECIFIES TITLE AND BANNER PAGE GENERATION. MUST BE THE LAST OPTION ON THE LINE. A BANNER PAGE IS INSERTED BETWEEN INPUT FILES. IT CONTAINS TWO LINES OF UP TO 14 LARGE CHARACTERS. THE BANNER_LINE SPECIFIES THE FIRST OF THESE LINES. IT IS READ AS RAW TEXT SO THAT SPACES ARE ACCEPTED. THE SECOND LINE IS THE LAST COMPONENT OF THE INPUT FILE TREENAME. TITLES ARE GENERATED AS IN -HEADER MODE. THE BANNER PAGE HAS THE SAME TITLE AS THE ROUTINE FOLLOWING IT. IF THE OPTIONAL BANNER_LINE IS OMITTED, THEN THAT LINE OF THE BANNER WILL BE LEFT BLANK.

-NHEADER

SUPPRESSES BOTH TITLE AND BANNER PAGE GENERATION. THE INPUT FILES ARE COPIED TO THE OUTPUT FILE WITHOUT MODIFICATION.

-EJECT

GENERATES A PAGE EJECT BETWEEN INPUT FILES. SUPPRESSES TITLE AND BANNER PAGE GENERATION.

-RESETP

RESETS SPOOLER PAGE NUMBERING BETWEEN INPUT FILES.

-NRESETP

DOES NOT RESET SPOOLER PAGE NUMBERING BETWEEN INPUT FILES. THIS IS THE DEFAULT.

-CLOSE

TRUNCATES AND CLOSES OUTFILE ON EXIT. THIS IS THE DEFAULT.

-OPEN

LEAVES OUTFILE OPEN ON EXIT. NO TRUNCATION OF OUTFILE IS DONE.

-TRUNCATE

ON EXIT, TRUNCATES OUTFILE BUT LEAVES IT OPEN.

-DELETE

DELETES INPUT FILE AFTER COPYING IT TO THE OUTPUT FILE. THIS OPTION HAS NO ABBREVIATION.

-NDELETE

DOES NOT DELETE INPUT FILE AFTER COPYING IT TO THE OUTPUT FILE. THIS IS THE DEFAULT.

-INSERT

GOES DIRECTLY INTO INSERT MODE (PROMPT IS ':') AND ACCEPTS A LIST OF FILES TO BE INSERTED INTO THE OUTPUT FILE. IF NEITHER -INSERT NOR -COMMAND IS SPECIFIED ON THE COMMAND LINE -INSERT IS ASSUMED.

-COMMAND

GOES INTO COMMAND MODE (PROMPT IS '>').

MODES OF EXECUTION

CONCAT RUNS IN TWO MODES, COMMAND MODE AND INSERT MODE. WHEN CONCAT STARTS RUNNING IT TYPES OUT " CONCAT REVXX.X"AND A NEW LINE. IF AN ERROR IS MADE IN THE CONCAT COMMAND LINE, CONCAT WILL ABORT. IF NO ERRORS WERE ENCOUNTERED AND -COMMAND WAS NOT SPECIFIED, INSERT MODE IS ENTERED AND A ':' CHARACTER WILL BE TYPED (THE PROMPT FOR INSERT MODE.) IF -COMMAND IS SPECIFIED ON THE COMMAND LINE, COMMAND MODE IS ENTERED AND A '>' PROMPT WILL BE TYPED.

INSERT MODE

FILES TO BE CONCATENATED MAY BE ENTERED ONE PER LINE (TREENAMES ARE ACCEPTED.) THE EXIT FROM INSERT MODE IS A BLANK OR NULL LINE. EXITING FROM INSERT MODE CAUSES CONCAT TO ENTER COMMAND MODE AND PROMPT WITH A '>'.

COMMAND MODE

COMMANDS ARE ENTERED ONE PER LINE. BLANK LINES ARE IGNORED. THE CHARACTERS '/*' CAUSE THE REST OF THE LINE TO BE IGNORED. THE EXIT FROM COMMAND MODE IS 'QUIT' WHICH EXITS FROM CONCAT. ALL COMMANDS MAY BE ABBREVIATED TO THREE LETTERS EXCEPT DELETE WHICH HAS NO ABBREVIATION AND QUIT WHICH MAY BE ABBREVIATED TO ONE LETTER.

VARIOUS COMMANDS ARE RECOGNIZED IN COMMAND MODE. MOST CAN ALSO BE USED AS COMMAND LINE OPTIONS. THEY ARE:

HEADER
: GENERATE TITLES BUT SUPPRESS BANNER PAGES

BANNER [BANNER_LINE]
: GENERATE BOTH BANNERS AND TITLES. BANNER_LINE IS THE FIRST LINE OF THE BANNER. IF OMITTED THAT LINE IS LEFT BLANK.

NHEADER
: SUPPRESS BOTH TITLES AND BANNER PAGES. FILES ARE COPIED WITHOUT CHANGE.

EJECT
: GENERATE PAGE EJECT BETWEEN FILES. SUPPRESS BOTH TITLES AND BANNER PAGES.

RESETP
: RESET SPOOLER PAGE NUMBERING BETWEEN INPUT FILES

NRESETP

DO NOT RESET PAGE NUMBERING

DELETE

DELETE INPUT FILES AFTER COPYING TO OUTPUT FILE. THIS OPTION HAS NO ABBREVIATION.

NDELETE

DO NOT DELETE INPUT FILES AFTER COPYING

TITLE [NEW_TITLE]

USE NEW_TITLE AS THE HEADER FOR THE NEXT INPUT FILE. IT IS READ AS RAW TEXT SO THAT SPACES ARE ACCEPTED. IF NEW_TITLE IS OMITTED THE FILE NAME IS USED.

QUIT

EXIT FROM CONCAT. THIS IS THE ONLY CLEAN WAY TO EXIT FROM CONCAT. THIS OPTION HAS A SINGLE LETTER ABBREVIATION.

INSERT [FILE_NAME_LIST]

IF FILE_NAME_LIST IS OMITTED, CONCAT WILL GO INTO INSERT MODE AND ACCEPT THE NAMES OF THE FILES TO BE CONCATENATED ONE PER LINE. TO EXIT FROM INSERT MODE, A BLANK OR NULL LINE MAY BE ENTERED. IF FILE_NAME_LIST IS SPECIFIED, THE FILES IN THE LIST ARE CONCATENATED INTO THE OUTPUT FILE WITHOUT ENTERING INSERT MODE. IF AN ERROR IS MADE IN THE LINE, THE REST OF THE LINE AFTER THE ERROR IS IGNORED. UP TO FORTY FILES MAY BE SPECIFIED ON ONE LINE, SEPARATED BY SPACES OR COMMAS. TREENAMES WITH IMBEDDED SPACES MUST BE ENCLOSED IN QUOTES (WHEN PASSWORDS ARE NEEDED.)

DEFAULTS:

THE DEFAULT OPTIONS FOR CONCAT ARE:

```
-IUNIT 1
-OUNIT 2
-HEADER
-NRESETP
-NDELETE
-VERIFY      (IF OUTFILE IS PROVIDED)
-CLOSE       (IF OUTFILE IS PROVIDED)
-OVERWRITE   (IF OUTFILE IS OMITTED)
-OPEN        (IF OUTFILE IS OMITTED)
-INSERT
```

(END)

```
**********************************************************************
**********************************************************************
**
**
**
**      JJJ    III    M    M   M    M   Y    Y
**       J      I     MM  MM   MM  MM   Y    Y
**       J      I     M M M    M M M     Y Y
**       J      I     M  M M   M  M M      Y
**   J   J      I     M    M   M    M      Y
**   J   J      I     M    M   M    M      Y
**    JJ       III    M    M   M    M      Y
**
**
**
**   RRRR    FFFFF   TTTTT   N    N  L          III    BBBB
**   R   R   F         T     NN   N  L           I     B   B
**   R   R   F         T     N N  N  L           I     B   B
**   RRRR    FFFF      T     N N  N  L           I     BBBB
**   R R     F         T     N N  N  L           I     B   B
**   R  R    F         T     N  NN  L    L      I     B   B
**   R   R   F         T     N    N  LLLLL    III    BBBB
**
**
**
**********************************************************************
**********************************************************************
```

FORTRAN LIBRARIES FOR REV. 17.0

DATE: APRIL 5, 1979

TO:

FROM:

SUBJECT: FORTRAN LIBRARIES FOR REV. 17.0

REFERENCE:

ABSTRACT

THIS DOCUMENT COVERS THREE AREAS OF INTEREST TO FORTRAN LIBRARY USERS:

1. NEW FORTRAN LIBRARY ROUTINES.
2. MODIFIED FORTRAN LIBRARY ROUTINES.
3. REORGANIZATION OF THE V-MODE FORTRAN LIBARAY.

1. NEW FORTRAN LIBRARY ROUTINES

ALL NEW ROUTINES FOR REV. 17.0 ARE DIRECT-ENTRANCE PRIMOS CALLS, AVAILABLE IN V-MODE ONLY. THEY ARE SIMPLY LISTED HERE--FOR COMPLETE DOCUMENTATION SEE , PRIMOS FOR REV. 17. THE NEW CALLS ARE MKONU$, RVONU$, SIGNL$, MKON$F, RVON$F, SGNL$F, MKLB$F, CNSIG$, COMLV$, CMLV$E, CL$GET, TTY$RS, AND GPATH$.

2. MODIFIED FORTRAN LIBRARY ROUTINES

NEW FUNCTIONALITY:

R3POFH — THE SHARED LIBRARY POINTER FAULT HANDLER HAS BEEN MODIFIED TO USE A NEW INTERFACE TO REV. 17 PRIMOS. NOTE THAT REV. 15 OR 16 SHARED LIBRARIES WILL NOT WORK WITH REV. 17 PRIMOS AND REV. 17 SHARED LIBRARIES WILL NOT WORK WITH PRE-REV. 17 VERSIONS OF PRIMOS.

MAIN — THE SHARED LIBRARY INSTALLER PROGRAM ALSO HAS A NEW INTERFACE TO REV. 17 PRIMOS.

F$IO — THE FORTRAN READ/WRITE/ENCODE/DECODE STATEMENT PROCESSOR HAS BEEN UPDATED:

1) IT NOW WORKS WITH DBG, THE HIGH LEVEL LANGUAGE DEBUGGER. SINGLE STEPPING AND VALUE TRACING MAY CAUSE A "FORMAT ERROR" IF DBG IS USED WITH A PRE-REV. 17 F$IO.

2) THE SPSS VARIANT OF FIO, SIO, HAS BEEN MERGED INTO F$IO AS A CONDITIONAL ASSEMBLY PARAMETER. "PMA F$IO 2/0" GIVES A NORMAL F$IO, WHILE "2/1" GIVES THE SPSS VERSION.

F$IOBF — THE FORTRAN I/O BUFFER HAS BEEN INCREASED IN SIZE TO 128 WORDS IN THE R-MODE (TAR24779, REV. 16.1) AND NON-SHARED V-MODE (REV. 15.0) LIBRARIES, AND 6K WORDS IN THE SHARED V-MODE LIBRARY (REV. 16.1).

F$HT — THE FORTRAN STOP/PAUSE STATEMENT PROCESSOR NOW SIGNALS "STOF$" AND "PAUSF$" AFTER PRINTING THE APPROPRIATE MESSAGE. THESE CONDITIONS ARE CAUGHT BY THE DEBUGGER.

SLEEP$ — THE R-MODE VERSION OF THIS ROUTINE IS NOW SMALLER, AS SPECIAL CODE REQUIRED FOR P300 SUPPORT HAS BEEN REMOVED.

TSRC$$ — HAS BEEN MADE A DIRECT-ENTRANCE CALL INTO PRIMOS FOR V-MODE. BOTH THE V-MODE AND R-MODE VERSIONS NOW SUPPORT UP TO 62 LOGICAL DISKS.

BUGS FIXED:

F$LT7X, F$RT7X — THE FORTRAN LEFT AND RIGHT TRUNCATE ROUTINES FOR LONG INTEGERS NOW GIVE VALID RESULTS FOR CALLS WHOSE SECOND ARGUMENT IS 0 MOD 16. (TAR10548, REV. 16.1)

F$IO — THE FORTRAN READ/WRITE/ENCODE/DECODE STATEMENT PROCESSOR NOW:

1) WILL NOT LOOP TRYING TO DECODE A BLANK FIELD. IT WILL INSTEAD TAKE THE ERROR RETURN IF ONE IS PROVIDED, OR GIVE THE "FORMAT/DATA MISMATCH" ERROR IF THERE IS NO ERROR RETURN. (TAR15360, REV. 16.1)

2) USES THE SCALE FACTOR PROPERLY ON INPUT. (TAR80595, REV. 16.1)

3) WILL READ AN "E" FORMAT CONSTANT CONTAINING BLANKS AFTER THE "E", SUCH AS "1.2E 7", AS A SINGLE NUMBER. (TAR13536, REV. 16.1)

4) CORRECTLY PROCESSES FREE FORMAT INPUT OF COMPLEX CONSTANTS IN V-MODE. R-MODE F$IO HAS NOT YET BEEN FIXED AND REQUIRES A BLANK FOLLOWING BOTH THE REAL AND IMAGINARY PARTS, EG "(1,2)" MUST BE INPUT AS "(1 ,2)". (REV. 17.0)

SETSIZ — 1) NO LONGER BOMBS OUT PROGRAMS WHOSE COMMAND LINES CONTAIN 8'S OR 9'S. (TAR15451, REV. 15.4, 16.1)

2) WILL NEVER GO INTO A LOOP UNDER PRIMOS II IN 32R MODE. (REV. 15.5, 16.4, 17.0)

TSRC$$ — NOW WORKS PROPERLY WITH K$GFTU. (REV. 16.2)

DSQR$X, DLOG$X, DEXP$X — MINOR BUGS IN THESE SHORT-CALLED SCIENTIFIC FUNCTIONS HAVE BEEN FIXED. (REV. 17.0)

ERRST$ — NO LONGER USES THE LIST COMMON BLOCK IN V-MODE. (REV. 17.0)

3. REORGANIZATION OF THE V-MODE FORTRAN LIBRARY

THE V-MODE FORTRAN LIBRARY SOURCES HAVE BEEN REORGANIZED TO MAKE INDIVIDUAL ROUTINES EASIER TO FIND AND THE LIBRARIES EASIER TO BUILD. THE DIRECTORIES FLIB1V, FLIB2V, FLIB3V, FLIB4V, FLIB6V, FLIB7V, TYPRSV, DOSPKV, IOCSV, AND VDSPK$ HAVE BEEN ELIMINATED, ALONG WITH THE FILES C_VFTL, C_VLIB, AND BT1IOR IN UFD LIB. A NEW DIRECTORY HAS BEEN CREATED CALLED VFTNLIB WHICH CONTAINS THE SUB-DIRECTORY SOURCES AND THE FILE C_VFTNLIB. SUB-DIRECTORIES BINARIES, LIB, AND SYSTEM WILL ALSO BE CREATED DURING THE LIBRARY BUILD PROCESS AND DELETED AFTER THE NEW LIBRARIES ARE INSTALLED ON THE SYSTEM UFD'S.

TO REBUILD AND INSTALL THE V-MODE FORTRAN LIBRARIES, SIMPLY SAY

"C) VFTNLIB>C_VFTNLIB". THIS COMMAND FILE INVOKES SEVERAL SUBSIDIARY COMMAND FILES ON THE SUB-UFD SOURCES: C_COMPILE CREATES THE SUB-UFD BINARIES AND COMPILES ALL THE SOURCE MODULES INTO THIS SUB-UFD. C_NPFTNLB CREATES THE NON-SHARED V-MODE LIBRARY ON SUB-UFD LIB; C_PFTNLB CREATES THE SHARED V-MODE LIBRARY ON SUB-UFD LIB, THE SUB-UFD SYSTEM, AND THE SHARED SEGMENT FILES ON THAT SUB-UFD; C_INSTALL COPIES THE CREATED FILES FROM THEIR RESPECTIVE SUB-UFD'S TO THE SYSTEM UFD OF THE SAME NAME. C_INSTALL WILL NEED TO BE MODIFIED IF THE UFD'S LIB OR SYSTEM ARE PASSWORDED.

BESIDES THE SUBSIDIARY COMMAND FILES, SUB-UFD SOURCES CONTAINS ALL THE LIBRARY SOURCES AND THE FILE ENTRY_POINT_DIRECTORY. THIS FILE IS AN INDEX OF SOURCES FILE NAMES SORTED BY ENTRY POINT (SUBROUTINE) NAMES. FOR EXAMPLE, IT INDICATES THAT THE SUBROUTINE C$M13 IS CONTAINED IN THE SOURCE FILE VC$M05.

A NEW FILE HAS BEEN ADDED TO THE UFD LIB. THE FILE M_SFTN IS A MAP OF THE SHARED PORTION OF THE FORTRAN LIBRARY, PRESENTLY IN SEGMENT 2014. IF A PROGRAM DIES IN THIS SEGMENT, CONSULTING THIS MAP WILL TELL THE USER THE ROUTINE IN WHICH IT DIED.

```
**********************************************************************************
**********************************************************************************
**
**
**
**      JJJ    III    M    M   M    M   Y    Y
**       J      I     MM  MM   MM  MM   Y    Y
**       J      I     M M M    M M M     Y Y
**       J      I     M M M    M M M      Y
**   J   J      I     M   M    M   M      Y
**   J   J      I     M   M    M   M      Y
**    JJ       III    M   M    M   M      Y
**
**
**
**   RRRR   L        AAA    BBBB    EEEEE  L
**   R   R  L       A   A   B   B   E      L
**   R   R  L       A   A   B   B   E      L
**   RRRR   L       AAAAA   BBBB    EEEE   L
**   R R    L       A   A   B   B   E      L
**   R  R   L   L   A   A   B   B   E      L   L
**   R   R  LLLLL   A   A   BBBB    EEEEE  LLLLL
**
**
**
**********************************************************************************
**********************************************************************************
```

SUBJECT: LABEL COMMAND

LABEL INITIALIZES MAGNETIC TAPES JUST AS MAKE INITIALIZES DISKS. LABEL WRITES EITHER IBM (9-TRACK EBCDIC OR 7-TRACK BCD) OR ANSI (9-TRACK ASCII) LEVEL 1 VOLUME LABELS FOLLOWED BY DUMMY HDR1 AND EOF1 LABELS. LABEL CAN ALSO BE USED TO READ EXISTING VOL1 AND HDR1 LABELS. ANSI LABELS ARE WRITTEN IN ACCORDANCE WITH THE AMERICAN NATIONAL STANDARDS INSTITUTE STANDARD ANSI X3.27-1978. IBM LABELS ARE WRITTEN IN ACCORDANCE WITH IBM'S SPECIFICATIONS (IBM MANUAL GC28-6680-5). ANY NON-STANDARD LABELS SUCH AS 7-TRACK ASCII OR USER-DEFINED LABELS CANNOT BE READ OR WRITTEN.

TO READ EXISTING LABELS TYPE THE COMMAND:

LABEL MTN [-TYPE TYPE]

TO WRITE LABELS TYPE THE COMMAND:

LABEL MTN [-TYPE TYPE] -VOLID VOLUMEID [-OWNER OWNER] [-ACCESS ACCESS]

MTN	IS THE TAPE DRIVE WHERE THE TAPE TO BE LABELLED IS LOCATED. N IS A NUMBER BETWEEN 0 AND 7. THIS KEYWORD IS REQUIRED AND MUST BE THE FIRST ON THE COMMAND LINE.
TYPE	IS THE TYPE OF LABEL DESIRED: -TYPE A = 9-TRACK ASCII (ANSI) (THIS IS THE DEFAULT) -TYPE B = 7-TRACK BCD (IBM) -TYPE E = 9-TRACK EBCDIC (IBM)
VOLUME-ID	IS A 1-6 CHARACTER STRING WHICH UNIQUELY IDENTIFIES THIS TAPE REEL. IF LESS THAN 6 CHARACTERS ARE SPECIFIED, THEY ARE BLANK-PADDED ON THE RIGHT. THE KEYWORDS "-VOLUME" OR "-VOL" MAY BE SUBSTITUTED FOR THE KEYWORD "-VOLID".
OWNER	IS 1-14 CHARACTERS LONG FOR ANSI LABELS, 1-10 CHARACTERS LONG FOR IBM LABELS. IF LESS THAN 14 (OR 10) CHARACTERS IS SPECIFIED, THEY ARE BLANK-PADDED ON THE RIGHT. IF THIS KEYWORD IS OMITTED, THE DEFAULT IS THE USER'S LOGIN NAME. THE KEYWORD "-OWN" MAY BE SUBSTITUTED FOR THE KEYWORD "-OWNER".
ACCESS	IS A SINGLE CHARACTER DEFINING ACCESS TO THIS TAPE. ACCESS IN NOT USED BY PRIME SOFTWARE BUT IS INCLUDED FOR COMPLETENESS. IF IT IS OMITTED, IT IS LEFT BLANK ON ANSI LABELS. ACCESS IS IGNORED FOR IBM LABELS.

IMPROPER USE OF THE LABEL COMMAND CAUSES AN ERROR MESSAGE TO BE PRINTED. THESE ERRORS ARE THE RESULT OF BAD SYNTAX IN THE LABEL COMMAND ITSELF OR A SYSTEM MAGNETIC TAPE I/O ERROR.

SYNTAX ERRORS

***DUPLICATE KEYWORD DETECTED

1. THE SAME KEYWORD WAS TYPED MORE THAN ONCE

***INVALID TAPE UNIT SPECIFIED

2. SOMETHING OTHER THAN MT0-MT7 WAS TYPED

***VOLUME ID SPECIFIED IS TOO LONG

3. THE VOLUME ID CANNOT BE LONGER THAN 6 CHARACTERS

***OWNER ID SPECIFIED IS TOO LONG

4. THE OWNER ID CANNOT BE LONGER THAN 14 CHARACTERS

***INVALID LABEL TYPE SPECIFIED

5. LABEL TYPE MUST BE ONE OF THE CHARACTERS "A", "E", OR "B"

***NO MAGNETIC TAPE UNIT SPECIFIED

6. A MAGNETIC TAPE UNIT IS REQUIRED

***VOLUME ID WAS NOT SPECIFIED

7. WHEN WRITING LABELS, A VOLUME ID IS REQUIRED

***OWNER ID SPECIFIED IS TOO LONG FOR TYPES B OR E

8. THE OWNER ID FOR IBM LABELS CANNOT BE LONGER THAN 10 CHARACTERS

***UNABLE TO WRITE TAPE LABEL ON THIS TAPE

9. A MAG TAPE WRITE ERROR OCCURRED & THE LABEL WAS NOT WRITTEN

***UNABLE TO READ TAPE LABEL ON THIS TAPE

10. A MAG TAPE READ ERROR OCCURRED & THE LABEL WAS NOT READ

***LABEL OPERATION ABORTED

11. ERROR 9,10,12, OR 14 OCCURRED AND LABEL ABORTS

***LABEL READ WAS NOT TYPE X

12. THE LABEL READ WAS NOT OF THE TYPE SPECIFIED

***ACCESS IGNORED FOR IBM LABELS (WARNING ONLY)

13. THIS IS A WARNING ONLY - PROCESSING CONTINUES

***VOL1 LABEL ALREADY EXISTS

14. ANSI STANDARDS PROHIBIT THE RE-WRITING OF VOL1 LABELS

UNRECOGNIZED KEYWORD. STRING (CMDL$A)

15. AN INVALID KEYWORD (STRING) APPEARED ON THE COMMAND LINE

SYSTEM ERRORS

MTN NOT ASSIGNED	USE COMMAND AS MTN BEFORE THE LABEL COMMAND
SUBR EOF	END-OF-FILE ON THE MAGNETIC TAPE
SUBR EOT	END-OF-TAPE
SUBR MTNO	THE TAPE DRIVE IS NOT OPERATIONAL
SUBR PERR	PARITY ERROR ON THE TAPE DRIVE
SUBR HERR	TAPE DRIVE HARDWARE ERROR
SUBR BADC	LABEL IMPROPERLY CALLED MAG TAPE SUBROUTINES

(IN THE ABOVE ERRORS, SUBR IS THE NAME OF THE MAG TAPE SUBROUTINE WHICH REPORTED THE ERROR. SEE PDR-3106 REFERENCE GUIDE SOFTWARE LIBRARY FOR MORE INFORMATION REGARDING THESE ERRORS).

IF LABEL SUCCESSFULY WRITES A LABEL, THE MESSAGE "TAPE LABEL WAS WRITTEN SUCCESSFULLY" IS DISPLAYED. ON READ OPERATIONS, LABEL PRINTS OUT THE VOLUME & OWNER IDS, CREATION DATE, ACCESS (ANSI TAPES ONLY), & OTHER INFORMATION.

THE COMMAND "LABEL -HELP" CAUSES LABEL TO PRINT OUT A DESCRIPTION OF THE COMMAND SIMILAR TO THAT FOUND IN THIS DOCUMENT.

FOR A COMPLETE DESCRIPTION OF TAPE LABELS AND THEIR USE, REFER TO THE IBM PUBLICATION GC28-6680, "OS TAPE LABELS" AND THE ANSI PUBLICATION X3.27-1969, "AMERICAN NATIONAL STANDARD MAGNETIC TAPE LABELS FOR INFORMATION INTERCHANGE".

```
**********************************************************************
**********************************************************************
**
**
**
**      JJJ    III    M   M   M   M   Y   Y
**       J      I     MM MM   MM MM   Y   Y
**       J      I     M M M   M M M    Y Y
**       J      I     M M M   M M M     Y
**    J  J      I     M   M   M   M     Y
**    J  J      I     M   M   M   M     Y
**     JJ      III    M   M   M   M     Y
**
**
**
**    RPPR    L       AAA    TTTTT   EEEEE
**    R   R   L      A   A     T     F
**    R   R   L      A   A     T     E
**    RRRR    L      AAAAA     T     EEEE
**    R R     L      A   A     T     F
**    R  R    L   L  A   A     T     F
**    R   R   LLLLL  A   A     T     EEEEE
**
**
**
**********************************************************************
**********************************************************************
```

REV 17.1 DOCUMENTATION - LATE

TAR 31767 - LATE ACCEPTED UNREASONABLE INPUT (SUCH AS '9999').

THE FOLLOWNG ENHANCEMENTS WERE ALSO MADE:

1. LATE USES SLEEP$ INSTEAD OF RECYCL
2. BETTER ERROR CHECKING OF INPUT
3. DISPLAYS A REV#

```
**********************************************************************************
**********************************************************************************
**
**
**
**        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 M     Y Y
**          J       I     M  M  M    M  M  M      Y
**      J   J       I     M     M    M     M      Y
**      J   J       I     M     M    M     M      Y
**       JJ       III     M     M    M     M      Y
**
**
**
**      RRRR    M     M    AAA      GGG     N     N  EEEEE  TTTTT      1
**      R   R   MM   MM   A   A    G   G    NN    N  E        T       11
**      R   R   M M M M   A   A    G        N N   N  E        T        1
**      RRRR    M  M  M   AAAAA    G        N N   N  EEEE     T        1
**      R  R    M     M   A   A    G  GG    N N   N  E        T        1
**      R   R   M     M   A   A    G   G    N  NN N  E        T        1
**      R    R  M     M   A   A     GGGG    N    N  EEEEE    T       111
**
**
**
**********************************************************************************
**********************************************************************************
```

MAGNET FOR REVS. 17.1, 16.6 AND 15.6

DATE:

TO:

FROM:

SUBJECT: MAGNET FOR REVS. 17.1, 16.6 AND 15.6

REFERENCE:

ABSTRACT

THIS DOCUMENT LISTS THE CHANGES TO THE MAGNETIC TAPE UTILITY PROGRAM, MAGNET, FOR REVS. 17.1, 16.6 AND 15.6.

MAGNET FOR REVS. 17.1, 16.6 AND 15.6

THE FOLLOWING FEATURES HAVE BEEN ADDED TO MAGNET:

1. SUPPORTS UPPER AND LOWER CASE INPUT FROM THE USER'S TERMINAL AND

2. SUPPOPTS TREE-NAMES FOR INPUT AND OUTPUT FILES.

THE FOLLOWING BUGS HAVF BEEN FIXED:

1. RECORD LINE NUMBERS ARE NOW RECORDED AS INTEGER*4 TO PREVENT WRAP-AROUND PROBLEMS IN LARGE FILES.

THF FOLLOWING INTERNAL IMPROVEMENTS HAVE BEEN INSTITUTED:

1. CALLS TO THE OBSOLETE ROUTINE AYENAY HAVE BEEN REPLACED WITH APPLIB CALLS TO YSNO$A,

2. T$MT CALLS NOW WAIT ON THE SEMAPHORE INSTEAD OF THE USER'S TIME-SLICE AND

3. THE INSERT FILE FMTCOM HAS BEEN MERGED WITH THE INSERT FILE MTUCOM.

```
**********************************************************************
**********************************************************************
**
**
**
**      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     Y
**   J   J      I     M    M   M    M     Y
**   J   J      I     M    M   M    M     Y
**    JJ       III    M    M   M    M     Y
**
**
**
**   RRRR    M    M    AAA     GGG    N    N  EEEEE  TTTTT   222
**   R   R   MM  MM   A   A   G   G   NN   N  E        T    2   2
**   R   R   M M M   A   A   G       N N  N  E        T        2
**   RRRR    M  M  M   AAAAA  G       N N  N  EEEE     T      2
**   R R     M     M   A   A   G  GG   N N  N  E        T     2
**   R  R    M     M   A   A   G   G   N  NN  E        T    2
**   R   R   M     M   A   A    GGGG   N    N  EEEEE    T   22222
**
**
**
**********************************************************************
**********************************************************************
```

MAGNET - BUGS FIXED IN 17.1

TAR-22634 - LOWER CASE INPUT NOW ACCEPTED
TAR-15346 - SHORT INTEGER PROBLEM FIXED
TAR-15535 - ODD LENGTH RECORD PROBLEM FIXED

```
**********************************************************************
**********************************************************************
**
**
**
**      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     Y
**   J   J      I     M    M  M    M     Y
**   J   J      I     M    M  M    M     Y
**    JJ       III    M    M  M    M     Y
**
**
**
**   RRRR    M    M   AAA    GGG   RRRR    SSS   TTTTT
**   R   R   MM  MM  A   A  G   G  R   R  S   S    T
**   R   R   M M M M A   A  G      R   R  S        T
**   RRRR    M  M  M AAAAA  G      RRRR    SSS     T
**   R R     M     M A   A  G  GG  R R        S    T
**   R  R    M     M A   A  G   G  R  R   S   S    T
**   R   R   M     M A   A   GGGG  R   R   SSS     T
**
**
**
**********************************************************************
**********************************************************************
```

SUBJECT- MAGSAV AND MAGRST FOR REV. 17

SUBJECT- MAGSAV AND MAGRST FOR REV. 17

THE FOLLOWING BUGS HAVE BEEN FIXED.

1. MAGRST NOW HANDLES THE CONDITION THAT 'A NON DATE RECORD FOLLWOS A UFD TREENAME RECORD'

2. MAGRST WILL PRINT AN ERROR MESSAGE AND PAUSE WHEN A DISK FULL CONDITION OCCURS. (TAR 11969)

3. MAGRST WILL PRINT THE PATHNAME OF THE FILE AT THE TIME A 'UNEXPECTED EOF CONDITION OCCURS.

4. MAGRST WILL NOW SET THE READ/WRITE LOCK CORRECTLY. (TAR 10554)

5. THE -LONG OPTION HAS BEEN REMOVED FROM THE USAGE LINE. (TAR 22800)

THE FOLLOWING BUGS HAVE BEEN FIXED IN MAGSAV

1. MAGSAV WILL SAVE A UFD WHICH HAS READ ONLY PERMISSION TO THE NON-OWNER AND FILES WITHIN THAT UFD WHICH PERMIT READ ACCESS TO THE NON-OWNER. PASSWORDS FOR THE SAVED UFD ARE SET TO NULL.

2. WHEN THE PROGRAM ASKS FOR A NEW TAPE, THE PROGRAM NOW CHECKS TO SEE IF THE NEW TAPE IS AT THE LOAD POINT. IF NOT, AND THE TAPE IS THE SECOND PHYSICAL REEL OF A LOGICAL TAPE, THE PROGRAM WILL QUERY THE USER TO SEE IF HE WANTS THE TAPE TO BE REWOUND. IF THE ANSWER IS 'YES', THE TAPE WILL BE REWOUND. IF THE ANSWER IS 'NO', THE PROGRAM WILL ASK FOR A NEW TAPE UNIT.

```
**********************************************************************************
**********************************************************************************
**
**
**
**      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     Y
**   J   J      I     M     M  M     M    Y
**   J   J      I     M     M  M     M    Y
**    JJ       III    M     M  M     M    Y
**
**
**
**   RRRR   RRRR   U   U  N   N   000   FFFFF  FFFFF    1
**   R   R  R   R  U   U  NN  N  0   0  F      F       11
**   R   R  R   R  U   U  N N N  0   0  F      F        1
**   RRRR   RRRR   U   U  N N N  0   0  FFFF   FFFF     1
**   R R    R R    U   U  N N N  0   0  F      F        1
**   R  R   R  R   U   U  N  NN  0   0  F      F        1
**   R   R  R   R   UUU   N   N   000   F      F      111
**
**
**
**********************************************************************************
**********************************************************************************
```

MODIFICATIONS TO RUNOFF

DATE:

TO:

FROM:

SUBJECT: MODIFICATIONS TO RUNOFF

REFERENCE:

ABSTRACT

THIS DOCUMENT LISTS THE CHANGES THAT HAVE BEEN MADE TO RUNOFF FOR REV 15.6. ALL OF THESE BUG FIXES HAVE BEEN MADE TO RUNOFF REV 16.4 OR 16.6, AND ARE ALSO DOCUMENTED IN EARLIER UPDATES OF THIS PET.

THE FOLLOWING CHANGES HAVE BEEN MADE IN RUNOFF FOR REV 15.6:

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) RUNOFF DID NOT ALLOW REVISION BARS TO BE PRINTED ON BLANK LINES IN THE MIDDLE OF A REVISED BLOCK OF TEXT. BY TURNING THE REVISION BAR ON WITH ".RBAR ALL", BLANK LINES WILL NOW GET THE RBAR.

2) ADJUST MODE DID NOT ALWAYS ADJUST QUITE RIGHT WHEN DEALING WITH UNDERLINED TEXT THAT ALSO INCLUDED PHANTOM HYPHENS. THIS HAS BEEN CORRECTED.

3) USING A PHANTOM HYPHEN IN A DECIMAL HEADING WHEN GENERATING A TABLE OF CONTENTS NO LONGER SAVES A SPACE FOR THE HYPHEN WHEN FILLING THE TABLE OF CONTENTS LINE WITH PERIODS BEFORE THE NUMBER. THIS MEANS THE PAGE NUMBERS WILL BE CORRECTLY LINED UP.

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) WHEN A .SM COMMAND HAPPENED TO COINCIDE WITH A PAGE EJECT OCCURRING BECAUSE OF A PREVIOUS COMMAND, THE NEW SIDE MARGINS DID NOT ACTUALLY TAKE EFFECT UNTIL THE NEXT PAGE. THIS HAS BEEN CORRECTED. OTHER COMMANDS THAT CAUSE A POSSIBLE BREAK AND EJECT TO SET UP A NEW PAGE ENVIRONMENT WILL ALSO NOW TAKE EFFECT ON THE APPROPRIATE PAGE RATHER THAN POSSIBLY WAITING AN EXTRA PAGE. (TAR 24980)

6) 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.

7) 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.

8) RUNOFF ALWAYS USED THE DEFAULT ERASE AND KILL CHARACTERS OR " AND ? EVEN IF THE USER HAD RESET THESE TO BE OTHER CHARACTERS. RUNOFF WILL NOW USE THE USER'S ERASE AND KILL CHARACTERS IN COMMAND MODE. (TARS 20194, 23668)

9) IF A DECIMAL HEADER GOT PUSHED TO THE TOP OF A PAGE BECAUSE OF NOT BEING ALLOWED TO LEAVE WIDOWS ON THE BOTTOM OF THE PREVIOUS PAGE AND A FLOAT FILE HAD BEEN ENTERED SOMETIME PREVIOUSLY, THERE WAS THE POSSIBILITY OF GETTING PART OR ALL OF THE HEADER FOLLOWED BY THE FLOAT FILE BEFORE GETTING THE TEXT THAT BELONGED WITH THE HEADER. THIS HAS BEEN CORRECTED BY CHECKING IF A FLOAT SHOULD START AFTER THE NEED, BEFORE PUTTING THE HEADER IN THE OUTPUT FILE. IF FSTART IS TRUE WE BACK UP THE INPUT FILE ONE LINE, BY THE SUBROUTINE BACKUP, DO THE FLOAT FILE AND COME BACK AND RE-READ THE DECIMAL HEADER COMMAND THAT CAUSED THE PROBLEM AND RE-DO IT. (TAR 23222)

10) .DL DID NOT WORK IF YOU HAPPENED TO DO A DLEVEL TO THE FIRST TIME

AT THAT LEVEL. .DL LEAVES THE USER AT THE TEXT MARGIN FOR THE SPECIFIED LEVEL. THE .DN COMMAND CHECKS IF THIS IS THE FIRST TIME AT THAT LEVEL AND IF IT IS, DOES NOT DO AN UNDENT TO THE HEADING MARGIN. SINCE FUTURE DECIMALIZATION COMMANDS ONLY DO RELATIVE AND NOT ABSOLUTE INDENTS THE MARGINS ARE OFF THEREAFTER BY THE AMOUNT OF THAT LEVEL'S TEXT INDENT. ADDING A FLAG TDL TO SAY WHEN IT WAS A .DL THAT GOT YOU TO THE LEVEL THE UNDENT WILL HAPPEN ANYWAY IF YOU ARE AT THE TOP NUMBER FOR THE LEVEL AND GOT THERE BY A .DL. (TAR 23221)

11) RUNOFF HAS BEEN MODIFIED TO ACCEPT TREENAMES UP TO 80 CHARACTERS IN LENGTH FOR .INSERT FILES. (TAR 12603)

12) WHEN PROCESSING THE .//// COMMAND FOR APPORTIONING TEXT ACROSS A LINE, RUNOFF OCCASIONALLY BLANKED OUT ANYTHING THAT MIGHT HAVE BEEN IN THE FIRST COLUMN IF THE APPORTIONED TEXT BELONGED IN A LATER COLUMN WHEN WORKING WITH MULTIPLE COLUMNS. (TAR 22802)

13) USING .+ OR .> WITHOUT FOLLOWING TEXT TO CREATE A BLANK LINE CAUSED RUNOFF TO PRINT TWO LINE FEEDS WHEN SENDING OUTPUT TO THE TERMINAL. BECAUSE THE TWO NEWLINES WERE PUT IN THE SAME WORD, THIS DID NOT SHOW UP WHEN THE OUTPUT WAS SENT TO A FILE AND SPOOLED. THE EXTRA LINES WERE NOT COUNTED AS FAR AS RUNOFF PAGES WERE CONCERNED EITHER WHICH CAUSED THOSE PAGES TO APPEAR LONGER THAN THEY ACTUALLY WERE. (TAR 80967)

14) THE BLANK CHARACTER DID NOT GET TRANSLATED INTO A SPACE WHEN IT WAS USED IN HEADERS AND FOOTERS.

15) DEFINED SYMBOLS COULD HAVE AT MOST 30 CHARACTERS BUT THE LOOP TO TRANSLATE THEM WENT UP TO 60 SO IF THE DEFINITION WAS LONGER THAN 30 THERE WAS GARBAGE ON THE END.

A RESTRICTION THAT RUNOFF USERS SHOULD BE AWARE OF IS, WHEN USING THE TWO COMMANDS .BREAK AND .INDENT N IN CONJUNCTION THE .BREAK SHOULD PRECEED THE .INDENT OR IT MAY NOT WORK AS EXPECTED.

```
**********************************************************************
**********************************************************************
**
**
**
**      JJJ    III    M    M   M    M   Y    Y
**       J      I     MM  MM   MM  MM   Y    Y
**       J      I     M M M    M M M     Y Y
**       J      I     M   M    M   M      Y
**   J   J      I     M   M    M   M      Y
**   J   J      I     M   M    M   M      Y
**    JJ       III    M   M    M   M      Y
**
**
**
**   DDDD   BBBB    GGG    III   N   N  FFFFF   000          RRRR    000   FFFFF
**   D   D  B   B  G   G    I    NN  N  F      0   0         R   R  0   0  F
**   D   D  B   B  G        I    N N N  F      0   0         R   R  0   0  F
**   D   D  BBBB   G        I    N N N  FFFF   0   0         RRRR   0   0  FFFF
**   D   D  B   B  G  GG     I    N N N  F      0   0         R R    0   0  F
**   D   D  B   B  G   G    I    N  NN  F      0   0   ..    R  R   0   0  F
**   DDDD   BBBB    GGGG   III   N   N  F       000    ..    R   R   000   F
**
**
**
**********************************************************************
**********************************************************************
```

HIGHER-LEVEL-LANGUAGE DEBUGGER (DBG)

DATE:

TO:

FROM:

SUBJECT: HIGHER-LEVEL-LANGUAGE DEBUGGER (DBG)

REFERENCE:

1 SCOPE

THIS DOCUMENT DESCRIBES THE USER-VISIBLE FEATURES OF PRIME'S HIGHER-LEVEL-LANGUAGE DEBUGGER (DBG). IT SUPERSEDES PRIME ENGINEERING DOCUMENT PE-TN-198 REVISION 1 ("DBG FUNCTIONAL SPECIFICATION") AS THE AUTHORITY ON THE DBG USER INTERFACE.

2 ABSTRACT

DBG IS AN ADVANCED, MULTILINGUAL SOFTWARE DEBUGGING TOOL CAPABLE OF INTERACTING WITH PROGRAMS WRITTEN IN FORTRAN (-74 AND -77), PL/P, PL/1 (SUBSET G AND FULL), AND LATER, COBOL.

DBG PROVIDES THE USER THE CAPABILITY OF CONTROLLING PROGRAM EXECUTION ON A REAL-TIME, INTERACTIVE BASIS ON A LEVEL WHICH REQUIRES KNOWLEDGE OF ONLY THE HIGHER LEVEL SOURCE LANGUAGE (AND BASIC DEBUGGER COMMANDS); HE NEED NOT BE FAMILIAR WITH THE MACHINE ARCHITECTURE TO INTERACTIVELY DEBUG HIS PROGRAM.

TABLE OF CONTENTS

CHAPTER I

CHAPTER II

CHAPTER III

CHAPTER IV

CHAPTER V

3 ABOUT THIS DOCUMENT

3.1 DOCUMENT FORMAT

THIS DOCUMENT IS DIVIDED INTO FIVE CHAPTERS.

CHAPTER I	DISCUSSES THE PRODUCT BACKGROUND AND PROVIDES AN OVERVIEW OF THE FUNCTIONALITY OFFERED BY DBG.
CHAPTER II	PRESENTS AN INTRODUCTION TO DEBUGGER USAGE. INCLUDED IS A DESCRIPTION OF THE BASIC DEBUGGER COMMANDS AND EXAMPLES OF THEIR USAGE, WITH REFERENCES INTO CHAPTER III FOR COMPLETE INFORMATION. A SAMPLE DEBUGGING SESSION IS INCLUDED AT THE END OF THIS CHAPTER.
CHAPTER III	IS THE FUNCTIONAL DESCRIPTION OF THE DEBUGGER. IT CONTAINS A FULL DESCRIPTION OF EACH DEBUGGER COMMAND, DBG INVOCATION AND TERMINATION, AND USAGE. COMMANDS ARE DESCRIBED BY CLASS. THE BREAKPOINT FACILITY AND ALL COMMANDS RELATED TO BREAKPOINTS, FOR EXAMPLE, ARE DESCRIBED IN SECTION 11.
CHAPTER IV	SUMMARIZES, IN TABULAR FORM, ALL OF THE DBG COMMAND AND USAGE INFORMATION PRESENTED IN THIS DOCUMENT.
CHAPTER V	PROVIDES SOME HINTS AND SUGGESTIONS FOR FINDING PROGRAM BUGS USING DBG.

3.2 TIPS TO THE READER

IT IS SUGGESTED THAT THE FIRST-TIME OR CASUAL READER START WITH CHAPTER I AND CONTINUE INTO CHAPTER II FOR AN OVERVIEW OF DBG FUNCTIONALITY AND USAGE.

THE NOVICE DBG USER SHOULD START IN CHAPTER II FOR AN INTRODUCTION TO DEBUGGER COMMANDS AND REFER TO CHAPTER III AS REQUIRED FOR INFORMATION ON ADVANCED COMMANDS AND OPTIONS.

AN EXPERIENCED DBG USER IN NEED OF A REFERENCE MIGHT USE THE TABULAR REFFERENCE INFORMATION CONTAINED WITHIN CHAPTER IV AND REFER TO CHAPTER III WHEN MORE COMPLETE DESCRIPTIONS ARE REQUIRED.

THE USER AT WIT'S END CONFRONTED WITH A PROGRAM BUG MIGHT FIND THE INFORMATION IN CHAPTER V USEFUL.

4 MOTIVATION

IN THE PAST, THE PROGRAMMER WHO CODED IN HIGHER LEVEL LANGUAGES COULD CHOOSE FROM TWO METHODS OF DEBUGGING HIS PROGRAM:

(1) IF HE WAS FAMILIAR WITH THE MACHINE ARCHITECTURE, HE COULD USE ONE OF THE MACHINE LEVEL DEBUGGING FACILITIES (TAP, PSD, OR VPSD). THE DRAWBACKS TO THIS METHOD ARE MANY; MOST NOTABLY, THE USER FOUND HIMSELF CONSTANTLY REFERRING TO LOAD MAPS AND COMPILER GENERATED CONCORDANCE LISTINGS, MANUALLY CALCULATING STATEMENT AND DATA ADDRESSES, WALKING BACK THE PROCEDURE STACK, ETC. DATA WAS MOST COMMONLY DISPLAYED IN OCTAL OR DECIMAL REPRESENTIONS OF 16 BIT WORDS, REQUIRING THE USER TO PERFORM A CONVERSION IF HE WANTED TO MANIPULATE A DATUM IN A REPRESENTATION FOREIGN TO THE DEBUGGER (PSD OR VPSD). EXPRESSION EVALUATION (AND EVEN INTERPRETATION OF FLOATING POINT CONSTANTS) WAS UNAVAILABLE.

THIS DEBUGGING TECHNIQUE PROVIDED YET ANOTHER SUBSTANTIATION THAT MACHINE LEVEL DEBUGGERS DO NOT LEND THEMSELVES WELL AS TOOLS FOR DEBUGGING PROGRAMS WRITTEN IN A HIGHER LEVEL LANGUAGE.

(2) FOR THE USER WHO WAS NOT INTIMATE WITH THE MACHINE ARCHITECTURE, STATEMENTS TO PRINT CHECKPOINT MESSAGES AND VALUES OF VARIABLES AT CERTAIN KEY POINTS COULD BE INSERTED INTO THE SOURCE PROGRAM, THE PROGRAM RECOMPILED, RELOADED, AND EXECUTED. THIS METHOD ALSO HAS ITS DISADVANTAGES, TENDING TO PRODUCE VOLUMINOUS OUTPUT AND REQUIRING FREQUENT RECOMPILATIONS; INDEED, THIS METHOD IS TRULY "BATCH MODE" PROGRAM DEVELOPMENT.

OBVIOUSLY, NEITHER TECHNIQUE DESCRIBED ABOVE IS PARTICULARLY DESIRABLE TO THE HIGHER LEVEL LANGUAGE PROGRAMMER CODING IN COBOL OR FORTRAN.

AS PRIME MOVED TOWARD THE LESS SOPHISTICATED END-USER MARKET, THE MESSAGE BECAME CLEAR THAT AN INTERACTIVE DEBUGGING TOOL COUCHED IN TERMS OF A HIGHER LEVEL LANGUAGE WAS REQUIRED TO ROUND OUT THE PROGRAM DEVELOPMENT ENVIRONMENT CURRENTLY PROVIDED.

5 FUNCTIONAL OVERVIEW

THIS SECTION SERVES AS AN OVERVIEW OF THE DEBUGGING ENVIRONMENT AND FEATURES PROVIDED BY DBG.

CENTRAL PROCESSOR:

THE DEBUGGER RUNS ON ANY CPU CAPABLE OF SUPPORTING 64V ADDRESSING MODE. PRESENTLY, THIS INCLUDES THE PRIME 350, 400, 450, 500, 550, 650, AND 750 PROCESSORS.

REQUIRED SOFTWARE:

IN ADDITION TO THE DEBUGGER ITSELF, THE PRIME-SUPPLIED SOFTWARE WHICH COMPRISES A PROGRAM DEVELOPMENT ENVIRONMENT SHOULD BE AVAILABLE. THIS INCLUDES THE PRIMOS OPERATING SYSTEM, COMPILERS, LOADERS, AND LIBRARIES, AS WELL AS PROGRAM PREPARATION UTILITIES (SUCH AS ED).

ADDRESSING MODES:

THE DEBUGGER OPERATES ON PROGRAMS WHICH EXECUTE IN EITHER 64V OR 32I MODES. NO S OR R MODE SUPPORT IS AVAILABLE OR PLANNED. THE DEBUGGER ITSELF EXECUTES IN 64V MODE.

LANGUAGES SUPPORTED:

PRESENTLY, LANGUAGES SUPPORTED INCLUDE FORTRAN-74, FORTRAN-77, AND PL/P. PL/1 (SUBSET G AND FULL) DEBUGGER SUPPORT WILL BE AVAILABLE COINCIDENTALLY WITH THE COMPILERS. COBOL SUPPORT IS PLANNED.

MEMORY REQUIREMENTS:

THE DEBUGGER PROCEDURE TEXT (WHICH IS SHARED) OCCUPIES THREE SEGMENTS. PER USER INFORMATION WHICH REQUIRES A FIXED AMOUNT OF SPACE INCLUDES COMMON AREA AND LINKAGE TEXT. THIS OCCUPIES ABOUT 48K WORDS. PER USER SPACE OF VARIABLE LENGTH INCLUDES STACK SPACE (AT LEAST 16K WORDS) AND SYMBOL TABLE SPACE. THE SYMBOL TABLE SIZE IS DIRECTLY DEPENDENT UPON THE NUMBER OF SYMBOLS, UNIQUE NAMES, AND PROGRAM BLOCKS DEFINED WITHIN THE PROGRAM. THIS MAY BE DETERMINED ONLINE (FROM WITHIN DBG) FOR ANY PROGRAM; SEE "DEBUGGER-DEFINED VARIABLES", DESCRIBED IN SECTION 12.1.4. ALL SYMBOL TABLE STORAGE IS ALLOCATED DYNAMICALLY.

PERFORMANCE:

PERFORMANCE WILL PRIMARILY BE LIMITED BY THE SPEED AT WHICH THE USER ENTERS COMMANDS OR EXPRESSIONS TO BE EVALUATED. THE TIME TO EXECUTE ALL COMMANDS SHOULD BE LESS THAN 1/2 CPU SECOND. THE TIME TO EVALUATE AN EXPRESSION SHOULD BE ABOUT 2 TIMES THE AMOUNT OF TIME SPENT COMPILING THAT EXPRESSION.

GENERALLY SPEAKING, WHEN USING THE DEBUGGER, THE USER IS MOST CONCERNED WITH RESPONSE TIME; RESPONSE TIME IS MORE DEPENDENT UPON SYSTEM LOAD CAUSED BY OTHER PROCESSES THAN BY ACTUAL DEBUGGER PERFORMANCE.

WORKING SET:

THE AUTHORS FEEL THAT IT IS IMPRACTICAL TO ATTEMPT TO QUANTIFY THE WORKING SET FOR THE DEBUGGER.

EACH DBG COMMAND REQUIRES DIFFERENT INFORMATION TO BE AVAILABLE IN ORDER FOR THE COMMAND TO BE EXECUTED; SOME REQUIRE ONLY A SMALL NUMBER OF DBG-OWNED PAGES TO BE REFERENCED, WHILE OTHERS, SPECIFICALLY THOSE WHICH REQUIRE THE USE OF THE SYMBOL TABLE, COULD REFERENCE A VERY LARGE AMOUNT OF MEMORY (OVER 50 PAGES IN MANY CASES). FURTHERMORE, UNLESS COMMANDS ARE ENTERED AT THE SPEED AT WHICH THEY ARE EXECUTED (WHICH IS MOST UNLIKELY), ALL MEMORY RESIDENT PAGES WILL BE PAGED OUT GIVEN A SUFFICIENT PAGE DEMAND FROM PROCESSES EXTERNAL TO THE DBG USER. (THE LAST STATEMENT APPLIES TO ANY INTERACTIVE USER).

IT IS FELT THAT THE ISSUES OF WORKING SET SIZE AND PERFORMANCE ARE RELATIVELY UNIMPORTANT FOR THE DEBUGGER. THE DBG USER SHOULD BE MORE THAN WILLING TO EXPEND MACHINE TIME AND MEMORY SPACE IN EXCHANGE FOR SAVINGS OF PROGRAMMER "THINK" TIME.

FEATURES:

PROGRAM CONTROL

- PROGRAM RESTART: AT ANY TIME, THE USER MAY RESTART THE PROGRAM BEING DEBUGGED
- BREAKPOINTING: THE USER MAY REQUEST THAT EXECUTION BE SUSPENDED AT A SPECIFIED STATEMENT IN, ENTRY TO, OR EXIT FROM A PROCEDURE
- ACTION LIST ON BREAKPOINT: THE USER MAY SPECIFY ONE OR MORE DEBUGGER COMMANDS TO BE EXECUTED EACH TIME A BREAKPOINT OCCURS
- CONDITIONAL BREAKPOINTING: THE USER MAY SPECIFY THAT A BREAKPOINT IS TO TAKE PLACE ONLY IF A GIVEN EXPRESSION YIELDS A "TRUE" RESULT (VARIATION ON BREAKPOINT ACTION LIST, ABOVE)

- SINGLE STEPPING: THE DEBUGGER OFFERS A COMPREHENSIVE SET OF COMMANDS TO PERMIT SINGLE STEPPING ACROSS STATEMENTS, INTO AND OUT OF CALLED PROCEDURES, ETC

- TRANSFER OF CONTROL: THE USER MAY REQUEST THAT CONTROL BE TRANSFERRED TO A SPECIFIED STATEMENT WHEN PROGRAM EXECUTION IS RESUMED

- CALL SUBROUTINE OR FUNCTION: THE USER MAY, FROM DEBUGGER COMMAND LEVEL, CALL A SUBROUTINE OR FUNCTION (OPTIONALLY SUPPLYING AN ARGUMENT LIST)

DATA MANIPULATION

- VALUE OF VARIABLE: THE DEBUGGER CAN PRINT THE VALUE OF ANY SCALAR, ARRAY, OR STRUCTURE, AS WELL AS AN ARRAY CROSS-SECTION

- EXPRESSION EVALUATION: ANY EXPRESSION ALLOWED BY THE SOURCE LANGUAGE MAY ALSO BE EVALUATED BY THE DEBUGGER

- ASSIGNMENT: THE USER MAY MODIFY THE VALUE OF A VARIABLE

- EXAMINE EXPRESSION TYPE: THE USER HAS THE CAPABILITY TO EXAMINE THE RESULTANT TYPE OF AN EXPRESSION (OR SIMPLE VARIABLE)

- BUILTIN (INTRINSIC) FUNCTIONS: ALL BUILTIN FUNCTIONS AVAILABLE IN BOTH FORTRAN AND PL/1 ARE INTERNAL TO THE DEBUGGER AND MAY BE REFERENCED IN AN EXPRESSION

TRACING

- TRACEPOINTING: THE USER MAY REQUEST THAT A TRACE MESSAGE BE OUTPUT AT A SPECIFIED STATEMENT IN, ENTRY TO, OR EXIT FROM A PROCEDURE

- STATEMENT TRACING: THE USER MAY SPECIFY THAT A TRACE MESSAGE IS TO BE OUTPUT PRIOR TO THE EXECUTION OF EACH STATEMENT OR EACH LABELLED STATEMENT

- ENTRY/EXIT TRACING: THE USER MAY ALSO SPECIFY THAT A MESSAGE IS TO BE PRINTED EACH TIME ANY PROCEDURE IS CALLED OR RETURNS

- VALUE TRACING: THE USER MAY SPECIFY THAT A GIVEN VARIABLE (OR VARIABLES) IS TO BE "WATCHED" AND A MESSAGE PRINTED ANY TIME THE VALUE OF THE VARIABLE IS CHANGED

- TRACEBACK: THE DEBUGGER HAS THE CAPABILITY TO PRINT A PROCEDURE CALL / RETURN STACK HISTORY

MISCELLANEOUS

- SOURCE FILE EXAMINATION: THE USER MAY INSPECT (BUT NOT CHANGE) SOURCE FILES WHILE EXECUTING WITHIN THE DEBUGGER - USING A SUBSET OF ED COMMANDS

- MACHINE LEVEL DEBUGGING: THE USER MAY INVOKE A BUILTIN VERSION OF VPSD TO DEBUG ASSEMBLY LANGUAGE CODE

6 INTRODUCTION TO DBG USAGE

THIS SECTION PRESENTS AN ELEMENTARY INTRODUCTION TO PROGRAM DEBUGGING USING DBG. IT DESCRIBES PROGRAM COMPILATION, LOADING, PROGRAM CONTROL, AND THE BASIC DEBUGGER COMMANDS WHICH MIGHT BE USED DURING A DEBUGGING SESSION. A SAMPLE DEBUGGING SESSION IS INCLUDED AT THE END OF THIS SECTION.

KEEPING IN MIND THAT THIS SECTION IS INTRODUCTORY, ONLY THE MOST BASIC COMMAND FORMATS AND DESCRIPTIONS ARE GIVEN. REFERENCES TO OTHER SECTIONS ARE PROVIDED WHERE APPROPRIATE, SHOULD THE READER DESIRE FURTHER INFORMATION ON A PARTICULAR TOPIC.

6.1 PROGRAM COMPILATION

THE POTENTIAL DEBUGGER USER MUST INFORM THE COMPILER THAT HE LATER INTENDS TO USE DBG. HE DOES SO BY INCLUDING THE "-DEBUG" COMPILE-TIME OPTION ON THE COMMAND LINE.

FOR EXAMPLE, TO COMPILE "MYPROGRAM" WITH THE FORTRAN COMPILER FOR LATER USE WITH DBG, ONE MIGHT ENTER:

```
OK, FTN MYPROGRAM -64V -DEBUG
```

INCLUSION OF THE "-DEBUG" OPTION CAUSES THE COMPILER TO OUTPUT THE INFORMATION NECESSARY FOR THE DEBUGGER TO RECOGNIZE AND MANIPULATE PROGRAM UNITS, SYMBOLS, AND STATEMENTS.

THE MEANING OF "-DEBUG" AND OTHER DBG-RELATED COMPILE-TIME OPTIONS IS DISCUSSED IN SECTION 23.

6.2 PROGRAM LOADING

PROGRAMS WHICH ARE COMPILED WITH THE "-DEBUG" OPTION ARE LOADED IN THE SAME WAY AS THOSE WHICH ARE NOT; IN OTHER WORDS, THE USER EXPERIENCES NO CHANGE IN THE WAY A PROGRAM IS LOADED.

TO LOAD THE PROGRAM "COMPILED" IN THE EXAMPLE ABOVE, ONE WOULD ENTER:

```
OK, SEG
# LOAD #MYPROGRAM
$ LOAD B MYPROGRAM
$ LIBRARY
LOAD COMPLETE
$ SAVE
$ QUIT

OK,
```

6.3 INVOKING AND TERMINATING DBG

THE DEBUGGER IS INVOKED FOLLOWING AN "OK" PROMPT ISSUED BY PRIMOS BY ENTERING THE "DBG" COMMAND FOLLOWED BY THE NAME OF THE SEG FILE CONTAINING THE PROGRAM TO BE DEBUGGED.

FOR EXAMPLE, TO DEBUG "#MYPROGRAM", THE FOLLOWING PRIMOS COMMAND IS ENTERED:

```
OK, DBG #MYPROGRAM

**DBG**   REVISION 17.0A  (06-FEBRUARY-1979)

>
```

WHEN THE DEBUGGER IS ENTERED, IT READS THE PROGRAM AND SYMBOL TABLE FROM THE SEG FILE INTO MEMORY AND PRINTS AN IDENTIFICATION MESSAGE, AS SHOWN ABOVE. THE RIGHT ANGLE BRACKET IS THE DEBUGGER'S PROMPT CHARACTER. WHEN IT APPEARS AT THE LEFT MARGIN OF THE USER'S TERMINAL, THE DEBUGGER IS AWAITING COMMAND INPUT.

THE USER MAY TERMINATE THE DEBUGGING SESSION AND RETURN TO PRIMOS COMMAND LEVEL USING THE "QUIT" COMMAND. FOR EXAMPLE,

```
> QUIT

OK,
```

6.4 USER PROGRAM CONTROL

A SESSION WITH DBG ALTERNATES BETWEEN PERIODS OF USER PROGRAM EXECUTION AND DEBUGGER COMMAND EXECUTION.

CONTROL IS INITIALLY PASSED TO DBG FROM PRIMOS WHEN THE DEBUGGER IS INVOKED. CONTROL PASSES FROM DBG TO THE USER PROGRAM WHEN

- THE USER GIVES A RESTART OR CONTINUE COMMAND TO RESTART OR CONTINUE PROGRAM EXECUTION,
- THE USER GIVES ONE OF THE SINGLE-STEP COMMANDS STEP, STEPIN, IN, OR OUT, OR
- THE USER CALLS A SUBROUTINE CONTAINED WITHIN THE USER PROGRAM, OR WHEN THE EVALUATION OF AN EXPRESSION INVOLVES A USER-DEFINED FUNCTION.

CONTROL RETURNS TO DBG WHEN

- THE USER PROGRAM ENCOUNTERS A BREAKPOINT PREVIOUSLY SET BY THE USER,
- THE PROGRAM COMPLETES EXECUTION OF THE NUMBER OF STATEMENTS IMPLIED OR EXPRESSED IN A SINGLE-STEP COMMAND,
- THE MAIN PROGRAM RETURNS, OR ANY PROGRAM UNIT STOPS, PAUSES, CALLS "EXIT", OR CALLS "ERRPR$" TO RETURN TO PRIMOS COMMAND LEVEL,
- IN ENTRY TRACE MODE, WHENEVER A PROCEDURE IS CALLED OR RETURNS,
- IN STATEMENT AND/OR VALUE TRACE MODES, WHENEVER A PROCEDURE IS CALLED OR RETURNS, AND PRIOR TO THE EXECUTION OF EACH STATEMENT,
- A USER SUBROUTINE OR FUNCTION RETURNS FROM A CALL MADE FROM DBG ON BEHALF OF THE USER,
- THE USER PROGRAM RAISES OR CAUSES TO BE RAISED A CONDITION FOR WHICH IT HAS DECLARED NO ON-UNIT (THIS INCLUDES PROGRAM ERRORS, SUCH AS POINTER FAULT, ACCESS VIOLATION, DIVIDE BY ZERO, ETC.), OR
- WHEN THE USER DEPRESSES THE "QUIT" (BREAK OR CONTROL-P) KEY AT HIS TERMINAL, PROVIDED THE USER PROGRAM HAS NO HANDLER FOR THE "QUIT$" CONDITION.

6.5 BASIC DBG COMMANDS

THIS SECTION DISCUSSES THE MINIMAL DBG COMMANDS REQUIRED TO DEBUG A PROGRAM.

6.5.1 "RESTART" - BEGIN PROGRAM EXECUTION

THE "RESTART" COMMAND CAUSES USER PROGRAM EXECUTION TO BE (RE-)STARTED. SEE SECTION 10.1 FOR A FULL DESCRIPTION. EXAMPLE:

```
> RESTART
```

6.5.2 "CONTINUE" - CONTINUE PROGRAM EXECUTION

THE "CONTINUE" COMMAND CAUSES PROGRAM EXECUTION TO CONTINUE FROM THE LAST BREAKPOINT, SINGLE-STEP OPERATION, OR CONDITION SIGNAL. SEE SECTION 10.2 FOR A FULL DESCRIPTION. EXAMPLE:

```
> CONTINUE
```

6.5.3 "BREAKPOINT" - SET A BREAKPOINT

THE "BREAKPOINT" COMMAND CAUSES A BREAKPOINT TO BE PLACED AT THE STATEMENT ON THE SPECIFIED LINE NUMBER. THE BREAKPOINT CAUSES CONTROL TO RETURN TO THE DEBUGGER PRIOR TO THE EXECUTION OF THE STATEMENT. OPTIONALLY, THE USER MAY CAUSE THE BREAKPOINT TRAP TO OCCUR CONDITIONALLY AND/OR SPECIFY A LIST OF DEBUGGER COMMANDS WHICH WILL AUTOMATICALLY BE EXECUTED WHENEVER THE BREAKPOINT TRAP OCCURS. SEE SECTION 11 FOR A FULL DESCRIPTION OF BREAKPOINTS AND RELATED COMMANDS. EXAMPLES:

```
> BREAKPOINT 30              (SET BREAKPOINT AT STMT ON LINE 30)

> BREAKPOINT SUBR\30         (BREAK AT LINE 30 IN "SUBR")
```

6.5.4 "CLEAR" - CLEAR A BREAKPOINT

THE "CLEAR" COMMAND CLEARS A BREAKPOINT. ITS FORMAT IS SIMILAR TO "BREAKPOINT" COMMAND. SEE SECTION 11.6 FOR A FULL DESCRIPTION. EXAMPLE:

```
> CLEAR 12                   (CLEAR BREAKPOINT ON LINE 12)
```

6.5.5 "CLEARALL" - CLEAR ALL BREAKPOINTS

THE "CLEARALL" COMMAND CAUSES ALL BREAKPOINTS WHICH HAVE BEEN SET TO BE CLEARED. THIS MAY OPTIONALLY BE LOCALIZED TO ONE PROGRAM UNIT. SEE SECTION 11.7 FOR A FULL DESCRIPTION. EXAMPLE:

```
> CLEARALL
```

6.5.6 "LISTALL" - LIST ALL BREAKPOINTS

THE "LISTALL" COMMAND CAUSES THE LOCATIONS OF ALL BREAKPOINTS TO BE PRINTED ON THE USER'S TERMINAL. LIKE THE "CLEARALL" COMMAND, THE BREAKPOINT LIST MAY OPTIONALLY BE LOCALIZED TO ONE PROGRAM UNIT. SEE SECTION 11.9 FOR A COMPLETE DESCRIPTION. EXAMPLE:

```
> LISTALL
TYPE  LOCATION
BRK   $MAIN\23, COUNT = 1
BRK   $MAIN\101, COUNT = 0
```

6.5.7 ": " - EVALUATE AN EXPRESSION

THE ": " (COLON, SPACE) COMMAND CAUSES DBG TO EVALUATE THE EXPRESSION (WHICH COULD BE A SIMPLE VARIABLE NAME) WHICH FOLLOWS IT AND TO PRINT THE RESULT. OPTIONALLY, THE USER MAY SELECT THE LANGUAGE EVALUATOR (PL/1, FORTRAN, COBOL) AND THE FORMAT IN WHICH THE RESULT IS PRINTED (DECIMAL, OCTAL, ASCII, FLOATING, ETC.). SEE SECTION 12 FOR COMPLETE INFORMATION. EXAMPLE:

```
> : ALPHA
ALPHA = 120.01
```

6.5.8 "LET" - CHANGE THE VALUE OF A VARIABLE

THE "LET" COMMAND MODIFIES THE VALUE OF A VARIABLE. SEE SECTION 12.5 FOR DETAILS. EXAMPLE:

```
> LET I = 1
```

6.5.9 "WHERE" - PRINT CURRENT LOCATION

THE "WHERE" COMMAND CAUSES THE LOCATION OF THE CURRENT BREAKPOINT TO BE PRINTED AT THE USER'S TERMINAL. SEE SECTION 14.1 FOR COMPLETE INFORMATION. EXAMPLE:

```
> WHERE
CURRENTLY AT CDEL\120.
```

6.5.10 "ARGUMENTS" - PRINT ARGUMENTS TO CURRENT ROUTINE

THE "ARGUMENTS" COMMAND CAUSES ALL ARGUMENTS TO THE CURRENT PROGRAM UNIT (SUBROUTINE, FUNCTION, PROCEDURE, ETC.) TO BE DISPLAYED ON THE USER'S TERMINAL. REFER TO SECTION 12.6 FOR DETAILS. EXAMPLE:

```
> ARGUMENTS
VAL1 = 120
ARY(1) = 1
ARY(2) = 0
ARY(3) = 0
ARY(4) = 0
FLAG = .TRUE.
```

6.5.11 "TRACEBACK" - PRINT STACK TRACE

THE "TRACEBACK" COMMAND CAUSES THE DEBUGGER TO PRINT THE CONTENTS OF THE PROCEDURE CALL / RETURN STACK IN READABLE FORMAT ON THE USER'S TERMINAL. SEE SECTION 15 FOR A FULL DESCRIPTION. EXAMPLE:

```
> TRACEBACK
CURRENTLY AT RECDEL\49.
STACK CONTAINS 3 FRAMES.

  3: OWNER IS "RECDEL".
     CALLED FROM $MAIN\291, RETURNS TO $MAIN\292.

  2: OWNER IS "$MAIN".
     CALLED FROM DEBUGGER, RETURNS TO DEBUGGER.

  1: DEBUGGER-OWNED FRAME.
```

6.5.12 "GOTO" - TRANSFER PROGRAM CONTROL

THE "GOTO" COMMAND ALLOWS THE USER TO CHANGE THE LOCATION AT WHICH PROGRAM EXECUTION IS TO BE RESUMED UPON ISSUANCE OF A "CONTINUE" OR SINGLE-STEP COMMAND. REFER TO SECTION 10.3 FOR COMPLETE DETAILS. EXAMPLE:

```
> GOTO 120                    (RESUME AT STATEMENT ON LINE 120)
```

6.5.13 "TYPE" - PRINT TYPE OF VARIABLE OR EXPRESSION

THE "TYPE" COMMAND CAUSES DBG TO EVALUATE THE EXPRESSION (OR SIMPLE VARIABLE NAME) WHICH FOLLOWS IT AND PRINT THE DATA TYPE, STORAGE CLASS (IF KNOWN), AND ARRAY BOUNDS. SEE SECTION 12.4 FOR A FULL DESCRIPTION. EXAMPLE:

```
> TYPE A
REAL*4 STATIC
```

6.5.14 "SOURCE" - INSPECT SOURCE FILE

THE "SOURCE" COMMAND ALLOWS THE USER TO PERUSE A SOURCE FILE. THE COMMAND MAY BE FOLLOWED BY ANY EDITOR ("ED") COMMAND WHICH DOES NOT MODIFY THE FILE. THE "SOURCE" COMMAND IS COMPLETELY DESCRIBED IN SECTION 22. EXAMPLE:

```
> SOURCE FIND #READ (; PRINT 10
```

6.5.15 "QUIT" - EXIT DBG

THE "QUIT" COMMAND CAUSES THE DEBUGGER TO EXIT TO PRIMOS COMMAND LEVEL. REFER TO SECTION 7 FOR DETAILS. EXAMPLE:

```
> QUIT
```

6.6 SAMPLE PROGRAM DEBUGGING SESSION

CONSIDER THE FOLLOWING (UNDEBUGGED) FORTRAN PROGRAM, WHICH IS USED AS AN EXAMPLE IN THE TEXT BELOW:

```
(0001) C    PRINT THE SQUARES OF THE NUMBERS 1 THROUGH 10.
(0002) C
(0003)          DO 100 I = 1, 10
(0004)          J = SQUARE (I)
(0005)          WRITE (1, 80) J
(0006) 80       FORMAT (I5)
(0007) 100      CONTINUE
(0008)          CALL EXIT
(0009)          END
(0010)
(0011)          INTEGER FUNCTION SQUARE (I)
(0012)          SQUARE = I ** 2
(0013)          RETURN
(0014)          END
```

ASSUMING THAT THIS PROGRAM RESIDES IN A FILE CALLED "SQUARES", IT WOULD BE COMPILED WITH THE FOLLOWING PRIMOS COMMAND:

```
OK, FTN SQUARES -64V -DEBUG
0000 ERRORS [<.MAIN.>FTN-REV17.0F]
0000 ERRORS [<SQUARE>FTN-REV17.0F]

OK,
```

THE "-DEBUG" ARGUMENT INFORMS THE FORTRAN COMPILER THAT THE PROGRAM TO BE COMPILED WILL LATER BE DEBUGGED USING DBG.

THE SEQUENCE OF COMMANDS USED TO LOAD THE PROGRAM IS IDENTICAL TO THAT USED WITHOUT DBG, NAMELY:

```
OK, SEG
# LOAD #SQUARES
$ LOAD B SQUARES
$ LIBRARY
LOAD COMPLETE
$ SAVE
$ QUIT

OK,
```

THERE NOW EXISTS A SEG RUN-FILE, "#SQUARES", CONTAINING A PROGRAM WHICH HAS BEEN COMPILED AND LOADED WITHOUT ERRORS. THE FIRST ATTEMPT AT EXECUTION FAILS, PRODUCING AN OUTPUT WHICH LOOKS LIKE THIS:

```
OK, SEG #SQUARES
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0

OK,
```

THE USER NOW ENTERS THE DEBUGGER USING THE FOLLOWING PRIMOS COMMAND:

```
OK, DBG #SQUARES

**DBG**   REVISION 17.0A  (06-FEBRUARY-1979)

>
```

THE USER DECIDES TO PLACE A BREAKPOINT FOLLOWING THE CALL TO FUNCTION "SQUARE" IN ORDER TO LOOK AT THE ARGUMENT AND RETURNED VALUE. HE ENTERS:

```
> BREAKPOINT 5
>
```

THIS BREAKPOINT COMMAND WILL CAUSE THE DEBUGGER TO REGAIN CONTROL IMMEDIATELY PRIOR TO THE EXECUTION OF THE STATEMENT ON SOURCE LINE 5, THE FORTRAN "WRITE" STATEMENT.

THE USER THEN BEGINS PROGRAM EXECUTION WITH THE DEBUGGER'S "RESTART"

COMMAND. THIS IS FOLLOWED BY THE OCCURRENCE OF THE BREAKPOINT. THE INTERACTION LOOKS LIKE THIS:

```
> RESTART

**** BREAKPOINTED AT $MAIN\5
>
```

THE BREAKPOINT MESSAGE SAYS, EFFECTIVELY, "I HAVE ENCOUNTERED THE BREAKPOINT WHICH YOU SET ON SOURCE STATEMENT NUMBER 5 IN THE FORTRAN MAIN PROGRAM." THE PROMPT CHARACTER INDICATES THAT THE DEBUGGER IS ONCE AGAIN AT COMMAND LEVEL AWAITING A COMMAND.

THE USER MAY NOW INSPECT THE VALUES OF VARIABLES "I" AND "J":

```
> :_I
I = 1
> :_J
J = 0
>
```

PRECEDING ANY VARIABLE NAME OR EXPRESSION WITH THE CHARACTERS ": " (COLON, SPACE) CAUSES THE VALUE OF THE VARIABLE OR THE RESULTANT VALUE OF THE EXPRESSION TO BE PRINTED ON THE USER'S TERMINAL.

THE USER SEES THAT THE VALUE OF "I" IS CORRECT ("I" IS THE CONTROL VARIABLE FOR THE DO LOOP; THIS IS THE FIRST ITERATION THROUGH THE LOOP), HOWEVER THE VALUE OF "J" IS IN ERROR. THE USER DECIDES TO SUSPEND PROGRAM EXECUTION EARLIER IN THE LOOP, IN FUNCTION "SQUARE" IMMEDIATELY AFTER THE FUNCTION VALUE IS COMPUTED. HE ENTERS:

```
> BREAKPOINT_SQUARE\13
> CONTINUE
    0

**** BREAKPOINTED AT SQUARE\13
```

(THE "0" WHICH APPEARS ABOVE IS OUTPUT BY THE FORTRAN "WRITE" STATEMENT ON SOURCE LINE 5.)

THE USER CHOOSES TO LOOK AT THE VALUE OF THE ARGUMENT TO THE FUNCTION AND THAT OF THE COMPUTED RETURNED VALUE:

```
> ARGUMENTS
I = 2
> :_SQUARE
SQUARE = 4
>
```

THE USER FINDS THAT BOTH THE ARGUMENT AND FUNCTION VALUES ARE CORRECT. HE CONTINUES EXECUTION UNTIL THE BREAKPOINT ON SOURCE STATEMENT 5 IN THE MAIN PROGRAM OCCURS BY ENTERING:

```
> CONTINUE

**** BREAKPOINTED AT $MAIN\5
>
```

ONCE AGAIN LOOKING AT THE RETURNED VALUE,

```
> :_J
J = 0
>
```

THE USER FINDS IT INCORRECT. HE SUSPECTS THAT THE DATA TYPES OF THE "SQUARE" FUNCTION MISMATCH BETWEEN THE FORTRAN MAIN PROGRAM AND "SQUARE". THIS IS CONFIRMED BY TYPING:

```
> TYPE_SQUARE
ENTRY CONSTANT EXTERNAL (REAL*4 FUNCTION)
> TYPE_SQUARE\SQUARE
INTEGER*2 STATIC
>
```

THE USER SEES THAT THE DATA TYPES DO INDEED MISMATCH (REAL*4 VS. INTEGER*2). UPON CORRECTING, RECOMPILING, AND RELOADING THE PROGRAM, HE FINDS THAT IT WORKS:

```
OK, DBG_#SQUARES

**DBG**   REVISION 17.0A  (06-FEBRUARY-1979)

> RESTART
    1
    4
    9
   16
   25
   36
   49
   64
   81
  100

EXIT.  PROGRAM EXIT FROM $MAIN\9 ($10+1).
> QUIT

OK,
```

THE "EXIT" MESSAGE INDICATES THAT THE PROGRAM CALLED THE SYSTEM SUBROUTINE "EXIT" FROM SOURCE LINE 9 IN THE FORTRAN MAIN PROGRAM. (THE "$1C+1" ALTERNATIVELY IDENTIFIES THE STATEMENT AS THE ONE FOLLOWING FORTRAN STATEMENT LABEL 10.)

THE "QUIT" COMMAND CAUSES THE DEBUGGER TO EXIT TO PRIMOS COMMAND LEVEL.

7 INVOKING AND TERMINATING THE DEBUGGER

THE DEBUGGER IS INVOKED FROM THE PRIMOS COMMAND LINE BY ENTERING A COMMAND OF THE FORM:

OK, DBG <PROGRAM-NAME> [<OPTION-1> [<OPTION-2> ...]]

WHERE "<PROGRAM-NAME>" IS THE NAME OF THE FILE CONTAINING THE EXECUTABLE IMAGE OF THE PROGRAM TO BE DEBUGGED. <OPTION-1>, <OPTION-2>, ETC. ARE OPTIONAL COMMAND LINE PARAMETERS WHICH DEFINE ATTRIBUTES OF DEBUGGER OPERATION. THE COMMAND LINE OPTIONS AVAILABLE ARE:

-VERIFY_SYMBOLS OR -VFYS
THIS OPTION INDICATES THAT ALL EXTERNAL SYMBOL DECLARATIONS ARE TO BE CHECKED FOR CONSISTENCY IN ALL PROGRAM UNITS CONTAINED WITHIN THE EXECUTABLE FILE. A WARNING MESSAGE IS PRINTED BY THE DEBUGGER DURING INITIALIZATION SHOULD IT ENCOUNTER EXTERNAL SYMBOL DECLARATIONS WHICH DIFFER. (DEFAULT)

-NO_VERIFY_SYMBOLS OR -NVFYS
THIS OPTION IS THE INVERSE OF THE -VERIFY_SYMBOLS OPTION, ABOVE. IT SUPPRESSES EXTERNAL SYMBOL CHECKING, THEREBY SPEEDING UP INITIALIZATION.

-VERIFY_PROC OR -VFYP
THIS OPTION SPECIFIES THAT THE PROCEDURE TEXT IS TO BE VERIFIED TO INSURE THAT STATEMENT BREAKPOINTS MAY BE SET WHERE APPLICABLE. A WARNING MESSAGE IS GENERATED IF THE DEBUGGER ENCOUNTERS A STATEMENT FOR WHICH THE PROCEDURE TEXT IS UNSUITABLE FOR PLACING A BREAKPOINT. (THE WARNING IS USUALLY THE RESULT OF MISUSING THE -POST_MORTEM OPTION (DESCRIBED BELOW).) (DEFAULT)

-NO_VERIFY_PROC OR -NVFYP
THIS OPTION IS THE INVERSE OF THE -VERIFY_PROC OPTION, DESCRIBED ABOVE. IT SPECIFIES THAT THE PROCEDURE TEXT IS NOT TO BE INSPECTED FOR IMPROPER FORMAT WITH REGARDS TO THE PLACEMENT OF BREAKPOINTS.

-COMINPUT OR -CO
THIS OPTION SPECIFIES THAT THE DEBUGGER IS TO ACCEPT INPUT FROM A COMMAND INPUT FILE.

-NO_COMINPUT OR -NCO
THIS OPTION SPECIFIES THAT THE DEBUGGER IS TO ACCEPT INPUT ONLY FROM THE USER TERMINAL; INPUT IS NOT TAKEN FROM A COMMAND INPUT FILE. (DEFAULT)

-FULL_INITIALIZE OR -FI
THIS OPTION CAUSES THE DEBUGGER TO READ AND PROCESS THE ENTIRE SYMBOL TABLE FROM THE SPECIFIED EXECUTABLE FILE PRIOR TO ENTERING COMMAND MODE. NORMALLY, INFORMATION IS READ FROM THE SYMBOL TABLE ONLY WHEN REQUIRED. THIS OPTION IS USEFUL FOR OBTAINING A

COMPLETE EXTERNAL SYMBOL MISMATCH SUMMARY AT INITIALIZATION TIME. (SEE "-VERIFY_SYMBOLS", ABOVE.) USE OF THIS OPTION WILL APPROXIMATELY TRIPLE THE INITIALIZATION TIME.

-QUICK_INITIALIZE OR -QI

THIS OPTION SPECIFIES THAT ONLY INFORMATION REQUIRED TO IDENTIFY EACH BLOCK IS TO BE LOADED AT INITIALIZATION TIME; THE REMAINDER OF THE SYMBOL TABLE IS LOADED AS REQUIRED DURING THE DEBUGGING SESSION. (DEFAULT)

-POST_MORTEM OR -PM

THIS OPTION SPECIFIES A POST-MORTEM ENTRY INTO THE DEBUGGER. IT CAUSES NO RESTORATION OF PROCEDURE TEXT, BUT RATHER ASSUMES THAT THE PROCEDURE TEXT HAS BEEN RESTORED AND THE USER STACK IS INTACT. IT IS INTENDED THAT THIS OPTION BE USED FOLLOWING A PROGRAM ABORT (VIA "ACCESS VIOLATION", ETC.) TO ATTEMPT TO DETERMINE THE CAUSE OF THE ERROR. ABSENCE OF THIS OPTION INDICATES THAT THE PROCEDURE TEXT IS TO BE RESTORED AND THAT THE PREVIOUS STACK INFORMATION (IF ANY) IS TO BE DISREGARDED.

THE -POST_MORTEM ARGUMENT IS NOT AVAILABLE AT REVISION 17.

WHEN THE DEBUGGER HAS COMPLETED ITS INITIALIZATION, IT PRINTS AN IDENTIFICATION MESSAGE ON THE USER TERMINAL, AND ENTERS COMMAND MODE (SIGNIFIED BY THE ">" PROMPT):

```
**DBG**  REVISION 17.0  (1-JANUARY-79)

>
```

APPEARANCE OF THE PROMPT CHARACTER INDICATES THAT THE USER MAY ENTER DEBUGGER COMMANDS.

TO TERMINATE THE DEBUGGING SESSION, ENTER THE "QUIT" (OR "Q") COMMAND:

```
> QUIT

OK,
```

THIS WILL CAUSE THE DEBUGGER TO EXIT TO PRIMOS COMMAND LEVEL, AS INDICATED BY THE "OK" MESSAGE.

TO TEMPORARILY SUSPEND THE DEBUGGING SESSION, ENTER THE "PAUSE" (OR "PA") COMMAND:

```
> PAUSE
TO RESUME DEBUGGING, TYPE 'START'.

OK,
```

AFTER RETURNING TO COMMAND LEVEL VIA THE "PAUSE" COMMAND, THE USER MUST BE ESPECIALLY CAREFUL NOT TO EXECUTE ANY COMMANDS WHICH WOULD DISTURB THE MEMORY IMAGE OF HIS PROGRAM, STACK, OR THE DEBUGGER'S SYMBOL TABLE.

8 COMMAND INPUT CONVENTIONS

8.1 GENERAL FORMAT

DEBUGGER COMMANDS ARE ENTERED FOLLOWING THE PROMPT CHARACTER (">") OUTPUT BY DBG. THE GENERAL COMMAND FORMAT IS:

> <COMMAND-NAME> [<MODIFIER>] [<ARGUMENT-1> [<ARGUMENT-2> ...]]

WHERE "<COMMAND-NAME>" IS THE NAME OF THE DEBUGGER COMMAND, "<MODIFIER>" IS AN OPTIONAL COMMAND MODIFIER ("ON" AND "OFF" ARE MOST COMMON), "<ARGUMENT-N>" REPRESENTS ONE OR MORE OPTIONAL COMMAND ARGUMENTS (EXPRESSION, SYMBOL NAME, STATEMENT IDENTIFIER, ETC.).

THE EXCEPTION TO THE ABOVE FORMAT IS THE "EVALUATE EXPRESSION" COMMAND, SIGNIFIED BY A COLON (":"). THE GENERAL FORMAT OF THIS COMMAND IS:

> :[<MODIFIER1>[,<MODIFIER2>]] <EXPRESSION>

THE TEXT IMMEDIATELY FOLLOWING THE COLON (WITH NO INTERVENING SPACE) REPRESENTS ONE OR MORE COMMAND MODIFIERS. THE TEXT FOLLOWING THE COMMAND MODIFIER(S) IS THE EXPRESSION BE EVALUATED. (NB: THIS FORMAT WAS CHOSEN TO EXPEDITE EXPRESSION EVALUATION.) EXPRESSION EVALUATION IS COMPLETELY DESCRIBED IN SECTION 12.

ALL LOWER CASE CHARACTERS ARE MAPPED TO UPPER CASE EXCEPT THOSE APPEARING WITHIN PAIRED QUOTES.

8.2 MULTIPLE COMMANDS PER LINE

MULTIPLE DEBUGGER COMMANDS MAY BE INCLUDED ON A SINGLE LINE BY PLACING A SEPARATOR CHARACTER, INITIALLY SEMICOLON (";"), BETWEEN COMMANDS.

FOR EXAMPLE,

> BREAKPOINT $MAIN\22; LET I = 10; RESTART

(THE SPACES FOLLOWING THE SEMICOLONS ARE INCLUDED FOR CLARITY; THEY ARE IGNORED BY DBG.)

COMMANDS ARE EXECUTED IN THE STANDARD LEFT TO RIGHT FASHION. SHOULD ANY OF THE COMMANDS CAUSE AN ERROR MESSAGE TO BE GENERATED, THE TEXT TO THE RIGHT OF THE OFFENDING COMMAND IS IGNORED. THE USER MAY, HOWEVER, EDIT AND RESUBMIT THE COMMAND LINE WITHOUT HAVING TO RETYPE IT. (SEE THE DESCRIPTION OF THE "RESUBMIT" COMMAND, SECTION 13.7.)

THE SEMICOLON IS INCLUDED LITERALLY IF IT APPEARS WITHIN PAIRED QUOTES, SQUARE BRACKETS (ACTION LISTS), OR IF IT IS PRECEDED BY AN

ESCAPE CHARACTER (INITIALLY UP-ARROW OR, ON SOME TERMINALS, CIRCUMFLEX ("^")).

8.3 SPECIAL CHARACTERS

CERTAIN CHARACTERS WHICH CAN BE ENTERED FROM THE USER TERMINAL HAVE SPECIAL SIGNIFICANCE TO THE DEBUGGER. INCLUDED IN THIS SET IS THE SEPARATOR CHARACTER (SEMICOLON), DESCRIBED ABOVE. OTHER SPECIAL CHARACTERS ARE:

- USER-SPECIFIC ERASE CHARACTER ERASES THE PREVIOUS CHARACTER TYPED. ASSUMING THAT THE USER ERASE CHARACTER WAS DOUBLE-QUOTE (THE SYSTEM DEFAULT), THE COMMAND LINE:

  > ENVIO"RONMENI $P"MAI;"N

  WOULD BE INTERPRETED BY DBG AS:

  > ENVIRONMENT $MAIN

- USER-SPECIFIC KILL CHARACTER CAUSES THE LINE TYPED THUSFAR TO BE IGNORED. ASSUMING THE USER KILL CHARACTER IS QUESTION-MARK (THE SYSTEM DEFAULT), THE COMMAND LINE:

  > BREAKPOINI 20?IRACEPOINI 20

  WOULD BE INTERPRETED BY THE DEBUGGER AS:

  > IRACEPOINI 20

- LEFT BRACKET BEGINS AN ACTION LIST, WHICH MUST BE TERMINATED WITH A MATCHING RIGHT BRACKET. FOR EXAMPLE,

  > BREAKPOINI SUBR1\\ENIRY [ARGS]

  ACTION LISTS ARE DESCRIBED WITH BREAKPOINTS, LATER IN THIS DOCUMENT.

- RIGHT BRACKET TERMINATES AN ACTION LIST. (SEE ABOVE.)

- QUOTE (SINGLE OR DOUBLE) BEGINS A TEXT STRING. WITHIN THIS TEXT STRING, THE SPECIAL MEANINGS OF SEMICOLON, LEFT BRACKET, RIGHT BRACKET, AND THE QUOTE CHARACTER WHICH DIDN'I BEGIN THE STRING (DOUBLE QUOTE IF THE STRING IS SURROUNDED BY SINGLE QUOTES, AND VICE VERSA) ARE IGNORED - THESE CHARACTERS ARE INTERPRETED LITERALLY. THIS STRING MUST BE TERMINATED WITH A MATCHING QUOTE. A QUOTE CHARACTER MAY BE INCLUDED IN THE STRING BY SUPPLYING TWO QUOTE CHARACTERS. EXAMPLES:

'THIS IS A VALID; QUOTED '' STRING.'

"SO'S THIS."

'THIS STRING IS INVALID - QUOTES MISMATCH"

"THIS STRING IS UNTERMINATED

. ESCAPE ALWAYS AFFECTS THE MEANING OF THE CHARACTER OR CHARACTERS WHICH IMMEDIATELY FOLLOW IT. THE TABLE BELOW DESCRIBES THE ACTION TAKEN GIVEN THE CHARACTER FOLLOWING ESCAPE:

CHARACTER	ACTION
<ERASE>	INSERT ERASE CHARACTER LITERALLY
<KILL>	INSERT KILL CHARACTER LITERALLY
[OR]	INSERT LITERAL LEFT OR RIGHT BRACKET
<ESCAPE>	INSERT LITERAL ESCAPE CHARACTER
QUOTE	INSERT LITERAL (SINGLE OR DOUBLE) QUOTE CHARACTER
<SEPARATOR>	INSERT LITERAL SEPARATOR CHARACTER
/ (SLASH)	INSERT NEWLINE (ASCII '212)
<NEWLINE>	CONTINUE ONTO NEXT PHYSICAL LINE (LOCAL C/R ONLY)
U OR U	MAP ALL FOLLOWING LOWER CASE CHARACTERS TO UPPER CASE
L OR L	MAP ALL FOLLOWING UPPER CASE CHARACTERS TO LOWER CASE
NNN	(3 DIGIT OCTAL NUMBER) INSERTS THE ASCII CHARACTER WHICH THIS VALUE REPRESENTS

EXAMPLES:

THE FOLLOWING EXAMPLES ASSUME THAT THE SYSTEM AND DEBUGGER SPECIAL CHARACTER DEFAULTS EXIST (ERASE IS ", LINE-KILL IS ?):

```
> SOURCE LOCATE ^[MACRO ^"STRING
```

ABOVE, THE SPECIAL MEANINGS OF LEFT BRACKET AND DOUBLE QUOTE ARE IGNORED BECAUSE THE CHARACTERS ARE PRECEDED BY THE ESCAPE CHARACTER. THE TEXT STRING TO BE LOCATED IS '[MACRO "STRING'.

```
> LET TEMPSTRING = 'LINE 1^/LINE 2'
```

ABOVE, A NEWLINE CHARACTER IS INSERTED FOLLOWING THE '1' IN THE TEXT STRING.

```
> : '^LTHIS IS MAPPED TO LOWER CASE, ^UTHIS TO UPPER.'
```

IS INTERPRETED BY THE DEBUGGER AS:

```
> : 'THIS IS MAPPED TO LOWER CASE, THIS TO UPPER.'
```

FINALLY,

```
> : 'THIS IS AN UNCOMMON WAY OF REPRESENTING A ^244.'
```

IS INTERPRETED BY THE DEBUGGER AS:

```
> : 'THIS IS AN UNCOMMON WAY OF REPRESENTING A $.'
```

NOTES ON THE USAGE OF ESCAPE:

1) THE ESCAPE CHARACTER REFERRED TO IS NOT THE ASCII ESCAPE CHARACTER, BUT RATHER A LOGICAL ESCAPE CHARACTER WHOSE DEFAULT REPRESENTATION IS AN UP-ARROW ("^"), SOMETIMES REPRESENTED AS CIRCUMFLEX.

2) FROM TIME TO TIME, THE SEQUENCE OF CHARACTERS STARTING WITH THE ESCAPE AND TERMINATING WITH THE LAST CHARACTER AFFECTED BY THE ESCAPE WILL BE REFERRED TO AS AN "ESCAPE SEQUENCE."

9 IDENTIFYING PROGRAM OBJECTS

THIS SECTION DEALS WITH THE IDENTIFICATION OF OBJECTS DEFINED BY A PROGRAM: PROGRAM BLOCKS, VARIABLES, AND STATEMENTS. MOST DEBUGGER COMMANDS REQUIRE ONE OR MORE PROGRAM OBJECTS TO BE SUPPLIED AS ARGUMENTS.

9.1 PROGRAM BLOCKS

THE TERM "PROGRAM BLOCK" IS UNIQUE TO THE DEBUGGER. IT IS INTENDED TO PROVIDE AN ALL-ENCOMPASSING DESCRIPTION OF THE "BASIC" PROGRAM UNIT FOR EACH SUPPORTED LANGUAGE. PROGRAM BLOCKS SERVE TO UNIQUELY IDENTIFY A VARIABLE OR STATEMENT.

THE DEFINITION OF "PROGRAM BLOCK" AS RELATED TO EACH LANGAUGE SUPPORTED BY THE DEBUGGER FOLLOWS:

- FORTRAN: THE FORTRAN PROGRAM BLOCK IS A MAIN PROGRAM, SUBROUTINE, OR FUNCTION. MAIN PROGRAMS ARE IDENTIFIED BY "$MAIN", SUBROUTINES AND FUNCTIONS BY THEIR RESPECTIVE NAMES.

- PL/1: A PL/1 PROGRAM BLOCK IS A PROCEDURE BLOCK OR BEGIN BLOCK. PROCEDURE BLOCKS ARE IDENTIFIED BY THE PROCEDURE NAME. BEGIN BLOCKS ARE IDENTIFIED BY THE TEXT "$BEGIN" IMMEDIATELY FOLLOWED WITH THE LINE NUMBER ON WHICH THE BEGIN BLOCK STARTS. ("$BEGIN112" IDENTIFIES A BEGIN BLOCK WHICH STARTS ON SOURCE LINE NUMBER 112, FOR EXAMPLE.)

  WHEN TWO OR MORE PROCEDURES OR BEGIN BLOCKS HAVE IDENTICAL NAMES, THE USER MUST PROVIDE ENOUGH QUALIFICATION TO DISTINGUISH ONE FROM THE OTHERS. PROGRAM BLOCKS ARE QUALIFIED IN A FASHION IDENTICAL TO THAT USED FOR STRUCTURE MEMBERS. CONSIDER THE FOLLOWING PROGRAM STRUCTURE:

```
A:    PROCEDURE ;
B:    PROCEDURE ;
C:    PROCEDURE ;
      END ;    /* (C) */
      END ;    /* (B) */
      END ;    /* (A) */

B:    PROCEDURE ;
B:    PROCEDURE ;
      END ;    /* (B-INNER) */
      END ;    /* (B-OUTER) */
```

  (ALTHOUGH IN PRACTICE IT IS PROBABLY UNDESIRABLE TO ADOPT THE NAMING SCHEME USED ABOVE, THE EXAMPLE PROVIDES AN EXCELLENT VEHICLE FOR EXPLAINING BLOCK

QUALIFICATION:)

EXTERNAL PROCEDURE "A" IS FULLY QUALIFIED IN THE REFERENCE "A".

INTERNAL PROCEDURE "B" WITHIN EXTERNAL PROCEDURE "A" IS FULLY QUALIFIED IN THE REFERENCE "A.B".

INTERNAL PROCEDURE "C" MAY BE REFERRED TO AS "C", "A.C", "B.C", OR IN FULLY QUALIFIED FORM AS "A.B.C". (NOTE THAT THERE ARE NO OTHER PROCEDURES NAMED "C".)

EXTERNAL PROCEDURE "B" IS FULLY QUALIFIED IN THE REFERENCE "B".

FINALLY, INTERNAL PROCEDURE "B" WITHIN EXTERNAL PROCEDURE "B" IS FULLY QUALIFIED BY "B.B".

THE DEBUGGER WILL PRINT AN ERROR MESSAGE ("AMBIGUOUS BLOCK REFERENCE") SHOULD A PROGRAM BLOCK NAME BE INSUFFICIENTLY QUALIFIED. NOTE THAT THE DEBUGGER WILL ALWAYS FULLY QUALIFY PROGRAM BLOCK NAMES WHEN OUTPUTTING THEM TO THE USER'S TERMINAL.

- COBOL: A COBOL PROGRAM BLOCK IS ONE COMPLETE PROGRAM. IT IS IDENTIFIED BY THE NAME SPECIFIED IN THE "PROGRAM-ID" STATEMENT.

9.2 ENVIRONMENTS

THE DEBUGGER MAINTAINS TWO DISTINCT "ENVIRONMENT" POINTERS WHICH ARE USED AS DEFAULTS TO CERTAIN COMMANDS. THESE ENVIRONMENT POINTERS ARE

- THE EXECUTION ENVIRONMENT POINTER. THIS POINTER DESCRIBES THE LOCATION AT WHICH EXECUTION IS TO BE RESUMED UPON ISSUANCE OF THE "CONTINUE" COMMAND OR ONE OF THE SINGLE-STEP CLASS COMMANDS.

  THE VALUE OF THE EXECUTION ENVIRONMENT POINTER MAY BE CHANGED BY USING THE "GOTO" COMMAND. IT IS UNDEFINED UNLESS ONE OR MORE ACTIVATIONS OF THE USER PROGRAM EXIST.

- THE EVALUATION ENVIPONMENT POINTER. THIS POINTER DEFINES THE DEFAULT PROGRAM BLOCK TO BE USED FOR FINDING VARIABLES, STATEMENTS, AND FOR EXAMINING SOURCE FILES.

  THE VALUE OF THIS POINTER MAY BE CHANGED USING THE

"ENVIRONMENT" COMMAND. WHEN THE DEBUGGER IS ENTERED FROM COMMAND LEVEL, THE VALUE OF THIS POINTER IS SET TO THE PROGRAM BLOCK CORRESPONDING TO THE USER'S MAIN PROGRAM. WHENEVER THE DEBUGGER IS REENTERED FROM THE USER PROGRAM (AS THE RESULT OF A BREAKPOINT, FOR EXAMPLE), THE VALUE OF THIS POINTER IS SET TO THE PROGRAM BLOCK CORRESPONDING TO THE EXECUTION ENVIRONMENT POINTER. WHEN THE DEBUGGER IS REENTERED FOLLOWING USER PROGRAM TERMINATION, THE VALUE OF THIS POINTER IS ONCE AGAIN SET TO THE PROGRAM BLOCK OF THE MAIN PROGRAM.

THE USER MAY THINK OF THE EVALUATION ENVIRONMENT POINTER AS NOTHING MORE THAN A "CONVENIENCE" ITEM. WITHOUT IT, EACH REFERENCE TO A VARIABLE OR STATEMENT WOULD REQUIRE AN ACCOMPANYING PROGRAM BLOCK NAME TO SPECIFY THE BLOCK IN WHICH THE OBJECT (VARIABLE, ETC.) WAS TO BE FOUND. USING THE EVALUATION ENVIRONMENT POINTER, ANY STATEMENT OR VARIABLE REFERENCES WHICH ARE NOT QUALIFIED WITH A PROGRAM BLOCK NAME ARE LOOKED UP IN THIS PROGRAM BLOCK.

9.3 ACTIVATION NUMBERS

ACTIVATION NUMBERS ARE USED TO SPECIFY A PARTICULAR ACTIVATION OF A PROCEDURE WHEN MORE THAN ONE ACTIVATION EXISTS. (MORE THAN ONE ACTIVATION OF A PROCEDURE CAN EXIST IF THE PROCEDURE CALLS ITSELF OR CAUSES ITSELF TO BE CALLED, AND/OR BY USING THE DEBUGGER "CALL" COMMAND.)

ACTIVATION NUMBERS MAY BE SPECIFIED AS ABSOLUTE OR RELATIVE TO THE MOST RECENT ACTIVATION:

- AN ABSOLUTE ACTIVATION NUMBER IS SPECIFIED BY AN UNSIGNED DECIMAL CONSTANT. FOR EXAMPLE, TO LOOK AT VARIABLE "INDEX" IN ACTIVATION 2 OF "RECURSE", ENTER THE COMMAND

  ```
  > :RECURSE\2\INDEX
  ```

  THE FIRST ACTIVATION OF A PROCEDURE IS ALWAYS ACTIVATION 1.

- A RELATIVE ACTIVATION NUMBER IS SPECIFIED BY A MINUS SIGN ("-") IMMEDIATELY FOLLOWED BY A DECIMAL CONSTANT. THIS IS THE NUMBER OF ACTIVATIONS TO COUNT BACKWARDS FROM THE MOST RECENT ACTIVATION OF THE SPECIFIED PROCEDURE. FOR EXAMPLE, IF THERE ARE 5 ACTIVATIONS OF "FACTORIAL", ENTERING THE DEBUGGER COMMAND

  ```
  > ENVIRONMENT FACTORIAL\-1
  ```

  CAUSES THE EVALUATION ENVIRONMENT TO BE SET TO ACTIVATION 4 OF "FACTORIAL".

WHENEVER THE DEBUGGER REFERS TO AN ACTIVATION OF A PROGRAM BLOCK, THE ACTIVATION NUMBER IS PRINTED ONLY IF THERE IS MORE THAN ONE ACTIVATION OF THE BLOCK.

9.4 VARIABLES

THE RULES FOR IDENTIFYING VARIABLES WITHIN THE DEBUGGER ARE IDENTICAL TO THE RULES ESTABLISHED BY THE HOST LANGUAGE. THE SYNTAX HAS BEEN EXPANDED, HOWEVER, AS THE USER IS ALLOWED TO REFERENCE ANY VARIABLE IN THE DEBUGGING ENVIRONMENT AT ANY TIME.

ONE OF THE FOLLOWING FORMATS IS USED TO IDENTIFY A VARIABLE:

(1) <VARIABLE-NAME>

(2) <PROGRAM-BLOCK-NAME> \ <VARIABLE-NAME>

(3) <PROGRAM-BLOCK-NAME> \ <ACTIVATION-NUMBER> \ <VARIABLE-NAME>

WHERE:

<VARIABLE-NAME> IS THE VARIABLE NAME, POSSIBLY QUALIFIED AND/OR SUBSCRIPTED, AS REPRESENTED IN THE HOST LANGUAGE.

<PROGRAM-BLOCK-NAME> IS A THE NAME OF A PROGRAM BLOCK, AS DESCRIBED IN SECTION 9.1.

<ACTIVATION-NUMBER> IS THE ACTIVATION NUMBER OF THE PROGRAM BLOCK. THIS IS ONLY USEFUL FOR AUTOMATIC AND BASED STORAGE (PL/1) OR STORAGE NOT DECLARED AS "SAVED" (FORTRAN).

- FORMAT (1) REFERS TO THE NAMED VARIABLE IN THE PROGRAM BLOCK SPECIFIED BY THE CURRENT EVALUATION ENVIRONMENT. FOR EXAMPLE, TO REFERENCE "VAR1" IN THE CURRENT BLOCK (THAT SPECIFIED BY THE EVALUATION ENVIRONMENT POINTER), YOU WOULD ENTER

```
VAR1
```

- FORMAT (2) REFERS TO THE VARIABLE IN THE SPECIFIED PROGRAM BLOCK. IF THERE IS MORE THAN ONE ACTIVATION OF THIS BLOCK, THE MOST RECENT ACTIVATION IS USED. FOR EXAMPLE, TO REFERENCE "VAR1" IN PROGRAM BLOCK "SUBR1", YOU WOULD ENTER

```
SUBR1\VAR1
```

o FORMAT (3) REFERS TO THE VARIABLE IN THE SPECIFIED PROGRAM BLOCK AND ACTIVATION. FOR EXAMPLE, TO REFERENCE "VAR1" IN (ABSOLUTE) ACTIVATION 2 OF PROGRAM BLOCK "SUBR1", YOU WOULD ENTER

SUBR1\2\VAR1

9.5 STATEMENTS

DEBUGGER COMMANDS SUCH AS "BREAKPOINT" AND "GOTO" REQUIRE THE USER TO IDENTIFY A STATEMENT WITHIN THE DEBUGGING ENVIRONMENT. FOR USER CONVENIENCE, THE DEBUGGER ALLOWS 2 GENERAL METHODS OF IDENTIFYING A STATEMENT; ONE BASED ON A LABEL, THE OTHER ON A SOURCE FILE LINE NUMBER.

ONE OF THE FOLLOWING FORMATS IS USED TO IDENTIFY A STATEMENT:

* (1) <SOURCE-LINE>

(2) <SOURCE-LINE> + <STATEMENT-OFFSET>

(3) <SOURCE-LINE> (<INSERT-LINE>)

(4) <SOURCE-LINE> (<INSERT-LINE> + <STATEMENT-OFFSET>)

* (5) <STATEMENT-LABEL>

(6) <STATEMENT-LABEL> + <LINE-OFFSET>

(7) <STATEMENT-LABEL> + <LINE-OFFSET> + <STATEMENT-OFFSET>

(THE FORMATS MARKED WITH "*" ARE THE MOST COMMON.)

WHERE:

<SOURCE-LINE> IS A SOURCE LINE NUMBER. THIS IS THE PHYSICAL LINE NUMBER IN THE SOURCE FILE ON WHICH THE STATEMENT RESIDES. (DON'T CONFUSE THIS WITH A FORTRAN STATEMENT NUMBER.)

<STATEMENT-OFFSET> IS THE NUMBER OF STATEMENTS TO COUNT FROM THE FIRST STATEMENT ON THE LINE IN ORDER TO FIND THIS STATEMENT. THE FIRST STATEMENT ON A LINE HAS A STATEMENT OFFSET OF 0.

<INSERT-LINE> IS AN INSERT FILE LINE NUMBER. (THAT IS, A PHYSICAL LINE NUMBER IN A FILE "$INSERT'ED" BY THE PRIMARY SOURCE FILE.)

<STATEMENT-LABEL> IS A FORTRAN STATEMENT NUMBER, PL/1 LABEL CONSTANT, OR COBOL PARAGRAPH NAME. FORTRAN STATEMENT LABELS MUST BE PRECEDED BY A DOLLAR SIGN ("$") ($100 IDENTIFIES FORTRAN

STATEMENT LABEL 100, FOR EXAMPLE.) COBOL PARAGRAPH NAMES WHICH START WITH A NUMERIC MUST ALSO BE PRECEDED BY A DOLLAR SIGN.

<LINE-OFFSET> IS THE NUMBER OF PHYSICAL SOURCE LINES FOLLOWING THE LINE CONTAINING <STATEMENT-LABEL>.

NOTE THAT ANY OF THE SEVEN FORMATS DESCRIBED ABOVE MAY BE PRECEDED BY A PROGRAM BLOCK NAME, SEPARATED BY A BACKSLASH:

<PROGRAM-BLOCK-NAME> \ <STATEMENT-IDENTIFIER>

ABSENCE OF THE PROGRAM BLOCK NAME INDICATES THAT THE PROGRAM BLOCK SPECIFIED BY THE EVALUATION ENVIRONMENT POINTER IS TO BE USED TO FIND THE GIVEN STATEMENT.

THE FOLLOWING FORTRAN FUNCTION IS USED AS AN EXAMPLE IN THE TEXT BELOW:

```
(0001)          INTEGER*2 FUNCTION ABSADD (I, J)
(0002)          INTEGER*2 I,J
(0003) 10       ABSADD = I + J
(0004) 20       IF (ABSADD .LT. 0) ABSADD = -ABSADD
(0005)          RETURN
(0006)          END
```

- FORMAT (1), ABOVE, IDENTIFIES THE LEFT-MOST STATEMENT ON THE SPECIFIED SOURCE LINE NUMBER. ASSUMING THAT "ABSADD" IS THE CURRENT EVALUATION ENVIRONMENT, THE STATEMENT REFERENCE

```
3
```

IDENTIFIES THE STATEMENT "ABSADD = I + J" ON SOURCE LINE 3.

- FORMAT (2) IS USEFUL WHEN MULTIPLE STATEMENTS APPEAR ON ONE LINE. IT IS USED TO SPECIFY A STATEMENT WHICH IS NOT THE LEFT-MOST STATEMENT ON THE SOURCE LINE. REFERRING TO THE EXAMPLE PROGRAM ABOVE, SOURCE LINE 4 HAS TWO STATEMENTS, THE IF BEING THE FIRST AND THE ARITHMETIC ASSIGNMENT STATEMENT (WHICH WILL BE EXECUTED SHOULD THE IF-EXPRESSION YIELD TRUE), THE SECOND. TO IDENTIFY THE ASSIGNMENT STATEMENT, YOU MIGHT USE

```
ABSADD\4+1
```

IF THE EVALUATION ENVIRONMENT WAS OTHER THAN "ABSADD", OR JUST

```
4+1
```

IF THE EVALUATION ENVIRONMENT WAS SET TO "ABSADD".

- FORMATS (3) AND (4) ARE INCLUDED FOR THOSE USERS WHO USE EXECUTABLE STATEMENTS IN $INSERT FILES. (NB: THE PRIMARY USE OF $INSERT FILES IS DATA DECLARATION.) IN THESE CASES, THE SOURCE LINE NUMBER IS THE LINE NUMBER IN THE PRIMARY SOURCE FILE WHICH INCLUDES THE $INSERT DIRECTIVE. THE INSERT LINE NUMBER AND STATEMENT OFFSET ARE USED AS DESCRIBED ABOVE (FORMATS (1) AND (2)). THESE WERE NOT INCLUDED IN THE EXAMPLE PROGRAM BECAUSE OF THEIR INFREQUENT USE. EXAMPLE:

  FFT02\12(2)

  SPECIFIES THE LEFT-MOST STATEMENT ON PHYSICAL LINE NUMBER 2 IN THE FILE $INSERT'ED ON PRIMARY FILE LINE NUMBER 12 IN ROUTINE "FFT02".

- FORMAT (5) IS USED TO IDENTIFY A STATEMENT BY THE LABEL ASSOCIATED WITH IT. AS STATED ABOVE, REFERENCES TO FORTRAN STATEMENT NUMBERS MUST BE IMMEDIATELY PRECEDED BY A DOLLAR SIGN. TO IDENTIFY FORTRAN STATEMENT NUMBER 20 IN "ABSADD", YOU MIGHT USE

  ABSADD\$20

  OR, IF THE EVALUATION ENVIRONMENT WAS SET TO "ABSADD", JUST

  $20

- FORMAT (6) ALLOWS IDENTIFICATION OF A STATEMENT BY SPECIFYING A STATEMENT LABEL AND LINE OFFSET. THE STATEMENT REFERENCED USING THIS FORMAT IS ALWAYS THE LEFT-MOST STATEMENT ON THE LINE "LINE-OFFSET" LINES FOLLOWING THE SOURCE LINE ON WHICH THE SPECIFIED LABEL IS DEFINED. TO IDENTIFY THE RETURN STATEMENT ON SOURCE LINE 5 USING THIS FORMAT, YOU WOULD ENTER

  $20+1

  NOTE THAT NO "BACKWARDS" REFERENCING IS AVAILABLE, E.G. YOU CANNOT REFERENCE THE STATEMENT ON SOURCE LINE 3 BY ENTERING

  $20-1

  FORMAT (6) SHOULD NOT BE CONFUSED WITH FORMAT (2), DESCRIBED ABOVE. RECALL THAT THE ITEM PRECEDING THE PLUS SIGN IN FORMAT (2) IS A SOURCE LINE NUMBER; THE ITEM IN FORMAT (6) IS A STATEMENT LABEL.

- FORMAT (7) IS SIMILAR IN FUNCTION TO FORMAT (6) BUT ALLOWS THE USER TO IDENTIFY A STATEMENT WHICH IS NOT THE LEFTMOST ON THE SOURCE LINE SPECIFIED BY STATEMENT-LABEL + LINE-OFFSET. TO IDENTIFY THE ARITHMETIC ASSIGNMENT STATEMENT ON SOURCE LINE 4

USING THIS FORMAT, YOU WOULD ENTER

```
$10+1+1
```

THE FIRST "+1" SPECIFIES THE NUMBER OF PHYSICAL SOURCE LINES BEYOND THE LINE ON WHICH FORTRAN STATEMENT LABEL 10 IS DEFINED. THIS IS THE SOURCE LINE ON WHICH THE STATEMENT WILL BE FOUND. THE RIGHT-MOST "+1" SPECIFIES THAT THIS IS THE SECOND STATEMENT ON THE LINE. (RECALL THAT THIS IS AN OFFSET AND THAT THE FIRST STATEMENT IS "+0".)

THE IDENTIFICATION OF PL/1 SOURCE STATEMENTS BASED UPON SOURCE LINE NUMBER DIFFERS SLIGHTLY FROM THAT OF FORTRAN STATEMENTS.

WHEN MULTIPLE PL/1 STATEMENTS APPEAR WITHOUT AN INTERVENING SEMICOLON, EACH STATEMENT IS CONSIDERED TO RESIDE ON THE SOURCE LINE WHICH CONTAINS THE FIRST TOKEN OF THE FIRST STATEMENT.

AN EXAMPLE BEST ILLUSTRATES THIS DIFFERENCE:

```
(0010)   IF I < 100 THEN                    /* 10   */
(0011)       IF I < N THEN                  /* 10+1 */
(0012)           /* INCREMENT */
(0013)           I = I + 1 ;                /* 10+2 */
(0014)       ELSE                           /* 14   */
(0015)           /* DECREMENT */
(0016)           I = I - 1 ;
(0017)
(0018)   /*   NOW UPDATE LINKED LIST   */
(0019)   CALL UPDATE_LIST (I) ;             /* 19   */
```

ALTHOUGH THE SECOND "IF" STATEMENT ACTUALLY RESIDES ON SOURCE LINE 11, ACCORDING TO THE ABOVE RULE, IT IS REFERENCED AS THE SECOND STATEMENT ON SOURCE LINE NUMBER 10. THE SAME REASONING APPLIES TO THE ARITHMETIC ASSIGNMENT STATEMENT ON SOURCE LINE 13.

THE ARITHMETIC ASSIGNMENT STATEMENT ON SOURCE LINE 16 IS THE SUBJECT OF THE ELSE CLAUSE ON LINE 14. AS THE WORD "ELSE" IS THE FIRST TOKEN IN THE ENSUING STATEMENT, ITS PRESENCE ON SOURCE LINE 14 DENOTES THAT THE ASSIGNMENT STATEMENT ON LINE 16 BE REFERENCED AS THOUGH IT RESIDED ON LINE 14.

10 PROGRAM CONTROL COMMANDS

THIS SECTION DESCRIBES THE DBG COMMANDS WHICH AFFECT USER PROGRAM EXECUTION.

10.1 RESTARTING THE PROGRAM - THE "RESTART" COMMAND

THE "RESTART" COMMAND IS USED TO START OR RESTART EXECUTION OF THE USER PROGRAM. ISSUANCE OF THIS COMMAND HAS THE FOLLOWING EFFECTS:

- ALL STACK FRAMES OWNED BY THE USER PROGRAM ARE RELEASED. THAT IS, THE STACK IS UNWOUND TO THE DEBUGGER-OWNED FRAME AT THE STACK ROOT. USERS OF THE "CALL" COMMAND WHO CALL PROCEDURES WHICH ARE BREAKPOINTED SHOULD SPECIFICALLY NOTE THAT USE OF THE "RESTART" COMMAND CAUSES THESE CALLED PROCEDURES TO BECOME INACTIVE.

- THE USER'S MAIN PROGRAM IS PROCEDURE-CALLED BY THE DEBUGGER WITH NO ARGUMENTS. SHOULD THE PROGRAM EXPECT ARGUMENTS, THE "RESTART" COMMAND PROCESSOR WILL PRINT AN APPROPRIATE ERROR DIAGNOSTIC AND RETURN TO DEBUGGER COMMAND MODE.

THE "RESTART" COMMAND MAY OPTIONALLY BE FOLLOWED BY A SINGLE STEP COMMAND TO EFFECT PROGRAM RESTART AND RETURN CONTROL TO DBG COMMAND LEVEL FOLLOWING THE EXECUTION OF A SPECIFIED NUMBER OF STATEMENTS.

THE FORMAT OF THE "RESTART" COMMAND (ABBREVIATED "RST") IS:

```
RESTART  [<STEP-COMMAND>]
```

WHERE:

<STEP-COMMAND> IS A "STEP", "STEPIN", OR "IN" COMMAND FOLLOWED BY THE APPROPRIATE ARGUMENTS. NOTE THAT NO SEMICOLON SEPARATES THE "RESTART" COMMAND FROM THE SINGLE STEP COMMAND.

IF NO STEP COMMAND IS SUPPLIED, THE PROGRAM IS RESTARTED AND EXECUTION CONTINUES UNTIL ONE OF THE EVENTS DESCRIBED IN SECTION 6.4 OCCURS, AT WHICH TIME CONTROL RETURNS TO THE DEBUGGER.

IF A STEP COMMAND FOLLOWS "RESTART", THE PROGRAM IS RESTARTED AS DESCRIBED ABOVE AND THEN SINGLE STEPPED AS DESCRIBED IN SECTION 16.

EXAMPLES:

1) TO RESTART PROGRAM EXECUTION:

```
> RESTART
```

2) TO RESTART PROGRAM EXECUTION AND STOP FOLLOWING THE PROCEDURE CALL TO THE MAIN PROGRAM:

```
> RESTART_IN
```

10.2 CONTINUING PROGRAM EXECUTION - THE "CONTINUE" COMMAND

THE "CONTINUE" COMMAND IS USED TO CONTINUE PROGRAM EXECUTION FOLLOWING A BREAKPOINT, CONDITION SIGNAL, OR SINGLE STEP OPERATION. THE PROGRAM IS RESUMED AT THE LOCATION SPECIFIED BY THE EXECUTION ENVIRONMENT POINTER, WHICH, UNLESS MODIFIED BY THE USER, IS THE LOCATION AT WHICH CONTROL WAS RETURNED TO THE DEBUGGER FROM THE USER PROGRAM. (THE VALUE OF THE EXECUTION ENVIRONMENT POINTER MAY BE MODIFIED USING THE "GOTO" COMMAND AND DISPLAYED USING THE "WHERE" COMMAND; SEE SECTIONS 10.3 AND 14.1 FOR DETAILS.)

- IF THE EXECUTION ENVIRONMENT POINTER DESCRIBES A STATEMENT, PROGRAM BLOCK ENTRY, OR PROGRAM BLOCK EXIT, CONTROL RETURNS TO THE USER PROGRAM AT THE ASSOCIATED LOCATION.

- IF THE EXECUTION ENVIRONMENT POINTER DESCRIBES THE DEBUGGER DEFAULT ON-UNIT FOR A PARTICULAR CONDITION, CONTROL RETURNS TO THAT ON-UNIT WHICH RETURNS TO THE PRIMOS CONDITION MECHANISM IN THE APPROPRIATE MANNER. SHOULD THE SIGNALLER OF THE CONDITION HAVE SPECIFIED THAT RETURN FROM THIS CONDITION IS ILLEGAL, AN "ILLEGAL_ONUNIT_RETURN$" CONDITION IS IMMEDIATELY SIGNALLED BY THE CONDITION MECHANISM AND CAUGHT BY THE DEBUGGER. THE USER MAY CONTINUE PROGRAM EXECUTION ONLY FOLLOWING ISSUANCE OF A "GOTO" COMMAND (SEE SECTION 10.3); PROGRAM EXECUTION MAY, HOWEVER, BE RESTARTED.

- IF THE VALUE OF THE EXECUTION ENVIRONMENT POINTER IS UNDEFINED, AN ATTEMPT TO CONTINUE WILL FAIL; THE DEBUGGER WILL PRINT AN APPROPRIATE ERROR DIAGNOSTIC AND RETURN TO DBG COMMAND MODE.

PROGRAM EXECUTION CONTINUES UNTIL ONE OF THE EVENTS DESCRIBED IN SECTION 6.4 OCCURS, AT WHICH TIME CONTROL RETURNS TO THE DEBUGGER.

THE FORMAT OF THE "CONTINUE" COMMAND (ABBREVIATED "C", WITH SYNONYMS "PROCEED" AND "PR") IS:

```
CONTINUE
```

EXAMPLE:

TO CONTINUE PROGRAM EXECUTION FROM THE LAST BREAKPOINT, ENTER

```
> C
```

10.3 TRANSFERRING PROGRAM CONTROL - THE "GOTO" COMMAND

THE "GOTO" COMMAND IS USED TO MODIFY THE VALUE OF THE EXECUTION ENVIRONMENT POINTER. THE EFFECT OF USING THIS COMMAND IS TO CAUSE CONTROL TO RETURN TO A SPECIFIC STATEMENT WHEN PROGRAM EXECUTION IS RESUMED, WITH EITHER THE "CONTINUE" COMMAND (SECTION 10.2) OR A SINGLE STEP COMMAND (SECTION 16).

THE USER MAY ONLY "GOTO" A STATEMENT IN A PROCEDURE WHICH IS ACTIVE. IF THE STATEMENT IS NON-LOCAL TO THE PROGRAM BLOCK SPECIFIED BY THE CURRENT EXECUTION ENVIRONMENT POINTER, A NON-LOCAL GOTO IS PERFORMED TO PROPERLY UNWIND THE STACK. IN ANY CASE, THE EVALUATION ENVIRONMENT IS SET TO THE "GONE-TO" PROGRAM BLOCK AND THE EVALUATION LANGUAGE IS SET TO THE SOURCE LANGUAGE OF THE NEW PROGRAM BLOCK.

THE "GOTO" COMMAND MAY BE USED TO BYPASS A DEBUGGER "CALL" FRAME, UNWIND THE STACK, AND RETURN FROM A PROGRAM BLOCK INVOKED WITH THE "CALL" COMMAND. (SEE SECTION 19 FOR DETAILS.)

THE GENERIC FORM OF THE "GOTO" (OR "GO TO") COMMAND IS:

```
GOTO [<PROGRAM-BLOCK-NAME> \ [<ACTIVATION-NUMBER> \ ]]
     <STATEMENT-IDENTIFIER>
```

WHERE:

<PROGRAM-BLOCK-NAME> IS THE NAME OF A PROGRAM BLOCK, AS DESCRIBED IN SECTION 9.1.

<ACTIVATION-NUMBER> IS A PROGRAM BLOCK ACTIVATION NUMBER, AS DESCRIBED IN SECTION 9.3.

<STATEMENT-IDENTIFIER> IS A STATEMENT IDENTIFIER, AS DESCRIBED IN SECTION 9.5.

IF THE PROGRAM BLOCK NAME IS OMITTED, THE SPECIFIED STATEMENT IS ASSUMED TO BE IN THE PROGRAM BLOCK DEFINED BY THE EVALUATION ENVIRONMENT POINTER.

IF AN ACTIVATION NUMBER IS SPECIFIED, THE STACK IS UNWOUND TO THIS ACTIVATION AND THE EXECUTION ENVIRONMENT POINTER IS SET TO THE SPECIFIED STATEMENT. IF NO ACTIVATION NUMBER IS SUPPLIED, THE MOST RECENT ACTIVATION OF THE PROGRAM BLOCK IS ASSUMED.

EXAMPLES:

1) TO SET THE EXECUTION POINTER TO FORTRAN STATEMENT LABEL 200 IN THE CURRENT EVALUATION ENVIRONMENT:

```
> GOTO_$200
```

2) TO SET THE EXECUTION ENVIRONMENT POINTER TO THE STATEMENT ON SOURCE LINE 12 IN THE FOURTH ACTIVATION OF PROCEDURE "FACT":

```
> GOTO_FACT\4\12
```

10.4 DEFINING THE MAIN PROGRAM - THE "MAIN" COMMAND

THE "MAIN" COMMAND IS USED

- TO DEFINE THE PROGRAM UNIT TO BE CALLED IN THE EVENT OF A "RESTART" COMMAND, AND
- TO CAUSE THE DEBUGGER TO PRINT THE NAME OF THE MAIN PROGRAM.

THE GENERIC FORM OF THE "MAIN" COMMAND IS:

```
MAIN [<PROGRAM-BLOCK-NAME>]
```

WHERE:

<PROGRAM-BLOCK-NAME> IS THE NAME OF A PROGRAM BLOCK, AS DESCRIBED IN SECTION 9.1.

IF A PROGRAM BLOCK NAME IS SUPPLIED, THE PROGRAM BLOCK WHICH IS PROCEDURE-CALLED ON A "RESTART" COMMAND IS SET TO THAT SPECIFIED.

IF THE PROGRAM BLOCK NAME IS OMITTED, THE DEBUGGER PRINTS THE NAME OF THE MAIN PROGRAM ON THE USER'S TERMINAL.

EXAMPLES:

1) TO DETERMINE THE NAME OF THE MAIN PROGRAM:

```
> MAIN
MAIN PROGRAM IS "PMAIN".
```

2) TO SET THE MAIN PROGRAM TO "FTMAIN":

```
> MAIN FTMAIN
```

10.5 UNWINDING THE STACK - THE "UNWIND" COMMAND

THE "UNWIND" COMMAND IS USED TO RELEASE ALL ACTIVATIONS OF THE USER PROGRAM AND DEBUGGER FROM THE PROCEDURE CALL / RETURN STACK AND CAUSE THE EXECUTION ENVIRONMENT POINTER TO BECOME UNDEFINED.

THIS COMMAND IS ESPECIALLY USEFUL WHEN MULTIPLE INVOCATIONS OF DBG EXIST AS A RESULT OF USING THE "CALL" COMMAND. (SEE SECTION 19 FOR DETAILS.)

THE FORMAT OF THE "UNWIND" COMMAND IS:

```
UNWIND
```

EXAMPLE:

```
(2:SUBRA) > UNWIND
>
```

11 BREAKPOINTS AND TRACEPOINTS

FREQUENTLY, IT IS USEFUL TO SUSPEND PROGRAM EXECUTION AT CERTAIN POINTS TO INSPECT VARIABLES, PROCEDURE CALL HISTORY, ETC.

THE DEBUGGER BREAKPOINT FACILITY ALLOWS THE USER TO SUSPEND PROGRAM EXECUTION:

- PRIOR TO THE EXECUTION OF A STATEMENT,
- IMMEDIATELY FOLLOWING THE ENTRY TO A PROGRAM BLOCK (THROUGH THE PRIMARY OR AN ALTERNATE ENTRY POINT), AND
- IMMEDIATELY FOLLOWING THE EXIT FROM A PROGRAM BLOCK.

FURTHERMORE, DBG ALLOWS THE USER TO DEFINE CONDITIONAL BREAKPOINTS AND ATTACH "ACTION LISTS" OF DEBUGGER COMMANDS TO BE EXECUTED WHEN A BREAKPOINT OCCURS.

TRACEPOINTS ARE SIMILAR TO BREAKPOINTS; HOWEVER, INSTEAD OF RETURNING CONTROL TO DEBUGGER COMMAND LEVEL OR TO AN ACTION LIST, ENCOUNTERING A TRACEPOINT CAUSES A MESSAGE TO BE PRINTED ON THE USER TERMINAL, FOLLOWED IMMEDIATELY BY RESUMPTION OF PROGRAM EXECUTION.

IN THE TEXT BELOW, REFERENCES TO BREAKPOINTS ALSO APPLY TO TRACEPOINTS UNLESS OTHERWISE NOTED.

11.1 IDENTIFYING A BREAKPOINT

THIS SECTION DESCRIBES THE FORMAT FOR IDENTIFYING A BREAKPOINT TO THE DEBUGGER.

(1) STATEMENT BREAKPOINTS ARE IDENTIFIED WITH A STATEMENT IDENTIFIER, OPTIONALLY QUALIFIED BY PROGRAM BLOCK NAME, AS DESCRIBED IN SECTION 9.5.

(2) ENTRY AND EXIT BREAKPOINTS ARE IDENTIFIED BY EITHER

<PROGRAM-BLOCK-NAME> \\ <BREAKPOINT-TYPE>

OR JUST

\ <BREAKPOINT-TYPE>

<PROGRAM-BLOCK-NAME> IS A PROGRAM BLOCK NAME, AS DESCRIBED IN SECTION 9.1. IF OMITTED (AS SHOWN IN THE SECOND FORM), THE PROGRAM BLOCK SPECIFIED BY THE CURRENT EVALUATION ENVIRONMENT IS USED. <BREAKPOINT-TYPE> IS THE TYPE OF BREAKPOINT, "ENTRY" (OR "EN") FOR AN ENTRY BREAKPOINT, "EXIT" (OR "EX") FOR AN EXIT BREAKPOINT. FOR EXAMPLE, TO IDENTIFY THE BREAKPOINT AT THE ENTRY TO PROCEDURE "CONVERT", YOU WOULD ENTER

```
CONVERT\\ENTRY
```

OR, IF THE EVALUATION ENVIRONMENT WAS "CONVERT", THIS COULD BE SHORTENED TO

```
\ENTRY
```

(3) TO IDENTIFY A BREAKPOINT AT AN ALTERNATE ENTRY TO A PROGRAM BLOCK, YOU WOULD ENTER

```
<PROGRAM-BLOCK-NAME> \ <ALTERNATE-ENTRY-NAME> \\ ALTERNATE
```

OR, IF THE EVALUATION ENVIRONMENT IS <PROGRAM-BLOCK-NAME>, THIS CAN BE ABBREVIATED TO

```
<ALTERNATE-ENTRY-NAME> \\ ALTERNATE
```

FOR EXAMPLE, TO IDENTIFY THE BREAKPOINT AT ALTERNATE ENTRY "CONVERT1" IN PROCEDURE "CONVERT", ENTER

```
CONVERT\CONVERT1\\ALTERNATE
```

THE SPECIFIER "ALTERNATE" CAN BE ABBREVIATED "ALT".

(4) A SHORT-HAND FORM OF SPECIFYING BREAKPOINTS AT THE ENTRY TO AND EXIT FROM A PROGRAM BLOCK IS AVAILABLE:

```
<PROGRAM-BLOCK-NAME> \
```

THIS SPECIFICATION IS ONLY RECOGNIZED FOLLOWING A "BREAKPOINT", "TRACEPOINT", "CLEAR", OR "LIST" COMMAND. E.G., TO SET A BREAKPOINT AT THE ENTRY TO AND EXIT FROM "COMPARE" (ASSUMING NO ALTERNATE ENTRIES EXIST), YOU WOULD ENTER

```
> BREAKPOINT COMPARE\
```

11.2 BREAKPOINT ATTRIBUTES

ASSOCIATED WITH EACH BREAKPOINT IS A COUNTER WHICH KEEPS TRACK OF THE NUMBER OF TIMES THAT THE BREAKPOINTED STATEMENT, ENTRY, OR EXIT HAS BEEN EXECUTED. THIS COUNTER IS RESET TO ZERO WHEN THE BREAKPOINT IS CREATED AND INCREMENTED BY ONE EACH TIME THE BREAKPOINTED LOCATION IS ENCOUNTERED.

THE USER CAN CAUSE THE BREAKPOINT TRAP TO OCCUR CONDITIONALLY BY USING THE "AFTER", "BEFORE", AND/OR "EVERY" BREAKPOINT OPTIONS. ASSOCIATED WITH EACH OPTION IS A VALUE WHICH IS COMPARED TO THE

VALUE OF THE COUNTER EVERY TIME THE BREAKPOINT IS ENCOUNTERED:

- IF "AFTER" IS SPECIFIED, THE BREAKPOINT TRAP OCCURS ONLY WHEN THE VALUE OF THE COUNTER EXCEEDS THE VALUE ASSOCIATED WITH THE "AFTER" SPECIFICATION.

- LIKEWISE, IF "BEFORE" IS SPECIFIED, THE TRAP OCCURS ONLY WHEN THE VALUE OF THE COUNTER IS LESS THAN THE VALUE ASSOCIATED WITH THE "BEFORE" SPECIFICATION.

- IF "EVERY" IS SPECIFIED, THE BREAKPOINT TRAP OCCURS EACH "N" ITERATIONS THROUGH THE BREAKPOINTED LOCATION, WHERE "N" IS THE VALUE ASSOCIATED WITH THE "EVERY" SPECIFICATION.

SHOULD MORE THAN ONE OF THESE OPTIONS BE ATTACHED TO A BREAKPOINT, THEY WORK INCLUSIVELY.

FOR EXAMPLE, TO SET A BREAKPOINT AT THE ENTRY TO "ADDARRAY" AND TO CAUSE THE TRAP TO OCCUR FOLLOWING THE SECOND OCCURRENCE AND PRIOR TO THE SIXTH OCCURRENCE (I.E. ON THE THIRD, FOURTH, AND FIFTH CALLS), YOU WOULD ENTER THE COMMAND

```
> BREAKPOINT ADDARRAY\\ENTRY -AFTER 2 -BEFORE 6
```

(THE "BREAKPOINT" COMMAND FORMAT AND OPTIONS ARE EXPLAINED IN DETAIL IN SECTION 11.4.)

ANOTHER ATTRIBUTE WHICH CAN BE ASSOCIATED WITH A BREAKPOINT IS THE "IGNORE" FLAG. PRESENCE OF THIS FLAG (WHICH CAN BE SET AND RESET FROM DEBUGGER COMMAND LEVEL) INFORMS THE DEBUGGER THAT THE BREAKPOINT TRAP IS NEVER TO BE TAKEN. WHENEVER AN IGNORED BREAKPOINT IS ENCOUNTERED, ONLY THE COUNTER IS INCREMENTED; USER PROGRAM EXECUTION THEN CONTINUES. "IGNORING" A BREAKPOINT IS USEFUL FOR RETAINING ALL INFORMATION ASSOCIATED WITH IT (ESPECIALLY ACTION LISTS) BUT SUPPRESSING THE TRAP.

11.3 ACTION LISTS

AN ACTION LIST IS A SET OF ONE OR MORE DEBUGGER COMMANDS TO BE EXECUTED WHEN A BREAKPOINT TRAP OCCURS. WHEN A TRAP OCCURS AT A BREAKPOINT WITH AN ATTACHED ACTION LIST, CONTROL IS NOT RETURNED TO DEBUGGER COMMAND LEVEL, BUT RATHER, DBG COMMANDS ARE READ AND EXECUTED FROM THE ACTION LIST. IF THE ACTION LIST TERMINATES WITHOUT RETURNING CONTROL TO THE PROGRAM (WITH A "CONTINUE" COMMAND, FOR EXAMPLE), CONTROL IS PASSED TO THE DEBUGGER COMMAND LEVEL, SIGNIFIED BY A "> " PROMPT IN THE LEFT MARGIN OF THE USER TERMINAL.

AN ACTION LIST IS SPECIFIED BY ENCLOSING ONE OR MORE DEBUGGER COMMANDS WITHIN PAIRED SQUARE BRACKETS ("[ ]"). FOR EXAMPLE,

```
[WHERE; : 'CONTINUING...'; CONTINUE]
```

ALSO, ACTION LISTS MAY BE NESTED, AS IN THE FOLLOWING DEBUGGER COMMAND:

```
> BREAKPOINT $MAIN\20 [IF I .NE. 20 [CONTINUE] ELSE [WHERE]]
```

IN THE ABOVE EXAMPLE, A BREAKPOINT IS SET IN THE FORTRAN MAIN PROGRAM AT THE STATEMENT ON SOURCE LINE NUMBER 20; WHEN THE BREAKPOINT TRAP IS TAKEN, THE VALUE OF VARIABLE "I" IS COMPARED TO 20 (SEE THE "IF" COMMAND DESCRIPTION, SECTION 13.3). IF THEY ARE NOT EQUAL, PROGRAM EXECUTION CONTINUES, ELSE THE VALUE OF THE EXECUTION ENVIRONMENT POINTER IS PRINTED (THE "WHERE" COMMAND IS DESCRIBED IN SECTION 14.1) AND CONTROL RETURNS TO THE DEBUGGER'S COMMAND LEVEL.

ACTION LISTS ARE NOT AVAILABLE ON TRACEPOINTS.

11.4 SETTING BREAKPOINTS - THE "BREAKPOINT" COMMAND

THERE ARE THREE USES OF THE "BREAKPOINT" COMMAND:

- TO SET A NEW BREAKPOINT - IF NO BREAKPOINT EXISTS AT THE SPECIFIED LOCATION, ONE IS CREATED WITH THE SUPPLIED ATTRIBUTES (IF ANY),

- TO MODIFIY THE ATTRIBUTES OF AN EXISTING BREAKPOINT - IF A BREAKPOINT ALREADY EXISTS AT THE SPECIFIED LOCATION, THE ATTRIBUTES ARE MODIFIED IN ACCORDANCE WITH THOSE SUPPLIED ON THE COMMAND LINE; VALUES OF ATTRIBUTES NOT SPECIFIED REMAIN UNCHANGED, AND

- TO CONVERT A TRACEPOINT INTO A BREAKPOINT - IF A TRACEPOINT EXISTS AT THE SPECIFIED LOCATION, IT IS REPLACED BY A BREAKPOINT WITH THE SPECIFIED ATTRIBUTES (IF ANY); ATTRIBUTES NOT SPECIFIED REMAIN UNCHANGED.

THE GENERIC FORMAT FOR THE "BREAKPOINT" COMMAND (ABBREVIATED "BRK") IS:

```
BREAKPOINT [<BREAKPOINT-IDENTIFIER>] [<ACTION-LIST>]
           [-AFTER <VALUE>] [-BEFORE <VALUE>] [-EVERY <VALUE>]
           [-COUNT <VALUE>] [-IGNORE < -NIGNORE] [-EDIT]
```

WHERE:

<BREAKPOINT-IDENTIFIER> IS A BREAKPOINT IDENTIFIER, AS DESCRIBED IN SECTION 11.1.

<ACTION-LIST> IS AN ACTION LIST, AS DESCRIBED IN SECTION 11.3.

<VALUE> IS A POSITIVE INTEGER VALUE.

SHOULD THE BREAKPOINT IDENTIFIER BE OMITTED, THE VALUE OF THE EXECUTION ENVIRONMENT POINTER IS USED.

IF AN ACTION LIST IS SPECIFIED, IT REPLACES THE ONE (IF ANY) ATTACHED TO THE BREAKPOINT. TO DELETE AN ACTION LIST, ENTER THE STRING "[]" (I.E., AN "EMPTY" ACTION LIST).

"-AFTER", "-BEFORE", AND "-EVERY" ARE USED TO SPECIFY THE RANGE AND FREQUENCY TO TRAP AT THIS BREAKPOINT AND RETURN CONTROL TO EITHER AN ACTION LIST (IF PRESENT) OR TO THE DEBUGGER'S COMMAND LEVEL. THEIR FUNCTIONS ARE DESCRIBED IN SECTION 11.2. TO DISABLE ONE OF THESE ATTRIBUTES, SUPPLY A VALUE OF ZERO.

"-COUNT" IS USED TO RESET THE BREAKPOINT COUNTER; IT IS SET TO THE VALUE WHICH FOLLOWS THIS OPTION. THE BREAKPOINT COUNTER IS DESCRIBED IN SECTION 11.2.

"-IGNORE" AND "-NIGNORE" ARE USED TO SET AND RESET THE "IGNORE" FLAG, RESPECTIVELY. THE "IGNORE" FLAG IS DESCRIBED IN SECTION 11.2.

"-EDIT" SPECIFIES THAT THE ACTION LIST ASSOCIATED WITH THIS BREAKPOINT IS TO BE EDITED. EDITING IS DESCRIBED IN SECTION 21.

EXAMPLES:

1) TO SET A BREAKPOINT AT LABEL "CONVERSIONLOOP" IN THE CURRENT EVALUATION ENVIRONMENT:

```
> BRK_CONVERSIONLOOP
```

2) TO SET A BREAKPOINT AT THE CURRENT EXECUTION POINTER (ASSUMING NONE ALREADY EXISTS):

```
> BRK
```

3) TO SET A BREAKPOINT AT THE ENTRY TO "CDELTA" AND TO CAUSE THE ARGUMENTS TO BE PRINTED EVERY TIME THE PROCEDURE IS CALLED, FOLLOWED BY RETURN TO DEBUGGER COMMAND LEVEL:

```
> BRK_CDELTA\\ENTRY_[ARGUMENTS]
```

4) TO DELETE THE ACTION LIST ATTACHED TO THE BREAKPOINT SHOWN IN EXAMPLE 3:

```
> BRK_CDELTA\\ENTRY_[]
```

5) TO SET A BREAKPOINT ON THE STATEMENT AT SOURCE LINE 75 IN THE FORTRAN MAIN PROGRAM AND HAVE THE BREAKPOINT TRAP OCCUR EVERY OTHER TIME THE STATEMENT IS EXECUTED:

```
> BRK $MAIN\75 -EVERY 2
```

6) TO MODIFY THE BREAKPOINT SET IN EXAMPLE 5 TO ONLY TRAP FOLLOWING THE TENTH OCCURRENCE AND AT EVERY EXECUTION OF THE STATEMENT THEREAFTER:

```
> BRK $MAIN\75 -AFTER 10 -EVERY 1
```

7) TO EDIT THE ACTION LIST ON THE CURRENT BREAKPOINT:

```
> BRK -ED
```

8) TO SUPPRESS TRAPPING ON THE OCCURRENCE OF THE BREAKPOINT AT THE EXIT OF THE CURRENT PROCEDURE BUT RETAIN THE BREAKPOINT AND ASSOCIATED ATTRIBUTES:

```
> BRK \EXIT -IGNORE
```

11.5 SETTING TRACEPOINTS - THE "TRACEPOINT" COMMAND

THE "TRACEPOINT" COMMAND (ABBREVIATED "TRA") IS USED TO:

- SET A NEW TRACEPOINT,
- MODIFY THE ATTRIBUTES OF AN EXISTING TRACEPOINT, AND
- CONVERT A BREAKPOINT TO A TRACEPOINT.

THE COMMAND SYNTAX AND SEMANTICS ARE SIMILAR TO THOSE OF THE "BREAKPOINT" COMMAND, EXCEPT THAT NO ACTION LIST MAY BE ATTACHED TO A TRACEPOINT. THE ACTION LIST ARGUMENT AND "-EDIT" OPTION ARE DISALLOWED IN A "TRACEPOINT" COMMAND.

THE GENERIC FORM OF A "TRACEPOINT" COMMAND IS:

```
TRACEPOINT  [<BREAKPOINT-IDENTIFIER>]  [-AFTER <VALUE>]
            [-BEFORE <VALUE>]  [-EVERY <VALUE>]  [-COUNT <VALUE>]
            [-IGNORE < -NIGNORE]
```

SEE THE DESCRIPTION OF THE "BREAKPOINT" COMMAND IN THE PRECEDING SECTION FOR THE INTERPRETATION OF "TRACEPOINT" COMMAND ARGUMENTS.

EXAMPLES:

1) TO SET A TRACEPOINT ON THE ENTRY TO PROCEDURE "UPDATEFILES":

```
> TRACEPOINT UPDATEFILES\\ENTRY
```

2) TO SET A TRACEPOINT ON FORTRAN STATEMENT LABEL 105 IN THE CURRENT EVALUATION ENVIRONMENT:

```
> TRA $105
```

11.6 CLEARING A BREAKPOINT - THE "CLEAR" COMMAND

THE "CLEAR" COMMAND (ABBREVIATED "CLR") IS USED TO CLEAR A BREAKPOINT.

THE GENERIC FORMAT FOR THE "CLEAR" COMMAND IS:

```
CLEAR  [<BREAKPOINT-IDENTIFIER>]
```

WHERE:

<BREAKPOINT-IDENTIFIER> IS A BREAKPOINT IDENTIFIER, AS DESCRIBED IN SECTION 11.1.

SHOULD THE BREAKPOINT IDENTIFIER BE OMITTED, THE VALUE OF THE EXECUTION ENVIRONMENT POINTER IS USED.

EXAMPLES:

1) TO CLEAR A BREAKPOINT AT THE STATEMENT ON SOURCE LINE 162 IN THE CURRENT EVALUATION ENVIRONMENT:

```
> CLEAR 162
```

2) TO CLEAR THE BREAKPOINTS AT THE ENTRY, EXIT, AND ALL ALTERNATE ENTRIES TO PROCEDURE "ADDRCOMP":

```
> CLR ADDRCOMP\
```

3) TO CLEAR THE BREAKPOINT AT THE LOCATION SPECIFIED BY THE CURRENT EXECUTION POINTER:

```
> CLEAR
```

11.7 CLEARING ALL BREAKPOINTS - THE "CLEARALL" COMMAND

THE "CLEARALL" COMMAND (ABBREVIATED "CLRA") IS USED TO CLEAR EITHER ALL BREAKPOINTS IN THE DEBUGGING ENVIRONMENT OR ALL BREAKPOINTS IN A SPECIFIED PROGRAM BLOCK.

THE GENERIC FORMAT FOR THE "CLEARALL" COMMAND IS:

```
CLEARALL  [<PROGRAM-BLOCK-NAME>  [-DESCEND]]
          [-BREAKPOINTS  <  -TRACEPOINTS]
```

WHERE:

<PROGRAM-BLOCK-NAME> IS THE NAME OF A PROGRAM BLOCK, AS DESCRIBED IN SECTION 9.1.

SHOULD NO ARGUMENTS BE SUPPLIED, ALL BREAKPOINTS AND TRACEPOINTS ARE CLEARED.

SHOULD THE "-BREAKPOINTS" ("-BRK") OR "-TRACEPOINTS" ("-TRA") OPTION BE SPECIFIED, ONLY BREAKPOINTS OR TRACEPOINTS ARE CLEARED. THESE TWO OPTIONS ARE MUTUALLY EXCLUSIVE.

IF A PROGRAM BLOCK NAME IS SUPPLIED, BREAKPOINTS AND/OR TRACEPOINTS IN THE SPECIFIED BLOCK ONLY ARE CLEARED. FURTHERMORE, IF "-DESCEND" ("-DSC") IS SPECIFIED, BREAKPOINTS AND/OR TRACEPOINTS IN THE SPECIFIED BLOCK AND ALL DESCENDANTS (CONTAINED BLOCKS) ARE CLEARED. THE "-DESCEND" OPTION IS USEFUL ONLY FOR PL/1 PROGRAMS.

EXAMPLES:

1) TO CLEAR ALL BREAKPOINTS AND TRACEPOINTS:

```
> CLEARALL
```

2) TO CLEAR ALL TRACEPOINTS, LEAVING BREAKPOINTS UNDISTURBED:

```
> CLRA -TRA
```

3) TO CLEAR ALL BREAKPOINTS AND TRACEPOINTS IN SUBROUTINE "PURGEC":

```
> CLRA PURGEC
```

4) TO CLEAR ALL BREAKPOINTS IN PL/1 PROCEDURE "ADDLINK" AND ALL CONTAINED PROCEDURES:

```
> CLRA_ADDLINK_-DSC_-BRK
```

11.8 DISPLAYING A BREAKPOINT - THE "LIST" COMMAND

THE "LIST" COMMAND IS USED TO PRINT ATTRIBUTES PERTAINING TO ONE BREAKPOINT OR TRACEPOINT.

THE GENERIC FORMAT AND SEMANTICS OF THE "LIST" COMMAND ARE IDENTICAL TO THOSE OF THE "CLEAR" COMMAND, NAMELY:

```
LIST  [<BREAKPOINT-IDENTIFIER>]
```

WHERE:

<BREAKPOINT-IDENTIFIER> IS A BREAKPOINT IDENTIFIER, AS DESCRIBED IN SECTION 11.1.

IF THE BREAKPOINT IDENTIFIER IS OMITTED, THE ATTRIBUTES OF THE BREAKPOINT OR TRACEPOINT CORRESPONDING TO THE EXECUTION ENVIRONMENT POINTER ARE PRINTED.

EXAMPLES:

1) TO LIST THE ATTRIBUTES FOR THE BREAKPOINT AT THE LOCATION SPECIFIED BY THE EXECUTION ENVIRONMENT POINTER:

```
> LIST
TYPE  LOCATION
BRK   $MAIN\4, COUNT = 0
```

2) TO LIST THE ATTRIBUTES OF THE BREAKPOINT AT ALTERNATE ENTRY "COMPUTE1" TO PROCEDURE "COMPUTE":

```
> LIST_COMPUTE\COMPUTE1\\ALT
TYPE  LOCATION
BRK   ALTERNATE ENTRY COMPUTE1 TO COMPUTE, COUNT = 0
```

3) TO LIST THE ATTRIBUTES OF THE ENTRY AND EXIT TRACEPOINTS TO "FUDGEFACTOR":

```
> LIST_FUDGEFACTOR\
TYPE  LOCATION
TRA   ENTRY TO FUDGEFACTOR, COUNT = 3, EVERY 3
TRA   EXIT FROM FUDGEFACTOR, COUNT = 2, EVERY 3
```

11.9 DISPLAYING ALL BREAKPOINTS - THE "LISTALL" COMMAND

THE "LISTALL" COMMAND (ABBREVIATED "LSTA") IS USED TO PRODUCE A LIST OF ALL BREAKPOINTS AND TRACEPOINTS SET. OPTIONALLY, THIS LIST MAY BE RESTRICTED TO ONE PROGRAM BLOCK.

THE GENERIC FORMAT AND SEMANTICS OF THE "LISTALL" COMMAND ARE IDENTICAL TO THOSE OF THE "CLEARALL" COMMAND, NAMELY:

```
LISTALL  [<PROGRAM-BLOCK-NAME>  [-DESCEND]]
         [-BREAKPOINTS  <  -TRACEPOINTS]
```

WHERE:

<PROGRAM-BLOCK-NAME> IS THE NAME OF A PROGRAM BLOCK, AS DESCRIBED IN SECTION 9.1.

IF NO ARGUMENTS ARE SUPPLIED, THE DEBUGGER PRINTS A LIST OF ALL BREAKPOINTS AND TRACEPOINTS WHICH HAVE BEEN SET BY THE USER.

SHOULD THE "-BREAKPOINTS" ("-BRK") OR "-TRACEPOINTS" ("-TRA") OPTION BE SPECIFIED, ONLY BREAKPOINTS OR TRACEPOINTS ARE PRINTED. THESE TWO OPTIONS ARE MUTUALLY EXCLUSIVE.

IF A PROGRAM BLOCK NAME IS SUPPLIED, ONLY BREAKPOINTS AND/OR TRACEPOINTS WITHIN THE SPECIFIED BLOCK ARE LISTED. FURTHERMORE, IF "-DESCEND" ("-DSC") IS SPECIFIED, BREAKPOINTS AND/OR TRACEPOINTS IN THE SPECIFIED BLOCK AND ALL DESCENDANTS ARE INCLUDED IN THE LIST.

EXAMPLES:

1) TO LIST ALL BREAKPOINTS AND TRACEPOINTS WHICH HAVE BEEN SET:

```
> LISTALL
TYPE   LOCATION
BRK    ENTRY TO $MAIN, COUNT = 1
BRK    $MAIN\120+1, COUNT = 0
       [IF I.GT.1000 [WHERE] ELSE [CONTINUE]]
BRK    $MAIN\134, COUNT = 0, AFTER 2
BRK    ADDITM\12, COUNT = 1
TRA    ENTRY TO AUDIT, COUNT = 1
```

2) TO LIST ALL TRACEPOINTS SET IN PROCEDURE "FATALERROR":

```
> LISTALL FATALERROR -TRA
TRACEPOINTS AT:
       ENTRY TO FATALERROR, COUNT = 0
       FATALERROR\4, COUNT = 0
```

3) TO LIST ALL BREAKPOINTS AND TRACEPOINTS SET IN THE FORTRAN MAIN PROGRAM:

```
> LSTA $MAIN
TYPE   LOCATION
BRK    $MAIN\35, COUNT = 0
BRK    $MAIN\35+1, COUNT = 0
```

12 EXPRESSION EVALUATION

THE DEBUGGER PROVIDES FOR THE USER THE CAPABILITY TO INSPECT OR MODIFY THE VALUES OF VARIABLES DEFINED BY THE PROGRAM. IN ADDITION, THE USER MAY EVALUATE EXPRESSIONS PERMITTED BY THE SOURCE LANGUAGE.

12.1 THE EVALUATE COMMAND

THE DEBUGGER COMMAND TO EVALUATE AN EXPRESSION IS A COLON (":"), OPTIONALLY FOLLOWED BY MODIFIERS SPECIFYING THE LANGUAGE OF EVALUATION ("LANGUAGE NAME") AND THE "PRINT MODE", I.E., THE FORMAT IN WHICH THE RESULT WILL BE PRINTED. THE GENERAL FORM OF THE COMMAND IS:

:[<MODIFIER-1> [, <MODIFIER-2>]] <EXPRESSION>

WHERE <MODIFIER-N> IS ONE OF THE FOLLOWING:

<LANGUAGE-NAME> ::= PL1 < FORTRAN < COBOL

<PRINT-MODE> ::= ASCII < BIT < DECIMAL <
EBCDIC < FLOAT < HEX < OCTAL

SEE SECTION 12.2 FOR MORE INFORMATION ON THE LANGUAGE OF EVALUATION. A DESCRIPTION OF THE INDIVIDUAL PRINT MODES MAY BE FOUND IN SECTION 12.3.

IF THE LANGUAGE NAME AND/OR PRINT MODE ARE SPECIFIED, THEY MUST IMMEDIATELY FOLLOW THE COLON, SEPARATED BY A COMMA IF BOTH ARE SUPPLIED. THE EXPRESSION MUST BE PRECEDED BY A SPACE.

THE LANGUAGE IN WHICH AN EXPRESSION IS EVALUATED IS DERIVED TO BE:

- THE LANGUAGE SPECIFIED IN THE CURRENT EVALUATE COMMAND BY <LANGUAGE-NAME>; IF NONE IS SPECIFIED THEN
- THE LANGUAGE EXPLICITLY SET BY THE USER VIA THE "LANGUAGE" COMMAND (SEE SECTION 12.2); IF THE LANGUAGE HAS NOT BEEN SET, THEN
- THE LANGUAGE OF THE PROGRAM BLOCK REPRESENTED BY THE EVALUATION ENVIRONMENT POINTER.

THE RESULT OF AN EVALUATION IS PRINTED IN A MODE WHICH IS DETERMINED AS FOLLOWS:

- THE MODE SPECIFIED IN THE CURRENT COMMAND BY <PRINT-MODE>; IF NONE IS SPECIFIED THEN
- THE PRINT MODE EXPLICITLY SET FOR THIS VARIABLE USING THE "PMODE" COMMAND (SEE SECTION 12.3); IF NO EXPLICIT PRINT MODE HAS BEEN DEFINED, THEN
- THE (DEFAULT) PRINT MODE WHICH CORRESPONDS TO THE DECLARED TYPE OF THE VARIABLE OR THE RESULTANT TYPE OF THE EXPRESSION.

EXAMPLES:

```
> : KEYS
KEYS = 14577

> :PL1, BIT KEYS
KEYS = 0011100011110001 (B)

> :OCTAL SUBR\ADDRESS
ADDRESS = 64017 120270 (O)

> :FORTRAN ITEMNO
ITEMNO = 1374
```

12.1.1 VARIABLES

THE USER MAY ACCESS ANY VARIABLE AS IT IS REFERENCED IN THE SOURCE LANGUAGE. THIS INCLUDES SCALAR VARIABLES, ARRAYS, ARRAY ELEMENTS, AND ARRAY CROSS-SECTIONS (SEE BELOW). IN PL/1, THE FOLLOWING ARE ALSO AVAILABLE: STRUCTURES AND STRUCTURE MEMBERS (FULLY QUALIFIED, PARTIALLY QUALIFIED, OR UNQUALIFIED), AND LOCATOR QUALIFIED REFERENCES.

THE RULES FOR IDENTIFYING A VARIABLE TO THE DEBUGGER ARE FULLY DESCRIBED IN SECTION 9.4. AN AUTOMATIC VARIABLE MAY BE REFERENCED ONLY WHEN ITS CONTAINING BLOCK IS ACTIVE.

ARRAY REFERENCES:

IT IS POSSIBLE TO REFERENCE A PORTION OF AN ARRAY BY SPECIFYING A STAR EXTENT (USING PL/1 NOTATION), OR A "BOUND PAIR" WHICH DESIGNATES THE RANGE OF A GIVEN DIMENSION USING A SPECIAL DBG NOTATION ("...").

SUBSTITUTING A STAR ("*") FOR A SUBSCRIPT INDICATES THAT THE FULL RANGE OF THAT DIMENSION WILL BE DISPLAYED OR OPERATED UPON. FOR EXAMPLE, TO REFERENCE THE ONE-DIMENSIONAL ARRAY FORMED BY THE ELEMENTS (1, 2), (2, 2), (3, 2) OF THE 3 X 3 ARRAY TICTACTOE, ENTER:

```
> :_TICTACTOE_(*,_2)
TICTACTOE(1) = 'X'
TICTACTOE(2) = 'X'
TICTACTOE(3) = 'O'
```

IT IS ALSO POSSIBLE TO LIMIT THE RANGE OF A DIMENSION BY SPECIFYING A BOUND PAIR OF THE FORM:

```
<LOWER-BOUND> ... <UPPER-BOUND>
```

WHERE <LOWER-BOUND> AND <UPPER-BOUND> MAY BE ANY VALID EXPRESSIONS WHICH REDUCE TO INTEGER VALUES. FOR EXAMPLE, GIVEN A THREE-DIMENSIONAL ARRAY, "CALENDAR", DECLARED AS "CALENDAR (12, 31, 1950:2000) CHARACTER (10) VARYING", THE FOLLOWING MAY BE ENTERED:

```
> :_CALENDAR_(DECEMBER,_25,_1979...1985)
CALENDAR(1979) = 'TUESDAY'
CALENDAR(1980) = 'THURSDAY'
CALENDAR(1981) = 'FRIDAY'
CALENDAR(1982) = 'SATURDAY'
CALENDAR(1983) = 'SUNDAY'
CALENDAR(1984) = 'TUESDAY'
CALENDAR(1985) = 'WEDNESDAY'
```

NOTE THAT THE RESULTANT TYPE OF AN ARRAY CROSS-SECTION IS AN ARRAY WHOSE NUMBER OF DIMENSIONS IS EQUAL TO THE NUMBER OF SUBSCRIPTS WITH EITHER A STAR OR A BOUND PAIR. THE RANGE OF SUCH A DIMENSION IS THE DECLARED RANGE IF THE SUBSCRIPT IS A STAR, OR THE SPECIFIED RANGE IF THE SUBSCRIPT IS A BOUND PAIR. FROM THE EXAMPLE ABOVE:

```
> TYPE_CALENDAR_(DECEMBER,_25,_1979...1985)
CHARACTER(10) VARYING AUTOMATIC  1 DIMENSIONAL ARRAY: (1979:1985)
```

12.1.2 EXPRESSIONS

THE DEBUGGER CAN EVALUATE ANY EXPRESSION PERMITTED BY THE SOURCE LANGUAGE. THIS INCLUDES THE ABILITY TO EVALUATE USER FUNCTIONS (WITH THE EXCEPTION OF FORTRAN STATEMENT FUNCTIONS), BUILTIN FUNCTIONS (SEE SECTION 12.1.3), PL/1 REPLACED SYMBOLS, FORTRAN "PARAMETERS", AND DEBUGGER-DEFINED VARIABLES (SEE SECTION 12.1.4).

EXAMPLES OF EXPRESSIONS:

```
> : BALANCE - (CHECKTOTAL + SVCCHARGE) + DEPOSITS
597.98

> : SUBTOT + TAXFNC (SUBTOT, TAXRAT)
153.2895

> : A .GT. 5
.TRUE.

> : LOC (GAMMA)
4002(0)/32

> : 'HELLO' // ' THERE'
'HELLO THERE'

> : NEXT (5) -> STUDENT
STUDENT.ID = '392486512'
STUDENT.NAME.LAST = 'JONES'
STUDENT.NAME.FIRST = 'WILLIAM'
STUDENT.NAME.MIDDLE = 'S'
STUDENT.COURSE = 'CS579'
STUDENT.GRADE = 'B+'
```

THE DEBUGGER SUPPORTS COMPARISON OF AGGREGATES IN PL/1 EVALUATION MODE. AS IN PL/1, THE RESULT OF COMPARING AN AGGREGATE VARIABLE TO A SCALAR OR TO ANOTHER AGGREGATE IS A BIT(1) R-VALUE OF THE SAME SHAPE AS THE OPERAND WITH THE GREATER NUMBER OF DIMENSIONS. FOR EXAMPLE, COMPARISON OF TWO ARRAYS PRODUCES AN ARRAY OF BIT(1) ELEMENTS, AND COMPARISON OF TWO STRUCTURES YIELDS A STRUCTURE OF THE SAME SHAPE WITH BIT(1) ELEMENTS. EACH MEMBER OF THE RESULTING VALUE INDICATES THE RESULT OF THE COMPARISON OF THE CORRESPONDING ELEMENTS.

THE NEXT TWO EXAMPLES USE THE FOLLOWING ARRAYS:

```
COUNTS(1) = 6          LIMITS(1) = 10
COUNTS(2) = 3          LIMITS(2) = 5
COUNTS(3) = 6          LIMITS(3) = 5
COUNTS(4) = 11         LIMITS(4) = 12
COUNTS(5) = 2          LIMITS(5) = 2
```

```
> : COUNTS <= LIMITS
(1) = '1'B
(2) = '1'B
(3) = '0'B
(4) = '1'B
(5) = '1'B
```

```
> : COUNTS = 6
(1) = '1'B
(2) = '0'B
(3) = '1'B
(4) = '0'B
(5) = '0'B
```

12.1.3 BUILTIN FUNCTIONS

THE FOLLOWING BUILTIN FUNCTIONS ARE SUPPORTED BY THE DEBUGGER:

PL/1:	ADDR	SUBSTR	BEFORE	TRANSLATE
	ADDREL	LENGTH	SEARCH	COPY
	PTR	REVERSE	INDEX	TIME
	NULL	AFTER	VERIFY	DATE
FORTRAN:	LOC	COS	CONJG	INT
	ABS	DCOS	ANINT	INTL
	IABS	CCOS	DNINT	INTS
	DABS	ATAN	NINT	REAL
	CABS	DATAN	IDNINT	DBLE
	AIMAG	ATAN2	MAX	CMPLX
	AINT	DATAN2	MAX0	IFIX
	DINT	TANH	AMAX1	IDINT
	LOG	SQRT	DMAX1	FLOAT
	ALOG	DSQRT	AMAX0	SNGL
	DLOG	CSQRT	MAX1	LS
	CLOG	MOD	MIN	LT
	LOG10	AMOD	MIN0	RS
	ALOG10	DMOD	AMIN1	RT
	DLOG10	SIGN	DMIN1	AND
	EXP	ISIGN	AMIN0	OR
	DEXP	DSIGN	MIN1	XOR
	CEXP	DIM	LGE	NOT
	SIN	IDIM	LGT	SHFT
	DSIN	DDIM	LLE	IRND
	CSIN	DPROD	LLT	RND

12.1.4 DEBUGGER-DEFINED VARIABLES

THE DEBUGGER DEFINES SEVERAL VARIABLES AT INITIALIZATION TIME WHICH MAY BE ACCESSED BY THE USER. THESE VARIABLES, WHICH ARE INTERNAL TO THE DEBUGGER, MAY BE EVALUATED OR MODIFIED AS ANY OTHER VARIABLE.

MACHINE REGISTERS

THE VALUES OF THE MACHINE REGISTERS ARE SAVED EACH TIME THE DEBUGGER IS RE-ENTERED FROM THE USER PROGRAM, AND STORED IN A STRUCTURE CALLED "$MR". CONTAINED IN THIS STRUCTURE ARE THE

FOLLOWING SUB-STRUCTURES:

SAVE_MASK	BIT STRING INDICATING WHICH REGISTERS HAVE BEEN SAVED
V	V-MODE REGISTERS (A, B, L, X, Y, E)
I	I-MODE REGISTERS (GENERAL REGISTERS 0 THROUGH 7)
BR	BASE REGISTERS (PB, SB, LB, XB)
KEYS	PROCESS KEYS

FOR EXAMPLE, TO DISPLAY THE CURRENT SAVED MACHINE STATE, ENTER:

```
> : $MR
$MR.SAVE_MASK = '111111111111'B
$MR.V.A = 3
$MR.V.B = 0
$MR.V.L = 196608
$MR.V.X = 17975
$MR.V.Y = 17424
$MR.V.E = 14
$MR.I.GR0 = 1679948336
$MR.I.GR1 = 0
$MR.I.GR2 = 196608
$MR.I.GR3 = 14
$MR.I.GR4 = 14
$MR.I.GR5 = 1141922692
$MR.I.GR6 = 0
$MR.I.GR7 = 1394231
$MR.BR.PB = 4001(3)/1046
$MR.BR.SB = 4037(3)/120244
$MR.BR.LB = 4002(0)/177404
$MR.BR.XB = 4037(3)/6734
$MR.BR.KEYS = '0001100000000000'B
```

BREAKPOINT_COUNTER

THE VALUE OF THE BREAKPOINT COUNTER OF THE MOST RECENT BREAKPOINT OR TRACEPOINT IS STORED IN THE DEBUGGER-DEFINED VARIABLE, "$COUNT". SEE SECTION 11.2 FOR INFORMATION ON THE BREAKPOINT COUNTER. THIS VARIABLE IS MOST USEFUL IN "IF" EXPRESSIONS IN CONDITIONAL BREAKPOINT ACTION LISTS. SEE SECTION 11.3 FOR DETAILS ON ACTION LISTS, SECTION 13.3 FOR INFORMATION ON THE "IF" COMMAND.

EXAMPLE:

```
> BRK A\24 [IF $COUNT > 6 [GOTO 26] ELSE [CONTINUE]]
```

INITIALIZATION_COUNTERS

THE STRUCTURE "$COUNTERS" CONTAINS COUNTS MADE DURING DEBUGGER INITIALIZATION RELATING TO PROGRAM SIZE AND SYMBOLS. THESE COUNTS ARE VALID ONLY IF THE "-FULL_INITIALIZE" OPTION IS SPECIFIED ON THE DBG COMMAND LINE.

THE INDIVIDUAL COUNTS HAVE THE FOLLOWING MEANINGS:

STATEMENTS	NUMBER OF STATEMENTS IN PROCEDURES COMPILED IN "DEBUG" MODE.
OUTER_BLOCKS	NUMBER OF EXTERNAL PROGRAM BLOCKS COMPILED IN "DEBUG" AND "PRODUCTION" MODES.
TOTAL_BLOCKS	NUMBER OF EXTERNAL AND INTERNAL PROGRAM BLOCKS COMPILED IN "DEBUG" AND "PRODUCTION" MODES.
TOP_LEVEL_SYMBOLS	NUMBER OF DECLARED SYMBOLS, NOT INCLUDING STRUCTURE MEMBERS.
NON_TOP_LEVEL_SYMBOLS	TOTAL NUMBER OF STRUCTURE MEMBERS.
PERMANENT_STORAGE	NUMBER OF WORDS ALLOCATED BY THE DEBUGGER FOR THE USER PROGRAM'S SYMBOL TABLE.
DATA_FILE_SIZE	SIZE IN WORDS OF THE DEBUGGER DATA FILE CONTAINED IN THE PROGRAM'S SEG FILE.

THE COUNTERS "OBJECT_GROUPS", "DISK_READS", AND "ALLOCATE_CALLS" ARE SIGNIFICANT TO DBG IMPLEMENTATION PERSONNEL ONLY.

FOR EXAMPLE, TO DISPLAY THE VALUES CONTAINED IN "$COUNTERS", ENTER:

```
> : $COUNTERS
$COUNTERS.STATEMENTS = 42
$COUNTERS.OUTER_BLOCKS = 1
$COUNTERS.TOTAL_BLOCKS = 2
$COUNTERS.OBJECT_GROUPS = 81
$COUNTERS.DISK_READS = 2
$COUNTERS.TOP_LEVEL_SYMBOLS = 15
$COUNTERS.NON_TOP_LEVEL_SYMBOLS = 10
$COUNTERS.ALLOCATE_CALLS = 96
$COUNTERS.PERMANENT_STORAGE = 1303
$COUNTERS.DATA_FILE_SIZE = 1370
```

12.2 CHANGING THE LANGUAGE OF EVALUATION - THE "LANGUAGE" COMMAND

IN ORDER TO EVALUATE AN EXPRESSION, THE DEBUGGER MUST KNOW THE LANGUAGE IN WHICH IT IS EXPRESSED. EACH TIME DEBUGGER COMMAND LEVEL IS RE-ENTERED FROM THE USER PROGRAM, THE DEFAULT LANGUAGE USED FOR EVALUATION IS SET TO CORRESPOND TO THE SOURCE LANGUAGE OF THE PROGRAM BLOCK REPRESENTED BY THE EVALUATION ENVIRONMENT POINTER.

IN ORDER TO SPECIFY THAT SUBSEQUENT EXPRESSIONS SHOULD BE EVALUATED ACCORDING TO THE SYNTAX OF A LANGUAGE OTHER THAN THE CURRENT DEFAULT, THE "LANGUAGE" COMMAND FOLLOWED BY A LANGUAGE NAME MAY BE USED.

THE GENERAL FORMAT OF THE "LANGUAGE" COMMAND (ABBREVIATED "LANG") IS:

LANGUAGE [PL1 < FORTRAN < COBOL]

IF NO ARGUMENT IS PRESENT, THE DEBUGGER PRINTS THE NAME OF THE CURRENT LANGUAGE ON THE USER'S TERMINAL.

THE PRESENCE OF A LANGUAGE NAME MODIFIER IN THE EVALUATE COMMAND WILL OVERRIDE THE LANGUAGE DEFINED USING THE "LANGUAGE" COMMAND (SEE SECTION 12.1).

EXAMPLES:

1) TO DISPLAY THE CURRENT (DEFAULT) LANGUAGE:

```
> LANGUAGE
LANGUAGE IS PL/1.
```

2) TO CHANGE THE EVALUATION LANGUAGE TO FORTRAN:

```
> LANGUAGE FORTRAN
```

12.3 SETTING THE PRINT MODE OF A VARIABLE -- THE "PMODE" COMMAND

EACH USER VARIABLE HAS ASSOCIATED WITH IT A "PRINT MODE" WHICH SPECIFIES THE FORMAT IN WHICH THE VALUE OF THE VARIABLE WILL BE PRINTED UPON EVALUATION. INITIALLY, THE DEFAULT PRINT MODE FOR EACH VARIABLE CORRESPONDS TO ITS DECLARED TYPE. THE USER MAY OVERRIDE THIS DEFAULT IN TWO WAYS:

- THE FORMAT OF THE RESULT OF AN EVALUATION MAY BE SPECIFIED BY DESIGNATING A PRINT MODE AT THE TIME OF THE EVALUATE COMMAND (SEE SECTION 12.1).

- THE PRINT MODE OF A VARIABLE MAY BE SET USING THE "PMODE" COMMAND. THIS PRINT MODE EXPLICITLY SET BY THE USER WILL BE USED THEREAFTER WHENEVER THAT VARIABLE IS EVALUATED, UNLESS OVERRIDDEN BY THE PRESENCE OF A PRINT MODE MODIFIER IN THE EVALUATE COMMAND.

THE GENERAL FORMAT OF THE "PMODE" COMMAND (ABBREVIATED "PM") IS:

PMODE <PRINT-MODE> <VARIABLE-1> [, <VARIABLE-2> ...]

WHERE:

<PRINT-MODE> IS: ASCII < BIT < DECIMAL < EBCDIC < FLOAT < HEX < OCTAL < DEFAULT

<VARIABLE-N> IS A VARIABLE NAME AS DESCRIBED IN SECTION 9.4.

VIEWING THE RESULT OF AN EVALUATION AS A STRING OF BITS, THE INDIVIDUAL PRINT MODE MODIFIERS CAUSE THIS RESULT TO BE INTERPRETED AS FOLLOWS:

ASCII	EACH GROUP OF EIGHT BITS IS PRINTED AS AN ASCII CHARACTER.
BIT	EACH BIT IS PRINTED AS A BINARY DIGIT.
DECIMAL	EACH GROUP OF 16 BITS IS PRINTED AS A SIGNED SINGLE-PRECISION DECIMAL NUMBER.
EBCDIC	NOT IMPLEMENTED AT THIS RELEASE.
FLOAT	EACH GROUP OF 32 BITS IS PRINTED AS A SINGLE-PPECISION FLOATING-POINT NUMBER.
HEX	EACH GROUP OF FOUR BITS IS PRINTED AS A HEXADECIMAL DIGIT.
OCTAL	EACH GROUP OF 16 BITS IS PRINTED AS AN UNSIGNED OCTAL NUMBER.
DEFAULT	THE PRINT MODE IS SET BACK TO THE DEFAULT MODE CORRESPONDING TO THE DECLARED TYPE OF THE VARIABLE.

IF A SPECIFIED VARIABLE IS A PL/1 STRUCTURE, THE PRINT MODE OF EACH MEMBER OF THAT STRUCTURE WILL BE SET.

EXAMPLES:

1) GIVEN THE VARIABLE "OPCODE" DECLARED AS "INTEGER*2", THE RESULT OF AN EVALUATION WOULD NORMALLY BE PRINTED IN DECIMAL:

```
> : OPCODE
OPCODE = 2471
```

2) TO INDICATE THAT "OPCODE" SHOULD BE PRINTED IN OCTAL, ENTER:

```
> PMODE_OCTAL_OPCODE
> :_OPCODE
OPCODE = 4647 (O)
```

3) TO RETURN TO THE DEFAULT PRINT MODE (DECIMAL), ENTER:

```
> PMODE_DEFAULT_OPCODE
```

4) TO SET THE PRINT MODES OF THE VARIABLES "OPCODE", "ADDR" (CONTAINED IN BLOCK "TRANSL"), AND "INFO" TO "BIT":

```
> PMODE_BIT_OPCODE,_TRANSL\ADDR,_INFO
```

12.4 THE "TYPE" COMMAND

THE DATA TYPE AND OTHER ATTRIBUTES OF A VARIABLE OR EXPRESSION MAY BE DETERMINED USING THE "TYPE" COMMAND. WHEN THE "TYPE" COMMAND IS GIVEN, THE DEBUGGER EVALUATES THE EXPRESSION WHICH FOLLOWS, THEN PRINTS THE ATTRIBUTES OF THE RESULTANT EXPRESSION.

THE FOLLOWING INFORMATION IS PRINTED, IF APPLICABLE:

- DATA TYPE
- PRECISION
- SCALE FACTOR
- STORAGE CLASS
- ARRAY DIMENSIONS AND BOUNDS

THE FORMAT OF THE "TYPE" COMMAND IS:

```
TYPE  <EXPRESSION>
```

WHERE <EXPRESSION> IS ANY EXPRESSION PERMITTED BY THE SOURCE LANGUAGE.

EXAMPLES:

```
> TYPE_BUFFER
INTEGER*2 COMMON /BUFFER/

> TYPE_TOTAL_*_5
REAL*4
```

```
> TYPE_IAXENC
ENTRY CONSTANT EXTERNAL (REAL*4 FUNCTION)

> TYPE_TITLE
CHARACTER(25) VARYING AUTOMATIC

> TYPE_LINK
POINTER AUTOMATIC  1 DIMENSIONAL ARRAY: (1:50)
```

12.5 MODIFYING THE VALUE OF A VARIABLE - THE "LET" COMMAND

THE "LET" COMMAND PERMITS THE USER TO ASSIGN A NEW VALUE TO ANY VARIABLE DEFINED BY THE PROGRAM.

THE FORMAT OF THE "LET" COMMAND IS:

```
LET <VARIABLE> = <EXPRESSION>
```

WHERE:

<VARIABLE> IS A USER VARIABLE NAME, AS DESCRIBED IN SECTION 9.4.

<EXPRESSION> IS ANY EXPRESSION PERMITTED BY THE SOURCE LANGUAGE WHOSE VALUE IS CONVERTIBLE TO THE DATA TYPE OF <VARIABLE>.

WHEN THE "LET" COMMAND IS GIVEN, THE EXPRESSION ON THE RIGHT SIDE OF THE ASSIGNMENT STATEMENT IS EVALUATED, AND THE RESULTING VALUE IS ASSIGNED TO THE USER VARIABLE DESCRIBED BY <VARIABLE>. TYPE CONVERSIONS ALLOWED BY THE SOURCE LANGUAGE ARE PERFORMED BY THE DEBUGGER BEFORE THE ASSIGNMENT TAKES PLACE. AN ERROR MESSAGE IS PRINTED IF THE CONVERSION REQUESTED IS ILLEGAL.

EXAMPLES:

```
> LET MAXENTRIES = 1000

> LET INDEX = FIRST + FREELIST (I)

> LET FLAG = .FALSE.
```

THE DEBUGGER SUPPORTS ASSIGNMENT TO STRUCTURES AND ARRAYS AS PERMITTED IN PL/1. WITHIN DBG, THE CAPABILITY TO ASSIGN TO ARRAYS HAS BEEN EXTENDED TO FORTRAN EVALUATION MODE. IF THE VARIABLE ON THE LEFT OF THE "LET" ASSIGNMENT STATEMENT IS AN AGGREGATE, THE EXPRESSION ON THE RIGHT MAY BE A SCALAR OR AN AGGREGATE OF "EQUIVALENT" SHAPE.

EXAMPLES:

1) TO ASSIGN EACH ELEMENT OF THE ONE-DIMENSIONAL ARRAY, "LIST", TO THE CORRESPONDING ELEMENT OF THE ARRAY CROSS-SECTION REFERENCED BY "TABLE (10, *)":

```
> LET TABLE (10, *) = LIST
```

2) TO SET EACH ELEMENT OF "TABLE" TO ZERO:

```
> LET TABLE = 0
```

3) TO ASSIGN EACH MEMBER OF THE PL/1 BASED STRUCTURE "STATUS" TO THE CORRESPONDING MEMBER OF THE STRUCTURE "CURRENTSTATUS", WHICH HAS THE SAME SHAPE:

```
> LET CURRENTSTATUS = LOCATOR -> STATUS
```

12.6 THE "ARGUMENTS" COMMAND

THE USER MAY DISPLAY THE VALUES OF ALL ARGUMENTS TO A GIVEN PROGRAM BLOCK USING THE "ARGUMENTS" COMMAND.

WHEN THE "ARGUMENTS" COMMAND IS GIVEN, EACH ARGUMENT OF THE PROGRAM BLOCK IS EVALUATED, AND ITS CURRENT VALUE IS PRINTED. IF THERE ARE NO ARGUMENTS TO THE SPECIFIED BLOCK, THE MESSAGE, "NO ARGUMENTS." IS PRINTED.

THE FORMAT OF THE COMMAND IS:

```
ARGUMENTS  [ <PROGRAM-BLOCK-NAME> [\ <ACTIVATION-NUMBER> ]  <
             <ALTERNATE-ENTRY-IDENTIFIER> ]
```

WHERE:

<PROGRAM-BLOCK-NAME> IS THE NAME OF A SUBROUTINE OR FUNCTION WHICH IS CURRENTLY ACTIVE. IDENTIFICATION OF PROGRAM BLOCKS IS FULLY DESCRIBED IN SECTION 9.1.

<ACTIVATION-NUMBER> IS AN OPTIONAL ACTIVATION NUMBER AS DESCRIBED IN SECTION 9.3.

<ALTERNATE-ENTRY-IDENTIFIER> IDENTIFIES AN ALTERNATE ENTRY TO A PROCEDURE; SEE SECTION 11.1.

AN ACTIVATION NUMBER MAY BE DESIGNATED IN ORDER TO OBTAIN THE ARGUMENTS TO A SPECIFIC ACTIVATION OF A BLOCK, IF MORE THAN ONE ACTIVATION EXISTS. IF THE ACTIVATION NUMBER IS OMITTED, THE MOST RECENT ACTIVATION IS ASSUMED.

IF THE "ARGUMENTS" COMMAND IS GIVEN WITHOUT A BLOCK NAME OR ALTERNATE ENTRY IDENTIFIER, THE ARGUMENTS TO THE PROGRAM BLOCK DESCRIBED BY THE EVALUATION ENVIRONMENT POINTER ARE PRINTED.

IT IS POSSIBLE TO DISPLAY THE ARGUMENTS TO ALL CALLED PROCEDURES DURING ENTRY TRACING BY SPECIFYING "ETRACE ARGS" (SEE SECTION 18.1).

EXAMPLES:

1) TO PRINT THE ARGUMENTS TO SUBROUTINE "SEARCH":

```
> ARGUMENTS SEARCH
GROUP = 12
NAME = 'TAYLOR'
RTN_INDEX = 0
```

2) TO OBTAIN THE ARGUMENTS OF THE BLOCK DESCRIBED BY THE EVALUATION ENVIRONMENT POINTER:

```
> ARGUMENTS
INTRAT = 0.8E-01
BALNC = 315.79
```

3) TO DISPLAY THE ARGUMENTS TO THE ALTERNATE ENTRY, "ACQUIRE" OF THE PROCEDURE "RESOURCE":

```
> ARGS RESOURCE\ACQUIRE\\ALTERNATE
RESOURCE_NO = 31
HOW_MANY = 2
```

13 DEBUGGER CONTROL COMMANDS

THE COMMANDS DESCRIBED IN THIS SECTION ARE USED TO DEFINE THE EVALUATION ENVIRONMENT, CONTROL DBG COMMAND EXECUTION, AND MODIFY THE INTERPRETATION OF SPECIAL CHARACTERS ENTERED ON THE COMMAND LINE.

13.1 SET EVALUATION ENVIRONMENT - THE "ENVIRONMENT" COMMAND

THE "ENVIRONMENT" COMMAND IS USED TO DEFINE THE EVALUATION ENVIRONMENT. AS DESCRIBED IN SECTION 9.2, THE EVALUATION ENVIRONMENT IS THE PROGRAM BLOCK IN WHICH ALL VARIABLES AND STATEMENTS ARE LOOKED UP IN THE ABSENCE OF AN EXPLICIT PROGRAM BLOCK QUALIFIER.

EACH TIME THE USER CHANGES THE EVALUATION ENVIRONMENT, THE PREVIOUS ENVIRONMENT POINTER IS PUSHED ONTO AN INTERNAL STACK KNOWN AS THE "EVALUATION ENVIRONMENT STACK". THE STACK PROVIDES THE USER WITH A SHORT-HAND WAY OF RESETTING THE EVALUATION ENVIRONMENT WITHOUT REENTERING THE NAME OF THE PROGRAM BLOCK AND ACTIVATION. (THIS FEATURE IS MOST USEFUL WHEN DEALING WITH LONG PROGRAM BLOCK NAMES.) THE STACK CONTAINS THE 10 MOST RECENT PROGRAM BLOCKS TO WHICH THE EVALUATION ENVIRONMENT HAS BEEN SET. IT IS FLUSHED WHEN USER PROGRAM EXECUTION IS RESUMED.

WHENEVER THE "ENVIRONMENT" COMMAND SUCCESSFULLY COMPLETES, THE EVALUATION LANGUAGE IS SET TO THE SOURCE LANGUAGE OF THE NEW ENVIRONMENT. IF THIS IS NOT THE SAME AS THE OLD LANGUAGE, A MESSAGE IS PRINTED AT THE USER'S TERMINAL.

THE GENERIC FORM OF THE "ENVIRONMENT" COMMAND (ABBREVIATED "ENV") IS:

```
ENVIRONMENT  [<PROGRAM-BLOCK-NAME>  [ \ <ACTIVATION-NUMBER>]] <
             [-POP]
```

WHERE:

<PROGRAM-BLOCK-NAME> IS THE NAME OF A PROGRAM BLOCK, AS DESCRIBED IN SECTION 9.1.

<ACTIVATION-NUMBER> IS AN ACTIVATION NUMBER, AS AS DESCRIBED IN SECTION 9.3.

IF NO ARGUMENT FOLLOWS THE "ENVIRONMENT" COMMAND, THE NAME OF THE PROGRAM BLOCK WHICH IS THE CURRENT EVALUATION ENVIRONMENT IS PRINTED. IF THERE IS MORE THAN ONE ACTIVATION OF THIS PROCEDURE, THE ACTIVATION NUMBER IS ALSO PRINTED. IF THERE ARE NO ACTIVATIONS OF THE PROCEDURE, THE DEBUGGER SO INDICATES BY PRINTING "(INACTIVE)" FOLLOWING THE PROGRAM BLOCK NAME.

IF A PROGRAM BLOCK NAME AND ACTIVATION NUMBER FOLLOW THE COMMAND, THE EVALUATION ENVIRONMENT POINTER IS SET TO THAT PROGRAM BLOCK AND ACTIVATION NUMBER. IF NO ACTIVATION NUMBER IS SUPPLIED, THE MOST RECENT ACTIVATION OF THE SPECIFIED PROCEDURE IS USED.

IF THE "-POP" ARGUMENT FOLLOWS THE ENVIRONMENT COMMAND, THE EVALUATION ENVIRONMENT IS SET TO THE TOP PROGRAM BLOCK AND ACTIVATION ON THE EVALUATION ENVIRONMENT STACK AND THIS TOP ITEM IS REMOVED FROM THE STACK.

EXAMPLES:

1) TO PRINT THE VALUE OF THE EVALUATION ENVIRONMENT POINTER, ENTER

```
> ENVIRONMENT
CURRENT EVALUATION ENVIRONMENT IS AGGR\2.
```

2) TO SET THE EVALUATION ENVIRONMENT TO SUBROUTINE "GETR", ENTER

```
> ENV_GETR
NEW LANGUAGE IS FORTRAN.
```

3) TO SET THE EVALUATION ENVIRONMENT TO THE PROGRAM BLOCK AT THE TOP OF THE EVALUATION ENVIRONMENT STACK, ENTER

```
> ENV_-POP
ENVIRONMENT IS ALUSIM.
```

13.2 PRINTING THE ENVIRONMENT STACK - THE "ENVLIST" COMMAND

THE "ENVLIST" COMMAND CAUSES THE DEBUGGER TO PRINT THE CURRENT EVALUATION ENVIRONMENT AND THE CONTENTS OF THE EVALUATION ENVIRONMENT STACK. (THE EVALUATION ENVIRONMENT STACK IS DESCRIBED IN THE PREVIOUS SECTION.)

THE FORMAT OF THE "ENVLIST" COMMAND (ABBREVIATED "EL") IS:

```
ENVLIST
```

EXAMPLE:

```
> ENVLIST
CURRENT EVALUATION ENVIRONMENT IS PLOT_LINE.
THE EVALUATION ENVIRONMENT STACK CONTAINS:

    PLOT_VECTOR\2
    PLOT_VECTOR\1
    PLOT_SETUP
```

13.3 CONDITIONAL COMMAND EXECUTION - THE "IF" COMMAND

THE "IF" COMMAND IS USED TO CONDITIONALLY EXECUTE ONE OR MORE DEBUGGER COMMANDS CONTINGENT UPON THE RESULT OF AN EXPRESSION. IT IS USEFUL ONLY IN BREAKPOINT ACTION LISTS.

THE GENERIC FORMAT OF THE "IF" COMMAND IS:

```
IF  <EXPRESSION>  <ACTION-LIST>  [ELSE  <ACTION-LIST>]
```

WHERE:

<EXPRESSION> IS ANY VALID EXPRESSION IN THE DEFAULT EVALUATION LANGUAGE (FORTRAN OR PL/1). THE RESULT OF THIS EXPRESSION MUST BE LOGICAL (TRUE OR FALSE) IF THE EVALUATION LANGUAGE IS FORTRAN; IT MUST BE A BITSTRING OR AN OBJECT CONVERTIBLE TO A BITSTRING IF PL/1. SEE SECTION 12 FOR INFORMATION ON EXPRESSION EVALUATION.

<ACTION-LIST> IS AN ACTION LIST, AS DESCRIBED IN SECTION 11.3.

EXECUTION OF THE "IF" COMMAND CAUSES THE EXPRESSION WHICH FOLLOWS IT TO BE EVALUATED. IF THE RESULTANT VALUE IS TRUE OR A BITSTRING IN WHICH ONE OR MORE BITS ARE 1, THE COMMANDS WITHIN THE (FIRST) ACTION LIST ARE EXECUTED AND THOSE WITHIN THE "ELSE" ACTION LIST (IF PRESENT) ARE IGNORED.

IF THE RESULT OF THE EXPRESSION IS FALSE OR ALL OF THE BITS IN THE BITSTRING ARE 0, THE COMMANDS IN THE FIRST ACTION LIST ARE IGNORED. IF AN "ELSE" ACTION LIST IS PRESENT, THE COMMANDS THEREIN ARE EXECUTED.

EXAMPLES:

1) BREAKPOINT AT THE STATEMENT ON LINE 21 IN FORTRAN SUBROUTINE "UCAL". IF THE VALUE OF VARIABLE "I" EXCEEDS 1000 PRINT THE CURRENT PROGRAM LOCATION AND THE VALUE, ELSE CONTINUE PROGRAM EXECUTION SILENTLY:

```
> BRK UCAL\21 [IF I.GT.1000 [WHERE; : I] ELSE [C]]
```

2) BREAKPOINT AT LABEL "USERABORT" IN PL/1 PROCEDURE "RESPOND". IF THE FIRST 5 CHARACTERS IN THE CHARACTER VARIABLE "INPUT" ARE "ABORT", RETURN TO DEBUGGER COMMAND LEVEL, PRINTING THE CURRENT PROGRAM LOCATION, ELSE CONTINUE PROGRAM EXECUTION.

```
> BRK RESPOND\USERABORT [IF SUBSTR (INPUT,1,5) = 'ABORT'
[WH] ELSE [C]]
```

(NOTE THE USAGE OF THE ESCAPE CHARACTER TO EFFECT LINE CONTINUATION.)

13.4 CONTROLLING ACTION LIST OUTPUT - THE "ACTIONLIST" COMMAND

NORMALLY, NO INDICATION OF ACTION LIST EXECUTION IS PRINTED AT THE USER'S TERMINAL. USING THE "ACTIONLIST" COMMAND, THE USER CAN CAUSE THE DEBUGGER TO PRINT THE COMMANDS CONTAINED WITHIN AN ACTION LIST PRIOR TO ITS EXECUTION.

WHEN AN ACTION LIST IS PRINTED ON THE USER TERMINAL, A NUMBER ENCLOSED WITHIN ANGLE BRACKETS ("<", ">") IS PRINTED AT THE LEFT MARGIN IMMEDIATELY PRECEDING THE ACTION LIST. THIS IS THE "ACTION LIST DEPTH COUNTER" - THE NESTING LEVEL OF THE CURRENT ACTION LIST. THIS COUNTER IS INCREMENTED BY 1 FOR EACH PENDING ACTION LIST IN WHICH 1 OR MORE COMMANDS REMAIN TO BE EXECUTED. IT IS DECREMENTED FOLLOWING THE EXECUTION OF THE LAST COMMAND IN AN ACTION LIST.

THE GENERIC FORM OF THE "ACTIONLIST" COMMAND (ABBREVIATED "AL") IS:

```
ACTIONLIST ;SUPPRESS < PRINT=
```

IF "SUPPRESS" IS SPECIFIED, NO INFORMATION IS PRINTED AT THE USER TERMINAL WHEN AN ACTION LIST IS EXECUTED.

IF "PRINT" IS SPECIFIED, THE COMMANDS CONTAINED WITHIN ALL ACTION LISTS ARE PRINTED PRIOR TO EXECUTION.

EXAMPLES:

1) TO SUPPRESS ACTION LIST OUTPUT, ENTER

```
> ACTIONLIST SUPPRESS
```

2) TO CAUSE ACTION LISTS TO BE PRINTED PRIOR TO EXECUTION, ENTER

```
> AL P
```

13.5 PRINT SPECIAL SYMBOLS - THE "PSYMBOL" COMMAND

THE "PSYMBOL" COMMAND CAUSES THE DEBUGGER TO PRINT A TABLE CONTAINING THE NAMES OF THE SPECIAL SYMBOLS AND THEIR CURRENT CHARACTER VALUES.

FOLLOWING IS A LIST OF SYMBOLS WHICH ARE CONSIDERED "SPECIAL" BY DBG AND WHOSE CHARACTER VALUES MAY BE MODIFIED USING THE "SYMBOL" COMMAND (DESCRIBED BELOW). THEIR FUNCTIONS ARE DECRIBED IN SECTION 8.

A) ERASE - ERASE SINGLE CHARACTER
B) KILL - IGNORE ALL INFORMATION TYPED THUSFAR ON COMMAND LINE
C) ESCAPE - GENERAL ESCAPE CHARACTER
D) SEPARATOR - COMMAND SEPARATOR
F) WILD - "SOURCE" COMMAND (ED) WILDCARD FOR FIND AND LOCATE OPERATIONS
F) BLANKS - "SOURCE" COMMAND (ED) MATCH ANY NUMBER OF BLANKS

THE FORMAT OF A "PSYMBOL" COMMAND (ABBREVIATED "PSYM") IS:

```
PSYMBOL
```

EXAMPLE:

```
> PSYMBOL
ERASE       "
KILL        ?
ESCAPE      ^
SEPARATOR   ;
WILD        !
BLANKS      #
```

13.6 MODIFYING A SYMBOL CHARACTER - THE "SYMBOL" COMMAND

THE "SYMBOL" COMMAND IS USED TO MODIFY THE VALUE OF A DEBUGGER CHARACTER SYMBOL. THE SYMBOLS AND THEIR DEFAULT VALUES ARE DESCRIBED IN THE PRECEDING SECTION.

THE FORMAT OF THE "SYMBOL" COMMAND (ABBREVIATED "SYM") IS:

SYMBOL <SYMBOL-NAME> <CHARACTER-VALUE>

WHERE:

<SYMBOL-NAME> IS THE NAME OF A CHARACTER SYMBOL: "ERASE", "KILL", "ESCAPE", "SEPARATOR", "WILD", "BLANKS".

<CHARACTER-VALUE> IS THE NEW CHARACTER VALUE OF THE SYMBOL. IT MAY NOT BE A SPACE, ALPHANUMERIC, OR IDENTICAL TO AN EXISTING CHARACTER SYMBOL VALUE.

THE "SYMBOL" COMMAND CAUSES THE CHARACTER VALUE OF THE SYMBOL NAME TO BE SET TO THE GIVEN CHARACTER.

EXAMPLES:

1) TO SET THE DEBUGGER COMMAND SEPARATOR TO AMPERSAND ("&"), ENTER

```
> SYMBOL SEPARATOR &
```

ALL FURTHER COMMAND LINES MUST CONTAIN AN AMPERSAND TO SEPARATE COMMANDS RATHER THAN A SEMICOLON (THE DEFAULT).

2) TO SET THE ERASE CHARACTER TO AT-SIGN ("@"), ENTER

```
> SYM ERASE @
```

13.7 RESUBMIT LAST COMMAND LINE - THE "RESUBMIT" COMMAND

THE "RESUBMIT" COMMAND ALLOWS THE USER TO EDIT AND RESUBMIT FOR EXECUTION THE LAST COMMAND LINE ENTERED.

THE DBG COMMAND LINE EDITOR IS INVOKED TO PERFORM THE EDITING. THIS FACILIY IS DESCRIBED IN SECTION 21.

THE FORMAT OF THE "RESUBMIT" COMMAND (ABBREVIATED "RSU") IS:

RESUBMIT

EXAMPLES:

1) ASSUME THAT THE FOLLOWING COMMAND LINE HAS BEEN ENTERED FROM THE TERMINAL:

```
> FTN ALPHA + SIN (THETA) ** 2
UNRECOGNIZED COMMAND (FTN).
PREFIX EXPRESSIONS WITH ': '.
```

INSTEAD OF RETYPING THE LINE, THE USER MAY EDIT AND RESUBMIT IT AS FOLLOWS:

```
> RESUBMIT
   FTN ALPHA + SIN (THETA) ** 2
: I:
   :FTN ALPHA + SIN (THETA) ** 2
:
230.1099
```

2) ASSUME THAT THE FOLLOWING COMMAND LINE HAS BEEN ENTERED:

```
> ENV PROG2; CALL CPOFF(1.,R); : R
ENV PROG2
NO SUCH PROCEDURE.
```

UPON REALIZING THAT HE SHOULD HAVE ENTERED "PR2" INSTEAD OF "PROG2", THE USER TYPES

```
> RSU
   ENV PROG2; CALL CPOFF(1.,R); : R
: _____DD
   ENV PR2; CALL CPOFF(1.,R); : R
:
R = 6.0
```

14 INFORMATION REQUEST COMMANDS

THIS SECTION DESCRIBES THE DEBUGGER COMMANDS WHICH ARE USED TO REQUEST INFORMATION ABOUT THE USER PROGRAM AND DEBUGGER.

14.1 PRINT EXECUTION ENVIRONMENT POINTER - THE "WHERE" COMMAND

THE "WHERE" COMMAND IS USED TO PRINT THE VALUE OF THE EXECUTION ENVIRONMENT POINTER. ALTERNATIVELY, IT MAY BE USED TO FIND THE PROGRAM LOCATION WHICH CORRESPONDS TO A GIVEN SEGMENT AND WORD MEMORY ADDRESS.

THE GENERIC FORM OF THE "WHERE" COMMAND (ABBREVIATED "WH") IS:

WHERE [<SEGMENT-NUMBER> / <WORD-NUMBER>]

WHERE:

<SEGMENT-NUMBER> IS A SEGMENT NUMBER, REPRESENTED IN OCTAL.

<WORD-NUMBER> IS A WORD ADDRESS, REPRESENTED IN OCTAL.

IF NO ADDRESS (SEGMENT AND WORD NUMBER) IS SPECIFIED, THE VALUE OF THE EXECUTION ENVIRONMENT POINTER IS PRINTED ON THE USER'S TERMINAL.

IF A SEGMENT AND WORD NUMBER ARE SPECIFIED, DBG ATTEMPTS TO IDENTIFY THAT ADDRESS IN TERMS OF A PROGRAM LOCATION (PROGRAM BLOCK NAME AND STATEMENT IDENTIFIER). IF SUCCESSFUL, THIS PROGRAM LOCATION IS PRINTED ON THE USER'S TERMINAL, ELSE A DIAGNOSTIC ("UNIDENTIFIABLE ADDRESS") IS PRINTED.

EXAMPLES:

1) TO PRINT THE VALUE OF THE EXECUTION ENVIRONMENT POINTER, ENTER

```
> WHERE
CURRENTLY AT SUBA\34 ($120).
```

2) TO IDENTIFY THE PROGRAM LOCATION WHICH CORRESPONDS TO ADDRESS 4001/1412, ENTER

```
> WH 4001/1412
$MAIN\102
```

14.2 PRINT IN-USE SEGMENTS - THE "SEGMENTS" COMMAND

USING THE "SEGMENTS" COMMAND, THE USER MAY REQUEST THAT THE DEBUGGER PRINT A LIST OF SEGMENTS WHICH ARE IN USE.

THE SEGMENTS ARE CLASSIFIED BY USAGE, AS FOLLOWS:

A) USER PROCEDURE TEXT, LINKAGE TEXT, AND DATA
B) DEBUGGER PROCEDURE TEXT
C) DEBUGGER LINKAGE TEXT, DATA, AND SYMBOL TABLE
D) PROCEDURE CALL/RETURN STACK

THE FORMAT OF THE "SEGMENTS" COMMAND (ABBREVIATED "SEGS") IS:

```
SEGMENTS
```

EXAMPLE:

```
> SEGMENTS
USER PROCEDURE TEXT, LINKAGE TEXT, AND DATA:
   4001, 4002.
DEBUGGER PROCEDURE TEXT:
   2040, 2041, 2042.
DEBUGGER LINKAGE TEXT, DATA, AND SYMBOL TABLE:
   4036, 4037.
STACK:
   4037.
```

14.3 PRINT STATEMENT / BLOCK INFORMATION - THE "INFO" COMMAND

THE "INFO" COMMAND ALLOWS THE USER TO REQUEST THAT INFORMATION ABOUT A PROGRAM BLOCK, ALTERNATE ENTRY TO A PROGRAM BLOCK, OR STATEMENT BE PRINTED AT THE TERMINAL.

FOR PROGRAM BLOCKS, THE FOLLOWING INFORMATION IS PRINTED:

A) PROCEDURE BASE START AND END ADDRESSES
B) LINKFRAME START ADDRESS (IF PROCEDURE-CALLED)
C) ECB ADDRESS (IF PROCEDURE-CALLED)
D) MOST RECENT INVOCATION'S STACK FRAME ADDRESS (IF PROCEDURE-CALLED
F) STARTING AND ENDING SOURCE LINE NUMBERS
F) SOURCE FILE NAME
G) MECHANISM BY WHICH BLOCK IS INVOKED (PROCEDURE- OR SHORT-CALL)
H) COMPILATION MODE (DEBUG OR PRODUCTION)

FOR ALTERNATE ENTRIES, THE FOLLOWING INFORMATION IS PRINTED:

A) ECB ADDRESS (IF PROCEDURE-CALLED)
B) SOURCE LINE NUMBER OF ENTRY STATEMENT
C) SOURCE FILE NAME
D) MECHANISM BY WHICH BLOCK IS INVOKED (PROCEDURE- OR SHORT-CALL)

FOR STATEMENTS, THE MEMORY ADDRESS OF THE FIRST INSTRUCTION IS PRINTED.

THE GENERIC FORM OF THE "INFO" COMMAND IS:

INFO ;<PROGRAM-BLOCK-NAME> \ < <ALTERNATE-ENTRY-IDENTIFIER> < <STATEMENT-IDENTIFIER>=

WHERE:

<PROGRAM-BLOCK-NAME> IS THE NAME IF A PROGRAM BLOCK AS DESCRIBED IN SECTION 9.1.

<ALTERNATE-ENTRY-IDENTIFIER> IDENTIFIES AN ALTERNATE ENTRY TO A PROCEDURE; SEE SECTION 11.1.

<STATEMENT-IDENTIFIER> IS A STATEMENT IDENTIFIER, AS DESCRIBED IN SECTION 9.5.

THE INFORMATION PRINTED FOR EACH TYPE OF ARGUMENT (PROGRAM BLOCK, ALTERNATE ENTRY, AND STATEMENT) IS DESCRIBED ABOVE. NOTE THAT A BACKSLASH MUST FOLLOW A PROGRAM BLOCK NAME TO DISTINGUISH IT FROM A STATEMENT LABEL.

EXAMPLES:

1) TO PRINT THE INFORMATION FOR PROGRAM BLOCK "AINIT", ENTER

```
> INFO_AINIT\
PB START 4001(0)/1000, PB END 4001(0)/1120,
LINKFRAME START 4002(0)/177404,
ECB 4002(0)/11, CURRENTLY NO ACTIVATIONS.
START LINE # 3, END LINE # 47 IN SOURCE FILE "DPR>AINIT".
BLOCK INVOKED VIA PROCEDURE CALL MECHANISM;
COMPILED IN DEBUG MODE.
```

2) TO PRINT THE DEBUGGER INFORMATION REGARDING ALTERNATE ENTRY "ACLIST" TO THE PROGRAM BLOCK DEFINED BY THE EVALUATION ENVIRONMENT POINTER, ENTER

```
> INFO ACLIST\\ALTERNATE
ECB ADDRESS IS 4002(0)/31.
ENTRY DEFINED ON SOURCE LINE 4 IN SOURCE FILE "COMP2".
CALLED VIA PROCEDURE CALL MECHANISM.
```

3) TO PRINT THE ADDRESS OF THE FIRST INSTRUCTION GENERATED FOR THE STATEMENT ON SOURCE LINE 37 IN SUBROUTINE "COMPAD", ENTER

```
> INFO COMPAD\37
FIRST INSTRUCTION AT 4001(0)/1061.
```

14.4 PRINT GENERAL STATUS - THE "STATUS" COMMAND

THE "STATUS" COMMAND CAUSES THE DEBUGGER TO PRINT THE FOLLOWING INFORMATION ON THE USER'S TERMINAL:

A) STATEMENT, ENTRY/EXIT, AND VALUE TRACE STATUS (ON/OFF)
B) EXECUTION ENVIRONMENT
C) EVALUATION ENVIRONMENT
D) EVALUATION LANGUAGE
E) SOURCE FILE NAME AND HOW IT WAS DERIVED (FROM EVALUATION OR EXECUTION ENVIRONMENT, OR EXPLICITLY SET BY USER)
F) ACTION LIST OUTPUT ON BREAKPOINT (PRINT/SUPPRESS)

THE FORMAT OF THE "STATUS" COMMAND IS:

```
STATUS
```

EXAMPLE:

```
> STATUS
TRACING:
    STATEMENT                   OFF
    ENTRY/EXIT                  OFF
    VALUE                       OFF

BREAKPOINT ACTION LIST          SUPPRESS
EXECUTION ENVIRONMENT           A\28
EVALUATION ENVIRONMENT          A
EVALUATION LANGUAGE             PL/1
SOURCE FILE NAME                ST.1 (EVALUATION ENV)
```

15 PROCEDURE CALL / RETURN STACK TRACE

THE DEBUGGER "TRACEBACK" FACILITY ALLOWS THE USER TO VIEW THE PROCEDURE CALL / RETURN STACK ON WHICH HIS PROGRAM RUNS.

15.1 INFORMATION PRODUCED BY "TRACEBACK"

THE "TRACEBACK" COMMAND CAUSES THE CALL/RETURN AND OWNERSHIP INFORMATION CONTAINED IN ALL OR SELECTED STACK FRAMES TO BE LISTED AT THE USER'S TERMINAL IN THE FORMATS DESCRIBED BELOW.

1) USER-OWNED PROCEDURE CALL FRAMES ARE IDENTIFIED BY PROGRAM BLOCK NAME (DERIVED FROM THE PB AT THE TIME OF THE NEXT SUCCESSIVE PROCEDURE CALL OR FAULT). FOR EACH USER-OWNED FRAME, THE FRAME OWNER (NAME OF THE PROGRAM BLOCK WHICH OWNS THE FRAME), CALLED-FROM, AND RETURNS-TO PROGRAM LOCATIONS ARE PRINTED. THE USER MAY REQUEST THAT MEMORY ADDRESSES WHICH CORRESPOND TO THESE PROGRAM LOCATIONS ALSO BE PRINTED. FURTHERMORE, THE USER MAY REQUEST THAT THE NAMES OF THE ON-UNITS DECLARED WITHIN THE FRAMES BE PRINTED.

   AN EXAMPLE OF A USEP-OWNED PROCEDURE CALL FRAME AS IT WOULD APPEAR IN A "TRACEBACK" OUTPUT IS

```
3: OWNER IS "SUBA".
   CALLED FROM $MAIN\43, RETURNS TO $MAIN\45.
```

2) FOR EACH CONDITION FRAME ON THE STACK, THE INFORMATION PRINTED INCLUDES CONDITION NAME, RAISED-FROM PROGRAM LOCATION, RETURNS-TO PROGRAM LOCATION, SIGNAL SOURCE (SOFTWARE OR HARDWARE-RELATED), AND CRAWLOUT INFORMATION, WHERE APPLICABLE.

   AN EXAMPLE OF A CONDITION FRAME AS IT MIGHT APPEAR IN A "TRACEBACK" LISTING IS

```
4: CONDITION FRAME FOR "ARITH$",
   RETURNS TO LOCATION 13(3)/1720.
   CONDITION INITIATED BY HARDWARE FAULT,
   DETECTED AT $MAIN\3.
   CORRECTIVE ACTION BY ON-UNIT IS REQUIRED.
```

3) FOR EACH FAULT FRAME ON THE STACK, THE FAULT TYPE, CODE, AND ADDRESS, AND THE RETURNS-TO PROGRAM LOCATION ARE LISTED.

   A FAULT FRAME WOULD APPEAR LIKE THIS IN A "TRACEBACK" LISTING:

```
3: FAULT FRAME; FAULT TYPE "ARITH" (50)
   FAULT RETURNS TO $MAIN\3.
   FAULT CODE 403, FAULT ADDRESS 0(0)/0.
```

4) DEBUGGER-OWNED FRAMES ARE NORMALLY LISTED IN "COMPRESSED" FORM, MEANING THAT THE TRACEBACK FACILITY WILL INDICATE ONLY THAT A BLOCK OF DEBUGGER-OWNED FRAMES EXIST BETWEEN TWO USER-OWNED FRAMES OR AT THE ROOT OF THE STACK. DEBUGGER-OWNED FRAMES AT THE TOP OF THE STACK ARE NEVER PRINTED. THE USER MAY OPTIONALLY REQUEST THAT THESE DEBUGGER FRAMES BE PRINTED IN "EXPANDED" FORM, LISTING THE LINKBASE ADDRESSES OF EACH FRAME OWNER AND PROCEDURE BASE CALLED-FROM AND RETURNS-TO ADDRESSES. (THIS INFORMATION IS MEANINGLESS FOR MOST USERS.)

   AN EXAMPLE OF COMPRESSED DEBUGGER-OWNED FRAMES AS THEY MIGHT BE LISTED BY "TRACEBACK" IS

```
14: DEBUGGER-OWNED FRAMES, THROUGH FRAME 4.
```

5) DEBUGGER "CALL" FRAMES ARE PRINTED SEPARATELY FROM OTHER DEBUGGER-OWNED FRAMES IF THE USER REQUESTS AN "EXPANDED" LIST OF DEBUGGER FRAMES. "CALL" FRAMES ARE GENERATED IMMEDIATELY PRIOR TO USER-OWNED FRAMES AS A RESULT OF CALLING A USER PROCEDURE FROM DEBUGGER COMMAND LEVEL. THEY ARE SIGNIFICANT TO THE USER BECAUSE THE CALLED PROCEDURE MAY NOT NON-LOCAL GOTO PAST THEM. (ATTEMPTING TO DO SO WOULD YIELD AN ERROR.) DEBUGGER "CALL" FRAMES ARE FULLY DESCRIBED IN SECTION 19.

   A "CALL" FRAME MIGHT LOOK LIKE THIS AS IT APPEARS IN A "TRACEBACK" LISTING:

```
16: DEBUGGER CALL FRAME.
    CALLED FROM DEBUGGER, RETURNS TO DEBUGGER.
```

15.2 THE "TRACEBACK" COMMAND

THE "TRACEBACK" COMMAND (ABBREVIATED "TR") INVOKES THE TRACEBACK FACILITY DESCRIBED ABOVE. ITS OPTIONAL ARGUMENTS SELECT THE FRAMES TO BE PRINTED, THE ORDER IN WHICH THEY ARE PRINTED, AND DETAIL LEVEL OF THE INFORMATION PRODUCED.

THE GENERIC FORMAT OF THE "TRACEBACK" COMMAND IS:

```
TRACEBACK [-FRAMES <VALUE> [-LEAST_RECENT]] [-FROM <VALUE>]
          [-TO <VALUE>] [-REVERSE] [-DBG] [-ONUNITS]
          [-ADDRESSES]
```

WHERE:

<VALUE> IS A POSITIVE NON-ZERO INTEGER VALUE.

SHOULD NO ARGUMENTS BE SUPPLIED, ALL FRAMES ON THE STACK ARE PRINTED IN (CHRONOLOGICALLY) MOST RECENT TO LEAST RECENT ORDER, STARTING WITH THE MOST RECENT USER-OWNED FRAME AND TERMINATING WITH THE FIRST FRAME GENERATED DURING THIS DEBUGGING SESSION. THE DETAIL OF THE INFORMATION IS AS DESCRIBED IN SECTION 15.1.

IF THE "-FRAMES" ARGUMENT IS SUPPLIED, THE NUMBER OF FRAMES WHICH ARE PRINTED IS LIMITED TO THE SPECIFIED VALUE. THESE WILL BE THE MOST RECENT <VALUE> FRAMES, UNLESS THE "-LEAST_RECENT" OPTION IS SPECIFIED, IN WHICH CASE THEY WILL BE THE LEAST RECENT <VALUE> FRAMES.

IF THE "-FROM" ARGUMENT IS PRESENT, THE TRACEBACK STARTS FROM THE FRAME NUMBER WHICH FOLLOWS "-FROM". ALL FRAMES ARE ASSIGNED A NUMBER BY THE DEBUGGER, THE LEAST RECENT BEING FRAME 1 AND THE MOST BEING FRAME <FRAMES-ON-STACK>, THE NUMBER OF FRAMES ON THE STACK. THE FRAME NUMBER FOR EACH FRAME IS PRINTED IN THE LEFT-HAND MARGIN.

THE "-TO" ARGUMENT SPECIFIES THE FRAME NUMBER OF THE LAST FRAME TO BE LISTED BY THE TRACEBACK FACILITY.

IF THE "-REVERSE" ARGUMENT IS SPECIFIED, THE FRAMES ARE LISTED IN LEAST RECENT TO MOST RECENT ORDER. THE OPPOSITE (MOST RECENT TO LEAST RECENT) IS THE DEFAULT.

IF THE "-DBG" ARGUMENT IS SUPPLIED, DEBUGGER FRAMES ARE LISTED IN EXPANDED FORM, AS DESCRIBED IN SECTION 15.1.

IF THE "-ONUNITS" ARGUMENT IS SUPPLIED, THE NAMES OF ALL ON-UNITS AND THEIR CORRESPONDING PROGRAM BLOCKS ARE PRINTED FOR EACH STACKFRAME.

IF THE "-ADDRESSES" ARGUMENT IS PRESENT, THE FOLLOWING ADDITIONAL INFORMATION IS INCLUDED (AS APPLICABLE) FOR EACH FRAME:

STACKFRAME ADDRESS
STACK AND ROOT SEGMENT NUMBERS
LINKBASE ADDRESS OF FRAME OWNER
CALLED-FROM AND RETURNS-TO ADDRESSES
ON-UNIT ECB ADDRESS (IF "-ONUNITS" SPECIFIED)

EXAMPLES:

THE FOLLOWING FORTRAN PROGRAM IS USED IN THE EXAMPLES BELOW:

```
(0001) C     DBG "TRACEBACK" DEMONSTRATION PROGRAM.
(0002)
(0003)       CALL A
(0004)       CALL EXIT
(0005)       END
(0006)
(0007)       SUBROUTINE A
(0008)       INTEGER K
(0009)       CALL B (K)
(0010)       RETURN
(0011)       END
(0012)
(0013)       SUBROUTINE B (I, J)
(0014)       J = I
(0015)       RETURN
(0016)       END
```

1) ASSUME A BREAKPOINT HAS BEEN PLACED AT THE ENTRY TO SUBROUTINE "A". WHEN WE REACH THIS BREAKPOINT A STACK TRACEBACK IS INVOKED AS FOLLOWED:

```
> TRACEBACK
CURRENTLY AT ENTRY TO A.
STACK CONTAINS 3 FRAMES.

  3: OWNER IS "A".
     CALLED FROM $MAIN\3, RETURNS TO $MAIN\4.

  2: OWNER IS "$MAIN".
     CALLED FROM DEBUGGER, RETURNS TO DEBUGGER.

  1: DEBUGGER-OWNED FRAME.
```

2) IF WE WERE INTERESTED IN THE MEMORY ADDRESSES AS WELL AS THE PROGRAM LOCATIONS PRINTED ABOVE, INCLUDING THE "-ADDRESSES" OPTION WOULD PRODUCE THIS OUTPUT:

```
> TRACEBACK -ADDRESSES
CURRENTLY AT ENTRY TO A.
STACK CONTAINS 3 FRAMES.

STACK SEGMENT IS 4037, ROOT SEGMENT IS 4037.

  3, 120270: OWNER IS "A", OWNER LB IS 4002(0)/177424.
     CALLED FROM $MAIN\3 (LOCATION 4001(0)/1001),
     RETURNS TO $MAIN\4 (LOCATION 4001(3)/1003).

  2, 120244: OWNER IS "$MAIN", OWNER LB IS 4002(0)/177400.
     CALLED FROM DEBUGGER (LOCATION 2020(0)/62237),
     RETURNS TO DEBUGGER (LOCATION 2020(3)/62241).

  1, 120012: DEBUGGER-OWNED FRAME.
```

3) CONTINUING PROGRAM EXECUTION, WE FIND THAT A POINTER FAULT ERROR OCCURS IMMEDIATELY FOLLOWING THE ENTRY TO SUBROUTINE "C". TO VIEW THE CONDITION AND FAULT FRAMES, WE WOULD REQUEST THE TOP 2 FRAMES TO BE PRINTED:

```
> TB -FR 2
CURRENTLY AT DEBUGGER DEFAULT ON-UNIT FOR "POINTER_FAULT$";
ON-UNIT RETURNS TO USER PROGRAM AT B\14.
STACK CONTAINS 6 FRAMES.


  6: CONDITION FRAME FOR "POINTER_FAULT$",
     RETURNS TO LOCATION 13(3)/1655.
     CONDITION INITIATED BY HARDWARE FAULT,
     DETECTED AT B\14.
     CORRECTIVE ACTION BY ON-UNIT IS REQUIRED.

  5: FAULT FRAME; FAULT TYPE "POINTER" (64)
     FAULT RETURNS TO B\14.
     FAULT CODE 100000, FAULT ADDRESS 4037(3)/120343.
```

4) TO LIST THE TWO LEAST RECENT FRAMES ON THE STACK AND THEIR ON-UNITS, EXPANDING THE DEBUGGER FRAME, ONE WOULD ENTER:

```
> IB -FR 2 -LR -DBG -ONU
CURRENTLY AT DEBUGGER DEFAULT ON-UNIT FOR "POINTER_FAULT$";
ON-UNIT RETURNS TO USER PROGRAM AT E\14.
STACK CONTAINS 6 FRAMES.

  2: OWNER IS "$MAIN".
     CALLED FROM DEBUGGER, RETURNS TO DEBUGGER.

  1: DEBUGGER-OWNED FRAME.
     CALLED FROM LOCATION 4000(0)/50,
     RETURNS TO LOCATION 4000(3)/52.
     ONUNIT FOR "STACK_OVF$" IS 4000(3)/44120.
     ONUNIT FOR "NONLOCAL_GOTO$" IS 4000(3)/44262.
     ONUNIT FOR "BAD_NONLOCAL_GOTO$" IS 4000(3)/44334.
     ONUNIT FOR "ARITH$" IS 4000(3)/44554.
     ONUNIT FOR "ILLEGAL_INST$" IS 4000(3)/44412.
     ONUNIT FOR "ANY$" IS 4000(3)/44066.
     ONUNIT FOR "QUIT$" IS 4000(3)/44474.
```

16 SINGLE STEP FACILITY

THE DEBUGGER'S SINGLE STEP FACILITY ALLOWS THE USER TO EXECUTE HIS PROGRAM

- ONE OR (N) STATEMENTS AT A TIME, OPTIONALLY STEPPING INTO OR ACROSS CALLS TO OTHER PROCEDURES,
- UNTIL THE NEXT PROCEDURE IS CALLED, OR
- UNTIL THE CURRENT PROCEDURE RETURNS.

16.1 EXECUTE (N) STATEMENTS - THE "STEP" AND "STEPIN" COMMANDS

THE "STEP" AND "STEPIN" COMMANDS ALLOW THE USER TO RESUME PROGRAM EXECUTION FOR A SPECIFIED NUMBER OF STATEMENTS AND THEN RETURN CONTROL TO THE DEBUGGER. THE "STEP" COMMAND CAUSES CALLED PROCEDURES TO BE STEPPED ACROSS, WHILE THE "STEPIN" COMMAND CAUSES CALLED PROCEDURES TO BE STEPPED INTO.

ASSOCIATED WITH THE "STEP" AND "STEPIN" COMMANDS IS A COUNTER INTERNAL TO DBG CALLED THE "STEP COUNTER". THE STEP COUNTER CONTAINS THE NUMBER OF STATEMENTS (LEFT) TO BE EXECUTED PRIOR TO RETURNING CONTROL TO DBG COMMAND MODE. THE USER MAY SPECIFY THE INITIAL VALUE OF THIS COUNTER WHEN HE EXECUTES A "STEP" OR "STEPIN" COMMAND; THE ASSUMED VALUE IF NONE IS SUPPLIED IS 1.

WHEN PROGRAM EXECUTION IS RESUMED, THE STEP COUNTER IS UPDATED AS FOLLOWS:

- IF THE COMMAND GIVEN WAS "STEPIN", THIS COUNTER IS DECREMENTED BY 1 FOLLOWING THE EXECUTION OF EACH STATEMENT IN THE USER PROGRAM; WHEN IT BECOMES 0, DBG RETURNS CONTROL TO COMMAND MODE.
- IF THE COMMAND GIVEN WAS "STEP", THIS COUNTER IS DECREMENTED FOLLOWING THE EXECUTION OF EACH STATEMENT IN THE PROGRAM BLOCK AND ACTIVATION SPECIFIED BY THE EXECUTION ENVIRONMENT POINTER AT THE TIME THE "STEP" COMMAND WAS ISSUED; WHEN IT BECOMES 0, THE DEBUGGER RETURNS CONTROL TO COMMAND MODE.

ADDITIONALLY, SHOULD THE PROGRAM BLOCK SPECIFIED BY THE EXECUTION ENVIRONMENT POINTER AT THE TIME THE "STEP" OR "STEPIN" COMMAND IS GIVEN RETURN, SINGLE STEPPING IS TERMINATED AND CONTROL RETURNS TO DEBUGGER COMMAND MODE.

THE GENERIC FORM OF THE "STEP" COMMAND (ABBREVIATED "S") IS:

```
STEP  [<VALUE>]
```

WHERE:

<VALUE> IS A POSITIVE INTEGER VALUE.

THE FORMAL INTERPRETATION OF THE VALUE WHICH FOLLOWS THE "STEP" COMMAND IS DEPENDENT UPON THE CURRENT VALUE OF THE EXECUTION ENVIRONMENT POINTER:

- IF THE EXECUTION ENVIRONMENT POINTER REPRESENTS A STATEMENT, THIS VALUE IS THE NUMBER OF STATEMENTS WHICH WILL BE EXECUTED.

- IF THE EXECUTION ENVIRONMENT POINTER REPRESENTS AN ENTRY TO A PROGRAM BLOCK, A VALUE OF 1 SIGNIFIES THAT THE PROLOGUE CODE (IF ANY) IS TO BE EXECUTED AND THAT EXECUTION IS TO CEASE PRIOR TO THE FIRST STATEMENT. A VALUE OF 2 INDICATES THAT THE PROLOGUE CODE AND FIRST STATEMENT ARE TO BE EXECUTED, ETC.

- IF THE EXECUTION ENVIRONMENT POINTER REPRESENTS AN EXIT FROM A PROGRAM BLOCK, A VALUE OF 1 SIGNIFIES THAT THE REMAINDER OF THE STATEMENT (IF ANY) WHICH CAUSED ENTRY TO THE JUST-RETURNED PROGRAM BLOCK IS TO BE EXECUTED AND THAT EXECUTION IS TO CEASE PRIOR TO THE NEXT STATEMENT. A VALUE OF 2 INDICATES THAT, IN ADDITION TO THE ABOVE, THE NEXT STATEMENT IS TO BE EXECUTED, ETC.

- IF THE EXECUTION ENVIRONMENT POINTER REPRESENTS THE DEBUGGER DEFAULT ON-UNIT FOR A PARTICULAR CONDITION, A VALUE OF 1 INDICATES THAT THE ON-UNIT IS TO RETURN AND THAT EXECUTION IS TO CEASE PRIOR TO THE NEXT STATEMENT. A VALUE OF 2 INDICATES THAT THE ON-UNIT IS TO RETURN AND THE NEXT STATEMENT IS TO BE EXECUTED, ETC.

- IF THE EXECUTION ENVIRONMENT POINTER IS UNDEFINED, A "RESTART STEP" OR "RESTART STEPIN" COMMAND IS IMPLIED. SEE SECTION 10.1 FOR DETAILS.

IF THE VALUE FOLLOWING THE "STEP" COMMAND IS OMITTED, IT IS ASSUMED TO BE 1.

THE GENERIC FORM OF THE "STEPIN" COMMAND (ABBREVIATED "SI") IS IDENTICAL TO THAT OF THE "STEP" COMMAND.

EXAMPLES:

THE FORTRAN PROGRAM BELOW IS USED AS THE BASIS FOR THE EXAMPLES THROUGHOUT THIS SECTION:

```
(0001)        CALL WRITEN (3)
(0002)        CALL WRITEN (4)
(0003)        CALL WRITEN (5)
(0004)        CALL EXIT
(0005)        END
(0006)
(0007)        SUBROUTINE WRITEN (N)
(0008)        WRITE (1, 10) N
(0009) 10     FORMAT ('ARGUMENT IS ', I3)
(0010)        RETURN
(0011)        END
```

1) ASSUMING THE DEBUGGER HAS JUST BEEN ENTERED FROM PRIMOS COMMAND LEVEL, TO RESTART THE PROGRAM AND STOP PRIOR TO THE FIRST EXECUTABLE STATEMENT, ENTER THE COMMAND

```
> RESTART_STEP

**** "STEP" COMPLETION AT $MAIN\1
>
```

THE "COMPLETION" MESSAGE INDICATES THAT THE NEXT STATEMENT TO BE EXECUTED IS THAT ON SOURCE LINE 1 IN THE FORTRAN MAIN PROGRAM.

2) TO EXECUTE THE FIRST STATEMENT IN THE PROGRAM, STEPPING ACROSS THE CALLED PROCEDURE, YOU WOULD ENTER

```
> STEP
ARGUMENT IS   3

**** "STEP" COMPLETION AT $MAIN\2
>
```

THE "ARGUMENT IS 3" MESSAGE WAS PRINTED BY "WRITEN", CALLED FROM THE STATEMENT ON SOURCE LINE 1 IN THE MAIN PROGRAM. NOTE THAT, HAD WE STEPPED INTO THE CALLED PROCEDURE, WE WOULD HAVE STOPPED PRIOR TO THE EXECUTION OF THE STATEMENT ON LINE 8.

3) ASSUMING WE'VE JUST COMPLETED EXAMPLE 2, TO STEP INTO "WRITEN" AND STOP AT THE "RETURN" STATEMENT ON LINE 10, YOU WOULD ENTER

```
> STEPIN_3
ARGUMENT IS   4

**** "IN" COMPLETION AT WRITEN\10 ($10+1)
>
```

THE 3 STATEMENTS EXECUTED WITH THIS COMMAND ARE THE "CALL" ON LINE 2 IN THE MAIN PROGRAM, THE "WRITE" ON LINE 8 IN

SUBROUTINE "WRITEN", AND THE "FORMAT" ON LINE 9 IN "WRITEN". (ALTHOUGH THE FORMAT STATEMENT IS NON-EXECUTABLE, THE USER MAY "GOTO" IT IN HIS FORTRAN PROGRAM AND BREAKPOINT AT IT USING THE DEBUGGER; HE MUST THEREFORE MUST ACCOUNT FOR IT WHEN USING A "STEP" OR "STEPIN" COMMAND.)

THE NOTE WHICH APPEARS IN SECTION 18.1 REGARDING "ETRACE" (ENTRY TRACE) COMMAND EXECUTION TIME IS APPLICABLE TO ALL SINGLE STEP COMMANDS. EXECUTION TIMES ARE SLIGHTLY LONGER.

16.2 CONTINUE UNTIL THE NEXT PROCEDURE CALL - THE "IN" COMMAND

THE "IN" COMMAND INSTRUCTS THE DEBUGGER TO CONTINUE PROGRAM EXECUTION UNTIL THE NEXT PROCEDURE IS CALLED.

WHEN THE "IN" COMMAND IS GIVEN, EXECUTION CONTINUES FROM THE LOCATION SPECIFIED BY THE EXECUTION ENVIRONMENT POINTER UNTIL EITHER A PROCEDURE IS CALLED OR THE PROGRAM BLOCK SPECIFIED BY THE EXECUTION ENVIRONMENT POINTER AT THE TIME THE "IN" COMMAND WAS ISSUED RETURNS.

IF THE EXECUTION ENVIRONMENT IS UNDEFINED AT THE TIME "IN" IS GIVEN, "RESTART IN" IS IMPLIED. SEE SECTION 10.1 FOR DETAILS.

THE FORMAT OF THE "IN" COMMAND IS:

```
IN
```

EXAMPLES:

THE EXAMPLES BELOW REFER TO THE SAMPLE FORTRAN PROGRAM IN SECTION 16.1.

1) ASSUMING THAT THE DEBUGGER HAS JUST BEEN ENTERED FROM PRIMOS COMMAND LEVEL, THE MAIN PROGRAM MAY BE ENTERED AND CONTROL RETURNED TO THE DEBUGGER BY ENTERING

```
> IN

**** "IN" COMPLETION AT ENTRY TO $MAIN
>
```

2) EXECUTION MAY BE CONTINUED UNTIL THE FIRST CALL TO "WRITEN" BY ENTERING

```
> IN

**** "IN" COMPLETION AT ENTRY TO WRITEN
>
```

16.3 EXITING THE CURRENT PROGRAM BLOCK - THE "OUT" COMMAND

THE "OUT" COMMAND IS USED TO CONTINUE PROGRAM EXECUTION UNTIL THE PROGRAM BLOCK SPECIFIED BY THE EXECUTION ENVIRONMENT POINTER AT THE TIME THE "OUT" COMMAND IS ISSUED RETURNS.

THE FORMAT OF THE "OUT" COMMAND IS:

```
OUT
```

EXAMPLES:

1) ASSUME THAT WE'RE AT THE ENTRY TO "WRITEN", USED IN THE ABOVE EXAMPLES. TO CONTINUE PROGRAM EXECUTION UNTIL THE PROCEDURE RETURNS, ENTER

```
> OUT

**** "OUT" COMPLETION AT EXIT FROM WRITEN INTO $MAIN\2
>
```

2) NOW, TO CONTINUE EXECUTION UNTIL THE MAIN PROGRAM "RETURNS",

```
> OUT

EXIT.  PROGRAM EXIT FROM $MAIN\4.
>
```

NOTE THAT IT WAS THE INTENT TO RESUME EXECUTION UNTIL THE MAIN PROGRAM RETURNED; HOWEVER, THE USER PROGRAM CALLED "EXIT", WHICH CAUSED THE "EXIT$" CONDITION TO BE SIGNALLED BY PRIMOS, WHICH IN TURN CAUSED THE DEBUGGER TO REGAIN CONTROL AND PRINT THE EXIT MESSAGE.

17 VALUE TRACING

THE "VALUE TRACE" FACILITY ALLOWS THE USER TO SPECIFY A LIST OF VARIABLES WHOSE VALUES ARE TO BE "WATCHED" DURING PROGRAM EXECUTION. THESE VARIABLES ARE KEPT IN A DEBUGGER TABLE KNOWN AS THE "WATCH LIST". A MESSAGE IS PRINTED EACH TIME THE VALUE OF ANY WATCHED VARIABLE CHANGES, GIVING THE FORMER VALUE, THE NEW VALUE, AND THE PROGRAM LOCATION AT WHICH THE CHANGE WAS DETECTED.

FOR ARRAYS AND STRUCTURES, A MESSAGE IS PRINTED ONLY FOR THOSE ELEMENTS WHICH HAVE CHANGED. ANY PORTION OF AN ARRAY OR STRUCTURE WHICH MAY BE REFERENCED MAY BE WATCHED.

WHEN VALUE TRACING IS ENABLED, THE PROGRAM IS EXECUTED IN SINGLE STEPS. FOLLOWING EACH STEP, THE SAVED VALUE OF A WATCHED VARIABLE IS COMPARED WITH ITS CURRENT VALUE AND ANY DISCREPANCIES ARE REPORTED.

EXAMPLE:

```
1:          INTEGER A
2:          DIMENSION A (3)
3:
4:          N = 0
5:          DO 10 I = 1, 3
6:          N = N + 1
7: 10       A(I) = N
```

IF THE VARIABLES N AND A FROM THE ABOVE PROGRAM ARE ON THE WATCH LIST, EXECUTION OF THE STATEMENTS ABOVE WILL YIELD THE FOLLOWING MESSAGES:

```
THE VALUE OF $MAIN\N HAS BEEN CHANGED AT $MAIN\$10 (7)
  FROM 0
  TO   1
THE FOLLOWING VALUES IN $MAIN\A HAVE BEEN CHANGED AT $MAIN\6
  A(1) FROM 0
       TO   1
THE VALUE OF $MAIN\N HAS BEEN CHANGED AT $MAIN\$10 (7)
  FROM 1
  TO   2
THE FOLLOWING VALUES IN $MAIN\A HAVE BEEN CHANGED AT $MAIN\6
  A(2) FROM 0
       TO   2
THE VALUE OF $MAIN\N HAS BEEN CHANGED AT $MAIN\$10 (7)
  FROM 2
  TO   3
THE FOLLOWING VALUES IN $MAIN\A HAVE BEEN CHANGED AT $MAIN\$10+1 (8)
  A(3) FROM 0
       TO   3
```

VARIABLES MAY BE ADDED TO THE WATCH LIST AT ANY TIME AND WILL REMAIN THERE UNTIL REMOVED BY THE "UNWATCH" COMMAND. THE WAY IN WHICH THE WATCHED VARIABLES ARE MONITORED DIFFERS FOR EACH STORAGE CLASS:

- THE VALUE OF A STATIC VARIABLE IS SAVED WHEN THE "WATCH" COMMAND IS GIVEN AND IS WATCHED THROUGHOUT THE DEBUGGING SESSION (UNLESS REMOVED BY THE "UNWATCH" COMMAND).

- EACH GENERATION OF AN AUTOMATIC VARIABLE IS WATCHED UNLESS A SPECIFIC GENERATION IS DESIGNATED BY THE USER. THE VALUE IS SAVED UPON BLOCK ENTRY AND MONITORED UNTIL THE BLOCK BECOMES INACTIVE.

- A PL/1 BASED VARIABLE IS SAVED AND WATCHED ACCORDING TO THE STORAGE CLASS OF ITS LOCATOR, I.E., QUALIFICATION BY A STATIC OR "CONSTANT" LOCATOR CAUSES THE BASED VARIABLE TO BE WATCHED AS THOUGH ITS STORAGE CLASS WERE STATIC, BY AN AUTOMATIC LOCATOR AS THOUGH ITS STORAGE CLASS WERE AUTOMATIC.

- NO FACILITY CURRENTLY EXISTS FOR WATCHING PL/1 VARIABLES OF THE STORAGE CLASS "CONTROLLED".

17.1 ADDING A VARIABLE TO THE WATCH LIST -- THE "WATCH" COMMAND

THE "WATCH" COMMAND ADDS ONE OR MORE VARIABLES TO THE WATCH LIST AND AUTOMATICALLY ENABLES VALUE TRACING.

THE FORMAT OF THE COMMAND (ABBREVIATED "WA") IS:

WATCH <VARIABLE-1> [, <VARIABLE-2> ...]

VARIABLE REFERENCES DESCRIBED IN SECTION 9.4.

IN ORDER TO INDICATE THAT AN AUTOMATIC VARIABLE SHOULD ONLY BE WATCHED IN A SPECIFIED ACTIVATION OF A BLOCK, AN ACTIVATION NUMBER MAY BE GIVEN. IF OMITTED, THE VARIABLE WILL BE WATCHED IN ALL ACTIVATIONS.

EXAMPLES:

(1) IN ORDER TO WATCH THE VARIABLE X, THE ARRAY LIST, AND THE ARRAY CROSS-SECTION SPECIFIED BY TABLE (2, *), IN THE CURRENT EVALUATION ENVIRONMENT, YOU WOULD ENTER:

> WATCH X, LIST, TABLE (2, *)

(2) VARIABLES IN ANOTHER BLOCK MUST BE FURTHER QUALIFIED:

```
> WA_SUBR1\Y,_SUBR2\ARRAY(8)
```

(3) IN ORDER TO WATCH (AUTOMATIC) VARIABLE N ONLY IN THE THIRD ACTIVATION OF PROCEDURE FACTORIAL, ENTER:

```
> WATCH_FACTORIAL\3\N
```

WATCHING_PL/1_BASED_VARIABLES:

WHEN A PL/1 BASED VARIABLE IS ADDED TO THE WATCH LIST, THE LOCATOR EXPRESSION IS EVALUATED AT THE TIME THE "WATCH" COMMAND IS GIVEN, AND THE RESULTING ADDRESS IS USED AS THE LOCATOR AS LONG AS THE VARIABLE REMAINS IN THE WATCH LIST. THEREFORE, AT THE TIME A BASED VARIABLE IS ADDED TO THE WATCH LIST:

(1) IF THE LOCATOR IS AUTOMATIC, THE BLOCK CONTAINING THE LOCATOR MUST BE ACTIVE, AND

(2) THE LOCATOR MUST BE INITIALIZED TO THE DESIRED ADDRESS.

FOR EXAMPLE, CONSIDER THE FOLLOWING PROGRAM:

```
 1: P1:  PROCEDURE;
 2:
 3:      DECLARE LAST_NAME CHARACTER (15);
 4:      DECLARE LOCATOR   POINTER;
 5:      DECLARE STRING8   BASED CHARACTER (8);
 6:
 7:
 8:      LAST_NAME = 'WASHINGTON      ';
 9:      LOCATOR = ADDR (LAST_NAME);
10:      LAST_NAME = 'JEFFERSON       ';
11:      END;    /* P1 */
```

IN ORDER TO WATCH LOCATOR -> STRING8, P1 MUST BE ACTIVE, AND LOCATOR MUST BE INITIALIZED, THEREFORE THE "WATCH" COMMAND CANNOT BE GIVEN UNTIL AFTER EXECUTION OF THE STATEMENT ON SOURCE LINE 9, AS FOLLOWS:

```
**** BREAKPOINTED AT P1\10
> WATCH_LOCATOR_->_STRING8
> CONTINUE
THE VALUE OF P1\LOCATOR -> STRING8 HAS BEEN CHANGED AT P1\11
  FROM 'WASHINGT'
  TO   'JEFFERSO'
```

ALTERNATIVELY, STRING8 MAY BE WATCHED ANY TIME AFTER ENTRY TO P1 BY SPECIFYING A CONSTANT POINTER:

```
> WATCH ADDR (LAST_NAME) -> STRING8
```

NOTE THAT EVEN IF THE VALUE OF THE LOCATOR IN THE USER PROGRAM CHANGES VALUE, THE ADDRESS COMPUTED WHEN THE BASED VARIABLE WAS ADDED TO THE WATCH LIST WILL CONTINUE TO BE USED FOR VALUE TRACING.

17.2 THE "WATCHLIST" COMMAND

THE "WATCHLIST" COMMAND (ABBREVIATED "WL") PRINTS THE NAMES OF THE VARIABLES CURRENTLY ON THE WATCH LIST.

FOR EXAMPLE:

```
> WATCHLIST
MAIN\X
MAIN\LIST
MAIN\TABLE(2,*)
SUBR1\Y
SUBR2\ARRAY(8)
FACTORIAL\3\N
```

17.3 THE "UNWATCH" COMMAND

THE "UNWATCH" COMMAND REMOVES ONE OR MORE VARIABLES FROM THE WATCH LIST.

THE FORMAT OF THE COMMAND (ABBREVIATED "UW") IS:

```
UNWATCH <VARIABLE-1> [, <VARIABLE-2> ...]
```

WHERE <VARIABLE-N> IS EXPLAINED IN SECTION 9.4.

AS IN THE "WATCH" COMMAND, IF THE PROCEDURE NAME IS OMITTED, THE BLOCK CORRESPONDING TO THE CURRENT EVALUATION ENVIRONMENT IS USED. THE FULLY QUALIFIED VARIABLE NAME MAY ALWAYS BE FOUND VIA THE "WATCHLIST" COMMAND.

FOR EXAMPLE, TO REMOVE THE VARIABLES LIST, Y, AND ARRAY(8) FROM THE WATCH LIST IN THE EARLIER EXAMPLE, YOU WOULD ENTER:

```
> UNWATCH MAIN\LIST, SUBR1\Y, SUBR2\ARRAY(8)
```

IT IS POSSIBLE TO DISCONTINUE VALUE TRACING WITHOUT "UNWATCHING" ALL THE VARIABLES IN THE WATCH LIST BY GIVING THE COMMAND "VTRACE OFF".

IF TRACING IS SUBSEQUENTLY RE-ENABLED, BY EITHER THE "WATCH" OR "VTRACE ON" COMMAND, THESE VARIABLES WILL AGAIN BE TRACED.

NOTE THAT THE "UNWATCH" COMMAND DOES NOT DISABLE VALUE TRACING UNLESS THE WATCH LIST BECOMES EMPTY.

17.4 THE "VTRACE" COMMAND

THE "VTRACE" COMMAND ALLOWS THE USER TO ENABLE OR DISABLE VALUE TRACING AT ANY TIME, WHILE RETAINING THE VARIABLES IN THE WATCH LIST.

THE FORMAT OF THE COMMAND (ABBREVIATED "VT") IS:

```
VTRACE ;ON < OFF=
```

THE "WATCH" COMMAND IMPLICITLY ENABLES VALUE TRACING. IF VALUE TRACING IS SUBSEQUENTLY TURNED OFF USING THE "VTRACE OFF" COMMAND, ALL VARIABLES CURRENTLY BEING WATCHED WILL REMAIN IN THE WATCH LIST, BUT SAVED VALUES WILL NOT BE UPDATED AND NO MESSAGE WILL BE PRINTED WHEN WATCHED VARIABLES CHANGE VALUES.

IF VALUE TRACING IS RE-ENABLED, EITHER USING THE "VTRACE ON" COMMAND OR BY ADDING VARIABLES TO THE WATCH LIST WITH THE "WATCH" COMMAND, ALL SAVED VALUES ARE IMMEDIATELY UPDATED TO CORRESPOND TO THE CURRENT VALUE OF THE VARIABLE, AND VALUE TRACING IS RESUMED.

18 PROGRAM TRACING

TWO DEBUGGER FACILITIES ARE AVAILABLE WHICH ALLOW THE USER TO TRACE PROGRAM EXECUTION CONTROL FLOW. THEY ARE:

- THE "ENTRY TRACE" FACILITY, WHICH CAUSES A MESSAGE TO BE PRINTED EACH TIME A PROCEDURE IS CALLED OR RETURNS, AND

- THE "STATEMENT TRACE" FACILITY, WHICH CAUSES A MESSAGE TO BE PRINTED PRIOR TO THE EXECUTION OF EACH STATEMENT. THE USER MAY OPTIONALLY LIMIT STATEMENT TRACE MESSAGES TO BE PRODUCED ON LABELLED STATEMENTS ONLY.

THESE PROGRAM TRACE FACILITIES ARE INDEPENDENT OF ONE ANOTHER AND MAY BE ENABLED OR DISABLED AT ANY POINT DURING THE DEBUGGING SESSION. FURTHERMORE, USE OF THE TRACE FACILITIES IS APPLICABLE TO THE ENTIRE DEBUGGING ENVIRONMENT. THE USER CANNOT, STRICTLY SPEAKING, ENABLE STATEMENT OR ENTRY/EXIT TRACING FOR ONE PARTICULAR ROUTINE. THIS EFFECT CAN BE ACHIEVED USING BREAKPOINTS. THE USER MAY SET A BREAKPOINT AT THE ENTRY OF THE PROCEDURE, SPECIFYING AN ACTION LIST WHICH CONTAINS THE APPROPRIATE TRACE ENABLE COMMANDS. IN ORDER TO CONTINUE PROGRAM EXECUTION, THE TRACE COMMANDS SHOULD BE FOLLOWED BY A "CONTINUE" COMMAND. SIMILARLY, TRACING CAN THEN BE DISABLED BY SETTING A CORRESPONDING EXIT BREAKPOINT WITH THE APPROPRIATE TRACE DISABLE COMMANDS CONTAINED WITHIN AN ACTION LIST.

18.1 ENTRY TRACING - THE "ETRACE" COMMAND

THE "ETRACE" COMMAND IS USED TO ENABLE OR DISABLE THE ENTRY TRACE FACILITY DESCRIBED ABOVE.

THE FORMAT FOR THE "ETRACE" COMMAND (ABBREVIATED "ET") IS:

```
ETRACE ;ON < ARGS < OFF=
```

THE "ON" MODIFIER ENABLES ENTRY TRACING. WHEN PROGRAM EXECUTION IS BEGUN OR RESUMED, THE DEBUGGER WILL CAUSE A MESSAGE TO BE PRINTED AS EACH PROCEDURE IS CALLED AND AS EACH PROCEDURE RETURNS. THE MESSAGE IS OF THE FORM:

```
**** ENTRY TO <PROGRAM-BLOCK-NAME>
```

OR

```
**** EXIT FROM <PROGRAM-BLOCK-NAME>
```

WHERE <PROGRAM-BLOCK-NAME> IS A PROGRAM BLOCK NAME, AS DESCRIBED IN SECTION 9.1.

IF THE "ARGS" MODIFIER IS SPECIFIED, THE ENTRY AND EXIT TRACE MESSAGES ARE GENERATED AS DESCRIBED ABOVE; HOWEVER, IN ADDITION,

THE VALUES OF ARGUMENTS PASSED TO THE CALLED ROUTINE ARE PRINTED UPON PROCEDURE ENTRY. THE VALUES OF THE ARGUMENTS ARE NOT PRINTED ON PROCEDURE EXIT. IN GENERAL, THERE IS NO WAY TO LOCATE THE ARGUMENTS PASSED TO A PROCEDURE AFTER THAT PROCEDURE RETURNS (BY THE TIME THE DEBUGGER REGAINS CONTROL, THE STACK FRAME HAS ALREADY BEEN RELEASED BY THE FIRMWARE PROCEDURE CALL/RETURN MECHANISM).

FOR EXAMPLE, THE FOLLOWING TEXT MIGHT BE PRINTED UPON THE ENTRY TO PL/1 PROCEDURE "GARBLE_ARGS", WHICH ACCEPTS A FIXED BINARY AND A CHARACTER STRING ARGUMENT:

```
**** ENTRY TO GARBLE_ARGS
NUMBER = 2002
TEXT = 'A SPACE ODDITY'
```

IF THE "OFF" MODIFIER IS SPECIFIED, THE ENTRY TRACE FACILITY IS DISABLED.

EXAMPLES:

1) TO ENABLE ENTRY TRACING AND CAUSE THE VALUES OF ARGUMENTS TO BE DISPLAYED UPON EACH PROCEDURE ENTRY:

```
> ETRACE ARGS
```

2) TO DISABLE ENTRY TRACING:

```
> ET OFF
```

NOTE: THE AMOUNT OF TIME USED TO EXECUTE THE "ETRACE" COMMAND IS DIRECTLY DEPENDENT UPON THE SIZE OF THE PROGRAM BEING DEBUGGED, SYSTEM LOAD, AND AMOUNT OF REAL MEMORY AVAILABLE. EACH USER-OWNED ECB AND STACK FRAME IS MODIFIED TO TRAP ANY PROCEDURE CALLS OR RETURNS. THE "ETRACE" COMMAND HAS BEEN OBSERVED TO TAKE BETWEEN 5-15 (ELAPSED) SECONDS WHEN DEBUGGING A LARGE SOFTWARE SYSTEM (192K WORDS OF PROCEDURE TEXT IN 250 PL/1 PROCEDURES) ON A LIGHTLY TO MODERATELY LOADED PRIME 500 WITH A VERY LARGE AMOUNT OF REAL MEMORY.

OTHER COMMANDS TO WHICH THIS OBSERVATION APPLIES ARE SO NOTED.

18.2 STATEMENT TRACING - THE "STRACE" COMMAND

THE "STRACE" COMMAND IS USED TO ENABLE OR DISABLE THE STATEMENT TRACE FACILITY DESCRIBED EARLIER IN THIS SECTION.

THE FORMAT FOR THE "STRACE" COMMAND (ABBREVIATED "ST") IS:

STRACE ;FULL < QUIET < OFF=

THE "FULL" MODIFIER DESIGNATES THAT THE DEBUGGER IS TO PRINT A MESSAGE PRIOR TO THE EXECUTION OF EACH STATEMENT IN THE PROGRAM. THIS MESSAGE IS OF THE FORM:

**** <STATEMENT-IDENTIFIER>

WHERE <STATEMENT-IDENTIFIER> IS A STATEMENT IDENTIFIER, DESCRIBED IN SECTION 9.5.

THE "QUIET" MODIFIER SPECIFIES THAT A STATEMENT TRACE MESSAGE IS TO BE PRINTED PRIOR TO THE EXECUTION OF EACH LABELLED STATEMENT.

THE "OFF" MODIFIER SPECIFIES THAT STATEMENT TRACING IS TO BE DISABLED.

EXAMPLES:

1) TO ENABLE STATEMENT TRACING ON ALL STATEMENTS:

> STRACE_FULL

2) TO DISABLE STATEMENT TRACING:

> ST_OFF

THE NOTE APPEARING EARLIER IN THIS SECTION REGARDING "ETRACE" COMMAND EXECUTION SPEED AND CHARACTERISTICS IS ALSO APPLICABLE TO THE "STRACE" COMMAND. THE ELAPSED TIMES ARE SLIGHTLY LONGER.

19 CALLING A USER PROCEDURE - THE "CALL" COMMAND

USER FUNCTIONS MAY BE EVALUATED AS PART OF AN EXPRESSION, AS DESCRIBED IN SECTION 12.1.2. IN ADDITION, ANY USER SUBROUTINE OR FUNCTION MAY BE CALLED FROM DEBUGGER COMMAND LEVEL USING THE "CALL" COMMAND.

THE FORMAT OF THE "CALL" COMMAND IS:

```
CALL  <VARIABLE>  [(<ARGUMENT-LIST>)]
```

WHERE:

<VARIABLE> IS A VARIABLE NAME AS DESCRIBED IN SECTION 9.4 WHICH MUST BE OF TYPE ENTRY CONSTANT OR ENTRY VARIABLE. THIS CORRESPONDS TO A FORTRAN SUBROUTINE/FUNCTION NAME OR VARIABLE DECLARED TO BE "EXTERNAL".

<ARGUMENT-LIST> IS A LIST OF EXPRESSIONS SUPPLIED AS ARGUMENTS TO THE CALLED PROCEDURE.

WHEN THE "CALL" COMMAND IS GIVEN, EACH ARGUMENT IS EVALUATED, THEN THE PROCEDURE IS CALLED BY THE DEBUGGER, WHICH SUPPLIES THE RESULTING VALUES AS ARGUMENTS. FOR EXAMPLE:

```
> CALL SIRSORI (LIST, KEY1 + 2)
```

IF THE USER PROCEDURE IS A FUNCTION, THE RETURNED VALUE IS PRINTED ON THE USER'S TERMINAL. FOR EXAMPLE:

```
> CALL GETIDX
RETURNED VALUE: 11
```

IF THE USER PROCEDURE IS NOT DECLARED WITHIN THE BLOCK CORRESPONDING TO THE EVALUATION ENVIRONMENT POINTER, QUALIFICATION BY A PROGRAM BLOCK NAME IS NECESSARY. FOR EXAMPLE:

```
> CALL PRINTNAME\PRINTNAME (12)
```

EACH "CALL" COMMAND EFFECTIVELY CAUSES A NEW INVOCATION OF THE DEBUGGER. THE DEBUGGER "CALL" COMMAND PROCESSOR CREATES A PROCEDURE-CALL STACK FRAME, KNOWN AS THE "DEBUGGER CALL FRAME", FROM WHICH THE USER PROCEDURE IS CALLED. A STACK TRACEBACK SHOWS THIS DEBUGGER-OWNED FRAME:

```
(1:INSERT) > TRACEBACK -FRAMES 2 -DBG
CURRENTLY AT INSERT\47.
STACK CONTAINS 19 FRAMES.

 19: OWNER IS "INSERT".
     CALLED FROM & RETURNS TO DEBUGGER CALL PROCESSOR.

 18: DEBUGGER "CALL" FRAME.
     CALLED FROM DEBUGGER, RETURNS TO DEBUGGER.
```

THIS CALL FRAME IS USED TO STORE DEBUGGER CONTROL INFORMATION PERTAINING TO THE PREVIOUS ACTIVATION OF THE DEBUGGER. FOR THIS REASON, THE USER PROGRAM MAY NOT EXECUTE A NON-LOCAL GOTO PAST ANY DEBUGGER CALL FRAME, EXCEPT BY USING THE DBG "GOTO" COMMAND; SEE SECTION 10.3.

IF EXECUTION IS SUSPENDED DURING THE CALLED USER PROCEDURE, THE DEBUGGER PRINTS A PROMPT OF THE FOLLOWING FORM:

```
( <CALL-LEVEL> : <PROCEDURE-NAME> ) >
```

WHERE:

<CALL-LEVEL> INDICATES THE LEVEL OF INVOCATION OF DBG.

<PROCEDURE-NAME> IS THE NAME OF THE CALLED USER PROCEDURE.

FOR EXAMPLE:

```
> BREAKPOINT INSERT\47
> BREAKPOINT RELINK\\ENTRY
> CALL INSERT (NEWITEM)

**** BREAKPOINTED AT INSERT\47
(1:INSERT) > CALL RELINK (PTR)

**** BREAKPOINTED AT ENTRY TO INSERT.RELINK
(2:RELINK) >
```

THE "UNWIND" COMMAND MAY BE USED TO UNWIND THE CALL/RETURN STACK TO THE FIRST INVOCATION OF THE DEBUGGER (CALL LEVEL ZERO) AND RELEASE ALL USER-OWNED STACK FRAMES; SEE SECTION 10.5.

20 MISCELLANEOUS COMMANDS

20.1 SET PRIMOS COMMAND LINE (COMANL) - THE "CMDLINE" COMMAND

THE "CMDLINE" COMMAND CAUSES THE DEBUGGER TO CALL THE PRIMOS SUBROUTINE "COMANL" TO READ A LINE INTO THE (STATIC) COMMAND LINE BUFFER.

THE FORMAT OF THE "CMDLINE" COMMAND (ABBREVIATED "CL") IS:

```
CMDLINE
```

THE USER SHOULD BE CAREFUL NOT TO ENTER THE NAME OF THE COMMAND, "SEG", OR "DBG" FOLLOWING THE "CMDLINE" COMMAND. THE FIRST TOKEN ENTERED WILL BE THE FIRST READ BY THE PROGRAM.

THE "COMANL" BUFFER IS UNDISTURBED BY DBG.

EXAMPLE:

```
> CMDLINE
ENTER COMMAND LINE:
-FREQ 12 -BRIEF
```

20.2 REPEAT COMMAND LINE - THE "*" COMMAND

THE DEBUGGER REPEAT COMMAND ("*") IS USED TO EXECUTE THE COMMAND LINE EITHER 'FOREVER' (UNTIL AN ERROR OCCURS), OR A SPECIFIC NUMBER OF TIMES.

THE "*" COMMAND MUST BE THE LAST ON THE COMMAND LINE, SEPARATED FROM PRECEDING COMMANDS BY THE SEPARATOR CHARACTER (INITIALLY ";"). TO REPEAT THE COMMAND LINE A SPECIFIC NUMBER OF TIMES, THE REPEAT COUNT SHOULD IMMEDIATELY FOLLOW THE "*". TO REPEAT THE COMMAND LINE 'FOREVER', DON'T SUPPLY A REPEAT COUNT. REPETITION CEASES AND CONTROL IS RETURNED TO THE USER TERMINAL WHEN ANY ERROR IS DETECTED. (AN ERROR MESSAGE WILL BE PRINTED.)

THE GENERIC FORM OF THE "*" COMMAND IS:

```
* <VALUE>
```

WHERE:

<VALUE> IS A POSITIVE INTEGER VALUE.

IF <VALUE> IS OMITTED, THE COMMAND LINE IS REPEATED 'FOREVER'; IF SUPPLIED, IT IS REPEATED <VALUE> TIMES.

EXAMPLES:

1) TO PRINT THE VALUES OF A(I) SQUARED, VARYING "I" FROM 1 TO 10 BY 1, ENTER

```
> LET I = 0
> LET I = I + 1; : A(I) ** 2.; *10
```

NOTE THAT "I" MUST BE A VARIABLE DEFINED WITHIN THE USER PROGRAM.

2) TO LOCATE ALL OCCURRENCES OF THE CHARACTER "&" IN THE CURRENT SOURCE FILE, ENTER

```
> SOURCE TOP
> SOURCE LOCATE &; *
BOTTOM
>
```

20.3 PRINT REFERENCE DOCUMENTATION INFO - THE "HELP" COMMAND

THE "HELP" COMMAND MAY BE USED TO FIND THE NAME OF THE MOST RECENT AND UP-TO-DATE DBG DOCUMENTATION.

THE FORMAT OF THE "HELP" COMMAND IS:

```
HELP
```

EXAMPLE:

```
> HELP
FOR HELP, REFER TO PRIME ENGINEERING TECHNICAL DOCUMENT PE-T-600.
```

20.4 EXECUTE PRIMOS COMMAND - THE "!" COMMAND

THE "!" COMMAND CAUSES THE DEBUGGER TO PASS THE TEXT WHICH FOLLOWS IT TO THE PRIMOS COMMAND PROCESSOR FOR EXECUTION.

THE USER MUST TAKE GREAT CARE IN NOT ENTERING A PRIMOS COMMAND WHICH WILL CAUSE THE MEMORY IMAGE OF THE DEBUGGER OR PROGRAM BEING DEBUGGED TO BE DISTURBED. "RESTORE", "RESUME", AND ALL "CMDNCO" COMMANDS ARE EXPRESSLY PROHIBITED, AS THEY CAUSE INFORMATION TO BE

LOADED INTO SEGMENT 4000. (AT PRESENT, THERE IS NO WAY FOR THE DEBUGGER TO DISALLOW THESE TYPES OF COMMANDS.) ANOMALOUS BEHAVIOR BY THE DEBUGGER FOLLOWING EXECUTION OF A PRIMOS COMMAND IS INDICATIVE OF MISUSE.

THE FORMAT OF THE "!" COMMAND IS:

```
! <PRIMOS-COMMAND-LINE>
```

WHERE:

<PRIMOS-COMMAND-LINE> IS A PRIMOS COMMAND LINE, TO BE PASSED VERBATIM TO THE PRIMOS COMMAND PROCESSOR.

EXAMPLE:

```
> ! ATTACH J.PAUL.JONES.ESQ
```

20.5 ENTER VPSD - THE "VPSD" COMMAND

THE "VPSD" COMMAND CAUSES THE DEBUGGER TO ENTER A BUILTIN COPY OF VPSD, THE 64V MODE PRIME SYMBOLIC DEBUGGER.

THE VPSD BASE REGISTERS AND GENERAL MACHINE REGISTERS ARE SET UP TO THEIR VALUES AT THE TIME DBG WAS RE-ENTERED FROM THE USER PROGRAM. ALTHOUGH THE USER MAY MODIFY THESE REGISTERS WHILE WITHIN VPSD, THE UPDATED VALUES ARE NOT RETURNED TO DBG AND THEREFORE NOT PLACED IN THE REGISTER SET WHEN PROGRAM EXECUTION IS CONTINUED.

TO RETURN TO DBG FROM VPSD, ENTER THE "Q" COMMAND FOLLOWING THE VPSD PROMPT CHARACTER ("1").

THE USER SHOULD NOT SET VPSD BREAKPOINTS WHILE RUNNING IN THE DEBUGGER; DOING SO WILL CAUSE ANOMALOUS BEHAVIOR. THIS RESTRICTION WILL BE LIFTED AT SOME TIME IN THE FUTURE.

THE FORMAT OF THE "VPSD" (OR "PSD") COMMAND IS:

```
VPSD
```

EXAMPLE:

```
> VPSD

1Q
>
```

21 COMMAND LINE EDIT FACILITY

THE COMMAND LINE EDIT FACILITY ALLOWS THE USER TO MODIFY THE CONTENT OF BREAKPOINT ACTION LISTS AND TO EDIT AND RESUBMIT THE MOST RECENT COMMAND LINE.

WHEN THE EDITOR IS INVOKED IT PRINTS THE LINE TO BE EDITED. THIS IS THE PREVIOUS COMMAND LINE (FOR THE "RESUBMIT" COMMAND) OR THE BREAKPOINT ACTION LIST (FOR "BREAKPOINT -EDIT"). THE USER MODIFIES THIS LINE BY POSITIONING THE CURSOR UNDER THE CHARACTER OR CHARACTERS TO BE AFFECTED AND ENTERING THE APPROPRIATE EDIT COMMANDS. WHEN THE USER TYPES THE RETURN KEY, THE LINE IS MODIFIED AS DIRECTED AND THE UPDATED LINE IS PRINTED AT THE TERMINAL. THE USER MAY CONTINUE IN THIS FASHION UNTIL THE LINE HAS BEEN MODIFIED AS DESIRED. WHEN DONE EDITING, THE USER TYPES THE 'RETURN' KEY, WITH NO EDIT CHARACTERS ON THE LINE, TO EITHER EXECUTE THE COMMAND LINE OR REPLACE THE ACTION LIST.

THE FOLLOWING EDIT OPERATIONS ARE IMPLEMENTED:

D - DELETE CHARACTER.

THE CHARACTER UNDER WHICH THE D IS POSITIONED IS DELETED. THE LINE IS "SHIFTED" TO THE LEFT TO FILL IN THE VACANT CHARACTER POSITION. FURTHER EDIT OPERATIONS ARE ALLOWED FOLLOWING A "D". FOR EXAMPLE,

```
      NOW IS THE TIME FOR ALL GOOD GOOD WOMEN...
:                              DDDDD     DD
      NOW IS THE TIME FOR ALL GOOD MEN...
:
```

F - DEFINE FIRST CHARACTER.

THE "F" EDIT OPERATION CAUSES THE CHARACTER UNDER WHICH IT IS POSITIONED TO BECOME THE FIRST CHARACTER OF THE LINE. THAT IS, THE PORTION OF THE LINE APPEARING PRIOR TO THE "F" IS TRUNCATED. FURTHER EDIT OPERATIONS ARE ALLOWED FOLLOWING THE "F". FOR EXAMPLE,

```
      THESE ARE THE TIMES WHICH TRY MEN'S SOULS...
:                               F
      TRY MEN'S SOULS
:
```

L - DEFINE LAST CHARACTER.

THE "L" EDIT OPERATION CAUSES THE CHARACTER UNDER WHICH IT IS POSITIONED TO BECOME THE LAST CHARACTER OF THE LINE. THE LINE IS TRUNCATED FOLLOWING THIS CHARACTER. FURTHER EDIT OPERATIONS ARE PERMITTED FOLLOWING THE "L" COMMAND. FOR EXAMPLE,

```
      IF ANYTHING CAN GO WRONG, IT WILL.
:                                     L
      IF ANYTHING CAN GO WRONG, IT WIL
```

A - APPEND TEXT TO END OF LINE.

THE "A" COMMAND CAUSES THE TEXT WHICH FOLLOWS TO BE APPENDED AT THE END OF THE LINE. ALL FURTHER TEXT ON THE COMMAND LINE IS INTERPRETED LITERALLY. FOR EXAMPLE,

```
      FOUR SCOR
: AE AND SEVEN YEARS
      FOUR SCORE AND SEVEN YEARS
:
```

I - INSERT TEXT.

THE "I" COMMAND CAUSES THE TEXT WHICH FOLLOWS IT TO BE INSERTED INTO THE LINE FOLLOWING THE CHARACTER UNDER WHICH THE "I" IS POSITIONED. NO FURTHER EDIT COMMANDS ARE PERMITTED ON THE COMMAND LINE. FOR EXAMPLE,

```
      OF THE LINE
: IBEGINNING
      BEGINNING OF THE LINE
:          I, MIDDLE
      BEGINNING, MIDDLE OF THE LINE
:                                 I.
      BEGINNING, MIDDLE OF THE LINE.
:
```

O - OVERLAY TEXT.

THE "O" COMMAND CAUSES THE CHARACTERS WHICH FOLLOW IT TO BE OVERLAYED ONTO THE LINE, STARTING AT THE CHARACTER UNDER WHICH THE "O" APPEARS. NO FURTHER EDIT COMMANDS ARE PERMITTED ON THE COMMAND LINE.

```
      MY KINGDOM FOR A HORSE!
:     OOOOWIFE
      MY WIFE FOR A HORSE!
```

Q - QUIT EDITING.

THE "Q" COMMAND CAUSES THE EDIT OPERATION TO BE ABORTED AND DBG TO RETURN TO COMMAND LEVEL. IF THE EDITOR WAS INVOKED VIA "RESUBMIT", THE COMMAND LINE IS NOT RESUBMITTED; IF INVOKED BY THE "-EDIT" ARGUMENT TO THE BREAKPOINT COMMAND, THE ACTION LIST

IS NOT UPDATED. FOR EXAMPLE,

```
        THIS IS NOT A DEBUGGER COMMAND!
: Q
>
```

THE EDIT COMMANDS MAY BE EITHER UPPER OR LOWER CASE; THEY APPEAR IN UPPER CASE HERE SOLELY FOR CLARITY.

EXAMPLES:

1) ASSUME THAT THE FOLLOWING COMMAND HAS BEEN ENTERED AT THE TERMINAL:

```
> CLEAR $MAIN\6; BRK $MAIN\20; ETRACE ON; CONTINUE

CLEAR $MAIN\6
NO SUCH BREAKPOINT.
```

SEEING THIS ERROR MESSAGE, THE USER REALIZES THAT THE BREAKPOINT IS AT LINE 7, NOT 6; HE ALSO REALIZES THAT ENTRY TRACING SHOULD BE TURNED OFF, NOT ON. THE USER THEN ENTERS THE FOLLOWING "RESUBMIT" COMMAND:

```
> RESUBMIT
   CLEAR $MAIN\6; BRK $MAIN\20; ETRACE ON; CONTINUE
: 07
   CLEAR $MAIN\7; BRK $MAIN\20; ETRACE ON; CONTINUE
: DDDDIOFF;
   CLEAR $MAIN\7; BRK $MAIN\20; ETRACE OFF; CONTINUE
:
```

2) CONSIDER THE ACTION LIST DISPLAYED BY THE FOLLOWING "LIST" COMMAND:

```
> LIST 30
TYPE  LOCATION
BRK   A\30, COUNT = 0
      [LET NA = 'C'; : NB; : UCOUNT; WHERE; CONTINUE]
```

TO REMOVE THE "LET" COMMAND AND EXECUTE THE "WHERE" COMMAND FIRST, THE USER ENTERS:

```
> BRK 30 -ED
    LET NA = 'C'; : NB; : UCOUNT; WHERE; CONTINUE
: ----------------F
    : NB; : UCOUNT; WHERE; CONTINUE
: IWHERE;
    WHERE; : NB; : UCOUNT; WHERE; CONTINUE
: ------------------------DDDDDDD
    WHERE; : NP; : UCOUNT; CONTINUE
:
> LIST 30
TYPE  LOCATION
BRK   A\30, COUNT = 0
      [WHERE; : NB; : UCOUNT; CONTINUE]
```

22 SOURCE FILE OPERATIONS

THIS SECTION DESCRIBES THE DBG "SOURCE" FACILITY WHICH ALLOWS THE USER TO PERUSE SOURCE FILES IN THE COURSE OF THE DEBUGGING SESSION.

THE SOURCE FILE FACILITY IS INVOKED USING THE "SOURCE" COMMAND. FOLLOWING IT, THE USER SPECIFIES ONE OR MORE "ED" (PRIME TEXT EDITOR) COMMANDS TO BE EXECUTED.

ALL "ED" COMMANDS WHICH READ BUT NOT MODIFY THE FILE HAVE BEEN IMPLEMENTED. THESE INCLUDE:

TOP	- POSITION LINE POINTER TO TOP OF FILE.
BOTTOM	- POSITION POINTER TO BOTTOM OF FILE.
BRIEF	- DON'T PRINT TARGET LINES OF FIND, LOCATE, POINT, AND NEXT OPERATIONS.
VERIFY	- PRINT TARGET LINES OF FIND, LOCATE, POINT, AND NEXT OPERATIONS.
PRINT	- PRINT LINE(S).
WHERE	- PRINT CURRENT LINE NUMBER.
POINT	- POSITION TO SPECIFIC LINE.
NEXT	- MOVE LINE POINTER FORWARD OR BACKWARD.
MODE	- SET EDIT MODE; THE ONLY MODE IMPLEMENTED IS "NUMBER" / "NNUMBER".
LOCATE	- LOCATE LINE WITH SPECIFIED TEXT STRING.
FIND	- LOCATE LINE WITH SPECIFIED TEXT STRING IN A GIVEN COLUMN.
SYMBOL	- SET CHARACTER SYMBOL; SEE DEBUGGER "SYMBOL" COMMAND (SECTION 13.6).
PSYMBOL	- PRINT CHARACTER SYMBOLS; SEE DEBUGGER "PSYMBOL" COMMAND (SECTION 13.5).
*	- REPEAT COMMAND LINE; SEE DEBUGGER "REPEAT" COMMAND (SECTION 20.2).

THE READER IS REFERRED TO THE APPROPRIATE PRIME DOCUMENTATION FOR INFORMATION ON THESE "ED" COMMANDS ("THE NEW USER'S GUIDE TO EDITOR AND RUNOFF", FDR3104).

IN ADDITION TO THE ABOVE "ED" COMMANDS, THE FOLLOWING HAVE BEEN ADDED AS A LOGICAL EXTENSION FOR THE DEBUGGING ENVIRONMENT:

EX	- SET THE SOURCE FILE NAME AND LINE POINTER TO CORRESPOND WITH THE STATEMENT OR ENTRY DESCRIBED BY THE EXECUTION ENVIRONMENT POINTER; THEN PRINT THE LINE. THIS IS ILLEGAL WHEN THE EXECUTION ENVIRONMENT POINTER DESCRIBES AN EXIT FROM A PROGRAM BLOCK.
NAME	- SET SOURCE FILE NAME. THE NAME OF THE SOURCE FILE IS SET TO THAT WHICH FOLLOWS THIS COMMAND. IT REMAINS SET UNTIL EITHER ANOTHER NAME IS SPECIFIED OR THE USER ISSUES A "SOURCE NAME -DEFAULT" COMMAND, AT WHICH TIME THE DEBUGGER SWITCHES BACK TO THE "NATURAL" SOURCE FILE. IF THE "SOURCE NAME" COMMAND IS ENTERED WITH NO FILENAME

FOLLOWING IT, THE NAME OF THE CURRENT SOURCE FILE AND ITS DERIVATION (FROM EXECUTION ENVIRONMENT POINTER, EVALUATION ENVIRONMENT POINTER, OR EXPLICITLY SPECIFIED BY THE USER) IS PRINTED. THE RULES FOR DERIVING THE SOURCE FILE NAME ARE DESCRIBED BELOW.

WHEN THE "SOURCE" COMMAND APPEARS ON A COMMAND LINE, ALL COMMANDS WHICH FOLLOW IT (INCLUDING THOSE PRECEDED BY THE SEPARATOR CHARACTER) ARE INTERPRETED BY DBG AS SOURCE COMMANDS. IN ORDER TO SUPPLY FURTHER DEBUGGER (AS OPPOSED TO "SOURCE") COMMANDS ON THE SAME COMMAND LINE, ENTER TWO ADJACENT SEPARATOR CHARACTERS.

THE DEBUGGER SELECTS THE FILE TO BE OPERATED UPON BY "SOURCE" COMMANDS USING THE FOLLOWING RULES:

1) WHEN THE DEBUGGER IS ENTERED FROM PRIMOS COMMAND LEVEL, THE SOURCE FILE IS THAT WHICH CONTAINS THE MAIN PROGRAM. THE LINE POINTER IS POSITIONED TO THE FIRST LINE IN THE PROGRAM.

2) WHEN THE DEBUGGER IS RE-ENTERED FROM THE USER PROGRAM AS A RESULT OF A BREAKPOINT OR CONDITION, THE SOURCE FILE NAME IS SET TO THAT CONTAINING THE PROGRAM BLOCK REPRESENTED BY THE EXECUTION ENVIRONMENT POINTER. THE LINE NUMBER IS THAT OF THE BREAKPOINTED STATEMENT (IF A STATEMENT BREAKPOINT), FIRST LINE IN THE PROGRAM BLOCK (IF AN ENTRY TO A PROGRAM BLOCK), OR NEXT STATEMENT TO BE EXECUTED (IF EXITING FROM A PROGRAM BLOCK). IT IS SET TO THE STATEMENT WHICH CAUSED THE CONDITION TO BE RAISED IF A CONDITION CAUSED DBG REENTRY.

3) WHEN THE USER EXECUTES AN "ENVIRONMENT" COMMAND, THE SOURCE FILE IS SET TO THAT WHICH CONTAINS THE PROGRAM BLOCK DESCRIBED BY THE NEW EVALUATION ENVIRONMENT POINTER. THE LINE NUMBER IS SET TO THE FIRST LINE IN THE PROGRAM BLOCK.

4) WHEN THE "SOURCE EX" COMMAND IS EXECUTED, THE SOURCE FILE NAME IS SET TO CORRESPOND TO THE EXECUTION ENVIRONMENT POINTER. THE LINE NUMBER IS SET AS DESCRIBED IN RULE 2, ABOVE.

THE GENERIC FORM OF THE "SOURCE" COMMAND IS:

SOURCE <SOURCE-COMMAND> [<ARGUMENT>]

WHERE:

<SOURCE-COMMAND> IS ONE OF THE "ED" (TEXT EDITOR) COMMANDS OR A DBG EXTENSION TO THESE COMMANDS, AS DESCRIBED ABOVE.

<ARGUMENT> IS AN OPTIONAL COMMAND ARGUMENT. THE TYPE OF THIS OBJECT (TEXT STRING, FILENAME, LINE NUMBER, ETC.) IS DEPENDENT UPON THE PARTICULAR SOURCE COMMAND.

EXAMPLES:

1) TO POSITION TO THE TOP LINE IN THE FILE AND PRINT THE FIRST 23 LINES, ENTER

```
> SOURCE_TOP; PRINT 23
```

2) TO PRINT THE CURRENT SOURCE FILE NAME, ENTER

```
> SOURCE_NAME
SOURCE FILE IS "PZE_TEST", BASED ON EVALUATION ENVIRONMENT.
```

3) TO PRINT THE CURRENT LINE POINTER IN THE SOURCE FILE, THEN THE DBG EXECUTION ENVIRONMENT POINTER, ENTER

```
> SOURCE_WHERE;; WHERE
```

23 PROGRAM COMPILATION MODES

THIS SECTION DESCRIBES THE THREE MODES IN WHICH PROGRAMS MAY BE COMPILED WITH THE REVISION 17 (AND LATER) FORTRAN AND PL/1 COMPILERS.

23.1 "DEBUG" MODE - PRODUCE FULL DEBUGGER INFORMATION

THE "DEBUG" COMPILATION MODE CAUSES THE COMPILER TO PRODUCE SYMBOL AND STATEMENT INFORMATION FOR THE DEBUGGER.

THE USER MAY SET BREAKPOINTS AT ANY STATEMENT IN, ENTRY TO OR EXIT FROM A PROGRAM COMPILED IN "DEBUG" MODE. ALL SYMBOLS DECLARED WITHIN THE PROGRAM MAY BE REFERENCED BY THE DBG USER.

PROGRAMS COMPILED IN "DEBUG" MODE ARE NON-OPTIMIZABLE. THE AMOUNT OF SPACE OCCUPIED BY THE PROCEDURE TEXT OF "DEBUG" MODE PROGRAMS IS 10-20% LARGER THAN THAT OCCUPIED BY "PRODUCTION" OR "NODEBUG" MODE PROGRAMS. FURTHERMORE, THE AMOUNT OF SPACE OCCUPIED BY THE LINK FRAME FOR EACH "DEBUG" MODE PROGRAM INCREASES 2 WORDS FOR EACH COMMON BLOCK (EXTERNAL STATIC VARIABLE) AND EXTERNAL ENTRY DECLARED.

THE COMMAND LINE OPTION TO SPECIFY "DEBUG" MODE IS "-DEBUG".

23.2 "PRODUCTION" MODE - PRODUCE LIMITED DEBUGGER INFORMATION

"PRODUCTION" MODE IS THE HYBRID OF "DEBUG" AND "NODEBUG" MODES. INFORMATION ABOUT EACH PROGRAM BLOCK AND SYMBOL IS PRODUCED BY THE COMPILER, BUT NO STATEMENT INFORMATION IS PRODUCED.

THE USER MAY PLACE BREAKPOINTS AT THE ENTRY TO AND EXIT FROM PROGRAMS (INCLUDING ALTERNATE ENTRIES), BUT MAY NOT BREAKPOINT AT INDIVIDUAL STATEMENTS. PROGRAM LOCATIONS ARE IDENTIFIED BY DBG TO THE USER BY PROGRAM BLOCK NAME PLUS NUMERICAL OFFSET, EXPRESSED IN OCTAL. (RECALL THAT NO STATEMENT INFORMATION IS PRODUCED BY THE COMPILERS.)

CODE PRODUCED DURING A PRODUCTION MODE COMPILATION IS OPTIMIZABLE. UNLIKE "DEBUG" MODE, NO EXTRA AMOUNT OF SPACE IS USURPED BY THE PROCEDURE TEXT, HOWEVER THE AMOUNT OF SPACE USED IN THE LINKAGE FRAME INCREASES 2 WORDS FOR EACH COMMON BLOCK (EXTERNAL STATIC VARIABLE) AND EXTERNAL ENTRY DECLARED.

THE COMMAND LINE OPTION TO SPECIFY A "PRODUCTION" MODE COMPILATION IS "-PROD".

PRODUCTION MODE IS INCLUDED TO PROVIDE AN INTERFACE BETWEEN DBG AND COMPILER-OPTIMIZED PROGRAMS. AS SOME CONSIDER THAT VERY LARGE PROGRAMS ARE NEVER 'FULLY' DEBUGGED, THE IMPLEMENTORS FELT IT UNREASONABLE TO REQUIRE THAT PROGRAMS BE COMPILED IN "DEBUG" (NON-OPTIMIZED) MODE TO BE USED WITH DBG. HOWEVER, IT IS SUGGESTED THAT, IN ORDER TO TAKE ADVANTAGE OF ALL OF THE FEATURES OFFERED BY

DBG, ALL INITIAL DEBUGGING TAKE PLACE USING THE "DEBUG" COMPILATION MODE.

23.3 "NODEBUG" MODE - PRODUCE NO DEBUGGER INFORMATION

THE "NODEBUG" COMPILATION MODE CAUSES THE COMPILER TO OUTPUT NO SYMBOL AND STATEMENT INFORMATION FOR LATER DIGESTION BY DBG. PROGRAMS COMPILED IN THIS MODE WILL NOT BE RECOGNIZED BY THE DEBUGGER, IN THAT NO STATEMENT, ENTRY, OR EXIT BREAKPOINTS MAY BE SET, NOR MAY ANY VARIABLES BE REFERENCED. THE ONLY MEANS THE DEBUGGER HAS OF IDENTIFYING A PROGRAM COMPILED IN "NODEBUG" MODE TO THE USER (IN A TRACEBACK LISTING, FOR EXAMPLE) IS USING SEGMENT AND WORD MEMORY ADDRESSES.

PROGRAMS COMPILED IN "NODEBUG" MODE MAY BE OPTIMIZED BY THE COMPILER.

THE COMMAND LINE OPTION TO SPECIFY "NODEBUG" MODE TO THE COMPILERS IS "-NODEBUG".

24 COMMAND SUMMARY

THIS SECTION SUMMARIZES BY CLASS ALL DBG COMMANDS AND OPTIONS DESCRIBED IN THIS DOCUMENT.

24.1 DBG INVOCATION OPTIONS

DBG INVOCATION OPTIONS ARE DESCRIBED IN SECTION 7. THE OPTIONS WHICH ARE STARRED ("*") BELOW ARE THE DEFAULTS.

-COMINPUT	ACCEPT INPUT FROM "COMINPUT" FILE
-NO_COMINPUT	DISALLOW INPUT FROM "COMINPUT" FILE (*)
-FULL_INITIALIZE	PERFORM FULL INITIALIZATION
-QUICK_INITIALIZE	PERFORM QUICK (PARTIAL) INITIALIZATION (*)
-POST_MORTEM	ENTER DBG FOLLOWING PROGRAM ERROR
-VERIFY_PROC	VERIFY PROCEDURE TEXT (*)
-NO_VERIFY_PROC	SUPPRESS PROCEDURE TEXT VERIFICATION
-VERIFY_SYMBOLS	VERIFY EXTERNAL SYMBOL DECLARATIONS (*)
-NO_VERIFY_SYMBOLS	SUPPRESS EXTERNAL SYMBOL VERIFICATION

24.2 PROGRAM CONTROL COMMANDS

THE FOLLOWING COMMANDS ARE DESCRIBED IN SECTION 10.

. RESTART [<STEP-COMMAND>]

RESTART PROGRAM EXECUTION

. CONTINUE < PROCEED

CONTINUE PROGRAM EXECUTION (FROM EXECUTION ENVIRONMENT POINTER)

. GOTO [<PROGRAM-BLOCK-NAME> \ [<ACTIVATION-NUMBER> \]] <STATEMENT-IDENTIFIER>

TRANSFER PROGRAM CONTROL TO SPECIFIED STATEMENT UPON CONTINUATION OR SINGLE STEP (SET EXECUTION ENVIRONMENT POINTER)

. MAIN [<PROGRAM-BLOCK-NAME>]

DEFINE OR DISPLAY MAIN PROGRAM

. UNWIND

UNWIND DEBUGGER STACK TO INVOCATION 0

24.3 BREAKPOINT AND TRACEPOINT RELATED COMMANDS

THIS SECTION SUMMARIZES THE BREAKPOINT AND TRACEPOINT RELATED COMMANDS DESCRIBED FULLY IN SECTION 11.

. BREAKPOINT [<BREAKPOINT-IDENTIFIER>] [<ACTION-LIST>]
[-AFTER <VALUE>] [-BEFORE <VALUE>] [-EVERY <VALUE>]
[-COUNT <VALUE>] [-IGNORE < -NIGNORE] [-EDIT]

SET OR MODIFY BREAKPOINT AT SPECIFIED LOCATION

-AFTER	BREAK WHEN COUNTER EXCEEDS <VALUE>
-BEFORE	BREAK WHEN COUNTER IS LESS THAN <VALUE>
-EVERY	BREAK EVERY <VALUE> TIMES THROUGH BREAKPOINT
-COUNT	SET BREAKPOINT COUNTER
-IGNORE	NEVER BREAK AT THIS LOCATION, BUT RETAIN BREAKPOINT INFORMATION
-NIGNORE	ALLOW BREAKPOINTS AT THIS STATEMENT (UNDOES "-IGNORE")
-EDIT	EDIT BREAKPOINT ACTION LIST

. TRACEPOINT [<BREAKPOINT-IDENTIFIER>] [-AFTER <VALUE>]
[-BEFORE <VALUE>] [-EVERY <VALUE>] [-COUNT <VALUE>]
[-IGNORE < -NIGNORE]

SET TRACEPOINT -- (SEE ABOVE FOR ARGUMENT INFORMATION)

. CLEAR [<BREAKPOINT-IDENTIFIER>]

CLEAR BREAKPOINT OR TRACEPOINT

. CLEARALL [<PROGRAM-BLOCK-NAME>] [-DESCEND]
[-BREAKPOINTS < -TRACEPOINTS]

CLEAR ALL BREAKPOINTS AND/OR TRACEPOINTS, EITHER THROUGHOUT THE PROGRAM OR CONTAINED WITHIN A SPECIFIED PROGRAM BLOCK

. LIST [<BREAKPOINT-IDENTIFIER>]

LIST A BREAKPOINT OR TRACEPOINT AND ITS ATTRIBUTES

. LISTALL [<PROGRAM-BLOCK-NAME>] [-DESCEND]
[-BREAKPOINTS < -TRACEPOINTS]

LIST ALL BREAKPOINTS AND/OR TRACEPOINTS, EITHER THROUGHOUT THE PROGRAM OR CONTAINED WITHIN A SPECIFIED PROGRAM BLOCK

24.4 EXPRESSION EVALUATION COMMANDS

THIS SECTION SUMMARIZES THE COMMANDS RELATED TO EXPRESSION EVALUATION AS DESCRIBED IN SECTION 12.

. :[<MODIFIER> [, <MODIFIER>]] <EXPRESSION>

EVALUATE EXPRESSION OR SIMPLE VARIABLE WHERE EACH <MODIFIER> IS A LANGUAGE NAME OR PRINT MODE:

<LANGUAGE-NAME> ::= PL1 < FORTRAN < COBOL

<PRINT-MODE> ::= ASCII < BIT < DECIMAL <
EBCDIC < FLOAT < HEX < OCTAL

. LANGUAGE [PL1 < FORTRAN < COBOL]

DEFINE OR DISPLAY LANGUAGE OF EVALUATION

. PMODE <PRINT-MODE> <VARIABLE> [, <VARIABLE> ...]

DEFINE PRINT MODE OF VARIABLE(S); <PRINT-MODE> MAY BE ONE OF THE FOLLOWING:

ASCII < BIT < DECIMAL < EBCDIC < FLOAT < HEX < OCTAL

. TYPE <EXPRESSION>

DISPLAY DATA TYPE AND ATTRIBUTES OF EXPRESSION OR SIMPLE VARIABLE

. LET <VARIABLE> = <EXPRESSION>

ASSIGN VALUE OF <EXPRESSION> TO <VARIABLE>

. ARGUMENTS [<PROGRAM-BLOCK-NAME> [\ <ACTIVATION-NUMBER>] < <ALTERNATE-ENTRY-IDENTIFIER>]

PRINT VALUES OF ARGUMENTS TO THE SPECIFIED PROGRAM BLOCK OR ALTERNATE ENTRY, OR TO THE PROGRAM BLOCK DESCRIBED BY THE EVALUATION ENVIRONMENT POINTER

24.5 DEBUGGER CONTROL COMMANDS

THE FOLLOWING DEBUGGER CONTROL COMMANDS ARE FULLY DESCRIBED IN SECTION 13.

. ENVIRONMENT [<PROGRAM-BLOCK-NAME> [\ <ACTIVATION-NUMBER>]] < [-POP]

SET EVALUATION ENVIRONMENT TO EITHER SPECIFIED NAME (AND ACTIVATION) OR TO TOP PROGRAM BLOCK ON EVALUATION ENVIRONMENT STACK

. ENVLIST

LIST CONTENTS OF EVALUATION ENVIRONMENT STACK

. IF <EXPRESSION> <ACTION-LIST> [ELSE <ACTION-LIST>]

EXECUTE COMMANDS IN FIRST ACTION LIST IF CONDITION YIELDS TRUE, ELSE EXECUTE COMMANDS IN SECOND ("ELSE") ACTION LIST, IF PRESENT

. ACTIONLIST ;SUPPRESS < PRINT=

CAUSE OUTPUT OF ACTION LIST TO TERMINAL TO BE ENABLED/DISABLED WHEN ACTION LIST IS EXECUTED

. PSYMBOL

PRINT SPECIAL SYMBOLS

. SYMBOL <SYMBOL-NAME> <CHARACTER-VALUE>

SET VALUE OF DBG CHARACTER SYMBOL

. RESUBMIT

EDIT AND RESUBMIT LAST COMMAND LINE FOR EXECUTION

24.6 INFORMATION REQUEST COMMANDS

THE FOLLOWING INFORMATION REQUEST COMMANDS ARE FULLY DESCRIBED IN SECTION 14.

. WHERE [<SEGMENT-NUMBER> / <WORD-NUMBER>]

PRINT VALUE OF EXECUTION ENVIRONMENT POINTER OR PROGRAM LOCATION OF GIVEN SEGMENT AND WORD MEMORY ADDRESS

. SEGMENTS

PRINT LIST OF IN-USE SEGMENTS

. INFO ;<PROGRAM-BLOCK-NAME> \ < <ALTERNATE-ENTRY-IDENTIFIER> < <STATEMENT-IDENTIFIER>=

PRINT DBG INFORMATION FOR GIVEN STATEMENT, PROGRAM BLOCK, OR ALTERNATE ENTRY

. STATUS

PRINT GENERAL DEBUGGER / USER PROGRAM STATUS

24.7 PROCEDURE CALL / RETURN STACK TRACE

THIS SECTION SUMMARIZES THE INFORMATION PRESENTED IN SECTION 15.

. TRACEBACK [-FRAMES <VALUE> [-LEAST_RECENT]] [-FROM <VALUE>] [-TO <VALUE>] [-REVERSE] [-DBG] [-ONUNITS] [-ADDRESSES]

PRINT STACK TRACE

-FRAMES	LIMIT LISTING TO <VALUE> FRAMES
-LEAST_RECENT	PRINT LEAST RECENT <VALUE> FRAMES
-FROM	START TRACEBACK FROM FRAME <VALUE>
-TO	COMPLETE TRACEBACK AFTER FRAME <VALUE>
-REVERSE	PRINT LISTING IN REVERSE ORDER
-DBG	LIST DBG FRAMES IN EXPANDED FORM
-ONUNITS	LIST ON-UNIT INFORMATION IN EACH FRAME
-ADDRESSES	PRINT SEGMENT/WORD ADDRESSES

24.8 SINGLE STEP FACILITY

THIS SECTION SUMMARIZES THE COMMANDS PRESENTED IN SECTION 16.

. STEP [<VALUE>]

SINGLE STEP, ACROSS PROCEDURE CALLS, <VALUE> STATEMENTS (ASSUMED 1 IF OMITTED)

. STEPIN [<VALUE>]

SINGLE STEP, INTO CALLED PROCEDURES, <VALUE> STATEMENTS (ASSUMED 1 IF OMITTED)

. IN

CONTINUE EXECUTION; STOP FOLLOWING NEXT PROCEDURE CALL

. OUT

CONTINUE EXECUTION; STOP FOLLOWING NEXT PROCEDURE RETURN

24.9 VALUE TRACING

THIS SECTION SUMMARIZES THE COMMANDS DESCRIBED IN SECTION 17.

. WATCH <VARIABLE> [, <VARIABLE>...]

ADD VARIABLE(S) TO THE WATCH LIST

. WATCHLIST

PRINT NAMES OF VARIABLES IN WATCH LIST

. UNWATCH <VARIABLE> [, <VARIABLE>...]

REMOVE VARIABLE(S) FROM THE WATCH LIST

. VTRACE ;ON < OFF=

ENABLE / DISABLE VALUE TRACING (LEAVES WATCH LIST UNDISTURBED)

24.10 PROGRAM TRACING

THIS SECTION SUMMARIZES THE COMMANDS DESCRIBED IN SECTION 18.

. FTRACE ;ON < ARGS < OFF=

ENABLE/DISABLE ENTRY AND EXIT TRACING

ON	ENABLE TRACING
ARGS	ENABLE TRACING; PRINT ARGUMENTS ON ENTRY
OFF	DISABLE TRACING

. STRACE ;FULL < QUIET < OFF=

ENABLE/DISABLE STATEMENT TRACING

FULL	ENABLE TRACING ON ALL STATEMENTS
QUIET	ENABLE TRACING ON LABELLED STATEMENTS ONLY
OFF	DISABLE TRACING

24.11 THE "CALL" COMMAND

THE "CALL" COMMAND IS FULLY DESCRIBED IN SECTION 19.

. CALL <VARIABLE> [(ARGUMENT-LIST)]

CALL USER PROCEDURE FROM DEBUGGER COMMAND LEVEL WITH SUPPLIED ARGUMENT LIST, IF ANY

24.12 MISCELLANEOUS COMMANDS

THE FOLLOWING COMMANDS ARE DESCRIBED IN SECTION 20.

. CMDLINE

INVOKE PRIMOS SUBROUTINE "COMANL" TO READ A LINE INTO THE COMMAND LINE BUFFER.

. * [<VALUE>]

REPEAT COMMAND LINE EITHER <VALUE> TIMES, OR "FOREVER" IF <VALUE> IS OMITTED

. HELP

PRINT NAME OF MOST UP-TO-DATE DBG DOCUMENTATION.

. ! <PRIMOS-COMMAND-LINE>

EXECUTE PRIMOS COMMAND LINE FROM DEBUGGER COMMAND LEVEL

. VPSD

INVOKE 64V MODE PRIME SYMBOLIC DEBUGGER

24.13 EDIT FACILITY

THIS SECTION SUMMARIZES THE EDIT FACILITY SUB-COMMANDS USED BY THE "RESUBMIT" AND "BREAKPOINT -EDIT" COMMANDS, AS DESCRIBED IN SECTION 21.

. D	DELETE CHARACTER
. F	SPECIFY FIRST CHARACTER
. L	SPECIFY LAST CHARACTER
. A <TEXT>	APPEND TEXT TO END OF LINE
. I <TEXT>	INSERT TEXT FOLLOWING CHARACTER UNDER WHICH "I" IS POSITIONED
. O <TEXT>	OVERLAY TEXT BEGINNING WITH CHARACTER UNDER WHICH "O" IS POSITIONED
. 0	RETURN TO DBG COMMAND LEVEL

24.14 THE "SOURCE" COMMAND

THIS SECTION SUMMARIZES THE DEBUGGER'S "SOURCE" COMMAND, FULLY DESCRIBED IN SECTION 22.

. SOURCE <SOURCE-COMMAND> [<ARGUMENT>]

EXECUTE ONE OF THE FOLLOWING "ED" COMMANDS:

TOP	NEXT
BOTTOM	MODE
BRIEF	LOCATE
VERIFY	FIND
PRINT	SYMBOL
WHERE	PSYMBOL
POINT	*

OR ONE OF THE FOLLOWING DBG "SOURCE" SUB-COMMANDS:

EX SET SOURCE FILE NAME AND LINE POINTER TO CORRESPOND TO EXECUTION ENVIRONMENT POINTER

NAME [<TREE-NAME> < -DEFAULT]
SET SOURCE FILE NAME TO <TREE-NAME>, OR RETURN TO "DEFAULT" SOURCE FILE NAME, OR PRINT NAME OF CURRENT SOURCE FILE IF NO ARGUMENT

25 DEBUGGING TECHNIQUES USING DBG

THIS SECTION SUGGESTS SOME DEBUGGING TECHNIQUES WHICH MAY BE APPLIED USING THE HIGHER-LEVEL-LANGUAGE DEBUGGER. IT IS BY NO MEANS INTENDED TO BE COMPLETE, BUT RATHER, TO DESCRIBE THE SYMPTOMS OF SOME OF THE MORE COMMON PROGRAMMING ERRORS AND TO SUGGEST WAYS OF ZEROING IN ON THE CAUSES. REMEMBER THAT DEBUGGING TECHNIQUES ARE, FOR THE MOST PART, A PERSONAL SKILL ACQUIRED ONLY THROUGH EXPERIENCE AND UNTOLD AMOUNTS OF FRUSTRATION.

IT IS ASSUMED THAT YOU, THE PROGRAMMER, HAVE A GOOD IDEA OF (1) WHAT YOUR PROGRAM IS SUPPOSED TO DO, AND (2) THE ALGORITHMS AND DATA STRUCTURES WHICH YOUR PROGRAM USES.

WE ASSUME THAT YOUR PROGRAM HAS BEEN COMPILED AND LOADED WITH NO ERRORS. (IF NOT, REFER TO THE APPROPRIATE PRIME DOCUMENTATION FOR AN INTERPRETATION OF THE COMPILER/LOADER ERROR DIAGNOSTICS; YOU SHOULD NOT ATTEMPT TO EXECUTE A PROGRAM WHICH HAS NOT BEEN COMPILED OR LOADED SUCCESSFULLY.) WHEN YOU EXECUTE THE PROGRAM, IT WILL EXHIBIT ONE OF THE FOLLOWING FOUR CHARACTERISTICS:

(1) IT WILL RUN UNTIL COMPLETION AND PRODUCE CORRECT RESULTS.

(2) IT WILL RUN UNTIL COMPLETION AND PRODUCE INCORRECT RESULTS OR NO RESULTS AT ALL.

(3) IT WILL TERMINATE ABNORMALLY, PRINTING AN ERROR DIAGNOSTIC AND RETURNING TO PRIMOS COMMAND LEVEL.

(4) IT WILL NOT TERMINATE.

25.1 CASE 1 - PROGRAM COMPLETES SUCCESSFULLY

SHOULD CASE 1 APPLY, IT IS ASSUMED THAT THE USER WISHES TO USE DBG TO LOOK AT THE FINAL STATE OF THE STATIC VARIABLES DECLARED WITHIN THE PROGRAM. IF THIS IS THE CASE, FIRST EXECUTE THE PROGRAM USING DBG (SEE SECTIONS 7 AND 10). WHEN A MESSAGE APPEARS INDICATING THAT PROGRAM EXECUTION HAS COMPLETED, THE USER MAY INSPECT PROGRAM-DEFINED DATA USING THE ": " (EVALUATE) COMMAND DEFINED IN SECTION 12. (THIS MESSAGE WILL BE "PROGRAM EXECUTION COMPLETE" IF THE MAIN PROGRAM RETURNS, "PROGRAM EXIT FROM <STATEMENT-ID>" IF A PROGRAM BLOCK CALLS "EXIT", OR "PROGRAM STOP AT <STATEMENT-ID>" IF A "STOP" STATEMENT IS EXECUTED.)

25.2 CASE 2 - PROGRAM COMPLETES, PRODUCES INCORRECT RESULTS

IF CASE 2 APPLIES, THE USER MIGHT ATTEMPT TO LOCATE THE PROBLEM BY TRACING PROGRAM EXECUTION AND DATA COMPUTATION.

IF IT IS OBVIOUS TO THE PROGRAMMER THAT THE ERROR LIES WITHIN ONE OR

A FEW STATEMENTS, SETTING BREAKPOINTS AND MONITORING THE VALUES OF VARIABLES WILL PROBABLY IDENTIFY THE SOURCE OF THE PROBLEM QUITE READILY.

IF, ON THE OTHER HAND, THE USER IS STYMIED AS TO WHERE TO BEGIN LOOKING FOR THE PROBLEM, A DIFFERENT TACK IS IN ORDER. OUR FAVORITE DEBUGGING TECHNIQUE IS BASED LOOSELY UPON A SIMPLE BINARY SEARCH ALGORITHM. BREAKPOINTS AND TRACEPOINTS ARE SET SPARSELY THROUGHOUT THE PROGRAM IN AN ATTEMPT TO FIND THE FIRST INDICATION THAT THINGS ARE AWRY. WHEN THIS OCCURS, THE "PROBLEM AREA" IS MORE DENSELY POPULATED WITH BREAKPOINTS UNTIL THE USER FINALLY ZEROES IN ON THE PROBLEM. MORE PRECISELY, THE METHOD IS:

(1) DETERMINE BY DEDUCTION, IF POSSIBLE, THE SECTION OF THE PROGRAM IN WHICH THE PROBLEM EXISTS. IF UNABLE TO LOCALIZE THE PRORLEM, THE ENTIRE PROGRAM IS CONSIDERED "SUSPECT".

(2) CHOOSE NO MORE THAN 8 "KEY" LOCATIONS WITHIN THE SUSPECTED CODE AND PLACE A BREAKPOINT OR TRACEPOINT AT EACH. KEY LOCATIONS ARE STATEMENTS, ENTRIES, AND EXITS AT WHICH THE USER MAY DETECT CORRECT OR INCORRECT PROGRAM OPERATION BY

   (A) INSPECTING THE VALUES OF VARIABLES, OR

   (B) SIMPLY KNOWING THAT PROGRAM CONTROL HAS ARRIVED THERE.

(3) WHEN THE PROGRAM ENCOUNTERS (OR FAILS TO ENCOUNTER) A KEY LOCATION AND THE USER DETECTS SOMETHING AMISS, THE SIZE AND LOCATION OF THE SUSPECTED CODE CAN BE MODIFIED APPROPRIATELY. STEPS 2 AND 3 MAY BE REPEATED UNTIL THE SUSPECTED CODE SPANS BUT A FEW STATEMENTS, AT WHICH TIME THE SIMPLER TECHNIQUE DESCRIBED ABOVE MAY BE EMPLOYED.

25.3 CASE 3 - PROGRAM ABNORMALLY TERMINATES

IF THE PROGRAM TERMINATES ABNORMALLY, THE USER MAY RE-EXECUTE THE PROGRAM USING DBG (SEE SECTIONS 7 AND 10). WHEN THE ERROR CONDITION WHICH CAUSED TERMINATION IS RAISED, THE DEBUGGER REGAINS CONTROL AND RETURNS TO COMMAND LEVEL. THROUGH THE USE OF THE "TRACEBACK" COMMAND, THE USER MAY DETERMINE THE PROGRAM LOCATION AT WHICH THE ERROR OCCURRED.

IF THE USER IS UNABLE TO DETERMINE THE CAUSE OF THE ERROR BY LOOKING AT THE SOURCE CODE FOR THE STATEMENT, THE CONDITION NAME MAY PROVIDE SOME ADDITIONAL INFORMATION.

IF THE ERROR IS THE RESULT OF AN ACCESS VIOLATION (CONDITION "ACCESS_VIOLATION$"), ILLEGAL SEGMENT REFERENCE ("ILLEGAL_SEGNO$"), POINTER FAULT ("POINTER_FAULT$"), OR ILLEGAL PAGE REFERENCE ("OUT_OF_BOUNDS$"), THE PROGRAM WAS PROBABLY ATTEMPTING TO

- REFERENCE THROUGH A POINTER, LABEL, OR ENTRY VARIABLE WHICH CONTAINS GARBAGE (I.E. HAS NOT BEEN INITIALIZED),
- REFERENCE A SUBROUTINE ARGUMENT WHICH HAS NOT BEEN SUPPLIED BY THE CALLER,
- CALL A PROCEDURE WHICH HAS NOT BEEN LOADED, OR
- REFERENCE AN EXTERNAL SYMBOL WHICH HAS NOT BEEN DEFINED.

AN ERROR OF THE TYPE DESCRIBED ABOVE MAY ALSO BE THE INDIRECT RESULT OF REFERENCES TO OUT OF BOUNDS ARRAY ELEMENTS. (IF THIS IS SUSPECT AND THE COMPILER HAS THE CAPABILITY TO GENERATE CODE TO CHECK FOR OUT OF BOUNDS ARRAY REFERENCES, THE USER IS ENCOURAGED TO USE IT; A COMPARABLE FACILITY IS UNAVAILABLE IN DBG.)

IF THE ERROR RESULTED FROM AN ILLEGAL OR RESTRICTED INSTRUCTION (CONDITIONS "ILLEGAL_INST$" AND "RESTRICTED_INST$"), THE USER PROGRAM EITHER INTENTIONALLY OR INADVERTANTLY EXECUTED AN INSTRUCTION WHICH WAS UNDEFINED ("ILLEGAL") OR UNABLE TO BE EXECUTED IN RINGS 1 OR 3 ("RESTRICTED"). IT IS MOST FREQUENTLY THE CASE THAT THE USER HAS OVERWRITTEN A LOCATION IN HIS PROGRAM VIA AN OUT OF BOUNDS ARRAY REFERENCE OR GARBAGED POINTER, OR HAS CAUSED CONTROL TO BE TRANSFERRED TO AN INCORRECT LOCATION. THE LATTER MAY BE DONE IN A VARIETY OF WAYS; OUR FAVORITE IS "GOING TO" AN UNINITIALIZED LABEL VARIABLE IN PL/1 OR SUPPLYING AN ILLEGAL ALTERNATE RETURN IN A FORTRAN SUBROUTINE CALL. WHEN CONFRONTED WITH THIS ERROR, A PROCEDURE CALL STACK TRACE USING THE DBG "TRACEBACK" COMMAND IS USUALLY A GOOD PLACE TO START.

IF THE ERROR OCCURS AS THE RESULT OF A NULL POINTER (CONDITION "NULL_POINTER$"), THE EXPLICIT OR IMPLIED POINTER REFERENCE IN THE FAULTING STATEMENT WAS ATTEMPTED THROUGH A POINTER WHOSE VALUE WAS, IN THE PL/1 SENSE, NULL (I.E., THE SEGMENT NUMBER WAS '7777).

IF THE ERROR WAS THE RESULT OF A LINKAGE FAULT (CONDITION "LINKAGE_FAULT$"), THE USER PROGRAM ATTEMPTED TO EXECUTE A PROCEDURE CALL TO A ROUTINE WHICH WAS NOT DEFINED IN THE OPERATING SYSTEM OR IN A SHARED LIBRARY. CHECK WITH THE SYSTEM ADMINISTRATOR AS TO THE WHEREABOUTS OF THE MISSING ROUTINE.

AN ARITHMETIC EXCEPTION ERROR (CONDITION "ARITH$") WILL OCCUR WHEN THE CPU DETECTS AN ILLEGAL ARITHMETIC OPERATION, SUCH AS AN EXPONENT OVERFLOW OR A DIVIDE BY ZERO. AN EXPLANATORY MESSAGE IS PRINTED BY DBG, INDICATING THE CAUSE OF THE CONDITION AND THE ACTION TAKEN BY THE DEBUGGER (IF ANY) TO ALLOW CONTINUATION OF EXECUTION. FIND THE SOURCE STATEMENT AND INSPECT THE VALUES OF THE WHICH COMPRISE THE EXPRESSION (IF NECESSARY) TO DETERMINE THE CAUSE.

A BAD NONLOCAL GOTO ERROR (CONDITION "BAD_NONLOCAL_GOTO$") INDICATES THAT THE USER ATTEMPTED TO "GO TO" A LABEL VARIABLE WHICH CONTAINED AN ILLEGAL DISPLAY (STACKFRAME) POINTER. FIND THE SOURCE STATEMENT AND, USING THE DBG EVALUATE COMMAND, EXAMINE THE LABEL VARIABLE. THE USER MIGHT CHOOSE TO EMPLOY THE VALUE TRACE FACILITY TO IDENTIFY THE LOCATIONS WITHIN THE PROGRAM THAT THE VARIABLE IS MODIFIED.

WE HAVE ATTEMPTED TO COVER THE MOST COMMON CONDITIONS WHICH MAY BE SIGNALLED BY THE OPERATING SYSTEM IN THE EVENT OF A PROGRAM ERROR. SYSTEM-SIGNALLED CONDITIONS WHICH DO NOT APPEAR HERE ARE DESCRIBED IN PE-T-468, REVISION 2, ENTITLED "SPECIFICATIONS FOR THE PRIMOS CONDITION MECHANISM".

25.4 CASE 4 - PROGRAM FAILS TO TERMINATE

CASE 4 APPLIES IF, AFTER A REASONABLE PERIOD OF TIME AS DETERMINED BY THE PROGRAMMER, THE PROGRAM FAILS TO TERMINATE. WE SHALL CONSIDER THIS CASE THE SO-CALLED "INFINITE LOOP".

AS IS DONE WITH THE OTHER CASES DESCRIBED ABOVE, THE USER SHOULD RE-EXECUTE THE PROGRAM UNDER CONTROL OF THE DEBUGGER. AFTER ALLOWING IT TO EXECUTE A SUFFICIENT AMOUNT OF TIME (PRESUMABLY, TO PERMIT IT TO ENTER THE INFINITE LOOP), THE USER MAY TYPE THE 'QUIT' KEY (BREAK OR CONTROL-P) TO CAUSE CONTROL TO RETURN TO THE DEBUGGER.

BY NOTING THE VALUE OF THE EXECUTION ENVIRONMENT POINTER, OPTIONALLY PERFORMING A TRACEBACK, CONTINUING EXECUTION, AND QUITTING AGAIN, THE USER SHOULD BEGIN TO GET SOME IDEA OF THE AREA IN WHICH THE LOOP EXISTS. (THIS LAST STEP SHOULD BE REPEATED AS MANY TIMES AS NECESSARY.)

THE AREA OF THE LOOP MAY BE FURTHER REFINED BY SETTING TRACEPOINTS AND BREAKPOINTS AT KEY LOCATIONS WITHIN THE SUSPECTED CODE AND CONTINUING EXECUTION.

99 INDEX