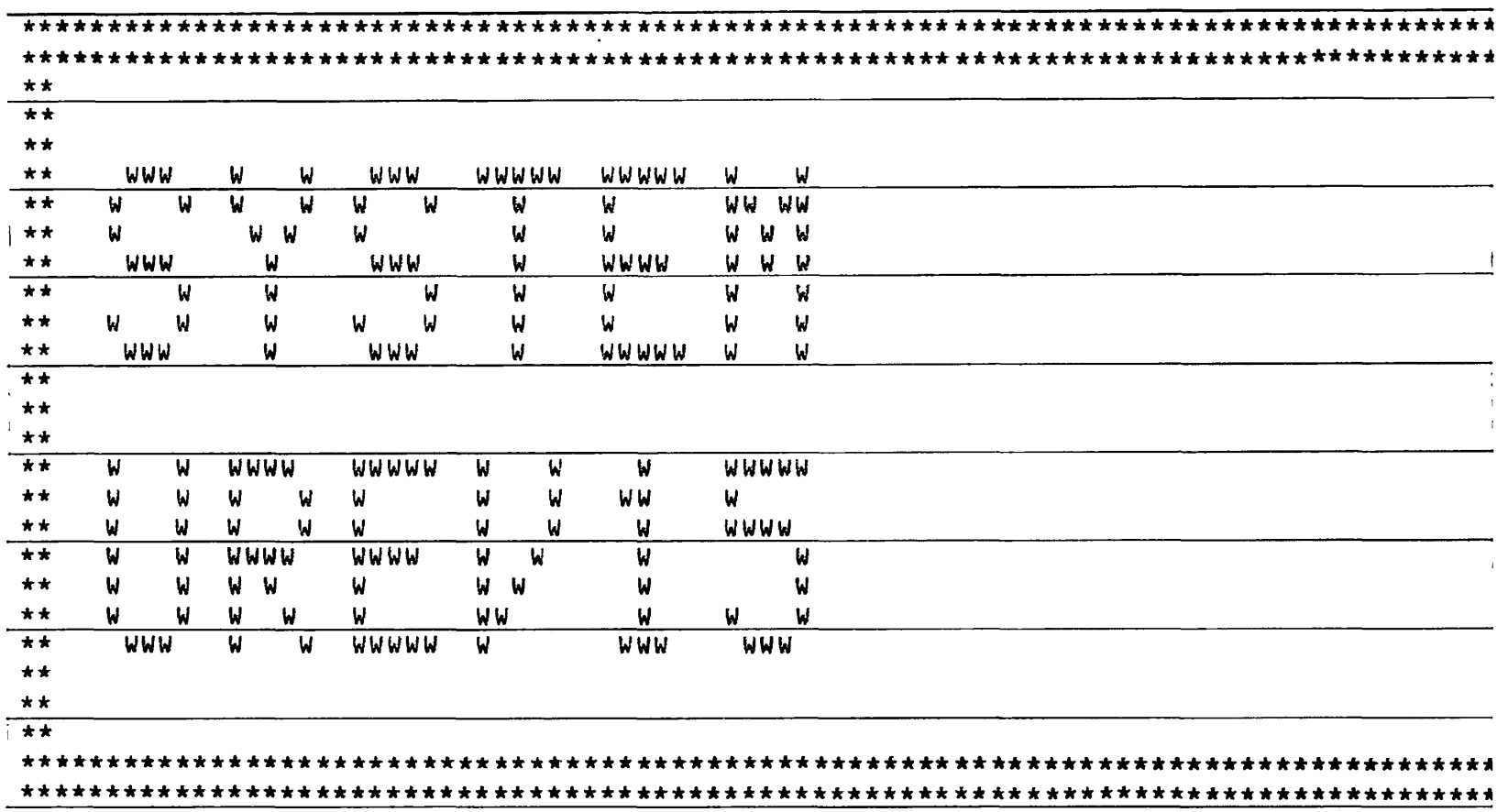

WWW W W WWW WWWWW WWWWW W W
W W W W W W W W W WW WW
W W W W W W W W W W W W
WWW W WWW W WWWW W W W
W W W W W W W W W
W W W W W W W W W W W
WWW W WWW W WWWWW W W
W W WWWW WWWWW W W W WWWWW
W W W W W W W W WW W
W W W W W W W W W WWWW
W W WWWW WWWW W W W W
W W W W W W W W W W
W W W W W WW W W W
WWW W W WWWWW W WWW WWW

DATE: MARCH 15, 1978

SUBJECT: BASICV COMPILER

INTRODUCTION

BASICV IS A SHARED SUBSYSTEM THAT IMPLEMENTS A COMPILED FORM OF THE BASIC LANGUAGE. BASICV CONTAINS THE FEATURES OF THE BASIC INTERPRETER (BASIC,DBASIC) AS A SUBSET. OLD PROGRAMS WRITTEN IN INTERPRETIVE BASIC SHOULD RUN UNDER BASICV WITHOUT MODIFICATION, BUT NOTE THAT DOUBLE-PRECISION FLOATING-POINT NUMBERS (AS IN DBASIC) ARE THE ONLY ONES SUPPORTED BY BASICV.

BASICV SUPPORTS A NUMBER OF FACILITIES NOT AVAILABLE UNDER THE INTERPRETER. THESE INCLUDE DYNAMIC ARRAY STORAGE, IMPROVED STRING-HANDLING CAPABILITY, LARGER AVAILABLE DATA SPACE (ONE SEGMENT), MULTI-LINE USER-DEFINED FUNCTIONS, ERROR-TRAP CONTROL, STRUCTURED CONDITIONALS (DO-DOEND, ETC.), INCREASED FILE CONTROL, AND HIGH-SPEED (COMPILED) EXECUTION.

USE_OF_BASICV

BASICV MAY BE INVOKED BY TYPING THE COMMAND BASICV AT PRIMOS COMMAND LEVEL. THE RESPONSE WILL BE A VERSION PRINTOUT FOLLOWED BY THE QUESTION 'NEW OR OLD ?'. ONE RESPONDS TO THIS BY TYPING EITHER NEW OR OLD, OPTIONALLY FOLLOWED BY A NON-EXISTENT OR PRE-EXISTING FILENAME, RESPECTIVELY. THE SUBSYSTEM THEN BRINGS YOU TO BASICV COMMAND LEVEL, AT WHICH POINT YOU CAN EDIT, COMPILE, AND RUN YOUR PROGRAM. ALTERNATIVELY, YOU MAY RUN A PROGRAM DIRECTLY FROM THE PRIMOS COMMAND LEVEL BY TYPING BASICV FOLLOWED BY A PROGRAM FILENAME. THIS PROGRAM MAY BE EITHER A SOURCE FILE OR A COMPILED BINARY FILE. THE FILE TYPE WILL BE SENSED BY BASICV AND EITHER COMPILED THEN EXECUTED OR SIMPLY EXECUTED DIRECTLY.

THE BASICV COMPILER AND RUN-TIME HANDLER ARE BOTH LOCATED IN SHARED SPACE (SEGMENT '2013). THIS MAKES THE SYSTEM QUITE FAST AND EFFICIENT SINCE ONLY A SMALL AMOUNT OF DATA MUST BE RESTORED TO SEGMENT '4000. NOTE THAT COMPILED BASIC CODE RUNS IN SEGMENT '4000 AND IS NOT SHAREABLE. BEWARE ALSO THAT, IN ORDER FOR BASICV TO RUN, BOTH THE COMMAND MUST APPEAR IN CMDNCO AND THE SHARED PART (BA2013) MUST BE INSTALLED BY USE OF THE SHARE COMMAND FROM THE SYSTEM TERMINAL.

DOCUMENTATION

THE SET OF BASICV COMMANDS AND STATEMENTS IS DESCRIBED IN DETAIL IN THE BASICV MANUAL, IDR-3058, WHICH IS AVAILABLE IN THE STOCKROOM.

DATE: MARCH 9, L978

SUBJECT: CHECKPOINT/RESTART RULES FOR REV. 15

INTRODUCTION

THE USER OF PROGRAMS WHICH RUN FOR LONG PERIODS OF TIME OR WHICH MODIFY DATA AND THEN REUSE THE SAME DATA MAY SOMETIMES NEED TO RECOVER FROM DISASTERS SUCH AS POWER FAILURES. IN ORDER TO DO SO, THE "STATE OF THE SYSTEM" MUST BE WRITTEN OUT PERIODICALLY. THE PROGRAM MAY THEN BE RESTARTED AT THE POINT WHERE THE "STATE OF THE SYSTEM" WAS LAST SAVED. THE POINT WHERE THE "STATE OF THE SYSTEM" WAS SAVED IS CALLED THE CHECKPOINT. THE PROCESS OF SAVING THE "STATE OF THE SYSTEM" IS CALLED CHECKPOINTING. THE PURPOSE OF THIS NOTE IS TO INDICATE SOME OF THE PROBLEMS AND REQUIREMENTS OF CHECKPOINTING AND RESTARTING. IT IS AIMED AT PROGRAMS WRITTEN IN FORTRAN AND COBOL. AS AN EXAMPLE AND INTRODUCTION, A METHOD OF CHECKPOINTING COMINPUT OR PHANTOM JOBS WITH EMPHASIS ON CX IS PRESENTED. RECOVERY IN THE DBMS AND MIDAS SYSTEMS ARE ALSO DISCUSSED.

THE CASE OF THE PHANTOM JOB SUBMITTED BY THE USER OR BY THE CX QUEUEING PROCESS IS RELATIVELY SIMPLE. IT REQUIRES JUST ONE PROGRAM. THE PROGRAM IS A COMMAND EXECUTED IN THE USER'S COMMAND FILE WHENEVER A CHECKPOINT IS TO BE WRITTEN OUT. THE SAME PROGRAM HANDLES RESTART.

FORTRAN AND COBOL PROGRAMS WILL USUALLY REQUIRE THE USER TO DESIGN CHECKPOINT AND RESTART FACILITIES AS PART OF HIS PROGRAM. IT IS BEST TO KEEP THE CHECKPOINT AND RESTART REQUIREMENTS IN MIND WHEN INITIALLY DESIGNING A PROGRAM BUT SUCH FEATURES CAN BE ADDED LATER. CHECKPOINT AND RESTART ARE COMPLICATED BY THE EXISTENCE OF LANGUAGE VARIABLES HIDDEN FROM THE USER AND THE NECESSITY OF REPOSITIONING I/O DEVICES TO THEIR PROPER PLACES. METHODS OF ACCOMPLISHING THIS ARE DICUSSED.

DBMS IS SELF CONTAINED AND CAN ROLL BACK OR UNDO A TRANSACTION WHICH WAS INTERRUPTED BY A BREAK OR OTHER SYSTEM INTERRUPTION.

MIDAS IS ALSO SELF CONTAINED, HOWEVER, IT HAS NO ROLLBACK FACILITY. IT DOES HAVE A TOOL TO AID RECOVERY WHEN THE DATA FILE IS DAMAGED. COMPLETE RECOVERY IS NOT GUARANTEED IF INTERRUPTION OCCURS WHILE ADDING A PRIMARY INDEX.

FORMS IS MENTIONED AND MANUAL SPOOLING OF ITS PRINT OUTPUT FILE IS DESCRIBED.

CX

AS AN INTRODUCTION TO CHECKPOINTING IN PRIMOS, THE EASIEST SYSTEM TO DISCUSS IS THE CX INITIATED PHANTOM JOB. THIS SYSTEM ILLUSTRATES SOME OF THE REQUIREMENTS AND PROBLEMS OF CHECKPOINTING IN GENERAL BUT CAN BE DESCRIBED IN A FAIRLY SIMPLE MANNER. THE TECHNIQUE USED HERE CAN BE USED FOR ANY PHANTOM OR COMINPUT JOB SINCE THEY SHARE A COMMON INPUT FORMAT. ONE DIFFERENCE WILL BE MENTIONED LATER.

THE CX SYSTEM IS A QUEUEING SYSTEM WHICH IS USED TO SUBMIT PHANTOM JOBS. THE FORMAT OF THE PHANTOM INPUT FILE IS ONE COMMAND PER LINE. IF A COMMAND SUCH AS LOAD REQUIRES FURTHER USER INPUT, IT ALSO IS IN THE COMMAND FILE.

A PHANTOM JOB SUBMITTED BY CX AND RUNNING WHEN A CRASH OCCURS STARTS FROM THE BEGINNING WHEN THE CX SYSTEM IS AGAIN STARTED UP. IF DATA HAS BEEN MODIFIED OR A LARGE NUMBER OF COMPILATIONS COMPLETED, THEN THE JOB SHOULD BE RESTARTED NOT AT THE BEGINNING BUT AT SOME LATER COMMAND IN THE INPUT FILE. TO DO THIS A NEW MECHANISM MUST BE INVENTED. THAT MECHANISM MUST DO THREE THINGS.

1. IT MUST INDICATE WHERE THE FILE IS TO BE RESTARTED, I.E., TAKE PERIODIC CHECKPOINTS.

2. IT MUST RECOGNIZE THAT THE JOB IS BEING RESTARTED.

3. IT MUST RESTORE THE STATE OF THE SYSTEM AND START THE JOB AFTER THE LAST CHECKPOINT.

SUCH A MECHANISM WILL NOW BE DESCRIBED.

THE RESTART MECHANISM REQUIRES A COMMAND WHICH WILL BE CALLED CHKPT. IT IS EXECUTED AT EACH DESIRED CHECKPOINT. A LISTING OF CHKPT IS SHOWN IN APPENDIX 1. THE CHKPT COMMAND TAKES TWO ARGUMENTS. THE FIRST IS THE TREENAME OF A CHECKPOINT FILE. THIS IS THE FILE WHERE THE RESTART DATA IS STORED. (THE PROGRAM SHOWN IN THE APPENDIX ASSUMES A TREENAME OF THE FORM FOO>BARR AND APPENDS THE CHARACTERS ".CHKPT" TO MAKE FOO>BAR.CHKPT). THE SECOND ARGUMENT IS A FLAG INDICATING THE FIRST EXECUTION OF CHKPT IN THE COMMAND FILE. IT IS ASSUMED THAT NO CHECKPOINT FILE EXISTS THE FIRST TIME THE COMMAND FILE IS EXECUTED. WHEN CHKPT IS EXECUTED, IT FIRST CHECKS TO SEE IF A CHECKPOINT FILE EXISTS. IF NOT, THEN ONE IS CREATED AND THE CURRENT POSITION OF THE FILE UNIT ASSOCIATED WITH THE COMMAND FILE, THE PHANTOM INPUT FILE, IS WRITTEN INTO THE CHECKPOINT FILE. IF THE CHECKPOINT FILE EXISTS BUT THE FIRST FLAG ARGUMENT IS MISSING THEN THE CURRENT POSITION IN THE COMMAND FILE IS SAVED AS BEFORE. FINALLY, CHKPT RECOGNIZES A RESTART SITUATION BY FINDING THAT A CHECKPOINT FILE EXISTS AND SEEING THE FIRST FLAG ARGUMENT. IT THEN USES THE POSITION IN THE CHECKPOINT FILE TO POSITION THE FILE UNIT ASSOCIATED WITH THE COMMAND FILE. WHEN CHKPT RETURNS, THEN NEXT COMMAND EXECUTED IS THE ONE AT THE NEW POSITION.

CONSIDER, FOR EXAMPLE, THE COMMAND FILE

```
A FOO
R CHANGE_DATA  DATA_FILE
R NEW_CHANGE_DATA  DATA_FILE
CX -E
```

THE COMMAND CHANGE_DATA AND NEW_CHANGE_DATA MODIFY THE SAME DATA FILE, DATA_FILE. IF NEW_CHANGE_DATA HAS STARTED THEN THE AIM IS TO PREVENT RUNNING CHANGE_DATA ON A RESTART. THE CHKPT COMMAND IS INSERTED INTO THE COMMAND FILE AT APPROPRIATE PLACES IN ORDER TO ACCOMPLISH THIS. WITH THE NEW COMMANDS INSERTED THE FILE IS AS SHOWN BELOW.

```
A FOO
R CHKPT FOO>CHUFD>CHANGE -FIRST
R CHANGE_DATA DATA_FILE
R CHKPT FOO>CHUFD>CHANGE
R NEW_CHANGE_DATA DATA_FILE
R CHKPT FOO>CHUFD>CHANGE
CX -E
```

THE FIRST TIME THIS COMMAND FILE IS RUN, A CHECKPOINT FILE, CHANGE.CHKPT WILL BE CREATED IN UFD FOO>CHUFD. THE CURRENT POSITION OF THE PHANTOM FILE UNIT IS READ AND SAVED INTO CHANGE.CHKPT. THE RECORDED POSITION IS THE POSITION OF THE BEGINNING OF THE LINE AFTER THE CHKPT COMMAND.

THE SECOND TIME CHKPT IS EXECUTED IN THE COMMAND FILE, IT FINDS CHANGE.CHKPT AND NOTICES THAT THERE IS NO "-FIRST" ARGUMENT. THIS TIME IT MERELY READS THE CURRENT POSITION OF THE PHANTOM INPUT FILE, RECORDS IT AND RETURNS.

IF EXECUTION OF THE COMMAND FILE IS INTERRUPTED AND THE COMMAND FILE IS RESTARTED FROM THE BEGINNING, THE CHKPT COMMAND FINDS THE CHECKPOINT FILE, FOO>CHUFD>CHANGE.CHKPT. THE CHKPT COMMAND ALSO NOTES THE "-FIRST" ARGUMENT AND THEREFORE READS THE POSITION OUT OF FOO>CHUFD>CHANGE.CHKPT, REPOSITIONS THE PHANTOM INPUT FILE TO THE SAVED VALUE AND RETURNS. UPON RETURN, THE INPUT FILE HAS BEEN REPOSTIONED TO THE LINE AFTER THE LAST CALL TO CHKPT IN THE PREVIOUS RUN. IN THE EXAMPLE ABOVE, THE NEXT LINE COULD BE

```
R CHANGE_DATA DATA_FILE
```

OR

```
R NEW_CHANGE_DATA DATA_FILE
```

OR

```
CX -E .
```

THE LAST CHKPT COMMAND WAS INSERTED SO THAT NOTHING WILL BE DONE IF ALL THE DESIRED WORK HAS BEEN COMPLETED BUT CX HAS NOT YET REMOVED THIS JOB FROM ITS TABLES OF RUNNING JOBS.

THE FILE FOO>CHUFD>CHANGE.CHKPT MUST BE DELETED OR ITS NAME CHANGED IN ORDER TO RERUN THIS COMMAND FILE FROM THE BEGINNING.

THIS MECHANISM ILLUSTRATES HOW PART OF THE JOB OF CHECKPOINTING IS DONE.

THERE ARE THREE ISSUES WHICH HAVE NOT BEEN DEALT WITH TO THIS POINT.

1. HOW ARE OPEN FILES SUCH AS LISTING OR BINARY FILES HANDLED?

2. HOW IS THE GENERAL PROBLEM OF "HIDDEN VARIABLES" SUCH AS THE FILE UNIT NUMBER OF THE INPUT FILE HANDLED?

3. WHAT HAPPENS INSIDE A COMMAND ON A SYSTEM INTERRUPTION AND HOW CAN THE COMMAND BE RESTARTED?

THE CHKPT COMMAND CAN BE EXPANDED TO HANDLE THE FIRST TWO DIFFICULTIES. OPEN DISK FILES CAN BE TREATED AS THE PHANTOM INPUT FILE ABOVE. THE NECESSARY DATA TO OPEN AND POSITION THEM CAN BE SUPPLIED BY OPTIONAL ARGUMENTS CONTAINING TREENAMES, UNIT NUMBERS AND OPEN MODES. MAGNETIC TAPE OUTPUT CAN BE CHECKPOINTED BY WRITING AN END OF FILE ON THE TAPE AND BY SPACING TO THE END OF FILE ON RESTART. FOR MAGNETIC TAPE INPUT, THE COOPERATION OF THE PROGRAMS WHICH READ THE TAPE IS NECESSARY. THEY MUST RECORD, IN A SEPARATE FILE, THE NUMBER OF FILES AND RECORDS READ SO THAT THIS CAN BE SAVED BY THE CHKPT COMMAND AND THE TAPE REPOSITIONED PROPERLY. THIS FUNCTION IS MORE REASONABLY A PART OF THE THE CHECKPOINT FUNCTION INSIDE OF THE COMMAND ITSELF.

THE ONLY HIDDEN VARIABLES FOR CHKPT ARE POSSIBLY THE PHANTOM UNIT NUMBER AND THE CURRENT ERASE AND KILL CHARACTERS ESTABLISHED BY THE USER. THE PHANTOM UNIT NUMBER CAN BE SET BY THE USER IF THE PHANTOM IS SUBMITTED BY THE USER HIMSELF AND NOT BY CX. THE PHANTOM UNIT NUMBER CAN BE HANDED TO CHKPT VIA AN ARGUMENT. THE DEFAULT PHANTOM OR COMINP UNIT NUMBER IS SIX. THE ERASE AND KILL CHARACTERS CAN BE READ BY THE CHKPT COMMAND USING THE ERKL$$ SUBROUTINE. THE ERKL$$ SUBROUTINE WAS DOCUMENTED IN THE REV 14 RELEASE. ITS CALLING SEQUENCE IS

```
CALL ERKL$$(KEY,ERASE,KILL,CODE)
```

WHERE KEY = 1 FOR READ AND 2 FOR SET THE ERASE AND KILL CHARACTERS. THE CHARACTERS ARE ZERO FILLED AND RIGHT JUSTIFIED IN THEIR RESPECTIVE WORDS.

RECOVERY INSIDE A COMMAND IS THE SUBJECT OF THE REST OF THIS DOCUMENT.

FORTRAN

THE DESIGN OF CHECKPOINTS VARIES IN DIFFICULTY DEPENDING NOT ONLY ON THE PROGRAM BEING CHECKPOINTED, BUT ALSO ON THE LANGUAGE IN WHICH THE PROGRAM IS WRITTEN. DIFFERENT LANGUAGES HIDE MORE OR LESS INFORMATION FROM THE USER.

FORTUNATELY, VERY LITTLE INFORMATION IS HIDDEN FROM THE FORTRAN PROGRAMMER.

IN ORDER TO CHECK POINT HIS PROGRAM, THE USER MUST KEEP A RECORD OF THE FOLLOWING:

1. USER INTERNAL VARIABLES, COUNTER VARIABLES, TOTALS, INDICES AND FLAGS WHICH AFFECT FURTHER OPERATION OF HIS PROGRAM.

2. I/O DEVICES WHICH ARE OPEN AND THEIR POSITIONS.

3. SYSTEM PARAMETERS.

SINCE PROGRAMS VARY WIDELY, IT IS IMPOSSIBLE TO DESCRIBE A SIMPLE ALGORITHM TO DO THE CHECKPOINTING SUCH AS THE CHKPT COMMAND FOR CX. ONLY THE DESIGNER KNOWS WHICH INDICES, COUNTERS, FLAGS AND TOTALS MUST BE SAVED IN HIS CHECKPOINT DESIGN. HERE, THE USER WILL BE GIVEN THAT ADDITIONAL INFORMATION ABOUT FORTRAN WHICH IS NECESSARY FOR THE DESIGN OF THE CHECKPOINT PROGRAM.

FORTRAN HAS NO HIDDEN COUNTERS OR VARIABLES. THE ONLY DIFFICULTIES INSIDE OF FORTRAN OCCUR IN FORTRAN FORMATTED I/O.

ALL I/O FROM DISK USING FORTRAN FORMATTED I/O OR USING PRWF$$, THE STANDARD FILE SYSTEM MODULE, CAN BE HANDLED SIMILAR TO THE CHKPT PROGRAM SHOWN ABOVE FOR CX. THE STANDARD FILE SYSTEM MODULE, PRWF$$, CAN BE USED TO FIND THE CURRENT POSITION OF ANY FILE UNIT. USING THE POSITION IN THE FILE TOGETHER WITH THE TREENAME OF THE FILE, ITS OPEN MODE AND THE SRCH$$ MODULE ANY SUCH FILE UNIT CAN BE CONNECTED TO ITS CORRESPONDING FILE, OPENED AND POSITIONED FOR RESTART. THUS, FOR EACH OPEN FILE UNIT, THE UNIT, ITS MODE OF OPENING ITS ASSOCIATED TREENAME (INCLUDING PASSWORDS) AND ITS POSITION MUST BE RECORDED IN ANY CHECKPOINT FILE.

THE HANDLING OF FORTRAN FORMATTED I/O FOR MAGNETIC TAPE IS LESS ELEGANT.

A READ STATEMENT WILL CAUSE ONE RECORD TO BE INPUT UNLESS THE FORMAT STATEMENT CONTAINS SLASHES. AS STATED IN THE FORTRAN PROGRAMMERS GUIDE, A SLASH (/) CONTROL CHARACTER MEANS PROCEED TO THE NEXT RECORD. A READ OR WRITE WITH THE FORMAT STATEMENT

```
10 FORMAT(I2,I2)
```

WILL CAUSE A SINGLE RECORD TO BE READ OR WRITTEN. A READ OR WRITE WITH THE FORMAT STATEMENT

```
20 FORMAT(I2/I2)
```

WILL CAUSE TWO RECORDS TO BE READ OR WRITTEN.

THE USER WHO WISHES TO CHECKPOINT AND RESTART HIS PROGRAM WILL HAVE TO RECORD THE NUMBER OF TAPE RECORDS READ OR WRITTEN AND SPACE EACH TAPE TO THE PROPER RECORD. DURING RESTART, A TAPE BEING READ WILL BE OPENED FOR READ WHILE A TAPE BEING WRITTEN WILL BE OPENED FOR READ AND WRITE. A CALL TO C$MO5 IN THE FORM

CALL C$M05 (5, 0, UNIT NUMBER, 0)

WILL SPACE THE TAPE FORWARD ONE RECORD (SEE REFERENCE GUIDE, SOFTWARE LIBRARY, PDR3106, SECTIONS 10 AND 18). FOR 7-TRACK TAPES, THE CALL SHOULD BE

CALL C$M07 (5,0, UNIT NUMBER, 0).

THE CALLS WILL BE IN A LOOP WITH THE COUNT BEING TAKEN FROM THE USER CHECKPOINT FILE. THE USE OF C$M05 OR C$M07 TO SPACE DOWN THE TAPE IS MORE EFFICIENT THAN USING A FORTRAN READ STATEMENT.

CARDS CAN BE READ IN A LOOP BUT FOR CARDS IT WOULD SEEM EASIER TO SIMPLY RERUN THE DECK FROM THE NEXT CARD.

THERE ARE OTHER TYPES OF I/O THAT COULD BE IN USE. THE USER COULD HAVE HIS OWN DISK DRIVER TO ACCESS AN ASSIGNED DISK OR HIS OWN TAPE DRIVER TO ACCESS A MOUNTED TAPE. IN SUCH A CASE IT US UP TO THE USER TO KEEP TRACK OF DISK OR TAPE POSITION IN ORDER TO PROVIDE FOR RESTART.

SYSTEM PARAMETERS WHICH A USER MAY WANT TO SAVE AND RESET FOR HIMSELF ARE THE ERASE OR KILL CHARACTERS. THESE PARAMETERS HAVE EFFECT ONLY ON TERMINAL INPUT PROCESSORS SUCH AS THE EDITOR, ED. STILL, IF THE USER DOES TERMINAL INPUT PROCESSING OR INPUT FROM A COMMAND FILE, IT WOULD BE WISE TO SAVE THE CURRENT ERASE AND KILL CHARACTERS AND TO RESET THE ON RESTART.

SUBROUTINES GIVE NO PARTICULAR DIFFICULTIES IN FORTRAN. THE USE OF COMMON ALLOWS SUBROUTINES TO SHARE DATA OR CONTROL INFORMATION WHICH MUST BE SAVED ON CHECKPOINT AND RESTORED IN RESTART. THIS ALSO MAKES IT EASY TO ISOLATE CHECKPOINT AND RESTART CODE IN THE USER'S PROGRAM OR A SUBROUTINE.

TWO QUESTIONS REMAIN TO BE ANSWERED, WHEN TO CHECKPOINT AND HOW TO AVOID AN INVALID CHCKPOINT. THE FIRST QUESTION HAS BEEN TOUCHED ON ABOVE. THE SECOND QUESTION APPLIES WHEN INTERRUPTION OCCURS WHILE WRITING A CHECKPOINT FILE. BOTH QUESTIONS WILL ALSO BE ANSWERED BELOW.

COBOL

THE COBOL LANGUAGE IN REV. 15 PROVIDES THE USER WITH LITTLE HELP FOR CHECKPOINTING HIS JOB. THERE ARE HIDDEN VARIABLES AND THERE IS NO WAY, AT PRESENT, TO READ FILE UNITS AND REPOSITION EITHER DISKS OR TAPES DIRECTLY ON RESTART.

THE PROBLEM OF HIDDEN VARIABLES SHOWS UP VERY CLEARLY IN TWO CASES. THE MOST COMMON CASE IS THAT OF THE PERFORM VERB. THE SENTENCE

"PERFORM B 5 TIMES"

WILL USE A HIDDEN INDEX TO COUNT UP TO FIVE. THIS INDEX IS NOT AVAILABLE TO THE USER. THE RETURN ADDRESS AFTER FINISHING THE

PERFORMED PARAGRAPH IS ALSO UNKNOWN TO THE USER. USERS WHO WISH TO CHECKPOINT ARE ADVISED NOT TO DO SO IN THE RANGE OF SUCH A PERFORM. IF A USER WISHES TO DO A CHECKPOINT INSIDE THE RANGE OF A PERFORM, THEN THE USER MUST SET UP HIS OWN INDEX TO COUNT THE TIMES THE PERFORM WAS DONE AND SAVE IT IN HIS CHECKPOINT FILE. IN ADDITION, THE FIRST ACTION INSIDE OF THE PARAGRAPH OR SECTION NAME "B" IN THE SENTENCE ABOVE, MUST BE A TEST FOR A RESTART FLAG. A TEST ON THE USERS INDEX OF PASSES THROUGH "B" MUST FOLLOW WITH A TRANSFER TO THE END OF "B" UNTIL THE PROPER COUNT HAS BEEN REACHED. AFTER ALL OF THIS, THE USER IS STILL FACED WITH THE PROBLEM OF CONTINUING FROM THE INSTRUCTION FOLLOWING THAT WHERE THE PERFORM WAS INVOKED. MORE GENERAL FORMS OF PERFORM USING THRU, BY OR WHILE OR OTHER OPTIONS MAKE IT ALMOST IMPOSSIBLE TO RESTART AND CORRECTLY SET THE FLOW OF CONTROL FROM INSIDE THE RANGE OF THE PERFORM.

THE ALTER VERB IS ANOTHER CONSTRUCT IN COBOL WHICH CAN CAUSE TROUBLES IN THE CHECKPOINT RESTART PROCESS. THE ALTER VERB CHANGES THE FLOW OF CONTROL. IT CHANGES THE TARGET ADDRESS IN A PARAGRAPH CONSISTING OF A SINGLE GO TO SENTENCE. ON RESTART, THE ALTERED GO TO STATEMENT WILL HAVE ITS ORIGINAL TARGET ADDRESS NOT THE NEW ADDRESS FURNISHED BY THE ALTER. THE WISE USER WHO WISHES TO CHECKPOINT AND RESTART WILL AVOID USE OF THE ALTER CONSTRUCT. IF AN ALTER MUST BE USED, THE USER MUST ASSURE HIMSELF THAT AT THE TIME AND POINT IN THE PROGRAM WHERE THE CHECKPOINT IS TAKEN, RESTORING THE ORIGINAL VALUE OF AN ALTERED TARGET ADDRESS WILL NOT AFFECT THE OPERATION OF THE PROGRAM.

THE FLEXIBILITY AND CAPABILITY OF I/O IN COBOL IS GREATER THAN THAT IN FORTRAN. COBOL I/O FALLS INTO THREE CATAGORIES,

1. SEQUENTIAL FILES,
2. INDEXED SEQUENTIAL AND INDEXED RELATIVE FILES,
3. INDEXED RANDOM AND INDEXED DYNAMIC FILES.

RECORDS OF AN UNINDEXED SEQUENTIAL FILE ARE READ OR WRITTEN SEQUENTIALLY INTO THE FILE. RECORDS IN THE SECOND CATAGORY, INDEXED SEQUENTIAL AND INDEXED RELATIVE, ARE READ OR WRITTEN SEQUENTIALLY AFTER FIRST POSITIONING ACCORDING TO A KEY OR INDEX. RECORDS IN THE THIRD CATAGORY REQUIRE THAT A KEY VALUE BE SPECIFIED FOR EACH READ OR WRITE.

COBOL USES THE MIDAS STORAGE AND RETRIEVAL SYSTEM TO HANDLE FILES IN THE SECOND AND THIRD CATAGORIES, I.E., INDEXED FILES. THE USER IS REFERRED TO THE DISCUSSION OF MIDAS, BELOW FOR INFORMATION ON CHECKING AND RESTORING SUCH FILES ON SYSTEM INTERRUPTION.

THE CHECKPOINT AND RESTART REQUIREMENTS OF INDEXED FILES IN COBOL ARE FEW. RANDOM FILES MUST ALWAYS SUPPLY A KEY OR INDEX ON READ OR WRITE. THE USER MUST HAVE SOME WAY OF CALCULATING THIS KEY AND CAN SUPPLY IT ON RESTART. IF THE KEY DEPENDS ON PREVIOUS DATA, THEN THAT DATA MUST BE SAVED IN THE CHECKPOINT FILE. INDEXED SEQUENTIAL AND INDEXED RELATIVE FILES MUST BE READ SEQUENTIALLY AND DYNAMIC FILES CAN BE READ SEQUENTIALLY. THE INDEX VALUE OF THE LAST RECORD SUCCESSFULLY READ OR WRITTEN MUST BE SAVED IN THE CHECKPOINT FILE IN ORDER TO REPOSITION THESE FILES ON RESTART. THE USER POSITIONS THESE FILE USING THE SAVED INDEX VALUE WITH THE START VERB.

THE PROBLEM OF REPOSITIONING SEQUENTIAL FILES ON RESTART IS DIFFICULT.

THIS DIFFICULTY ARISES FROM THE FACT THAT SEQUENTIAL READING AND WRITING USES DATA NOT ACCESSIBLE TO THE USER. THE USER HAS NO WAY OF POSITIONING DISKS OR TAPES ON RESTART AND INFORMING HIS PROGRAM OF THAT FACT EXCEPT THROUGH USE OF THE READ OR WRITE VERBS. THUS, THE USER MUST GO THROUGH ALL READS AS MANY TIMES AS HAD BEEN DONE FROM THE OPENING OF HIS FILES TILL THE TIME OF THE SYSTEM INTERRUPTION. THIS IS TRUE BOTH FOR DISKS AND TAPES.

SEQUENTIAL FILES BEING WRITTEN MUST BE OPENED AS I-O FILES AND READ UP TO THE POINT WHERE THE LAST CHECKPOINT WAS TAKEN. IN THE CASE OF MULTI-REEL FILES, THIS MEANS OPENING THE FILE AND STARTING WITH THE FIRST REEL.

CHECKPOINTS SHOULD BE TAKEN BEFORE A READ OR AFTER A WRITE TO MINIMIZE THE NUMBER OF VARIABLES WHICH HAVE TO BE SAVED. ANY MOVE'S AFTER A READ OR BEFORE A WRITE USUALLY INCREASES THE AMOUNT OF INFORMATION WHICH NEEDS TO BE SAVED.

SUBROUTINES POSE ANOTHER PROBLEM IN COBOL. IF A SUBROUTINE DOES ANY I/O, KEEPING THE FILES OPEN BETWEEN CALLS, OR IF IT SAVES ANY DATA WHICH IS NECESSARY FOR THE NEXT CALL, THEN THE SUBROUTINE MUST BE CHECKPOINTED. EACH SUCH SUBROUTINE MUST BE CALLED IN THE MAIN CHECKPOINT CODE OR BY EACH OTHER. IN ANY CASE, EACH SUCH SUBROUTINE MUST BE INFORMED THAT A CHECKPOINT IS TAKING PLACE. THE SUBROUTINE CAN WRITE ITS OWN CHECKPOINT RECORD OR PASS THE INFORMATION TO BE SAVED BACK TO THE CALLING ROUTINE. ON RESTART, THESE SUBROUTINES MUST BE INFORMED THAT RESTART IS TAKING PLACE AND MUST BE PASSED THE NECESARY RESTART INFORMATION IF IT WAS NOT SAVED INDEPENDENTLY. IF THE SUBROUTINE HAD AN I/O DEVICE OPEN, THEN IT MUST REOPEN THE I/O DEVICE AND POSITION IT CORRECTLY AS DESCRIBED ABOVE.

ONE WAY OF INFORMING SUBROUTINES ABOUT CHECKPOINT AND RESTART IS TO PASS A KEY ARGUMENT. THE KEY WOULD TAKE VALUES CORRESPONDING TO A NORMAL CALL, A CHECKPOINT CALL OR A RESTART CALL.

ONE FINAL ASPECT OF COBOL MUST BE TAKEN INTO ACCOUNT ON A SYSTEM INTERRUPTION AND RESTART. COBOL DOES NOT OUTPUT DIRECTLY TO A PRINTER. INSTEAD, IT OUTPUTS TO A PRINT FILE WHICH IS LATER SPOOLED OR QUEUED FOR PRINTING.

EACH TIME A COBOL PROGRAM STARTS UP, IT WILL CREATE SUCH A FILE IN THE CURRENT UFD. THE NAME OF THE FILE WILL BE SIX CHARACTERS IN LENGTH. THE FIRST FOUR CHARACTERS ARE THE FIRST FOUR CHARACTERS OF THE PROGRAM-ID AND THE LAST TWO CHARACTERS ARE NUMBERS. COBOL FIRST LOOKS FOR FILES WITH SUCH NAMES AND CREATES A NEW ONE USING THE NEXT HIGHEST AVAILABLE NUMBER. A COBOL RUN INVOKED WITH THE ID FUDGE WILL CREATE A FILE FUDGXX. THE NEW FILE WILL HAVE THE EXACT NAME FUDG03 IF FILES FUDG01 FUDG02 AND EVEN FUDG05 ALREADY EXIST IN THAT UFD. THE DATE AND TIME AS GIVEN BY THE FUTIL COMMAND LIST (SEE REFERENCE GUIDE, FILE MANAGEMENT SYSTEM, PDR 3110) IS ALSO USEFUL IN DETERMINING THE LATEST PRINT FILE TO BE SPOOLED. THIS DATE IS INITIALLY SET WHEN THE FILE IS CREATED AND UPDATED WHEN THE FILE IS CLOSED. ON SYSTEM INTERRUPTION, THE PRINT FILE HAS NOT BEEN CLOSED SO THE DATE AND TIME ARE THE THOSE AT THE TIME THE FILE WAS CREATED.

THE PRINT FILE WILL NOT BE SPOOLED ON A SYSTEM CRASH. IT IS UP TO THE USER TO FIND THAT FILE AND SPOOL IT.

IN SUMMARY, COBOL HAS HIDDEN VARIABLES ASSOCIATED WITH THE PERFORM VERB THE ALTER VERB AND WITH THE READ AND WRITE VERBS. THE USER SHOULD BE CAREFUL OF THE USE OF PERFORM AND ALTER AND SHOULD AVOID CHECKPOINTING IN THE RANGE OF A PERFORM.

THE USER MUST REPEAT READ AND SUBSTITUTE READ FOR WRITE TO REPOSITION HIS DISK FILE OR TAPE FOR SEQUENTIAL FILES.

SUBROUTINES MAY HAVE TO BE CALLED TO CHECKPOINT THEMSELVES OR PASS THE NECESSARY DATA BACK TO THE MAIN CHECKPOINT SECTION OF CODE.

FINALLY, THE USER MUST FIND AND SUBMIT THE APPROPRIATE PRINT FILE TO THE SPOOLER.

DBMS

THE DBMS OR DATA BASE MANAGEMENT SYSTEM HAS THE ABILITY TO RECOVER FROM SYSTEM INTERRUPTION. IT CAN IDENTIFY THE USER AND KEEPS A RECORD OF ALL HIS TRANSACTIONS. DBMS CAN ROLL BACK A USER'S TRANSACTION AND LEAVE THE USER'S DATA BASE IN THE STATE IT WAS BEFORE THE TRANSACTION WAS INITIATED.

THE COMMAND CLUP WILL CAUSE THE DATA BASE TO BE ROLLED BACK. CLUP WILL UNDO AN INCOMPLETE, OPEN TRANSACTION. IT IDENTIFIES THE TRANSACTION TO BE ROLLED BACK VIA THE USER NUMBER ASSOCIATED WITH A PARTICULAR TERMINAL OR PHANTOM. FOR THIS REASON, CLUP MUST BE EXECUTED FROM THE SAME TERMINAL OR PHANTOM AS THE ORIGINAL TRANSACTION.

CLUP IS A COMMAND WHICH TAKES NO ARGUMENTS.

THE DBMS SYSTEM OPENS ITS OWN FILES IN ORDER TO MANIPULATE THE DATA BASE. THE USER NEED NOT WORRY ABOUT POSITIONING THESE FILES AS THEY ARE OPENED AT THE BEGINNING OF A TRANSACTION AND CLOSED AT THE END OF THE TRANSACTION.

THERE ARE TWO OTHER COMMANDS USEFUL TO THE USER. THESE ARE IN THE DATABASE ADMINISTRATOR'S COMMAND PROCESSOR. THE COMMANDS ARE

SAVE SCHEMA NAME OF SCHEMA

AND

RESTORE SCHEMA NAME OF SCHEMA.

THESE COMMANDS ARE USED TO SAVE THE COMPLETE DATABASE ON TAPE OR DISK. BOTH COMMANDS PROMPT

TAPE OR DISK?

AND PROMPT FURTHER FOR SAVE FILE NAME OR UNIT NUMBER. THE RESTORE SCHEMA COMMAND ALLOWS RENAMING OF THE FILE TO BE RESTORED IF THERE IS A NAME CONFLICT WITH AN EXISTING FILE. THESE COMMANDS ARE DOCUMENTED IN THE PRIME COMPUTER USER'S GUIDE FOR THE DATABASE ADMINISTRATOR, IDR 3043, AND PRIME TECHINCAL UPDATE, PTU 46. A FUTURE VERSION OF RESTORE SCHEMA WILL ALLOW ROLL FOWARD USING SAVED AFTER IMAGES AND THE LOG FILE. THE RESTORED SCHEMA WILL THEN BE BROUGHT UP TO DATE WITH LITTLE USER INTERVENTION.

IT IS ADVISABLE TO KEEP ONE OR MORE COPIES OF AN ACTIVE (FREQUENTLY ALTERED) DATABASE ON MAGNETIC TAPE OR ANOTHER DISK PACK OR PARTITION FOR BACKUP USING THE SAVE SCHEMA COMMAND OF DBACP. IT IS ALSO ADVISABLE TO SAVE A DATABASE BEFORE USING ANY VOLATILE DBACP COMMANDS SUCH AS PACK AREA OR RUNNING A DML PROGRAM WHICH "CLEANS UP" THE DATABASE BY DELETING RECORD OCCURRENCES (PERIODIC PURGING).

MIDAS

MIDAS IS A DATA MANAGEMENT SYSTEM. IT DOES NOT HAVE THE ABILITY TO ROLL BACK OR UNDO CHANGES IT MAKES TO A DATA BASE. A SYSTEM INTERRUPTION WHEN ADDING A PRIMARY INDEX CAN RESULT IN THE LOSS OF DATA.

THE MIDAS SYSTEM OPENS AND CLOSES ITS OWN FILE UNITS TO MANIPULATE THE DATABASE. THE USER NEED NOT CONCERN HIMSELF WITH THOSE FILE UNITS FOR CHECKPOINTING.

THE MIDAS SYSTEM DOES PROVIDE A COMMANDS FOR AID IN RECOVERY. THE REPAIR COMMAND RECREATES THE DATA BASE. IT COPIES OVER ALL RECOVERABLE INFORMATION. THE REPAIR COMMAND CAN LOSE AT MOST TWO RECORDS FROM DAMAGE DUE TO SYSTEM INTERRUPTION UNLESS THE DISK HAS BEEN WRITTEN OVER. THE RECORDS LOST MAY HAVE NO RELATION TO THE RECORD BEING MANIPULATED BY THE USER PROGRMA AT THE TIME OF THE CRASH. THE DAMAGED RECORD MAY ONLY BE CHANGING ITS POSITION TO MAKE ROOM FOR A NEW RECORD.

THE REMAKE COMMAND REFOMATS AND REARRANGES DATA IN THE CURRENT DATA BASE. ON SYSTEM INTERRUPTION, THE USER SHOULD RUN REMAKE FOR INDICES ONLY. THIS TAKES MUCH LESS TIME THAN REPAIR AND WILL TELL THE USER IF ANY DAMAGE HAS BEEN DONE TO HIS DATA BASE. ONLY IF DAMAGE HAS BEEN DONE SHOULD REPAIR BE USED.

COMPLETE DESCRIPTIONS OF REPAIR AND REMAKE ARE GIVEN IN PRIME TECHNICAL UPDATE 39 ON MIDAS.

IT IS A GOOD IDEA TO CHECKPOINT PROGRAM USING MIDAS JUST BEFORE AN UPDATE OR JUST BEFORE DATA IS READ VIA MIDAS.

THE USER IS STRONGLY ADVISED TO KEEP A SEPARATE COPY OF VOLATILE DATABASES IN CASE DATA IS DELETED OR OTHERWISE LOST. IT IS ALSO ADVISABLE TO KEEP SUCH COPIES ON MAGNETIC TAPE OR A SEPARATE DISK PACK OR DISK VOLUME.

FORMS

THE FORMS SYSTEM IS USE TO FORMAT DATA EITHER FOR OUTPUT TO A TERMINAL OR A PRINTER. THERE IS NO CHECKPOINTING AS SUCH NECESSARY FOR THIS SYSTEM. IT IS MENTIONED ONLY TO REMIND THE USER THAT ITS PRINTER OUTPUT IS PLACED IN A FILE AND THAT FILE IS SPOOLED AND DELETED WHEN THE JOB IS FINISHED. THE FILE IS CREATED IN THE USERS HOME UFD WITH THE NAME PR##DD, WHERE DD IS A TWO DIGIT NUMBER. THE DIGITS ARE THE USER NUMBER PRINTED ON LOGIN OR WITH THE STATUS USERS COMMAND. AS FOR COBOL, THE DATE AND TIME LAST MODIFIED CAN BE USED TO HELP DETERMINE WHICH FILE THE USER SHOULD SPOOL ON SYSTEM INTERRUPTION.

WHERE TO CHECKPOINT

CHECKPOINTS ARE BEST TAKEN AT POINTS WHERE MINIMUM DATA MUST BE SAVED AND AT SOME REASONABLE FREQUENCY. IN MANY APPLICATIONS DATA IS INPUT

FROM A TAPE OR A DISK FILE, PROCESSED, A DATABASE IS REFERENCED OR UPDATED AND THE PROCESS IS REPEATED. A GOOD PLACE TO CHECKPOINT SUCH A SYSTEM IS AFTER THE DATABASE HAS BEEN UPDATED BUT BEFORE NEW DATA IS READ. THIS IS PARTICULARLY SUITABLE FOR DBMS BECAUSE OF THE ROLLBACK FEATURE. THE DBMS USER SHOULD CHECKPOINT AT THE COMPLETION OF A TRANSACTION.

HANDLING CHECKPOINT FILES

AN INVALID CHECKPOINT FILE CAN BE CREATED WHEN USING A SINGLE CHECKPOINT FILE AND OVERWRITING IT REPEATEDLY AT EACH CHECKPOINT. A SYSTEM INTERRUPTION OCCURING WHILE THE CHECKPOINT IS BEING WRITTEN OUT LEAVES NO VALID CHECKPOINT FILE.

THERE ARE A NUMBER OF WAYS TO SOLVE THIS PROBLEM. ONE WAY IS TO SIMPLY LET THE CHECKPOINT FILE GROW CONTINUALLY, ADDING EACH NEW SET OF CHECKPOINT DATA TO THE END OF THE FILE. A SPECIAL SYMBOL SUCH AS "END OF CHECKPOINT" CAN BE USED TO INDICATE THE END OF THE CHECKPOINT DATA SET. ON RESTART THE DATA FILE IS CHECKED FOR THE LAST COMPLETE SET OF CHECKPOINT DATA AND THE PROGRAM WILL CONTINUE FROM THE CORRESPONDING CHECKPOINT.

AN ALTERNATIVE METHOD TO AVOID HAVING NO VALID CHECKPOINT FILE IS TO USE TWO CHECKPOINT FILES ALTERNATELY. THE LATEST COMPLETE CHECKPOINT FILE IS DETECTED BY EXAMINING THE DATE TIME STAMP ASSOCIATED WITH EACH FILE. THE DATE TIME MODIFIED STAMP IS SET WHEN THE FILE IS CREATED AND UPDATED EACH TIME THE FILE IS CLOSED. IF THE FILES ARE OPENED AND THEN CLOSED EACH TIME A CHECKPOINT IS TAKEN, THEN THE DATE TIME MODIFIED STAMP WILL INDICATE THE LATEST COMPLETE CHEKPOINT FILE.

THE DATE TIME STAMP EXISTS IN THE UFD ENTRY ASSOCIATED WITH EACH FILE. THE UFD ENTRY IS ACCESSED VIA A CALL TO RDEN$$ (SEE REFERENCE GUIDE, FILE MANAGAEMENT SYSTEM, PDR3110). THE CALL TO RDEN$$ REQUIRES THAT THE UFD CONTAINING THE ENTRY BE ALREADY OPEN AND THE FILE UNIT NUMBER FOR THAT OPEN UFD IS PASSED TO RDEN$$ AS ONE OF ITS ARGUMENTS. THE SRCH$$ CALL (SEE PDR 3110) IS USED TO OPEN THE UFD FOR READING IN THE CURRENT UFD OR TSRC$$ (SEE PDR3110) IS USED TO OPEN THE A UFD ANYWHERE IN THE PRIMOS FILE STRUCTURE. THE COBOL PROGRAM IN APPENDIX 2 OPENS THE CURRENT UFD FOR READING AND CALLS RDEN$$ TO FIND THE FILE COB.CHKPT. FDATE AND FTIME ARE THE DATE AND TIME LAST MODIFIED IN BINARY.

SUMMARY

THE CHECKPOINT AND RESTART PROCESS MUST ACCOMPLISH THREE THINGS.

1. IT MUST SAVE THE "STATE OF THE PROGRAM" AT A GIVEN POINT.

2. IT MUST RECOGNIZE A RESTART VIA AN ARGUMENT OR SOME OTHER MEANS.

3. IT MUST RESTORE THE "STATE OF THE PROGRAM" ON RESTART.

TO SAVE THE "STATE OF THE PROGRM," A CHECKPOINT FILE MUST BE WRITTEN OUT. AT LEAST TWO SUCH FILES SHOULD BE USED TO AVOID THE PROBLEM OF AN INVALID CHECKPOINT FILE CAUSED BY A CRASH WHILE WRITING IT OUT.

THE "STATE OF THE PROGRAM" IS DEFINED BY:

1. ALL USER VARIABLE NEEDED FOR FURTHER CALCULATION OR CONTROL IN THE PROGRAM SUCH AS RUNNING TOTALS OR COMPUTED GO TO VARIABLES.

2. ALL SYSTEM VARIABLES WHICH AFFECT FURTHER OPERATION OF THE PROGRAM SUCH AS THE HIDDEN INDICES AND RETURN POINTERS IN PERFORM OPERATIONS IN COBOL.

3. ALL OPEN I/O DEVICE POSITIONS.

4. POSSIBLE SYSTEM VARIABLES SUCH AS THE CURRENT ERASE AND KILL CHARACTERS.

A KIND OF CHECKPOINTING IN CX IS DEMONSTRATED. THE VERSION ILLUSTRATED DOES NOT SAVE THE COMPLETE STATE OF THE "PROGRAM" BUT MENTION IS MADE OF HOW TO EXTEND IT TO DO SO.

FORTRAN IS DISCUSSED. THERE ARE NO HIDDEN OR USER INACCESSABLE CONTROL VARIABLES USED BY FORTRAN. SAVING DISK FILE UNIT POSITIONS IS ILLUSTRATED FOR CX AND SAVING MAGNETIC TAPE RECORD INDICES IS DISCUSSED. IT US UP TO THE USER TO RECORD, SAVE AND REPOSITION ANY ATTACHED I/O DEVICES. ANY VARIABLES WHICH ARE IN SUBROUTINES CAN BE CHECKPOINTED BY A SINGLE BLOCK OF CHECKPOINT CODE IF THOSE VARIABLES ARE IN A COMMON AREA ALSO ACCESSIBLE TO THE CHECKPOINT CODE.

COBOL IS DISCUSSED. THE PROBLEMS OF USER INACCESSIBLE VARIABLES USED BY PERFORM AND ALTER ARE DISCUSSED. CHECKPOINTS SHOULD BE TAKEN AT A POINT WHERE THE INDEX OR RETURN VALUE OF A PERFORM OR THE TARGET ADDRESS OF AN ALTER'ED PARAGRAPH WILL NOT AFFECT THE FURTHER OPERATION OF THE USER PROGRAM. THE HIDDEN VARIABLES ASSOCIATED WITH READ AND WRITE ARE DISCUSSED. READ'S MUST BE COUNTED, AND AFTER OPENING AT THE BEGINNING OF THE FILE OR TAPE EXECUTED IN A LOOP WITHOUT OTHER PROCESSING IN ORDER TO REPOSITION THE FILE OR TAPE. WRITE FILES MUST BE OPENED AS I-O AND POSITIONED AS FOR READ FILES. COBOL DOES NOT HAVE THE CONCEPT OF COMMON STORAGE AREA SHARED BY ALL SUBROUTINES. SUBROUTINES WHICH HAVE DATA WHICH MUST BE SAVED MUST BE CALLED BY THE CHECKPOINT AND RESTART CODE AND MUST BE INFORMED THAT THIS IS A CHECKPOINT OR RESTART CALL. ALTERNATIVELY, THEY MAY ALWAYS CALL WITH SUCH INFORMATION AND RETURN IN THEIR ARGUMENTS THE INFORMATION NECESSARY FOR CHECKPOINTING.

SUBROUTINES WHICH OPEN THEIR OWN FILES AND DO I/O CAN PASS INDICES OR POSITIONS BUT MUST BE CALLED DURING RESTART TO OPEN AND REPOSITION THEIR FILES. COBOL PRINT FILES MUST BE SPOOLED BY THE USER. THE NAME OF THE CURRENT PRINT FILE IS NOT KNOWN BY THE USER IN COBOL BUT CAN BE FOUND BY HIM BY INVESTIGATING THE CURRENT DIRECTORY IN WHICH HIS

APPENDIX_1

```
(0001)  C CHKPT -- A ROUTINE TO CRUDELY CHECKPOINT A COMINPUT OR CX RUN
(0002)  C WRITTEN BY M.A.MEER 01/10/78
(0003)  C
(0004)  C SYSCOM>KEYS.F       MNEMONIC KEYS FOR FILE SYSTEM (FTN)     31
 MAY, 1977
(0004)        NOLIST
(0005)  C SYSCOM>ERRD.F       MNEMONIC CODES FOR FILE SYSTEM (FTN)     6
 SEPT, 1977
(0005)        NOLIST
(0006)  C
(0007)        INTEGER ARG(16),NAME(16),INFO(8),ALEN,CODE,BLANKS,NLEN,NW
DS
(0008)        INTEGER CONT(2),TYPE,WD,TXT(16,10),TXTL,CNAM(3,4),NTXT,NC
NAM
(0009)        INTEGER P(2)
(0010)        INTEGER*4 POS,DZERO
(0011)        EQUIVALENCE (POS,P(1))
(0012)        LOGICAL FIRST
(0013)        DATA BLANKS/'  '/,DZERO/000000/,ALEN/16/
(0014)        DATA CNAM/'RDTK$$','CHKPT ','TSRC$$','PRWF$$'/
(0015)        DATA TXT/'NO NAME INPUT                    ',
(0016)       X'FINDING NAME.CHKPT                ',
(0017)       X'OPENING FOR READ                  ',
(0018)       X'OPENING FOR WRITE                 ',
(0019)       X'CLOSING                           ',
(0020)       X'READ NAME.CHKPT                   ',
(0021)       X'POSITIONING UNIT 6                ',
(0022)       X'GETTING POSITION 6                ',
(0023)       X'WRITING NAME.CHKPT                ',
(0024)       X'NAME TOO LONG                     '/
(0025)  C
(0026)        FIRST = .FALSE.                    /* INITIALIZE FIRST FLAG
 */
(0027)        NAME(1)=BLANKS                     /* INITIALIZE NAME */
(0028)  C
(0029)  C GET ARGS
(0030)  C
(0031)  10    CALL RDTK$$(1,INFO,ARG,ALEN,CODE)
(0032)        IF (CODE.NE.0) GO TO 1010          /* ERROR RDTK$$ */
(0033)  C
(0034)        IF (INFO(1).EQ.6) GO TO 30         /* NO MORE ARGS */
(0035)        IF (ARG(1).NE.'-F') GO TO 15       /* MUST BE NAME */
(0036)        FIRST = .TRUE.                     /* SET FIRST FLAG */
(0037)        GO TO 10                           /* GET NEXT ARG */
(0038)  C
(0039)  15    NLEN=INFO(2)                       /* MUST BE NAME, GET ITS
 LENGTH */
(0040)        NWDS = RS(NLEN+1,1)                /* GET NUMBER OF WORDS I
N THE NAME */
(0041)        DO 20 I=1,NWDS                     /* COPY THE NAME */
(0042)             NAME(I)=ARG(I)
```

```
(0043)  20    CONTINUE
(0044)        GO TO 10                              /* GET NEXT ARG */
(0045)  C
(0046)  C PROCESS ARGS
(0047)  C
(0048)  30    IF(NAME(1).EQ.BLANKS) GO TO 1020 /* NO NAME ERROR */
(0049)        IF (NLEN+6.GT.32) GO TO 1015      /* NAME TOO LONG */
(0050)  C
(0051)  C CHECK FOR FILE 'NAME.CHKPT'
(0052)  C
(0053)        IF(RT(NLEN,1).EQ.0) GO TO 32      /* DOES NAME END IN MIDD
LE OF WORD */
(0054)        NAME(NWDS)=LT(NAME(NWDS),8)+:256 /* ADD '.' TO RIGHT HALF
 WORD */
(0055)        NAME(NWDS+1)='CH'
(0056)        NAME(NWDS+2)='KP'
(0057)        NAME(NWDS+3)='T '
(0058)        GO TO 33
(0059)  32    NAME(NWDS+1)='.C'                 /* INPUT NAME ENDS ON WO
RD BOUNDRY */
(0060)        NAME(NWDS+2)='HK'
(0061)        NAME(NWDS+3)='PT'
(0062)  C
(0063)  33    NWDS = NWDS+3
(0064)        NLEN = NLEN+6                     /* UPDATE CHARCTER COUNT
 */
(0065)  C
(0066)  C SEE IF FILE EXISTS
(0067)  C
(0068)        CONT(1) = 0                       /* INDEX OF FIRST CHARAC
TER IN NAME */
(0069)        CONT(2) = NLEN                    /* NUMBER OF CHARACTERS
IN NAME */
(0070)        CALL TSRC$$(K$EXST,NAME,15,CONT,TYPE,CODE)
(0071)        CALL TONL
(0072)  C
(0073)        IF (CODE.EQ.0) GO TO 35           /* IT EXISTS */
(0074)        IF (CODE.NE.E$FNTF) GO TO 1030    /* ERROR SEARCHING */
(0075)  C
(0076)  C IF FIRST CALL, CREATE IT.
(0077)        CODE = 0
(0078)        CONT(1) = 0
(0079)  C
(0080)        IF (FIRST) CALL TSRC$$(K$WRIT,NAME,15,CONT,TYPE,CODE)
(0081)        IF (CODE.NE.0) GO TO 1032         /* ERROR SEARCHING */
(0082)        GO TO 45                          /* READ POSITION OF 6, T
HE INPUT UNIT */
(0083)  C
(0084)  35    IF (.NOT.FIRST) GO TO 40
(0085)  C
(0086)  C FIRST, SO READ POSITION FROM NAME.CHKPT AND POSITION FILE UNI
T 6
(0087)  C
(0088)        CONT(1) = 0
```

```
(0089)          CALL TSRC$$(K$READ,NAME,15,CONT,TYPE,CODE)
(0090)          IF (CODE.NE.0) GO TO 1031
(0091)          CALL PRWF$$(K$READ,15,LOC(POS),2,DZERO,WD,CODE)
(0092)          IF (CODE.NE.0) GO TO 1040          /* PRWF$$ ERROR */
(0093)          CALL PRWF$$(K$POSN+K$PREA,6,LOC(POS),0,POS,WD,CODE)    /*
POSITION 6 */
(0094)          IF (CODE.NE.0) GO TO 1041          /* ERROR POSITIONING 6 *
/
(0095)          GO TO 100                          /* RETURN */
(0096)    C
(0097)    C FOUND IT AND NOT FIRST, SO, READ CURRENT POSITION OF 6 AND SA
VE IT
(0098)    C
(0099)    40    CONT(1) = 0
(0100)          CALL TSRC$$(K$WRIT,NAME,15,CONT,TYPE,CODE)  /* OPEN NAME.
CHKPT */
(0101)          IF (CODE.NE.0) GO TO 1032          /* ERROR SEARCHING */
(0102)    45    CALL PRWF$$(K$RPOS,6,LOC(POS),0,POS,WD,CODE)
(0103)          IF (CODE.NE.0) GO TO 1042          /* ERROR READING POSITIO
N */
(0104)          CALL PRWF$$(K$WRIT,15,LOC(POS),2,DZERO,WD,CODE)
(0105)          IF (CODE.NE.0) GO TO 1043          /* ERROR WRITING */
(0106)    C
(0107)    C DONE
(0108)    C
(0109)    100   CALL SRCH$$(K$CLOS,0,0,15,0,CODE)    /* CLOSE NAME.CHKPT *
/
(0110)          IF (CODE.NE.0) GO TO 1033           /* ERROR CLOSING */
(0111)          CALL EXIT
(0112)    C
(0113)    C ERROR HANDLING
(0114)    C
(0115)    1010  NCNAM = 1                          /* RDTK$$ */
(0116)          NTXT = 1
(0117)          TXTL = 0
(0118)          CALL ERRPR$(K$IRTN,CODE,ARG,ALEN,CNAM(1,NCNAM),6)
(0119)          GO TO 2000
(0120)    C
(0121)    1015  NTXT = 10                         /* NAME TOO LONG */
(0122)          TXTL = 13
(0123)          CALL ERRPR$(K$IRTN,CODE,NAME,NLEN,CNAM(1,2),6)
(0124)          GO TO 1025
(0125)    C
(0126)    1020  NTXT = 1                          /* NO NAME INPUT */
(0127)          TXTL = 13
(0128)    1025  NCNAM = 2                         /* CHKPT */
(0129)          GO TO 2000
(0130)    C
(0131)    1030  NTXT = 2                          /* FINDING NAME.CHKPT */
(0132)          TXTL = 18
(0133)          GO TO 1039
(0134)    C
(0135)    1031  NTXT = 3                          /* OPENING FOR READ */
(0136)          TXTL = 16
```

```
(0137)          GO TO 1039
(0138)    C
(0139)    1032  NTXT = 4                                /* 'OPENING FOR WRITE' *
/
(0140)          TXTL = 17
(0141)          GO TO 1039
(0142)    C
(0143)    1033  NTXT = 5                                /* 'CLOSING' */
(0144)          TXTL = 7
(0145)          GO TO 1039
(0146)    C
(0147)    1039  NCNAM = 3                               /* TSRC$$ */
(0148)          CALL ERRPR$(K$IRTN,CODE,NAME,NLEN,CNAM(1,NCNAM),6)
(0149)          GO TO 2000
(0150)    C
(0151)    1040  NTXT = 6                                /* READING NAME.CHKPT */
(0152)          TXTL = 15
(0153)          GO TO 1049
(0154)    C
(0155)    1041  NTXT = 7                                /* POSITIONING UNIT 6 */
(0156)          TXTL = 18
(0157)          GO TO 1049
(0158)    C
(0159)    1042  NTXT = 8                                /* GETTING POSITION OF 6
 */
(0160)          TXTL = 18
(0161)          GO TO 1049
(0162)    C
(0163)    1043  NTXT = 9                                /* WRITING NAME.CHKPT */
(0164)          TXTL = 18
(0165)    C
(0166)    1049  NCNAM = 4                               /* PRWF$$ */
(0167)          CALL ERRPR$(K$IRTN,CODE,NAME,NLEN,CNAM(1,NCNAM),6)
(0168)          GO TO 2000
(0169)    C
(0170)    C
(0171)    2000  CALL ERRPR$(K$NRTN,CODE,TXT(1,NTXT),TXTL,CNAM(1,NCNAM),6)
(0172)    C
(0173)          END
```

```
  ALEN     I          000005    0007S   0013I   0031A   0118A
  ARG      I          000262    0007S   0031A   0035    0042    0118A
  BLANKS   I          000002    0007S   0013I   0027    0048
  CNAM     I          000006    0008S   0014I   0118A   0123A   0148A   0167A
0171A
  CODE     I          001053    0007S   0031A   0032    0070A   0073    0074
0077M
                                0080A   0081    0089A   0090    0091A   0092
0093A
                                0094    0100A   0101    0102A   0103    0104A
0105
                                0109A   0110    0118A   0123A   0148A   0167A
0171A
  CONT     I          000302    0008S   0068M   0069M   0070A   0078M   0080A
```

```
0088M
                                        0089A   0099M   0100A
  DZERO   J              000003         0010S   0013I   0091A   0104A
  E$FNTF  I  PARAMETER                  0005S   0074
  ERRPR$  R  EXTERNAL    000000         0118    0123    0148    0167    0171
  EXIT    R  EXTERNAL    000000         0111
  FIRST   L              001054         0012S   0026M   0036M   0080    0084
  I       I              001055         0041M   0042
  INFO    I              000304         0007S   0031A   0034    0039
  K$CLOS  I  PARAMETER                  0004S   0109
  K$EXST  I  PARAMETER                  0004S   0070
  K$IRTN  I  PARAMETER                  0004S   0118    0123    0148    0167
  K$NRTN  I  PARAMETER                  0004S   0171
  K$POSN  I  PARAMETER                  0004S   0093
  K$PREA  I  PARAMETER                  0004S   0093
  K$READ  I  PARAMETER                  0004S   0089    0091
  K$RPOS  I  PARAMETER                  0004S   0102
  K$WRIT  I  PARAMETER                  0004S   0080    0100    0104
  LOC     I  EXTERNAL    000000         0091    0093    0102    0104
  LT      I  EXTERNAL    000000         0054
  NAME    I              000314         0007S   0027M   0042M   0048    0054M   0055M
0056M
                                        0057M   0059M   0060M   0061M   0070A   0080A
0089A
                                        0100A   0123A   0148A   0167A
  NCNAM   I              001060         0008S   0115M   0118    0128M   0147M   0148
0166M
                                        0167    0171
  NLEN    I              001061         0007S   0039M   0040    0049    0053    0064M
0069
                                        0123A   0148A   0167A
  NTXT    I              001062         0008S   0116M   0121M   0126M   0131M   0135M
0139M
                                        0143M   0151M   0155M   0159M   0163M   0171
  NWDS    I              001063         0007S   0040M   0041    0054    0055    0056
0057
                                        0059    0060    0061    0063M
  POS     J              000334         0010S   0013S   0091A   0093A   0102A   0104A
  PRWF$$  R  EXTERNAL    000000         0091    0093    0102    0104
  RDTK$$  R  EXTERNAL    000000         0031
  RS      R  EXTERNAL    000000         0040
  RT      R  EXTERNAL    000000         0053
  SRCH$$  R  EXTERNAL    000000         0109
  TONL    R  EXTERNAL    000000         0071
  TSRC$$  R  EXTERNAL    000000         0070    0080    0089    0100
  TXT     I              000022         0008S   0015I   0171A
  TXTL    I              001064         0008S   0117M   0122M   0127M   0132M   0136M
0140M
                                        0144M   0152M   0156M   0160M   0164M   0171A
  TYPE    I              001065         0008S   0070A   0080A   0089A   0100A
  WD      I              001066         0008S   0091A   0093A   0102A   0104A

  _10                    000342         0031D   0037    0044
  _100                   000637         0095    0109D
```

_1010	000653	0032	0115D			
_1015	000676	0049	0121D			
_1020	000713	0048	0126D			
_1025	000717	0124	0128D			
_1030	000722	0074	0131D			
_1031	000727	0090	0135D			
_1032	000734	0081	0101	0139D		
_1033	000741	0110	0143D			
_1039	000746	0133	0137	0141	0145	0147D
_1040	000765	0092	0151D			
_1041	000772	0094	0155D			
_1042	000777	0103	0159D			
_1043	001004	0105	0163D			
_1049	001010	0153	0157	0161	0166D	
_15	000367	0035	0039D			
_20	000401	0041	0043D			
_2000	001027	0119	0129	0149	0168	0171D
_30	000407	0034	0048D			
_32	000437	0053	0059D			
_33	000446	0058	0063D			
_35	000521	0073	0084D			
_40	000572	0084	0099D			
_45	000607	0082	0102D			

0000 ERRORS [<.MAIN.>FTN-REV14.2]

APPENDIX 2

```
__REV 15.0  COBOL     SOURCE FILE:  CALTST                                  0
3/08/78     11:31
 (0001)           ID DIVISION.
 (0002)           PROGRAM-ID.  CALTST.
 (0003)           ENVIRONMENT DIVISION.
 (0004)           DATA DIVISION.
 (0005)           WORKING-STORAGE SECTION.
 (0006)           77  KEY-VAL      USAGE COMP VALUE 1.
 (0007)           77  KEY-VAL1 USAGE COMP VALUE 5.
 (0008)           77  NAME     PIC A(32) VALUE 'COB.CHKPT'.
 (0009)           77  VNAME    USAGE COMP VALUE -1.
 (0010)           77  NAMLEN   USAGE COMP VALUE 9.
 (0011)           77  VNAMLN   USAGE COMP VALUE 2.
 (0012)           77  FUNIT    USAGE COMP VALUE 15.
 (0013)           77  TYPE     USAGE COMP.
 (0014)           77  CODE     USAGE COMP.
 (0015)           01  RDEN-BUF.
 (0016)               02 ECW   USAGE COMP.
 (0017)               02 ENAME  PIC A(32).
 (0018)               02 PROTEC USAGE COMP.
 (0019)               02 JUNK  PIC XX.
 (0020)               02 FILTYP USAGE COMP.
 (0021)               02 FDATE USAGE COMP.
 (0022)               02 FTIME USAGE COMP.
 (0023)               02 JUNK1 PIC XXXX.
 (0024)
 (0025)           77  BUFLEN USAGE COMP VALUE 24.
 (0026)           77  WORDS-READ  USAGE COMP.
 (0027)           PROCEDURE DIVISION.
 (0028)           R1.
 (0029)               CALL 'SRCH$$' USING KEY-VAL VNAME VNAMLN FUNIT
                                  TYPE CODE.
 (0030)               DISPLAY 'CODE ' CODE.
 (0031)               DISPLAY 'TYPE ' TYPE.
 (0032)               CALL 'RDEN$$' USING KEY-VAL1 FUNIT RDEN-BUF BUFLEN
 (0033)                                WORDS-READ NAME NAMLEN CODE.
 (0034)               DISPLAY 'CODE ' CODE.
 (0035)               DISPLAY 'WORDS-READ ' WORDS-READ.
 (0036)               DISPLAY 'DATE ' FDATE.
 (0037)               STOP RUN.
A P R O G R A M    S T A T I S T I C S


EXECUTABLE CODE SIZE: 194 WORDS.
CONSTANT POOL SIZE: 19 WORDS.
TOTAL PURE PROCEDURE SIZE: 213 WORDS.

WORKING-STORAGE SIZE: 100 BYTES.
TOTAL LINKFRAME SIZE: 76 WORDS.

STACK SIZE: 31 WORDS.
```

```
TRACE MODE: OFF.

NO ARGUMENTS EXPECTED.

37 SOURCE LINES.

0000 ERRORS  0000 WARNINGS,  P400/500 COBOL REV 15.00 <CALTST>
```

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.

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] OUTPUTFILE [-CONTROL_ARGS]

ORIGFILE IS THE TREE NAME OF THE "ORIGINAL" FILE. FILEB THROUGH FILEE ARE TREE NAMES OF FILES WHICH TRACE THEIR ANCESTRY TO ORIGFILE.

OUTPUTFILE IS THE TREE NAME OF THE MERGED OUTPUT FILE.

CONTROL ARGS:

-MINL # SETS THE MINIMUM NUMBER OF LINES WHICH MUST MATCH FOLLOWING A DISCREPANCY IN ORDER TO RESYNCH ALL FILES. THE DEFAULT VALUE IS -MINL 3.

-FORCE CAUSES FILEB TO BE A "PREFERRED" FILE WHEN CONFLICTS EXIST BETWEEN SEVERAL FILES. WHEN -FORCE IS USED, THE USER OF MRGF IS NEVER QUERIED TO RESOLVE A CONFLICT (SEE BELOW).

-BRIEF 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.

EXAMPLE:

CONSIDER THE FOLLOWING THREE FILES:

FILEA	FILEB	FILEC
THE	THE	THE
QUICK	QUICK	QUICK
BROWN	RED	BROWN
FOX	FOX	FOX
JUMPS	JUMPS	JUMPS
OVER	OVER	OVER
THE	THE	THE
LAZY	SLEEPING	SNORING
DOG	DOG	DOG

A MRGF OF THESE FILES WOULD PRODUCE THE FOLLOWING:

```
      A8       LAZY
      CHANGED TO
      B8       SLEEPING
      BUT ALSO CHANGED TO
      C8       SNORING
      EDIT
   $ B
   $ GO

      MERGE FINISHED.
      1 MANUAL CHANGE.
      1 AUTOMATIC CHANGE AS FOLLOWS:
           1 FROM FILE B.
```

IN THE ABOVE EXAMPLE, THE LINES PRECEDED BY A "$" WERE TYPED BY THE USER. THE MERGED OUTPUT FILE FROM THE ABOVE MRGF WOULD APPEAR AS FOLLOWS:

```
THE
QUICK
RED
FOX
JUMPS
OVER
THE
SLEEPING
DOG
```

NOTE THAT IF THE -FORCE CONTROL ARGUMENT HAD BEEN USED, THE SAME MERGED OUTPUT WOULD HAVE BEEN PRODUCED. HOWEVER, THE CHANGE FROM FILEB WOULD HAVE BEEN INSERTED AUTOMATICALLY, AND THE USER WOULD NOT HAVE BEEN QUERIED.

ACKNOWLEDGEMENT:

THE MRGF COMMAND IS BASED ON THE MERGE_ASCII COMMAND OF MULTICS, WHICH WAS IMPLEMENTED BY ROBERT E. MULLEN OF HONEYWELL INFORMATION SYSTEMS, INC. THE ALGORITHMS USED IN THE MRGF COMMAND WERE "BORROWED" EXTENSIVELY FROM THOSE DEVELOPED BY MULLEN.

SUBJECT: COBOL, REV 15.0

1. INTRODUCTION

THIS DOCUMENT DESCRIBES THE CHANGES BETWEEN REV 14 AND REV 15 COBOL. THE COMPILER HAS MANY USER VISIBLE ENHANCEMENTS.

2. LARGER ADDRESS SPACE

REV 14 AND BELOW WERE RESTRICTED TO A MAXIMUM OF A 64K BYTE ADDRESS SPACE. THIS WAS FURTHER CUT DOWN BY 4K BYTES FOR EACH FILE DECLARED AND FOR EACH ARGUMENT PASSED. THE SIZE OF A DATA ITEM (GROUP OR ELEMENTARY) COULD NOT EXCEED 4K BYTES.
THESE RESTRICTIONS HAVE BEEN RELAXED OR REMOVED FOR REV 15. THE NEW CHARACTERISTICS ARE:

- THE TOTAL ADDRESS SPACE WHICH A PROGRAM USES NO LONGER HAS AN EXPLICIT LIMIT.
- THE MAXIMUM DATA ITEM SIZE IS NOW 32K BYTES.
- THE OCCURS COUNT MAY NOT EXCEED 32767.
- THERE MAY NOW BE UP TO 126 FILES DECLARED. OBVIOUSLY, THERE ARE INSUFFICIENT FILE SYSTEM UNITS AVAILABLE TO SUPPORT THIS MANY FILES OPEN SIMULTANEOUSLY.

NATURALLY, IN R MODE, THE TOTAL PROGRAM + DATA SIZE MUST NOT EXCEED 64K WORDS. IN 64V MODE, THIS EXTENDED ADDRESSING IS DONE THROUGH COMPILER GENERATED COMMON BLOCKS.

3. STREAMLINED COMPILER

THE REV 15 COBOL COMPILER IS ROUGHLY TWICE AS FAST AS OLDER COMPILERS. WORKING SET SIZE HAS ALSO BEEN SIGNIFICANTLY REDUCED, SO COMPILATION SPEED ON SMALL MEMORY SYSTEMS SHOULD IMPROVE SIGNIFICANTLY.

4. EXTENSIONS

REV 15 COBOL HAS THE FOLLOWING NEW FEATURES:

- OPEN EXTEND FOR SEQUENTIAL DISK FILES
- FULL IF STATEMENTS (EXCEPT ARITHMETIC EXPRESSION OPERANDS)
- V MODE MAG TAPE SUPPORT

5. BUG FIXES

- COPY STATEMENTS MAY NOW CONTAIN TEXT AFTER THE COPY CLAUSE. IN THE LISTING FILE THE LINE NUMBERING OF THE COPY FILE IS NOW INDEPENDANT OF THE LINE NUMBERS OF THE SOURCE. FOR EXAMPLE:

```
(0069)          COPY FILE.  VALUE 213.
[0001]                INSERTED
[0002]                  TEXT
[0003]                   ..
[NNNN]                   ..
(0069)          COPY FILE.  VALUE 213.
(0070)
(0071)          ETC, ETC, ETC.
```

- LEVEL 88 (DECIMAL) IS NOW FUNCTIONING PROPERLY FOR ALL CASES.
- COMP-3 USAGE NO LONGER CAUSES 'INCOMPLETE TREE' PROBLEMS.
- SYNTAX ONLY COMPILATION (-B NO) IS NOW WORKING CORRECTLY.
- 'DECIMAL POINT IS COMMA' IS FUNCTIONAL IN 64V.

6. IMPORTANT NOTE

REV 14 AND EARLIER COBOL LIBRARIES ARE INCOMPATIBLE WITH THE REV 15 COMPILER.

DATE: FEBRUARY 21, L978

SUBJECT: SECOND CARD READER/PUNCH/PRINTER AND
LINE PRINTER CONTROLLER (URC) AND
APPLICABLE SOFTWARE CHANGES FOR REV. 15

CARD READER/PUNCH/PRINTER OPERATION

PRIME 400 AND 500 PROCESSOR HARDWARE AND SOFTWARE NOW SUPPORT TWO URC CONTROLLERS. THE SECOND CONTROLLER USES DMA CHANNEL '37, AND HAS A DEVICE ADDRESS OF '05.

ANY COMBINATION OF TWO OF THE TWO TYPES OF MPC CONTROLLERS (DESCRIBED BELOW) IS ALLOWED.

1. URC1 - DRIVES UP TO TWO LINE PRINTERS AND ONE 80 COLUMN CARD-READER

2. URC2 - DRIVES ONE LINE PRINTER AND ONE CARD PROCESSOR (CARD READER/PUNCH/PRINTER)

THE FOLLOWING SOFTWARE CHANGES HAVE BEEN MADE TO SUPPORT THE ADDITION CONTROLLER AND PERIPHERALS.

I/O LIBRARY DRIVERS UNIT PAPAMETER ENCODING IS NOW:

ROUTINE	UNIT	DEVICE	CONTROLLER	FUNCTION
I$AC03	0	CR0	FIRST	
	1	*CR1	SECOND	READ A CARD
O$AC03	0	CR0	FIRST	
	1	*CR1	SECOND	PUNCH A CARD
I$AC15	0	CR0	FIRST	
	1	*CR1	SECOND	READ AND PRINT A CARD
O$AC15	0	CR0	FIRST	
	1	*CR1	SECOND	PUNCH AND PRINT A CARD
O$AL06	0	PR0	FIRST	
	1	PR1	FIRST	PRINT ONE LINE ON
	2	*PR2	SECOND	A LINE PRINTER
	3	*PR3	SECOND	

*NEW DEVICE FOR REV. 15

CARD_READER/PUNCH_UTILITIES

THE FOLLOWING DESCRIBES CHANGES TO COMMANDS CRMPC, CPMPC AND PURGING OF CPPMPC.

CRMPC, CPMPC THE COMMAND FORMAT IS CHANGED TO ALLOW OPTIONAL SPECIFICATION OF: 1) CARD READER/PROCESSOR UNIT NUMBER (DEFAULTS TO 0) AND, 2) CARDS WILL BE INTERPRETED (PRINTED AS READ/PUNCHED FOR CARD READERS/PROCESSORS WITH THIS FEATURE). A FILENAME MUST BE GIVEN. THE FORMAT IS SHOWN BELOW IN THE TWO FOLLOWING EXAMPLES.

```
CRMPC TREENAME -PRINT -CR0

        OR

CPMPC -CR1 TREENAME -PRINT
```

NOTE THAT COMMAND ARGUMENTS MAY BE SPECIFIED IN ANY ORDER AND ONLY TREENAME MUST BE SPECIFIED.

CPPMPC PURGED AS FUNCTIONALTY NOW CONTAINED IN CPMPC. CPPMPC WAS EQUIVALENT TO:

```
CPMPC -PRINT TREENAME
```

THE FOLLOWING DESCRIBES LC-50 OPERATIONS AND CARD PUNCH-CARD READ COMMANDS.

CARD READER/PUNCH AS AN ASSIGNABLE DEVICE

THE CARD READER/PUNCH IS CONSIDERED TO BE ONE DEVICE. A USER MAY ASSIGN AND OPERATE BOTH THE CARD READER AND PUNCH BY:

OK, AS CR0 OR OK, AS CR1

THE CARD PROCESSOR HAS THE CAPABILITY TO READ AS WELL AS PUNCH SO IT MUST BE CONSIDERED AS A SINGLE ASSIGNABLE DEVICE.

ASSIGN COMMAND

THE ASSIGN COMMAND FOR THE CARD READER IS MULTI-PURPOSED. IT FIRST FUNCTIONS TO ASSIGN THE CARD PROCESSOR TO A USER. SUBSEQUENT PROCESSOR OPERATIONS SHOULD BE PRECEDED BY THE USER ALWAYS RE-ASSIGNING THE PROCESSOR. THIS SERVES TO (1) CLEAR READER/PUNCH BUFFERS AND (2) TURNS OFF READ-AHEAD. READ-AHEAD ACTION IS INITIATED UPON READ REQUESTS AND IT FUNCTIONS TO KEEP THE CARD READER BUFFER FULL REGARDLESS OF THE NUMBER OF READ REQUESTS. THIS SERVES TO OPTIMIZE SPEED PERFORMANCE OF THE CARD PROCESSOR. FOR THESE REASONS AN 'AS CR#' SHOULD PRECEDE ANY CARD PROCESSOR OPERATIONS.

PUNCHING

CPMPC - CPMPC IS A UTILITY COMMAND IN CMDNCO THAT WILL PUNCH A FILE ON A CARD DECK. COMMAND CPMPC IS GIVEN AS FOLLOWS:

OK, CPMPC IREENAME -OPTION1 -OPTION2

CPMPC DOES NOT PUNCH AN END-OF-FILE CARD AT THE END OF A DECK OF CARDS. A CARD PUNCHED '$E' IS USED TO SPECIFY AN END OF FILE FOR READ OPERATIONS. A '$E' IS USEFUL TO SEPARATE CARD DECKS GROUPED TOGETHER ON A READ OPERATION.

BEFORE EACH RUNNING OF CPMPC THE OPERATOR SHOULD:

1. ASSIGN THE CARD READER/PUNCH. THIS IS DONE BY TYPING:

OK, AS CR0 OR OK, AS CR1

THE CARD READER/PUNCH SHOULD BE ASSIGNED BEFORE ANY PUNCH OR READ CARD COMMANDS ARE GIVEN.

2. IF AN ERROR CODE IS DISPLAYED IN THE DIGITAL READOUT OF THE LC-50 IT SHOULD BE CLEARED BY PRESSING THE CANCEL AND THEN THE CLEAR TRACK BUTTONS ON THE CARD PROCESSOR'S CONTROL PANEL. NOW THE LC-50 CAN BE STARTED AND THE OPERATOR MAY (RE)ASSIGN IT AND RUN CPMPC.

IF A PUNCH JOB IS TO BE ABORTED, THE OPERATOR SHOULD FIRST QUIT RUNNING PUNCH PROGRAM VIA TERMINAL AND THEN FOLLOW ABOVE STEPS AS IF TO CLEAR AN ERROR. IF THE LC-50 GOES INTO A NULL STATE (NEITHER START NOR STOP FUNCTIONS) IT MAY BE NECESSARY TO MOMENTARILY POWER DOWN THE LC-50. THIS STATE MAY RESULT FROM ABORTING A PUNCH AND PRINT OPERATION.

AN OPERATOR MAY STOP AND ADD MORE CARDS TO THE CARD READER/PUNCH'S INPUT HOPPER BY PRESSING STOP, OR ALLOWING INPUT HOPPER TO RUN OUT OF CARDS. MORE CARDS SHOULD BE ADDED, AND BY PRESSING START, THE PUNCH OPERATION WILL CONTINUE. IF CARDS ARE ALLOWED TO RUN OUT AN ERROR CODE OF 1 WILL BE INDICATED BY THE DIGITAL READOUT. DO NOT CANCEL, BUT ONLY ADD MORE CARDS AND PRESS START.

READING

CRMPC - CRMPC IS A COMMAND IN CMDNCO THAT WILL READ A DECK OF CARDS INTO A DISK FILE. COMMAND CRMPC IS GIVEN AS FOLLOWS:

OK, CRMPC TREENAME -CR# -PRINI

CARDS WILL BE READ UNTIL EITHER A '$E' IS READ OR THE CARD PROCESSOR IS STOPPED BY ALLOWING CARDS TO RUN OUT IN THE INPUT HOPPER OR BY PRESSING THE STOP BUTTON. THE '$E' CARD NOTIFIES CRMPC TO CLOSE THE FILE BEING READ. CRMPC COMMAND MAY AGAIN BE ISSUED TO READ ANOTHER CARD DECK. UPON AN INPUT HOPPER EMPTY ERROR CONDITION CAUSED BY RUNNING OUT OF CARDS OR UPON AN OPERATOR PRESSING STOP, CRMPC CAUSE A RETURN TO COMMAND MODE. THE FILE CONTAINING CARDS JUST READ CAN BE CLOSED BY TYPING:

OK, C ALL OR
OK, C TREENAME

IF IT IS DESIRABLE TO CONTINUE READING INTO SAME FILE, DO NOT CLOSE THE FILE BUT INSTEAD LOAD IN MORE CARDS TO BE READ, PRESS START ON LC-50 AND AT THE TERMINAL RUNNING CRMPC TYPE 'S (RETURN)'. EXAMPLE:

```
OK, CRMPC_FILEA
GO

OK, S          COMMAND MODE RETURNED ON HOPPER EMPTY
GO             OR OPERATOR STOP

OK, S
GO
```

THIS SEQUENCE CAN BE CONTINUED UNTIL A '$E' IS READ OR AN OPERATOR CLOSES THE FILE.

IF THE CARD READER/PUNCH INPUT HOPPER GOES EMPTY DURING A READ OPERATION THE DIGITAL READOUT WILL NOT DISPLAY A 1 (AS IN THE CASE FOR HOPPER EMPTY ON PUNCH OPERATIONS) AND THE STOP BUTTON WILL LIGHT BUT WON'T BLINK.

BEFORE EACH RUNNING OF CRMPC THE OPERATOR SHOULD:

1. ASSIGN THE CARD PROCESSOR, DONE BY A

   OK, AS CR#

2. IF AN ERROR CODE IS DISPLAYED BY THE DIGITAL READOUT, IT SHOULD BE RESET BY PRESSING CANCEL AND CLEAR TRACK. THEN PRESS START.

IF A READ JOB IS TO BE ABORTED, THE OPERATOR SHOULD FIRST QUIT VIA TERMINAL AND THE CARD READER WILL STOP. THE FILE BEING CREATED SHOULD THEN BE CLOSED AND THEN DELETED IF DESIRED.

VERIFY OPERATION

THE LC-50 CONTAINS A POST-PUNCH READ STATION AND CHECK ALL DATA PUNCHED. TO FURTHER VERIFY A PUNCH OPERATION, IT IS SUGGESTED THAT A PUNCH DECK BE READ IN AND COMPARED AGAINST THE SOURCE FILE USING THE COMMAND CMPF.

ERROR RECOVERY

THE LC-50 MULTI-FUNCTION CARD PROCESSOR DESCRIBES ERROR RECOVERY PROCEDURES. ERROR RECOVERY PROCEDURES NOT DETAILED IN THIS DOCUMENT ARE DESCRIBED IN THE LC-50 DOCUMENT. ERROR CODES ARE LISTED THERE IN.

THE FOLLOWING SOFTWARE SUPPORTS THE LC-50 CARD READER/PUNCH. NOTE THAT T$CMPC AND I$AC03 ARE DOCUMENTED IN PRIME'S LIBRARY GUIDE.

SOFTWARE ROUTINES I$PMPC, O$AC03, I$AC15, O$AC15

I$PMPC - RAW DATA MOVER FOR THE CARD PUNCH. THE CALLING SEQUENCE IS:

```
CALL T$PMPC (UNIT, LOC(BUFFER), WORD COUNT, INST., STATV)
```

T$PMPC IS CALLED BY DRIVER O$AC03,O$AC15 AND I$AC15, THE CARD PUNCH DRIVER. THESE ROUTINES ARE LIBRARY ROUTINES AND MAY BE CALLED BY THE USER.

ARGUMENTS ARE:

UNIT - CARD PUNCH UNIT.

LOC(BUFFER) - A POINTER TO A BUFFER THAT HOLDS DATA TO BE PUNCHED. DATA IS PACKED TWO CHARACTERS PER WORD.

WORD COUNT - NUMBER OF WORDS TO PUNCH ON A CARD FROM BUFFER.

INST. - INSTRUCTION REQUIRED TO BE SENT TO CARD PUNCH. INSTRUCTIONS ARE:

BIT SET	INSTRUCTION	MEANING
1	:100000	READ STATUS
3	:20000	PROCESS IN BINARY MODE
4	:10000	FEED A CARD
5	:4000	READ A CARD
6	:2000	PUNCH A CARD
7	:1000	PRINT A CARD
8	:400	STACK A CARD

IF BIT 7 IS SET, BITS 15,16 ARE DECODED AS FOLLOWS:

15,16

00	PRINT THE PUNCHED DATA
01	PRINT THE READ DATA
10	PRINT BOTH PUNCHED AND READ DATA
11	PRINT DATA FROM MAIN MEMORY. DATA IS ASCII ONLY.

EXAMPLE: TO PUNCH A CARD, INSTRUCTION WOULD BE AN OCTAL :12400
MEANING:

(A) FEED A CARD
(B) PUNCH A CARD AND
(C) STACK A CARD

STATV - THREE WORD STATUS VECTOR WHERE:
STATV(1) - NOT USED
STATV(2) - DEVICE STATUS
STATV(3) - NUMBER OF WORDS READ

POSSIBLE DEVICE STATUS OF PUNCH IS AS FOLLOWS:

BIT SET	OCTAL VALUE	CONDITION
1-8		DISPLAY ERROR NUMBER FROM PUNCH
9	:200	ON-LINE
11	:40	ILLEGAL CODE
13	:10	HARDWARE ERROR
14	:4	OPERATOR INTERVENTION REQUIRED

NOTE: STATUS IS ONLY RETURNED ON READ REQUESTS.

T$PMPC CAN PERFORM ONLY ONE OF THE THREE FOLLOWING FUNCTIONS PER CALL PUNCH, READ OR PRINT (INTERPRET), IN ADDITION TO OTHER FUNCTIONS SUCH AS STACK OR FEED A CARD.

O$AC03 - CARD PUNCH DRIVER.

THE CALLING SEQUENCE IS:

CALL O$AC03 (UNIT, BUFFER, WORD COUNT, ALT. RET.)

ARGUMENTS ARE:

UNIT - CARD PUNCH UNIT

BUFFER - BUFFER CONTAINING LINE TO BE PUNCHED

WORD COUNT - NUMBER OF WORDS TO BE PUNCHED

ALT. RET. - ALTERNATE RETURN (CURRENTLY IGNORED)

O$AC15, I$AC15 - PUNCH OR READ AND INTERPRET CARD DRIVERS.

O$AC15 AND I$AC15 HAVE THE SAME CALLING SEQUENCES AS O$AC03 AND I$AC03 AND FUNCTION SIMILIARLY. O$AC15 WILL PUNCH AND INTERPRET ONE LINE, I$AC15 READS AND INTERPRETS ONE LINE.

PROGRAM EXECUTES.

DBMS IS DISCUSSED. FOR AN INCOMPLETE TRANSACTION, THE COMMAND CLUP WITH NO ARGUMENTS WILL RESTORE THE DATA BASE TO ITS STATE BEFORE THE CRASH OCCURRED. THE CLUP COMMAND MUST BE ISSUED FROM THE SAME TERMINAL OR PHANTOM (HAVING THE SAME USER NUMBER) ON WHICH THE PROGRAM WAS RUNNING AT THE TIME OF THE CRASH. IT IS SUGGESTED THAT THE USER CHECKPOINT JUST AFTER A TRANSACTION IS COMPLETED. MENTION IS ALSO MADE OF THE SAVE SCHEMA AND RESTORE SCHEMA COMMANDS WHICH SAVE AND RESTORE THE COMPLETE DATABASE.

MIDAS IS DISCUSSED BRIEFLY. THE MIDAS DATA BASE CAN BE DAMAGED BY A SYSTEM INTERRUPTION WHILE ADDING PRIMARY INDICES. IN THIS CASE, COMPLETE RECOVERY IS NOT GUARANTEED. THE REPAIR COMMAND IS EXPLAINED IN PRIME TECHNICAL UPDATE NO. 39 SECTION 2.4. IT CAN REPAIR MOST DAMAGE AND INDICATE WHERE THE DAMAGE WAS DONE. IT CAN OFTEN RETURN ENOUGH INFORMATION SO THAT THE USRE CAN RECOVER DATA THAT REPAIR CANNOT. IT IS SUGGESTED THAT THE USER FIRST RUN REMAKE ON INDICES ONLY TO CHECK FOR DAMAGE AS REPAIR TAKES MUCH TIME ON LARGE DATABASES.

FORMS IS MENTIOND TO REMIND THE USER TO SPOOL ANY PRINT FILES WHERE PRODUCED BY THE INTERRUPTED JOB.

SUGGESTIONS ARE GIVEN TO THE USER AS TO WHEN TO CHECKPOINT AND HOW TO HANDLE CHECKPOINT FILES.

ALL STATEMENTS IN THIS NOTE APPLY TO REV. 15. IT IS HOPED THAT COBOL AND MIDAS WILL BE IMPROVED TO MAKE RECOVERY EASIER FOR THE USER. IT IS ALSO HOPED THAT CLUP WILL IN THE FUTURE TAKE AN ARGUMENT TO REMOVE THE "SAME TERMINAL" RESTRICTION.

DATE: MARCH 9, 1978

SUBJECT: CX MODIFICATIONS AT REV 15

1 SCOPE

THIS DOCUMENT DESCRIBES THE MODIFICATIONS TO THE CX SUBSYSTEM AVAILABLE AT REV 15.

2 NETWORK SUPPORT

A COMMAND LINE OPTION, "-ON", NOW EXISTS WHICH ALLOWS THE USER TO SPECIFY THE LOGICAL DEVICE NUMBER CONTAINING THE CX QUEUE TO BE USED:

```
CX <FILENAME> -ON LDEV

CX -Q -ON LDEV

CX -SXX -ON LDEV
```

WHERE "LDEV" IS THE (LOCAL) LOGICAL DISK NUMBER OF A REMOTE DISK CONTAINING THE CX QUEUE.

3 INTERNAL CHANGES

THE CX MASTER PHANTOM NOW CALLS THE (NEW) PHANT$ ROUTINE TO SPAWN THE SLAVE PHANTOM. THIS GIVES THE MASTER THE OPPORTUNITY TO RECOVER FROM THE NO FREE PHANTOMS ERROR WHICH PREVIOUSLY CAUSED IT TO ABORT.

THIS FEATURE PROHIBITS THE USAGE OF THE REV 15 CX ON PRE-REV 15 SYSTEMS.

SUBJECT: REV 15 CHANGES TO SORT AND EDB

A. SORT

1. RUNS 2-3 TIMES AS FAST.

2. WILL ACCEPT LOWER AND UPPER CASE CHARACTERS; SORTING THE LOWER CASE AS IF THEY WERE UPPER CASE, BUT OUTPUTTING THEM AS LOWER CASE.

3. CHECKS THE VALIDITY OF THE OUTPUT FILE BEFORE DOING THE SORT.

4. THE MAXIMUM RECORD SIZE MUST NOT EXCEED 512 CHARACTERS (256 WORDS). TO INCREASE THE BUFFER SIZE, THE PARAMETER EB$EOF IN THE MODULE SRTCOM MUST BE REDEFINED - THEN THE SORT SOURCE MUST BE RE-ASSEMBLED VIA C<-SUBS. LIKEWISE, IF THE TOTAL KEY LENGTH EXCEEDS 316 CHARACTERS (158 WORDS), THE PARAMETER EB$LNG IN SRTCOM MUST BE REDEFINED.

5. C<-VSUB CAN BE USED TO BUILD A V-MODE VERSION OF SORT WHICH WILL NOT RUN AS FAST AS THE 64-R MODE VERSION. HOWEVER, THE BUFFER SIZES CAN BE INCREASED WITHOUT TAKING ANY MEMORY SPACE USED FOR THE SHELL SORT.

6. THE BOBS$$ SUBROUTINE NO LONGER EXISTS.

B. EDB

1. PRINTS DIRECT ENTRY NAMES.

2. REPLAC COMMAND ALLOWS AN OBJECT MODULE TO BE REPLACED BY ONE OR MORE MODULES; E.G.,

```
OK, EDB LIBEXP
GO
ENTER, F ALL
ENT1A ENT1B ENT1C ENT2D ENT2E ENT3G ENT4H
.BOTTOM.
ENTER, Q

OK, EDB B<-NFILE3
GO
ENTER, F ALL
ENT3F ENT3G
.BOTTOM.
ENTER, Q

OK, EDB LIBEXP LIBNEW
GO
ENTER, R ENT3G B<-NFILE3
ENT1A ENT1B ENT1C ENT2D ENT2E ENT3G
ENTER, C ALL
ENT4H
.BOTTOM.
ENTER, Q

OK, EDB LIBNEW
GO
ENTER, F ALL
ENT1A ENT1B ENT1C ENT2D ENT2E ENT3F ENT3G ENT4H
.BOTTOM.
ENTER, Q
```

SUBJECT: REV 15 FORMS ENHANCEMENTS

1 SCOPE

THIS DOCUMENT DESCRIBES THE IMPROVEMENTS MADE TO PRIME'S FORMS MANAGEMENT SYSTEM AT REV 15.

2 ABSTRACT

THE VERSION OF FORMS RELEASED AT REV 15 HAS UNDERGONE CONSIDERABLE CHANGES. AMONG THESE ARE:

- INTERNAL DATA-BASE AND PROCEDURES REDESIGN, PRODUCING:
  AVERAGE RUN-TIME SPEED INCREASE BY A FACTOR OF 2-3
  RUN-TIME PACKAGE MEMORY REQUIREMENTS DECREASED BY ABOUT 1/2
- FUNCTION KEY SUPPORT
- EXPLICIT CURSOR POSITION COMMAND
- ZERO-FILL ON JUSTIFY
- "FORCEREAD" COMMAND TO FORCE OPERATOR DATA ENTRY
- REV 13-14 BUGS REPAIRED
- ETC...

3 MEMORY REQUIREMENTS

THE FOLLOWING TABLE COMPARES THE MEMORY REQUIREMENTS OF THE (64R MODE) REV 14 TO REV 15 FORMS LIBRARIES. ALL VALUES ARE IN WORDS (DECIMAL).

	REV 14	REV 15
CODE (RUN-TIME PROPER):	8906	5033
CODE (DEVICE DRIVERS):	2735	2758
COMMON DATA:	7697	2972
TOTAL:	19338	10583

4 RUN-TIME PACKAGE TIMING COMPARISONS

A STUDY WAS MADE TO DETERMINE THE SPEED INCREASES AFFORDED BY VARIOUS PORTIONS OF THE NEW RUN-TIME PACKAGE AS COMPARED TO PRIOR RELEASES.

4.1 TIMING STUDY OVERVIEW

FOR THE PURPOSES OF THE STUDY, 2 FORM DEFINITIONS WERE CONSTRUCTED. THE FIRST WAS A 'SMALL' FORM DEFINITION, CONSISTING OF:

10 CHARACTERS IN 2 PROTECTED FIELDS (OUTPUT-ONLY)
20 UNPROTECTED CHARACTERS ON THE TERMINAL

21 CHARACTERS IN THE INPUT RECORD (SINGLE SUBSTREAM)
NO VALIDATION CRITERIA

ON THE OPPOSITE END OF THE SPECTRUM, A 'LARGE' FORM DEFINITION WAS CONSTRUCTED WHICH CONSISTED OF:

358 CHARACTERS IN 30 PROTECTED FIELDS (OUTPUT-ONLY)
199 UNPROTECTED CHARACTERS ON THE TERMINAL
199 CHARACTERS TO BE INPUT IN 6 SUBSTREAMS
VALIDATION CRITERIA ON APPROXIMATELY 2/3 OF THE INPUT DATA

THE PROGRAM USED TO PERFORM THE TEST SIMPLY STARTED A CLOCK, CALLED RDASC AND WRASC TO PREFORM THE DESIRED FUNCTIONS, AND STOPPED THE CLOCK.

ALL TIMES PUBLISHED ARE IN SECONDS AND INCLUDE BOTH CPU AND DISK I/O TIME. THE TIMING TESTS WERE MADE WITH THE 64R MODE FORMS LIBRARIES ON A LIGHTLY LOADED PRIME 400 WITH 512KW.

4.2 INVOKE TEST

THE INVOKE TEST MEASURED THE AVERAGE INVOKE/RELEASE TIME AT REV 14 AND REV 15 FOR BOTH SMALL AND LARGE FORM DEFINITIONS. THE CALLS TO THE DEVICE DRIVER TO DO DEVICE INITIALIZATION AND CLOSE-OUT WERE DUMMIED OUT, I.E. THIS TEST ONLY MEASURES THE INVOKE AND RELEASE PROCESSORS.

FORM	REV 14	REV 15	INCREASE FACTOR
SMALL	.289	.103	2.81
LARGE	.536	.212	2.53

4.3 OUTPUT TEST

THIS TEST MEASURED THE TIME TAKEN BY THE FORMS RUN-TIME PACKAGE PROPER TO PROCESS OUTPUT DATA. LIKE THE INVOKE TEST, THE DEVICE DRIVER WAS DISABLED.

FORM	REV 14	REV 15	INCREASE FACTOR
SMALL	.025	.0097	2.57
LARGE	.121	.027	4.47

4.4 INPUT TEST

THE INPUT TEST MEASURED THE TIME TAKEN BY THE FORMS RUN-TIME PACKAGE TO PROCESS INPUT DATA. AS DESCRIBED ABOVE, DEVICE DRIVER TIMES ARE NOT INCLUDED.

FORM	REV 14	REV 15	INCREASE FACTOR
SMALL	.029	.010	2.90
LARGE	.106	.030	3.51

4.5 TRANSACTION TEST

THE TRANSACTION TEST MEASURED THE TIME TAKEN FOR A "TYPICAL" FORMS TRANSACTION. THE TRANSACTION CONSISTED OF:

- INVOKING THE FORM DEFINITION,
- DISPLAYING THE FORM,
- READING OPERATOR-ENTERED DATA FROM THE FORM,
- AND RELEASING THE FORM DEFINITION

THE TEST WAS MADE WITH BOTH TERMINALS SUPPORTED BY PRIME, THE INFOTON VISTAR/3 AND PERKIN-ELMER 1200 (OWL).

TERMINAL	FORM	REV 14	REV 15	INCREASE FACTOR
VISTAR3	SMALL	0.404	0.157	2.57
	LARGE	1.650	0.526	3.14
OWL	SMALL	0.374	0.190	1.97
	LARGE	1.150	0.470	2.45

THE READER WILL NOTICE THAT THE SPEED INCREASES IN THE TRANSACTION TEST ARE IMPROPORTIONAL TO THE INCREASES IN OTHER TESTS, A CHARACTERISTIC WHICH IS ATTRIBUTED TO THE DEVICE DRIVERS.

PART OF THE REV 15 RUN-TIME SPEED-UP COMES FROM MODIFICATIONS MADE TO THE DEVICE DRIVERS. THE OUTPUT PROCESSORS NOW BUFFER ALL OUTPUT CHARACTERS (INSTEAD OF OUTPUTTING THEM INDIVIDUALLY) AND USE AS FEW CALLS AS POSSIBLE (TO TNOUA) TO DO OUTPUT, THUS ELIMINATING MUCH OF THE OPERATING SYSTEM OVERHEAD. INPUT PROCESSORS NOW CALL C1IN (WHICH SVC'S OR DOES A DIRECT ENTRANCE CALL) INSTEAD OF T1IN (WHICH PTRAP'S).

5 FUNCTION KEY SUPPORT

A METHOD NOW EXISTS WHEREBY THE APPLICATION PROGRAM CAN TRAP AND PROCESS A FUNCTION KEY INPUT BY THE OPERATOR.

5.1 USER IMPLEMENTATION

TWO NEW COMMANDS HAVE BEEN ADDED TO THE RUN-TIME PACKAGE,

```
##FKEYS ON     ENABLES FUNCTION KEYS, AND, CONVERSELY,
##FKEYS OFF    DISABLES FUNCTION KEYS.
```

TO PROVIDE COMPATIBILITY WITH PREVIOUS RELEASES, FUNCTION KEYS ARE INITIALLY DISABLED WHEN A FORM IS INVOKED.

WHEN ENABLED AND THE OPERATOR DEPRESSES A FUNCTION KEY, FORMS TRANSFERS CONTROL TO THE END-OF-FILE RETURN SPECIFIED IN THE READ STATEMENT (END= IN FORTRAN, 'AT END...' OR DECLARATIVE PARAGRAPH IN COBOL). FOR COBOL PROGRAMS, THE FUNCTION KEY NUMBER IS STORED IN THE FILE STATUS ITEM SPECIFIED IN THE SELECT STATEMENT; IT IS THE

PROGRAMMER'S RESPONSIBILITY TO INSURE THAT THE FILE STATUS CLAUSE IS SPECIFIED. FOR FORTRAN PROGRAMS, THE USER MUST DECLARE A COMMON BLOCK CALLED "FKYCM$" CONTAINING 1 INTEGER (16 BIT) ITEM. THIS IS SET TO THE NUMBER OF THE FUNCTION KEY DEPRESSED.

WHEN FUNCTION KEYS ARE DISABLED, THEY PERFORM AS THEY DID AT PREVIOUS RELEASES, I.E. HAVE NO EFFECT.

AN OBVIOUS REMINDER: THE USER SHOULD NOT WRITE AN APPLICATION WHICH IMPLEMENTS FUNCTION KEYS UNLESS ALL TERMINALS WHICH ARE TO RUN THIS APPLICATION ARE EQUIPPED WITH THEM.

5.2 DEVICE DRIVER CONSIDERATIONS

NO CHANGES ARE REQUIRED FOR USER-WRITTEN DEVICE DRIVERS WHEN UPGRADING TO A REV 15 FORMS SYSTEM IF THE USER WISHES TO MAINTAIN REV 14 FUNCTIONALITY. HOWEVER, IF THE USER DESIRES TO SUPPORT FUNCTION KEYS AND/OR EXPLICIT CURSOR POSITIONING (DESCRIBED ELSEWHERE IN THIS DOCUMENT) SOME MINIMAL CHANGES ARE IN ORDER.

THE FILE 'FORMS>RUN>DEVCM$' MUST BE INSERTED INTO THE DEVICE DRIVER PRIOR TO ANY DATA OR EXECUTABLE STATEMENTS. THIS FILE CONTAINS DECLARATIONS FOR THE FUNCTION KEY ENABLE/DISABLE FLAG (LOGICAL VARIABLE FKEYS, SET TO TRUE IF FUCTION KEYS ENABLED, FALSE IF DISABLED) AND THE NUMBER OF THE FUNCTION KEY DEPRESSED (INTEGER VARIABLE FKEYNO).

WHEN THE SUBROUTINE IS CALLED WITH AN INPUT-FORM REQUEST (FUNCTION CODE 3), THE DEVICE DRIVER SHOULD CHECK FOR A FUNCTION KEY INPUT AND HONOR IT IF AND ONLY IF FKEYS IS TRUE; ELSE PERFORM THE SAME (ERROR) HANDLING AS DONE AT REV 14. IF FKEYS IS TRUE, FKEYNO SHOULD BE SET TO THE NUMBER OF THE FUNCTION KEY DEPRESSED AND RETURN SHOULD BE MADE TO THE CALLER. NOTE THAT THE SUBROUTINE WHICH CALLS THE DEVICE DRIVER, NOT THE DEVICE DRIVER ITSELF, PROVIDES ALL OF THE USER-VISIBLE RETURN HANDLING (THRU EOF RETURN) DESCRIBED ABOVE.

6 EXPLICIT CURSOR POSITIONING

REV 15 FORMS ALLOWS THE APPLICATIONS PROGRAM TO SPECIFY THE FIELD AT WHICH THE CURSOR WILL POSITIONED WHEN THE NEXT OPERATOR INPUT IS REQUIRED FROM THE TERMINAL.

6.1 USER IMPLEMENIATION

A NEW COMMAND HAS BEEN ADDED TO THE RUN-TIME PACKAGE TO SPECIFY AN EXPLICIT CURSOR POSITION:

```
##POSITION FIELDNAME
```

ON NEXT TERMINAL INPUT REQUEST FOLLOWING THIS COMMAND THE CURSOR

WILL BE POSITIONED TO THE FIRST CHARACTER OF THE SPECIFIED FIELD. NOTE THAT THIS COMMAND IS ONLY APPLICABLE TO THE NEXT READ; FOLLOWING READS WILL HAVE THE CURSOR POSITIONED AT THE FIRST UNPROTECTED POSITION ON THE TERMINAL UNLESS ANOTHER POSITION COMMAND IS ISSUED.

6.2 DEVICE DRIVER CONSIDERATIONS

THE INSERT FILE 'FORMS>RUN>DEVCM$' MENTIONED ABOVE ALSO CONTAINS DECLARATIONS FOR CURSOR POSITION RELATED INFORMATION.

THE INTEGER VARIABLE XPOS SHOULD BE CHECKED IN THE INPUT-FORM (FUNCTION 3) PROCESSOR PRIOR TO ALLOWING THE OPERATOR TO ENTER DATA. IF NON-ZERO, THE CURSOR SHOULD BE POSITIONED TO THE LOCATION REPRESENTED BY VARIABLES XPOS (COLUMN) AND YPOS (LINE). IF XPOS IS ZERO, THE CURSOR SHOULD BE POSITIONED TO THE FIRST UNPROTECTED POSITION ON THE SCREEN (AS IN PRIOR RELEASES).

7 FORMS DEFINITION LANGUAGE MODIFICATIONS

TWO MODIFICATIONS HAVE BEEN MADE TO FDL AT REV 15:

- ZERO-FILL ON JUSTIFY
- STREAM/FORMAT NAME RESTRICTION

7.1 ZERO FILLED JUSTIFICATION SPECIFICATION

THE USER MAY NOW SPECIFY ON A STREAM FIELD THAT SUPPLIES ZEROES INSTEAD OF SPACES WHEN PERFORMING JUSTIFICATION AT RUN-TIME. TWO NEW STREAM FIELD ATTRIBUTES HAVE BEEN DEFINED:

ZERO-FILL	SPECIFIES THAT ZEROES ARE TO BE USED FOR PADDING INSTEAD OF SPACES. THIS SHOULD ONLY BE USED (AND ONLY MAKES SENSE WITH) RIGHT AND LEFT JUSTIFY OPERATIONS.
SPACE-FILL	SUPPLIES THE STANDARD SPACE CHARACTER ON JUSTIFICATION.

NOTE THAT THESE ATTRIBUTES MAY NOT BE SPECIFIED ON FORMAT DESCRIPTOR FIELDS.

7.2 STREAM/FORMAT DESCRIPTOR NAMING RESTRICTION

AT REV 15 THERE IS NO LONGER AN OPTION OF NAMING A FORMAT DESCRIPTOR DIFFERENT FROM A STREAM DESCRIPTOR, I.E. THE FORMAT PARAMETER TO THE STREAM STATEMENT HAS BEEN REMOVED. THE REASON STEMS FROM THE STREAM/FORMAT LINKING CONVENTIONS WHICH HAVE CHANGED DRASTICALLY SINCE REV 14 (LINKING IS NOW DONE WHEN A FORM IS ADDED TO THE FORMS

CATALOG AND NOT AT RUN-TIME, A PROCESS WHICH CONTRIBUTES HEAVILY TO THE REV 15 SPEEDUP).

8 TERMINAL INPUT OVERRIDE COMMAND

A COMMAND NOW EXISTS WHICH PERMITS THE PROGRAMMER TO FORCE OPERATOR INPUT FROM THE TERMINAL, THEREFORE OVERRIDING FORMS' TERMINAL INPUT PROTOCOL WHEN PROCESSING MULTIPLE SUBSTREAMS. IN THE PAST, PHYSICAL TERMINAL INPUT WAS ONLY DONE WHEN THE APPLICATIONS PROGRAM EXECUTED THE FIRST READ OR ATTEMPTED TO READ A SUBSTREAM WHICH HAD ALREADY BEEN READ. IF THE PROGRAMMER WANTED TO FORCE OPERATOR INPUT, HE HAD TO GO OUT OF HIS WAY TO RE-READ AN ALEADY-READ SUBSTREAM AND THEN READ THE SUBSTREAM HE ACTUALLY WANTED. THE NEW RUN-TIME COMMAND,

```
##FORCEREAD
```

INSTRUCTS FORMS TO WAIT FOR OPERATOR INPUT ON THE NEXT READ STATEMENT.

9 REMOVAL OF SHARED BUFFER POOL FACILITY

THE SHARED BUFFER POOL FACILITY IN THE 64V MODE SHARED VERSION OF FORMS HAS DISAPPEARED. IT WAS FOUND THAT THIS FACILITY WAS ADVERSELY AFFECTING ALL NON-SHARED USERS (ESPECIALLY THOSE ON LOW-MEMORY P300 SYSTEMS). THE APPROXIMATELY 7000 (DECIMAL) WORDS WHICH WERE OCCUPIED BY THE BUFFER POOL MANAGER CODE AND COMMON DID NOT WARRANT ANY SMALL SPEED INCREASE IT AFFORDED P400 USERS.

10 CONFIGURABLE I/O LIST

THE REVISED DATA BASE FORMATS IN THE FORMS RUN-TIME PACKAGE INCLUDES A NEW I/O LIST ALLOCATION SCHEME. THE DEFAULT I/O LIST SIZE IS 2500 WORDS (DECIMAL). IF THE USER RUNS A PROGRAM WHICH INVOKES A FORM THAT EXCEEDS THIS CAPACITY, FORMS PRINTS THE ERROR MESSAGE:

```
REQUIRED= NNNN, AVAILABLE= 2500.
I/O LIST OVERFLOW.
```

THE USER MAY ALLOCATE A LARGER I/O LIST IN HIS (FORTRAN) PROGRAM BY INSERTING THE FOLLOWING 3 STATEMENTS:

```
      PARAMETER IOLSIZ=DESIRED_SIZE
      COMMON /IOBCM$/ IBUF(3), IOL(IOLSIZ)

      DATA IBUF /IOLSIZ, 0, 0/
```

ALL ITEMS ARE 16 BIT INTEGERS.

THE USER MAY ENLARGE THE I/O LIST SIZE IN A COBOL PROGRAM BY WRITING

THE ABOVE CODE FOLLOWED BY AN 'END' STATEMENT, COMPILING IT WITH FORTRAN, AND LOADING IT AFTER HIS COBOL PROGRAM. THE BINARY MODULE CONTAINING THE REDEFINITION OF THE I/O LIST MUSI BE LOADED PRIOR TO LOADING THE FORMS LIBRARY.

THE USER MAY ALSO MODIFY THE DEFAULT BUFFER POOL SIZE BY CHANGING THE 'IOLSIZ' DECLARATION IN 'FORMS>RUN>IOLDEF' AND RE-COMPILING THE RUN-TIME SYSTEM.

11 RUN-TIME ERROR DIAGNOSTICS

THE RUN-TIME ERROR HANDLING SCHEME HAS BEEN REWRITTEN AT REV 15 TO PROVIDE THE USER WITH MORE INFORMATION PERTAINING TO THE ERROR CONDITION. ERROR DIAGNOSTICS ARE NO LONGER LIMITED TO ONE LINE IN LENGTH AND ARE STORED IN A FILE INSTEAD OF IN MEMORY AT RUN-TIME. ALSO, ERRORS ARE NO LONGER LOGGED IN A FILE IN THE USER'S DIRECTORY. MORE ABOUT THE NEW ERROR HANDLER CAN BE FOUND IN THE LATEST REVISION OF THE FORMS DOCUMENT.

TO MAINTAIN COMPATIBILITY WITH PREVIOUS RELEASES, A SOURCE OF RTERR$ (THE OLD ERROR HANDLER) HAS BEEN INCLUDED IN THE RUN SUBDIRECTORY UNDER FORMS FOR ANY DEVICE DRIVERS WRITTEN BY USERS WHICH CALL IT. THIS HAS NOT BEEN COMPILED INTO THE FORMS LIBRARY AS IT EXISTS ON THE MASTER DISK.

12 FAP MODIFICATIONS

THE FORMS ADMINISTRATIVE COMMAND PROCESSOR (FAP) NOW STORES AN OWNER NAME ASSOCIATED WITH EVERY FORM DEFINITION ADDED OR REPLACED IN THE FORMS CATALOG. THE OWNER NAME IS THE 6-CHARACTER LOGIN NAME OF THE USER ADDING OR REPLACING THE FORM DEFINITION.

THE CONTROL DIRECTORY FILE FORMATS HAVE BEEN MODIFIED CONSIDERABLY AT REV 15 AND ARE DESCRIBED IN THE LATEST REVISION OF THE FORMS DOCUMENT. BECAUSE OF THE FILE STRUCTURE CHANGES, THE LAST-ACCESSED DATE IS NO LONGER RECORDED (OR PRINTED) BY FAP OR THE RUN-TIME PACKAGE.

13 BUG FIXES, ETC.

1) ALL STANDARD DEVICE DRIVERS NOW CALL C1IN INSTEAD OF T1IN TO DO INDIVIDUAL CHARACTER INPUT FROM THE TERMINAL. THE USER IS WARNED THAT THIS MAY CAUSE SOME PROGRAMS TO FAIL WHICH ARE INVOKED BY COMMAND FILES THAT ARE NEVER TERMINATED WITH A CALL TO COMI$$. THE TELL-TALE SYMPTOM IS A COMINPUT FILE EOF ERROR AFTER THE SCREEN IS CLEARED OR A FORM IS DISPLAYED.
2) THE ROUTINE WHICH PARSED A COMMAND LINE OUTPUT BY THE APPLICATIONS PROGRAM WOULD NOT RECOGNIZE THE LAST ITEM ON THE LINE IF THE LAST CHARACTER IN THE OUTPUT BUFFER WAS NON-BLANK.

3) O$ALO6 IS NO LONGER CALLED BY THE PRINTER OUTPUT ROUTINE (O$FMO6) IF THERE IS NO FORM INVOKED ON THE PRINTER; INSTEAD, THE PRINTER OUTPUT IS IGNORED.
4) THE OWL DEVICE DRIVER DID NOT PROPERLY HANDLE INPUT SEQUENCE AND SIZE ERROR RECOVERY (USUALLY CAUSED BY INPUT BUFFER OVERFLOW).
5) THE OWL DEVICE DRIVER IMPROPERLY HANDLED ERASING A DISPLAYED FIELD IN WHICH THE NODISPLAY BIT WAS SET AFTER THE LAST MODIFY OPERATION.
6) THE RUN-TIME COMMAND VALIDATE IS NOW THE "ACCEPTED" WAY OF ASKING FOR INPUT VALIDATION STATUS. THE STAT COMMAND WILL CONTINUE TO BE SUPPORTED BUT NOT DOCUMENTED.
7) THE RUN-TIME COMMAND PROCESSOR NOW ACCEPTS LOWER CASE COMMANDS.

14 INSTALLING REV 15 FORMS

TO INSTALL REV 15 FORMS, EXECUTE THE COMMAND FILE "C_INST" IN THE FORMS DIRECTORY. THIS COPIES FDL AND FAP TO CMDNCO AND THE LIBRARIES (RFORMS, VFORMS) TO THE LIBRARY UFD. ADDITIONALLY, IF THE USER IS UPGRADING FROM A REV 13 OR 14 FORMS SYSTEM, HE SHOULD RUN THE COMMAND FILE "C_R15", ALSO IN THE FORMS DIRECTORY.

WHEN UPGRADING FROM A REV 13 OR 14 SYSTEM TO REV 15, THE USER NEED NOT RELOAD ANY OF HIS RUN FILES; HOWEVER, TO TAKE ADVANTAGE OF THE SPEED INCREASES AFFORDED BY REV 15, HE MUST RELOAD HIS PROGRAMS WITH THE NEW LIBRARY.

DATE: APRIL 7, 1978

SUBJECT: PRIME FORMS MANAGEMENT SYSTEM, REV 15

1 SCOPE

THIS DOCUMENT DESCRIBES THE PRIME FORMS MANAGEMENT SYSTEM (FORMS) IN IT'S ENTIRETY AT REV 15. IT OBSOLETES ALL PRIOR RELEASES OF PE-T-296.

2 PURPOSE

THE PRIME FORMS MANAGEMENT SYSTEM PROVIDES A DATA FORMATTING SERVICE TO APPLICATIONS PROGRAMS WHICH FACILITATES I/O WITH FORMATTED (PAGE ORIENTED) DEVICES. FORMS ALLOWS AN APPLICATIONS PROGRAM TO COMMUNICATE WITH A PAGE ORIENTED DEVICE AS THOUGH IT WERE RECORD ORIENTED.

3 ABSTRACT

PROGRAMS WRITTEN IN STANDARD LANGUAGES SUCH AS FORTRAN AND COBOL CAN INTERACT WITH FORMATTED (SOMETIMES CALLED "BLOCK MODE") TERMINALS IN ONE OF TWO WAYS. THE USER CAN SUPPLY, IN THE APPLICATION PROGRAM, ALL OF THE CONTROL CHARACTER SEQUENCES TO THE TERMINAL TO OUTPUT EACH INDIVIDUAL FIELD (DATA AREA). THIS USUALLY INVOLVES FIVE OR MORE BYTES OR CURSOR POSITIONING INFORMATION, FIELD IDENTIFIER AND ATTRIBUTE BYTES, THE FIELD DATA, AND A FIELD TERMINATOR BYTE. HE CAN INTERPRET THE CHARACTER INPUT STREAM FROM THE TERMINAL, DISTINGUISH INPUT DATA FROM CONTROL DATA AND PROCESS EACH ACCORDINGLY. THIS IS AN AWKWARD TASK TO PROGRAM IN FORTRAN AND VERGING ON IMPOSSIBLE IN COBOL. ALTHOUGH THERE ARE SHORT CUTS (A SUBROUTINE PACKAGE, FOR EXAMPLE), THE PROGRAMMER WILL FIND HIMSELF MORE CONCERNED ABOUT DEVICE PECULIARITIES THAN WITH THE ORIGINAL TASK HE SET OUT TO ACCOMPLISH. ONCE THE PROGRAM IS WRITTEN, CHANGING THE FORMAT OF THE SCREEN DEFINITION DESCRIBED WITHIN THE PROGRAM IS EXCESSIVELY DIFFICULT. THE PROGRAM MUST BE ALL BUT REWRITTEN SHOULD ANOTHER TERMINAL REQUIRE SUPPORT. IN SHORT, PROGRAM MAINTENANCE IS TIME CONSUMING AND COSTLY.

THE ALTERNATIVE IS THE PRIME FORMS MANAGEMENT SYSTEM. FORMS ALLOWS THE USER TO DESCRIBE HIS DATA FORMATS IN A FORM DEFINITION LANGUAGE COMPLETELY SEPERATE FROM HIS PROGRAM. THE FORM DEFINITION SERVES AS AN INTERFACE BETWEEN THE APPLICATIONS PROGRAM AND THE PAGE ORIENTED DEVICE IN USE. IT DESCRIBES EACH DATA FIELD TRANSFERRED TO OR FROM THE

APPLICATIONS PROGRAM BY IT'S POSITION IN THE INPUT OR OUTPUT RECORD AND RELATES THIS TO THE FIELD'S POSITION ON THE DEVICE, IT'S LENGTH, DISPLAY ATTRIBUTES (BLINK, REVERSE VIDEO, WRITE-PROTECTED, ETC) JUSTIFICATION, VALIDATION, ETC. DATA IS TRANSFERRED BETWEEN THE APPLICATIONS PROGRAM AND THE DEVICE USING READ AND WRITE STATEMENTS IN THE HOST LANGUAGE, USING FORMS AS A "MEDIUM".

FORMS CONSISTS OF THREE PRIMARY COMPONENTS. A FORM DEFINITION LANGUAGE TRANSLATOR (FDL) TRANSLATES SOURCE FORM DEFINITIONS INTO A USABLE BINARY FORM. A CATALOG MAINTENANCE TOOL (FAP, THE FORMS ADMINISTRATIVE COMMAND PROCESSOR) IS USED TO UPDATE A SYSTEM-WIDE FORM DIRECTORY, WHICH CONTAINS ALL FORM DEFINITIONS AVAILABLE FOR USE BY APPLICATIONS PROGRAMS. THE FORMS RUN-TIME PACKAGE (SYSTEM LIBRARY FILES RFORMS FOR 64R MODE AND VFORMS FOR 64V MODE) IS A COLLECTION OF SUBROUTINE WHICH INTERACTS WITH THE APPLICATIONS PROGRAM AND INPUT/OUTPUT DEVICES TO PROVIDE ALL I/O HANDLING AT EXECUTION TIME.

FORMS IS DEVICE INDEPENDENT. THE USER MAY DEFINE A FORM TO BE IMPLEMENTED ON ANY TERMINAL AND/OR THE SYSTEM (SPOOLED) LINE-PRINTER (FORMS WORKS WITH ANY PAGE ORIENTED "TWO-DIMENSIONAL" DEVICE, HARD OR SOFT COPY). MULTIPLE TERMINAL TYPES MAY SIMULTANEOUSLY RUN THE SAME APPLICATIONS PROGRAM, AS PHYSICAL DEVICE SELECTION IS DEFERRED UNTIL EXECUTION TIME. TERMINALS NEED NOT BE EXTREMELY INTELLIGENT TO BE USED WITH FORMS; THEY MUST POSESS FIELD WRITE ENABLE/PROTECT, ABSOLUTE CURSOR POSITIONING, AND BLOCK MODE TRANSMISSION CAPABILITIES AS DESCRIBED IN FULL LATER IN THIS DOCUMENT. PRIME CURRENTLY SUPPORTS A HOST OF DEVICES WHICH MAY BE USED WITH FORMS. SHOULD THE USER WISH TO IMPLEMENT A NON-STANDARD TERMINAL, HE NEED ONLY WRITE A DEVICE DRIVER SUBROUTINE FOLLOWING THE GUIDELINES SET FORTH LATER IN THIS DOCUMENT.

4 SYSTEM CONCEPTS

4.1 FORM DEFINITION

A FORM DEFINITION DESCRIBES USER AND DEVICE DATA FORMATS IN TWO PARTS. THE FIRST PART DESCRIBES THE INPUT/OUTPUT RECORD FORMATS OF THE APPLICATION PROGRAM, I.E. THE LOCATION OF EACH DATA ITEM IN THE RECORD. THE SECOND THEN DESCRIBES HOW EACH OF THESE DATA ITEMS, OR FIELDS, ARE TO APPEAR ON THE DEVICE. TO CLARIFY THIS, CONSIDER THE FOLLOWING EXAMPLE:

A PROGRAMMER IS DESIGNING A SIMPLE INQUIRY PROGRAM WHICH, USING FORMS, ALLOWS HIM TO DISPLAY ENTRIES ON THE TERMINAL FROM A KEYED INDEX FILE. PROGRAM OPERATION WILL CONSIST OF ENTERING AN EMPLOYEE ID NUMBER FROM THE TERMINAL; USING THE GIVEN ID, THE PROGRAM PERFORMS A FILE LOOKUP AND DISPLAYS THE INFORMATION TO WHICH IT PERTAINS. IF THE ID ENTERED IS ZERO OR SPACES, THE PROGRAM WILL EXIT.

IN DESIGNING HIS DATA RECORD, THE PROGRAMMER DECIDES ON THE FOLLOWING FORMAT:

COLUMNS	ITEM NAME	DATA TYPE
1-4	EMPLOYEE ID	NUMERIC
5-34	EMPLOYEE NAME	ALPHABETIC
35-64	STREET ADDRESS	ALPHANUMERIC
65-84	CITY	ALPHABETIC
85-86	STATE	ALPHABETIC
87-91	ZIP CODE	NUMERIC
92-103	PHONE NUMBER	NUMERIC / SPECIAL

OR, GRAPHICALLY REPRESENTED,

```
  1--4   5---34   35------64   65--84   85---86   87-91   92--103
< ID  < NAME  < ADDRESS  < CITY  < STATE  < ZIP  < PHONE  <
```

THIS INFORMATION WILL SERVE AS THE BASIS FOR THE FIRST PART OF THE FORM DEFINITION, KNOWN AS THE DATA STREAM DESCRIPTOR. THE DATA STREAM DESCRIPTOR CONTAINS STREAM (DESCRIPTOR) FIELDS, WHICH DESCRIBE EACH ITEM IN THE USER'S INPUT/OUTPUT RECORD(S). THIS DESCRIPTION MUST INCLUDE THE LENGTH OF THE ITEM AND, IMPLICITLY OR EXPLICITLY, IT'S POSITION WITHIN THE DATA RECORD (STARTING CHARACTER POSITION). THE FIELD DESCRIPTION MAY OPTIONALLY INCLUDE A JUSTIFICATION SPECIFICATION (FORMS WILL PROVIDE LEFT OR RIGHT JUSTIFICATION OR CENTERING, ALONG WITH A ZERO-FILL OR SPACE-FILL OPTION WHEN INPUTTING DATA WITH JUSTIFICATION), AND VALIDATION (FORMS WILL VALIDATE INPUT DATA UNDER A SPECIFIED MASK OR SERIES OF MASKS AND ALLOW THE OPERATOR TO CORRECT THE DATA SHOULD IT BE JUDGED INCORRECT - THIS ALLEVIATES CHARACTER BY CHARACTER VALIDATION BY THE APPLICATIONS PROGRAM).

THE PROGRAMMER MUST THEN DESIGN THE FORMAT OF THE DATA AS IT IS TO APPEAR ON THE DEVICE. HE MUST TAKE INTO CONSIDERATION THE DISPLAY SIZE (NUMBER OF COLUMNS AND LINES AVAILABLE), ATTRIBUTES WHICH HE MAY APPLY TO EACH DATA FIELD (WRITE ENABLED, BLINKED, REVERSE VIDEO, ETC), LENGTH OF EACH FIELD AS IT IS TO BE DISPLAYED (WHICH MAY DIFFER FROM THE LENGTH OF THE FIELD IN THE INPUT OR OUTPUT RECORD) AND ANY FIELD PROXIMITY RESTRICTIONS IMPOSED BY THE DEVICE (SIMPLY STATED, SOME TERMINALS REQUIRE TWO FIELDS TO BE SEPERATED BY ONE OR MORE BLANKS).

WHILE ASSIGNING PHYSICAL DEVICE POSITIONS AND ATTRIBUTES FOR EACH DATA FIELD IN THE DATA STREAM DESCRIPTOR (DESCRIBED ABOVE), THE USER MAY ALSO SPECIFY TITLES, CALLED LITERAL DATA WHICH WILL BE DESCRIBED IN THE FORM DEFINITION AND APPEAR ON THE DEVICE WITH HIS APPLICATIONS PROGRAM DATA. THIS LITERAL DATA IS USUSALLY USED TO DESCRIBE OR IDENTIFY DATA FIELDS WHICH FOLLOW.

THE PROGRAMMER NOW LAYS OUT THE FOLLOWING INFORMATION WHICH WILL BE USED TO CONSTRUCT THE SECOND HALF OF THE FORM DEFINITION, KNOWN AS THE DEVICE FORMAT DESCRIPTOR:

LINE	COLUMN	CONTENTS	LENGTH	ATTRIBUTES

2	2	'EMPLOYEE ID'	11	WRITE-PROTECTED
2	20	EMPLOYEE ID	4	WRITE-ENABLED
4	2	'NAME'	4	WRITE-PROTECTED
4	20	EMPLOYEE NAME	30	WRITE-PROTECTED
6	2	'ADDRESS'	3	WRITE-PROTECTED
6	20	STREET ADDRESS	30	WRITE-PROTECTED
7	20	CITY	20	WRITE-PROTECTED
7	45	STATE	2	WRITE-PROTECTED
7	50	ZIP	3	WRITE-PROTECTED
9	2	'HOME PHONE'	10	WRITE-PROTECTED
9	20	PHONE NUMBER	12	WRITE-PROTECTED

ACCORDING TO THE ABOVE INFORMATION, WHEN THE FORM IS OUTPUT TO THE TERMINAL WHEN THE APPLICATIONS PROGRAM IS EXECUTED, IT SHOULD LOOK AS FOLLOWS:

```
    ....*....1....*....2....*....3....*....4....*....5....*....
   ------------------------------------------------------------
1 <
2 <  EMPLOYEE ID          ____
3 <
4 <  NAME                 ********************************
5 <
6 <  ADDRESS              ********************************
7 <                       **********************     **   *****
8 <
9 <  HOME PHONE           ************
```

(THE LINE AND COLUMN MARKERS HAVE BEEN PROVIDED FOR EASE OF POSITION IDENTIFICATION ONLY; THEY WILL NOT APPEAR WHEN THE FORM IS OUTPUT. THE UNDERLINES REPRESENT WRITE ENABLED DATA, I.E. THAT WHICH THE OPERATOR MAY MODIFY. THE ASTERISKS REPRESENT WRITE PROTECTED DATA, THAT WHICH MAY NOT BE MODIFIED BY THE OPERATOR.)

4.2 MAPPING

MAPPING CREATES A RELATIONSHIP BETWEEN A STREAM DESCRIPTOR FIELD AND A FORMAT DESCRIPTOR FIELD. IT BINDS THE RECORD POSITION AND ITEM LENGTH INFORMATION (, ETC) CONTAINED IN THE STREAM FIELD TO THE DEVICE POSITION AND DISPLAY ATTRIBUTE INFORMATION CONTAINED IN THE FORMAT FIELD. A "BOUND" FIELD CONTAINS ALL INFORMATION AVAILABLE REGARDING THE DATA THEREIN: IT'S RECORD AND DEVICE POSITION AND LENGTH, JUSTIFICATION, VALIDATION, AND DISPLAY ATTRIBUTES.

FIELDS IN THE DATA STREAM DESCRIPTOR ARE MAPPED BY NAME TO CORRESPONDING ITEMS IN THE DEVICE FORMAT DESCRIPTOR. THESE NAMES NEED NOT MATCH THE DATA NAMES USED WITHIN THE APPLICATIONS PROGRAM, BUT COMMON SENSE AND GOOD PROGRAMMING PRACTICE DICTATE THAT THE USER SHOULD, WHEREVER PRACTICABLE KEEP THE NAMES AS CLOSELY MATCHED AS POSSIBLE (A DATA ITEM CALLED EMPLOYEE-NAME IN A COBOL PROGRAM MIGHT BE REPRESENTED BY A FIELD CALLED EMPLNAME IN A FORM DEFINITION (AS THERE'S AN EIGHT CHARACTER MAXIMUM FOR NAMES WITHIN FORMS), BUT SHOULD NOT BE REPRESENTED BY EMPLID OR SOMETHING MORE OBSCURE).

STREAM DESCRIPTOR FIELDS WHICH ARE MAPPED (THERE ARE SOME WHICH ARE NOT; THESE WILL BE DISCUSSED LATER) CONTAIN THE NAME OF THE FORMAT DESCRIPTOR FIELD TO WHICH THEY MAP IN THE FIELD DEFINITION. THEREFORE, FORMAT DESCRIPTOR FIELDS WHICH ARE "MAPPED TO" FROM STREAM DESCRIPTOR FIELDS ARE ALSO ASSIGNED THE NAME. TWO FIELDS ARE BOUND WHEN THE NAME SPECIFIED IN THE STREAM FIELD DESCRIPTOR AND THE FIELD NAME SPECIFIED IN THE FORMAT FIELD DESCRIPTOR ARE IDENTICAL.

BECAUSE THE TWO PARTS OF THE FORM DEFINITION ARE DESCRIBED SEPERATELY TO THE FORM DEFINITION LANGUAGE, THEY MUST IN SOME WAY BE RELATED TO EACH OTHER. AS STREAM FIELDS ARE MAPPED TO FORMAT FIELDS, SO ARE STREAM DESCRIPTORS TO FORMAT DESCRIPTORS. EACH DATA STREAM DESCRIPTOR AVAILABLE WITHIN THE SYSTEM HAS ASSOCIATED WITH IT A UNIQUE NAME. (WHEN THE APPLICATIONS PROGRAM DESIRES TO USE A FORM DEFINITION, IT IS IDENTIFIED BY THIS NAME.) FORMAT DESCRIPTORS WHICH CORRESPOND TO A PARTICULAR STREAM DESCRIPTOR ARE ASSIGNED AN IDENTICAL NAME.

A BRIEF EXAMPLE MAY PROVE USEFUL. REFER TO THE EXAMPLE DESCRIBED EARLIER IN SECTION 4. THIS SUBSECTION SIMPLY SAYS THAT THE STREAM DESCRIPTOR MUST BE NAMED THE SAME AS THE FORMAT DESCRIPTOR. ALSO, THAT THE STREAM DESCRIPTOR FIELD DESCRIBING THE EMPLOYEE ID NUMBER MUST MAP TO THE FORMAT FIELD CONTAINING THE EMPLOYEE ID; THE STREAM FIELD CONTAINING THE EMPLOYEE NAME MUST MAP TO THE FORMAT FIELD DESCRIBING THE EMPLOYEE NAME, ETC.

4.3 APPLICATION PROGRAMMING WITH FORMS

WRITING AN APPLICATION PROGRAM USING FORMS TO PROVIDE I/O PROCESSING IS LITTLE DIFFERENT FROM USING THE STANDARD (RECORD ORIENTED) I/O PROCESSORS.

TERMINAL I/O IS ACCOMPLISHED THROUGH READS AND WRITES TO LOGICAL UNIT ONE IN FORTRAN OR TO A DEVICE SELECTED AND ASSIGNED TO THE TERMINAL IN COBOL. OUTPUT TO THE SYSTEM LINE-PRINTER IS DONE WITH WRITES TO LOGICAL UNIT FOUR IN FORTRAN (THE STANDARD PRINTER OUTPUT UNIT). IN COBOL, HOWEVER, A FILE MUST BE ASSIGNED TO THE DEVICE NAMED OFFLINE-PRINTER, AS FILES ASSIGNED TO THE PRINTER ARE ACTUALLY OUTPUT TO A DISK FILE IN THE USER'S UFD.

CONSIDER THE FOLLOWING COBOL RECORD DESCRIPTION (IT CORRESPONDS TO THE RECORD DESCRIPTION IN THE EXAMPLE SHOWN EARLIER):

```
01  EMPLOYEE-IO-RECORD.
    02  EMPLOYEE-ID            PIC 9(4).
    02  EMPLOYEE-NAME          PIC X(30).
    02  STREET-ADDRESS         PIC X(30).
    02  CITY                   PIC X(20).
    02  STATE                  PIC X(2).
    02  ZIP                    PIC 9(5).
    02  PHONE                  PIC X(12).
```

THE DATA CONTAINED WITHIN THIS RECORD MAY BE OUTPUT TO THE TERMINAL WITH THE FOLLOWING WRITE STATEMENT:

```
WRITE EMPLOYEE-IO-RECORD.
```

IF FORMS IS NOT IN USE, THE DATA OUTPUT MAY LOOK LIKE THIS (NOTE THAT THE COLUMN ALLOCATION IS NOT EXACT DUE TO SPACE RESTRICTIONS):

```
0287JOHN SMITH   123 MEADOW LANE  ANYWHERE  MA01760617-655-5012
```

UNDER FORMS, THE SAME WRITE STATEMENT WOULD CAUSE THE DATA CONTAINED IN THE RECORD TO APPEAR AT THE APPROPRIATE LOCATIONS ON THE TERMINAL, AS DESCRIBED BY THE FORM DEFINITION.

INPUT WORKS IN MUCH THE SAME WAY. WHEN FORMS IS NOT IN USE, A READ STATEMENT WAITS FOR THE OPERATOR TO ENTER ONE LINE OF TEXT, WHICH IS THEN RETURNED IN THE INPUT RECORD. WITH FORMS, A READ STATEMENT WAITS FOR THE OPERATOR TO ENTER DATA INTO THE WRITE ENABLED FIELDS AND, AFTER HE DEPRESSES THE 'TRANSMIT DATA' KEY (OR HOWEVER IT MAY BE DESIGNATED ON THE PARTICULAR TERMINAL TYPE IN USE), INPUTS THE DATA FROM THE TERMINAL, CONSTRUCTING THE INPUT RECORD AS THE DATA IS READ, AND RETURNS THE RECORD TO THE APPLICATIONS PROGRAM. IN THIS INSTANCE, THE APPLICATION PROGRAM IS INSENSITIVE TO WHETHER OR NOT FORMS IS IN USE.

DIFFERENCES BETWEEN STANDARD I/O AND FORMS I/O ARISE WHEN THE USER PROGRAM IS REQUIRED TO 'COMMUNICATE' WITH THE FORMS RUN-TIME COMMAND PROCESSOR. THIS MODULE IS USED FOR INVOKING A PARTICULAR FORM DEFINITION, MODIFYING FIELD ATTRIBUTES 'ON THE FLY', RELEASING THE FORM, AND OTHER CONTROL OPERATIONS WHICH ARE DISCUSSED LATER.

RUN-TIME COMMANDS ARE EXECUTED BY WRITING THEM (YES, USING A WRITE STATEMENT) TO THE DEVICE FOR WHICH THEY ARE INTENDED! ALL OUTPUT FOR THE TERMINAL AND LINEPRINTER IS FILTERED THROUGH FORMS. IF FORMS "SEES" A RECORD CONTAINING A COMMAND (WHICH IS IDENTIFIED BY TWO HASH MARKS IN THE FIRST TWO COLUMNS) THE COMMAND LINE IS PARSED AND THE COMMAND INTERPRETED AND EXECUTED BY FORMS.

FOR EXAMPLE, IF THE PROGRAMMER WANTED TO USE THE FORM DEFINITION 'MYFORM' ON THE LINE-PRINTER, HE MIGHT WRITE (IN FORTRAN),

```
      WRITE (4,100)
100   FORMAT ('##INVOKE MYFORM')
```

SPECIFIC COMMANDS AND SYNTAX ARE DISCUSSED LATER.

IT IS WORTHWHILE TO NOTE THAT FORMS HAS BOTH AN "ON" AND AN "OFF" MODE. WHILE NO FORM IS IN USE ON A DEVICE, FORMS IS CONSIDERED TO BE INACTIVE ("OFF") FOR THAT DEVICE. WHEN FORMS IS INACTIVE, ALL DATA I/O IS HANDLED NORMALLY, I.E. IN RECORD I/O MODE. WHEN A FORM DEFINITION IS INVOKED, ALL DATA I/O IS HANDLED BY FORMS.

4.4 SUBSTREAMS

UNTIL NOW, WE HAVE DISCUSSED FORM DEFINITION AND USAGE WHICH INVOLVES ONLY ONE DATA RECORD. SHOULD THE USER DESIRE, HE MAY LOGICALLY SEPERATE HIS DATA INTO SEVERAL RECORDS, CALLED SUBSTREAMS, WITHIN THE FORM DEFINITION. EACH SUBSTREAM IS TRANSFERRED AS AN INDIVIDUAL RECORD WHEN THE APPLICATIONS PROGRAM IS PERFORMING INPUT AND OUTPUT.

STREAM DESCRIPTOR FIELDS MAY BE SPECIFIED (IN FDL SOURCE FORM) TO BE "INPUT ONLY" OR "OUTPUT ONLY" INSTEAD OF THE DEFAULT "INPUT/OUTPUT". AS THE NAME IMPLIES, INPUT ONLY FIELDS ARE PROCESSED ONLY ON INPUT (READ) OPERATIONS AND IGNORED ON WRITE OPERATIONS. CONVERSELY, OUTPUT ONLY FIELDS ARE PROCESSED ONLY ON OUTPUT (WRITE) OPERATIONS. WHEN THE APPLICATION PROGRAM WRITES DATA TO A FORM WHICH CONTAINS MULTIPLE SUBSTREAMS, THE FIRST RECORD OUTPUT IS INTERPRETED ACCORDING TO THE FIRST SUBSTREAM WHICH CONTAINS ONE OR MORE OUTPUT FIELDS, I.E. THE FIRST SUBSTREAM WHICH CONTAINS OTHER THAN ALL INPUT ONLY FIELDS. THE SECOND OUTPUT RECORD, IF ANY, IS INTERPRETED ACCORDING TO THE NEXT SEQUENTIAL SUBSTREAM WHICH CONTAINS AT LEAST ONE OUTPUT FIELD, AND SO ON. THE REVERSE IS TRUE FOR INPUT OPERATIONS.

IF THE USER ATTEMPTS TO PERFORM A DATA TRANSFER AND THE NEXT SUBSTREAM TO BE PROCESSED IS BEYOND THE LAST SUBSTREAM DEFINED IN THE FORM, A WRAP-AROUND OCCURS TO THE FIRST SUBSTREAM.

SHOULD THE USER BE PROCESSING DATA IN ONE DIRECTION OF TRANSFER AND THEN SWITCH TO THE OPPOSITE DIRECTION, HE WILL START THE NEW DATA TRANSFER WITH THE FIRST SUBSTREAM WHICH CONTAINS DATA TO BE PROCESSED BY THIS TYPE OF TRANSFER. FOR EXAMPLE, IF THE USER WAS PERFORMING READ OPERATIONS AND STOPPED ON THE SECOND SUBSTREAM OF A FORM DEFINITION WHICH CONTAINS SIX SUBSTREAMS AND THEN EXECUTES A WRITE OPERATION, THE DATA WILL BE WRITTEN TO THE FIRST SUBSTREAM WHICH CONTAINS AT LEAST ONE OUTPUT FIELD (AS DESCRIBED ABOVE).

THE USER MAY INTERRUPT THIS SEQUENTIAL METHOD OF SUBSTREAM HANDLING AND SPECIFY, IN A FORMS RUN-TIME COMMAND, THAT A SPECIFIC SUBSTREAM IS TO BE OUTPUT ACCORDING TO THE NEXT WRITE STATEMENT OR INPUT ACCORDING TO THE NEXT READ.

NOTE: THIS SECTION MAY SEEM COMPLEX FOR THE CASUAL OR FIRST TIME READER; HOWEVER, ONCE HAVING GAINED FAMILIARITY WITH SIMPLE SINGLE-RECORD FORM DEFINITIONS, SUBSTREAMS SHOULD APPEAR SOMEWHAT EASIER TO UNDERSTAND.

4.5 PROGRAM / DEVICE INTERACTION

APPLICATIONS PROGRAMS WHICH ARE USED WITH FORMS ARE ENTIRELY INSENSITIVE TO THE TERMINAL TYPE IN USE. WITHIN EACH "FORMAT DESCRIPTOR" ARE SEPERATE DATA FORMAT DESCRIPTIONS FOR EACH DEVICE ON WHICH THAT FORMAT DESCRIPTOR IS TO BE USED. STATED MORE SIMPLY, IF FORMAT DESCRIPTOR "XYZ" IS GOING TO BE USED ON DEVICES "A", "B", AND

"C", A SEPERATE DATA FORMAT DESCRIPTION EXISTS WITHIN FORMAT DESCRIPTOR "XYZ" FOR EACH OF THESE DEVICES.

PHYSICAL TERMINAL TYPE SELECTION OCCURS AT EXECUTION TIME. A FILE EXISTS WITHIN THE FORMS DIRECTORY (A UFD WITHIN THE SYSTEM USED EXCLUSIVELY BY FORMS) WHICH DESCRIBES THE TYPE OF TERMINAL ASSOCIATED WITH EACH OF THE 64 DISCRETE USERS ON THE SYSTEM. THE CURRENT TERMINAL TYPE IS READ FROM THIS FILE WHEN FORMS IS INITIALIZED. WHEN ANY FORM DEFINITION IS INVOKED, THE TERMINAL TYPE IS REFERENCED TO DETERMINE THE CORRECT FORMAT DESCRIPTOR TO BE USED. (THIS, OBVIOUSLY, DOES NOT APPLY WHEN INVOKING A FORM DEFINITION FOR USE WITH THE LINEPRINTER.) THE TERMINAL DESCRIPTION FILE MAY BE MODIFIED AT ANY TIME USING THE FORMS UTILITY PROGRAM (FAP).

THE FORMS LIBRARIES CONTAIN BOTH PRIME-SUPPLIED AND USER-WRITTEN DEVICE DRIVERS SUPPORTED WITHIN THE INSTALLATION. DEVICE DRIVERS ARE SUBROUTINES WHICH PROVIDE ALL LOW-LEVEL I/O FUNCTIONS REQUIRED TO USE THE TERMINAL WITH FORMS. WHEN THE APPLICATIONS PROGRAM IS LOADED WITH THE FORMS LIBRARY, IT IS EXECUTABLE FROM ANY TERMINAL FOR WHICH THERE WAS A DEVICE DRIVER AT LOAD TIME. A COMPLETE DESCRIPTION OF DEVICE DRIVERS IS PROVIDED ELSEWHERE IN THIS DOCUMENT.

4.6 FORM DEFINITION CATALOG

THE FORM DEFINITION CATALOG IS A SEGMENT DIRECTORY WHICH CONTAINS THE BINARY REPRESENTATIONS (GENERATED BY FDL) OF ALL STREAM AND FORMAT DESCRIPTORS AVAILABLE WITHIN THE SYSTEM. AFTER A FORM DEFINITION IS TRANSLATED BY FDL, IT IS ADDED TO (OR UPDATED IN) THE FORM DEFINITION CATALOG WITH THE FAP MAINTENANCE PROGRAM. THE FORMS CATALOG DIRECTORY RESIDES WITH THE OTHER FORMS SYSTEM RELATED FILES IN THE FORMS SYSTEM UFD (CALLED "FORMS*").

5 SYSTEM SOFTWARE COMPONENTS

5.1 FORM DEFINITION LANGUAGE TRANSLATOR

THE FORM DEFINITION LANGUAGE (FDL) TRANSLATOR TRANSLATES SOURCE FORM DESCRIPTIONS INTO BINARY DESCRIPTIONS WHICH, ONCE ADDED TO THE FORMS CATALOG, ARE AVAILABLE FOR USE BY APPLICATIONS PROGRAMS. ADDITIONALLY, IT PRODUCES A LISTING FILE CONSISTING OF THE ORIGINAL SOURCE TEXT, ERROR DIAGNOSTICS PRODUCED DURING TRANSLATION, SYNONYM AND REPEAT BLOCK EXPANSION (EXPLAINED LATER), AND A GRAPHICAL AND TABULAR DESCRIPTION OF THE DATA FORMATS DESCRIBED WITHIN THE FORM DEFINITION SOURCE TEXT.

THE FORMS DEFINITION LANGUAGE AND TRANSLATOR ARE DESCRIBED IN FULL LATER IN THIS DOCUMENT.

5.2 FORMS ADMINISTRATIVE COMMAND PROCESSOR

THE FORMS ADMINISTRATIVE COMMAND PROCESSOR (FAP) PROVIDES A FACILITY FOR THE SYSTEM ADMINISTRATOR TO ADD, UPDATE, LIST, AND REMOVE FORM DEFINITIONS FROM THE CATALOG, LIST AND UPDATE THE TERMINAL CONFIGURATION DIRECTORY (THE FILE WHICH CONTAINS A TERMINAL TYPE FOR EACH FORMS USER ON THE SYSTEM), AND TO REGENERATE THE DEVICE DRIVER TABLES WHEN A NEW DEVICE DRIVER IS ADDED TO THE SYSTEM. FAP ALLOWS THE USER TO LOG ANY CHANGES MADE TO THE FORM DEFINITION CATALOG AND THE TERMINAL CONFIGURATION DIRECTORY IN A SEPERATE DISK FILE TO PROVIDE A COMPLETE RECORD OF ALL UPDATES TO THE FORMS SYSTEM FILES.

FAP COMMANDS AND SYNTAX ARE DESCRIBED LATER IN THIS DOCUMENT.

5.3 FORMS RUN-TIME LIBRARY

THE FORMS LIBRARY CONTAINS SUBROUTINES WHICH PROVIDE THE REQUIRED FUNCTIONALITY TO INTERFACE THE APPLICATIONS PROGRAM TO THE TERMINAL USING A PREDEFINED FORM DEFINITION. THIS INCLUDES USER COMMAND HANDLING, FORM LOOKUP, INTERNAL DATA MANIPULATION, JUSTIFICATION, VALIDATION, TERMINAL I/O, ETC.

THE FORMS LIBRARY CONTAINS REPLACEMENTS FOR TWO IOCS TABLES WITHIN THE FORTRAN LIBRARY (THE WRITE ASCII AND READ ASCII TABLES) AND THEREFORE MUST BE LOADED PRIOR TO THE FORTRAN LIBRARY. ALSO, THE USER SHOULD BE AWARE THAT THE LIBRARY CONTAINS INITIALIZED COMMON AND IS THEREFORE PRONE TO A MEMORY OVERFLOW (MO) ERROR MESSAGE FROM LOAD OR HILOAD IF LOADED INCORRECTLY. THE USER WILL NOT ENCOUNTER THIS PROBLEM WITH THE V-MODE AND R-MODE VIRTUAL LOADERS (VLOAD AND SEG).

TWO FORMS LIBRARY FILES EXIST, ONE FOR 32R-64R MODE (RFORMS) AND THE OTHER FOR 64V MODE (VFORMS).

A TYPICAL LOAD SEQUENCE FOR A 64R MODE COBOL PROGRAM WHICH USES KI/DA AND FORMS MIGHT BE:

```
OK, LOAD                    INVOKE LOADER
GO
$ LOAD B_MYPROG             LOAD MAIN PROGRAM
$ LIBRARY COBKID            LOAD COBOL LIB WITH KI/DA
$ LIBRARY RFORMS            LOAD FORMS LIB
$ LIBRARY SPOOL$            SPOOL SUBROUTINE PACKAGE
$ LIBRARY                   FORTRAN LIBRARY
LOAD COMPLETE
$ SAVE *MYPROG              SAVE MEMORY IMAGE FILE
$ QUIT                      EXIT FROM LOAD

OK,
```

6 FORMS DEFINITION LANGUAGE

6.1 GENERAL SYNTAX

FDL SUPPORTS A FREE-FORMAT INPUT LINE, MUCH LIKE THE PRIME MACRO ASSEMBLER (PMA).

ALL FORM DESCRIPTOR, SUBSTREAM, AND FIELD NAMES START IN THE FIRST CHARACTER POSITION OF THE LINE AND ARE FOLLOWED BY AT LEAST ONE SPACE. DESCRIPTOR STATEMENTS MAY START ANYWHERE AFTER COLUMN 1, AND OCCUPY COLUMNS 2 THRU 72. COLUMNS 73 THRU 80 ARE IGNORED. ITEMS IN THE INPUT LINE MUST BE SEPERATED BY EITHER A SPACE OR A COMMA UNLESS OTHERWISE NOTED. LOWER CASE CHARACTERS ARE MAPPED TO UPPER CASE, WITH THE EXCEPTION OF CHARACTERS IN A LITERAL STRING (ENCLOSED WITHIN SINGLE QUOTES).

SHOULD AN INPUT RECORD CONTAIN TOO MANY CHARACTERS TO FIT ON ONE LINE, THE PROGRAMMER MAY CONTINUE HIS SOURCE TEXT BY PLACING A SEMICOLON (;) AS THE LAST CHARACTER OF THE OFFENDING LINE. NOTE THAT INPUT ITEMS (WORDS, TEXT STRINGS, ETC.) MAY NOT BE SPLIT ACROSS 2 LINES. THERE IS NO LIMIT TO THE NUMBER OF CONTINUATION LINES IN A SOURCE RECORD; THERE IS HOWEVER, A 240 CHARACTER LIMIT PER RECORD.

IF THE FIRST CHARACTER OF A LINE IS AN ASTERISK, THAT LINE IS TREATED AS A COMMENT, LISTED IN THE OUTPUT FILE AND IGNORED. IF THE FIRST CHARACTER IS A SINGLE QUOTE ('), THE LINE IS TREATED AS A COMMENT, BUT CAUSES AN EJECT PAGE IN THE LISTING AND BECOMES THE NEW PAGE HEADER.

IN ADDITION TO FULL LINE COMMENTS (LINES BEGINNING WITH AN ASTERISK OR SINGLE QUOTE), IN-LINE COMMENTS ARE SUPPORTED. IN-LINE COMMENTS

ARE PRECEDED BY A FORE-SLASH AND ASTERISK ('/*') AND FOLLOWED BY AN ASTERISK AND A FORE-SLASH ('*/'). SHOULD THE PROGRAMMER PLACE THE IN-LINE COMMENT AS THE LAST ITEM ON THE LINE, THE TERMINATING CHARACTERS ('*/') MAY BE OMITTED. NOTE THAT, UNLIKE FORTRAN, IN-LINE COMMENTS MAY NOT OCCUR WITHIN AN ITEM (EG, IN THE MIDDLE OF A NAME OR TEXT STRING).

EXAMPLES:

```
* THIS IS A COMMENT LINE

' THIS WILL CAUSE A PAGE-EJECT AND WILL BECOME THE NEW HEADER

LABEL   FIELD ABC, LENGTH 6     /* THIS IS AN IN-LINE COMMENT

LABEL   FIELD ABC, /*THIS TOO IS AN IN-LINE COMMENT*/ LENGTH 6

NAME    FIELD 'FOUR SCORE AND SEVEN YEARS AGO... ' ;
        POSITION (10,10) PROTECT /* CONTINUATION LINE
```

6.2 NAMING CONVENTIONS

THE RULES FOR NAMING FORM DESCRIPTORS, FIELDS, AND SUBSTREAMS ARE:

- NAME LENGTH: 1-8 CHARACTERS
- FIRST CHARACTER MUST BE ALPHABETIC
- PERMITTED CHARACTERS: A-Z, 0-9

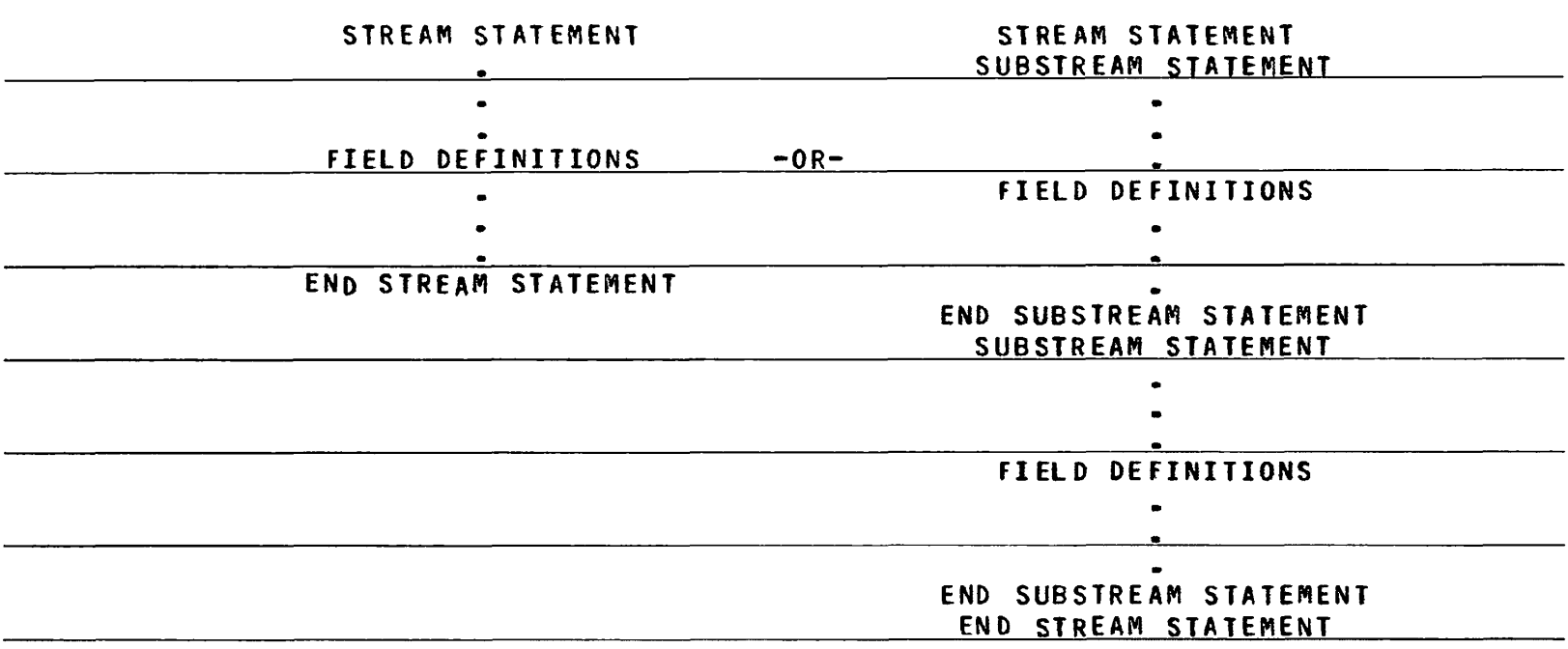
EXAMPLES:

PERMITTED	NOT PERMITTED	WHY
SHIPFORM	GAZORKLEFORM	NAME TOO LONG
FORM5	5FORM	BAD 1ST CHARACTER (5)
AMTOWED	OWED$	ILLEGAL CHARACTER ($)

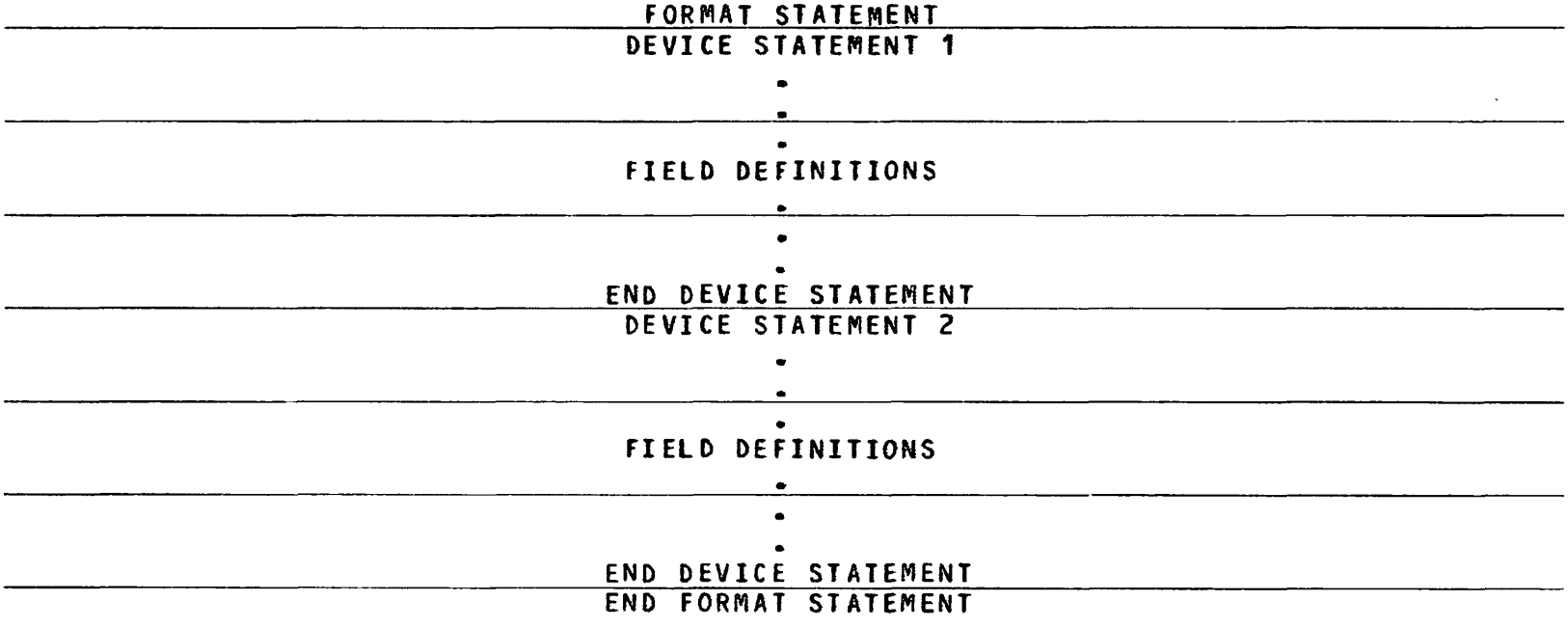
6.3 FORM DESCRIPTION STRUCTURE

THE FOLLOWING DIAGRAMS REPRESENT THE VARIOUS FORM DEFINITION STRUCTURES FOR BOTH THE STREAM DESCRIPTOR AND FORMAT DESCIPTOR.

STREAM_DESCRIPTOR



FORMAT_DESCRIPTOR



6.4 FORM DEFINITION DELIMITER STATEMENTS

THE FDL STATEMENTS DESCRIBED BELOW ARE USED TO SPECIFY THE BEGINNING AND END OF A FORM DEFINITION OR A SECTION OF A FORM DEFINITION. THESE STATEMENTS DO NOT DESCRIBE DATA FORMATS BUT RATHER ARE USED TO IDENTIFY STREAM AND FORMAT DESCRIPTORS, SUBSTREAMS, AND DEVICE DESCRIPTIONS WITHIN A FORMAT DESCRIPTOR.

6.4.1 STREAM STATEMENT

THIS STATEMENT DEFINES THE BEGINNING OF A DATA STREAM DESCRIPTOR. THE NAME FIELD MUST CONTAIN A UNIQUE STREAM DESCRIPTOR NAME, I.E. ONE WHICH DOES NOT CONFLICT WITH ANY OTHER STREAM DESCRIPTOR DEFINED WITHIN THE SYSTEM.

EXAMPLE:

```
SHIPFORM    STREAM
```

6.4.2 END STREAM STATEMENT

THIS STATEMENT DEFINES THE END OF A DATA STREAM DESCRIPTOR.

EXAMPLE:

```
            END STREAM
```

6.4.3 SUBSTREAM STATEMENT

THE SUBSTREAM STATEMENT DEFINES THE BEGINNING OF A SUBSTREAM DESCRIPTION. IF A NAME IS SUPPLIED IN THE NAME FIELD, THE APPLICATIONS PROGRAM WILL BE ABLE TO TRANSFER DATA TO AND FROM THIS SUBSTREAM BY REFERENCING IT BY THE SUPPLIED NAME (SEE THE GENERAL SUBSTREAM DESCRIPTION AND THE RUN-TIME COMMAND DESCRIPTION, ELSEWHERE IN THIS DOCUMENT.)

EXAMPLES:

```
USERDATA    SUBSTREAM

            SUBSTREAM
```

6.4.4 END SUBSTREAM STATEMENT

THIS STATEMENT TERMINATES A SUBSTREAM DESCRIPTION. EACH SUBSTREAM STATEMENT MUST HAVE A CORRESPONDING END SUBSTREAM STATEMENT.

EXAMPLE:

```
                END SUBSTREAM
```

6.4.5 FORMAT STATEMENT

THE FORMAT STATEMENT DEFINES THE BEGINNING OF A DEVICE FORMAT DESCRIPTOR. THE CONTENTS OF THE NAME FIELD DEFINES THE NAME OF THE FORMAT DESCRIPTOR. THIS MUST BE EQUIVALENT TO THAT OF THE STREAM DESCRIPTOR WITH WHICH THIS FORMAT DESCRIPTOR WILL BE USED.

EXAMPLE:

```
   USERDATA    FORMAT
```

6.4.6 END FORMAT STATEMENT

THIS STATEMENT TERMINATES THE DEVICE FORMAT DESCRIPTOR AND MUST BE THE LAST STATEMENT THEREIN.

EXAMPLE:

```
                END FORMAT
```

6.4.7 DEVICE STATEMENT

THE DEVICE STATEMENT SPECIFIES THE NAME OF THE DEVICE TO WHICH THE FOLLOWING FIELD DEFINITIONS PERTAIN. IT IS USED IN THE FORMAT DESCRIPTOR IMMEDIATELY FOLLOWING A FORMAT OR END DEVICE STATEMENT (SEE 'FORM DEFINITION STRUCTURE', ABOVE).

EXAMPLE:

```
                DEVICE VISTAR3
```

6.4.8 END DEVICE STATEMENT

THIS STATEMENT DEFINES THE END OF A DEVICE DESCRIPTION WITHIN A FORMAT DESCRIPTOR.

EXAMPLE:

```
                END DEVICE
```

6.5 STREAM DESCRIPTOR FIELD DEFINITION

THE FIELDS DEFINED WITHIN THE STREAM DESCRIPTOR IDENTIFY THE LOCATION OF EACH DATA ITEM WITHIN THE INPUT OR OUTPUT RECORD, IT'S LENGTH, AND OPTIONAL JUSTIFICATION AND VALIDATION.

STREAM FIELDS MAY BE DEFINED TO BE INPUT ONLY, OUTPUT ONLY, OR INPUT-OUTPUT (DEFAULT). AS THE NAME IMPLIES, INPUT ONLY FIELDS ARE PROCESSED ON INPUT OPERATIONS ONLY, THEY ARE IGNORED ON OUTPUT. THE REVERSE IS TRUE FOR OUTPUT ONLY FIELDS. INPUT-OUTPUT FIELDS ARE PROCESSED ON BOTH INPUT AND OUTPUT OPERATIONS. USING INPUT ONLY AND OUTPUT ONLY FIELDS, THE PROGRAMMER MAY DESCRIBE SEPERATE INPUT AND OUTPUT RECORD FORMATS IN A SINGLE STREAM DESCRIPTOR.

THERE ARE SIX TYPES OF STREAM DESCRIPTOR FIELDS. EACH EITHER DESCRIBES AN ITEM WITHIN THE USER'S DATA RECORD OR DESCRIBES A LITERAL STRING TO BE MAPPED TO A FIELD DEFINED IN THE DEVICE DESCRIPTOR. THE FIELD TYPES ARE:

- DIRECT: A DIRECT FIELD MAPS THE DATA ITEM IN THE THE USER'S INPUT OR OUTPUT RECORD TO THE NAMED DEVICE FORMAT DESCRIPTOR FIELD.

  FDL SPECIFICATION:

  ```
  FIELD DF-FIELD-NAME
  ```

- INPUT_LITERAL: AN INPUT LITERAL FIELD RETURNS A LITERAL STRING TO THE USER'S DATA RECORD ON AN INPUT OPERATION. IT IS IGNORED ON OUTPUT OPERATIONS.

  FDL SPECIFICATION:

  ```
  FIELD 'LITERAL TEXT STRING'
  ```

- OUTPUT_LITERAL: THIS FIELD TYPE DEFINES A LITERAL TEXT STRING TO BE MAPPED INTO A DEVICE FORMAT FIELD ON OUTPUT OPERATIONS. IT IS IGNORED ON INPUT OPERATIONS. NO DATA IS TRANSFERRED TO OR FROM THE USER'S INPUT/OUTPUT RECORD.

  FDL SPECIFICATION:

  ```
  FIELD (DF-FIELD-NAME,'LITERAL TEXT STRING'), OUTPUT
  ```

- INPUT_EMPTY_CONDITIONAL: AN INPUT EMPTY CONDITIONAL (IEC) FIELD FUNCTIONS THE SAME AS AN INPUT ONLY OR INPUT-OUTPUT DIRECT FIELD WITH ONE EXCEPTION. IF THE DATA FIELD DISPLAYED ON THE DEVICE CONTAINS SPACES, THE SUPPLIED LITERAL STRING IS RETURNED INSTEAD OF THE BLANKS. IEC FIELDS MAY NOT BE OUTPUT ONLY. ALSO, IEC FIELDS REQUIRE AN INPUT/OUTPUT SPECIFICATION.

  FDL SPECIFICATION:

```
FIELD (DF-FIELD-NAME,'LITERAL TEXT STRING'), IOSPEC
```

- FILLER: FIELDS DEFINED AS FILLERS PERFORM NO DATA TRANSFER BETWEEN THE APPLICATIONS PROGRAM AND THE DEVICE. THEY SERVE ONLY TO DEFINE A GAP IN THE INPUT OR OUTPUT RECORD. ON INPUT AND/OR OUTPUT OPERATIONS, THE NUMBER OF CHARACTERS DESIGNATED BY THE LENGTH PARAMETER IN THE FILLER FIELD DEFINITION ARE SKIPPED.

  FDL SPECIFICATION:

```
FIELD FILLER
```

- SYSTEM INFORMATION: A SYSTEM INFORMATION FIELD (SIF) ACTS LIKE AN OUTPUT LITERAL FIELD IN THAT IT IS PROCESSED ON OUTPUT OPERATIONS ONLY AND IT MAPS DATA INTO A SELECTED DEVICE FORMAT FIELD. THE DATA MAPPED, HOWEVER, IS NOT A LITERAL TEXT STRING BUT RATHER A SYSTEM RELATED PIECE OF INFORMATION, SUCH AS CURRENT TIME, DATE, USER NAME AND NUMBER, OR FORM NAME.

  SIF NAMES, CONTENTS, FORMAT, AND LENGTH ARE DESCRIBED BELOW.

```
DATE1:      DATE, YY/MM/DD              ;8 CHARACTERS=
DATE2:      DATE, DD-MMM-YY             ;9      "     =
DATE3:      DATE, MM/DD/YY              ;8      "     =
DATE4:      DATE, DD.MM.YY              ;8      "     =
TIME1:      TIME, HH:MM                 ;5      "     =
TIME2:      TIME, HH:MM XM              ;8      "     =
USERNAME:   USER LOGIN NAME, XXXXXX     ;6      "     =
USERNUM:    USER NUMBER, NN             ;2      "     =
FORMNAME:   FORM NAME, XXXXXXXX         ;8      "     =
```

  FDL SPECIFICATION:

```
FIELD (DF-FIELD-NAME,SIF-NAME)
```

EACH FIELD WITH A DIRECT, OUTPUT LITERAL, INPUT EMPTY CONDITIONAL, OR SYSTEM INFORMATION FIELD TYPE IS IDENTIFIED BY A 1-8 CHARACTER NAME WHICH MUST BE UNIQUE WITHIN THIS STREAM DEFINITION. THE USER MAY SUPPLY THIS NAME IN THE LEFT MARGIN IN THE FIELD DEFINITION STATEMENT. IF NOT EXPLICITLY DEFINED, THE FIELD NAME IS ASSUMED TO BE THE NAME OF THE FORMAT FIELD TO WHICH THE STREAM FIELD IS MAPPED. IF THE APPLICATION PROGRAM DESIRES TO MODIFY ANY ATTRIBUTES OF THIS FIELD, THE FIELD NAME IS GIVEN AS AN ARGUMENT TO A FORMS RUN TIME COMMAND (DESCRIBED LATER.)

THE FOLLOWING PARAMETERS ARE POSITION INDEPENDENT. THEY MAY APPEAR IN THE FIELD DEFINITION AFTER THE MAPPING OR LITERAL SPECIFICATION.

6.5.1 LENGTH PARAMETER

THIS PARAMETER DEFINES THE NUMBER OF CHARACTERS CONTAINED IN THE FIELD. THE KEYWORD LENGTH MUST BE FOLLOWED BY A POSITIVE NON-ZERO INTEGER.

USAGE:

FIELD TYPE	REMARKS
DIRECT	REQUIRED
INPUT-LITERAL	OPTIONAL - IF OMITTED, DEFAULTS TO TEXT STRING LENGTH; IF SUPPLIED, TEXT STRING IS PADDED/TRUNCATED AS REQUIRED TO MEET GIVEN LENGTH
OUTPUT-LITERAL	SAME AS INPUT-LITERAL
EMPTY-CONDITIONAL	SAME AS INPUT-LITERAL
FILLER	REQUIRED
SYSTEM-INFO	IGNORED

6.5.2 JUSTIFY PARAMETER

THE JUSTIFY PARAMETER DEFINES THE JUSTIFICATION TO OCCUR WHEN ANY DATA IS LOGICALLY MOVED TO THIS FIELD. IT MUST BE FOLLOWED BY ONE OF THE FOLLOWING KEYWORDS.

- NONE SPECIFIES NO JUSTIFICATION
- LEFT THE FIELD IS LEFT JUSTIFIED, RIGHT PADDED
- RIGHT THE FIELD IS RIGHT JUSTIFIED, LEFT PADDED
- CENTER THE FIELD IS CENTERED

NOTE THAT 'JUSTIFY NONE' HAS THE SAME EFFECT AS NOT SPECIFYING THE JUSTIFY PARAMETER AT ALL.

IF JUSTIFICATION IS SPECIFIED ON BOTH THE STREAM DESCRIPTOR AND FORMAT DESCRIPTOR FIELDS, THE DATA IS JUSTIFIED AS PER THE STREAM DESCRIPTOR FIELD SPECIFICATION ON INPUT AND AS PER THE FORMAT DESCRIPTOR FIELD ON OUTPUT.

USAGE:

FIELD TYPE	REMARKS
DIRECT	OPTIONAL
INPUT-LITERAL	OPTIONAL
OUTPUT-LITERAL	OPTIONAL
EMPTY-CONDITIONAL	OPTIONAL
FILLER	IGNORED
SYSTEM-INFO	IGNORED

6.5.3 SPACE-FILL AND ZERO-FILL PARAMETERS

THESE MUTUALLY EXCLUSIVE PARAMETERS DEFINE THE FILL CHARACTER TO BE USED WHEN PERFORMING LEFT OR RIGHT JUSTIFICATION. FOR EACH CHARACTER POSITION THE DATA IS SHIFTED, EITHER A SPACE OR ZERO IS SUPPLIED ON THE END FROM WHICH THE SHIFT IS TAKING PLACE. NOTE, HOWEVER, THAT IF THE OPERATOR ENTERS THE FIELD SUCH THAT THE DATA IS ALREADY RIGHT JUSTIFIED AND RIGHT JUSTIFICATION WITH ZERO-FILL IS SPECIFIED IN THE FORM DEFINITION, LEFT-MOST SPACES (IF ANY) WILL NOT BE REPLACED WITH ZEROES. IF NEITHER PARAMETER IS SPECIFIED, SPACE-FILL IS ASSUMED.

USAGE:

APPLIES ONLY TO FIELDS WITH LEFT OR RIGHT JUSTIFICATION. SEE JUSTIFY USAGE, ABOVE.

6.5.4 INPUT, OUTPUT, AND INPUT-OUTPUT PARAMETERS

THESE PARAMETERS, WHICH FORM A MUTUALLY EXCLUSIVE TRIO, DEFINE THE DIRECTION OF DATA TRANSFER IN WHICH THIS FIELD SHOULD BE PROCESSED AS OUTLINED IN THE INTRODUCTION TO STREAM DESCRIPTOR FIELDS, EARLIER IN THIS SECTION.

USAGE:

FIELD TYPE	REMARKS
DIRECT	OPTIONAL; DEFAULT IS INPUT-OUTPUT
INPUT-LITERAL	DEFAULT TO INPUT, IF SPECIFIED, MUST BE INPUT
OUTPUT-LITERAL	MUST BE SPECIFIED AS OUTPUT
EMPTY-CONDITIONAL	MUST BE SPECIFIED AS INPUT OR INPUT-OUTPUT
FILLER	OPTIONAL; DEFAULT IS INPUT-OUTPUT
SYSTEM-INFO	IGNORED

6.5.5 VALIDATE PARAMETER

THIS PARAMETER DEFINES THE VALIDATION TO TAKE PLACE ON THE FIELD DATA WHEN INPUT FROM THE DEVICE. THE KEYWORD VALIDATE IS FOLLOWED BY ONE OR MORE VALIDATION MASKS, ENCLOSED WITHIN SINGLE QUOTES AND OPTIONALLY SEPARATED BY THE WORD 'OR'.

WHEN A FIELD WITH A VALIDATION SPECIFICATION IS TRANSFERRED TO THE USER'S INPUT RECORD AT RUN-TIME, THE DATA IS CHECKED AGAINST THE VALIDATION MASK(S) SUPPLIED. IF ONE OF THE VALIDATION TESTS IS PASSED, THE NEXT FIELD IS TRANSFERRED TO THE INPUT RECORD. IF THE DATA FAILS ALL TESTS, FORMS PERFORMS ONE OF TWO ACTIONS SPECIFIED BY THE FIX / NOFIX PARAMETERS DESCRIBED BELOW.

A VALIDATION MASK CONSISTS OF A STRING OF CHARACTERS, EACH DEFINING A CERTAIN CRITERION FOR THE CORRESPONDING CHARACTER IN THE FIELD. IF THE LENGTH OF THE VALIDATION MASK IS LESS THAN THAT OF THE DATA FIELD, THE LAST CHARACTER OF THE VALIDATION MASK IS LOGICALLY REPEATED UNTIL THE DATA FIELD IS EXHAUSTED.

FOLLOWING IS A LIST OF VALIDATION MASK CHARACTERS AND THEIR MEANINGS:

MASK CHARACTER	VALIDATION CRITERIA
9	NUMERIC (0-9)
A	ALPHABETIC (A-Z, a-z)
X	ALPHANUMERIC (0-9, A-Z, a-z)
.	PERIOD
/	FORE-SLASH
B	SPACE (BLANK)
$	DOLLAR SIGN
-	DASH
	ANY CHARACTER
N	NUMERIC CHARACTER (0-9, +, -, OR BLANK)
F	FLOATING NUMERIC (0-9, +, -, ., BLANK)
U	UNSIGNED INTEGER (0-9, BLANK)
P	PERSONAL NAME (A-Z, a-z, ., ', OR BLANK)
Z	ALPHABETIC CHARACTER OR SPACE

USAGE:

FIELD TYPE	REMARKS
DIRECT	OPTIONAL
INPUT-LITERAL	IGNORED
OUTPUT-LITERAL	IGNORED
EMPTY-CONDITIONAL	OPTIONAL
FILLER	IGNORED
SYSTEM-INFO	IGNORED

6.5.6 FIX/NOFIX PARAMETERS

WHEN A FIELD WITH ONE OR MORE VALIDATION MASKS FAILS TO MEET ANY OF THE SPECIFIED VALIDATION CRITERIA, THE PROGRAMMER HAS THE OPTION OF FORCING THE OPERATOR TO CORRECT THE DATA BEFORE FORMS RETURNS IT TO THE APPLICATIONS PROGRAM.

IF FIX IS SPECIFIED, THE DATA MUST PASS ONE OR MORE OF THE SUPPLIED VALIDATION TESTS BEFORE IT IS RETURNED TO THE APPLICATIONS PROGRAM. IF THE DATA FAILS ALL VALIDAION TESTS, FORMS PRINTS AN ERROR MESSAGE IN THE LOWER RIGHT CORNER OF THE SCREEN AND POSITIONS THE CURSOR TO THE FIRST CHARACTER POSITION OF THE FIELD IN ERROR. THE OPERATOR MAY THEN CORRECT THE ERROR AND RE-TRANSMIT THE FORM DEFINITION TO THE COMPUTER.

IF THE NOFIX PARAMETER IS SPECIFIED, THE DATA IS RETURNED TO THE PROGRAM WHETHER OR NOT IT PASSES ANY OF THESE VALIDATION TESTS. WHEN THE INPUT RECORD IS COMPLETE, FORMS RETURNS TO THE ERROR RETURN LOCATION INSTEAD OF TAKING THE STANDARD RETURN. THIS MEANS THAT AN ERR= CLAUSE MUST BE PRESENT IN A FORTRAN READ STATEMENT SHOULD ANY FIELDS IN THE FORM DEFINITION CONTAIN A NOFIX PARAMETER. COBOL PROGRAM MUST CONTAIN A DECLARATIVE PARAGRAPH TO DO "ERROR" PROCESSING. A VALIDATION ERROR MAY BE IDENTIFIED BY EITHER A FORTRAN OR COBOL PROGRAM BY INSPECTING THE TWO-CHARACTER ERROR CODE IN THE USER'S ERROR VECTOR BY CALLING THE GETERR SYSTEM SUBROUTINE. FORMS WILL SET THIS CODE TO 'VA' FOR VALIDATION ERRORS.

IN MOST CASES, IT IS MUCH MORE CONVENIENT TO REQUIRE THE DATA IN THE PROPER FORMAT WHEN IT REACHES THE APPLICATIONS PROGRAM (I.E. USING THE FIX PARAMETER) THUS ELIMINATING THE TASK OF INSPECTING MULTIPLE FIELDS ON A CHARACTER-BY-CHARACTER BASIS, WHICH MAY BE UNNATURAL OR IMPOSSIBLE IN THE HOST LANGUAGE.

IF FIX OR NOFIX IS NOT SPECIFIED, FIX IS ASSUMED.

USAGE:

FIELD TYPE	REMARKS
DIRECT	OPTIONAL
INPUT-LITERAL	IGNORED
OUTPUT-LITERAL	IGNORED
EMPTY-CONDITIONAL	OPTIONAL
FILLER	IGNORED
SYSTEM-INFO	IGNORED

6.5.7 START PARAMETER

THE START PARAMETER ALLOWS THE USER TO SPECIFY THE CHARACTER POSITION OCCUPIED BY THE FIRST CHARACTER OF THIS DATA FIELD WITHIN THE USER'S INPUT OR OUTPUT RECORD. IT 'S FUNCTION IS EQUIVALENT TO THAT OF THE 'T' FORMAT DESCRIPTOR IN A FORTRAN FORMAT STATEMENT. START ALLOWS OVERLAPPING OF INPUT/OUTPUT FIELDS, A FUNCTION NOT AVAILABLE WITH THE 'FILLER' MAP SPECIFICATION.

THE WORD START MUST BE FOLLOWED BY AN INTEGER NUMBER, WHICH REPRESENTS THE ABSOLUTE CHARACTER POSITION (WITHIN THE USER'S DATA RECORD) OF THE FIRST CHARACTER OF THIS FIELD.

WARNING: IF START IS SPECIFIED IN AN INPUT-ONLY FIELD, THE CHARACTER POINTER GETS RESET FOR THE INPUT RECORD BUT NOT FOR THE OUTPUT RECORD. THE INVERSE IS TRUE FOR OUTPUT-ONLY FIELDS. THIS IS REFLECTED IN THE INPUT AND OUTPUT STREAM DESCRIPTOR FORMATS GENERATED BY FDL IF THE -IOFLIST OPTION IS SPECIFIED.

USAGE:

FIELD_TYPE	REMARKS
DIRECT	OPTIONAL
INPUT-LITERAL	OPTIONAL
OUTPUT-LITERAL	OPTIONAL
EMPTY-CONDITIONAL	OPTIONAL
FILLER	OPTIONAL
SYSTEM-INFO	IGNORED

EXAMPLES:

```
*   DIRECT FIELD TYPE.
        FIELD IDNUM, LENGTH 5

*   LITERAL INPUT FIELD, RETURN STARTING IN COLUMN 30
        FIELD 'LITERAL INPUT STRING', START 30

*   OUTPUT LITERAL FIELD
        FIELD (HEADER,'HEADER TEXT'), OUTPUT

*   INPUT EMPTY CONDITIONAL FIELD TYPE
        FIELD (EMPLNAME,'NO EMPLOYEE NAME SPECIFIED'), ;
              INPUT-OUTPUT

*   FILLER FIELD
        FIELD FILLER, LENGTH 12

*   SYSTEM INFORMATION FIELD
        FIELD (OUTDATE,DATE3)

*   INPUT-ONLY FIELD, JUSTIFY AND VALIDATE
        FIELD AGE, LENGTH 3, JUSTIFY RIGHT, INPUT, ;
              VALIDATE '999' OR 'B'
```

6.6 DEVICE FORMAT DESCRIPTOR FIELD STATEMENT

FIELDS DEFINED WITHIN THE DEVICE FORMAT DESCRIPTOR DESCRIBE THE APPEARANCE OF DATA ON THE PAGE ORIENTED DEVICE. THIS DEFINITION INCLUDES FIELD COORDINATES, LENGTH, JUSTIFICATION, AND ANY DISPLAY ATTRIBUTES TO BE ASSOCIATED WITH THE DATA (WRITE ENABLE / PROTECT, BLINK, REVERSE VIDEO, ETC.)

THERE ARE TWO TYPES OF DEVICE FORMAT DESCRIPTOR FIELD STATEMENTS:

- MAPPED: A "MAPPED" FIELD IS ACTUALLY MAPPED TO BY A FIELD DEFINED IN THE DATA STREAM DESCRIPTOR. ALL MAPPED FIELDS CONTAIN A 1-8 CHARACTER NAME STARTING IN THE LEFT MARGIN IN THE FIELD DEFINITION STATEMENT.

ANY MAPPED FIELDS DEFINED IN THE FORMAT DESCRIPTOR AND NOT MAPPED TO BY A STREAM FIELD ARE IGNORED. ANY STREAM DESCRIPTOR FIELD WHICH MAPS TO A NON-EXISTANT FORMAT DESCRIPTOR FIELD IS ALSO IGNORED.

0 LITERAL: A LITERAL FIELD CONTAINS A TEXT STRING SPECIFIED IN THE FORMAT DESCRIPTOR FIELD DEFINITION. LITERAL FIELDS ARE USED TO SUPPLY TAGS (TITLES) FOR INFORMATION DISPLAYED ON THE DEVICE AND USUALLY USED TO IDENTIFY MAPPED FIELDS.

THE LITERAL DATA IS SPECIFIED IMMEDIATELY FOLLOWING THE FIELD STATEMENT AND MUST BE ENCLOSED WITHIN SINGLE QUOTES. THE NAME FIELD (LEFT MARGIN) MUST BE BLANK (NO MAPPING IS DONE FROM A STREAM DESCRIPTOR FIELD.)

THE FOLLOWING PARAMETERS MAY FOLLOW THE FIELD STATEMENT IN A MAPPED FIELD AND THE LITERAL SPECIFICATION IN A LITERAL FIELD. THEY ARE ALL NON-POSITIONAL, I.E. THEY MAY OCCUR ANYWHERE IN THE FIELD DEFINITION. NOTE THAT ALL PARAMETERS APPLY TO BOTH THE MAPPED AND LITERAL DEVICE DESCRIPTOR FIELD TYPES. ALL ARE OPTIONAL UNLESS OTHERWISE NOTED.

6.6.1 LENGTH PARAMETER

THIS PARAMETER DEFINES THE LENGTH OF THE FIELD AS IT IS TO APPEAR OF THE DEVICE. IT MUST BE FOLLOWED BY A POSITIVE NON-ZERO INTEGER, WHICH REPRESENTS THE FIELD LENGTH IN CHARACTERS. THIS PARAMETER IS REQUIRED ON MAPPED FIELDS AND IS OPTIONAL ON LITERAL FIELDS. IF OMITTED, THE FIELD LENGTH IS ASSUMED TO BE THE LENGTH OF THE LITERAL STRING.

NOTE THAT THE LENGTH OF A FIELD IN THE STREAM DESCRIPTOR MAY DIFFER FROM THE LENGTH OF A FIELD IN THE DEVICE FORMAT DESCRIPTOR. RECALL THAT THE STREAM FIELD DEFINES THE LENGTH IN THE INPUT/OUTPUT RECORD OF THE APPLICATIONS PROGRAM AND THE DEVICE FORMAT FIELD LENGTH DEFINES THE LENGTH OF THE FIELD ON THE INPUT/OUTPUT DEVICE. IF THEY DIFFER, THE DATA IS TRUNCATED OR PADDED AS REQUIRED.

6.6.2 POSITION PARAMETER

THE POSITION PARAMETER DEFINES THE POSITION (COLUMN AND LINE) OF THE FIRST CHARACTER IN THE FIELD. THE KEYWORD IS FOLLOWED BY THE COLUMN AND LINE (X,Y) ADDRESS, ENCLOSED WITHIN PARENTHESIS AND SEPERATED BY A COMMA. THIS PARAMETER IS MANDATORY ON BOTH MAPPED AND LITERAL FIELDS.

6.6.3 JUSTIFY PARAMETER

THE JUSTIFY PARAMETER DEFINES THE JUSTIFICATION TO TAKE PLACE WHEN DATA IS LOGICALLY MOVED TO (THROUGH) THIS FIELD. REFER TO THE DESCRIPTION OF THE JUSTIFY PARAMETER IN THE STREAM DESCRIPTOR FIELD DESCRIPTION FOR INFORMATION ON IT'S USE. THIS PARAMETER IS OPTIONAL ON BOTH MAPPED AND LITERAL FIELDS AND IS DEFAULTED TO JUSTIFY NONE IF NOT SPECIFIED.

6.6.4 ATTRIBUTE PARAMETERS

THE FOLLOWING EIGHT PARAMETERS ARE USED TO DESCRIBE THE DISPLAY CHARACTERISTICS OF THE FIELD DATA WHEN IT IS OUTPUT TO THE DEVICE. IF A DEVICE DOES NOT SUPPORT A CERTAIN FEATURE, SUCH AS REVERSE VIDEO OR BLINK, THE ATTRIBUTE IS IGNORED. NOTE THAT A WORD IN SQUARE BRACKETS FOLLOWING THE ATTRIBUTE NAME MEANS THAT IT IS SYNONYMOUS WITH THE PRECEDING ATTRIBUTE (EG., ENABLE IS SYNONYMOUS WITH NOPROTECT).

6.6.4.1 NOPROTECT [ENABLE] PARAMETER

THIS PARAMETER, WHICH IS MUTUALLY EXCLUSIVE WITH PROTECT, DECLARES THIS FIELD TO BE WRITE-ENABLED UPON DISPLAY TO THE USER-TERMINAL. WHEN DISPLAYED ON THE LINE-PRINTER THE FIELD IS UNDERLINED (IF UNDERLINING IS AVAILABLE).

6.6.4.2 PROTECT PARAMETER

THIS PARAMETER DECLARES THAT THIS FIELD IS TO BE DISPLAYED WRITE-PROTECTED WHEN OUTPUT TO THE USER TERMINAL. WHEN OUTPUT TO THE LINE-PRINTER, IT IS NOT UNDERLINED (DISPLAYED NORMALLY). IF NEITHER PROTECT NOR NOPROTECT IS SPECIFIED, PROTECT IS ASSUMED.

6.6.4.3 BLINK PARAMETER

THIS PARAMETER DEFINES THIS FIELD TO BE BLINKED WHEN DISPLAYED ON THE TERMINAL. IT HAS NO EFFECT IN A DEVICE DESCRIPTOR FOR THE PRINTER.

6.6.4.4 NOBLINK PARAMETER

THIS PARAMETER DEFINES THIS FIELD AS NOT BLINKED WHEN DISPLAYED ON THE USER-TERMINAL. IF BOTH BLINK AND NOBLINK ARE OMITTED, THE DEFAULT IS NOBLINK.

6.6.4.5 REVERSE VIDEO PARAMETER

THIS PARAMETER CAUSES THE FIELD TO BE DISPLAYED IN REVERSE VIDEO WHEN OUTPUT TO THE USER-TERMINAL. IT HAS NO EFFECT WHEN OUTPUT IS TO THE LINE-PRINTER.

6.6.4.6 NORMAL VIDEO PARAMETER

THIS PARAMETER DECLARES THE FIELD TO BE DISPLAYED IN NORMAL VIDEO WHEN OUTPUT. IF BOTH THE REVERSE VIDEO AND NORMAL VIDEO PARAMETERS ARE OMITTED, THE DEFAULT IS NORMAL VIDEO.

6.6.4.7 NODISPLAY [HOLD] PARAMETER

THIS PARAMETER CAUSES THIS FIELD NOT TO BE DISPLAYED WHEN THE FORM IS OUTPUT. IT IS VALID ON ALL TERMINAL AND LINE-PRINTER DEVICE TYPES.

6.6.4.8 DISPLAY [FREE] PARAMETER

THIS ATTRIBUTE CAUSES THIS FIELD TO BE DISPLAYED WHEN THE FORM IS OUTPUT TO EITHER THE TERMINAL OR THE LINE-PRINTER. IF BOTH THE DISPLAY AND NODISPLAY PARAMETERS ARE OMITTED, THE DEFAULT IS DISPLAY.

EXAMPLES:

```
*   MAPPED FIELD, NOT WRITE-PROTECTED
INVNUM    FIELD  POSITION (70,2), LENGTH 6, NOPROTECT

*   LITERAL FIELD
          FIELD 'LITERAL STRING TEST', POSITION (1,4) ;
                REVERSE VIDEO
```

6.7 PROGRAMMING AIDS

FOLLOWING IS A DESCRIPTION OF STATEMENTS DESIGNED TO ASSIST THE PROGRAMMER DESIGNING A FORM. THEY INCLUDE A MACRO CAPABILITY AND ITERATIVE FIELD GENERATION.

6.7.1 THE DEFINE [DEF] STATEMENT

THIS STATEMENT ALLOWS THE PROGRAMMER TO DEFINE A MACRO. CURRENTLY, A MACRO CONSISTS SIMPLY OF ONE TEXT ITEM REPLACING ANOTHER ITEM OR TEXT STRING (I.E., A SYNONYM). FUTURE PLANS CALL FOR IMPLEMENTATION OF MACRO ARGUMENTS.

A DEFINE (OR DEF) STATEMENT MUST BE PRECEDED BY THE NAME OF THE MACRO, STARTING IN THE LEFT MARGIN. THE STATEMENT NAME MUST BE FOLLOWED BY ONE OR MORE SPACES, AND THEN BY THE MACRO TEXT.

WHENEVER THE MACRO NAME IS ENCOUNTERED AS A SINGLE ITEM WITHIN AN INPUT LINE (NOT IN A LITERAL TEXT STRING), THE MACRO NAME IS REPLACED BY THE GIVEN DEFINITION. ALL MACROS MUST BE DEFINED BEFORE THEY ARE USED.

MACRO DEFINITIONS ARE NOT RETAINED BETWEEN FORM DEFINITIONS, I.E., THEY ARE 'ERASED' AFTER EACH END STREAM AND END FORMAT STATEMENT. THEY ARE, HOWEVER, RETAINED ACROSS END DEVICE / DEVICE STATEMENTS IN FORMAT DESCRIPTORS.

EXAMPLES:

```
FLD         DEFINE  FIELD
LEN         DEFINE  LENGTH
POS         DEFINE  POSITION
D1X         DEFINE  5
D1Y         DEFINE  10
*
*
*   FIELD DEFINITION USING ABOVE MACRO DEFINITIONS
DATA1       FLD, POS (D1X,D1Y), LEN 10
*
*   NOTE THAT THIS IS HAS THE SAME FUNCTION AS:
DATA1       FIELD, POSITION (5,10), LENGTH 10
```

6.7.2 ITERATIVE FIELD GENERATION

THIS FEATURE OF FDL ALLOWS THE PROGRAMMER TO GENERATE MULTIPLE BLOCKS OF FIELD STATEMENTS WITH ONLY ONE BLOCK DEFINITION. FIELDS TO BE GENERATED IN THIS MANNER MUST BE ENCLOSED WITHIN REPEAT AND END REPEAT STATEMENTS (SEE BELOW).

ITERATIVE FIELD GENERATION IS PERMITTED IN BOTH STREAM DESCRIPTOR AND DEVICE FORMAT DESCRIPTOR DEFINITIONS. IN BOTH, A 2-DIGIT ITERATION NUMBER IS APPENDED TO ANY FIELD NAME FOUND IN EITHER THE LEFT MARGIN OR IMMEDIATELY FOLLOWING A STREAM FIELD STATEMENT. IF THE FIELD NAME IS SEVEN OR EIGHT CHARACTERS, IT IS TRUNCATED TO 6 CHARACTERS TO PERMIT THE ITERATION NUMBER TO BE APPENDED. THE SAME IS TRUE FOR DEVICE FORMAT ("MAPPED TO") FIELD NAMES ENCOUNTERED IN DIRECT, OUTPUT-LITERAL, AND INPUT/EMPTY-CONDITIONAL STREAM DESCRIPTOR FIELDS.

6.7.2.1 THE REPEAT STATEMENT

THIS STATEMENT DEFINES THE BEGINNING OF AN ITERATIVE FIELD GENERATION (REPEAT) BLOCK. IT MUST BE FOLLOWED BY AN INTEGER NUMBER, GREATER THAN ZERO, WHICH REPRESENTS THE NUMBER OF ITERATIONS TO MAKE THRU THE FOLLOWING FIELD DEFINITIONS. THE ITERATION COUNTER IS INITIALLY SET TO ONE AND IS INCREMENTED BY ONE EACH PASS THROUGH THE REPEAT BLOCK. WHEN THE COUNTER EXCEEDS THE SPECIFIED REPEAT COUNT, THE STATEMENT IMMEDIATELY FOLLOWING THE END REPEAT (SEE BELOW) IS PROCESSED.

ONLY FIELD STATEMENTS ARE PERMITTED WITHIN A REPEAT BLOCK.

6.7.2.2 THE END REPEAT STATEMENT

THIS STATEMENT TERMINATES A REPEAT BLOCK. FOR EACH REPEAT STATEMENT, THERE MUST BE A CORRESPONDING END REPEAT STATEMENT. REPEAT BLOCKS MAY NOT BE NESTED.

6.7.2.3 RELATIVE POSITION PARAMETER SPECIFICATION

A SECOND FORM OF THE POSITION PARAMETER IS AVAILABLE TO FIELDS DEFINED WITHIN A REPEAT BLOCK. THIS PERMITS THE FIELD COORDINATES TO BE RELATIVE TO THE CURRENT ITERATION NUMBER INSTEAD OF ABSOULTE LINE AND COLUMN.

RELATIVE POSITIONING IS SPECIFIED BY PLACING A PLUS OR MINUS SIGN IMMEDIATELY PRECEDING THE LINE AND/OR COLUMN DEFINITION IN THE POSITION PARAMETER. THE ABSOLUTE LINE OR COLUMN NUMBER IS COMPUTED BY ADDING OR SUBTRACTING THE CURRENT ITERATION NUMBER TO OR FROM THE SPECIFIED OFFSET.

EXAMPLE:

```
*
*   THIS BLOCK WILL BE REPEATED 3 TIMES
*
            REPEAT 3
LASTNM      FIELD  LENGTH 20, POSITION (10,+7)
FRSTNM      FIELD  LENGTH 10, POSITION (35,+7)
MIDDIN      FIELD  LENGTH 1,  POSITION (50,+7)
            END REPEAT
```

```
*
*
*   NOTE THAT THIS HAS THE SAME FUNCTION AS:
*
LASTNM01   FIELD  LENGTH 20, POSITION (10,8)
FRSTNM01   FIELD  LENGTH 10, POSITION (35,8)
MIDDIN01   FIELD  LENGTH 1,  POSITION (50,8)
LASTNM02   FIELD  LENGTH 20, POSITION (10,9)
FRSTNM02   FIELD  LENGTH 10, POSITION (35,9)
MIDDIN02   FIELD  LENGTH 1,  POSITION (50,9)
LASTNM03   FIELD  LENGTH 20, POSITION (10,10)
FRSTNM03   FIELD  LENGTH 10, POSITION (35,10)
MIDDIN03   FIELD  LENGTH 1,  POSITION (50,10)
```

6.8 LISTING CONTROL STATEMENTS

6.8.1 THE NOLIST STATEMENT

THIS STATEMENT DISABLES THE LISTING OF ALL FDL STATEMENTS, MACRO AND REPEAT BLOCK EXPANSIONS, EXCEPT FOR THOSE CONTAINING ERRORS. IT IS OVERRIDDEN ONLY BY THE 'FULL LIST' COMMAND LINE OPTION.

6.8.2 THE LIST STATEMENT

THIS STATEMENT ENABLES THE LISTING OF FDL STATEMENTS, MACRO AND REPEAT BLOCK EXPANSIONS AFTER BEING DISABLED BY THE NOLIST STATEMENT. IT IS OVERRIDDEN BY THE 'ERRORS ONLY' COMMAND LINE OPTION.

6.8.3 THE EJECT STATEMENT

THIS STATEMENT CAUSES THE LISTING TO EJECT TO THE TOP OF A NEW PAGE WHEN THE LISTING FILE IS OUTPUT (SPOOLED) TO THE LINE-PRINTER. THE OLD PAGE HEADER IS RETAINED. FOR A NEW PAGE HEADER, REFER TO THE SECTION ENTITLED 'GENERAL SYNTAX'. NOTE THAT THIS STATEMENT HAS NO EFFECT IF THE LISTING IS TURNED OFF (VIA THE 'ERRORS ONLY' OPTION OR 'NOLIST' STATEMENT).

6.9 ALTERNATE INPUT FILE ($INSERT) DIRECTIVE

A METHOD EXISTS WHEREBY THE PROGRAMMER CAN 'INSERT' THE CONTENTS OF ANOTHER FDL SOURCE FILE INTO HIS PRIMARY INPUT FILE AT TRANSLATION-TIME. THIS IS ACCOMPLISHED BY PLACING THE '$INSERT' DIRECTIVE IN THE LEFT MARGIN OF THE INPUT LINE, FOLLOWED BY AT LEAST ONE SPACE, AND THEN THE TREE-NAME OF THE DISK FILE TO BE INSERTED. INPUT IS THEN OBTAINED FROM THE INSERT (ALTERNATE) DISK FILE UNTIL THE END OF FILE IS ENCOUNTERED. WHEN EOF IS REACHED, FDL RESUMES

PROCESSING THE PRIMARY INPUT FILE AT THE LINE FOLLOWING THE $INSERT DIRECTIVE. NOTE THAT NO ACTUAL MODIFICATION OF THE MAIN INPUT FILE IS DONE; THIS TEMPORARILY "SWITCHES" THE INPUT FLOW FROM THE PRIMARY TO THE ALTERNATE INPUT FILE.

THE $INSERT DIRECTIVE PROVIDES A CONVENIENT METHOD OF INCORPORATING A COMMON MACRO DEFINITION FILE INTO AN FDL SOURCE FILE.

EXAMPLE:

```
$INSERT <SOFTWR> FORMS> MACROS
```

6.10 USING FDL

FDL IS INVOKED BY ENTERING THE EXTERNAL COMMAND "FDL" FOLLOWING THE "OK, " PROMPT ISSUED BY PRIMOS. THE COMMAND MAY BE FOLLOWED BY AN INPUT FILE NAME AND / OR A LIST OF TRANSLATION OPTIONS. ALTHOUGH COMMAND LINE FORMATS HAVE BEEN STANDARDIZED FOR ALL PRIME TRANSLATORS, A COMPLETE LIST OF GENERAL COMMAND LINE OPTIONS FOLLOWS:

OPTION	DEFINITION
TNAME	INPUT (SOURCE) TEXT IS TO BE OBTAINED FROM DISK FILE SPECIFIED BY TREE-NAME TNAME - THIS MAY ONLY APPEAR IMMEDIATELY FOLLOWING THE COMMAND NAME
-INPUT TNAME	SAME AS ABOVE BUT MAY APPEAR ANYWHERE ON THE COMMAND LINE
-INPUT TTY	SOURCE TEXT IN TO BE OBTAINED FROM THE USER TERMINAL
-LISTING	LISTING FILE IS TO BE GENERATED
-LISTING TNAME	LISTING IS TO BE WRITTEN TO DISK FILE SPECIFIED BY TREE-NAME TNAME
-LISTING NO	NO LISTING FILE IS TO BE PRODUCED
-LISTING TTY	LISTING IS TO BE PRINTED ON USER TERMINAL
-LISTING SPOOL	LISTING FILE IS TO BE ROUTED DIRECTLY TO SPOOL QUEUE - THE NAME OF THE SPOOL FILE IS PRINTED ON THE USER TERMINAL PRIOR TO START OF TRANSLATION
-BINARY	BINARY FILE IS TO BE GENERATED
-BINARY TNAME	BINARY FILE IS TO BE GENERATED - NAME IS SPECIFIED BY TREE-NAME TNAME
-BINARY NO	NO BINARY FILE IS TO BE GENERATED

IF A -BINARY OR -LISTING OPTION IS NOT FOLLOWED BY A NAME, THE BINARY FILE IS WRITTEN TO EITHER THE FILE OPEN ON FILE UNIT 3 OR TO A FILE CALLED B_INPUTFILENAME IF NO FILE IS OPEN. SIMILARLY, THE LISTING FILE IS WRITTEN TO EITHER THE FILE OPEN ON FILE UNIT 2 OR TO A FILE CALLED L_INPUTFILENAME IF NONE IS OPEN.

FOLLOWING IS A LIST OF FDL-SPECIFIC OPTIONS:

OPTION	DEFINITION
-OBJLIST	LIST EMITTED OBJECT TEXT
-MACLIST	GENERATE EXPANDED MACRO LISTING
-ERRLIST	GENERATE ERRORS-ONLY LISTING
-EXPLIST	OVERRIDE NOLIST PSEUDO-OP
-ERRTERM	LIST ERRORS ON USER TERMINAL
-IOFLIST	LIST I/O AND DEVICE FORMATS
-REPLIST	EXPAND & LIST REPEAT BLOCKS

EACH OPTION EXCEPT EXPLIST MAY BE PRECEDED BY A 'NO' TO REVERSE THE OPTION'S MEANING, FOR EXAMPLE '-NOMACLIST' SPECIFIES THAT EXPANDED MACRO LISTING IS TO BE SUPPRESSED. AN ARGUMENT MAY BE ABBREVIATED TO THE MINIMUM NUMBER OF CHARACTERS REQUIRED TO DISTINGUISH IT FROM OTHER ARGUMENTS.

A TYPICAL FDL COMMAND LINE MIGHT LOOK LIKE:

```
OK, FDL FDEF15, -LISTING SPOOL, -BINARY NO, -OBJ, -MAC
```

EACH INSTALLATION MAY CHOOSE A SET OF DEFAULT OPTIONS FOR THE FDL TRANSLATOR. AS RELEASE BY PRIME, THE FOLLOWING OPTIONS ARE STANDARD:

```
-LISTING   -BINARY   -IOFLIST   -ERRTERM
```

ALL OTHER OPTIONS ARE DISABLED. FDL DEFAULTS ARE SET BY THE A-REGISTER SETTING IN THE TRANSLATOR'S MEMORY-IMAGE FILE. THE USER MAY SELECT HIS OWN DEFAULT OPTIONS BY RESTORE'ING A COPY OF FDL AND SAVE'ING IT WITH THE DESIRED BITS SET PROPERLY IN THE A-REGISTER.

THE FOLLOWING TABLES SHOW THE A-REGISTER BIT SETTINGS FOR FDL OPTIONS AND DEVICE CODES:

OPTIONS		DEVICE CODES
BIT	SET FOR...	
		0 > NONE
1	-OBJLIST	1 > TERMINAL
2	-MACLIST	2 > PAPER TAPE
3	-ERRLIST	3 > CARD READER
4	-EXPLIST	4 > PRINTER
5	-ERRTERM	5 > MAGTAPE
6	-IOFLIST	6 > UNDEFINED
7	-REPLIST	7 > DISK FILE
8-10	INPUT DEVICE	
11-13	LISTING DEVICE	
14-16	BINARY DEVICE	

THE DEFAULT A-REGISTER SETTING IS '6777.

AFTER EACH STREAM OR FORMAT DESCRIPTOR IS TRANSLATED, FDL PRINTS A MESSAGE AT THE USER TERMINAL CONTAINING THE NUMBER OF ERRORS ENCOUNTERED IN THE SOURCE TEXT AND THE FDL REVISION NUMBER.

6.11 FDL ERROR MESSAGES

ALL ERRORS GENERATED BY THE FDL TRANSLATOR ARE OF THE FORM:

```
C#NN    TEXT MESSAGE
```

WHERE 'NN' REPRESENTS A UNIQUE TWO-DIGIT ERROR CODE FOR EACH TYPE OF ERROR. THE MESSAGE PRINTED IS A ONE-LINE DIAGNOSTIC OF THE CAUSE OF THE ERROR AND POSSIBLY WHAT ACTION HAS BEEN TAKEN BY THE TRANSLATOR. FOLLOWING IS A TABLE WHICH ELABORATES ON THE ERROR CODES GENERATED BY FDL. UNLESS OTHERWISE INDICATED, THE OCCURRENCE OF AN ERROR DICTATES THAT THE STATEMENT HAS BEEN IGNORED BY FDL.

C#00 BAD STATEMENT FORMAT.
THE CONTENTS OF THE STATEMENT FIELD IS NOT AN ALPHANUMERIC TEXT ITEM.

C#01 STATEMENT NOT RECOGNIZED.
THE STATEMENT FIELD DOES NOT CONTAIN A VALID FDL STATEMENT.

C#02 ARGUMENT REQUIRED.
AN ARGUMENT IS REQUIRED FOLLOWING THE STATEMENT NAME.

C#03 ARGUMENT TOO LONG.
A TEXT ITEM EXCEEDS 80 CHARACTERS IN LENGTH.

C#04 MULTIPLY DEFINED MACRO.
A MACRO BY THE SAME NAME ALREADY EXISTS. THIS STATEMENT IS IGNORED AND THE PREVIOUS MACRO DEFINITION IS RETAINED.

C#05 BAD NAME FIELD.
THE NAME FIELD (STARTING IN THE LEFT MARGIN) CONTAINS AN ILLEGAL CHARACTER.

C#06 NAME REQUIRED.
A NAME MUST BE PRESENT IN THE NAME FIELD (STARTING IN THE LEFT MARGIN). THIS ERROR IS GENERALLY ISSUED BECAUSE A MAPPED FIELD IN THE DEVICE FORMAT DESCRIPTOR IS MISSING A NAME.

C#07 STATEMENT FIELD IS BLANK.
A NAME WAS PRESENT IN THE NAME FIELD, BUT NO STATEMENT FOLLOWED.

C#08 NO END STATEMENT; END ASSUMED.
AN END-OF-FILE WAS ENCOUNTERED WHILE PROCESSING A STREAM OR FORMAT DESCRIPTOR. AN END STREAM OR END FORMAT IS ASSUMED HERE.

C#09 NOT PROCESSING STREAM DESCRIPTOR.
AN END STREAM OR SUBSTREAM STATEMENT WAS ISSUED AND A STREAM DESCRIPTOR IS NOT BEING PROCESSED.

C#10 END SUBSTREAM MISSING. IT IS ASSUMED HERE.
AN END STREAM STATEMENT WAS ISSUED WHILE A SUBSTREAM BLOCK WAS BEING PROCESSED. AN END SUBSTREAM IS ASSUMED PRIOR TO THE END STREAM.

C#11 NOT PROCESSING SUBSTREAM
AN END SUBSTREAM STATEMENT WAS ISSUED WHILE NOT PROCESSING A SUBSTREAM BLOCK.

C#12 NOT PROCESSING FORMAT
AN END FORMAT OR DEVICE STATEMENT WAS ISSUED WHILE NOT PROCESSING A FORMAT DESCRIPTOR.

C#13 END DEVICE MISSING. IT IS ASSUMED HERE.
AN END FORMAT WAS ENCOUNTERED WHILE STILL PROCESSING A DEVICE DESCRIPTION. AN END DEVICE IS GENERATED PRIOR TO THE END FORMAT.

C#14 NOT PROCESSING DEVICE BLOCK.
A FIELD DEFINITION WAS ISSUED AFTER A FORMAT STATEMENT, BUT BEFORE A DEVICE BLOCK WAS STARTED.

C#15 END STATEMENT MISSING; IT IS ASSUMED HERE.
A STREAM OR FORMAT DESCRIPTOR WAS NOT TERMINATED BEFORE ANOTHER WAS STARTED. AN END STREAM OR END FORMAT IS GENERATED PRIOR TO THIS STATEMENT.

C#17 BAD PARAMETER.
THIS INDICATES THAT AN UNRECOGNIZABLE PARAMETER WAS PRESENT ON A FIELD STATEMENT.

C#18 INVALID FORMAT NAME.
THE NAME SUPPLIED FOLLOWING THE FORMAT PARAMETER IN THE STREAM STATEMENT DOES NOT CONFORM TO THE NAMING CONVENTIONS DISCUSSED EARLIER IN THIS DOCUMENT.

C#19 NAME NOT PERMITTED.
A NAME WAS PRESENT ON A STATEMENT WHICH DOES NOT PERMIT ONE. THIS USUALLY MEANS THAT A LITERAL FIELD IN THE DEVICE FORMAT DESCRIPTOR CONTAINS A NAME.

C#21 ALREADY PROCESSING SUBSTREAM.
A SUBSTREAM STATEMENT WAS ISSUED WHILE ALREADY PROCESSING A SUBSTREAM BLOCK.

C#22 VALIDATION STRING MISSING.
THE VALIDATE PARAMETER IS PRESENT ON A STREAM DESCRIPTOR FIELD, BUT IS NOT FOLLOWED BY ANY VALIDATION MASKS.

C#23 BAD JUSTIFY PARAMETER.
THE JUSTIFY PARAMETER IN THE FIELD DESCRIPTOR IS NOT FOLLOWED BY ONE OF IT'S FOUR VALID ARGUMENTS.

C#24 MAPPING SPECIFICATION REQUIRED.
A STREAM FIELD DESCRIPTOR IS NOT FOLLOWED BY A MAPPING SPECIFICATION.

C#25 BAD MAPPING SPECIFICATION.
A STREAM FIELD DESCRIPTOR IS NOT FOLLOWED BY A VALID MAPPING SPECIFICATION.

C#26 BAD LENGTH SPECIFICATION.
THE LENGTH PARAMETER IN EITHER STREAM OR DEVICE DESCRIPTOR IS NOT FOLLOWED BY A VALID NUMERIC ARGUMENT.

C#27 BAD INPUT-OUTPUT SPECIFICATION.
AN INPUT, OUTPUT, OR INPUT-OUTPUT PARAMETER HAS BEEN MISUSED. THIS USUALLY MEANS THAT INPUT-OUTPUT OR OUTPUT HAS BEEN ISSUED WHEN PROCESSING AN INPUT-LITERAL FIELD.

C#28 MAP FIELD NAME TOO LONG.
THE "MAP TO" FIELD NAME IN A STREAM DESCRIPTOR FIELD IS LONGER THAN 8 CHARACTERS.

C#29 ALREADY PROCESSING DEVICE BLOCK.
A DEVICE STATEMENT HAS BEEN ISSUED WHILE ALREADY PROCESSING A DEVICE BLOCK.

C#30 SYNTAX ERROR.
THIS GENERAL ERROR MESSAGE IS ISSUED WHENEVER TWO ITEMS IN A FIELD DEFINITION ARE SEPERATED BY AN ILLEGAL CHARACTER.

C#31 BAD POSITION PARAMETER.
THE POSITION PARAMETER IN A DEVICE FORMAT DESCRIPTOR FIELD IS NOT FOLLOWED BY A VALID ARGUMENT.

C#32 POSITION OUT OF RANGE.
ONE OR MORE OF THE ARGUMENTS IN THE POSITION PARAMETER IS ZERO.

C#33 LENGTH PARAMETER MISSING.
THE LENGTH DECLARATION FOR A STREAM OR DEVICE FORMAT DESCRIPTOR FIELD IS REQUIRED BUT NOT SUPPLIED.

C#34 POSITION PARAMETER MISSING.
THE POSITION PARAMETER IN A DEVICE DESCRIPTOR FIELD IS NOT SUPPLIED.

C#35 UNRECOGNIZED SYSTEM INFORMATION FIELD NAME.
THE NAME SPECIFIED IN A SYSTEM INFORMATION FIELD IS UNRECOGNIZED.

C#36 INPUT/OUTPUT SPECIFICATION NOT PERMITTED.
AN INPUT, OUTPUT, OR INPUT-OUTPUT SPECIFICATION WAS INCLUDED ON A SYSTEM INFORMATION FIELD DEFINITION.

C#37 UNRECOGNIZED PARAMETER.
SEE C#17.

C#38 NOT PROCESSING STREAM/DEVICE FORMAT BLOCK.
A FIELD DEFINITION HAS BEEN ISSUED OUTSIDE OF A STREAM OR DEVICE FORMAT DESCRIPTOR. THIS AND ALL OTHER FIELD DECLARATIONS UP TO THE NEXT STREAM, FORMAT, OR DEVICE STATEMENT ARE IGNORED. NOTE THAT THIS ERROR MESSAGE IS ISSUED ONLY ONCE PER VIOLATION.

C#39 MULTIPLY DEFINED SYMBOL.
A FIELD NAME HAS BEEN REDEFINED WITHIN THE SAME STREAM OR DEVICE DESCRIPTOR. THIS FIELD IS PROCESSED NORMALLY, BUT WILL PRODUCE UNDESIRED RESULTS AT RUN-TIME.

C#40 BAD START SPECIFICATION.
THE ARGUMENT FOLLOWING THE START SPECIFICATION IN THE STREAM FIELD DEFINITION IS NOT NUMERIC AND GREATER THAN ZERO.

C#41 ILLEGAL MACRO ARGUMENT SPECIFIER.
THE ITEM FOLLOWING THE ARGUMENT REFERENCE SYMBOL (#) IS NOT NUMERIC AND GREATER THAN ZERO. NOTE THAT THIS ERROR CODE SHOULD NOT BE EMITTED BY FDL AT THIS RELEASE.

C#42 EOF ENCOUNTERED BEFORE END REPEAT.
AN END-OF-FILE WAS ENCOUNTERED ON THE INPUT FILE BEFORE A REPEAT BLOCK WAS TERMINATED. THIS USUALLY CAUSES ABORTION OF THE TRANSLATION.

C#43 END REPEAT MISSING - REPEAT BLOCK IGNORED.
AN END STATEMENT WAS ENCOUNTERED WHILE PROCESSING A REPEAT BLOCK. THE ENTIRE REPEAT BLOCK IS IGNORED AND THE END STATEMENT PROCESSED.

C#44 STATEMENT NOT ALLOWED WITHIN REPEAT BLOCK.
A STATEMENT OTHER THAN A FIELD STATEMENT WAS FOUND WITHIN A REPEAT BLOCK. THE STATEMENT IS IGNORED; PROCESSING OF THE REPEAT BLOCK CONTINUES.

C#45 INPUT/OUTPUT SPECIFICATION REQUIRED.
A INPUT/EMPTY-CONDITION OR OUTPUT-LITERAL FIELD DID NOT CONTAIN A REQUIRED INPUT OR OUTPUT STATEMENT.

C#46 INCONSISTENT SUBSTREAM USAGE.
A FIELD DEFINITION APPEARS OUTSIDE OF A SUBSTREAM BLOCK IN A MULTI-RECORD STREAM DEFINITION -OR- THE USER HAS ATTEMPTED TO START A SUBSTREAM DEFINITION WHEN PREVIOUSLY DEFINED FIELDS DO NOT RESIDE WITHIN A SUBSTREAM. THIS ERROR MESSAGE IS ONLY ISSUED ONCE PER STREAM DESCRIPTOR.

6.12 FDL TEMPORARY FILES

IN THE PROCESS OF TRANSLATING A SOURCE FILE, THE FDL TRANSLATOR MAY PRODUCE ONE OR MORE OF THE FOLLOWING FILES:

NAME	FORMAT	CONTENTS
ER##UU	ASCII	ERROR DEFINITIONS (*)
RP##UU	ASCII	CURRENT REPEAT BLOCK
IN##UU	ASCII	INPUT STREAM/SUBSTREAM DEFINITION
OU##UU	ASCII/ BINARY	OUTPUT STREAM/SUBSTREAM FORMAT DEVICE FORMAT MAP

NOTE THAT ALL FILES ARE CREATED -AND- DELETED BY FDL; THE ONLY WAY THAT THE USER CAN "SEE" THESE IS IF HE BREAKS (CONTROL/P'S) OUT OF THE TRANSLATOR OR PERFORMS A LISTF WHILE ANOTHER USER IS RUNNING FDL IN THE SAME UFD.

* THE 'UU' IN THE FILE NAME DENOTES THE CURRENT USER # - THIS PERMITS MULTIPLE FDL TRANSLATIONS SIMULTANEOUSLY WITHIN THE SAME DIRECTORY.

7 FORMS ADMINISTRATIVE COMMAND PROCESSOR

THE FORMS ADMINISTRATIVE COMMAND PROCESSOR PROVIDES THE SYSTEM ADMINISTRATOR WITH THE NECESSARY TOOLS TO CREATE AND MAINTAIN THE FORM DEFINITION CATALOG, CONFIGURE NEW TERMINALS AND NEW DEVICE DRIVERS INTO THE FORMS SYSTEM, AND TO OBTAIN THE STATUS OF THE SYSTEM.

7.1 FAP COMMANDS

THE FOLLOWING IS A DESCRIPTION OF THE NINE COMMANDS SUPPORTED BY FAP. ALL COMMAND NAMES MAY BE ABBREVIATED TO 3 CHARACTERS.

7.1.1 CREATE COMMAND

THE CREATE (OR CREATE DIRECTORY) COMMAND ALLOWS THE SYSTEM ADMINISTRATOR TO CREATE A SKELETON FORMS CATALOG.

IF THE FORMS SYSTEM DIRECTORY (FORMS*) EXISTS, FAP CREATES EMPTY CATALOG AND TERMINAL CONFIGURATION FILES AND PRINTS A MESSAGE INFORMING THE USER THAT THESE FILES WERE CREATED.

IF THE FORMS DIRECTORY DOES NOT EXIST, FAP REQUESTS A DISK VOLUME-ID ON WHICH THE UFD IS TO BE CREATED. THE USER MUST THEN ENTER THE VOLUME-ID (DSKRAT NAME) OF THE PACK/PARTITION WHICH WILL CONTAIN THE FORMS DIRECTORY. FAP THEN ASKS THE USER FOR THE MFD OWNER PASSWORD ON THIS VOLUME. AFTER THIS INFORMATION HAS

BEEN ENTERED, THE FORMS UFD, CATALOG, AND TERMINAL CONFIGURATION FILES ARE CREATED. THE CREATE COMMAND WILL PRODUCE AN ERROR IF THE FORMS CATALOG ALREADY EXISTS. TO CREATE A FRESH COPY, THE OLD FILE MUST FIRST BE TREDEL'ETED WITH FUTIL.

SHOULD THE USER CREATE THE FORMS* UFD WITH FAP, HE MUST COPY THE FOLLOWING FILES TO THIS DIRECTORY PRIOR TO EXECUTING AN APPLICATIONS PROGRAM WHICH USES FORMS:

DCF.AS RUN.ER DCF.BN

THESE FILES MAY BE FOUND IN THE FORMS* UFD AS RELEASED ON THE MASTER DISK.

ALL ERROR MESSAGES PRODUCED BY THE CREATE COMMAND PROCESSOR ARE SELF-EXPLANATORY.

FOLLOWING IS AN EXAMPLE OF CREATE COMMAND DIALOGUE. NOTE THAT ALL UNDERLINED DATA IS ENTERED BY THE USER.

```
OK, FAP
GO

FAP REV 15   11-FEB-78

* CREATE

UFD "FORMS*" DOES NOT EXIST.
SHALL I CREATE IT? YES
ENTER DISK VOLUME-ID:  IS/A
ENTER OWNER PASSWORD  (IT WON'T ECHO):  ABCXYZ
THIS MFD IS FULL, TRY AGAIN.

ENTER DISK VOLUME-ID:  SOFIWR
ENTER OWNER PASSWORD  (IT WON'T ECHO):  XXXXXX

TCB CREATED.
DIRECTORY CREATED.

*
```

ON ANY INPUT REQUEST WITHIN THE CREATE DIALOGUE, THE USER MAY TYPE CONTROL/C TO ABORT CREATION AND RETURN TO THE FAP COMMAND LEVEL.

7.1.2 ADD COMMAND

THIS COMMAND ALLOWS THE ADDITION OF FORM DEFINITIONS TO THE FORMS CATALOG. THE NAME OF THE BINARY FORM DEFINITION FILE, GENERATED BY THE FDL TRANSLATOR, MUST FOLLOW THE ADD COMMAND. THIS FILE NAME USUALLY STARTS WITH 'B_'. NOTE THAT ONE BINARY FILE MAY CONTAIN MORE THAN ONE FORM DEFINITION, EG. IF THERE WAS ONE STREAM DESCRIPTOR AND ONE FORMAT DESCRIPTOR WITH DEFINITIONS FOR

THREE DEVICES, THE BINARY FILE CONTAINS FOUR FORM DEFINITIONS. FAP CONSIDERS EACH DEVICE DESCRIPTOR UNDER ONE FORMAT DESCRIPTOR TO BE A SEPERATE FORM.

THE ADD COMMAND ONLY ADDS NEW MODULES TO THE FORMS CATALOG; ANY ATTEMPT TO REDEFINE A FORM ALREADY RESIDING IN THE FORMS CATALOG WITH THE ADD COMMAND CAUSES THE NEW FORM DEFINITION TO BE IGNORED AND A WARNING MESSAGE PRINTED ON THE USER TERMINAL.

THE INPUT (BINARY) FILE NAME MAY OPTIONALLY BE FOLLOWED BY THE PARAMETER LIST OR LIST UPDATES. IF THIS IS SPECIFIED, ALL FORM DEFINITIONS ADDED TO THE FORMS DIRECTORY ARE LISTED BY NAME ON THE TERMINAL.

WHEN THE ENTIRE BINARY FILE HAS BEEN PROCESSED, THE NUMBER OF MODULES ADDED AND IGNORED (DUE TO DUPLICATE ENTRIES) IS PRINTED.

IF ANY TRANSLATION ERRORS WERE GENERATED BY FDL, THE MESSAGE 'WARNING! "FORM-NAME" CONTAINS ERRORS' WILL BE OUTPUT. THE USER SHOULD CORRECT THE SOURCE FILE AND RE-TRANSLATE IT WITH FDL. THIS BINARY FORM DEFINITION WILL PROBABLY GENERATE UNDESIRABLE RESULTS AT RUN-TIME.

EXAMPLES:

```
* ADD_B-FM03
01 DEFINITION ADDED.
* ADD_B-FM04_LIST

  DEDUCT    STR              V00   ADDED
  DEDUCT    FMT   VISTAR3    V00   ADDED
  DEDUCT    FMT   PRINTER    V00   ADDED

03 DEFINITIONS ADDED.
*
```

7.1.3 REPLACE COMMAND

THIS COMMAND FUNCTIONS THE SAME AS THE ADD COMMAND, BUT CAUSES ANY FORM DEFINITIONS IN THE FORMS CATALOG WHICH ARE REDEFINED IN THE INPUT (BINARY) FILE TO BE REPLACED WITH THE NEW DEFINITION. ANY FORM DEFINITIONS IN THE BINARY FILE THAT ARE NOT DEFINED IN THE CATALOG ARE ADDED.

EXAMPLES:

```
* REPLACE_B=F019
02 DEFINTIONS REPLACED.
* REPLACE_B=F020
01 DEFINITION ADDED   03 DEFINITIONS REPLACED.
*
```

7.1.4 PURGE COMMAND

THIS COMMAND PURGES FORM DEFINITIONS FROM THE FORMS CATALOG. THE COMMAND MUST BE FOLLOWED BY A FORM NAME SPECIFICATION (SEE BELOW), WHICH DESIGNATES WHAT FORM DEFINITIONS ARE TO BE PURGED. IT MAY ALSO BE FOLLOWED BY THE WORD LIST OR LIST UPDATES, WHICH WILL CAUSE ALL PURGED FORMS TO BE LISTED BY NAME ON THE USER TERMINAL.

THE FORM NAME SPECIFICATION DESIGNATES THE FORM DEFINITIONS TO WHICH THIS COMMAND APPLIES. BOTH PURGE AND LIST COMMANDS USE THIS OPTION.

THE FORM NAME SPECIFICATION IS ENCLOSED WITHIN PARENTHESIS AND HAS THE FOLLOWING FORMATS:

```
FORM-NAME
FORM-NAME.TYPE
FORM-NAME.TYPE:DEVICE

TYPE  =  STR    FOR STREAM DESCRIPTOR
         FMT    FOR FORMAT DESCRIPTOR
```

IF ONLY THE FORM-NAME IS SPECIFIED, THIS COMMAND RELATES TO ALL FORMS WITH THE GIVEN NAME, ANY TYPE AND ANY DEVICE (IF FORMAT). IF THE SECOND SPECIFICATION IS USED, THE COMMAND RELATES TO ALL FORMS OF THE GIVEN NAME AND TYPE. IF THE TYPE IS FMT, IT RELATES TO ALL DEVICE DESCRIPTORS WITHIN THE FORMAT DEFINITION. IF THE THIRD TYPE OF SPECIFICATION IS USED, THE COMMAND RELATES TO THE ONE DEFINITION THAT CONTAINS THE SAME NAME, TYPE, AND DEVICE. NOTE THAT THIS CONSTRUCTION SHOULD ONLY BE USED ON FORMAT DESCRIPTORS (THERE IS NO DEVICE DEFINITION FOR A STREAM DESCRIPTOR!).

IF ANY ITEM IN THE FORM NAME SPECIFIER (FORM-NAME, TYPE, OR DEVICE) IS SPECIFIED AS AN ASTERISK (*) OR THE WORD ANY, THIS WILL CAUSE NO CHECK TO BE MADE ON THIS ITEM WHEN SCANNING THE FORMS CATALOG.

UP TO 20 FORM NAMES MAY BE SPECIFIED WITHIN THE PARENTHESIS, SEPERATED BY COMMAS.

EXAMPLES:

(TAXFORM)	ALL FORMS OF NAME 'TAXFORM', ANY TYPE, ANY DEVICE
(TAXFORM.STR)	TAXFORM, STREAM DEFINITION
(TAXFORM.FMT:PRINTER)	PRINTER FORMAT DEFINITION FOR TAXFORM
(*.*:VISTAR3)	ALL VISTAR3 DEVICE FORMAT DEF'S
(*.STR)	ALL STREAM DESCRIPTORS
(TAXFORM,SHIPFORM)	ALL FORMS WITH NAMES TAXFORM OR SHIPFORM, ANY TYPE, ANY DEVICE

PURGE EXAMPLES:

```
* PURGE (FM0020)
01 DEFINITION PURGED.
* PURGE (FM0021,FM0022,FM0023,FM0024) LIST

  FM0021      STR                 V02   PURGED
  FM0022      STR                 V00   PURGED
  FM0024      STR                 V00   PURGED
  FM0024      FMT   PRINTER       V00   PURGED
  FM0024      FMT   VISTAR3       V00   PURGED

05 DEFINITIONS PURGED.
*
```

7.1.5 LIST COMMAND

THIS COMMAND CAUSES ALL OR PART OF THE FORMS CATALOG TO BE LISTED BY NAME AND TYPE. THIS MAY BE FOLLOWED BY A FORM NAME SPECIFIER (SEE ABOVE) TO SELECTIVELY LIST A PART OF THE CATALOG. IF THE FORM NAME SPECIFIER IF OMITTED, THE ENTIRE CATALOG IS LISTED. IF THE PHRASE FILE <FILENAME> OR ON FILE <FILENAME> IS INCLUDED, THE CATALOG LISTING IS OUTPUT TO THE SPECIFIED FILE. IF THE PHRASE ON TERMINAL IS SPECIFIED, OR IF THE ON FILE SPECIFIER IS OMITTED, THE LISTING IS WRITTEN TO THE USER TERMINAL.

THE INFORMATION LISTED IN THE CATALOG LISTING INCLUDES:

- FORM-NAME, TYPE, AND DEVICE (IF ANY)
- VERSION NUMBER
- OWNER (LOGIN) NAME
- CREATION, LAST ACCESS, LAST MODIFIED DATES (FILE OUTPUT ONLY)

EXAMPLES:

```
* LIST

FORMS DIRECTORY ON THURSDAY, FEBRUARY 2, 1978 AT 9:45 PM

 NAME       TYPE    DEVICE     VER    OWNER

HDRF01      STR                V02    JIMW
HDRF01      FMT     VISTAR3    V00    JIMW
HDRF02      STR                V00    DAVEW
HDRF02      FMT     VISTAR3    V00    DAVEW

04 ENTRIES.
* LIST (HDRF01) ON FILE CATLOG
*
```

7.1.6 QUIT COMMAND

THE QUIT COMMAND CAUSES FAP TO EXIT AND RETURN TO PRIMOS COMMAND LEVEL. FAP MAY BE RE-ENTERED BY TYPING THE START (S) COMMAND.

EXAMPLE:

```
* QUIT

OK,
```

7.1.7 JOURNAL COMMAND

THE JOURNAL COMMAND ALLOWS THE USER TO LOG HIS TRANSACTIONS WITH THE FORMS CATALOG IN AN ASCII FILE WHICH CAN BE SPOOLED TO THE LINE-PRINTER. ALL ADD, REPLACE, PURGE, AND TCB (SEE BELOW) TRANSACTIONS ARE RECORDED IN THE JOURNAL FILE.

THIS COMMAND MAY BE USED TO ENABLE OR DISABLE THE LOGGING FUNCTION. TO DISABLE IT, THE COMMAND JOURNAL OR JOURNAL STOP MAY BE ISSUED. TO ENABLE IT, THE COMMAND JOURNAL <FILENAME> OR JOURNAL START ON <FILENAME> MAY BE ISSUED.

EXAMPLE:

```
* JOURNAL LOG000
* ADD B=F01
08 DESCRIPTIONS ADDED.
* JOURNAL STOP
*
```

7.1.8 TCB COMMAND

THE TCB COMMAND MODIFIES THE TERMINAL CONFIGURATION FILE. THIS FILE CONTAINS A 64 BY 4 WORD TABLE WHICH DESCRIBES THE TERMINAL TYPE FOR EACH FORMS USER ON THE (LOCAL) COMPUTER SYSTEM. IT IS USED IN CONJUNCTION WITH THE DEVICE CONTROL FILE (DCF) AT RUN-TIME TO SELECT THE TERMINAL DEVICE DRIVER FOR A GIVEN FORMS USER. BOTH TCB AND DCF FILES ARE EXPLAINED IN DETAIL LATER IN THIS DOCUMENT.

THE TCB COMMAND MAY BE FOLLOWED BY THE WORD LIST TO DUMP THE CONTENTS OF THE TCB ON THE USER TERMINAL. OPTIONALLY, LIST MAY BE FOLLOWED BY A FILE NAME. IF SO, THE CONTENTS OF THE TCB WILL BE DUMPED TO THE SPECIFIED FILE.

TO MODIFY THE TCB, THE COMMAND MAY OPTIONALLY BE FOLLOWED BY ONE OF THREE NOISE WORDS TO REFLECT THE TYPE OF OPERATION BEING PERFORMED: ADD, CHANGE, OR DROP. THIS MUST THEN BE FOLLOWED BY THE USER NUMBER FOR WHICH THIS OPERATION APPLIES, FROM 1 TO 64. IF THE USER WISHES TO DROP THE CURRENT TCB ENTRY FOR THIS USER, HE MAY TERMINATE THE COMMAND LINE BY TYPING RETURN. IF THE USER ATTEMPTS TO DROP AN NON-EXISTANT ENTRY, FAP PRINTS A WARNING MESSAGE AND RETURNS TO COMMAND MODE. IF HE WISHES TO ADD OR CHANGE THE TERMINAL TYPE, HE MUST TYPE THE 1-8 CHARACTER TERMINAL NAME. IF THE SPECIFIED USER ALREADY HAD AN ENTRY, THE NAME OF THE OLD TERMINAL TYPE IS PRINTED ON THE TERMINAL.

EXAMPLES:

```
* TCB LIST

TERMINAL CONFIGURATION ON FRIDAY, MARCH 4, 1977 AT 9:03 PM

USER    TERMINAL

 4      VISTAR3
12      Z9003
13      VISTAR3
20      Z9003

* TCB 3 VISTAR3              (SET USER 3 = VISTAR3)
* TCB 12 B500                (CHANGE USER 12 TO B500)
WAS Z9003.
* TCB 13                     (DROP USER 13'S ENTRY)
* TCB LIST
```

```
TERMINAL CONFIGURATION ON FRIDAY, MARCH 4, 1977 AT 9:03 PM

USER    TERMINAL

 3      VISTAR3
 4      VISTAR3
12      B500
20      Z9003

*
```

7.1.9 GENERATE COMMAND

THE GENERATE COMMAND IS ISSUED WHENEVER THE DEVICE CONTROL FILE HAS BEEN MODIFIED. THIS IS NORMALLY THE CASE WHEN A NEW DEVICE DRIVER HAS BEEN ADDED TO THE SYSTEM OR A DEVICE DRIVER HAS BEEN REMOVED. THIS COMMAND CREATES THREE $INSERT FILES AND ONE BINARY FILE FORMS* DIRECTORY.

THE FILES GENERATED ARE AS FOLLOWS:

- DEVEXT - EXTERNAL DECLARATION STATEMENTS FOR RUN-TIME DEVICE DRIVERS
- DEVDAC - 64R MODE DRIVER DISPATCH TABLE
- DEVIP - 64V MODE DRIVER DISPATCH TABLE
- DCF.BN - BINARY REPRESENTATION OF THE NEW DEVICE CONTROL FILE (DCF.AS)

THE GENERATE COMMAND SHOULD BE ISSUED AND A NEW RUN-TIME I/O PACKAGE ASSEMBLED EACH TIME THE DEVICE CONTROL FILE (DCF) IS MODIFIED.

7.1.10 LINK COMMAND

THE LINK COMMAND HAS BEEN ADDED TO FAP TO ALLOW USERS TO UPGRADE TO THE CURRENT REV WITHOUT HAVING TO REBUILD EACH FORM DEFINITION. IT ALSO MAY BE USED TO RECOVER FROM VARIOUS FORM DEFINITION FILE INCONSISTENCIES WHEN A RUN TIME ERROR DICTATES SUCH RECOVERY IS NECESSARY.

THERE ARE TWO FORMS OF THE LINK COMMAND. "LINK ALL" SPECIFIES THAT ALL FORM DEFINITIONS CONTAINED IN THE FORMS DIRECTORY ARE TO BE (RE-) LINKED. IF THE COMMAND IS FOLLOWED BY A FORM NAME SPECIFICATION, AS DESCRIBED ABOVE, ONLY THE SPECIFIED FORM DEFINITIONS ARE LINKED.

LINKING IS THE PROCESS WHICH COMBINES THE STREAM AND FORMAT DESCRIPTORS INTO ONE MELDED FORM DEFINITION. THIS FORM DEFINITION IS THEN STORED IN A FILE IN THE FORMS>LNK.FD DIRECTORY FOR FASTER ACCESS AT EXECUTION TIME. IT IS THE LINKED FORM

DEFINITION AND NOT THE INDIVIDUAL STREAM AND FORMAT DESCRIPTOR THAT IS USED WHEN A FORM IS INVOKED AT EXECUTION TIME. PRIOR TO REV 15, FORM DEFINITIONS WERE LINKED AT EXECUTION TIME, A PROCESS WHICH CAUSED A LENGTHY DELAY WHEN A FORM WAS INVOKED.

THE LINKED FORM DEFINITIONS ARE INVISIBLE TO THE USER. THEY ARE AUTOMATICALLY CREATED OR UPDATED BY FAP WHEN A FORM DEFINITION IS ADDED TO OR REPLACED IN THE CATALOG. THE CORRESPONDING LINK FILE IS DELETED WHEN A FORMAT DESCRIPTOR IS PURGED.

7.2 USING FAP

FAP IS INVOKED BY TYPING THE COMMAND 'FAP' FOLLOWING THE 'OK, ' PROMPT ISSUED BY THE OPERATING SYSTEM. FAP PRINTS A HEADER LINE, FOLLOWED BY THE CURRENT REVISION NUMBER. IF BIT 1 IN THE A-REGISTER IS SET WHEN FAP IS STARTED, ALL UPDATES TO THE FORMS DIRECTORY AND TERMINAL CONFIGURATION TABLE ARE AUTOMATICALLY RECORDED IN A FILE CALLED 'FAP.UP' IN THE FORMS CONTROL DIRECTORY. IT IS STRONGLY RECOMMENDED THAT IF THIS OPTION IS TO BE USED ALL OF THE TIME, FAP BE RESTORE'D AND SAVE'D WITH THE A-REGISTER SET APPROPRIATELY. WHEN THIS OPTION IS USED, THE JOURNAL COMMAND IS DISABLED.

7.3 FAP ERROR MESSAGES

LIKE FDL, ALL FAP ERROR MESSAGES ARE OF THE FORM:

```
T#NN   TEXT MESSAGE
```

THE 'T' IN THE ERROR CODE REPRESENTS THE ERROR TYPE. AT PRESENT, THERE ARE THREE SUCH TYPES:

- F - FILE SYSTEM/INPUT FILE/CONTROL BLOCK ERROR
- S - SYNTAX ERROR
- T - TCB OR DCF FORMAT ERROR

THE 'NN' REPRESENTS A 2-DIGIT ERROR NUMBER, UNIQUE FOR EACH ERROR MESSAGE GENERATED BY FAP.

FOLLOWING IS A LIST OF ERROR MESSAGES AND EXPLANANTIONS:

F#00 CONTROL BLOCK UFD DOES NOT EXIST.
AN OPERATION OTHER THAN CREATE WAS ATTEMPTED AND THE FORMS UFD ('FORMS*') DOES NOT EXIST ON THE SYSTEM.

F#01 CONTROL BLOCK DIRECTORY DOES NOT EXIST.
AN OPERATION OTHER THAN CREATE WAS ATTEMPTED AND THE FORMS SEGMENT DIRECTORY ('FMS.**') DOES NOT EXIST WITHIN THE FORMS UFD.

F#04 INPUT FILE IS EMPTY.
THE INPUT FILE SPECIFIED IN AN ADD OR REPLACE COMMAND IS EMPTY.

F#05 PREMATURE EOF.
AN EOF WAS ENCOUNTERED ON THE INPUT FILE IN AN ADD OR REPLACE COMMAND BEFORE THE END-OF-DATA RECORD. THE MODULE IS DELETED FROM THE CONTROL DIRECTORY. THIS IS USUALLY CAUSED BY THE USER DEPRESSING THE BREAK KEY IN THE MIDDLE OF AN FDL COMPILATION.

F#06 FILE DOES NOT EXIST.
THE INPUT FILE SPECIFIED IN AN ADD OR REPLACE COMMAND DOES NOT EXIST IN THE CURRENT UFD.

F#07 BAD INPUT FILE.
THE INPUT FILE SPECIFIED IN AN ADD OR REPLACE COMMAND IS NOT A VALID FDL BINARY FILE. NO ACTION IS TAKEN WITH THIS FILE.

F#08 I/O LIST OVERFLOW, LINK SUPPRESSED.
FAP RAN OUT OF ROOM WHILE ATTEMPTING TO LINK A STREAM DEFINITION TO A FORMAT DEFINITION. THE INTERNAL I/O BUFFER MUST BE ENLARGED BEFORE THIS FORM DEFINITION MAY BE ADDED. INCREASE THE VALUE OF IOLSIZ IN THE $INSERT FILE FORMS>FAP>IOBUF$ AND REBUILD FAP.

F#09 STREAM/FORMAT BUFFER OVERFLOW.
FAP RAN OUT OF ROOM ATTEMPTING TO READ A STREAM OR FORMAT DESCRIPTOR BINARY FILE. THE BUFFER MUST BE ENLARGED AND FAP REBUILT BEFORE THIS FORM DEFINITION CAN BE ADDED. INCREASE THE VALUE OF SFBSIZ IN THE $INSERT FILE FORMS>FAP>IOBUF$ AND REBUILD FAP.

F#10 ERROR READING STREAM / FORMAT DESCRIPTION.
A FILE SYSTEM ERROR OCCURRED WHILE ATTEMPTING TO LOAD A STREAM OR FORMAT DESCRIPTOR.

F#11 ERROR READING / DELETING LINK FILE.
A FILE SYSTEM ERROR OCCURRED WHEN FAP WAS TRYING TO PURGE A LINKED FORM DEFINITION FILE.

F#12 ERROR RENAMING LINK FILE.
AN ERROR OCCURRED WHEN FAP ATTEMPTED TO RENAME A LINK FILE FOLLOWING A PURGE OPERATION.

S#00 FILE NAME REQUIRED.
AN ADD OR REPLACE COMMAND WAS ISSUED, BUT NO FILE NAME FOLLOWED. THE COMMAND IS IGNORED.

S#01 BAD FORM NAME SPECIFIER.
THE FORM NAME SPECIFIER CONTAINED A SYNTAX ERROR. THIS COMMAND IS IGNORED.

S#02 BAD ARGUMENT.

ONE OF THE PARAMETERS IN THE COMMAND LINE WAS NOT RECOGNIZED. THE COMMAND IS IGNORED.

S#03 BAD TYPE.
THE FORM NAME SPECIFIER CONTAINED A TYPE DECLARATION OTHER THAN STR (STREAM) OR FMT (FORMAT). THIS COMMAND IS IGNORED.

S#04 NO FORM NAME SPECIFIED.
A PURGE COMMAND WAS ISSUED WITHOUT A REQUIRED FORM NAME SPECIFIER. THE PURGE COMMAND IS IGNORED.

S#05 MISSING ARGUMENT.
THE TCB COMMAND WAS ISSUED WITHOUT ANY FOLLOWING USER NUMBER. THE COMMAND IS IGNORED.

S#06 BAD USER NUMBER.
THE USER NUMBER SPECIFIED IN THE TCB COMMAND IS NOT AN INTEGER NUMBER GREATER THAN ZERO. THE TCB COMMAND IS IGNORED.

S#07 BAD TERMINAL NAME.
THE USER ATTEMPTED TO ASSIGN THE NAME 'PRINTER' AS A TERMINAL TYPE IN A TCB COMMAND. THIS IS NOT PERMITTED AND THE TCB COMMAND IS IGNORED.

T#00 DCF DEVICE INTERLUDE FIELD ERROR.
THE DEVICE INTERLUDE NUMBER FIELD IN THE GIVEN DCF ENTRY IS NOT NUMERIC OR GREATER THAN ZERO. THE DCF MUST BE EDITED AND CORRECTED BEFORE CONTINUING.

T#01 DCF DEVICE NAME FIELD ERROR.
THE DEVICE NAME FIELD IN THE GIVEN DCF ENTRY CONTAINS AN ILLEGAL CHARACTER OR IS EMPTY. THE DCF MUST BE EDITED AND CORRECTED BEFORE CONTINUING.

T#02 DCF DEVICE ABBREVIATION FIELD ERROR.
THE DEVICE ABBREVIATION FIELD IN THE GIVEN DCF ENTRY IS EMPTY OR CONTAINS A SPACE OR ILLEGAL CHARACTER. THE DCF MUST BE EDITED AND CORRECTED BEFORE CONTINUING.

T#03 TCB LINE/COLUMN FIELD ERROR.
THE LINE OR COLUMN SPECIFICATION FIELD IN THE GIVEN DCF ENTRY IS EMPTY, CONTAINS A NON-NUMERIC VALUE, OR IS LESS THAN 1. THE DCF MUST BE EDITED AND CORRECTED BEFORE CONTINUING.

T#04 MAX DEVICE NUMBER EXCEEDED IN DCF.
THE DCF CONTAINS AN ENTRY WITH A DEVICE INTERLUDE NUMBER GREATER THAN 50. THIS ERROR IS ISSUED FROM THE GENERATE COMMAND ONLY. ONLY 50(!) DEVICES MAY BE IN USE AT ONE TIME.

T#05 DEVICE CONTROL FILE EMPTY.
THE DCF IS EMPTY AND THE USER ISSUED A TCB OR GENERATE COMMAND.

T#06 TERMINAL UNDEFINED.
THE TERMINAL TYPE SPECIFIED IN THE TCB COMMAND IS NOT PRESENT IN THE DCF.

8 FORMS RUN TIME PACKAGE

THE FORMS RUN TIME PACKAGE INVISIBLY PERFORMS ALL FORM DEFINITION LOOKUP, COMMAND PROCESSING, DATA MANIPULATION, AND DEVICE INPUT/OUTPUT.

8.1 FORMS RUN TIME COMMANDS

THE APPLICATIONS PROGRAM ISSUES A FORMS COMMAND BY WRITING A RECORD CONTAINING THE COMMAND (AND ARGUMENTS) TO THE APPROPRIATE DEVICE (LOGICAL UNIT 1 FOR THE FORM IN USE ON THE TERMINAL, LOGICAL UNIT 4 FOR THE LINE-PRINTER.) THE RECORD IS PRECEDED BY TWO HASH MARKS (#) IN ORDER TO BE IDENTIFIED BY FORMS AS A COMMAND.

FOR EXAMPLE, TO INVOKE FORM 'FD4190' ON THE TERMINAL AND PROTECT FIELDS FIELDA, FIELDB, AND FIELDC, A FORTRAN APPLICATIONS PROGRAM WOULD EXECUTE THE FOLLOWING STATEMENTS:

```
      WRITE (1,400)
400   FORMAT ('##INVOKE FD4190'/
     +        '##PROTECT FIELDA, FIELDB, FIELDC')
```

FOLLOWING IS A DESCRIPTION OF ALL COMMANDS AVAILABLE IN THE RUN TIME PACKAGE. FOR CLARITY, THEY ARE EACH DEPICTED IN UPPER CASE, HOWEVER ALL LOWER CASE CHARACTERS IN FORMS COMMANDS ARE MAPPED TO UPPER CASE.

8.1.1 INVOKE COMMAND

THIS COMMAND DEFINES THE FORM DEFINITION TO BE USED. IT IS FOLLOWED BY THE FORM (STREAM DESCRIPTOR) NAME.

WHEN THE INVOKE COMMAND IS ISSUED, FORMS SEARCHES THE CATALOG FOR THE SPECIFIED FORM DEFINITION. IF FOUND, IT IS READ INTO MEMORY AND INITIALIZED. IF NOT FOUND, AN ERROR MESSAGE IS PRINTED AND RETURN IS MADE TO SYSTEM COMMAND LEVEL. WHEN A FORM IS INVOKED ON A DEVICE, ALL INPUT AND OUTPUT REQUESTS FOR THAT DEVICE ARE TRAPPED AND HANDLED BY THE RUN TIME PACKAGE. WHEN THE FORM DEFINITION IS SUBSEQUENTLY RELEASED, ALL I/O IS HANDLED NORMALLY (VIA CALLS TO STANDARD IOCS SUBROUTINES.)

IF ANOTHER FORM DEFINITION WAS INVOKED AND NOT RELEASED PRIOR TO ISSUANCE OF THIS INVOKE COMMAND, AN IMPLIED RELEASE OCCURS.

EXAMPLE:

```
          WRITE (1,120)
  120     FORMAT ('##INVOKE TAXD01')
```

8.1.2 RELEASE COMMAND

THE RELEASE COMMAND SPECIFIES THAT THE CURRENT FORM DEFINITION IS NO LONGER TO BE USED. ALL FUTURE I/O IS PROCESSED NORMALLY UNTIL THE NEXT INVOKE COMMAND.

EXAMPLE:

```
          WRITE (1,900)
  900     FORMAT ('##RELEASE')
```

8.1.3 VALIDAIE COMMAND

THIS COMMAND CAUSES THE RUN-TIME PACKAGE TO RETURN THE VALIDATION STATUS OF ALL INPUT DATA ON THE NEXT READ STATEMENT(S).

THE STATUS IS RETURNED IN THE FORM OF A 2-DIGIT NUMBER FOR EACH INPUT/EMPTY-CONDITIONAL OR DIRECT FIELD THAT IS NOT DECLARED AS OUTPUT-ONLY. IT IS NOT RETURNED FOR INPUT-LITERAL FIELDS. THE TWO-DIGIT NUMBER RETURNED REPRESENTS ONE OF THE FOLLOWING THREE CONDITIONS:

VALUE	CONDITION
-1	THE DATA FAILED ALL VALIDATION TESTS.
00	NO VALIDATION SPECIFIED FOR THIS FIELD.
>0	THIS IS THE NUMBER OF THE FIRST VALIDATION MASK THAT THE DATA PASSED. VALIDATION MASKS ARE NUMBERED IN THE ORDER IN WHICH THEY APPEAR IN THE FIELD DEFINITION.

THE VALIDATION STATUS IS RETURNED IN THE SAME MANNER THAT DATA IS RETURNED ON A READ STATEMENT. IF THERE ARE MULTIPLE SUBSTREAM DEFINITIONS, THE USER MUST DO MULTIPLE READS TO INPUT THE VALIDATION STATUS FOR ALL FIELDS. THE VALIDATE COMMAND CAUSES THE NEXT READ STATEMENT TO INPUT THE VALIDATION STATUS OF THE FIRST SUBSTREAM IN THE STREAM DESCRIPTOR UNLESS A SUBSTREAM COMMAND IS ISSUED BEFORE IT. THE VALIDATE FUNCTION IS DISABLED AND NORMAL DATA INPUT RESUMED WHEN EITHER THE END OF THE STREAM DESCRIPTOR IS ENCOUNTERED OR A SUBSTREAM OR FORCEREAD COMMAND IS ISSUED.

EXAMPLE:

```
C--- INPUT FIELD VALIDATION.  FIRST SUBSTREAM CONTAINS 5 INPUT
C    FIELDS, SECOND CONTAINS 4.
C
      WRITE (1,300)                      /* POS TO FIRST SUBSTREAM
300   FORMAT ('##SUBSTREAM ONE'/'##VALIDATE')
      READ (1,310) (IVAL(I),I=1,9)  /* READ VALIDATION
310   FORMAT(5I2/4I2)
```

8.1.4 SUBSTREAM COMMAND

THIS COMMAND DEFINES THE SUBSTREAM TO BE PROCESSED ON THE NEXT READ OR WRITE STATEMENT. THE SUBSTREAM NAME MUST FOLLOW THE COMMAND AND BE SEPARATED FROM IT BY AT LEAST ONE SPACE. IF THE NAMED SUBSTREAM DOES NOT EXIST IN THE STREAM DESCRIPTOR, AN ERROR MESSAGE WILL BE GENERATED AND THE PROGRAM WILL ABORT.

EXAMPLE:

```
      WRITE (1,200)
200   FORMAT ('##SUBSTREAM NAMADDR')
```

8.1.5 CLEAR COMMAND

THIS COMMAND CLEARS ALL UNPROTECTED DATA DISPLAYED ON THE USER TERMINAL. IT ALSO CAUSES ALL DATA ITEMS MARKED AS UNPROTECTED AND DISPLAYED IN THE INPUT/OUTPUT LIST TO BE RESET TO SPACES. THIS IS A FAST AND CONVENIENT METHOD TO ERASE ALL OPERATOR-INPUT DATA. ALTERNATELY, THE USER MAY OUTPUT SPACES TO ALL UNPROTECTED FIELDS ON THE FORM.

IF THIS COMMAND IS FOLLOWED BY THE WORD 'ALL', THE ENTIRE DISPLAY IS ERASED. THIS IS USUALLY DONE IMMEDIATELY PRIOR TO A RELEASE COMMAND. THIS OPTION WAS ADDED TO ALLOW THE APPLICATIONS PROGRAM TO LEAVE THE TERMINAL IN A 'HUMAN' STATE BEFORE EITHER EXITING TO COMMAND LEVEL OR PERFORMING STANDARD RECORD I/O. SHOULD THE USER ATTEMPT TO PERFORM MORE INPUT OR OUTPUT OPERATIONS THROUGH FORMS, THE FORM DEFINITION WILL BE RE-DISPLAYED.

EXAMPLES:

```
        WRITE (1,400)
400     FORMAT ('##CLEAR')
            .
            .
C--- CLEAN UP BEFORE EXITING TO COMMAND LEVEL.
C
        WRITE (1,5020)
5020    FORMAT ('##CLEAR ALL'/ '##RELEASE')
```

8.1.6 ATTRIBUTE MODIFICATION COMMANDS

THE APPLICATIONS PROGRAM MAY DYNAMICALLY CHANGE THE ATTRIBUTES OF A FIELD BY ISSUING ONE OF THE ATTRIBUTE COMMANDS DESCRIBED BELOW. FROM ONE TO TWENTY STREAM DESCRIPTOR FIELD NAMES MAY BE PLACED AS ARGUMENTS, EACH SEPERATED BY AT LEAST ONE SPACE. THE ACTUAL MODIFICATION OCCURS AT THE NEXT WRITE OR READ IN WHICH DATA (AS OPPOSED TO FORMS COMMANDS) IS TRANSFERRED TO OR FROM THE DEVICE.

THE FOLLOWING TABLE DESCRIBES EACH OF THE EIGHT ATTRIBUTE MODIFICATION COMMANDS AND THREE SYNONYMS.

COMMAND/SYNONYM	DESCRIPTION
PROTECT	WRITE-PROTECTS FIELD
NOPROTECT/ENABLE	WRITE-ENABLES FIELD
RVIDEO	FIELD DISPLAYED IN REVERSE VIDEO
NVIDEO	FIELD DISPLAYED IN NORMAL VIDEO
BLINK	BLINKS FIELD WHEN DISPLAYED
NOBLINK	FIELD IS NOT BLINKED WHEN DISPLAYED
DISPLAY/FREE	FIELD IS DISPLAYED WHEN FORM IS OUTPUT
NODISPLAY/HOLD	FIELD IS NOT DISPLAYED WHEN FORM IS OUTPUT

EXAMPLE:

```
        WRITE (1,300)
300     FORMAT ('##PROTECT NAME, IDNUMBER, ADDRESS'/
     +          '##NOPROTECT REMARK1 REMARK2 REMARK3')
```

8.1.7 PRINT COMMAND

THIS COMMAND ALLOWS THE USER TO OUTPUT THE CURRENT FORM AND OPERATOR-ENTERED DATA FROM THE TERMINAL TO EITHER THE SPOOLED LINE-PRINTER OR A LOCAL PRINTER ATTACHED TO THE INDIVIDUAL TERMINAL. THIS PERMITS THE PROGRAM TO PRINT THE CURRENT TRANSACTION ON A HARD-COPY DEVICE WITHOUT DEFINING A SEPERATE FORMAT DESCRIPTOR FOR THE LINE-PRINTER.

THE PRINT COMMAND CAUSES THE FORM, AS IT APPEARS ON THE TERMINAL, TO BE INSERTED INTO THE SPOOLED PRINTER QUEUE. IF A PRINTER FORM HAS BEEN INVOKED AND THE PRINT COMMAND IS ISSUED, ALL FORMS OUTPUT TO THE PRINTER BY THE APPLICATION PROGRAM ARE PRINTED PRIOR TO THE DATA ON THE TERMINAL. NOTE THAT THIS DOES NOT RELEASE THE FORM INVOKED ON THE LINE-PRINTER.

IF THE PRINT COMMAND IS FOLLOWED BY THE WORD LOCAL, THE FORM AND ASSOCIATED DATA IS PRINTED ON THE HARDCOPY DEVICE ATTACHED TO THE USER'S TERMINAL. OBVIOUSLY, THIS OPERATION IS ONLY VALID ON A TERMINAL WITH A HARDCOPY UNIT ATTACHED.

EXAMPLES:

```
        WRITE (1,200)          /* WRITE TRANS. TO SYS PRINTER
200     FORMAT ('##PRINT')
```

8.1.8 POSITION COMMAND

THE POSITION COMMAND ALLOWS THE APPLICATIONS PROGRAMMER TO SPECIFY THE FIELD TO WHICH THE CURSOR WILL BE POSITIONED ON THE NEXT READ OPERATION. THIS COMMAND IS ONLY APPLICABLE TO THE NEXT READ; FOLLOWING READS WILL POSITION THE CURSOR TO THE FIRST UNPROTECTED CHARACTER POSITION ON THE TERMINAL UNLESS SUBSEQUENT POSITION COMMANDS ARE ISSUED.

EXAMPLE:

```
C--- POSITION TO FIELD SPECIFIED BY CONTENTS OF 'FLDNAM'.
C
        WRITE (1,250) FLDNAM
250     FORMAT ('##POSITION ',4A2)
```

8.1.9 FORCEREAD COMMAND

THIS COMMAND ALLOWS THE PROGRAMMER TO FORCE FORMS TO WAIT FOR AND PROCESS OPERATOR INPUT FROM THE TERMINAL, THEREFORE PROVIDING A FACILITY TO OVERRIDE THE NORMAL INPUT PROTOCOL WHEN PROCESSING A FORM DEFINITION WITH MULTIPLE SUBSTREAMS. NORMALLY, TERMINAL INPUT OCCURS WHEN THE APPLICATIONS PROGRAM EXECUTES THE FIRST READ STATEMENT AFTER THE FORM IS INVOKED, ISSUES A READ STATEMENT FOLLOWING A WRITE STATEMENT, OR ATTEMPTS TO READ A SUBSTREAM WHICH HAS ALREADY BEEN READ. THE FORCEREAD COMMAND CAUSES TERMINAL INPUT ON THE NEXT READ OPERATION, WHETHER OR NOT THE NEXT SUBSTREAM TO BE PROCESSED HAS ALREADY BEEN READ. REFER TO THE SECTION ON SUBSTREAM PROCESSING, EARLIER IN THIS DOCUMENT, FOR MORE DETAIL.

EXAMPLE:

```
        WRITE (1,200)
200     FORMAT ('##FORCEREAD')
        READ (1,210) IREC
210     FORMAT (32A2)
```

8.1.10 FKEYS COMMAND

THIS COMMAND ENABLES OR DISABLES OPERATOR FUNCTION KEY INPUT. IF THE COMMAND IS FOLLOWED BY THE ARGUMENT "OFF", FUNCTION KEYS ARE DISABLED; IF "ON", FUNCTION KEYS ARE ENABLED.

WHEN FUNCTION KEYS ARE DISABLED, THEY HAVE NO EFFECT IF ENTERED. THE STANDARD RESPONSE IS TO PLACE A WARNING MESSAGE ON THE TERMINAL AND WAIT FOR THE OPERATOR TO HIT THE CORRECT TRANSMIT KEY. WHEN A FORM IS INVOKED, AN IMPLICIT "FKEYS OFF" COMMAND OCCURS.

WHEN FUNCTION KEYS ARE ENABLED, A TWO DIGIT CODE IS AUTOMATICALLY APPENDED TO EACH INPUT RECORD FOLLOWING THE RIGHT MOST FIELD DEFINED IN THE (SUB)STREAM. THIS FIELD CONTAINS THE NUMBER OF THE FUNCTION KEY WHICH WAS DEPRESSED WHEN THE DATA WAS TRANSMITTED FROM THE DEVICE. IF THE NORMAL "TRANSMIT" KEY WAS DEPRESSED, THIS FIELD CONTAINS '00'.

IF MULTIPLE SUBSTREAMS ARE DESCRIBED IN THE FORM DEFINITION, THE FUNCTION KEY NUMBER IS APPENDED TO EACH.

IT IS THE PROGRAMMER'S RESPONSIBILITY TO INSURE THAT THE TWO CHARACTER POSITIONS REQUIRED FOR THE FUNCTION KEY FIELD ARE AVAILABLE AT THE END OF EACH INPUT RECORD. IF THESE TWO CHARACTER POSITIONS ARE NOT AVAILABLE, THE FUNCTION KEY NUMBER IS NOT RETURNED.

THE APPLICATIONS PROGRAM MAY DEFINE EACH FUNCTION KEY TO PERFORM SOME SPECIAL "ESCAPE" FUNCTION, SUCH AS REQUEST A NEW FORM DEFINITION, EXIT PROGRAM, PERFORM A DATABASE UPDATE WITH THE NEW DATA ENTERED ON THE TERMINAL, ETC.

AN OBVIOUS REMINDER: THE USER SHOULD NOT WRITE AN APPLICATIONS PROGRAM WHICH MAKES USE OF FUNCTION KEYS UNLESS ALL TERMINALS WHICH ARE TO RUN THIS PROGRAM ARE EQUIPPED WITH THEM.

8.2 RUN TIME FILE HANDLING

THE REV 15 VERSION OF FORMS USUALLY REQUIRES ONLY ONE FILE UNIT FOR ALL FILE I/O. THIS UNIT NUMBER IS SET TO 16 AS FORMS IS RELEASED BY PRIME. SHOULD THE USER REQUIRE THE USE OF THIS UNIT, HE MAY CHANGE THE DECLARATION OF VARIABLE TMPU IN THE FILE FORMS>RUN>BLKDAT AND RE-COMPILE THE RUN TIME PACKAGE.

THE EXCEPTION OCCURS WHEN THE USER USES THE SYSTEM PRINTER DEVICE DRIVER. IN ORDER TO COPY FILES INTO THE SPOOL QUEUE, TWO FILE UNITS ARE REQUIRED. UNITS 15 AND 16 ARE USED IN THE VERSION OF PR$IO RELEASED BY PRIME. TO ALLOCATE TWO OTHER UNITS, MODIFY VARIABLES FUNITC (=15), PRINFO(1) (=15), AND PRINFO(2) (=16) IN THE FILE FORMS>IOS>PR$IO. ALL ARE DECLARED IN DATA STATEMENTS.

8.3 ERROR HANDLING

ALL EXECUTION TIME ERROR DIAGNOSTICS GENERATED BY FORMS ARE SELF-EXPLANATORY.

ERROR TEXT IS STORED IN THE FILE FORMS*>RUN.ER. EACH LINE IN THE FILE IS PRECEDED BY A NUMERIC KEY, BETWEEN 1 AND 9999. IF A DIAGNOSTIC REQUIRES MORE THAN ONE LINE (AS DO MOST), EACH FOLLOWING LINE CONTAINS THE SAME NUMERIC KEY AS THE FIRST. THE END OF THE DIAGNOSTIC OCCURS WHEN A LINE WITH A DIFFERENT NUMERIC KEY IS ENCOUNTERED.

THE PROCEDURE WHICH CALLS THE FORMS ERROR HANDLER MAY SUPPLY FROM 0 TO 3 TEXTUAL ARGUMENTS. AN ARGUMENT IS INSERTED INTO THE ERROR DIAGNOSTIC WHEN A PERCENT SIGN (%) FOLLOWED BY A VALUE 1-3 IS ENCOUNTERED IN THE TEXT STRING.

THE CALLING SEQUENCE FOR THE RUN TIME ERROR HANDLER IS:

```
CALL FM$ERR (KEY, FSCODE, TEXT1, LEN1, TEXT2, LEN2, TEXT3, LEN3)
```

KEY	..	NUMERIC KEY OF THE ERROR DIAGNOSTIC TO BE PRINTED. IF THIS DIAGNOSTIC IS NOT INCLUDED IN THE FILE, AN ERROR MESSAGE IS PRINTED CONTAINING THE ERROR NUMBER. ALL ERRORS GENERATED BY FORMS HAVE CORRESPONDING TEXT MESSAGES IN RUN.ER.
FSCODE	..	THE FILE SYSTEM CODE ASSOCIATED WITH THIS ERROR CONDITION. IF THIS ERROR IS NOT A RESULT OF A FILE SYSTEM ERROR, THIS VALUE SHOULD BE ZERO.
TEXTN	..	TEXT ARGUMENT N. IF NOT REFERENCED BY THE ERROR DIAGNOSTIC, THIS NEED NOT BE SUPPLIED.
LENN	..	LENGTH IN CHARACTERS OF THE CORRESPONDING TEXT ARGUMENT. IF THIS ARGUMENT IS NOT USED, IT MAY BE OMITTED.

FOLLOWING IS AN EXCERPT FROM THE RUN.ER FILE:

```
1,ERROR OPENING .LINK FILE FOR FORM DEFINITION "%1", DEVICE "%2".
2,PROBLEM READING .LINK FILE HEADER FOR FORM DEFINITION "%1",
2,DEVICE "%2".  THIS MAY INDICATE THAT THE USER HIT CONTROL-P
2,(BREAK) WHILE ADDING THE FORM DEFINITION TO THE FORMS CATALOG.
2,THE MOST ADVISABLE COURSE OF ACTION IS TO INVOKE FAP AND ISSUE
2,A "LINK ALL" COMMAND.
3,ERROR ATTACHING TO FORMS LINKED-DEFINITION DIRECTORY (LNK.FD).
3,IF YOU HAVE NOT YET DONE SO, RUN THE "C_R15" COMMAND FILE TO
```

```
3,UPGRADE YOUR INSTALLATION TO A REV-15 VERSION OF FORMS.
4,ERROR ATTACHING TO FORMS CONTROL DIRECTORY (FORMS*). IF YOU
4,HAVE NOT YET DONE SO, INVOKE FAP AND ISSUE A CREATE COMMAND TO
4,BUILD THIS DIRECTORY.
5,FORMS COMMAND ERROR. A RECORD WAS WRITTEN TO THE TERMINAL
5,WHICH CONTAINED ONLY '##' IN THE FIRST 2 CHARACTER POSITIONS
5,(I.E. NO COMMAND WAS SPECIFIED).
```

USERS WHO WRITE THEIR OWN DEVICE DRIVERS MAY MAKE USE OF THIS ERROR HANDLING FACILITY. NUMERIC ERROR CODES (KEYS) 1-999 ARE RESERVED FOR USE BY PRIME. USERS MAY ALLOCATE ANY ERROR CODE ABOVE AND INCLUDING 1000.

8.4 SHARED FORMS LIBRARY

8.4.1 SHARED LIBRARY INITIALIZATION

IN ORDER TO USE THE FORMS RUN-TIME SYSTEM WITH THE SHARED LIBRARIES AT REV 15 AND FUTURE RELEASES, THE FOLLOWING CALL SHOULD BE PRESENT IN THE APPLICATIONS PROGRAM PRIOR TO THE FIRST INVOKE COMMAND:

```
CALL FORM$I                    (FORTRAN)

CALL 'FORM$I'.                 (COBOL)
```

THIS REQUIREMENT IS APPLICABLE TO THE 64V MODE SHARED VERSION OF FORMS AND WILL BE IGNORED FOR 64R MODE AND NON-SHARED 64V MODE; HOWEVER, FORESIGHT SUGGESTS THAT THIS CALL BE PRESENT IN ALL APPLICATIONS PROGRAMS.

8.4.2 LOADING THE SHARED LIBRARY

IT IS A SYSTEM ADMINISTRATION DECISION WHETHER OR NOT TO SUPPORT THE SHARED LIBRARIES AVAILABLE AT REV 15. IF SHARED FORMS IS SUPPORTED, SHARED COBOL, KI/DA, AND FORTRAN MUST BE SUPPORTED AS WELL. IF SHARED LIBRARIES ARE IN USE, THE FORMS SHARED LIBRARY FILE WILL BE NAMED VFORMS.

8.5 CONFIGURABLE I/O LIST

THE FORMS RUN-TIME PACKAGE CONTAINS A FIXED LENGTH BUFFER, CALLED THE I/O LIST, WHICH HOLDS THE CURRENT FORM DEFINITION. THE DEFAULT I/O LIST SIZE IS 2500 WORDS (DECIMAL). IF THE USER RUNS A PROGRAM WHICH INVOKES A FORM THAT EXCEEDS THIS CAPACITY, FORMS PRINTS THE ERROR MESSAGE:

```
REQUIRED= NNNN, AVAILABLE= 2500.
I/O LIST OVERFLOW.
```

THE USER MAY ALLOCATE A LARGER I/O LIST IN HIS (FORTRAN) PROGRAM BY INSERTING THE FOLLOWING 3 STATEMENTS:

```
      PARAMETER IOLSIZ=DESIRED_SIZE
      COMMON /IOBCM$/ IBUF(3), IOL(IOLSIZ)

      DATA IBUF /IOLSIZ, 0, 0/
```

ALL ITEMS ARE 16 BIT INTEGERS.

THE USER MAY ENLARGE THE I/O LIST SIZE IN A COBOL PROGRAM BY WRITING THE ABOVE CODE FOLLOWED BY AN 'END' STATEMENT, COMPILING IT WITH FORTRAN, AND LOADING IT AFTER HIS COBOL PROGRAM. THE BINARY MODULE CONTAINING THE REDEFINITION OF THE I/O LIST MUSI BE LOADED PRIOR TO LOADING THE FORMS LIBRARY.

THE USER MAY ALSO MODIFY THE DEFAULT BUFFER POOL SIZE BY CHANGING THE 'IOLSIZ' DECLARATION IN 'FORMS>RUN>IOLDEF' AND RE-COMPILING THE RUN-TIME SYSTEM.

THIS FEATURE IS NOT AVAILABLE WHEN USING THE 64V MODE SHARED FORMS LIBRARY AVAILABLE AT REV 15, HOWEVER THE DEFAULT SIZE FOR THE SHARED LIBRARY IS 7000 WORDS.

9 INSTALLATION

9.1 DIRECTORY INFORMATION

THE FORMS MANAGEMENT SYSTEM IS SUPPLIED IN A SINGLE DIRECTORY ON THE MASTER DISK. WITHIN THIS DIRECTORY, CALLED FORMS, ARE SUBDIRECTORIES WHICH CONTAIN SOURCES FOR THE VARIOUS COMPONENTS OF THE SYSTEM AND COMMAND FILES WHICH ARE USED TO BUILD THE SYSTEM.

HERE IS A BREAKDOWN OF EACH FILE WITHIN THE FORMS UFD:

FILE	TYPE	DESCRIPTION
FDL	SUBUFD	SOURCES FOR FDL TRANSLATOR
FAP	SUBUFD	SOURCES FOR FAP UTILITY
RUN	SUBUFD	SOURCES FOR RUN TIME PACKAGE
IOS	SUBUFD	SOURCES FOR IOCS INTERFACE, DEVICE DRIVERS
DOC	SUBUFD	SOURCE FOR THIS DOCUMENT
FORMS*	SUBUFD	SKELETON FORMS CATALOG, SYSTEM FILES
C_RLIB	COMINP	CREATES 64R LIBRARY FROM INDIV. OBJECTS
C_VLIB	COMINP	CREATES 64V LIBRARY FROM INDIV. OBJECTS
RFORMS	OBJECT	64R MODE FORMS RUN TIME SYSTEM
VFORMS	OBJECT	64V MODE FORMS RUN TIME SYSTEM
MACROS	INSERT	$INSERT FILE CONTAINING SAMPLE FDL MACRO DEF'S
C_INST	COMINP	INSTALLS NEW FORMS SYSTEM

C_R15 COMINP UPGRADES CURRENT FORMS SYSTEM TO REV 15

9.2 INSTALLING A NEW VERSION OF FORMS

TO INSTALL THE CURRENT VERSION OF FORMS ON A SYSTEM ON WHICH THERE IS NO EXISTING COPY (I.E. TO CREATE A NEW FORMS INSTALLATION):

1) USE FUTIL TO COPY THE FORMS* SUBUFD FROM THE FORMS UFD TO THE MFD ON ANY STARTED UP LOCAL DISK, THUS MAKING FORMS* A FIRST-LEVEL DIRECTORY. FORMS WILL NOT WORK PROPERLY IF THE FORMS* DIRECTORY RESIDES ON A DISK ON A REMOTE SYSTEM ACCESSED VIA PRIMENET.

2) EXECUTE THE C_INST COMMAND FILE IN THE FORMS UFD TO TO COPY THE FAP AND FDL PROGRAMS TO CMDNCO AND TO COPY THE RFORMS AND VFORMS OBJECT FILES TO THE LIBRARY UFD LIB. THE FORMS SYSTEM IS NOW READY FOR USE.

9.3 UPGRADING A CURRENT INSTALLATION

TO UPGRADE AN EXISTING FORMS INSTALLATION TO REV 15:

1) EXECUTE THE C_INST COMMAND FILE TO COPY THE NEW RUN FILES TO CMDNCO AND LIBRARIES TO LIB.

2) EXECUTE THE C_R15 COMMAND FILE. THIS WILL CREATE THE NECESSARY FILES IN THE FORMS* DIRECTORY TO SUPPORT REV 15.

NOTE THAT THIS WILL NOT AFFECT EXISTING PROGRAMS WHICH HAVE BEEN LOADED WITH THE REV 13 OR 14 LIBRARIES. THE USER IS ENCOURAGED, HOWEVER, TO RELOAD HIS PROGRAMS WITH THE NEW LIBRARIES TO TAKE ADVANTAGE OF THE PERFORMANCE IMPROVEMENTS AND ADDITIONAL FEATURES OFFERED AT THIS RELEASE.

9.4 REBUILDING FORMS

THE FOLLOWING COMMAND FILES ARE AVAILABLE TO REBUILD ALL OR PART OF THE FORMS MANAGEMENT SYSTEM:

TREE NAME		DESCRIPTION
FORMS> FDL>	C_SUBS	COMPILE FDL SUBR'S
	C_FDL	COMPILE FDL MAIN, LOAD & SAVE
	C_LOAD	LOAD & SAVE FDL
FORMS> FAP>	C_SUB	COMPILE FAP SUBR'S
	C_FAP	COMPILE, LOAD & SAVE FAP
	C_LOAD	LOAD & SAVE FAP
FORMS> RUN>	C_FMR	COMPILE 64R MODE FORMS RUN TIME PKG
	C_FMV	COMPILE 64V MODE FORMS RUN TIME PKG
FORMS> IOS>	C_IOR	COMPILE 64R MODE I/O SYSTEM
	C_IOV	COMPILE 64V MODE I/O SYSTEM
FORMS>	C_RLIB	BUILD 64R MODE LIBRARY FROM OBJECTS

C_VLIB	BUILD 64V MODE LIBRARY FROM OBJECTS
C_BLD	BUILD ENTIRE FORMS SYSTEM

ALL COMMAND FILES DESCRIBED ABOVE GENERATE NO LISTING OF THE COMPILED SOURCE. FOR EACH COMMAND FILE WHICH COMPILES SOURCE TEXT EXISTS A CORRESPONDING COMMAND FILE, WHICH IN ADDITION TO GENERATING A BINARY (OBJECT) FILE, ALSO PRODUCES A COMPILATION LISTING. THE NAMES OF THESE COMMAND FILES MAY BE DETERMINED BY CONCATENATING THE STANDARD COMMAND FILE NAME WITH THE CHARACTER 'L'. FOR EXAMPLE, TO COMPILE THE 64R MODE VERSION OF THE RUN TIME PACKAGE AND GENERATE A LISTING, ONE WOULD RUN THE COMMAND FILE C_FMRL INSTEAD OF C_FMR.

10 DEVICE INPUT/OUTPUT SYSTEM

THIS SECTION DESCRIBES THE LAYOUT AND OPERATION OF THE DEVICE INPUT/OUTPUT SYSTEM.

THE DEVICE I/O SYSTEM LOGICALLY SURROUNDS THE RUN TIME PACKAGE PROPER AND CONSISTS OF TWO PARTS. THE FIRST IS AN IOCS INTERLUDE TO ROUTE ALL TERMINAL AND LINEPRINTER I/O REQUESTS TO OR THROUGH FORMS. THE SECOND PART PERFORMS ALL DEVICE MAPPING AND INPUT/OUTPUT WITH A FORMATTED DEVICE.

10.1 IOCS INTERLUDE

THE IOCS INTERLUDE INTERFACES THE STANDARD PRIME INPUT/OUTPUT CONTROL SYSTEM TO FORMS. INCLUDED IN THIS MODULE ARE REPLACEMENTS FOR THE STANDARD READ AND WRITE ASCII TABLES (RATBL AND WATBL.) THESE TABLES CAUSE FORMS SUBROUTINES I$FM01, O$FM01, AND O$FM06 TO BE CALLED TO PROCESS TERMINAL INPUT, TERMINAL OUTPUT, AND LINEPRINTER OUTPUT, RESPECTIVELY.

INPUT AND OUTPUT IS PROCESSED BY THESE SUBROUTINES AS DESCRIBED BELOW:

- IF THE FIRST TWO CHARACTER POSITIONS IN AN OUTPUT RECORD CONTAIN TWO HASH MARKS (#), THE OUTPUT RECORD IS PASSED TO THE FORMS COMMAND INTERPRETER (FM$CMD).

- OTHERWISE, IF NO FORM IS INVOKED ON THE ASSOCIATED DEVICE, THE FORMS SUBROUTINE CALLS THE STANDARD IOCS SUBROUTINE FOR THAT DEVICE. O$AA01 IS USED FOR TERMINAL OUTPUT, I$AA12 FOR INPUT. LINE PRINTER OUTPUT IS IGNORED (WHICH IS STANDARD IN THE PRIME T/S ENVIORNMENT; ANOTHER METHOD IS USED TO WRITE FILES TO THE SPOOLED LINEPRINTER.)

- IF A FORM IS IN USE ON THE ASSOCIATED DEVICE, THE INPUT OR OUTPUT REQUEST IS TRANSFERRED TO THE FORMS SUBROUTINE FM$IN (INPUT) OR FM$OUT (OUTPUT.)

10.2 DEVICE I/O MECHANISM

THE DEVICE I/O MECHANISM IS THE "BACK END" TO THE FORMS RUN TIME PACKAGE, INTERFACING THE BODY OF THE SYSTEM TO THE FORMATTED DEVICE(S) IN USE. IT DOES THIS THROUGH A MAPPING SCHEME, DESCRIBED BELOW, AND A COLLECTION OF DEVICE DRIVER SUBROUTINES, ONE FOR EACH DEVICE SUPPORTED IN THE INSTALLATION.

10.2.1 DEVICE DEFINITION DATABASE

TWO FILES EXIST IN THE FORMS SYSTEM UFD (FORMS*) WHICH DESCRIBE THE DEVICE CONFIGURATION OF THE INSTALLATION.

10.2.1.1 DEVICE CONTROL FILE

THE DEVICE CONTROL FILE (CALLED DCF.AS) DESCRIBES EACH DEVICE IN TERMS OF A UNIQUE LOGICAL DEVICE NUMBER (LDN), DEVICE NAME, DEVICE DRIVER SUBROUTINE NAME, AND PAGE CAPACITY IN LINES AND COLUMNS.

THE LOGICAL DEVICE NUMBER, NOT TO BE CONFUSED WITH AN IOCS LOGICAL UNIT NUMBER, USED IN CONJUNCTION WITH THE DEVICE DRIVER SUBROUTINE NAME, IS USED TO DETERMINE THE EXECUTION TIME SUBROUTINE ADDRESS FOR THE DEVICE DRIVER FOR A GIVEN DEVICE NAME.

THE FORMAT FOR THE DEVICE CONTROL FILE, WHICH MAY BE MODIFIED WITH THE TEXT EDITOR, IS AS FOLLOWS:

EACH ENTRY DESCRIBES ONE DEVICE. IT CONSISTS OF A SINGLE LINE OF TEXT WITH FIVE ITEMS, EACH SEPERATED BY COMMAS:

LDN, DEVICE NAME, DEVICE DRIVER NAME, LINES, COLUMNS

- "LDN" IS THE UNIQUE LOGICAL DEVICE NUMBER ASSOCIATED WITH THE DEVICE. IT MUST BE IN THE RANGE 1-99. LOGICAL DEVICES 1-9 ARE RESERVED FOR USE BY PRIME. USER WRITTEN DEVICE DRIVERS MAY USE ANY LDN OVER 9.
- "DEVICE NAME" IS THE 1-8 CHARACTER DEVICE NAME. IT MUST CONFORM TO THE NAMING CONVENTIONS SET FORTH IN THE FDL DESCRIPTION.
- "DEVICE DRIVER NAME" IS A TWO CHARACTER ABBREVIATION OF THE NAME OF THE DEVICE DRIVER SUBROUTINE. THE FULL NAME OF THE DEVICE DRIVER SUBROUTINE IS 'XX$IO', WHERE 'XX' IS THE TWO CHARACTER ABBREVIATION.
- "LINES, COLUMNS" IS THE PAGE SIZE.

THE CONTENT OF THE DEVICE CONTROL FILE AS RELEASED IS:

```
1,PRINTER,PR,66,132
3,VISTAR3,V3,24,80
```

4,OWL1200,OW,24,80

10.2.1.2 TERMINAL CONFIGURATION FILE

ANOTHER FILE EXISTS WHICH DESCRIBES THE DEVICE NAME FOR EACH FORMS TERMINAL ON THE SYSTEM. EACH USER ON THE PRIMOS SYSTEM IS ASSIGNED A UNIQUE USER NUMBER BASED ON THE PHYSICAL LINE TO WHICH HIS TERMINAL IS CONNECTED. THIS TERMINAL CONFIGURATION FILE (TCB.BN) SPECIFIES THE TERMINAL NAME FOR EACH OF THE UP TO 64 FORMS USERS (TERMINALS) CONNECTED TO THE SYSTEM. THE FAP UTILITY IS USED TO MODIFY AND INSPECT THIS FILE.

10.2.2 DEVICE MAPPING SCHEME

THIS SECTION DESCRIBES THE MAPPING SCHEME USED BY THE RUN TIME PACKAGE.

THE FORMS RUN TIME PACKAGE HAS A ONCE-ONLY SECTION OF CODE TO PERFORM ALL INITIALIZATION. AMONG OTHER FUNCTIONS, THIS OBTAINS THE DEVICE NAMES FOR THE TERMINAL AND PRINTER.

THE TERMINAL DEVICE NAME IS OBTAINED FROM THE TERMINAL CONFIGURATION FILE, BASED ON THE USER'S SYSTEM USER # (ASSIGNED BY PRIMOS.) THE PRINTER NAME IS HARD-WIRED INTO FORMS AS "PRINTER".

ONCE THE DEVICE NAME IS KNOWN, THE LOGICAL DEVICE NUMBER IS OBTAINED FROM THE DEVICE CONTROL FILE. TWO VERSIONS OF THE DCF EXIST. "DCF.AS" IS THE ASCII (EDIT) VERSION WHICH MAY BE CHANGED AT ANY TIME BY THE USER. "DCF.BN" IS THE BINARY VERSION WHICH IS USED BY THE RUN-TIME PACKAGE. THIS FILE IS GENERATED BY FAP UPON EXECUTION OF THE GENERATE COMMAND. TWO VESIONS OF THE FILE EXIST TO INSURE THE ACTIVE COPY (DCF.BN) IS CONCURRENT WITH THE DEVICE ADDRESS TABLES, DESCRIBED BELOW.

THE LOGICAL DEVICE NUMBER IS RETAINED BY FORMS AND USED TO IDENTIFY THE DEVICE TO THE DEVICE INTERLUDE SUBROUTINE. THIS ROUTINE DISPATCHES TO APPROPRIATE DEVICE DRIVER USING A SUPPLIED LDN. THE DISPATCH OPERATION IS PERFORMED USING DEVICE ADDRESS TABLES. THESE TABLES ARE GENERATED BY FAP (USING THE GENERATE COMMAND) AND COMPILED INTO THE I/O SYSTEM. EACH TABLE ENTRY CONTAINS THE ADDRESS OF A DEVICE DRIVER. POSITION WITHIN THE TABLE CORRESPONDS TO THE LOGICAL DEVICE NUMBER.

TWO DEVICE ADDRESS TABLES EXIST, ONE FOR THE 64R MODE VERSION OF THE RUN TIME PACKAGE (CALLED "DEVDAC"), THE OTHER FOR 64V ("DEVIP"). A THIRD FILE DECLARES EACH DEVICE DRIVER EXTERNAL NAME ("DEVEXT"). ALL RESIDE IN THE FORMS* UFD.

THE CONTENT OF THE 64R DEVICE ADDRESS TABLE AS RELEASED BY PRIME IS:

```
DAC     PR$IO           LDN 1 = PRINTER
OCT     0               NO ENTRY WITH LDN 2
DAC     V3$IO           LDN 3 = VISTAR3
DAC     OW$IO           LDN 4 = OWL1200
```

10.3 USER-WRITTEN DEVICE DRIVERS

SHOULD THE USER HAVE A TERMINAL CAPABLE OF USE BY, BUT NOT SUPPORTED BY FORMS, HE MAY WRITE HIS OWN DEVICE DRIVER.

10.3.1 TERMINAL REQUIREMENTS

ANY TERMINAL TO BE USED WITH FORMS MUST HAVE THE FOLLOWING CAPABILITIES:

- INTERNALLY BUFFERED (BLOCK TRANSMISSION) MODE
- PROTECTED FIELDS
- ABSOLUTE CURSOR POSITIONING
- DATA MODIFICATION ONCE DISPLAYED
- CLEAR ENTIRE SCREEN/CLEAR UNPROTECTED DATA COMMANDS

OTHER FEATURES THAT COULD BE TAKEN ADVANTAGE OF BY THE FORMS SYSTEM OR DEVICE DRIVER INCLUDE:

- BLINK
- REVERSE VIDEO
- UNDERLINING
- KEYBOARD LOCK
- INPUT AND/OR OUTPUT SPACE COMPRESSION

10.3.2 DEVICE DRIVER SPECIFICATION

DEVICE DRIVERS MUST BE NAMED 'XX$IO', WHERE THE 'XX' REPRESENTS THE 2-CHARACTER ABBREVIATION USED IN THE DEVICE CONTROL FILE. THEY HAVE THE FOLLOWING CALLING SEQUENCE:

```
CALL XX$IO ( FUNCTION, IOLIST )
```

FUNCTION IS ONE ON THE FOLLOWING 9 FUNCTION CODES:

1 -- INITIALIZE DEVICE: RESET ALL DEVICE LOGIC, CLEAR THE ENTIRE SCREEN, AND ENTER BLOCK TRANSMISSION MODE (IF THIS IS A SOFTWARE FUNCTION).

2 -- OUTPUT INITIAL FORM: WRITE THE CONTENTS OF THE ENTIRE I/O DATA LIST (IOLIST) TO THE SCREEN. THE DEVICE DRIVER SHOULD RESET BITS 1, 2, 3, AND 4 OF THE ATTRIBUTE WORD FOR EACH ENTRY AND SET BIT 5 FOR

EACH FIELD DISPLAYED. IT SHOULD NOT DISPLAY ANY FIELDS WITH THE 'NODISPLAY' BIT (BIT 14) SET. THE LOCATION OF THE CURSOR FOLLOWING THE OUTPUT OPERATION MAY BE UNDEFINED.

3 -- INPUT FORM: FIRST, THE CURSOR SHOULD BE POSITIONED AS FOLLOWS:

- IF DEVCM$ VARIABLE XPOS IS ZERO, THE CURSOR SHOULD BE PLACED AT THE FIRST CHARACTER POSITION OF THE FIRST UNPROTECTED FIELD DISPLAYED ON THE TERMINAL. THE DEVCM$ COMMON BLOCK IS DESCRIBED LATER.
- IF XPOS IS NON-ZERO, THE CURSOR SHOULD BE POSITIONED TO LOCATION (XPOS,YPOS) ON THE DEVICE.

THE DEVICE DRIVER SHOULD WAIT FOR THE OPERATOR TO FILL IN THE DISPLAYED FORM AND PROCESS THE INPUT AS IT IS TRANSMITTED FROM THE TERMINAL. AS IT RECEIVES THE DATA, THE DRIVER IS RESPONSIBLE FOR INSERTING IT INTO DATA AREA IN EACH FIELD IN THE I/O LIST. ONLY FIELDS WITH THE 'DISPLAYED' AND 'ENABLED' BITS SET IN THE ATTRIBUTE WORD SHOULD BE INPUT. IT SHOULD BE NOTED THAT ON A FULL DUPLEX LINE, THE DEVICE DRIVER SHOULD DISABLE THE ECHO AND AUTO-LINEFEED GENERATION WITH A CALL TO DUPLX$; THIS MUST BE RESTORED AFTER THE DATA HAS BEEN INPUT. IF POSSIBLE, A BRIEF PROMPT MESSAGE SHOULD BE OUTPUT IN AN OUT-OF-THE-WAY PLACE ON THE SCREEN, INFORMING THE OPERATOR THAT THERE IS AN INPUT REQUEST PENDING.

IF A FUNCTION KEY WAS DEPRESSED, THE DEVICE DRIVER SHOULD CHECK THE LOGICAL VARIABLE FKEYS IN THE DEVCM$ COMMON BLOCK. IF FALSE, REFUSE THE FUNCTION KEY REQUEST BY WAITING FOR THE PROPER TRANSMIT KEY TO BE TYPED. IF FKEYS IS TRUE, SAVE THE FUNCTION KEY NUMBER IN THE DEVCM$ VARIABLE FKEYNO (INTEGER) AND PROCESS THE DATA AS DESCRIBED ABOVE.

4 -- MODIFY EXISTING FORM: THE DEVICE DRIVER MUST EXAMINE EACH ENTRY IN THE I/O LIST AND UPDATE THOSE FIELDS WITH ATTRIBUTE BITS 1, 2, OR 4 SET. FOLLOWING IS THE RECOMMENDED LOGIC FOR THE MODIFY PROCESSOR:

- IF DATA CHANGED, ENABLE/PROTECT CHANGED, OR FIELD ATTRIBUTE CHANGED BITS ARE ALL RESET, PROCESS NEXT FIELD, ELSE
- SAVE CURRENT ATTRIBUTE WORD IN A TEMP AND RESET BITS 1-4 (DATA/ATTRIBUTES MODIFIED) OF THE ATTRIBUTE WORD IN IOLIST, THEN
- EXTRACT FIELD LENGTH, AND X,Y COORDINATES FROM IOLIST, THEN

- IF THE FIELD IS CURRENTLY DISPLAYED AND 'NODISPLAY' BIT IS SET, ERASE THIS FIELD FROM DISPLAY, RESET BIT 5 IN IOLIST ENTRY, AND PROCESS NEXT FIELD, ELSE
- IF FIELD IS NOT CURRENTLY DISPLAYED AND 'NODISPLAY' BIT IS RESET, DISPLAY THE FIELD ACCORDING TO THE SUPPLIED ATTRIBUTES AND X,Y COORDINATES AND SET BIT 5 IN IOLIST ENTRY, ELSE
- IF 'NODISPLAY' BIT IS SET, IGNORE THIS FIELD AND PROCESS NEXT, ELSE
- IF ENABLE/PROTECT CHANGED BIT IS SET AND SPECIAL HANDLING IS REQUIRED TO ACCOMODATE THIS CHANGE, PERFORM THIS SPECIAL HANDLING; EITHER WAY,
- IF ATTRIBUTE CHANGED BIT IS SET, UPDATE THE FIELD USING THE NEW ATTRIBUTES AND PROCESS THE NEXT FIELD, ELSE
- UPDATE THE DATA AND PROCESS THE NEXT FIELD

5 -- CLEAR ENTIRE SCREEN. ALL INFORMATION DISPLAYED ON THE SCREEN SHOULD BE ERASED.

6 -- CLEAR UNPROTECTED DATA ON SCREEN. ALL UNPROTECTED INFORMATION ON THE SCREEN SHOULD BE ERASED.

7 -- CLOSE DEVICE: THIS FUNCTION CODE IS USED TO TERMINATE DEVICE USAGE AFTER A RELEASE COMMAND AND IS APPLICABLE PRIMARILY TO THE LINE-PRINTER DRIVER; TERMINAL DEVICE DRIVERS SHOULD SWITCH THE TERMINAL BACK TO CONVERSATIONAL MODE (IF SOFTWARE FUNCTION).

8 -- CORRECT DATA: THE DEVICE DRIVER MUST SCAN THE I/O LIST FOR THE FIRST FIELD WITH THE 'DATA-INVALID' ATTRIBUTE BIT SET (SEE BELOW), POSITION THE CURSOR TO THE FIRST CHARACTER POSITION OF THIS FIELD, AND ALLOW THE OPERATOR TO RE-ENTER THE DATA. IT IS RECOMMENDED THAT AN ERROR/PROMPT MESSAGE BE DISPLAYED IN AN OUT-OF-THE-WAY PLACE, INFORMING THE OPERATOR THAT THE SPECIFIED FIELD HAS FAILED ALL VALIDATION TESTS AND THAT IT MUST BE RE-ENTERED.

9 -- PRINT LOCAL: WRITE THE CONTENTS OF THE ENTIRE SCREEN TO THE LOCAL PRINTER ATTACHED TO THE TERMINAL; THIS FEATURE MUST BE SUPPORTED BY THE PARTICULAR TERMINAL HARDWARE IN USE. THE DEVICE DRIVER SHOULD RETURN TO THE CALLER WHEN THE PRINTER HAS COMPLETED PRINTING.

IOLIST IS AN ARRAY THAT CONTAINS THE CONTROL AND DATA DEFINITIONS FOR EACH FIELD IN THE FORM. IT CONTAINS 7 HEADER WORDS AND AT LEAST 1 DATA WORD FOR EACH ENTRY. THE ARRAY SHOULD BE ACCESSED BY THE DEVICE DRIVER USING A

POINTER TO THE BEGINNING OF THE FIELD (SUPPLIED BY THE RUN-TIME PACKAGE) ADDED TO AN OFFSET. THIS OFFSET SHOULD BE SPECIFIED IN THE FORM OF A PARAMETER'ED SYMBOL, AS DEFINED BELOW.

THE FOLLOWING PARAMETERS REPRESENT EACH OF THE CONTROL WORDS, PLUS THE START OF THE DATA AREA. THE DEVICE DRIVER SHOULD BE OBLIVIOUS TO THEIR ACTUAL VALUES, AS THESE MAY CHANGE WHEN NEW CONTROL INFORMATION IS ADDED. THE PARAMETER DECLARATIONS MAY BE MADE THRU A $INSERT FILE CALLED 'IOPARM' IN THE DIRECTORY CONTAINING THE SOURCE OF THE I/O SYSTEM (AS RELEASED, FORMS>IOS>IOPARM).

IOLK -- LINK TO NEXT ENTRY IN CHAIN BY POSITION; THIS IS NOT USED BY DEVICE DRIVERS

IOVP -- STREAM DEFINITION FIELD POINTER FOR THIS ENTRY; THIS IS NOT USED BY DEVICE DRIVERS

IORP -- FORMAT DEFINITION FIELD POINTER FOR THIS ENTRY; THIS IS NOT USED BY DEVICE DRIVERS

IOSZ -- FIELD LENGTH, IN CHARACTERS

IOAT -- FIELD ATTRIBUTES, AS FOLLOWS:

BIT	DEFINITION
1	SET BY FORMS IF DATA HAS CHANGED SINCE LAST DISPLAY RESET BE DEVICE DRIVER WHEN DATA HAS BEEN UPDATED ON DEVICE
2	SET BY FORMS IF ENABLE/PROTECT ATTRIBUTE HAS CHANGED SINCE LAST DISPLAY RESET BY DEVICE DRIVER WHEN FIELD HAS BEEN UPDATED ON DEVICE
3	SET BY FORMS IF FIELD HAS FAILED ALL SUPPLIED VALIDATION TESTS RESET BY DEVICE DRIVER WHEN FIELD HAS BEEN RE-ENTERED FROM DEVICE
4	SET BE FORMS IF ANY FIELD ATTRIBUTES HAS BEEN MODIFIED SINCE LAST DISPLAY RESET BY DEVICE DRIVER WHEN FIELD HAS BEEN UPDATED ON DEVICE
5	SET BY DEVICE DRIVER IF FIELD IS CURRENTLY DISPLAYED ON DEVICE RESET BY DEVICE DRIVER IF FIELD IS CURRENTLY NOT DISPLAYED ON DEVICE (INITIALLY RESET)
13	SET BY FORMS IF FIELD SHOULD BE BLINKED WHEN DISPLAYED RESET BY FORMS IF FIELD SHOULD NOT BE BLINKED WHEN DISPLAYED
14	SET BY FORMS IF FIELD SHOULD NOT BE DISPLAYED OR SHOULD BE ERASED IF CURRENTLY DISPLAYED

RESET BY FORMS IF FIELD SHOULD BE DISPLAYED

15 SET BY FORMS IF FIELD SHOULD BE DISPLAYED IN REVERSE VIDEO
RESET BY FORMS IF FIELD SHOULD BE DISPLAYED IN NORMAL VIDEO

16 SET BY FORMS IF FIELD SHOULD BE WRITE-ENABLED (NOT PROTECTED)
RESET BY FORMS IF FIELD SHOULD BE WRITE-PROTECTED

IOYX -- LINE AND COLUMN COORDINATES:
- LEFT BYTE = LINE # (Y)
- RIGHT BYTE = COLUMN # (X)

IOPG -- PHYSICAL PAGE #; THIS IS NOT USED BY DEVICE DRIVERS

IODA -- START OF TEXT DATA; DATA IS IN ASCII FORMAT, PACKED 2 CHARACTERS PER WORD, BLANK FILLED

THE INITIALIZE, CLEAR, CLOSE, AND PRINT FUNCTIONS (1, 5, 6, 7, AND 9) ARE ALL RELATIVELY STRAIGHTFORWARD. THESE OPERATIONS DO NOT HAVE TO PROCESS DATA FROM THE I/O LIST AND THEREFORE SHOULD ASSUME IT TO BE VOID.

THE OUTPUT, INPUT, MODIFY, AND CORRECT FUNCTIONS (2, 3, 4, AND 8) ALL NEED TO TRAVERSE THE I/O LIST AND PROCESS (OR AT LEAST INSPECT) EACH FIELD THEREIN. THE DEVICE DRIVER MUST DEPEND ON THE RUN-TIME SYSTEM TO PROVIDE A POINTER FOR THE START OF EACH FIELD DEFINITION IN THE I/O LIST. THE RUN-TIME PACKAGE CONTAINS 2 SUBROUTINES CALLABLE BY THE DEVICE DRIVER FOR SUCH A PURPOSE. THEY ARE:

FMS$RE - RESETS THE INTERNAL (RUN-TIME PACKAGE) FIELD POINTER TO THE BEGINNING OF THE CURRENT PAGE. THIS ROUTINE MUST BE CALLED AT THE BEGINNING OF THE OUTPUT, INPUT, MODIFY, AND CORRECT-DATA FUNCTION PROCESSORS. IT MAY BE CALLED AGAIN TO RESET THE POINTER TO THE FIRST FIELD IN THE PAGE WHEN NECESSARY (EG, ON AN INPUT ERROR).

CALLING SEQUENCE:

```
CALL FMS$RE
```

FMS$NF - RETURNS THE POINTER TO THE NEXT FIELD IN THE I/O LIST TO BE PROCESSED. IF THE POINTER IF 0, THE END OF PAGE OR END OF I/O LIST HAS BEEN ENCOUNTERED. FIELDS ARE RETURNED TO THE CALLER IN LINE/COLUMN SEQUENCE.

CALLING SEQUENCE:

```
CALL FMS$NF ( POINTER )
```

A COMMON DEFINITION INSERT FILE MUST BE INCLUDED IN THE DEVICE DRIVER BY THE DIRECTIVE "$INSERT FORMS>RUN>DEVCM$". THE COMMON BLOCK CONTAINS 4 VARIABLES WHICH ARE USED BY THE INPUT FORM (FUNCTION 3) PROCESSOR. THEY ARE:

FKEYS - LOGICAL VARIABLE SET TO TRUE IF FUNCTION KEYS ARE ENABLED, FALSE IF DISABLED. IF A FUNCTION KEY IS STRUCK AND FKEYS IS FALSE, THE FUNCTION KEY SHOULD BE IGNORED; IF TRUE, THE FUNCTION KEY NUMBER SHOULD BE STORED IN FKEYNO.

FKEYNO - 16 BIT INTEGER WHICH IS SET BY THE DEVICE DRIVER TO THE NUMBER OF THE FUNCTION KEY DEPRESSED. SHOULD ONLY BE SET IF FKEYS IS TRUE.

XPOS - 16 BIT INTEGER COLUMN NUMBER WHICH THE CURSOR IS TO BE POSITIONED PRIOR TO AN INPUT OPERATION. IF ZERO, POSITION THE CURSOR TO THE FIRST ENABLED CHARACTER POSITION ON THE DISPLAY.

YPOS - 16 BIT INTEGER LINE NUMBER FOR CURSOR POSITIONING PRIOR TO INPUT. IT IS ONLY VALID IF XPOS IS NON-ZERO.

A TEMPLATE FOR A DEVICE DRIVER IS INCLUDED WITH THE FORMS SYSTEM. IT IS SUGGESTED THAT THE USER FOLLOW THIS TEMPLATE WHEN WRITING A DEVICE DRIVER.

10.3.3 INSIALLING THE DEVICE DRIVER

TO INSTALL A NEW DEVICE DRIVER INTO THE FORMS RUN-TIME LIBRARY, THE USER SHOULD FOLLOW THE STEPS OUTLINED BELOW:

A) OBTAIN A LISTING OF THE DEVICE CONTROL FILE AND CHOOSE A FREE LOGICAL UNIT NUMBER ABOVE 10 (THE FIRST 10 ARE RESERVED BY PRIME). APPEND AN ENTRY TO THE DCF CONTAINING THE SELECTED LOGICAL UNIT NUMBER, DEVICE NAME, FIRST 2 CHARACTERS OF THE DRIVER NAME (REMEMBER, THE LAST 3 MUST BE '$IO'), AND THE DIMENSIONS OF THE DEVICE IN ACCORDANCE WITH THE FORMAT DESCRIBED IN THE SECTION ENTITLED "DEVICE MAPPING SCHEME", ABOVE. FOR EXAMPLE, THE VISTAR/3 ENTRY, WHOSE LOGICAL UNIT NUMBER IS 3, DRIVER NAME IS 'V3$IO', AND DIMENSIONS ARE 24 BY 80, WOULD LOOK AS FOLLOWS:

```
3, VISTAR3, V3, 24, 80
```

B) ATTACH TO THE DIRECTORY CONTAINING THE SOURCE FOR THE INPUT/OUTPUT SYSTEM AND COPY INTO IT THE SOURCE FOR THE DEVICE DRIVER TO BE INSTALLED.

C) EDIT THE C_IOR (64R MODE) AND C_IOV (64V MODE) COMMAND FILES AND INSERT A LINE TO COMPILE THE NEW DEVICE DRIVER

AFTER THE PR$IO ROUTINE.

D) RUN FAP AND ISSUE THE GENERATE COMMAND TO CREATE THE NEW DEVICE TABLES AND DCF WHICH WILL INCLUDE THE NEW DRIVER.

E) EXECUTE THE C_IOR AND/OR C_IOV COMMAND FILE(S) TO CREATE A NEW INPUT/OUTPUT SYSTEM.

F) ATTACH TO THE FIRST-LEVEL FORMS SOURCE DIRECTORY ('FORMS') AND EXECUTE THE COMMAND FILE 'C_RLIB' TO CREATE A 64R MODE LIBRARY AND/OR C_VLIB TO CREATE A 64V MODE LIBRARY.

THE USER MAY NOW MODIFY THE TCB ENTRIES FOR THE USERS WHICH HAVE THE NEW TERMINAL AND RELOAD HIS APPLICATIONS PROGRAM WITH THE NEW VERSION OF THE LIBRARY. IT IS STRONGLY RECOMMENDED THAT THE NEW LIBRARY NOT BE INSTALLED IN THE 'LIB' UFD UNTIL THE NEW DEVICE DRIVER HAS BEEN PROVEN TO WORK.

10.4 PRIME-SUPPLIED DEVICE DRIVERS

AT PRESENT, THE FORMS SYSTEM AS RELEASED BY PRIME SUPPORTS THE FOLLOWING THREE DEVICE DRIVERS:

. OFFLINE PRINTER	(PRINTER)
. INFOTON VISTAR/3 (MODIFIED)	(VISTAR3)
. PERKIN-ELMER OWL	(OWL1200)

10.4.1 OFFLINE PRINTER DEVICE DRIVER

THE PRINTER DEVICE DRIVER WRITES A FORM (OR FORMS) TO THE LINE-PRINTER SPOOL QUEUE. WHEN THE INVOKE COMMAND IS ISSUED TO THE LINE-PRINTER (IOCS LOGICAL UNIT 4), A FILE CALLED PR##NN (WHERE 'NN' REPRESENTS THE USER NUMBER) IS OPENED. IF IT ALREADY EXISTS, THE FILE POINTER IS POSITIONED TO THE END OF FILE, WHERE THE NEW FORM DEFINITION WILL BE WRITTEN. IF IT DOES NOT EXIST, IT IS CREATED, AFTER A RECORD IS WRITTEN CONTAINING THE CONTROL CODE FOR THE LINE-PRINTER TO ENTER FORTRAN FORMS-CONTROL MODE.

WHEN A FORM IS OUTPUT, ONE ASCII RECORD IS WRITTEN FOR EACH LINE DEFINED IN THE FORM. THE FIRST LINE CONTAINS A '1' IN COLUMN 1, WHICH CAUSES THE PRINTER TO EJECT TO THE TOP OF A NEW PAGE. ANY ENABLED FIELDS ARE UNDERSCORED (WITH THE '_' CHARACTER).

WHEN THE FORM IS RELEASED, THE FILE IS COPIED INTO THE SPOOL QUEUE, WITH THE APPROPRIATE SPOOL FILE HEADER AND FILE NAME. IT IS THEN CLOSED AND DELETED FROM THE HOME UFD. NOTE THAT THE PR##NN FILE SHOULD NEVER APPEAR IN THE HOME UFD AFTER THE PROGRAM HAS BEEN COMPLETED; IF IT DOES, IT MEANS THAT THE PRINTER FORM WAS NOT RELEASED.

BECAUSE OF THE NEW SPOOL SUBSYSTEM AT REV 13, TWO (2) VERSIONS OF

PR$IO ARE SUPPLIED. SOURCE FILE PR$IO CONTAINS THE VERSION OF THE PRINTER DRIVER THAT IS COMPAITIBLE WITH THE NEW SPOOLER. THIS IS THE SUBROUTINE THAT IS IN THE RFORMS AND VFORMS LIBRARIES AS RELEASED ON THE MASTER DISK. SOURCE FILE OPR$IO CONTAINS THE PRINTER DRIVER THAT IS COMPATIBLE WITH THE OLD (PRE REV 13) SPOOLER. TO REBUILD THE FORMS LIBRARIES TO WORK WITH THE OLD SPOOL SUBSYSTEM, RENAME PR$IO TO NPR$IO AND THEN OPR$IO TO PR$IO. THE I/O SYSTEM MAY THEN BE REBUILT WITH C_IOR AND C_IOV, AFTER WHICH THE RFORMS AND VFORMS LIBRARIES MAY BE REBUILT WITH C_RLIB AND C_VLIB, AS OUTLINED ABOVE.

10.4.2 VISTAR/3 DEVICE DRIVER

THE INFOTON VISTAR/3 DEVICE DRIVER (V3$IO) IS DESIGNED AROUND A SPECIALLY MODIFIED VISTAR/3 (WITH MICROCODE AND HARDWARE UPDATES) AVAILABLE THROUGH PRIME.

THE DEVICE DIMENSIONS ARE 24 LINES BY 80 COLUMNS (1920 CHARACTERS), ALL OF WHICH EXCEPT THE 15 CHARACTER POSITIONS IN THE LOWER RIGHT OF THE SCREEN ARE AVAILABLE FOR USE BY THE APPLICATIONS PROGRAM. THESE CHARACTER POSITIONS CONTAIN ONE OF THE FOLLOWING PROMPT OR ERROR MESSAGES FROM THE DEVICE DRIVER:

(SPACES):
    INPUT NOT ALLOWED

ENTER
    ENTER DATA INTO UNPROTECTED FIELDS ON FORM,
    DEPRESS 'XMIT PAGE' KEY WHEN DONE

ERROR, RE-ENTER (BLINKING)
    A CHARACTER WAS LOST ON THE LAST TRANSMISSION -
    DEPRESS 'XMIT PAGE' KEY

DATA ERROR (REVERSE VIDEO)
    A FIELD (OR FIELDS) DOES NOT MEET THE SPECIFIED
    VALIDATION CRITERIA - THE CURSOR IS POSITIONED TO
    THE 1ST CHARACTER POSITION OF THE OFFENDING FIELD
    CORRECT THE DATA AND DEPRESS THE 'XMIT PAGE' KEY

ALL UNPROTECTED FIELDS ARE DISPLAYED SURROUNDED BY SQUARE BRACKETS ('[' AND ']') AND ARE DISPLAYED IN FULL INTENSITY. ALL PROTECTED FIELDS ARE DISPLAYED IN HALF INTENSITY. NOTE THAT CARE MUST BE TAKEN TO ALLOW FOR THE SQUARE BRACKETS ON UNPROTECTED FIELDS WHEN DESIGNING THE FORM. THE SQUARE BRACKETS MAY BE SUPPRESSED AS AN INSTALLATION OPTION BY SETTING THE VARIABLE "ENCL" IN THE DEVICE DRIVER (FORMS>IOS>V3$IO) TO ZERO.

TO OPERATE THE VISTAR/3 WITH A PROGRAM USING FORMS, THE DIP-SWITCHES IN THE REAR OF THE DISPLAY MUST BE SET AS FOLLOWS:

EOT CHARACTER:	CR
MODE:	BLOCK
LINE-SPEED:	(USER-SELECTABLE)
SEC CHANNEL:	OFF
PARITY:	NONE
FDUX/HDUX:	(USER-SELECTABLE)
STOP BITS:	2
ROLL/PAGE:	ROLL

10.4.3 OWL DEVICE DRIVER

THE PERKIN-ELMER OWL1200 DEVICE DRIVER IS DESIGNED FOR A STOCK OWL TERMINAL AND IS CAPABLE OF SUPPORTING FUNCTION KEYS.

THE DEVICE DIMENSIONS ARE 24 LINES BY 80 COLUMNS (1920) CHARACTERS. THE FIRST CHARACTER POSITION, (1,1) AND THE LAST 6 CHARACTER POSITIONS, (75,24) THROUGH (80,24), ARE NOT AVAILABLE FOR USE BY THE USER'S FORM DEFINITION. ALSO, THE CHARACTER POSITIONS IMMEDIATELY PRECEDING AND FOLLOWING A FIELD WITH ANY ATTRIBUTE OTHER THAN "PROTECT" MUST BE VACANT.

WHEN AN INPUT OPERATION OCCURS, THE DATA ON THE SCREEN MAY BE TRANSMITTED TO THE COMPUTER BY USING ANY OF THE "SEND" KEYS ON THE RIGHT HAND KEYPAD. IF FUNCTION KEYS ARE DISABLED, STRIKING F1 WILL ALSO TRANSMIT THE SCREEN DATA. IF FUNCTION KEYS ARE ENABLED, STRIKING ANY OF THE FUNCTION KEYS WILL SEND THE DATA TO THE COMPUTER AND MAKE AVAILABLE TO THE APPLICATION PROGRAM THE NUMBER OF THE FUNCTION KEY DEPRESSED. NOTE THAT THE NUMBER OF FUNCTION KEYS MAY BE EXPANDED TWOFOLD BY USING SHIFT-FN. THIS CAUSES 16 TO BE ADDED TO THE FUNCTION KEY VALUE.

WHEN OPERATOR INPUT IS REQUIRED, ONE OF THE FOLLOWING PROMPT MESSAGE IS PRINTED IN THE LOWER RIGHT CORNER:

ENTER
: OPERATOR INPUT IS REQUIRED - DEPRESS ONE OF THE SEND OR FUNCTION KEYS WHEN DONE

DATA?
: THE DATA IN THE FIELD TO WHICH THE CURSOR IS POSITIONED DOES NOT CONFORM TO ANY OF THE VALIDATION CRITERIA SPECIFIED IN THE FORM DEFINITION - RE-ENTER THE DATA AND DEPRESS THE SEND KEY

SEQ?
: THE DATA WAS NOT TRANSMITTED FROM THE TERMINAL IN THE PROPER SEQUENCE - THIS USUALLY INDICATES THAT A CHARACTER WAS LOST DURING TRANSMISSION - DEPRESS THE APPROPRIATE SEND OR FUNCTION KEY AGAIN

SIZE?

TOO MANY CHARACTERS WERE SENT FOR A GIVEN FIELD IN THE FORM DEFINITION - AS ABOVE, THIS USUALLY INDICATES THAT A CHARACTER WAS LOST DURING TRANSMISSION - DEPRESS THE APPROPRIATE SEND OR FUNCTION KEY AGAIN

NO SPECIAL SWITCH SETTINGS ARE REQUIRED WHEN A FORMS PROGRAM IS RUN ON THE OWL.

APPENDIX A - SAMPLE FORM DEFINITION

THIS LISTING WAS PRODUCED FROM AN FDL SOURCE FILE.

```
(0001)   *   PRIMEATS, FORMS, JRW, 78/02/12
(0002)   *   PRIME AUTHORIZATION TO SHIP FORM -- FORMS DEMO
(0003)   *   COPYRIGHT 1978, PRIME COMPUTER, FRAMINGHAM MA
(0004)   *
(0005)   *
(0006)   ADMN377     STREAM
(0007)   *
(0008)   *--- ($INSERT FORMS>MACROS).
(0011)               LIST
(0012)   *
(0013)   *
(0014)   *--- HEADER INFORMATION.
(0015)   *
(0016)   HEADER      SUBSTREAM
(0017)               F (FORMNAME,FORMNAME)
(0018)               F ATSNUM, LEN 6, JUSTIFY RIGHT, ZERO-FILL, OUTPUT
(0019)               END SUBSTREAM
(0020)   *
(0021)   *
(0022)   *--- SHIP TO NAME AND ADDRESS.
(0023)   *
(0024)   NAMADR      SUBSTREAM
(0025)               F NAME, LEN 30, V 'P' OR 'B'
(0026)               REPEAT 3
(0027)               F ADDR, LEN 30
(0028)               END REPEAT
(0029)   *
(0030)               F ATTN, LEN 30
(0031)               END SUBSTREAM
(0032)   *
(0033)   *
(0034)   *--- SHIP VIA / HOW, ACCOUNTING, MISC INFO.
(0035)   *
(0036)   GENERAL     SUBSTREAM
(0037)               F SHIPVIA, LEN 1, V '9' OR 'B'
(0038)               F SHIPHOW, LEN 1, V '9' OR 'B'
(0039)   *
(0040)               F REPL, LEN 1, V 'A' OR 'B'
(0041)               F INTC, LEN 1, V 'A' OR 'B'
(0042)               F BILL, LEN 1, V 'A' OR 'B'
(0043)               F SONUM, LEN 8, V '99-99999' OR 'B'
(0044)               F CHGN, LEN 8, V '99-99999' OR 'B'
(0045)               F CPO, LEN 8, V '99-99999' OR 'B'
(0046)               F ACCOTHER, LEN 30
(0047)   *
(0048)               F AIRSPARE, LEN 1, V 'A' OR 'B'
(0049)               F INS, LEN 9, JUSTIFY RIGHT, ZERO-FILL, V 'F'
(0050)               END SUBSTREAM
```

```
(0051)    *
(0052)    *
(0053)    *--- ITEM INFORMATION.
(0054)    *
(0055)    ITEMS       SUBSTREAM
(0056)                REPEAT 4
(0057)                F PART, LEN 15, JUSTIFY RIGHT, SPACE-FILL
(0058)                F DESCR, LEN 30, JUSTIFY LEFT
(0059)                F SN, LEN 8, JUSTIFY RIGHT, ZERO-FILL
(0060)                F QTY, LEN 4, JUSTIFY RIGHT, ZERO-FILL, V '9' OR 'B'
(0061)                F RTN, LEN 1, V 'A' OR 'B'
(0062)                END REPEAT
(0063)                F MORE, LEN 1, V 'A' OR 'B'
(0064)                END SUBSTREAM
(0065)    *
(0066)    *
(0067)    *--- ERROR / WARNING MESSAGE.
(0068)    *
(0069)    ERROR       SUBSTREAM
(0070)                F ERR, LEN 40, OUTPUT
(0071)                END SUBSTREAM
(0072)    *
(0073)    *
(0074)    *
(0075)                END STREAM

0000 ERRORS (FDL, REV 15 - 16-FEB-78)
```

INPUT STREAM DESCRIPTOR STREAM: ADMN377

SUBSTREAM NAME	SUBSTREAM NUMBER	COLUMN BOUNDARIES	FIELD NAME	FIELD LENGTH
NAMADR	2	1- 30	NAME	30
NAMADR	2	31- 60	ADDR01	30
NAMADR	2	61- 90	ADDR02	30
NAMADR	2	91-120	ADDR03	30
NAMADR	2	121-150	ATTN	30
GENERAL	3	1	SHIPVIA	1
GENERAL	3	2	SHIPHOW	1
GENERAL	3	3	REPL	1
GENERAL	3	4	INTC	1
GENERAL	3	5	BILL	1
GENERAL	3	6- 13	SONUM	8
GENERAL	3	14- 21	CHGN	8
GENERAL	3	22- 29	CPO	8
GENERAL	3	30- 59	ACCOTHER	30
GENERAL	3	60	AIRSPARE	1
GENERAL	3	61- 69	INS	9
ITEMS	4	1- 15	PART01	15
ITEMS	4	16- 45	DESCR01	30
ITEMS	4	46- 53	SN01	8
ITEMS	4	54- 57	QTY01	4
ITEMS	4	58	RTN01	1
ITEMS	4	59- 73	PART02	15
ITEMS	4	74-103	DESCR02	30
ITEMS	4	104-111	SN02	8
ITEMS	4	112-115	QTY02	4
ITEMS	4	116	RTN02	1
ITEMS	4	117-131	PART03	15
ITEMS	4	132-161	DESCR03	30
ITEMS	4	162-169	SN03	8
ITEMS	4	170-173	QTY03	4
ITEMS	4	174	RTN03	1
ITEMS	4	175-189	PART04	15
ITEMS	4	190-219	DESCR04	30
ITEMS	4	220-227	SN04	8
ITEMS	4	228-231	QTY04	4
ITEMS	4	232	RTN04	1
ITEMS	4	233	MORE	1

OUTPUT STREAM DESCRIPTOR STREAM: ADMN377

SUBSTREAM NAME	SUBSTREAM NUMBER	COLUMN BOUNDARIES	FIELD NAME	FIELD LENGTH
HEADER	1	1- 6	ATSNUM	6
NAMADR	2	1- 30	NAME	30
NAMADR	2	31- 60	ADDR01	30
NAMADR	2	61- 90	ADDR02	30
NAMADR	2	91-120	ADDR03	30
NAMADR	2	121-150	ATTN	30
GENERAL	3	1	SHIPVIA	1
GENERAL	3	2	SHIPHOW	1
GENERAL	3	3	REPL	1
GENERAL	3	4	INTC	1
GENERAL	3	5	BILL	1
GENERAL	3	6- 13	SONUM	8
GENERAL	3	14- 21	CHGN	8
GENERAL	3	22- 29	CPO	8
GENERAL	3	30- 59	ACCOTHER	30
GENERAL	3	60	AIRSPARE	1
GENERAL	3	61- 69	INS	9
ITEMS	4	1- 15	PART01	15
ITEMS	4	16- 45	DESCR01	30
ITEMS	4	46- 53	SN01	8
ITEMS	4	54- 57	QTY01	4
ITEMS	4	58	RTN01	1
ITEMS	4	59- 73	PART02	15
ITEMS	4	74-103	DESCR02	30
ITEMS	4	104-111	SN02	8
ITEMS	4	112-115	QTY02	4
ITEMS	4	116	RTN02	1
ITEMS	4	117-131	PART03	15
ITEMS	4	132-161	DESCR03	30
ITEMS	4	162-169	SN03	8
ITEMS	4	170-173	QTY03	4
ITEMS	4	174	RTN03	1
ITEMS	4	175-189	PART04	15
ITEMS	4	190-219	DESCR04	30
ITEMS	4	220-227	SN04	8
ITEMS	4	228-231	QTY04	4
ITEMS	4	232	RTN04	1
ITEMS	4	233	MORE	1
ERROR	5	1- 40	ERR	40

```
(0076)   *   PRIMEATS, FORMS, JRW, 78/02/12
(0077)   *   PRIME AUTHORIZATION TO SHIP FORM -- FORMS DEMO
(0078)   *   COPYRIGHT 1978, PRIME COMPUTER, FRAMINGHAM MA
(0079)   *
(0080)   *
(0081)   ADMN377     FORMAT
(0082)               DEVICE OWL1200
(0083)   *
(0084)   *--- ($INSERT FORMS>MACROS).
(0085)   *   MACROS, FORMS>MACROS, JRW, 77/09/02
(0085)   *   MACRO DEFINITIONS FOR FORMS DEFINITION LANGUAGE TRANSLATOR
(0085)   *   COPYRIGHT 1977, PRIME COMPUTER, FRAMINGHAM MA
(0085)   *
(0085)   *
(0085)   F          DEF FIELD
(0085)   V          DEF VALIDATE
(0085)   LEN        DEF LENGTH
(0085)   POS        DEF POSITION
(0085)   IN         DEF INPUT
(0085)   OUT        DEF OUTPUT
(0085)   JUS        DEF JUSTIFY
(0085)   R          DEF RIGHT
(0085)   L          DEF LEFT
(0085)   C          DEF CENTER
(0085)   NP         DEF NOPROTECT
(0085)   RV         DEF REVERSE VIDEO
(0085)   BL         DEF BLINK
(0085)   *
(0085)   *---END FORMS>MACROS
(0086)   *
(0087)   *
(0088)   *--- HEADER LINE INFORMATION:
(0089)   *
(0090)               F 'FORM' POS (2,1)
(0091)   FORMNAME    F LEN 8, POS (7,1)
(0092)               F 'ATS #' POS (20,1)
(0093)   ATSNUM      F LEN 6, POS (26,1)
(0094)   *
(0095)   *
(0096)   *--- SHIP TO INFORMATION:
(0097)   *
(0098)               F 'SHIP TO ' POS (2,3), RV
(0099)               F 'NAME' POS (12,3)
(0100)   NAME        F LEN 30, POS (24,3), NP
(0101)               F 'ADDRESS' POS (12,4)
(0102)               REPEAT 3
(0103)   ADDR        F LEN 30, POS (24,+3), NP
(0104)               END REPEAT
(0105)               F 'ATTENTION' POS (12,7)
(0106)   ATTN        F LEN 30 POS (24,7), NP
(0107)   *
(0108)   *
(0109)   *--- SHIP VIA INFORMATION:
(0110)   *
```

```
(0111)              F 'SHIP VIA' POS (2,9), RV
(0112)   SHIPVIA    F LEN 1, POS (12,9), NP
(0113)              F '**VIA CODES**' POS (62,1)
(0114)              F '1> PICKUP' POS (62,2)
(0115)              F '2> PARCEL POST' POS (62,3)
(0116)              F '3> UPS' POS (62,4)
(0117)              F '4> FIRST CLASS' POS (62,5)
(0118)              F '5> SPEC DELIV' POS (62,6)
(0119)              F '6> TRUCK' POS (62,7)
(0120)              F '7> PRI PARCEL' POS (62,8)
(0121)              F '8> AIR FREIGHT' POS (62,9)
(0122)              F '9> FEDR EXPR' POS (62,10)
(0123)   *
(0124)   *
(0125)   *---SHIP HOW INFORMATION:
(0126)   *
(0127)              F 'SHIP HOW' POS (20,9), RV
(0128)   SHIPHOW    F LEN 1, POS (32,9), NP
(0129)              F '**HOW CODES**' POS (62,12)
(0130)              F '1> PREPAID', POS (62,13)
(0131)              F '2> C.O.D.', POS (62,14)
(0132)              F '3> PREPAID/ADD', POS (62,15)
(0133)              F '4> COLLECT', POS (62,16)
(0134)   *
(0135)   *
(0136)   *--- ACCOUNTING INFORMATION:
(0137)   *
(0138)              F 'ACCOUNT ' POS (2,11), RV
(0139)              F 'REPLACE/SHORT SHIP?' POS (12,11)
(0140)   REPL       F LENGTH 1, POS (33,11), NP
(0141)              F 'S.O. #', POS (37,11)
(0142)   SONUM      F LENGTH 8, POS (50,11), NP
(0143)              F 'INTERNAL CHARGE?' POS (12,12)
(0144)   INTC       F LEN 1, POS (33,12), NP
(0145)              F 'CHARGE #' POS (37,12)
(0146)   CHGN       F LEN 8, POS (50,12), NP
(0147)              F 'BILLABLE?', POS (12,13)
(0148)   BILL       F LEN 1, POS (33,13), NP
(0149)              F 'COST P.O. #', POS (37,13)
(0150)   CPO        F LEN 8, POS (50,13), NP
(0151)              F 'OTHER:' POS (12,14)
(0152)   ACCOTHER   F LEN 30, POS (20,14), NP
(0153)   *
(0154)   *
(0155)   *--- MISCELLANEOUS INFORMATION:
(0156)   *
(0157)              F 'MISC    ' POS (2,16), RV
(0158)              F 'INSURE FOR   $' POS (12,16)
(0159)   INS        F LEN 9, POS (28,16), NP
(0160)              F 'AIR SPARE?' POS (40,16)
(0161)   AIRSPARE   F LEN 1, POS (52,16), NP
(0162)   *
(0163)   *
(0164)   *--- ITEM DESCRIPTION.
```

```
(0165)   *
(0166)                F 'PART NO' POS (5,18)
(0167)                F 'DESCRIPTION' POS (29,18)
(0168)                F 'S/N' POS (53,18)
(0169)                F 'QTY' POS (61,18)
(0170)                F 'RTN' POS (66,18)
(0171)                REPEAT 4
(0172)   PART         F LEN 15, POS (2,+18), NP
(0173)   DESCR        F LEN 30, POS (19,+18), NP
(0174)   SN           F LEN 8, POS (51,+18), NP
(0175)   QTY          F LEN 4, POS (61,+18), NP
(0176)   RTN          F LEN 1, POS (67,+18), NP
(0177)                END REPEAT
(0178)   *
(0179)                F 'MORE?' POS (72,22)
(0180)   MORE         F LEN 1, POS (79,22), NP
(0181)   *
(0182)   *
(0183)   *--- ERROR MESSAGE, ETC.
(0184)   *
(0185)   ERR          F LEN 40, POS (2,24)
(0186)   *
(0187)   *
(0188)   *
(0189)                END DEVICE
(0190)                END FORMAT
```

D E V I C E M A P FORMAT: ADMN377, DEVICE: OWL1200, SIZE: 24 BY 80, PAGE: 1

```
....*....1....*....2....*....3....*....4....*....5....*....6....*....7....*....8
< FORM ********    ATS # ******                                  **VIA CODES**  <
<                                                                 1> PICKUP      <
< SHIP TO   NAME  _____________________________                   2> PARCEL POST <
<           ADDRESS _____________________________                 3> UPS         <
<                   _____________________________                 4> FIRST CLASS <
<                   _____________________________                 5> SPEC DELIV  <
<           ATTENTION _____________________________               6> TRUCK       <
<                                                                 7> PRI PARCEL  <
< SHIP VIA _      SHIP HOW   _                                    8> AIR FREIGHT <
<                                                                 9> FEDR EXPR   <
< ACCOUNT   REPLACE/SHORT SHIP? _   S.O. #      _______                         <
<           INTERNAL CHARGE?    _   CHARGE #    _______   **HOW CODES**        <
<           BILLABLE?           _   COST P.O. # _______   1> PREPAID           <
<           OTHER: _____________________________          2> C.O.D.           <
<                                                         3> PREPAID/ADD       <
< MISC      INSURE FOR  $ _______  AIR SPARE? _           4> COLLECT           <
<                                                                              <
<   PART NO                 DESCRIPTION              S/N      QTY  RTN         <
< ______________ ____________________________ _______ ____ _                  <
< ______________ ____________________________ _______ ____ _                  <
< ______________ ____________________________ _______ ____ _                  <
< ______________ ____________________________ _______ ____ _      MORE? _     <
<                                                                              <
< ****************************************                                     <
```

APPENDIX B - SAMPLE FORMS PROGRAM

THIS FORTRAN PROGRAM IS USED WITH THE FORM DEFINITION SHOWN IN THE PREVIOUS SECTION.

```
C   ATSINP, FORMS, JRW, 78/02/23
C   FORMS DEMO PROGRAM - INPUT ATS INFO, STORE IN DISK FILE
C   COPYRIGHT 1978, PRIME COMPUTER INC, FRAMINGHAM
C
C
C--- THIS PROGRAM INPUTS ATS INFORMATION FROM THE TERMINAL AND
C    STORES THE INFO IN A DISK FILE.
C
C    TWO FILES ARE USED:
C
C       ATS.C   IS THE CONTROL FILE - IT CONTAINS THE NEXT ATS NUMBER TO BE
C               ASSIGNED.
C       ATS.D   IS THE DATA FILE.  AS EACH ATS FORM IS ENTERED, IT IS APPENDED
C               TO THIS FILE IN THE FORMAT SHOWN IN THE PROGRAM.
C
C
C--- TO TERMINATE THE PROGRAM, ENTER A NULL NAME FIELD.
C
C--- TO ENTER MORE THAN 4 ITEM LINES, ENTER A NON-SPACE CHARACTER (EXCEPT N)
C    IN THE 'MORE' FIELD.
C
C--- THIS PROGRAM MAY BE USED BY A SINGLE USER IN ANY GIVEN DIRECTORY AT
C    ONE TIME.  NO PROVISION IS MADE FOR CONCURRENT ACCESS TO THE DATA FILES.
C
C
C
      COMMON /F$IOBF/ B(150)                    /* EXTENDED I/O BUFFER
C
      INTEGER NAMADR(75), VIA, HOW, REPL, INTC, BILL, SONUM(4),
     +        CHGNUM(4), CPO(4), ACOTHR(15), AIRSPR, INS(5),
     +        TYPE, CODE, NWIO, ATSNUM, B, I, J, MORE, FLD1(4,4),
     +        YESNOB(4), ACTBUF(4,3), FLD2(4,3)
C
      INTEGER PART(8,4), DESCR(15,4), SN(4,4), QTY(4), RTN(4)
C
      LOGICAL HDROUT
C
C
      EQUIVALENCE (YESNOB(1),REPL), (YESNOB(2),INTC),
     +            (YESNOB(3),BILL), (YESNOB(4),AIRSPR),
     +            (ACTBUF(1,1),SONUM), (ACTBUF(1,2),CHGNUM),
     +            (ACTBUF(1,3),CPO)
C
C
$INSERT SYSCOM>KEYS.F
C
```

```
C
      DATA FLD1 /'REPL     ','INTC     ','BILL     ','AIRSPARE'/
      DATA FLD2 /'SONUM    ','CHGN     ','CPO      '/
C
C
C---EXTEND TERMINAL, FILE I/O BUFFERS.
C
      CALL ATTDEV(1,1,0,150)
      CALL ATTDEV(6,7,2,150)
C
C
C---OPEN FILES, INVOKE FORM ON TERMINAL.
C
      CALL SRCH$$(K$CLOS,0,0,1,0,CODE)
      CALL SRCH$$(K$CLOS,0,0,2,0,CODE)
C
      CALL SRCH$$(K$RDWR,'ATS.C',5,1,TYPE,CODE)
      IF (CODE.NE.0) CALL ERRPR$(K$NRTN,CODE,'ATS.C',5,0,0)
      CALL SRCH$$(K$RDWR,'ATS.D',5,2,TYPE,CODE)
      IF (CODE.NE.0) CALL ERRPR$(K$NRTN,CODE,'ATS.D',5,0,0)
C
      CALL PRWF$$(K$POSN+K$PRER,2,LOC(0),0,10000000,NWIO,CODE)
C
      WRITE (1,20)
20    FORMAT('##INVOKE ADMN377')
C
C
C---ASSIGN NEXT ATS #.
C
100   CALL PRWF$$(K$POSN+K$PREA,1,LOC(0),0,000000,NWIO,CODE)
      READ (5,120,ERR=160,END=160) ATSNUM
120   FORMAT(I6)
      GO TO 200
C
C
C---HERE ON EOF, ETC.
C
160   ATSNUM=0
C
C
C---ASSIGN NEXT SEQUENTIAL ATS #.
C
180   ATSNUM=ATSNUM+1
C
C
C---WRITE ATS#, CLEAR VARIABLE DATA, ERROR MESSAGE.
C
200   WRITE (1,210) ATSNUM
      HDROUT=.FALSE.                          /* HEADER NOT OUTPUT TO DISK FILE
210   FORMAT('##SUBSTREAM HEADER'/I6/'##CLEAR'/'##SUBSTREAM ERROR'/' ')
C
C
C---READ IN NAMES, ADDRESSES, AND ACCOUNTING INFORMATION.
C
```

```
220     READ (1,240) NAMADR, VIA, HOW, REPL, INTC, BILL, SONUM,
     +             CHGNUM, CPO, ACOTHR, AIRSPR, INS
C
        IF(NAMADR(1).EQ.'  ') GO TO 5000      /* BLANK NAME => EXIT
240     FORMAT(75A2/2I1,3A1,12A2,15A2,A1,4A2,A1)
C
C
C---VALIDATE INPUT DATA.
C
        IF (VIA.LT.1.OR.VIA.GT.9) GO TO 1000
        IF (HOW.LT.1.OR.HOW.GT.4) GO TO 1020
C
C
C---CHECK YES/NO RESPONSES.
C
        DO 250 I=1,4
        IF (YESNOB(I).GE.'A'.AND.YESNOB(I).LE.'Z') /* MAP => UPPER CASE
     +    YESNOB(I)=AND(YESNOB(I),:157777)
        IF (YESNOB(I).NE.'Y'.AND.YESNOB(I).NE.'N') GO TO 1040
        IF (I.EQ.4) GO TO 250
        IF (YESNOB(I).EQ.'Y'.AND.ACTBUF(1,I).EQ.'  ') GO TO 1060
        IF (YESNOB(I).EQ.'N'.AND.ACTBUF(1,I).NE.'  ') GO TO 1080
250     CONTINUE
C
C
C---GET ITEM DATA.
C
400     READ (1,420) ((PART(J,I),J=1,8), (DESCR(J,I),J=1,15),
     +                (SN(J,I),J=1,4), QTY(I), RTN(I), I=1,4), MORE
420     FORMAT(4(7A2,A1,15A2,4A2,I4,A1),A1)
C
C
C---CHECK INPUT DATA VALIDITY.
C
500     DO 520 I=1,4
        IF (DESCR(1,I).EQ.'  ') GO TO 520     /* IGNORE BLANK LINE
        IF (RTN(I).GE.'A'.AND.RTN(I).LE.'Z') /* MAP => UPPER CASE
     +    RTN(I)=AND(RTN(I),:157777)
        IF (RTN(I).NE.'Y'.AND.RTN(I).NE.'N') GO TO 1100
520     CONTINUE
C
C
C---WRITE DATA TO DISK FILE.
C
        DO 550 I=1,4
        IF (DESCR(1,I).EQ.'  ') GO TO 550     /* IGNORE BLANK LINE
        IF (HDROUT) GO TO 540
C
C
C---WRITE ACCOUNT HEADER TO DISK FILE.
C
        WRITE (6,525) ATSNUM, NAMADR, VIA, HOW, REPL, INTC, BILL, SONUM,
     +                CHGNUM, CPO, ACOTHR, AIRSPR, INS
525     FORMAT('*ATS',I6/5(15A2/),2I1,3A1,12A2/15A2/A1,4A2,A1)
```

```
C
      HDROUT=.TRUE.
C
C
C---WRITE INDIVIDUAL ITEM LINE TO DISK FILE.
C
C
540   WRITE (6,545) (PART(J,I),J=1,8), (DESCR(J,I),J=1,15),
     +              (SN(J,I),J=1,4), QTY(I), RTN(I)
545   FORMAT(7A2,A1,15A2,4A2,I4,A1)
C
550   CONTINUE
C
C
C---CHECK FOR MORE ITEM LINES.
C
      IF (MORE.EQ.' '.OR.MORE.EQ.'N'.OR.MORE.EQ.'N') GO TO 180
C
      WRITE (1,580)
580   FORMAT('##SUBSTREAM ITEMS'/' '/'##SUBSTREAM ITEMS'/
     +       '##POSITION PART01')
      GO TO 400                              /* NEXT SET OF ITEM LINES
C
C
C---INCORRECT DATA IN ACCOUNTING FIELDS.
C
1000  WRITE (1,1010)
1010  FORMAT('##SUBSTREAM ERROR'/
     +       'VIA CODE MUST BE 1-9'/
     +       '##POSITION SHIPVIA')
      GO TO 220
C
1020  WRITE (1,1030)
1030  FORMAT('##SUBSTREAM ERROR'/
     +       'HOW CODE MUST BE 1-4'/
     +       '##POSITION SHIPHOW')
      GO TO 220
C
C
C---YES/NO ANSWER REQ'D.
C
1040  WRITE (1,1050) (FLD1(J,I), J=1,4)
1050  FORMAT('##SUBSTREAM ERROR'/
     +       'YES/NO (Y OR N) RESPONSE REQUIRED'/
     +       '##POSITION ',4A2)
      GO TO 220
C
C
C---ACCUNT NUMBER FIELD BLANK.
C
1060  WRITE (1,1070) (FLD2(J,I), J=1,4)
1070  FORMAT('##SUBSTREAM ERROR'/
     +       'ACCOUNT # REQUIRED FOR YES RESPONSE'/
     +       '##POSITION ',4A2)
```

```
        GO TO 220
C
C
C---SURPLUS ACCOUNT NUMBER.
C
1080    WRITE (1,1090) (FLD2(J,I), J=1,4)
1090    FORMAT('##SUBSTREAM ERROR'/
     +          'ACCOUNT # NOT PERMITTED FOR NO RESPONSE'/
     +          '##POSITION ',4A2)
        GO TO 220
C
C
C---RETURN CODE FIELD BLANK.
C
1100    WRITE (1,1110) I
1110    FORMAT('##SUBSTREAM ERROR'/
     +          'YES/NO (Y OR N) RESPONSE REQUIRED'/
     +          '##POSITION RTN',B'##'/
     +          '##SUBSTREAM ITEMS')
        GO TO 400
C
C
C---HERE TO EXIT.  UPDATE ATS # IN CONTROL FILE.
C
5000    CALL PRWF$$(K$POSN+K$PREA,1,LOC(0),0,000000,NWIO,CODE)
        WRITE(5,120) ATSNUM
        CALL PRWF$$(K$TRNC,1,LOC(0),0,000000,NWIO,CODE)
C
        CALL SRCH$$(K$CLOS,0,0,1,0,CODE)
        CALL SRCH$$(K$CLOS,0,0,2,0,CODE)
C
        WRITE (1,5020)
5020    FORMAT('##CLEAR ALL'/'##RELEASE')
C
        CALL EXIT
C
C
C
        END
```

APPENDIX C - FORMS SYSTEM DIRECTORY CONTENTS

THIS SECTION BRIEFLY DESCRIBES THE FILES CONTAINED IN THE FORMS SYSTEM UFD (FORMS*).

FMS.** SEGMENT DIRECTORY WHICH CONTAINS THE FORM DEFINITION CATALOG AND EACH INDIVIDUAL STREAM AND FORMAT DESCRIPTOR. A FURTHER BREAKDOWN BY SEGMENT FOLLOWS:

(0)	MODULE NAME FILE (MNF). CONTAINS NAME AND TYPE OF EACH FORM DEFINITION.
(1)	MODULE INFORMATION FILE. SUPPLEMENTAL INFORMATION FOR EACH FORM DEFINITION NAMED IN THE MNF. CONTAINS CREATION, LAST MODIFY DATES, OWNER NAME, DEVICE TYPE IF FORMAT DESCRIPTOR, AND THE SEGMENT DIRECTORY ENTRY NUMBER OF THE FILE CONTAINING THE ACTUAL FORM DEFINITION.
(2)	RESERVED.
(3-N)	FORM DEFINITION FILES.

DCF.AS DEVICE CONTROL FILE. THIS ASCII FILE CONTAINS A DESCRIPTION FOR EACH DEVICE IN USE IN THE FORMS INSTALLATION. THE FILE FORMAT IS FOUND EARLIER IN THIS DOCUMENT.

DCF.BN DEVICE CONTROL FILE. THIS BINARY FILE IS GENERATED BY THE FAP PROGRAM UPON ISSUANCE OF THE GENERATE COMMAND. IT IS USED TO INSURE CONCURRENCE BETWEEN THE DEVICE DRIVER DISPATCH TABLES (ALSO GENERATED BY THE GENERATE COMMAND) AND THE DCF.

RUN.ER ERROR DEFINITION FILE. THIS FILE CONTAINS THE TEXT MESSAGE FOR EACH RUN TIME ERROR DIAGNOSTIC.

LNK.FD THIS UFD CONTAINS LINKED FORM DEFINITIONS. FILE NAMES ARE CONSTRUCTION BY CONCATENATING 'L.' WITH A HASH OF THE FORM DESCRIPTOR NAME. THE CONTENTS OF THIS DIRECTORY ARE USUALLY INVISIBLE TO THE USER.

TCB.BN TERMINAL CONFIGURATION FILE. THIS FILE CONTAINS A TABLE WHICH MAPS EACH USER NUMBER TO A TERMINAL TYPE. FOR A DESCRIPTION, SEE THE SECTION ENTITLED "DEVICE I/O SYSTEM".

DEVEXT EXTERNAL DECLARATIONS FOR DEVICE DRIVER DISPATCH TABLES.

DEVDAC 64R MODE DEVICE DRIVER DISPATCH TABLE.

DEVIP 64V MODE DEVICE DRIVER DISPATCH TABLE.

APPENDIX D - TEMPLATE DEVICE DRIVER

FOLLOWING IS A LISTING OF THE TEMPLATE FORTRAN SOURCE FILE WHICH MAY BE USED TO WRITE A DEVICE DRIVER.

```
C   TEMPLATE, FORMS>IOS, JRW, 78/02/16                                -REV 15-
C   FORMS DEVICE DRIVER TEMPLATE
C   COPYRIGHT 1976, PRIME COMPUTER, FRAMINGHAM MA
C
C
C
      SUBROUTINE XX$IO(FUNC,IOLST)
C
C
C--- THIS MODULE IS INTENDED FOR USE AS A TEMPLATE TO AID THE USER
C    IN CONSTRUCTING A FORMS DEVICE DRIVER.  USER-SUPPLIED CODE IS
C    REQUIRED WHERE SO NOTED.
C
C
C   ARGUMENTS:   FUNC    =  I/O FUNCTION (SEE FORMS DOC'NT FOR DESCRIPTIONS)
C                IOLST   =  FIELD DEFINITION TABLE
C
C
$INSERT IOPARM
$INSERT FORMS>RUN>DEVCM$
C
      INTEGER FUNC, IOLST(1), L, X, Y, M, PTR
C
C
C---DISPATCH TO APPROPRIATE FUNCTION HANDLER.
C
      GO TO (1000,2000,3000,4000,5000,6000,7000,8000,9000), FUNC
      RETURN                                    /* IGNORE ILLEGAL CALL
C
C
C---INITIALIZE DEVICE.
C
C--> THE USER SHOULD INSERT DEVICE INITIALIZATION CODE HERE.  THIS SHOULD
C    INCLUDE CLEARING THE SCREEN AND ENTERING BLOCK TRANSMISSION MODE
C    (IF AVAILABLE THRU SOFTWARE).
C
1000  *******
      RETURN
C
C
C---OUTPUT INITIAL FORM.
C
2000  CALL FMS$RE                               /* RESET I/O LIST POINTER
C
C
C---HERE TO PROCESS NEXT FIELD
```

```
C
2010    CALL FMS$NF(PTR)                              /* PICK UP POINTER TO NEXT FLD
        IF(PTR.EQ.0) GO TO 2020                       /* END OF CHAIN
        L=IOLST(PTR+IOSZ)                             /* FIELD LENGTH
        X=RT(IOLST(PTR+IOYX),8)                       /* X,Y COORDINATES
        Y=RS(IOLST(PTR+IOYX),8)
        M=IOLST(PTR+IOAT)                             /* ATTRIBUTES
        IOLST(PTR+IOAT)=RT(IOLST(PTR+IOAT),4) /* STRIP OFF UPDATE BITS
        IF(AND(M,4).NE.0) GO TO 2010                  /* NODISPLAY OPTION - IGNORE FIELD
        IOLST(PTR+IOAT)=IOLST(PTR+IOAT)+:4000 /* MARK AS 'DISPLAYED'
C
C
C--> INSERT OUTPUT-FIELD CODE HERE.
C
C     THE FOLLOWING INFORMATION IS AVAILABLE:
C
C       L                 =  FIELD LENGTH, IN CHARACTERS
C       M                 =  FIELD ATTRIBUTES
C       X,Y               =  COLUMN, ROW COORDINATES
C       IOLST(PTR+IODA)   =  (L) CHARACTERS OF TEXT DATA
C
        ******
        GO TO 2010                                    /* PROCESS NEXT FIELD
C
C
C--> INSERT ANY CODE NECESSARY TO CLEAN UP AFTER AN OUTPUT OPERATION.
C
2020    ******
        RETURN
C
C
C---INPUT FORM FROM TERMINAL.
C
3000    CALL FMS$RE                                   /* RESET POINTER
C
C
C--> NOTE: IF TERMINALS IN THIS INSTALLATION RUN IN HALF-DUPLEX,
C           REMOVE THE FOLLOWING LINE:
C
        CALL DUPLX$(:140000)                          /* INHIBIT DUPLEX, AUTO-LF
C
C
C---CHECK SPECIAL CURSOR POSITIONING.
C
        IF(XPOS.EQ.0) GO TO 3020
C
C
C--> INSERT CODE HERE TO POSITION THE CURSOR TO LOCATION (XPOS,YPOS)
C
        ******
        GO TO 3200
C
C
C--> INSERT CODE HERE TO POSITION THE CURSOR TO THE FIRST ENABLED CHARACTER
```

```
C   POSITION ON THE TERMINAL.
C
3020    ******
C
C
C---HERE TO INPUT NEXT FIELD FROM CRT.
C
3200    CALL FMS$NF(PTR)
        IF(PTR.EQ.0) GO TO 3800                 /* END OF PAGE
        IOLST(PTR+IOAT)=AND(IOLST(PTR+IOAT),:7777) /* STRIP UPDATE BITS
C
C
C---INSURE FIELD IS DISPLAYED AND WRITE-ENABLED.
C
        IF(AND(IOLST(PTR+IOAT),:4001).NE.:4001) GO TO 3200
        L=IOLST(PTR+IOSZ)                       /* FIELD LENGTH
C
C
C--> BECAUSE OF THE WIDE VARIETY OF TERMINALS AND EVEN MORE METHODS OF
C    BLOCK TRANSMISSION, IT WOULD BE FRUITLESS TO TRY TO ANTICIPATE ANY MORE
C    THAN WE ALREADY HAVE REGARDING INPUT, SO...
C
C--> INSERT CODE HERE THAT WILL INPUT THE CURRENT FIELD FROM THE TERMINAL.
C    IT SHOULD, AS MUCH AS POSSIBLE, CHECK FOR AND CORRECT CHARACTER-LOSS
C    ERRORS.
C
C    IF A FUNCTION KEY IS INPUT AND LOGICAL VARIABLE FKEYS IS SET TO TRUE,
C    THE INTEGER VARIABLE FKEYNO SHOULD BE SET TO THE NUMBER OF THE FUNCTION
C    KEY AND THE DATA READ FROM THE DEVICE AS IF A NORMAL TRANSMIT KEY
C    WAS HIT.
C
C    (L) CHARACTERS OF INPUT DATA SHOULD BE INSERTED INTO THE I/O LIST
C    STARTING AT IOLST(PTR+IODA).  IF THIS DEVICE SUPPORTS INPUT DATA
C    COMPRESSION, THE DATA MUST BE EXPANDED ACCORDINGLY.
C
3400    ******
        GO TO 3200                              /* PROCESS NEXT FIELD
C
C
C---HERE WHEN FORM INPUT IS COMPLETED.
C
3800    CONTINUE
C
C
C--> INSERT ANY CODE NECESSARY TO 'CLEAN UP' BEFORE RETURNING TO THE CALLER.
C    IF TERMINALS IN THIS INSTALLATION RUN IN HALF-DUPLEX, REMOVE THE FOLLOWING
C    LINE OF CODE:
C
        CALL DUPLX$(0)                          /* RESTORE ECHO, AUTO-LF
        ******
C
        RETURN
C
C
```

```
C---MODIFY FORM PROCESSOR.
C
4000    CALL FMS$RE                              /* RESET POINTER
C
C
C---HERE TO PICK UP NEXT FIELD.
C
4005    CALL FMS$NF(PTR)
        IF(PTR.EQ.0) GO TO 4800
C
C
C---EXTRACT ATTRIBUTES.
C
        L=IOLST(PTR+IOSZ)                        /* FIELD LENGTH
        M=IOLST(PTR+IOAT)                        /* ATTRIBUTES
        IF(AND(M,:150000).EQ.0) GO TO 4005       /* NOT MODIFIED
        X=RT(IOLST(PTR+IOYX),8)                  /* X,Y COORDINATES
        Y=RS(IOLST(PTR+IOYX),8)
        IOLST(PTR+IOAT)=AND(IOLST(PTR+IOAT),:7777) /* STRIP MODIFY BITS
        IF(AND(M,:4004).EQ.:4004) GO TO 4200 /* DISPLAYED - ERASE
        IF(AND(M,:4004).EQ.0) GO TO 4300     /* NOT DISPLAYED - DISPLAY
        IF(AND(M,4).NE.0) GO TO 4005         /* 'NODISPLAY' FLAG SET
C
        IF(AND(M,:40000).EQ.0) GO TO 4100    /* ENABLE/PROTECT UNCHANGED
C
C
C---FALL THRU HERE IF ENABLE/PROTECT STATUS HAS CHANGED ON CURRENT FIELD.
C
C--> INSERT CODE HERE IF SPECIAL HANDLING IS REQUIRED WHEN ENABLE/PROTECT
C    STATUS CHANGES ON A FIELD.  IF NO SPECIAL HANDLING IS REQUIRED, JOIN
C    PROCESSING AT STMT # 4100.
C
        ******
        GO TO 4150                               /* SEE IF DATA HAS CHANGED
C
C
C---HERE TO CHECK IF ANY ATTRIBUTES HAVE CHANGED SINCE LAST DISPLAY:
C
4100    IF(AND(M,:10000).EQ.0) GO TO 4150    /* CHECK DATA MODIFIED
C
C
C--> INSERT CODE HERE TO MODIFY THE ATTRIBUTES OF THE DISPLAYED DATA.
C
        ******
C
C
C---SEE IF DATA HAS CHANGED SINCE LAST DISPLAY.
C
4150    IF(M.GE.0) GO TO 4005                    /* NOT CHANGED, PROCESS NEXT FIELD
C
C
C--> INSERT CODE HERE TO DISPLAY THE NEW FIELD DATA.
C
        ******
```

```
      GO TO 4005                                /* PROCESS NEXT FIELD
C
C
C---HERE IF FIELD WAS PREVIOUSLY DISPLAYED AND NOW HAS 'NODISPLAY' OPTION
C   BIT SET - ERASE IT FROM THE DISPLAY.
C
4200  IOLST(PTR+IOAT)=AND(IOLST(PTR+IOAT),:3777) /* MASK OFF DISPLAY BIT
C
C
C--> INSERT CODE TO ERASE THE CURRENT FIELD FROM THE DISPLAY.
C
      ******
      GO TO 4005                                /* PROCESS NEXT FIELD
C
C
C---HERE IF FIELD PREVIOUSLY WAS NOT DISPLAYED AND NOW HAS 'NODISPLAY' BIT
C   RESET - DISPLAY THE FIELD.
C
4300  IOLST(PTR+IOAT)=OR(IOLST(PTR+IOAT),:4000) /* SET 'DISPLAY' BIT
C
C
C--> INSERT CODE HERE TO DISPLAY A PREVIOUSLY NON-DISPLAYED FIELD.
C
      ******
      GO TO 4005
C
C
C---HERE WHEN MODIFY OPERATION COMPLETE.
C
C--> INSERT ANY CODE HERE NECESSARY TO 'CLEAN UP' AFTER A MODIFY OPERATION.
C
4800  ******
      RETURN
C
C
C---CLEAR ENTIRE SCREEN.
C
C--> INSERT CODE HERE TO CLEAR THE SCREEN.
C
5000  ******
      RETURN
C
C
C---CLEAR UNPROTECTED SCREEN.
C
C--> INSERT CODE TO CLEAR ONLY THE UNPROTECTED AREAS OF THE SCREEN.
C
6000  ******
      RETURN
C
C
C---RELEASE DEVICE.
C
C--> THIS CALL IS USUALLY IGNORED, BUT FEEL FREE TO INSERT ANY CODE HERE
```

```
C     THAT WILL LEAVE THE TERMINAL IN A 'HUMAN' STATE.
C
7000    ******
        RETURN
C
C
C---CORRECT DATA ERROR.  SCAN THE I/O LIST FOR THE FIRST ENTRY WITH THE
C    'DATA ERROR' BIT (BIT 3) SET AND POSITION THE CURSOR TO THAT FIELD.
C
8000    CALL FMS$RE                                 /* RESET POINTER
C
C
C---HERE TO CHECK NEXT FIELD
C
8100    CALL FMS$NF(PTR)                            /* GET NEXT FIELD POINTER
        IF(PTR.EQ.0) STOP                           /* SHOULD --NEVER-- GET HERE!!
        IF(AND(IOLST(PTR+IOAT),:20000).EQ.0) GO TO 8100 /* NOT THIS ONE!
C
C
C---FALL THRU HERE WHEN WE'VE FOUND A FIELD WITH THE DATA ERROR BIT SET.
C
        IOLST(PTR+IOAT)=AND(IOLST(PTR+IOAT),:157777) /* REMOVE DATA ERROR BIT
        X=RT(IOLST(PTR+IOYX),8)                     /* GET X,Y COORDINATES
        Y=RS(IOLST(PTR+IOYX),8)
C
C
C--> INSERT CODE HERE THAT WILL POSITION THE CURSOR TO THE GIVEN X,Y
C    COORDINATES.  THEN, RE-ENTER THE DATA FROM EITHER THE SPECIFIC FIELD
C    IN ERROR OR THE ENTIRE FORM.  ALL THAT THIS MAY REQUIRE IS A
C    JUMP INTO THE INPUT-FORM PROCESSOR.  FUNCTION KEYS SHOULD BE IGNORED
C    AND THE VALUE OF FKEYNO UNDISTURBED.
C
        ******
C
C
C---HERE TO PRINT THE FORM ON THE LOCAL PRINTER.
C
C--> INSERT CODE TO OUTPUT THE CURRENT SCREEN CONTENTS TO THE LOCAL
C    PRINTER.  IF THIS OPTION IS NOT SUPPORTED, SIMPLY RETURN TO THE CALLER.
C
9000    ******
C
C
C
        END
```

DATE: APRIL 3, L978

SUBJECT: FTN REV 15

FTN IN REV.15 HAS BEEN UPGRADED IN TWO MAJOR AREAS: IT SUPPORTS IBM FORTRAN-H COMPATIBLE DIRECT-ACCESS I/O STATEMENTS, AND GENERALIZED SUBSCRIPTS IN ARRAY EXPRESSIONS. IT ALSO ALLOWS CERTAIN PROGRAMS TO HAVE FASTER COMMON BLOCK ACCESS IN 64V MODE. THIS DOCUMENT DESCRIBES THESE FEATURES.

1 FORTRAN DIRECT ACCESS CAPABILITY

1.1 INTRODUCTION

THE FORTRAN COMPILER AND RUN-TIME LIBRARY NOW SUPPORT DIRECT ACCESS READ AND WRITE STATEMENTS. READ AND WRITE STATEMENTS MAY CONTAIN A RECORD NUMBER SO THAT THE RECORDS OF A FILE CAN BE ACCESSED IN RANDOM ORDER. THIS CONTRASTS WITH SEQUENTIAL ACCESS IN WHICH RECORD # N-1 MUST BE READ OR WRITTEN BEFORE RECORD # N. THE SYNTAX IMPLEMENTED INSURES COMPATABILITY WITH BOTH IBM FORTRAN AND NEW ANSI STANDARD FORTRAN. PREVIOUSLY THE DIRECT ACCESS CAPABILITY WAS SUPPORTED BY INSERTING CALLS TO THE POSFIL SUBROUTINE PRIOR TO READ AND WRITE STATEMENTS.

1.2 DIRECT ACCESS READ AND WRITE STATEMENTS

THE SYNTAX OF THE DIRECT ACCESS READ AND WRITE STATEMENTS IS:

```
READ(U'R,F,ERR=S) I/O LIST

READ(U,F,REC=R,ERR=S) I/O LIST

WRITE(U'R,F,ERR=S) I/O LIST

WRITE(U,F,REC=R,ERR=S) I/O LIST
```

U - IS A LONG OR SHORT INTEGER CONSTANT OR VARIABLE WHOSE VALUE IS THE UNIT NUMBER

R - IS THE LONG OR SHORT INTEGER EXPRESSION WHOSE VALUE IS THE RECORD NUMBER

F - IS THE OPTIONAL FORMAT SPECIFIER

S - IS THE OPTIONAL ERROR SPECIFIER

NOTICE THAT THE END= SPECIFIER IS NOT ALLOWED IN THE DIRECT ACCESS READ STATEMENT. THIS RESTRICTION IS CONSISTENT WITH BOTH IBM FORTRAN AND THE NEW ANSI STANDARD FORTRAN.

1.3 USAGE

SPECIAL ACTION IS REQUIRED BY THE USER WHEN CREATING AND OPENING FILES TO BE USED FOR DIRECT ACCESS I/O. FILES USED FOR DIRECT ACCESS I/O SHOULD BE DAM FILES ON NEW PARTITIONS. IF THE FILE IS FORMATTED, THE ATTDEV SUBROUTINE MUST BE CALLED SO THAT FIXED LENGTH RECORDS ARE WRITTEN. THE ATTDEV SUBROUTINE IS ALSO USED TO SET THE RECORD LENGTH.

DAM FILES ARE CREATED BY OPENING A FILE THAT DOESN'T EXIST WITH THE K$NDAM SUBKEY IN A SRCH$$ OR TSRC$$ CALL. DIRECT ACCESS I/O STATEMENTS MAY REFER TO SAM FILES BUT NOT WITHOUT PERFORMANCE DEGRADATION. DAM FILES ON OLD PARTITIONS ARE LESS EFFECIENT THAN DAM FILES ON NEW PARITIONS WHEN THE SIZE OF THE FILE INCREASES BEYOND A CERTIAN SIZE (1,048,576 WORDS ON A STORAGE MODULE, 193,600 ON OTHER DISKS).

THE ATTDEV SUBROUTINE MAY BE USED TO ALTER THE MAPPING OF FORTRAN UNITS TO FILE SYSTEM UNITS AS WELL AS TO CHANGE THE RECORD SIZE FROM THE DEFAULT OF 60 WORDS. THE RECORDS OF A DIRECT ACCESS FORMATTED FILE MUST BE MADE FIXED LENGTH. THIS IS DONE BY SETTING THE SECOND ARGUMENT OF ATTDEV TO 8. THE RECORDS OF AN UNFORMATTED FILE ARE FIXED LENGTH BY DEFAULT. REMEMBER THAT IF THE RECORD LENGTH OF ANY FILE EXCEEDS 66 WORDS, A COMMON DECLARATION FOR F$IOBF MUST BE INCLUDED. THE SIZE OF F$IOBF MUST BE AS LARGE AS THE LARGEST RECORD SIZE.

A PROGRAM THAT CREATES A DIRECT ACCESS FILE WILL FAIL IF IT ATTEMPTS TO WRITE RECORD # N BEFORE RECORD # N-1 HAS BEEN WRITTEN. THIS PROBLEM CAN BE AVOIDED BY HAVING A SEPARATE PROGRAM THAT CREATES THE FILE BY WRITING ITS RECORDS SEQUENTIALLY. ONCE THE FILE HAS BEEN CREATED IT CAN BE READ OR WRITTEN IN RANDOM ORDER.

AFTER A DIRECT ACCESS I/O STATEMENT, THE FILE IS POSITIONED AT THE RECORD FOLLOWING THE ONE JUST TRANSFERRED. IF THE DIRECT ACCESS FILE IS BEING ACCESSED SEQUENTIALLY, IT IS NOT NECESSARY TO INCLUDE THE RECORD NUMBER. A NORMAL READ OR WRITE STATEMENT MAY BE USED. THIS WILL ENHANCE PERFORMANCE BY ELIMINATING THE POSITIONING CALL.

FORMATTED FILES THAT ARE USED FOR DIRECT ACCESS I/O MAY BE EXAMINED BY THE EDITOR. HOWEVER, THEY MUST NOT BE MODIFIED USING THE EDITOR. THE EDITOR PERFORMS SPACE COMPRESSION ON RECORD CAUSING THE RECORDS TO BECOME VARIABLE LENGTH. FILES USED FOR DIRECT ACCESS I/O MUST HAVE FIXED LENGTH RECORDS.

1.4 IBM COMPATABILITY

THE READ AND WRITE STATEMENT SUPPORTED ARE IDENTICAL TO IBM FORTRAN. THE DEFINE FILE AND FIND STATEMENTS OF IBM FORTRAN ARE NOT SUPPORTED. NOTICE THAT THE RECORD SIZE IN THE DEFINE FILE STATEMENT MUST APPEAR IN THE ATTDEV CALL. HOWEVER, THE RECORD SIZE IN THE DEFINE FILE STATEMENT IS MEASURED IN BYTES OR 32 BIT WORDS RATHER THAN 16 BIT WORDS AS REQUIRED BY ATTDEV. IF THE U SPECIFIER IS USED IN THE DEFINE FILE STATEMENT, THE RECORD SIZE OF THE DEFINE FILE STATEMENT SHOULD BE DOUBLED FOR THE ATTDEV CALL. OTHERWISE THE RECORD SIZE SHOULD BE HALVED.

NOTICE THAT THE ATTDEV CALL REQUIRES INTEGER*2 ARGUMENTS. IF THE INTL OPTION IS USED DURING COMPILATION, CONSTANTS USED AS ARGUMENTS IN THE ATTDEV CALLS MUST BE CONVERTED TO INTEGER*2 (E.G. INTS(8)).

THERE IS NO ANALOG OF THE DEFINE FILE ASSOCIATED VARIABLE IN PRIME'S IMPLEMENTATION OF DIRECT ACCESS FILES. IN IBM FORTRAN, THE VALUE OF THE ASSOCIATED VARAIBLE IS THE NUMBER OF THE RECORD THAT FOLLOWS THE RECORD JUST TRANSFERRED.

1.5 IMPLEMENTATION

IF A RECORD NUMBER APPEARS IN A READ OR WRITE STATEMENT, A CALL TO F$PO IS MADE WITH THE UNIT NUMBER AND RECORD NUMBER AS ARGUMENTS. F$PO IS SIMILAR TO POSFIL EXCEPT THAT ITS RECORD NUMBER IS A 32 BIT RATHER THAN 16 BIT INTEGER.

1.6 EXAMPLE 1

```
C     THIS PROGRAM CREATES A DIRECT ACCESS FILE
C     NOTICE CALLS TO ATTDEV AND SRCH$$
C
C
      IMPLICIT INTEGER*2 (A-Z)
C
      PARAMETER NUMREC=100              /* NUMBER OF RECORDS IN FILE
      PARAMETER RECSIZ=40               /* SIZE OF RECORDS IN INTEGER
*2 WORDS
      PARAMETER UNIT=5                  /* UNIT # USED IN FORTRAN REA
DS & WRITES
      PARAMETER FUNIT=1                 /* FILE UNIT # USED IN SRCH$$
$INSERT SYSCOM>KEYS.F
C
C     ATTDEV CALL - FORCES FIXED LENGTH RECORDS
C                   ESTABLISHES MAPPING OF UNIT TO FUNIT
C                   SET RECORD SIZE
      CALL ATTDEV(UNIT,8,FUNIT,RECSIZ)
C
C     OPEN FILE, USE K$NDAM SUBKEY TO FORCE DAM FILE
      CALL SRCH$$(K$WRIT+K$NDAM,'T$SCRATCH',9,FUNIT,TYPE,CODE)
      IF(CODE.NE.0)CALL ERRPR$(K$NRTN,CODE,0,0,0,0)
```

```
C
C       IF FILE ALREADY EXISTS, IT MIGHT NO BE DAM FILE
        IF(TYPE.NE.1) WRITE(1,1)
1       FORMAT('NOT A DAM FILE')
C
        DO 10 I=1,NUMREC
C       THE RECORD NUMBER IN THE FOLLOWING STATEMENT IS UNNECESSARY BE
CAUSE
C       THE RECORDS ARE BEING WRITTEN SEQUENTIALLY
10      WRITE(UNIT'I,2)I
2       FORMAT('THIS IS THE ',I3,' RECORD')
        CALL EXIT
        END
```

1.7 EXAMPLE 2

```
C       THIS PROGRAM RANDOMLY ACCESSES PREVIOUSLY CREATED
C       DIRECT ACCESS FILE
C
C
        IMPLICIT INTEGER*2 (A-Z)
C
        INTEGER*2 IBUF(40)
        PARAMETER NUMREC=100                /* NUMBER OF RECORDS IN FILE
        PARAMETER RECSIZ=40                 /* SIZE OF RECORDS IN INTEGER
*2 WORDS
        PARAMETER UNIT=5                    /* UNIT # USED IN FORTRAN REA
DS & WRITES
        PARAMETER FUNIT=1                   /* FILE UNIT # USED IN SRCH$$
$INSERT SYSCOM>KEYS.F
C
C       ATTDEV CALL - FORCES FIXED LENGTH RECORDS
C                     ESTABLISHES MAPPING OF UNIT TO FUNIT
C                     SET RECORD SIZE
        CALL ATTDEV(UNIT,8,FUNIT,RECSIZ)
C
C       OPEN THE FILE
        CALL SRCH$$(K$RDWR,'T$SCRATCH',9,FUNIT,TYPE,CODE)
        IF(CODE.NE.0)CALL ERRPR$(K$NRTN,CODE,0,0,0,0)
C
C       CHECK IF DAM FILE
        IF(TYPE.NE.1) WRITE(1,1)
1       FORMAT('NOT A DAM FILE')
C
30      WRITE(1,6)
6       FORMAT('RECORD #?')
        READ(1,7)REC
7       FORMAT(I6)
        WRITE(1,2)
2       FORMAT('READ OR WRITE?')
        READ(1,3)I
3       FORMAT(A1)
        IF(I.EQ.'R')GO TO 10
```

```
      IF(I.EQ.'W')GO TO 20
      CALL EXIT
C
10    READ(UNIT,4,REC=REC)IBUF
4     FORMAT(40A2)
      WRITE(1,4)IBUF
      GO TO 30
C
20    WRITE(1,8)
8     FORMAT('RECORD INFO?')
      READ(1,4)IBUF
      WRITE(UNIT'REC,4)IBUF
      GO TO 30
      END
```

2 GENERALIZED SUBSCRIPTS.

2.1 INTRODUCTION.

PREVIOUS TO REV. 15, FTN ALLOWED ONLY ANSI 1966 FORTRAN-IV SYNTAX IN ARRAY EXPRESSIONS IN ASSIGNMENT, IF, CALL, COMPUTED GOTO, STATEMENT FUNCTIONS, AND I/O STATEMENTS. THIS LIMITED THE POSSIBLE SUBSCRIPT EXPRESSION SYNTAX TO THE FOLLOWING:

```
C
V
C*V
C*V+C
C*V-C
```

WHERE C IS AN INTEGER CONSTANT
V IS AN INTEGER VARIABLE

THIS LIMITATION HAS BEEN LIFTED IN REV. 15 FTN. FTN NOW ALLOWS ANY LONG OR SHORT INTEGER EXPRESSION AS AN ARRAY SUBSCRIPT.

2.2 USE OF GENERALIZED SUBSCIPTS

ARRAY REFERENCES HAVE THE FORM

```
A(S1,S2,...,SN)
```

WHERE A IS THE ARRAY NAME
SI IS A SUBSCRIPT EXPRESSION, FOR 1<=I<=7

A SUBSCRIPT EXPRESSION IS ANY LEGAL FORTRAN INTEGER EXPRESSION. IT

MAY CONTAIN CONSTANTS, VARIABLES, FUNCTION REFERENCES, INTRINSIC REFERENCES, AND OTHER ARRAY REFERENCES. THE NESTING LIMIT ON ANY EXPRESSION IS 32 LEVELS OF PARENTHESES, WHETHER NORMAL, ARRAY, OR FUNCTION REFERENCE PARENTHESES. NON-INTEGER CONSTANTS AND VARIABLES ARE NOT ALLOWED WITHIN SUBSCRIPT EXPRESSIONS: HOWEVER, CONVERSION FUNCTIONS MAY BE USED TO CONVERT NON-INTEGER EXPRESSIONS TO INTEGER WITHIN A SUBSCRIPT EXPRESSION.

2.3 EXAMPLE

THE FOLLOWING IS A FORTRAN PROGRAM WHICH ILLUSTRATES THE USE OF GENERALIZED SUBSCRIPTS. FOR THAT REASON, IT CONTAINS SOME RATHER BIZARRE EXPRESSIONS WHICH CANNOT BE DESCRIBED AS THE RESULT OF GOOD CODING PRACTICE.

```
C
C           GENERALIZED SUBSCRIPTS
C
            REAL A(100,100),B(10),Z
            INTEGER G(3,4,5),H(3000),I,J,K
C           ASSIGNMENT
            Z=A(G(H(25*K**2),2,RS(I,H(2))),INTS(Z-A(1,10*H(J))))
     *      +B(INTS(POOP(2)))
C           IF
            IF(Z.NE.B(RS(K,H(K*5)))) GOTO 1000
C           CALL
1000        CALL POOP1(A(H(INTS(POOP(1))),G(1,J*2,1)),Z)
C           ETC.
            END
```

NOTE THAT POOP IS A REAL FUNCTION.

3 64V MODE COMMON

RECENT ENHANCEMENTS IN FTN AND SEG ALLOW SOME 64V MODE FORTRAN PROGRAMS FASTER ACCESS TO VARIABLES IN COMMON. IF A COMMON BLOCK IS LOADED INTO THE SAME SEGMENT AS THE PROCEDURE AREA OR LINK AREA WHICH ACCESSES IT, THE COMPILED PROGRAM WILL ADDRESS THE COMMON VARIABLES DIRECTLY, RATHER THAN THROUGH A TWO-WORD INDIRECT POINTER. THUS, CAREFUL LOADING OF ROUTINES WITH FREQUENTLY ACCESSED COMMON AREAS INTO THE SAME SEGMENT IN 64V MODE WILL GARNER AN APPRECIABLE INCREASE IN EXECUTION SPEED.

AN OFFSHOOT OF THESE ENHANCEMENTS IS THE FACT THAT FORTRAN 64V PROGRAMS COMPILED WITH REV. 15 FTN MUST BE LOADED USING REV. 15 SEG. USING AN OLDER VERSION OF SEG IN THIS CASE WILL RESULT IN SEG ERRORS.

DATE: MARCH 14, 1978

SUBJECT: CHANGES TO THE FORTRAN LIBRARY FOR REV. 15

FORTRAN LIBRARY CHANGES FOR REV. 15 FALL INTO TWO CATAGORIES: CHANGES TO SPECIFIC ROUTINES AND THE SHARED LIBRARY FACILITY.

CHANGES_TO_SPECIFIC_ROUTINES

NEW ROUTINES:

PHANT$	ENABLES A USER PROGRAM TO START UP A PHANTOM PROCESS. THIS ROUTINE IS DESCRIBED IN THE DOCUMENTATION FOR REV. 15 PRIMOS IV, PE-T-412.
F$PO	SUPPORT FOR FORTRAN DIRECT-ACCESS I/O.
E$52	RAISE THE COMPLEX NUMBER IN THE COMPLEX ACCUMULATOR (AC1-AC4) TO A REAL POWER, LEAVING THE RESULT IN THE COMPLEX ACCUMULATOR.
E$55	RAISE THE COMPLEX NUMBER IN THE COMPLEX ACCUMULATOR TO A COMPLEX POWER, LEAVING THE RESULT IN THE COMPLEX ACCUMULATOR.
C$15	NOW IN V-MODE LIBRARY.
C$51	NOW IN V-MODE LIBRARY.
FLCONV	THIS V-MODE EQUIVALENT OF FULCON INCLUDES DRIVERS FOR 9-TRACK ASCII AND EBCDIC AND 7-TRACK ASCII AND BCD MAGTAPE.

MODIFIED ROUTINES:

SLEEP$	IS NOW AVAILABLE FOR USE ON SYSTEMS WITH A P300 CPU.
DSIN DCOS	THE RANGE OF THESE ROUTINES HAS BEEN EXPANDED TO !ARGUMENT! < 3.37E9 FOR V-MODE, AND !ARGUMENT! < 1.69E9 FOR R-MODE. NOTE HOWEVER THAT NEAR THESE EXTREME END-POINTS THE FUNCTIONS' RESULT HAS ONLY ABOUT FOUR DIGITS OF ACCURACY.
F$IOBF	HAS BEEN INCREASED IN SIZE TO 128 WORDS. NOTE THAT IN THE SHARED LIBRARY THIS CAN NO LONGER BE EXPANDED BY LOADING A USER-CREATED F$IOBF BEFORE THE STANDARD ONE.
ERRSET GETERR	THESE ROUTINES HAVE BEEN REWRITTEN AND ARE NOW PURE.
BRAKE$ DUPLX$	THESE PRIMOS ROUTINES HAVE BEEN MODIFIED. SEE PE-T-412 FOR DETAILS.

SGDR$$
SATR$$

SHARED LIBRARY

THERE IS NEW A VERSION OF THE FORTRAN LIBRARY WHICH MAY BE SHARED AMONG ALL ITS USERS. IT MUST BE INSTALLED (SHARED) AT SYSTEM STARTUP TIME BY RUNNING THE COMMAND FILE SYSTEM>C SHLB, WHICH ALSO INSTALLS THE SHARED KI/DA AND, WHERE APPROPRIATE, COBOL AND FORMS LIBRARIES. ONCE THIS HAS BEEN DONE, USERS MAY USE THE NEW SHARED VERSION BY SUBSTITUTING

```
LI SFTNLB
IL
```

FOR "LI" IN THE LOAD PROCESS. NOTE THAT IT IS NOT POSSIBLE TO MIX SHARED AND UNSHARED LIBRARIES. IF ONE SHARED LIBRARY IS USED, THE SHARED VERSION OF ANY OTHER LIBRARY FOR WHICH A SHARED VERSION EXISTS MUST ALSO BE USED. MIXING SHARED AND UNSHARED LIBRARIES WILL PRODUCE UNPREDICTABLE UNDESIREABLE RESULTS.

THE SYSTEM MANAGER MAY INSTALL THE SHARED FORTRAN LIBRARY AS THE DEFAULT LIBRARY BY CNAMEING SFTNLB TO PFTNLB. HERE ESPECIALLY IT IS IMPORTANT TO REMEMBER THE NON-MIXABILITY OF SHARED AND NONSHARED LIBRARIES, AND TO CNAME THE KI/DA, COBOL AND FORMS LIBRARIES AS WELL. FOR A COMPLETE DISCUSSION OF THE BENEFITS AND PERILS OF SHARED LIBRARIES, SEE PE-T-434, SHARED LIBRARIES FOR REV. 15.

DATE: MARCH 7,1978

SUBJECT: CHANGES FOR FUTIL REVISION 15

THREE NEW COMMANDS HAVE BEEN PUT INTO FUTIL REVISION 15. THESE COMMANDS ARE SRWLOC, TRESRW AND UFDSRW. THEY SET THE PER-FILE READ-WRITE LOCK FOR A FILE, A TREE, AND ALL FILES IN THE CURRENT UFD, RESPECTIVELY. THE FORMAT OF THESE COMMANDS CORRESPOND TO THE FORMAT OF THE PROTECT-CLASS COMMANDS, I.E.:

SRWLOC FILENAME #
TRESRW FILENAME #
UFDSRW # NLEVELS

IS THE READ-WRITE LOCK. IF OMITTED, 0 IS THE DEFAULT. NLEVELS IS THE NUMBER OF LEVELS TO GO DOWN DOING THE SETTING. THE READ-WRITE LOCK IS INTERPRETED MODULO 4; 0 MEANS USE THE SYSTEM READ-WRITE LOCK, 1 MEANS ALLOW MULTIPLE READERS OR ONE WRITER, 2 MEANS ALLOW MULTIPLE READERS AND ONE WRITER, 3 MEANS ALLOW MULTIPLE READERS AND MULTIPLE WRITERS.

TO OUTPUT A FILE'S READ-WRITE LOCK, USE THE RWLOCK OPTION IN THE LISTF COMMAND IN FUTIL. A READ-WRITE LOCK OF 0 APPEARS AS "SYS", 1 APPEARS AS "W/NR", 2 APPEARS AS "1WNR", AND 3 IS SHOWN BY "NWNR".

OTHER MINOR CHANGES:

1. FUTIL NOW PRINTS (UFD) INSTEAD OF BEGIN DURING A LISTF COMMAND IF IT DOES NOT INTEND TO LIST THAT UFD. THIS WAY, BEGINS AND ENDS ARE TRULY NESTED. OF COURSE, IF IT ENCOUNTERS A SEGDIR INSTEAD OF A UFD AND STILL WON'T BE PRINTING IT'S CONTENTS, IT DOESN'T PRINT (UFD)...IN THIS CASE, (SEG) IS PRINTED.

2. FUTIL NO LONGER PRINTS PER-FILE READ-WRITE LOCKS FOR UFD NAMES WHEN ASKED, SINCE THEY DON'T APPLY.

3. FUTIL NOW IGNORES NULL LINES AND ACCEPTS LOWER CASE INPUT. ALSO, THE INPUT LINE IS TYPED OUT FOR A BAD SYNTAX ERROR SO THAT USE OF " AND ? DURING INPUT DON'T CAUSE THE UP-ARROW TO POINT TO THE WRONG PLACE.

4. FUTIL NOW PRINTS "(NO FIRST LINE)" WHEN A FILE SEEN BY A LISTF WITH THE FIRST OPTION DOESN'T HAVE A FIRST LINE INSTEAD OF ": NO FIRST LINE", WHICH COULD CONCEIVABLY BE AMBIGUOUS.

BUG FIXES:

1. THE CLEAN COMMAND NO LONGER LEAVES PROTECTION FOR FILES BELOW CURRENT LEVEL AT 7 0. INSTEAD, IT LEAVES THEM THE WAY THEY WERE.

2. A UFDDEL, WHEN IT ENCOUNTERS AN ERROR, WILL NO LONGER PRINT MULTIPLE ERROR MESSAGES ON THAT ONE FILE. NOW IT WILL PRINT THE APPROPRIATE ONES (THE ONE ERROR, AND A DIRECTORY NOT EMPTY FOR EVERY FATHER UFD).

3. DOING A UFDDEL FROM THE MFD NO LONGER PRODUCES SYNTACTICALLY INFINITE ERROR MESSAGES WHEN FUTIL TRIES TO ATTACH TO THE MFD AND READS THE MFD AS AN ENTRY, TRIES TO ATTACH, AD INFINITUM. FUTIL NOW TREATS THE MFD AS A SPECIAL FILE.

4. TWO SMALL SYNTAX BUGS WERE CORRECTED: A) FROM NOW ON, VOLUME NAMES OR NUMBERS USED AS A PREFIX (I.E. BEGINNING WITH <) MUST NOW ALSO END WITH >. IN FUTIL REVISION 14 ANY CHARACTER WOULD SUFFICE, B) NO LONGER MAY THE FIRST DIGIT OF A SEGMENT DIRECTORY FILE OR SUB-SEGMENT DIRECTORY (I.E. THE FIRST DIGIT OF THE NUMBER IN PARENTHESES) NOT BE A DIGIT, I.E. IN FUTIL REVISION 14, (A19) WAS INTERPRETED AS A LEGAL SEGMENT FILENAME. ALSO, VOLUME NUMBERS NO LONGER GET MIS-INTERPRETED IF THEY ARE OCTAL 20 OR ABOVE, THEY NOW WIN UP TO 77777 OCTAL.

NOTE:

FUTIL REVISION 15 WILL NOT WORK ON 32K SDOS. WE RECOMMEND THAT USERS OF 32K SDOS USE AN EARLIER REVISION OF FUTIL.

ANOTHER NOTE:

SINCE FUTIL REVISION 15 NOW ACCEPTS LOWER CASE INPUT, IT IS CONCEIVABLE THAT WHILE FUTIL TRANSLATES LOWER CASE TO UPPER CASE FOR COMMANDS, FILENAMES, AND OPTIONS, SOME USERS MAY FORGET THAT PASSWORDS ARE STILL TAKEN AS IS, I.E. A UFD WITH A PASSWORD OF ABC CAN ONLY BE ATTACHED TO AS AN OWNER BY FUTIL IF THE PASSWORD IS ENTERED LITERALLY AS ABC (UPPERCASED). THIS IS ONE OF THE FIRST POSSIBILITIES TO CONSIDER BEFORE DECIDING THAT A BUG EXISTS IN FUTIL.

DATE: MARCH 15, 1978

SUBJECT: KI/DA FOR REV. 15

KI/DA FOR REV. 15 CONTAINS NO NEW FEATURES. THE ONLY ENHANCEMENT TO KI/DA HAS BEEN THE CREATION OF A VERSION OF KI/DA WHICH CAN BE SHARED ON THE P-400 FOR V-MODE PROGRAMS.

TO INSTALL THE SHARED KI/DA LIBRARY THE USER MUST PLAN TO USE ALL THE SHARED LIBRARIES SUPPLIED FOR HIS SYSTEM. THE SHARED LIBRARIES ARE INSTALLED AT STARTUP TIME BY RUNNING THE COMMAND FILE C_SHLB IN UFD SYSTEM.

PROGRAMS WHICH WILL USE THE SHARED LIBRARIES MUST BE REBUILT USING THE NEW SHARED LIBRARY OBJECT FILES IN UFD LIB. LOAD MAPS FOR THESE RELOADED PROGRAMS WILL BE CONSIDERABLY SMALLER AND WILL REFLECT THE FACT THAT THE KI/DA ROUTINES AND MANY OF THE OTHER LIBRARY ROUTINES HAVE BECOME DIRECT ENTRY CALLS.

SHOULD THERE BE ANY NEED TO BEBUILD KI/DA THE COMMAND FILE C_SKLB IN UFD KI/DA SHOULD BE RUN TO REBUILD THE SHARED LIBRARY RUN FILES. IT IS NOT NECESSARY TO REBUILD APPLICATIONS USING THE SHARED LIBRARY. WHEN A NEW SHARED LIBRARY IS INSTALLED USERS AUTOMATICALLY BENEFIT FROM THE MODIFIED LIBRARY WHEN LINKS ARE SNAPPED AT RUN TIME.

DATE: JANUARY 31, 1978

SUBJECT: LOAD REV. 15 THE VIRTUAL LOADER FOR P300 PROGRAMS

AT REV 15 THE VIRTUAL LOADER FORMERLY CALLED VLOAD CAN BE USED TO REPLACE ALL COPIES OF THE LOADER (LOAD, HILOAD, LOAD20, ETC.) UNDER PRIMOS IV, PRIMOS III AND 64K PRIMOS II. THE COMMAND FORMAT IS THE SAME AS THAT OF THE OLD LOADER, HOWEVER, THE FUNCTIONALITY HAS BEEN CONSIDERABLY ENHANCED. SOME NEW COMMANDS HAVE BEEN ADDED AND ERROR REPORTING HAS BEEN IMPROVED TO PROVIDE SHORT TEXT DESCRIPTIONS OF THE ERROR CONDITION IN PLACE OF THE OLD TWO CHARACTER CODES. IN ADDITION 'Q' AND 'A' ARE NOW ACCEPTED FOR 'QU(IT)' AND 'AT(TACH)' RESPECTIVELY. NOTE THAT THE NEW COMMANDS ARE PRIMARILY INTENDED TO GIVE GREATER FLEXIBILITY TO USERS LOADING VERY BIG PROGRAMS. FOR THE AVERAGE USER LOAD BEHAVES EXACTLY AS THE OLD LOADER DOES. FUTURE PLANS FOR IMPROVING LOAD INCLUDE AN IMPROVED COMMAND LINE HANDLER AND OTHER FEATURES TO MAKE LOADING EASIER.

A WORD OF WARNING

LOAD WILL ATTEMPT TO LOAD THE MEMORY IMAGE IN THE ACTUAL LOCATIONS TO BE USED IN THE RUNFILE. WHEN THIS IS NOT POSSIBLE OTHER LOCATIONS WILL BE USED OR INFORMATION WILL BE WRITTEN TO A TEMPORARY FILE. FOR THIS REASON, PRIMARILY, THERE ARE SOME INCOMPATIBILITIES AND MINOR DIFFERENCES BETWEEN USING LOAD AND USING ONE OF THE OLD LOADERS. USERS SHOULD FAMILIARIZE THEMSELVES WITH THESE DIFFERENCES. THEY ARE FULLY DESCRIBED IN SECTION 4 OF THIS DOCUMENT.

THE FOLLOWING NEW FEATURES ARE DESCRIBED IN SECTION 2:

1 ENHANCEMENTS TO THE LOAD FAMILY OF COMMANDS
2 TREENAME USAGE
3 SAVING AND EXECUTING LOAD PROGRAMS
4 THE CHECK COMMAND
5 THE PB(REAK) COMMAND
6 THE SY(MBOL) COMMAND
7 THE ERROR COMMAND
8 THE SZ (SECTOR ZERO) COMMAND
9 THE SS (SAVE SYMBOLS) COMMAND FOR USE WITH EXPUNGE
10 THE DC (DEFER COMMON DEFINITION) COMMAND
11 THE EN (ENTIRE SAVE) COMMAND FOR CREATING OVERLAYS
12 THE PA (PAUSE) COMMAND FOR TEMPORARILY LEAVING LOAD
13 LOAD ERROR MESSAGES

SECTION 1 EXPLAINS THE MAJOR DIFFERENCES BETWEEN LOAD AND THE OLD LOADERS. SECTION 3 DESCIBES HOW TO RELOCATE LOAD.

1 PRINCIPLES OF OPERATION OF LOAD

LOAD USES THE SAME COMMAND STRUCTURE AS THE OLD LOADERS AND FOR MOST USERS OLD COMMAND FILES SHOULD RUN EXACTLY AS THEY ALWAYS HAVE. THE MAJOR DIFFERENCE BETWEEN LOAD AND THE OLD LOADERS IS THAT LOAD CAN - WHEN NECESSARY - USE MEMORY LOCATIONS OTHER THAN THOSE REQUIRED IN THE ACTUAL RUN FILE AS "BUFFERS" AND CAN PAGE THESE BUFFERS OUT TO A TEMPORARY FILE TO MAKE ROOM FOR ADDITIONAL PAGES OF INFORMATION. SINCE LOAD IS NOT REQUIRED TO SHARE LOAD SPACE WITH THE PROGRAM BEING LOADED, LOAD IS NOT LIMITED BY SIZE RESTRICTIONS AND THEREFORE IT CAN HAVE ENHANCED FUNCTIONALITY.

WHEN LOADING A PROGRAM, LOAD FIRST ATTEMPTS TO USE AS BUFFER SPACE THE ACTUAL LOCATIONS SPECIFIED BY THE MEMORY IMAGE BEING BUILT. WHEN THIS IS NOT POSSIBLE AVAILABLE MEMORY SPACE WILL BE USED AS BUFFERS AND IF THIS IS NOT SUFFICIENT, BUFFERS WILL BE PAGED OUT OF MEMORY INTO A TEMPORARY FILE. FOR PRACTICAL PURPOSES THIS MEANS THAT LOAD WILL LOAD 32R MODE PROGRAMS "IN PLACE" AND THESE PROGRAMS WILL BE EXECUTABLE DIRECTLY - AS IN THE PAST. THAT IS TO SAY:

```
LO B_FOO
LI
EX
```

WILL LOAD FOO, LINK IT TO THE FORTRAN LIBRARY AND THEN TRANSFER CONTROL TO THE LOADED PROGRAM - JUST AS THE OLD LOADERS DID.

USERS LOADING 64R MODE PROGRAMS MAY FIND THAT LOAD MUST USE ITS BUFFERING ABILITIES AND WILL BE REQUIRED TO SAVE THE LOADED IMAGE FIRST. ONCE THE IMAGE IS SAVED, LOAD CAN RESUME IT ON BEHALF OF THE USER.

THE LOAD TEMPORARY FILE IS OPENED WHEN VLOAD IS FIRST INVOKED AND CLOSED WHEN THE QUIT OR EXECUTE COMMANDS ARE GIVEN. IT IS OPENED IN THE HOME UFD WITH AN UNIQUE NAME OF THE FORM T$XXXX, WHERE XXXX IS ANY NUMBER FROM 0000 TO 9999. THE USER REMAINS ATTACHED TO THE HOME UFD ONCE THIS OPEN IS DONE.

2 NEW FEATURES OF LOAD

2.1 ENHANCEMENTS TO THE LOAD FAMILY OF COMMANDS

THE LOAD FAMILY OF COMMANDS CONSISTS OF LOAD, FORCE LOAD, P/LO, F/LO AND P/F/LO. THE LIBRARY COMMAND IS NOT INCLUDED. LIBRARIES MAY, HOWEVER, BE LOADED (LO LIB>FTNLIB - FOR EXAMPLE). THE NEW COMMAND FORMAT ALLOWS THE USER TO MAKE USE OF THE ABILITY OF COMPILERS AND PMA TO TELL THE LOAD HOW LONG A MODULE IS BEFORE IT IS LOADED. THIS ALLOWS LOAD TO PLACE A BASE AREA AT THE END OF A MODULE PRIOR TO LOADING IT SO THAT NOT ALL LINKS MUST BE PLACED IN AREAS PRECEDING THE MODULE (WHICH ARE SOMETIMES OUT OF REACH).

THERE ARE THREE FORMS OF EACH OF THE LOAD FAMILY OF COMMANDS:

```
1 LO FNAME ADDR SETB1 SETB2 .... SETB8
2 LO FNAME * SETB1 SETB2 .... SETB9
3 LO FNAME SYMBOL SETB1 SETB2 .... SETB9
```

WHERE UPPER CASE IS USED TO DENOTE TEXT FIELDS AND LOWER CASE TO DENOTE NUMERIC PARAMTERS. THE SAMPLE COMMAND 'LO' MAY BE REPLACED BY ANY OTHER OF THE MEMBERS OF THE FAMILY TO ACHIEVE THE SAME EFFECTS WITH THESE SPECIAL PURPOSE COMMANDS.

IN FORM 1 OF THE COMMANDS THE FIST NUMERIC PARAMETER (ADDR) IS INTERPRETED AS THE STARTING LOCATION OF THE LOAD. NOTE THAT:

```
LO FNAME
```

AND

```
LO FNAME 10000
```

ARE THE SIMPLEST FORMS OF THE COMMAND AND LOOK JUST LIKE THE OLD LOADER COMMANDS. NO BASE AREAS ARE INDICATED, SO NONE ARE PLACED. THE REMAINING NUMERIC PARAMETERS (SETB1, SETB2 ...) ARE INTERPRETED AS THE SIZE OF LINKAGE AREAS TO BE INSERTED BEFORE AND AFTER MODULES IN THE OBJECT FILE AS IT IS LOADED. FOR EXAMPLE, IF THERE ARE 4 MODULES (SUBROUTINES) IN THE OBJECT FILE B_SUB4 THE COMMAND:

```
LO B_SUB4 2000 10 20 0 40 50
```

WILL CAUSE THE PROGRAM BREAK (PBRK) TO BE SET TO 2000, THEN AT 2000 A BASE AREA OF LENGTH 10 WILL BE CREATED BEFORE THE FIRST ROUTINE, ONE OF LENGTH 20 AFTER THE FIRST ROUTINE, NONE AFTER THE SECOND ROUTINE, A BASE AREA OF LENGTH 40 AFTER THE THIRD ROUTINE AND A BASE AREA OF LENGTH 50 AFTER THE FOURTH ROUTINE. THE LOADER KNOWS ABOUT EACH BASE AREA BEFORE LOADING OF THE MODULE WHICH IT FOLLOWS BEGINS, HENCE IT CAN BE USED BY THE ROUTINE BEING LOADED.

FORM 2 OF THE COMMANDS USES THE PLACE INDICATOR '*' TO DENOTE THE CURRENT LOAD LOCATION. THE NUMERIC PARAMETERS (SETB1, SETB2, ..., SETB9) ARE INTERPRETED AS BASE AREA LENGTHS AS ABOVE. FOR EXAMPLE:

```
FO B_BIG * 30 30
```

PLACES BASE AREAS OF LENGTH 30 BEFORE AND AFTER THE FIRST ROUTINE IN OBJECT FILE B_BIG. NOTE THAT THIS DIFFERS FROM THE METHOD OF ACCOMPLISHING A POST-BASE AREA PROVIDED WITH THE OLD LOADER(!).

FINALLY FORM 3 OF THE COMMANDS ALLOWS THE USER TO SPECIFY THE LOAD ADDRESS SYMBOLICALLY. FOR EXAMPLE IF THE SYMBOL FSTEND HAS THE VALUE (LOCATION) 10000

```
LO B_MIDDLE FSTEND
```

CAUSES LOAD TO BEGIN LOADING AT 10000. THE ONLY REQUIRMENT IS THAT

FSTEND BE A DEFINED SYMBOL. THIS CAN BE ACCOMPLISHED EITHER BY THE USE OF THE SYMBOL COMMAND OR THROUGH SYMBOL DEFINITION FROM AN OBJECT MODULE. AS IN THE OTHER TWO FORMS OF THE COMMAND THE NUMERIC PARAMETERS ARE INTERPRETED AS BASE AREA LENGTHS.

SHOULD THE NUMBER OF NUMERIC PARAMETERS POSSIBLE FOR BASE AREA LENGTHS (8 OR 9) BE INSUFFICIENT, THE LAST ONE MAY BE SET TO 177777. THIS WILL CAUSE LOAD TO ASK FOR MORE LENGTHS.

NOTE: THE ORIGINAL USE OF THE NUMERIC PARAMETERS IN THE OLD LOADER COMMAND LINE - LO FNAME ADDR BASADR BASLNT - IS NOI SUPPORTED. IN THIS CASE 'ADDR' WAS THE STARTING ADDRESS OF THE LOAD, 'BASADR' WAS THE ADDRESS FOR A BASE AREA AND 'BASLNT' WAS THE LENGTH OF THE AREA.

2.2 ENHANCEMENTS TO THE MAP COMMAND

THREE NEW MAP FUNCTIONS ARE AVAILABLE. THESE ARE MAPS 6, 7 AND 10.

MAP 6 PRINTS UNDEFINED SYMBOLS, SORTED ALPHABETICALLY. MAP 7 PRINTS ALL SYMBOLS SORTED ALPHABETICALLY. MAP 10 MUST BE WRITTEN TO A FILE AND CREATES A SPECIAL SYMBOL MAP FOR USE BY PSD.

IN ADDITION THE MAP HAS BEEN SLIGHTLY REFORMATTED. COMMON BLOCK NAMES ARE NOW SEPARATED FROM OTHER SYMBOLS AND REPORTED IN A SEPARATE SECTION OF THE MAP. TWO NUMERIC FIELDS FOLLOW THE COMMON BLOCK NAME. THE FIRST IS THE LOCATION OF THE COMMON BLOCK. THE SECOND IS THE LENGTH OF THE COMMON BLOCK IN OCTAL, IF THIS VALUE IS KNOWN. THE LENGTH OF A COMMON BLOCK IS KNOWN IF IT IS FIRST DECLARED TO THE LOADER AS A COMMON NAME. EXTERNAL SYMBOLS FIRST DECLARED TO THE LOADER AS ENTRY POINTS BY PMA MODULES DO NOT HAVE A KNOWN LENGTH.

2.3 TREENAME USAGE

LOAD WILL ACCEPT EITHER TREENAMES OR LOCAL FILE NAMES WHENEVER A FILE NAME IS SUPPLIED. IN PARTICULAR THE LOAD FAMILY OF COMMANDS MAY MAKE USE OF TREENAMES AND THE MAP COMMAND MAY USE THEM. WHEN A TREENAME IS SUPPLIED LOAD FIRST ATTACHES TO THE HOME UFD, THEN TEMPORARILY FOLLOWS THE PATH SUPPLIED TO OPEN THE FILE. ONCE THE FILE HAS BEEN OPENED LOAD REATTACHES TO HOME.

IF THE NAME SUPPLIED IS NOT A TREENAME - 'DOES NOT CONTAIN THE CHARACTER '>' - LOAD DOES NOT CHANGE ANY TEMPORARY ATTACHES THE USER MAY HAVE MADE TO OTHER UFD'S AS .NOT.HOME.

THE EFFECT OF THIS ALGORTHM FOR HANDLING ATTACHES IS THAT FOR THE MOST PART LOAD BEHAVES LIKE THE OLD LOADER IN REGARD TO UFD ATTACHMENTS. NOTE, HOWEVER, THE DISCUSSION OF TEMPORARY FILES IN SECTION 4.

2.4 SAVING AND EXECUTING LOAD PROGRAMS

LOAD IS A VIRTUAL LOADER, HOWEVER, WHEN POSSIBLE IT USES THE ACTUAL MEMORY LOCATIONS REFERENCED AS ITS BUFFERS. THIS MEANS THAT PROGRAMS WHICH LIE ENTIRELY WITHIN ITS BUFFER SPACE CAN BE STARTED BY THE 'EXECUTE' COMMAND WITHOUT FIRST SAVING THE RUNFILE. AS DELIVERED THE BUFFER SPACE IS ALL OF MEMORY BELOW :122000. THIS MEANS THAT ALL 32R, 16S AND 32S PROGRAMS CAN BE EXECUTED.

WHEN A 64R MODE PROGRAM HAS BEEN LOADED, WHICH CONTAINES LOADED INFORMATION OUTSIDE OF THE BUFFER SPACE LOAD WILL RESPOND TO THE EX COMMAND WITH "CAN'T - PLEASE SAVE". ONCE THE IMAGE HAS BEEN SAVED IT MAY BE EXECUTED.

LOAD'S SAVE COMMAND LOOKS JUST LIKE THE OLD LOADER'S. AT SAVE TIME THE USER SUPPLIES A FILENAME. IF THE EX COMMAND IS GIVEN, THE TEMPORARY FILE IS CLOSED AND DELETED. OTHERWISE LOADING MAY CONTINUE, AS IN THE PAST.

2.5 THE CHECK COMMAND

THE CHECK COMMAND ALLOWS THE USER TO CHECK THE VALUE OF THE CURRENT PBRK AGAINST THE VALUE OF A SYMBOL OR NUMBER. THIS WILL BE FOUND TO BE USEFUL WHEN IT IS NECESSARY TO LOAD MODULES OUT OF ORDER AND BELOW PREVIOUSLY LOADED INFORMATION OR WHEN A MODULE IS NOT SUPPOSED TO EXCEED A CERTAIN SIZE. THE FORMAT OF THE COMMAND IS:

```
CH [SYMBOLNAME] [PAR1 [-] PAR2 PAR3 ... PAR6]
```

SYMBOLNAME IS A 6 CHARACTER SYMBOL WHICH MUST BE DEFINED IN THE SYMBOL TABLE. SYMBOLNAME IS OPTIONAL. PAR1 THROUGH PAR6 ARE NUMERIC PARAMETERS WHICH WILL BE SUMMED TO FORM AN ADDRESS OR OFFSET FROM SYMBOLNAME. EACH NUMBER MAY BE PRECEDED BY "- ", IN WHICH CASE IT WILL BE NEGATED (SEE BELOW).

FOR EXAMPLE IF THE VALUE (LOCATION) OF OVRFLW IS :17777 AND THE PBRK IS 20010, THEN:

```
CH OVRFLW
```

WILL REPORT '000011 WDS OVERLAP'. IF ON THE OTHER HAND PBRK WAS 17770, THE CHECK WOULD REPORT '000007 WDS TO SPARE'.

SIMILARLY:

```
CH 50000 - 20     WILL COMPARE PBRK WITH :47760
CH OVRFLW 1000    WILL COMPARE PBRK WITH :20777
```

2.6 THE PB(REAK) COMMAND

THIS COMMAND ALLOWS THE USER TO SET PBRK TO THE VALUE OF EITHER A SYMBOL PLUS OFFSET OR A NUMBER. AS WITH THE CHECK COMMAND IT IS INTENDED TO FACILITATE COMPLICATED LOADS. THE FORMATS OF THE COMMAND ARE:

```
PB [SYMBOLNAME] [PAR1 [-] PAR2 ... PAR6]
PB * PAR1 [[-] PAR2 ... PAR6]
```

SYMBOLNAME IS AN OPTIONAL SYMBOL NAME WHICH MUST BE A DEFINED SYMBOL. PAR1 THROUGH PAR6 ARE NUMERIC PARAMETERS WHICH WILL BE SUMMED. IF SYMBOLNAME IS PRESENT THE SUM OF THE NUMBERS WILL BE TREATED AS AN OFFSET FROM SYMBOLNAME. IF "*" IS PRESENT THE SUM WILL BE TREATED AS AN OFFSET FROM THE CURRENT PBRK. IF NEITHER IS PRESENT THE SUM WILL BE THE ACTUAL VALUE TO WHICH PBRK IS SET.

FOR EXAMPLE:

```
PB 10000 10
```

MOVES THE VALUE OF PBRK TO 10010. SIMILARLY, IF OLDEND IS A SYMBOL WITH THE VALUE :17456, THEN:

```
PB OLDEND 10
```

WILL SET PBRK TO :17466.

IF PBRK IS CURRENTLY :1000, THEN:

```
PB * 10000 - 77
```

WILL SET PBRK TO :10701.

2.7 THE SY(MBOL) COMMAND

THE SY(MBOL) COMMAND ALLOWS THE USER TO ESTABLISH LOCATIONS IN THE MEMORY MAP WHICH CAN THEN BE USED FOR THE LOCATIONS OF COMMON BLOCKS OR TO PROVIDE RELOCATION POINTS FOR THE COURSE OF THE LOAD. IT MAY ALSO BE USED TO SATISFY UNSATISFIED REFERENCES. THE COMMAND HAS THREE FORMS:

```
SY SNAME OLDSYM [PAR1 [-] PAR2 ... PAR6]
SY SNAME ADDR [PAR2 [-] PAR3 ... PAR6]
SY SNAME * [PAR1 [-] PAR2 ... PAR3]
```

THE FIRST FORM ALLOWS THE USER TO EQUATE TWO SYMBOLS OR TO EQUATE THE NEW SYMBOL TO AN OFFSET FROM THE OLD. FOR EXAMPLE:

```
SY SNAME OLDSYM
```

IN THE EXAMPLE ABOVE SNAME WILL BE EQUATED TO OLDSYM. OLDSYM MUST

BE A DEFINED SYMBOL IN THE SYMBOL TABLE. THE SECOND FORM ALLOWS THE USER TO SET THE VALUE OF A SYMBOL TO A NUMBER - FOR EXAMPLE:

```
SY SNAME 1300 20 - 7 10     SETS SNAME TO :1321
```

THE THIRD FORM OF THE COMMAND PERMITS THE USER TO SET THE VALUE OF THE SYMBOL TO THE VALUE OF THE CURRENT PBRK PLUS THE RESULT OF SUMMING THE NUMERIC PARAMETERS. FOR EXAMPLE THE SEQUENCE OF COMMANDS:

```
SY OVRFLW *
LO B_TEST * 10 20
CH OVRFLW 10 567 20
```

WILL ALLOW THE USER TO BE SURE THAT THE MODULE TEST DOES NOT OCCUPY MORE THAN :567 LOCATIONS.

2.8 THE ER(ROR) COMMAND

THIS COMMAND ALLOWS THE USER TO DETERMINE WHAT ACTION LOAD WILL TAKE WHEN AN ERROR OCCURS. THE FORM OF THE COMMAND IS:

```
ER NUM
```

WHERE 'NUM' IS 0, 1 OR 2. THE DEFAULT IS 1. UNDER THIS CONDITION 'MULTIPLE INDIRECT' ERRORS ARE NOTED ON THE TERMINAL, BUT LOAD CONTINUES LOADING THE MODULE. ALL OTHER ERRORS CAUSE THE LOAD OF THE OFFENDING OBJECT FILE TO TERMINATE AND LOAD RETURNS TO ITS COMMAND LEVEL. THIS IS THE WAY IN WHICH THE OLD LOADER HANDLES ERRORS.

IF ER IS SET TO 0 ERRORS GENERATED BY THE SZ COMMAND (SEE BELOW) ARE ALSO TREATED THIS WAY. ALL OTHER ERRORS ABORT THE LOADING OF THE OBJECT FILE.

IF ER IS SET TO 2, ALL ERRORS ABORT TO PRIMOS IV.

	ER VALUE		
	0	1	2
CONTINUE ACTION	MI SZ	MI	NONE
TERMINATE	ALL BUT MI SZ	ALL BUT MI	NONE
ABORT	NONE	NONE	ALL

2.9 THE SZ (SECTOR ZERO) COMMAND

IN SOME CASES THE USER EITHER WISHES TO HAVE NO LINKS IN SECTOR ZERO, OR WISHES TO PREVENT SOME ROUTINES FROM USING SECTOR ZERO LINKS. THE SZ COMMAND PROVIDES A MEANS OF ACCOMPLISHING THIS. THE COMMAND HAS THE FORM:

```
SZ (NO)
SZ YES
```

IF 'SZ' OR 'SZ NO' IS GIVEN AS THE COMMAND LOAD WILL NOT PUT LINKS IN SECTOR ZERO. INSTEAD IT WILL FLAG THE ATTEMPT GIVING THE LOCATION OF THE INSTRUCTION ATTEMPTING TO LINK. THIS WILL NORMALLY TERMINATE THE LOADING OF THE OBJECT FILE. HOWEVER, IF ER(ROR) HAS BEEN SET TO 0, LOADING WILL CONTINUE AND THUS GENERATE A LIST OF SECTOR ZERO LINK ATTEMPTS.

IF 'SZ YES' IS GIVEN AS THE COMMAND LINKING IS AGAIN PERMITTED IN SECTOR ZERO.

NOTE THAT IF A NEGATIVE FORM OF THE SZ COMMAND HAS BEEN GIVEN, A SECTOR ZERO BASE AREA WILL BE CREATED BUI NO LINKS WILL BE PUT INTO IT WHILE THE SZ .NOT. IS IN EFFECT.

2.10 THE SS (SAVE SYMBOLS) COMMAND

THE EXPUNGE COMMAND IN THE PAST HAS ALLOWED USERS TO DELETE SYMBOLS FROM THE SYMBOL TABLE WITH THE OPTION OF SAVING COMMON NAMES. THE SS COMMAND ADDS A NEW CLASS OF SYMBOLS WHICH WILL NOT BE EXPUNGED. THE FORM OF THE COMMAND IS:

```
SS SYMBOLNAME
```

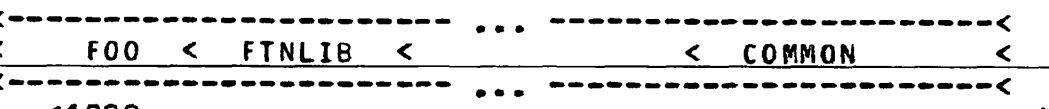
WHERE SYMBOLNAME IS A 6 CHARACTER NAME IN THE SYMBOL TABLE. ALL SYMBOLS THUS REFERENCED WILL NOT BE DELETED IF THE SYMBOL TABLE IS EXPUNGED.

2.11 THE DC (DEFER COMMON DEFINITION) COMMAND

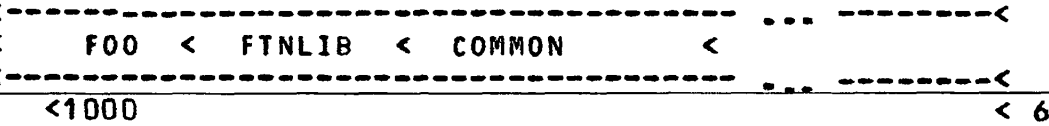
THIS COMMAND ALLOWS THE USER TO DEFER DEFINITION OF COMMON BLOCKS UNTIL CONVENIENT OR UNTIL A SAVE OR EXECUTE COMMAND IS GIVEN. THE FORMAT OF THE COMMAND IS:

```
DC (END)
```

IF "END" IS MISSING THE DEFER COMMON FEATURE IS TURNED ON. IF THE "END" IS PRESENT IT IS TURNED OFF. AS DELIVERED, DEFER COMMON IS TURNED OFF AND COMMON WILL BE CREATED AS IN THE PAST. COMMON WHICH HAS BEEN DEFERRED WILL BE CREATED IMMEDIATELY FOLLOWING THE LOADED PROGRAM INSTEAD OF BEING DEFINED AS "DOWN FROM CMHIGH". FOR EXAMPLE IF A USER HAS LOADED FOO AND THE LIBRARY WITH LOAD, MEMORY ALLOCATION WILL USUALLY LOOK LIKE:



IF THE PROGRAM WERE LOADED UNDER THE DC OPTION, THE LOAD MAP WOULD LOOK LIKE:



COMMON WILL BE DEFINED (ALLOCATED) AS FOLLOWS

1 WHEN THE USER GIVES THE "DC END" COMMAND ALL COMMON IS DEFINED.
2 WHEN THE LOADING PROGRAM ATTEMPTS TO INITIALIZE A COMMON BLOCK, THAT BLOCK IS DEFINED.
3 WHEN A PMA MODULE CREATES A COMMON BLOCK IT IS DEFINED.
4 WHEN A SAVE OR EXECUTE COMMAND IS GIVEN ALL COMMON IS DEFINED.

2.12 THE EN (ENTIRE SAVE) COMMAND

FROM TIME TO TIME USERS WISH TO CREATE "OVERLAYS" WHICH ARE LOADED WITH AND RUN WITH A COMMON MAIN PROGRAM. THIS FEATURE HAS NOW BEEN BUILT INTO THE LOADER. WHEN THE MAIN PROGRAM AND COMMON COMMON BLOCKS HAVE BEEN CREATED, THE EN COMMAND CAN BE USED TO SAVE THE ENTIRE STATE OF THE LOADER, COMPLETE WITH TEMPORARY FILE. IN PARTICULAR THE CURRENT RUNNING COPY OF THE LOADER IS SAVED AS A RUN FILE TOGETHER WITH ALL BUFFERS AND DATA BASES. THE CURRENT COPY OF THE TEMPORARY FILE IS COPIED INTO A NEW TEMPORARY FILE AND THE ORIGINAL CLOSED. THE NEW TEMPORARY FILE ALSO HAS A NAME OF THE FORM T$XXXX. THE CURRENT COPY OF THE LOADER AND NEW TEMPORARY FILE MAY THEN BE USED TO CREATE THE FIRST OVERLAY. SUBSEQUENT OVERLAYS MAY BE CREATED USING THE SAVED COPY. THE FORMAT OF THE COMMAND IS:

```
EN FNAME
```

FNAME IS THE NAME OF THE SAVED COPY OF THE LOADER. A COMMENTED EXAMPLE OF THE USE OF EN FOLLOWS:

```
OK, LOAD
GO
$ LO B_MAIN 10000          LOAD MAIN PROGRAM ABOVE OVERLAYS
$ LI                       SATISFY ITS LIBRARY REFERENCES
$ SAVE *MAIN
$ EN MAINLDR               CREATE SAVED COPY OF LOADER
$ LO B_OVER1 1000          BUILD FIRST OVERLAY
$ LI
$ SAVE *OVER1              AND SAVE IT
```

```
$ QU

OK, R MAINLDR
GO
$ EN MAINLDR                    PRESERVE STATE OF TEMPORARY FILE
$ LO B_OVER2 1000               CREATE SECOND OVERLAY
$ LI
$ SAVE *OVER2                   AND SAVE IT
$ QU

OK, R MAINLDR                   ONLY 3 OVERLAYS
GO
$ LO B_OVER3 1000
$ LI
$ SAVE *OVER3
$ QU

OK,
```

NOTE IN THE EXAMPLE THAT THE OVERLAYS ARE SANDWICHED BETWEEN THE MAIN PROGRAM AND ITS LINKS. THIS AVOIDS SECTOR ZERO LINK CONFLICTS. NOTE ALSO THAT WHEN THE SECOND OF THE THREE OVERLAYS IS TO BE CREATED, AN EN COMMAND IS THE FIRST LOADER COMMAND GIVEN. THIS SERVES TO PRESERVE THE "INCOMING" TEMPORARY FILE FOR USE IN BUILDING THE NEXT OVERLAY.

AS PRESENTLY IMPLEMENTED THE SAVE RANGE FOR THE OVERLAYS IS INCLUSIVE OF ALL LOCATIONS LOADED UP TO THE TIME OF THE SAVE. IT IS UP TO THE USER TO CHANGE THE SAVE RANGES USING THE PRIMOS SAVE COMMAND AS APPROPRIATE.

2.13 THE PAUSE COMMAND

IT IS NO LONGER POSSIBLE TO QUIT THE LOADER AND RETURN TO IT AND CONTINUE LOADING. QUIT CLOSES AND DELETES THE TEMPORARY FILE AND THUS CHANGES THE LOADER'S STATE. TO LEAVE THE LOADER FOR THE PURPOSE OF EXECUTING INTERNAL COMMANDS THE USER MAY NOW TYPE "PAUSE". THE TEMPORARY FILE IS LEFT OPEN AND A COMMAND OF "S" TO PRIMOS WILL RESUME THE LOADER AT ITS PREVIOUS STATE.

2.14 LOAD ERROR MESSAGES

BASE SECTOR 0 FULL - ALL LOCATIONS IN THE SECTOR ZERO BASE AREA HAVE BEEN USED. USE THE AU COMMAND TO GENERATE BASE AREAS AT REGULAR INTERVALS, OR USE THE SETB OR LOAD COMMANDS TO SPECIFICALLY PLACE BASE AREAS.

BAD OBJECT FILE - THE OBJECT TEXT IS NOT RECOGNIZABLE. THIS USUALLY OCCURS WHEN AN ATTEMPT IS MADE TO LOAD SOURCE CODE OR WHEN THE OBJECT TEXT WAS COMPILED OR ASSEMBLED FOR SEGMENTED LOADING.

COMMON TOO LARGE - DEFINITION OF THIS COMMON BLOCK CAUSES COMMON TO

WRAP AROUND THROUGH ZERO. MOVING THE TOP OF COMMON - WITH THE COMMON COMMAND - MAY HELP.

COMMON OUT OF REACH - COMMON ABOVE :100000 IS OUT OF REACH OF THE CURRENT LOAD MODE (16S, 32S OR 32R). USE THE MODE COMMAND TO SET THE LOAD MODE TO 64R.

SECTORED LOAD MODE INVALID - A MODULE COMPILED OR ASSEMBLE TO LOAD IN "R" MODE HAS BEEN LOADED IN "S" MODE. USE THE MODE COMMAND TO RESET THE LOAD MODE. IT MIGHT BE A GOOD IDEA TO BE SURE THAT ALL MODULES ARE CORRECTLY WRITTEN AS THE DEFAULT LOAD MODE IS 32R.

64R LOAD MODE INVALID - A MODULE COMPILED OR ASSEMBLED TO RUN IN ONLY 32K OF MEMORY IS BEING LOADED IN 64R MODE. RECOMPILE OR REASSEMBLE OR CHANGE THE LOAD MODE WITH THE LOADER'S MODE COMMAND.

PROGRAM TOO LARGE - THE PROGRAM HAS LOADED INTO THE LAST LOCATION IN MEMORY AND HAS WRAPPED AROUND TO LOAD IN LOCATION 0. THE PROGRAM SIZE MUST BE DECREASED.

PROGRAM-COMMON OVERLAP - THE MODULE BEING LOADED IS ATTEMPTING TO LOAD CODE INTO AN AREA RESERVED FOR COMMON. USE THE LOADER'S COMMON COMMAND TO MOVE COMMON UP HIGHER.

SNAME XXXXXX NEED SECTOR ZERO LINK - AT LOCATION XXXXXX A LINK IS REQUIRED FOR DESECTORING THE INSTRUCTION. NO BASE AREAS ARE WITHIN REACH EXCEPT SECTOR ZERO. THE LAST REFERENCED SYMBOL WAS SNAME. THIS MESSAGE IS ONLY GENERATED WHEN THE SZ COMMAND HAS BEEN GIVEN. SNAME MAY BE THE NAME OF A COMMON BLOCK, THE NAME OF THE ROUTINE TO WHICH THE LINK SHOULD BE MADE, OR THE NAME OF THE MODULE BEING LOADED.

XXXXXX MULTIPLE INDIRECT - A MODULE LOADING IN 64R MODE REQUIRES A SECOND LEVEL OF INDIRECTION AT LOCATION XXXXXX. THIS MESSAGE USUALLY RESULTS WHEN AN ATTEMPT IS MADE TO LOAD CODE ASSEMBLED FOR 32R MODE IN 64R MODE. IT CAN ALSO HAPPEN IF BASE AREAS HAVE ACCIDENTALLY BEEN LOADED INTO WITH CODE AS THE RESULT OF A BAD LOAD STREAM.

SYMBOL NOT FOUND - AN ATTEMPT IS BEING MADE TO EQUATE TWO SYMBOLS WITH THE SYMBOL COMMAND AND THE "OLD" SYMBOL DOES NOT EXIST.

ALREADY EXISTS ! - AND ATTEMPT IS BEING MADE TO DEFINE A NEW SYMBOL, HOWEVER, THE SYMBOL NAME IS ALREADY A DEFINED SYMBOL IN THE SYMBOL TABLE.

SYMBOL UNDEFINED - AN ATTEMPT IS BEING MADE TO EQUATE TWO SYMBOLS, HOWEVER, THE "OLD" SYMBOL IS AN UNDEFINED SYMBOL IN THE SYMBOL TABLE.

CAN'T - PLEASE SAVE - THE EXECUTE COMMAND HAS BEEN GIVEN FOR A RUN FILE WHICH HAS REQUIRED VIRTUAL LOADING. SAVE THE RUNFILE AND THE GIVE THE EXECUTE COMMAND.

SYMBOL TABLE FULL - THE SYMBOL TABLE HAS GROWN DOWN TO LOCATION :4000. THE LAST BUFFER CANNOT BE ASSIGNED TO THE SYMBOL TABLE. REBUILD LOAD TO LOAD IN HIGHER MEMORY LOCATIONS (SEE ABOVE), OR REDUCE THE NUMBER OF SYMBOLS IN THE LOAD.

REFERENCE TO UNDEFINED COMMON - AN ATTEMPT IS BEING MADE TO LINK TO A COMMON NAME WHICH HAS NOT BEEN DEFINED. THIS USUALLY HAPPENS TO USERS WRITING THEIR OWN TRANSLATORS.

SNAME ILLEGAL COMMON REDEFINITON - AN ATTEMPT IS BEING MADE TO REDEFINE COMMON BLOCK SNAME TO A LONGER LENGTH. THE USER'S PROGRAM SHOULD BE EXAMINED FOR CONSISTENT COMMON DEFINITIONS. AT THE VERY LEAST THE LONGEST DEFINITION FOR A COMMON BLOCK SHOULD BE FIRST.

CAN'T DEFER COMMON, OLD OBJECT TEXT - THE DEFER COMMON COMMAND HAS BEEN GIVEN AND A MODULE CREATED WITH A PRE-REV 14 COMPILER OR ASSEMBLER HAS BEEN ENCOUNTERED. IT IS NOT POSSIBLE TO DEFER COMMON IN THIS CASE. THE MODULE MUST BE RECREATED WITH A REV 15 COMPILER OR ASSEMBLER.

XXXXXX NO POST BASE AREA, OLD OBJECT TEXT - A POST BASE AREA HAS BEEN SPECIFIED FOR MODULE WHICH WAS CREATED WITH A PRE-REV 14 COMPILER OR ASSEMBLER. NO BASE AREA IS CREATED. RECREATE THE OBJECT TEXT WITH A REV 15 COMPILER OR ASSEMBLER. THIS IS NOT A FATAL ERROR.

3 PRINCIPLES OF RELOCATING THE VIRTUAL LOADER

AS DELIVERED LOAD IS LOADED BETWEEN 122770 AND ABOUT 144000. THE STARTING ADDRESS FOR THIS VERSION IS 123000. ALL MEMORY BELOW 122770 IS USED FOR VIRTUAL BUFFERS AND THE SPACE FOR THE SYMBOL TABLE. BUFFERS INITIALLY OCCUPY ALL SPACE UP TO 122000. BUFFERS ARE REMOVED FROM THE TOP AS MORE SYMBOL TABLE SPACE IS REQUIRED.

THE LOAD POINT MAY BE CHANGED DOWNWARD IF NECESSARY TO FIT UNDER PRIMOS II OR UPWARDS TO INCREASE THE BUFFER AND SYMBOL TABLE SPACE. AS PRESENTLY DELIVERED, LOAD FITS UNDER 64K PRIMOS AND HAS ROOM FOR A COPY OF HPSD ABOVE IT UNDER THE OTHER PRIMOS'S.

A SAMPLE LOAD COMMAND FILE FOR LOADING LOAD FOLLOWS. THREE OF THE LINES IN THIS COMMAND FILE ARE UNDERLINED. TO RECONFIGURE LOAD, THE NUMBER - 122770, BELOW - SHOULD BE CHANGED BY THE AMOUNT THAT LOAD IS TO BE MOVED. VLOAD WILL HAVE A STARTING ADDRESS 10 (OCTAL) LOCATIONS ABOVE THE NUMBER SUPPLIED. THE VALUE SUPPLIED TO THE COMMON COMMAND SHOULD ALSO BE CHANGED BY A LIKE AMOUNT. FOR EXAMPLE IF 'CO 144000' IS REPLACED BY 'CO 141000' AND AND 122770 CHANGED TO 117770, LOAD WILL BE MOVED DOWN 3000 LOCATIONS AND WILL HAVE A STARTING ADDRESS OF 120000.

```
*     C_LOAD, LOAD, CEH, 08/13/77
*     COMMAND FILE TO BUILD LOAD - USUALLY
*     COPYRIGHT 1977, PRIME COMPUTER INC., FRAMINGHAM, MA.
*
```

```
PMA LOAD15 1/707
PMA_CREATB_1/707_121770
FTN VSUBRS 1/105707 1
FTN CMNFTN  1/105707 1
FTN MAPS 1/5707 1
PMA CMNPMA 1/707
FILMEM ALL
LOAD
CO 144000
SZ
ER 2
MO D64R
LO B_LOAD15_121770
LO B_CMNPMA
LO B_CREATB
LO B_CMNFTN * 10 0 10 4 6 6 12 10 177777
0 0 0 14 6 5 0 4 177777
0 0 6 12 6 5 6
LO B_MAPS * 0 20 22
LO B_VSUBRS * 0 16 0 4 4 16 10 24 177777
30 4 0 10
LO LIB>APPLIB * 0 10 5 3
SE * 2
LO LIB>FTNLIB * 0 10 4 3
MA M_SAVE
SA *LOAD
QU
DELETE B_LOAD15
DELETE B_VSUBRS
CO TTY
```

4 OLD LOADER/LOAD COMPATIBILITY ISSUES.

4.1 AFFECT OF THE TEMPORARY FILE

THE OPENING OF THE TEMPORARY FILE MAY CAUSE THE USER TO LOSE A TEMPORARY ATTACH POINT AS LOAD ATTACHES TO THE HOME UFD TO OPEN IT. CONSIDERATION WAS GIVEN TO OPENING THE TEMPORARY FILE AT OTHER TIMES (IF AND WHEN NEEDED, FOR EXAMPLE) AND/OR IN THE CURRENT UFD. BOTH OF THESE ALTERNATIVES WERE DISCARDED IN FAVOR OF HAVING THE TEMPORARY FILE OPENED IN A KNOWN PLACE AND AT A KNOWN TIME. USERS COUNTING ON A TEMPORARY ATTACH POINT WHEN RUNNING A COMMAND FILE, WILL FIND THOSE COMMAND FILES MUST BE CHANGED.

USERS WHO NARMALLY (OR OCCASIONALLY) EXIT FROM THE LOADER VIA BREAK OR CTL-P WILL LEAVE THE TEMPORARY FILE OPEN ON UNIT 4. THE FILE WILL HAVE TO BE DELETED BY THE USER AFTER IT HAS BEEN CLOSED.

4.2 AFFECT OF VIRTUAL LOADING

IF THE LOADER MUST RESORT TO BUFFERED LOADING (IE, MUST LOAD INTO ADDRESSES ABOVE THE TOP OF ITS BUFFER POOL) THE USER WILL HAVE TO SAVE THE LOADED OBJECT BEFORE IT CAN BE EXECUTED. USERS WHO MAKE USE OF THE LOAD AND EXECUTE FEATURE OF THE OLD LOADERS MAY BE AFFECTED IN THIS CASE.

4.3 AFFECT OF CHANGES TO THE FORMAT OF THE LOAD FAMILY OF COMMANDS

SPECIFICATION OF BASE AREAS FOR THE LOAD FAMILY OF COMMANDS HAS BEEN CHANGED. THE NEW METHODS ARE MORE NATURAL AND SAFER AS THE USER NEEDS TO KNOW LESS ABOUT THE ULTIMATE LOAD WITH THE NEW FORMS. HOWEVER, USERS HAVING COMMAND FILES USING THE OLD METHODS ARE AFFECTED.

4.4 LOCATION OF DEFAULT COMMON

WITH EACH OF THE COPIES OF THE OLD LOADER THE TOP OF COMMON WAS DETERMINED BY THE TOP OF THE LOADER. WITH LOAD THE TOP OF COMMON IS ARBITRARILY SET TO :100000 - THE MAXIMUM ALLOWED FOR 32R MODE LOADS. AS WITH THE OLD LOADER THE TOP OF COMMON MAY BE RESET BY EXECUTING A COPY OF LOAD AND SAVING AWAY THE EXECUTED COPY - WITH CHANGED TOP OF COMMON. IN THIS FASHION RUNFILES EQUIVALENT TO LOAD20, LOAD, HILOAD, ETC. CAN BE CREATED. IT IS HOPED, HOWEVER, THAT IN THE LONG RUN THE DEFER COMMON OPTION WILL BECOME THE STANDARD AND THE WHOLE CONCEPT OF "TOP OF COMMON" WILL DISAPPEAR.

4.5 SIZE OF LOAD

THE OLD LOADER OCCUPIES :4000 LOCATIONS COMPARED WITH OVER :22000 FOR LOAD. THIS MEANS THE RUN FILE OF LOAD WILL TAKE MORE ROOM ON THE DISK AND THAT THE WORKING SET AT RUN TIME IS GREATER. THE ROUTINES IN LOAD HAVE BEEN GROUPED TO REDUCE THE POTENTIAL FOR PAGING, BUT IT CAN NOT BE GUARANTEED THAT LOAD WILL NOT PAGE MORE THAN THE OLD LOADERS. CONSEQUENTLY USERS USING LOAD MAY FIND THAT IT CONTRIBUTES TO THRASHING ON SYSTEMS PRONE TO THIS PROBLEM. TESTS ON THE ENGINEERING SYSTEM INDICATE THAT THE WORKING SET OF LOAD IS ABOUT TWICE THAT OF THE OLD LOADER.

ON THE OTHER HAND, LOAD HAS A MORE SOPHISTICATED METHOD FOR HANDLING THE SYMBOL TABLE. TIMING TESTS ON THE ENGINEERING P400 INDICATE THAT LOAD IS AS FAST AS THE OLD LOADERS AT WORST AND UP TO 25% FASTER IF THE LOAD IS LARGE WITH A GREAT DEAL OF SYMBOL TABLE REFERENCING. THE FOLLOWING TABLE SHOWS THE RESULTS OF THE TIMING TESTS.

VERSION	DATE	TIME	CPU MIN	DISK MIN	TOTAL TM
RPG - HILOAD	02-13-77	10:55:32	0.226	0.038	0.264
VLOAD REV 14	02-13-77	10:55:56	0.281	0.034	0.314
LOAD REV 15	02-13-77	10:56:26	0.198	0.035	0.233

LIB'S - HILOAD	02-13-77	11:03:52	0.302	0.044	0.346
VLOAD REV 14	02-13-77	11:02:08	0.347	0.046	0.393
LOAD REV 15	02-13-77	11:03:05	0.215	0.050	0.265
FTN - HILOAD	02-13-77	11:06:49	0.153	0.053	0.205
LOAD REV 15	02-13-77	11:07:14	0.153	0.063	0.216
PMA - HILOAD	02-13-77	11:15:18	0.136	0.059	0.195
LOAD REV 15	02-13-77	11:15:54	0.145	0.052	0.197

THE FIRST GROUP OF THREE LINES REPRESENTS LOADING THE RPG RUNTIME LIBRARY, KIDALB AND THE FORTRAN LIBRARY. LOADING AN RPG PROGRAM WOULD BE SIMILAR.

THE SECOND GROUPING (LIB'S) REPRESENTS FORCE LOADING APPLIB AND FTNLIB. IN THIS CASE THE LARGEST NUMBER OF SYMBOLS IS GENERATED.

THE LAST TWO GROUPINGS REPRESENT LOADING THE FORTRAN COMPILER AND PMA RESPECTIVELY.

SUBJECT: EVENT LOGGING IN PRIMOS III AND IV

SEVERAL PROBLEMS IN THE -SPOOL AND -PURGE OPTIONS HAVE BEEN FIXED. DISK MOUNT (DSKNAM) ENTRIES ARE NOW PROCESSED. A '-I <TRNAME>' OPTION IS AVAILABLE ON THE COMMAND LINE. THE ECCC SYNDROME FOR 'LP' IS NOW INTERPRETED AS 'MB' -- MULTIBIT; BIT 6 OF DSWSTATL, WHICH USED TO BE 'LP' IS NOW 'OP' -- OVERALL PARITY. LOGREC QUOTA CHECKING CAN BE TURNED OFF WITH THE LOGREC CONFIGURATION

THE EVENT LOGGING MECHANISM FOR REV 14 OF PRIMOS III AND IV IS UNCHANGED FROM REV 23 THE LOGPRT PROGRAM, HOWEVER, HAS HAD SEVERAL ENHANCEMENTS ADDED. THESE ARE REFLECTED IN SECTION 2.3. NOTE IN PARTICULAR THAT THE OPERATION OF LOGPRT IS IDENTICAL UNDER PRIMOS II, III, AND IV WITH THE SINGLE EXCEPTION OF THE -SPOOL OPTION, WHICH IS NOT SUPPORTED UNDER PRIMOS II.

1 GENERAL INFORMATION

1.1 FIRST-LEVEL EVENT LOGGER -- LOGEV1

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

1.2 SECOND-LEVEL EVENT LOGGER -- LOGEV2

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

LOGEV2 DOES NOT DUMP LOGBUF IF THE FILE LOGREC DOES NOT EXIST IN CMDNCO OR IF THE CONFIGURATION COMMAND LOGREC HAS BEEN USED TO SET THE LOGREC QUOTA (SEE BELOW) TO A NEGATIVE VALUE (SEE PE-T-412). THIS ALLOWS OPERATION WITH A WRITE-PROTECTED

COMMAND DEVICE. (NOTE: IF THE COMMAND DEVICE IS WRITE-PROTECTED AND A LOGREC FILE EXISTS IN CMDNCO AND A 'LOGREC 177777' HAS NOT BEEN ISSUED, A DISK WRITE-PROTECT ERROR MESSAGE WILL BE PRINTED ON THE SYSTEM CONSOLE EVERY MINUTE.)

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

```
A CMDNCO 0 1
L LOGREC
C 2
A
```

THE SIZE OF LOGREC IS CONTROLLED BY A PARAMETER IN LOGEV2 AND IS CURRENTLY 4096 WORDS. IF LOGREC EXCEEDS THIS SIZE, LOGEV2 WILL PRINT THE MESSAGE 'EXCEEDING QUOTA ON LOGREC' ON THE SYSTEM CONSOLE UNLESS THE LOGREC QUOTA HAS BEEN SET TO 0 USING THE CONFIGURATION COMMAND LOGREC (SEE PE-T-412). IT WILL, HOWEVER, CONTINUE TO LOG INFORMATION TO LOGREC AND PRINT THE MESSAGE EVERY MINUTE UNTIL LOGREC IS EMPTIED BY THE LOGPRT ROUTINE (SEE BELOW). (ALTERNATIVELY, OF COURSE, LOGREC CAN BE DELETED.)

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

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

1.3 LOGPRT -- DUMP CONTENTS OF LOGREC

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

```
R LOGPRT [<DESTINATION>] [<OPT> <OPT> ...]
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

NEW SPECIFICATION:

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

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

INPUT TREENAME:

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

2 LOGPRT PROCESSING

LOGPRT OUTPUTS A HEADER LINE FOLLOWED BY FORMATTED ENTRIES, ONE OR MORE LINES PER ENTRY. THE FOLLOWING ENTRIES ARE CURRENTLY DEFINED. (ALL NUMBERS ARE OCTAL EXCEPT WHERE NOTED.)

09:01:20 WED 16 FEB 1977

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

COLD START

A COLD START OF PRIMOS WAS PERFORMED.

WARM START

A WARM START OF PRIMOS WAS PERFORMED.

MACHINE CHECK (XXX) DSWSTAT= SSSSSS SSSSSS DSWRMA= YYYYY RRRRRR RRRRRR DSWPB= PPPPPP PPPPPP

A MACHINE CHECK OCCURRED. DSWSTAT, DSWRMA, AND DSWPB CONSTITUTE THE DSW AT THE TIME OF THE CHECK. 'XXX' IS AN ENCODING OF THE MACHINE CHECK CODE AND 'NOT RCM PARITY' IN DSWSTATH AS FOLLOWS:

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

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

MISSING MEMORY DSWSTAT= ...

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

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

A MEMORY PARITY ERROR OCCURRED. 'XXXX' IS EITHER 'ECCC' (CORRECTED) OR 'ECCU' (UNCORRECTED). 'PPN,WN=PPPPPP WWWWW' IDENTIFIES THE PHYSICAL PAGE AND WORD NUMBER OF THE ERROR. FOR AN ECCC ERROR, THE PPN IS FOLLOWED BY 'BIT=XX', WHERE 'XX' IDENTIFIES THE BIT IN ERROR -- 1-15 FOR BITS 1-15, RP FOR RIGHT PARITY, LP FOR LEFT PARITY, C2, C4, C5 FOR OTHER CHECK BITS, MB FOR MULTIBIT, NE FOR NO ERROR. (THIS IS TAKEN FROM THE ECCC SYNDROME FIELD IN DSWSTATL.) FOLLOWING THE BIT IDENTIFICATION IS 'OP=X', WHERE X IS 0 OR 1 AND REFLECTS THE SETTING OF DSWSTATL BIT 6 (OVERALL PARITY).

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

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

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

DISK MOUNT: PACKNAME ON DVNO

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

MACHINE CHECK USER= NN PC= PPPPPP

THIS IS THE FORMAT OF A MACHINE CHECK MESSAGE ON A PRIME 300. USER GIVES THE USER NUMBER (DECIMAL), PC GIVES HIS PC AT THE TIME OF THE CHECK.

MEMORY PARITY

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

MEMORY PARITY PPN= PPPPPP WN= WWWWW CONTENTS= CCCCCC

THIS IS THE FORMAT OF AN ENTRY FOR A PRIME 300 MEMORY PARITY ERROR ENCOUNTERED DURING A WARM START MEMORY SCAN. GIVEN ARE THE PHYSICAL PAGE NUMBER (PPN), WORD NUMBER OFFSET IN THE PAGE (WN), AND INCORRECT CONTENTS.

MISSING MEMORY

THIS IS A PRIME 300 MISSING MEMORY CHECK ENTRY.

LOGBUF OVERFLOW -- NNNNN ENTRIES LOST

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

SHUTDOWN BY OPERATOR

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

*** LOGREC EMPTY ***

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

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

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

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

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

3 MODIFYING THE EVENT LOGGING MECHANISM

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

3.1 INCREASING THE SIZE OF LOGBUF

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

3.2 ADDING EVENT TYPES

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

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

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

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

3.2.1 EVENT MESSAGE FORMAT

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

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

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

3.2.2 CURRENTLY DEFINED EVENT TYPES

CURRENTLY, THE FOLLOWING EVENT TYPES ARE DEFINED.

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

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

3.2.3 CALLING LOGEV1 -- PRIMOS III

IN PMA:

```
CALL LOGEV1
DAC  MESSAGE
```

IN FORTRAN:

```
CALL LOGEV1(MESSAGE)
```

3.2.4 CALLING LOGEV1 -- PRIMOS IV

IN PMA, CODE INSIDE SEG4:

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

IN PMA, CODE OUTSIDE SEG4:

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

IN FORTRAN:

```
     CALL LOGEV1(MESSAGE)
```

3.2.5 MODIFYING LOGPRT

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

```
        TYPE=<TYPE> DATA=<WORD1> <WORD2> ... <WORD7>
```

(NOTE THAT ONLY 7 DATA WORDS ARE ALLOWED FOR THESE TYPES.) TO ADD A NEW TYPE, ADD A LABEL TO THE COMPUTED GOTO FOLLOWING STATEMENT $400. AT THE NEW LABEL (BETWEEN $1950 AND $2000), CALL THE STORE ROUTINE TO PERFORM THE REQUIRED FORMATTING.

THE CALLING SEQUENCE FOR STORE IS AS FOLLOWS:

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

TEXT	A TEXT STRING TO BE PRINTED.
TXTLEN	THE LENGTH IN CHARACTERS IN TEXT. IF ZERO, NO TEXT IS PRINTED.
ARRAY	AN ARRAY OF WORDS TO BE TRANSLATED AND ENTERED IN THE OUTPUT LINE. ENTRY(1) IS THE FIRST DATA WORD OF THE EVENT MESSAGE. ENTTYP AND ENTLEN CONTAIN THE TYPE AND LENGTH OF THE ENTRY.
NW	THE NUMBER OF WORDS IN ARRAY. IF ZERO, NO WORDS ARE TRANSLATED.
DEC	OCTAL/DECIMAL FLAG. IF ZERO, TRANSLATION IS TO OCTAL WITH NO LEADING ZERO SUPPRESSION. IF NON-ZERO, TRANSLATION IS TO DECIMAL WITH LEADING ZEROES SUPPRESSED.

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

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

TO REBUILD LOGPRT, RUN THE COMMAND FILE C_LOG IN SYSTEM>LOGUFD. THIS WILL CREATE A RUN FILE CALLED *LOG. *LOG CAN THEN BE MOVED TO CMDNCO AND RENAMED TO LOGPRT.

3.3 CHANGING THE SIZE OF LOGREC

THE SIZE OF LOGREC (OVER WHICH THE 'EXCEEDING...' MESSAGE IS PRINTED) IS DEFINED IN THE SOURCE FILE LOGEV2 BY THE INTEGER VARIABLE QUOTA (CURRENTLY 4096 WORDS). THIS CAN BE MODIFIED AS DESIRED (OR PATCHED IN PR0006).

3.4 CHANGING LOGPRT'S DEFAULT INPUT/OUTPUT FILENAMES

THE DEFAULT INPUT NAME -- '<0>CMDNCO>LOGREC' -- IS IN THE ARRAY INPNAM. THE SIZE OF INPNAM AND LENGTH OF THE NAME, INNAML, SHOULD BE SET TO THE NUMBER OF WORDS AND CHARACTERS IN INPNAM RESPECTIVELY. THE DEFAULT OUTPUT NAME (LOGLST) IS IN THE ARRAY OUTNAM, WHICH IS ALWAYS 16 WORDS LONG, BLANK PADDED.

DATE: JUNE 9, 1978

SUBJECT: BOOTING FROM A MAGSAV TAPE

1 OVERVIEW

IT IS NOW POSSIBLE TO BOOTSTRAP FROM ANY TAPE PRODUCED BY MAGSAV REV 14.1 OR LATER. THE BOOT PROGRAM, WHICH IS SAVED IN EACH LOGICAL TAPE HEADER RECORD, WILL LOAD EITHER A SPECIFIED FILE NUMBER (NUMBER 1, 2, ETC.) OR A FILE WITH AN OPERATOR-SPECIFIED TREENAME. TAPES PRODUCED BY THE NEW MAGSAV ARE READABLE BY ALL VERSIONS OF MAGRST. FILES TO BE LOADED MUST BE IN STANDARD SAVE-FILE FORMAT.

REV 1 MODIFICAIIONS: PE-T-375 REV 1 REFLECTS THE FOLLOWING MODIFICATIONS TO THE MAGTAPE BOOT. TWO NEW SWITCH SETTINGS (8 AND 9) HAVE BEEN DEFINED TO CONTROL RELOCATION OF THE BOOT PROGRAM. PROBLEMS INVOLVING LOADING OF PROGRAMS RESIDING ABOVE 32K HAVE BEEN FIXED. IN ADDITION, A NEW SECTION ON LOADING PRIMOS II, MAKE, AND MAGRST HAS BEEN ADDED.

THE MAGTAPE BOOT DESCRIBED HEREIN CORRESPONDS TO MAGSAV REV 15.

2 LOADING PROCEDURE

2.1 SENSE SWITCH DEFINITIONS

MOUNT THE TAPE ON ANY DRIVE, CONTROLLER 1 (FIRST CONTROLLER). THE TAPE SHOULD BE AT LOAD POINT (OR BEYOND), BUT DOES NOT HAVE TO BE ONLINE. SET THE SENSE SWITCHES AS FOLLOWS:

```
          1       8       16
<N<NNN<NNN<RR<CI-<101<          ('-' => DON'T CARE)
```

NNNNNNN BITS 1-7 SPECIFY THE NUMBER OF THE FILE ON THE LOGICAL TAPE TO BE LOADED, THE FIRST FILE BEING 1 (SEE SECTION 5.1). IF BITS 1-7 ARE ZERO, THE BOOT PROGRAM PROMPTS FOR A TREENAME AS DESCRIBED BELOW.

RR BITS 8 AND 9 CONTROL THE RELOCATION OF THE BOOT PROGRAM. THE BOOT WILL RELOCATE ITSELF, IF NECESSARY, SO THAT ITS ENDING ADDRESS IS AT, IF SWITCHES 8 AND 9 ARE:

- 00 - THE END OF PHYSICAL MEMORY
- 01 - 16K
- 10 - 32K

11 - 48K

S IF BIT 10 IS 0, THE LOADED PROGRAM IS AUTOMATICALLY STARTED. IF BIT 10 IS A 1, THE BOOT WILL HALT WITH '000001 IN THE DATA LIGHTS AFTER LOADING THE PROGRAM. HITTING START WILL THEN START THE PROGRAM.

C IF BIT 11 IS 1, THE BOOT PROGRAM WILL HALT AT '260 TO ALLOW THE SOC AND OPTION A CONTROL WORDS TO BE SET FOR A NON-10 CPS TERMINAL (SEE BELOW).

T IF BIT 12 IS 0, A 9-TRACK DRIVE IS ASSUMED. IF BIT 12 IS 1, A 7-TRACK DRIVE IS ASSUMED.

SWITCHES 8-10 AND 12 MUST BE LEFT IN POSITION UNTIL THE SELECTED PROGRAM HAS BEEN LOADED. THE OTHER SWITCHES CAN BE RESET ANY TIME AFTER THE TREENAME PROMPT HAS BEEN PRINTED OR THE TAPE SEARCH HAS STARTED.

2.2 TREENAME SPECIFICATION

IF SENSE SWITCHES 1-7 ARE ZERO, THE BOOT PROGRAM INITIALIZES THE SYSTEM ASR, PRINTS 'TREENAME=', AND WAITS FOR OPERATOR INPUT. (THE STANDARD VERSION OF THE BOOT PROGRAM ASSUMES A 10 CPS TERMINAL. THIS CAN BE MODIFIED AS DESCRIBED BELOW.)

THE FORMAT OF THE TREENAME IS A SERIES OF UP TO 8 VALID FILE SYSTEM NAMES SEPARATED BY '>'S WITH NO EMBEDDED BLANKS, E.G.:

```
*DISCT2-6
MFD>T&M>TEST
MFD>CMDNCO>STANDALONE
```

ENTERING A NULL LINE, QUESTION MARK, OR ANY CHARACTER WHOSE VALUE IS LESS THAN '240 (E.G., ALL CONTROL CHARACTERS) WILL CAUSE THE TREENAME PROMPT TO BE REPEATED. OTHERWISE NO CHECKING IS DONE ON THE VALIDITY OF THE TREENAME ENTERED.

3 BOOT OPERATIONS

THE BOOT PROGRAM FIRST SIZES MEMORY AND CHECKS SENSE SWITCH 11 TO SEE IF PATCHING OF THE CONSOLE SPEED SELECTION CONTROL WORDS IS DESIRED. IF SWITCH 11 IS UP, THE BOOT HALTS AT LOCATION '260 AND AND THE SPEED SELECTION WORDS CAN BE PATCHED AS DESCRIBED IN SECITION 5.4.

NEXT, SENSE SWITCHES 1-7 ARE CHECKED TO SEE IF A TREENAME PROMPT IS REQUIRED. IF IT IS (SWITCHES 1-7 ALL DOWN), THE ASR IS INITIALIZED USING THE B, X, AND KEYS FROM THE BOOT'S SAVE FILE VECTOR (A LA DISK BOOT); THE TREENAME PROMPT IS PRINTED; AND A

TREENAME IS READ. THE BOOT PROGRAM THEN SEARCHES TO THE END OF THE LOGICAL TAPE FOR EITHER FILE NNNNNNN OR THE SPECIFIED TREENAME. IF THE FILE IS NOT FOUND, THE BOOT HALTS WITH '000005 IN THE DATA LIGHTS.

WHEN THE REQUESTED FILE IS FOUND, THE FIRST DATA BLOCK IS READ AND THE SAVE VECTOR EXAMINED. IF NECESSARY, THE BOOT RELOCATES ITSELF OUT OF THE WAY OF THE PROGRAM BEING LOADED. IF THIS IS IMPOSSIBLE, THE BOOT HALTS WITH '000003 IN THE DATA LIGHTS. THE PROGRAM TO BE LOADED MUST HAVE A STARTING ADDRESS ABOVE '3440 OR AN ENDING ADDRESS BELOW THE MEMORY SIZE MINUS '2663, E.G., '175115 FOR A 64K SYSTEM. THE EFFECTIVE MEMORY SIZE CAN BE ADJUSTED MODULO 16K USING SENSE SWITCHES 8 AND 9.

IF AT ANY TIME THE BOOT ENCOUNTERS AN ERROR READING FROM THE TAPE, IT WILL HALT WITH THE STATUS WORD IN THE DATA LIGHTS.

FINALLY, IF THE PROGRAM IS LOADED WITHOUT ERROR, THE BOOT LOADS A, B, X, AND THE KEYS AND CHECKS SWITCH 10. IF ON, THE BOOT HALTS WITH '000001 IN THE DATA LIGHTS. IF NOT ON, THE PROGRAM IS JMPED TO.

4 SUMMARY OF HALT CODES (IN DATA LIGHTS)

000001 (HALT AT '624 OR '175164) THE REQUESTED PROGRAM HAS BEEN LOADED, AND AUTO-START HAS BEEN SUPPRESSED (SWITCH 10 UP). HITTING START WILL RESULT IN CONTROL BEING PASSED TO THE LOADED PROGRAM. (NOTE: AT THE HALT, A, B, X, ETC. HAVE BEEN LOADED FROM THE PROGRAM'S SAVE VECTOR.)

000002 (HALT AT '506) THE TREENAME RECORD FOR THE REQUESTED FILE WAS NOT FOLLOWED BY A DATA RECORD.

000003 (HALT AT '627) THE REQUESTED PROGRAM CANNOT BE LOADED BY THE BOOT PROGRAM: ITS SA IS LESS THAN '3440 AND ITS EA IS ABOVE THE MEMORY SIZE MINUS '2663.

000005 (HALT AT '442) THE END OF THE LOGICAL TAPE WAS ENCOUNTERED BEFORE THE REQUESTED PROGRAM WAS FOUND.

000021 (HALT AT '260) SWITCH 11 IS UP, CAUSING THE BOOT TO HALT SO THAT THE SOC AND OPTION A CONTROL WORDS CAN BE PATCHED FOR A NON-10 CPS TERMINAL (SEE BELOW).

XXXXXX (HALT AT '1) A TAPE READ ERROR OR UNEXPECTED STATUS CONDITION OCCURRED. THE DATA LIGHTS DISPLAY THE LAST STATUS WORD READ FROM THE CONTROLLER. STATUS BIT DEFINITIONS ARE:

100000 - PARITY ERROR.

040000 - RUNAWAY TAPE.
020000 - CRC ERROR.
010000 - LRC ERROR.
004000 - LOW DMX RANGE.
002000 - PERMANENT ERROR.
001000 - READ-AFTER-WRITE ERROR.
000400 - FILE MARK DETECTED.
000200 - READY (OK).
000100 - ONLINE (OK).
000040 - END OF TAPE DETECTED.
000020 - REWINDING.
000010 - BEGINNING OF TAPE.
000004 - PROTECTED (OK).
000002 - OVERRUN.
000001 - REWIND COMPLETE.

5 SPECIAL CONSIDERATIONS

5.1 CALCULATING TAPE FILE NUMBER

THE BEST WAY TO CALCULATE THE POSITION OF A FILE ON TAPE IS TO INDEX THE TAPE WHEN RUNNING MAGSAV. THE NUMBER OF EACH FILE CORRESPONDS TO THE ORDER OF THE FILES PRINTED IN THE INDEX. NOTE THAT IF SUBUFDS ARE BEING SAVED, EACH SUBUFD ITSELF HAS A FILE ON THE TAPE.

5.2 LOADING FROM A P300 7-TRACK TAPE

THE MAGTAPE BOOT PROGRAM MUST BE MANUALLY STARTED IF IT IS LOADED FROM A 7-TRACK TAPE GENERATED ON A P300. AFTER THE MACHINE HALTS FOLLOWING THE LOAD FROM TAPE, START AT '400. (LOAD '400 INTO LOCATION '7 AND RUN.)

5.3 LOADING FROM A LOGICAL TAPE OTHER THAN THE FIRST

TO LOAD FROM A LOGICAL TAPE OTHER THAN THE FIRST, THE FIRST LOGICAL TAPE MUST BE SKIPPED OVER. THIS CAN BE DONE BY SPECIFYING A NON-EXISTENT FILE NUMBER OR TREENAME. AFTER THE BOOT HALTS WITH '000005 IN THE DATA LIGHTS, RESET THE SWITCHES (IF NECESSARY) AND RESTART AI '400.

5.4 RUNNING ON A NON-10 CPS CONSOLE

IF THE SYSTEM CONSOLE IS TO BE USED TO ENTER A TREENAME AND THE CONSOLE WILL NOT RUN AT 10 CPS AND THE BOOT PROGRAM HAS NOT BEEN MODIFIED TO THE APPROPRIATE SPEED (SEE BELOW), RAISE SWITCH 11 BEFORE STARTING THE BOOT PROCEDURE. THIS WILL CAUSE THE BOOT TO HALT AFTER SIZING MEMORY (AT LOCATION '260) WITH '21 IN THE DATA LIGHTS. LOCATIONS '220-'222 CAN THEN BE PATCHED AS FOLLOWS:

SPEED	'220	'221	'222
110 BAUD	110	27	74000
300 BAUD	1010	76	34000
1200 BAUD	2010	373	34000
9600 BAUD	3410	3735	34000

AFTER PATCHING THE BOOT, SELECT RUN AND HIT START TO CONTINUE.

6 MODIFYING MAGSAV OR BOOT PARAMETERS

6.1 MAGSAV MODIFICATIONS

THE MAGTAPE BOOT PROGRAM IS INCORPORATED INTO THE MAGSAV RUNFILE -- *MAGSAV -- BY RUNNING THE PROGRAM *LBOOT IN MTUFD. THIS RESTORES *MAGSAV, OVERLAYS *MTBOOT, AND SAVES *MAGSAV. THIS IS DONE AUTOMATICALLY BY RUNNING C_MAGSAV OR C_SAVLOAD.

6.2 MODIFYING THE BOOT PROGRAM

THE SOURCE OF THE BOOT PROGRAM IS MTBOOT. A NEW RUNFILE CAN BE GENERATED BY RUNNING C_MTBOOT. TO CHANGE THE SPEED OF THE TERMINAL INITIALIZED BY THE BOOT PROGRAM TO 30 CPS, ENTER THE COMMANDS:

```
REST *MTBOOT
SA *MTBOOT 4/1010 76 34000
R *LBOOT
```

THIS WILL GENERATE A *MAGSAV INCORPORATING THE NEW *MTBOOT. (PARAMETERS FOR OTHER SPEEDS ARE AS NOTED IN SECTION 5.4.)

6.3 MAGSAV OUTPUT BUFFER DEFINITION IN LBOOT

LBOOT ASSUMES THAT THE OUTPUT BUFFER FOR THE LOGICAL TAPE HEADER RECORD IN MAGSAV IS LOCATED AT '4000 (DEFINED IN THE SVRSTR MODULE). IF THE LOCATION OF THIS BUFFER -- OUTBUF -- MOVES, THE PARAMETER OUTBUF IN LBOOT SHOULD BE CHANGED APPROPRIATELY TO BE LOC(OUTBUF)+'14.

7 BOOTING PRIMOS II, MAKE, MAGRSI, ETC.

THE PROCEDURES DESCRIBED ABOVE ARE SUFFICIENT TO COVER THE LOADING OF MOST STANDALONE PROGRAMS (E.G., T&M'S). THEY ARE INSUFFICIENT FOR MORE COMPLICATED OPERATIONS SUCH AS COMPLETELY RESTORING A DISK FROM TAPE. THESE COMPLICATIONS ARISE FROM TWO SOURCES. FIRST, PRIMOS II, AS DISTRIBUTED ON THE MASTER DISK, IS IN A FORM THAT EXPECTS TO BE RELOCATED BY THE DISK BOOT PROGRAM. SINCE THE MAGTAPE BOOT DOES NOT PERFORM THIS RELOCATION, IT MUST BE DONE MANUALLY BEFORE PRIMOS II IS PLACED ON THE TAPE. SECOND, TO MAKE, FIXRAT, OR MAGRST A DISK, PRIMOS II ALONE IS INSUFFICIENT -- THE APPROPRIATE UTILITY MUST BE BOOTED WITH PRIMOS II OR AFTER PRIMOS II HAS BEEN BOOTED. BOTH THESE METHODS ARE DESCRIBED BELOW.

7.1 A POSSIBLE SCENARIO

ASSUME THAT IT IS DESIRED TO CREATE A UFD THAT CONTAINS A MINIMAL SET OF FILES NECESSARY TO BRING A SYSTEM UP COMPLETELY FROM TAPE. THE FOLLOWING IS ONE POSSIBLE SCENARIO. (COMMENTS ARE IN UPPER/LOWER CASE AND OFFSET BY '/*'; USER INPUT IS IN LOWER CASE.)

```
OK, CREATE WORK
OK, A WORK 0 2
OK, FUTIL
GO
> F DOS
> C *DOS32,RDOS64,*DOS64    /* SEE PE-T-366 FOR DESCRIPTIONS OF
                            /* THE VARIOUS VERSIONS OF DOS.
> F CMDNCO
> C MAKE,MAGRST,FIXRAT,COPY,FUTIL

/* YOU COULD GET ALL OF CMDNCO HERE IF DESIRED.

> QU

/* RELOCATE *DOS32 USING HPSD.

OK, REST *DOS32   /* THIS IS A 32K VERSION OF R-DOS
OK, P
```

```
SA,EA,P,A,B,X,K=
4000 17775 71000 60000 0 0 4000

PB,SB,LB,XB:
64000/71000 0/0 0/0 0/0
OK, HPSD
GO

$C 4000 17775 64000  /* '64000 IS THE "LOW" FROM PE-T-366

$QU

OK, REST MAKE   /* RESTORE MAKE BELOW THE RELOCATED *DOS32
OK, P
SA,EA,P,A,B,X,K=
66 32000 1000 1 0 177764 24100

PB,SB,LB,XB:
64000/1000 0/0 0/0 0/0
OK, SAVE DOSMAK 66 77777 71000 60000 0 0 4000

/* THE FILE 'DOSMAK' CONTAINS BOTH PRIMOS II AND MAKE.
/* IT CAN BE LOADED AS DESCRIBED IN THE NEXT SECTION.
/* NEXT, DO THE SAME THING FOR MAGRST.

OK, REST MAGRST
OK, P
SA,EA,P,A,B,X,K=
66 35145 1000 0 0 0 6000

PB,SB,LB,XB:
64000/1000 0/0 0/0 0/0
OK, SAVE DOSRST 66 77777 71000 60000 0 0 4000

/* THE NEXT FEW COMMANDS MOVE A 64K VERSION OF R-DOS TO ITS
/* ACTUAL HOME IN MEMORY. NOTE THAT Q-DOS IS PROBABLY A BETTER
/* CHOICE ON 64K SYSTEMS (IT SUPPORTS NEW PARTITIONS).

OK, REST RDOS64
OK, P
SA,EA,P,A,B,X,K=
4000 17775 171000 160000 0 0 4000

PB,SB,LB,XB:
64000/171000 0/0 0/0 0/0
OK, HPSD
GO

$C 4000 17775 164000

$QU

OK, SAVE RDOS64 164000 177777 171000 160000 0 0 4000
```

```
/* NEXT WE RELOCATE Q-DOS.

OK, REST *DOS64
OK, P
SA,EA,P,A,B,X,K=
10000 57540 161000 120000 0 0 4000

PB,SB,LB,XB:
64000/161000 0/0 0/0 0/0
OK, PSD    /* NOTE: DON'T USE HPSD HERE
GO

$C 10000 57540 130000

$QU

OK, SA QDOS64 130000 177777 161000 120000 0 0 4000

/* NOW MOVE THE CONTENTS OF WORK TO TAPE.

OK, AS MT0
OK, MAGSAV
GO
REV. 14.1
TAPE UNIT (9 TRK): 0
ENTER LOGICAL TAPE NUMBER: 1
TAPE NAME: WORK
DATE (MM DD YY):
REV NO: 14
NAME OR COMMAND: $I INDEX
NAME OR COMMAND: *
*** START OF SAVE ***
*** END OF SAVE ***

/* AT THIS POINT YOU COULD ATTACH TO AND SAVE ANOTHER UFD.
/* ALTERNATIVELY, A SECOND LOGICAL TAPE COULD BE CREATED.

NAME OR COMMAND: $R

OK, MAGRST    /* CHECK THE TAPE
GO
REV. 14.1
YOU ARE NOT ATTACHED TO AN MFD
TAPE UNIT (9 TRK): 0
ENTER LOGICAL TAPE NUMBER: 1
NAME: WORK
DATE(MM DD YY): 12-02-77
REV NO:     14
REEL NO:      1
READY TO RESTORE: NW
*** STARTING INDEX ***
*DOS32
RDOS64
```

```
*DOS64
MAKE
MAGRST
FIXRAT
COPY
FUTIL
DOSMAK
DOSRST
QDOS64
*** END LOGICAL TAPE ***
*** INDEX COMPLETE ***

OK, U MTO
OK,
```

NOTE IN THE ABOVE THAT WE DID NOT BUILD A 64K VERSION OF DOSMAK OR DOSRST. THIS IS BECAUSE THE COMBINED SA, EA OF THE RESULTING MEMORY IMAGE WOULD LEAVE NO PLACE FOR THE MAGTAPE BOOT PROGRAM TO RELOCATE ITSELF. THEREFORE, WHEN USING A 64K VERSION OF PRIMOS II (RDOS64 OR QDOS64), BOOTING IN MAKE, MAGRST, OR OTHER UTILITY MUST BE DONE AS A SEPARATE OPERATION. THIS IS DESCRIBED BELOW.

7.2 LOADING AND RUNNING MAKE

THERE ARE TWO WAYS OF LOADING MAKE -- USING THE DOSMAK FILE GENERATED ABOVE OR LOADING A 64K VERSION OF PRIMOS II, THEN LOADING MAKE. BOTH PROCEDURES WILL ALLOW CREATION OF AN OLD OR NEW PARTITION. USING DOSMAK IS SOMEWHAT SIMPLER, SINCE ONLY ONE FILE NEED BE LOADED FROM TAPE. ON THE OTHER HAND, ONCE A 64K VERSION OF PRIMOS II IS LOADED, IT CAN BE USED FOR MULTIPLE OPERATIONS (E.G., A SUBSEQUENT MAGRST). REMEMBER THAT DOSMAK IS BASED IN R-DOS, WHICH DOES NOT UNDERSTAND NEW PARTITIONS. THE FOLLOWING DEMONSTRATES BOTH METHODS.

```
/* MOUNT TAPE, SET SWITCHES TO '000005, MASTER CLEAR AND LOAD.

TREENAME=DOSMAK

PRIMOS II REV 14.0 09/26/77 (AT 070000)


OK: S OLD 1000   /* OMIT 'OLD' TO CREATE A NEW PARTITION
GO
MAKE, REV 14.1.
BUILDING OLD PARTITION.
PHYSICAL DISK: 50
...
/* CONTINUE WITH MAKE
...
DISK CREATED
```

```
OK:

USING  A 64K VERSION OF PRIMOS II, THE SEQUENCE IS AS FOLLOWS:

/* MOUNT TAPE, ETC. AS ABOVE.

TREENAME=QDOS64    /* THIS THE RELOCATED VERSION OF *DOS64

PRIMOS II REV 14.1 11/08/77 (AT 170000)

OK:

/* NOW WE HAVE TO LOAD IN MAKE. SET THE SWITCHES TO '000505.
/* THIS WILL CAUSE THE BOOT TO RELOCATE TO 32K (BELOW QDOS64)
/* AND HALT BEFORE STARTING THE LOADED PROGRAM.
/* MASTER CLEAR AND LOAD.

TREENAME=MAKE

/* THE BOOT WILL HALT WITH '000001 IN THE DATA LIGHTS
/* AFTER MAKE IS LOADED. MASTER CLEAR AND START AT '170000.

OK: S 1000     /* START UP MAKE FROM QDOS64
GO
MAKE, REV 14.1.
BUILDING NEW PARTITION
...
/* CONTINUE WITH MAKE
...
DISK CREATED

OK:
```

7.3 RUNNING MAGRST

LOADING AND RUNNING MAGRST PROCEDES IN A MANNER AS DESCRIBED ABOVE FOR MAKE WITH TWO EXCEPTIONS: (1) IF A NEW PARTITION WAS CREATED, DOSRST CANNOT BE USED, SINCE IT IS RUNNING UNDER R-DOS; (2) IF A 64K VERSION OF PRIMOS II WAS USED ABOVE, IT NEED NOT BE RELOADED. FOR EXAMPLE, CONTINUING WITH THE LATTER EXAMPLE FROM ABOVE, THE SEQUENCE WOULD BE:

```
/* MASTER CLEAR, SET SWITCHES TO '000505, LOAD.

TREENAME=MAGRST

/* THE BOOT HALTS WITH '000001 IN THE DATA LIGHTS.
```

```
/* MASTER CLEAR AND START AT '170000 TO GET BACK INTO QDOS64.

OK: STARTU 50     /* START DISK JUST CREATED
OK: A DOS    /* FIRST WANT TO GET BOOTABLE DOS'S FROM TAPE
OK: S 1000     /* START UP MAGRST
REV. 14.1
YOU ARE NOT ATTACHED TO AN MFD
TAPE UNIT (9 TRK): 1
(TAPE NOT AT LOAD POINT)
ENTER LOGICAL TAPE NUMBER: 1
NAME: WORK
DATE(MM DD YY): 12-05-77
REV NO:       14
REEL NO:       1
READY TO RESTORE: $I
READY TO RESTORE: YE
*** STARTING RESTORE ***
*DOS32
RDOS64
*DOS64
MAKE
MAGRST
FIXRAT
COPY
FUTIL
DOSMAK
DOSRST
QDOS64
*** END LOGICAL TAPE ***
*** RESTORE COMPLETE ***

OK:
```

AT THIS POINT THE UFD DOS CONTAINS BOOTABLE VERSIONS OF PRIMOS II -- *DOS32 AND *DOS64. FUTIL CAN NOW BE RUN TO MOVE MAKE, MAGRST, ETC. TO CMDNCO. (ALTERNATIVELY, TWO PARTIAL RESTORES CAN BE DONE DIRECTLY INTO DOS AND CMDNCO.)

7.4 CONSOLE SPEED SELECTION FOR PRIMOS II

AT REV 14 ALL VERSIONS OF PRIMOS II PICK UP THE CONSOLE SPEED SELECTION FROM THE DISK BOOT. TO SET THE CONSOLE SPEED TO OTHER THAN 10 CPS WHEN LOADING FROM TAPE, RAISE SWITCH 10 PRIOR TO LOADING PRIMOS II. THE MAGTAPE BOOT WILL HALT BEFORE STARTING PRIMOS II WITH '000001 IN THE DATA LIGHTS. THEN LOAD '4000 INTO THE A-REGISTER AND SET LOCATIONS '1004, '1005, AND '1006 TO THE THREE CONTROL WORDS LISTED IN SECTION 5.4. (NOTE: UNDER PRIMOS II, THE LOW-ORDER 8 BITS OF '1006 ALSO GIVES THE NUMBER OF DELAYS TO USE AFTER CARRIAGE RETURNS.) HITTING START WILL THEN CAUSE PRIMOS II TO RESET THE CONSOLE TO THE DESIRED SPEED. THIS PROCEDURE IS NECESSARY EACH TIME PRIMOS II IS RELOADED FROM TAPE (BUT NOT WHEN STARTING AT THE PRIMOS RESTART ADDRESS).

DATE: MARCH 9, 1978

SUBJECT: CHANGES TO MAGSAV/MAGRST FOR REV 15

1. THE REV 15 VERSIONS OF MAGSAV AND MAGRST HAVE IMPROVED RECOVERY FROM TAPE IO ERRORS. THE REPORTING OF TAPE ERRORS HAS ALSO BEEN CHANGED. IF RECOVERABLE ERRORS OCCUR ON THE TAPE, THE TOTAL NUMBER OF SUCH ERRORS WILL BE PRINTED WHEN THE END OF THE TAPE IS REACHED. A SEPARATE ERROR MESSAGE FOR EACH RECOVERABLE ERROR WILL NO LONGER BE PRINTED.

2. KEYWORDS HAVE BEEN ADDED TO THE COMMAND LINE FOR SPECIFYING LONG RECORDS AND 7 TRACK TAPE FORMAT. THE REGISTER SETTINGS THAT WERE PREVIOUSLY REQUIRED FOR THESE FEATURES ARE STILL SUPPORTED.

3. AN OPTION TO ALLOW INCREMENTAL SAVING OF FILES HAS BEEN ADDED. THE DUMPED SWITCH IN THE UFD ENTRY IS USED FOR THIS PURPOSE. WHEN A FILE IS MODIFIED, ITS DUMPED SWITCH IS TURNED OFF. WHEN THE UPDATE OPTION (-UPDT) IS USED WITH MAGSAV, THE DUMPED SWITCH IS SET ON FOR EACH FILE OR DIRECTORY THAT IS SAVED. IF THE MAGSAV PROGRAM IS RUN WITH THE INCREMENTAL SAVE OPTION (-INC), ONLY FILES THAT HAVE A ZERO DUMPED SWITCH WILL BE SAVED. (I.E. ONLY FILES THAT HAVE BEEN MODIFIED SINCE THE LAST TIME THE MAGSAV PROGRAM WAS RUN, WILL BE SAVED.)

MAGSAV - ADDITIONAL COMMAND LINE ARGUMENTS.

-LONG USE A 1024 WORD RECORD SIZE. DEFAULT SIZE IS 512 WORDS PER RECORD.

-7TRK USE 7 TRACK TAPE FORMAT. DEFAULT IS 9 TRACK.

-UPDT UPDATE. THE DUMPED SWITCH IN THE UFD ENTRY WILL BE SET FOR FILES AND DIRECTORIES THAT ARE SAVED. DEFAULT IS NOT TO SET THE DUMPED SWITCH.

-INC INCREMENTAL DUMP. ONLY FILES AND DIRECTORIES THAT HAVE A ZERO DUMPED SWITCH WILL BE SAVED. DEFUALT IS TO SAVE ALL FILES AND DIRECTORIES.

MAGSAV - ADDITIONAL ACTION COMMANDS.

$UPDT ON TURN ON UPDATE. THE DUMPED SWITCH WILL BE SET FOR ALL

	FILES AND DIRECTORIES SAVED. SAME AS THE OPTION -UPDT.
$UPDT OFF	TURN OFF UPDATE.
$INC ON	TURN ON INCREMENTAL DUMP. SAME AS THE OPTION -INC
$INC OFF	TURN OFF INCREMENTAL DUMP.

MAGRST - ADDITIONAL COMMAND LINE ARGUMENTS.

-7TRK USE 7 TRACK TAPE FORMAT. DEFAULT IS 9 TRACK.

DATE: JUNE 9, 1978

SUBJECT: REV 15 PMA

1 SCOPE

THIS DOCUMENT DESCRIBES THE CHANGES MADE TO THE VERSION OF THE PRIME MACRO ASSEMBLER AVAILABLE AT REVISION 15.

2 SYNOPSIS

AMONG THE MODIFICATIONS MADE TO PMA AT THIS REVISION ARE:

- ERROR DIAGNOSTIC ON 64V INSTRUCTIONS WHEN ASSEMBLING WITH C64R
- ERROR SUMMARY AT END OF LISTING
- ABBREVIATED CONCORDANCE
- CORRECTED SOURCE LINE NUMBERING IN LISTING FILE

3 IMPROVED ERROR HANDLING

AN 'S' ERROR IS NOW GENERATED ON 64V MODE OPCODES WHEN ENCOUNTERED IN A SOURCE FILE WHERE C64R HAS BEEN SPECIFIED. A PROGRAMMER WHO CODES IN BOTH 64R AND 64V MODES MIGHT UNCONSCIOUSLY WRITE 64V-ONLY INSTRUCTIONS (SUCH AS TAX OR BRANCHES) IN R MODE. THIS ERROR CONDITION WAS NOT PREVIOUSLY DETECTED BY THE ASSEMBLER.

THE ERROR BACKTRACK FEATURE INTRODUCED AT REV 13 HAS BEEN REPLACED BY AN ERROR SUMMARY AT THE END OF THE LISTING FILE. THE SUMMARY LISTS THE LINE NUMBER AND ERROR DIAGNOSTIC FOR EACH LINE CONTAINING AN ERROR. IF A LINE CONTAINS MULTIPLE ERRORS, EACH ERROR DIAGNOSTIC IS PRINTED.

4 ABBREVIATED CONCORDANCE

IT IS NOW POSSIBLE TO OMIT FROM THE CONCORDANCE ALL NON-RELATIVE MODE SYMBOLS WHICH HAVE BEEN DEFINED BUT NOT OTHERWISE REFERENCED. THIS FEATURE IS CONVENIENT WHEN USING SYSCOM INSERT FILES TO INCLUDE ONLY THOSE SYMBOLS ACTUALLY REFERENCED WITHIN THE PROGRAM IN THE CONCORDANCE.

TO USE THIS FEATURE, SPECIFY THE -XREFS OPTION ON THE PMA COMMAND LINE. TO MAKE THIS FEATURE THE INSTALLATION DEFAULT, RESTORE PMA, OBTAIN THE CURRENT A-REGISTER SETTING WITH A PM COMMAND, AND INCLUSIVELY OR INTO THIS VALUE '10000 (I.E. SET BIT 4). TO OBTAIN A FULL CONCORDANCE IF ABBREVIATED CONCORDANCE IS THE INSTALLATION DEFAULT, SPECIFY THE -XREFL

COMMAND LINE OPTION.

5 SOURCE LINE NUMBERING

PRIOR TO THIS RELEASE, THE ASSEMBLER RESET SOURCE LINE NUMBERS FOR EACH MODULE WITHIN ONE SOURCE FILE. ASSEMBLY LISTINGS AND ERROR DIAGNOSTICS OF SECOND AND LATER MODULES WITHIN THE SOURCE CONTAINED INCORRECT SOURCE LINE NUMBERS. THIS PROBLEM HAS BEEN CORRECTED AT REV 15.

ABSTRACT

THIS DOCUMENT DESCRIBES THE CHANGES MADE TO PRIMOS III FOR REVISION 15. BOTH USER AND OPERATOR VISIBLE MODIFICATIONS ARE DESCRIBED.

1 CONFIGURATION AND OPERATIONAL MODIFICATIONS

1.1 BUILDING PRIMOS III

THE BUILD PROCEDURES FOR PRIMOS III, REV 15 ARE UNCHANGED FROM REV 14.

1.2 VERSIONS OF PRIMOS III

PRIMOS III IS DISTRIBUTED IN TWO VERSIONS, THE CHARACTERISTICS OF WHICH ARE AS FOLLOWS:

1) 16 USER VERSION WITH NETWORK SUPPORT FOR THE IPC CONTROLLER.

2) 32 USER VERSION WITHOUT ANY NETWORK SUPPORT.

2. FILE SYSTEM CHANGES

2.1 SGDR$$ - SEGMENT DIRECTORY MANIPULATION -NEW KEYS

TWO NEW KEYS ARE RECOGNIZED BY SGDR$$. THEY ARE SIMILAR TO K$SPOS. THE KEY K$FULL MOVES THE FILE TO THE NEXT NON-EMPTY ENTRY AND K$FREE MOVES TO THE NEXT VACANT ENTRY.

K$FULL MOVE THE FILE POINTER OF FUNIT TO THE POSITION GIVEN BY THE VALUE OF ENTRYA.
IF THE POSITION CONTAINS A FILE,
SET ENTRYB TO THE VALUE OF ENTRYA.
IF THE POSITION IS EMPTY, SEARCH FOR THE FIRST NON-EMPTY E
NTRY FOLLOWING
THE POSITION SPECIFIED.
IF A NON-EMPTY ENTRY EXISTS, SET ENTRYB TO THE POSITION OF THAT ENTRY.
IF THE EOF IS REACHED AND A ENTRY WITH A FILE HAS NOT BEEN FOUND, THEN
RETURN -1 IN ENTRYB.
IF EOF IS REACHED ON K$FULL, THE FILE POINTER IS LEFT AT E
OF.

K$FREE ACT IN THE SAME MANNER AS K$FULL, BUT FIND AN ENTRY THAT D
OES NOT

CONTAIN A FILE.

2.2 RDEN$$ -POSITION IN OR READ FROM A UFD -NEW KEYS

IT IS NOW POSSIBLE TO GET THE ENTRY INFORMATION FOR A SPECIFIED NAME USING RDEN$$.

K$NAME POSITION TO THE START OF THE ENTRY SPECIFIED BY NAME AND NAMLEN. READ AS MUCH OF THE ENTRY AS WILL FIT INTO BUFFER.
SET RNW TO THE NUMBER OF WORDS READ.
IF THE ENTRY IS NOT IN THE DIRECTORY, THE CODE E$FNTF IS RETURNED.

2.3 SATR$$ - SET A FILE'S ATTRIBUTES

THE HANDLING OF CHANGES TO THE DATE-TIME-MODIFIED FIELD AND THE DUMPED FLAG BY SATR$$ HAS BEEN CHANGED TO MATCH THE SUPERVISOR'S USE OF THESE FIELDS.
WHEN THESE FIELDS ARE CHANGED FOR AN ENTRY, THE DATE-TIME-MODIFIED FIELD OF
THE UFD CONTAINING THAT ENTRY WILL NOT BE CHANGED.

WHEN THE PROTECTION ATTRIBUTES OF THE FILE ARE CHANGED.
THE DATE-TIME-MODIFIED AND THE DUMPED BIT OF THE PARENT IS UPDATED.
THE DUMPED BIT FOR THE FILE WILL ALSO BE TURNED OFF.
THIS CHANGE IS TEMPORARY. IT IS BEING MADE BECAUSE THE MAGSAV PROGRAM ONLY DUMPS THE ATTRIBUTES OF A FILE WHEN THAT FILE IS DUMPED.

2.4 CNAM$$ - CHANGE NAME OF A FILE

WHEN THE NAME OF A FILE IS CHANGED, THE DUMPED BIT OF THE PARENT AS WELL AS OF THE FILE ARE UPDATED. THIS CHANGE IS ALSO TEMPORARY, AND HAS BEEN PLACED THERE BECAUSE OF THE WAY THE MAGSAV PROGRAM HANDLES INCREMENTAL DUMPS.

2.5 DUPLX$ MODIFICATION

THE PRIMOS DUPLX$ SVC IS NOW A FUNCTION THAT RETURNS THE TERMINAL CONFIGURATION WORD AND INTERNAL BUFFER NUMBER AS THE VALUE OF THE FUNCTION. IN ADDITION, IF THE KEY PASSED TO DUPLX$ IS EQUAL TO -1, NO UPDATING OF THE CONFIGURATION WORD TAKES PLACE -- THE CURRENT VALUE IS JUST RETURNED. (REMEMBER TO DECLARE DUPLX$ AS AN INTEGER FUNCTION IF THE RETURNED VALUE IS TO BE USED.) THE CURRENT DEFINITION OF THE CONFIGURATION WORD IS AS FOLLOWS:

BIT	MASK	MEANING IF BIT IS ON
1	100000	HALF DUPLEX

2	040000	DO NOT ECHO LINE FEED AFTER CARRIAGE RETURN
5-8	007400	RESERVED
9-16	000377	INTERNAL BUFFER NUMBER (READ-ONLY)

3. KERNEL CHANGES

3.1 TMAIN MODIFICATIONS

THE SVC INTERLUDES HAVE BEEN MOVED OUT OF TMAIN, WHICH IS PERMANENTLY WIRED DOWN, AND HAVE BEEN PLACED IN PAGEABLE AREA. THIS HAS FREED UP TWO PAGES OF LOCKED MEMORY.
PRI300 NOW HAS A NEW SUB-UFD CALLED IO.16/IO.32 WHICH CONTAINS THE SVC INTERLUDE BINARIES.

3.2 PHANT$ -- START PHANTOM SVC

A NEW SVC (NUMBER '605) IS AVAILABLE TO START A PHANTOM USER.

FUNCTION: TO START A PHANTOM USER.

CALLING SEQUENCE: CALL PHANT$(FILNAM,NAMLEN,UNIT,USER,CODE)
INTEGER FILNAM(1),NAMLEN,UNIT,USER,CODE

FILNAM	THE NAME OF THE COMMAND INPUT FILE TO BE RUN BY THE PHANTOM.
NAMLEN	THE LENGTH IN CHARACTERS OF 'FILNAM'.
UNIT	THE FILE UNIT ON WHICH TO OPEN 'FILNAM'. IF UNIT IS 0, UNIT 6 WILL BE USED.
USER	A VARIABLE RETURNED AS THE USER NUMBER OF THE PHANTOM.
CODE	THE RETURN CODE. IF 0, THE PHANTOM WAS INITIATED SUCCESSFULLY. IF CODE = E$NPHA, NO PHANTOMS WERE AVAILABLE. OTHER VALUES OF 'CODE' ARE FILE SYSTEM ERROR INDICATIONS.

3.3 PHANTOM TTY REQUEST MODIFICATION

THE 'PHANTOM TTY REQUEST' MESSAGE, SENT TO THE SYSTEM CONSOLE WHEN A PHANTOM USER REQUESTS TERMINAL INPUT, HAS BEEN MODIFIED TO:

USER NN: PHANTOM TTY REQUEST

WHERE 'NN' IS THE (DECIMAL) NUMBER OF THE PHANTOM USER. FILE SYSTEM MODIFICATIONS

3.4 PRIVILEGE TO OPEN CURRENT UFD

WHEN THE CURRENT UFD IS BEING OPENED (AS VIA A CALL TO SRCH$$ WITH A NAME OF K$CURR) ON A NEW PARTITION, THE ACCESS PRIVILEGES HAVE BEEN CHANGED FROM <7 0> TO <7 1>. THE EFFECT OF THIS IS THAT NOW THE CURRENT UFD CAN BE OPENED FOR READING ON A NEW PARTITION WHEN ATTACHED AS NONOWNER. ON AN OLD PARTITION THE ACCESS PRIVILEGE REMAINS <7 0> (TO PREVENT DIVULGING THE UFD PASSWORDS TO A NONOWNER).

3.5 PHANTOM COMMAND MODIFICATION

ON SUCCESSFUL INITIATION OF A PHANTOM USER, THE PHANTOM COMMAND WILL NOW RESPOND WITH A MESSAGE TO THE TERMINAL OF THE INITIATING USER:

```
PHANTOM IS USER NN
```

WHERE 'NN' IS THE (DECIMAL) NUMBER OF THE PHANTOM JUST INITIATED.

3.6 LOGOUT ALL OPTION

USER 1 CAN NOW OPTIONALLY SPECIFY 'ALL' ON A LOGOUT COMMAND:

```
LO ALL
```

THIS COMMAND WILL FORCE-LOGOUT ALL USERS EXCEPT USER 1. IT WILL ALSO PERFORM AN INTERNAL 'MAXUSR 1', THUS NOT ALLOWING SUBSEQUENT LOGINS (UNTIL MAXUSR HAS BEEN RAISED AGAIN). THIS COMMAND CAN BE USED JUST PRIOR TO A 'SH ALL' TO ALLOW A MORE ORDERLY SHUTDOWN OF PRIMOS (FOR EXAMPLE, ALL USERS WILL EXECUTE THE EXTERNAL LOGOUT PROGRAM).

3.7 SHUTDOWN-ALL PROMPT

A 'SH ALL' COMMAND FROM USER 1 WILL NOW RESULT IN THE PROMPT:

```
REALLY?
```

THE RESPONSE MUST BE 'YES' FOR THE SHUTDOWN TO TAKE PLACE.

3.9 MAXUSR -- NEW COMMAND

THIS IS A PRIVILEGED COMMAND THAT CAN BE USED TO SET THE MAXIMUM NUMBER OF USERS THAT CAN BE LOGGED IN. THE MAXUSR COUNT INCLUDES THE NUMBER OF PHANTOMS. THE DEFAULT VALUE OF MAXUSR IS THE MAXIMUM NUMBER OF USERS AS SPECIFIED IN THE CONFIG COMMAND. IF A USER ATTEMPTS TO LOGIN WHEN THE MAXIMUM NUMBER OF USERS HAVE BEEN REACHED, THE SYSTEM RESPONDS WITH THE ERROR MESSAGE: MAX. NO. OF USERS EXCEEDED. THE COMMAND IS USED BY TYPING,FROM THE SYSTEM CONSOLE,
MAXUSR N

WHERE N IS THE NO. OF USERS DESIRED. THIS COMMAND IS ESPECIALLY USEFUL WHEN IT IS DESIRED TO KEEP THE SYSTEM RUNNING (FOR BACKUPS, FOR INSTANCE) BUT NOT TO PERMIT ANY OTHER USERS FROM USING IT.

4. NETWORK CHANGES

NETWORK MODIFICATIONS

IN PRIMOS III PRIMENET SUPPORTS COMMUNICATIONS AMONG PRIME PROCESSORS OVER ONE COMMUNICATIONS MEDIUM: THE IPC.
WHEN SO CONFIGURED, PRIMOS PROVIDES NETWORK SERVICES OVER THIS MEDIUM TO A COLLECTION OF REMOTE NODES. AS A RESULT OF THIS INCREASED SUPPORT, THE USER-AVAILABLE NETWORKING AND INTERPROCESS COMMUNICATION PRIMITIVES HAVE BEEN SLIGHTLY MODIFIED. (THESE PRIMITIVES WERE ORIGINALLY DESCRIBED IN PE-T-284.) THE PRIMITIVES AFFECTED ARE CONECT, RJCON, GETCON, AND NTSTAT. THE CHANGES TO EACH ARE BRIEFLY DESCRIBED BELOW.

CONECT: THE CONECT PRIMITIVE OPTIONAL PARAMETER 'NUMTYP' IS NOW FURTHER DEFINED. AS BEFORE, WHEN NUMTYP IS NOT SPECIFIED, PRIMOS WILL ATTEMPT TO ESTABLISH A CONNECTION USING AN IPC LINK. WHEN NUMTYP IS SPECIFIED, IT REFERS TO THE TYPE OF COMMUNICATIONS PATH TO USE FOR THE CONNECTION, AND (IF AVAILABLE) WHICH OF THE MULTIPLE PATHS OF THAT TYPE TO USE. THE LOW BYTE OF NUMTYP IS THE NETWORK TYPE, AND THE HIGH BYTE IS RESERVED FOR THE LINE NUMBER. CURRENT LEGAL VALUES FOR NUMTYP ARE: 0 (IPC).

RJCON: THE PRIMITIVE SUPPLIED TO REJECT A CONNECTION NOW ALSO TAKES A 'NUMTYP' AS AN ADDITIONAL ARGUMENT. NUMTYP IS USED EXACTLY AS FOR CONECT. THE REST OF THE ARGUMENTS ARE UNAFFECTED. THE NEW CALLING SEQUENCE FOR RJCON IS:

```
CALL RJCON (NODNAM, USER, STATUS, NUMTYP)
```

GETCON: THE STATUS PARAMETER TO A GETCON CALL IS NOW REQUIRED TO BE A TWO WORD ARRAY. THE FIRST WORD IS THE STATUS OF THE CALL (AS BEFORE). THE SECOND WORD IS NOW THE ANALOG TO THE NUMTYP PARAMETER IN THE CONECT CALL. THIS SECOND STATUS FIELD CONTAINS THE NUMBER THAT CORRESPONDS TO THE NETWORK TYPE OVER WHICH THE PENDING REQUEST CAME.

NTSTAT: THE NTSTAT PRIMITIVE ALLOWS USERS TO OBTAIN CURRENT NETWORK STATUS INFORMATION. THE CHANGES MADE TO NTSTAT SIMPLY EXTEND THE STATUS REPORTING TO INCLUDE ALL OF THE PRIMENET COMMUNICATIONS MEDIA. WHEN REQUESTING STATUS INFORMATION FOR A PARTICULAR NODE (NTSTAT KEYS 2 AND 3), THE USER MUST SPECIFY IN THE 'P1' PARAMETER THE LINE NUMBER/NETWORK TYPE OF THE APPROPRIATE COMMUNICATIONS PATH. (AGAIN, THE FORM THIS ARGUMENT TAKES IS THE SAME AS THE 'NUMTYP' IN CONECT.) FURTHER, CALLING NTSTAT WITH A KEY OF 4 NOW FILLS IN THE FOURTH AND SIXTH WORDS OF THE DATA ARRAY WITH THE NUMBER OF IPC, AND RING NETWORK LINES TO THE SPECIFIED NODE. THE FIFTH ITEM IN THE ARRAY IS RESERVED, AND WILL ALWAYS BE ZERO.

5 CORRECTED REV 14.1, 14.2 PROBLEMS

5.1 SETIME CORRECTIONS

THE SETIME COMMAND AT REV 14.1 AND 14.2 ACCEPTED A YEAR LATER THAN 1977 IN THE OLD FORM, I.E., WITH A SINGLE DIGIT OF YEAR SPECIFIED. IT NOW REQUIRES TWO DIGITS IN ALL CASES.

1978 WAS DETERMINED INADVERTENTLY TO BE A LEAP YEAR.

5.2 SRCH$$ -- TYPE RETURNED INCORRECTLY ACROSS NETWORK

CALLS TO SRCH$$ THAT CAUSED A REMOTE REFERENCE RETURNED THE TYPE PARAMETER WITH AN INCORRECT VALUE.

5.3 RDEN$$ -- SPURIOUS E$IREM ERRORS

RDEN$$ WOULD ON OCCASION INCORRECTLY RETURN AN E$IREM (ILLEGAL REMOTE REFERENCE) ERROR.

5.4 PTRAP --SYSTEM CONSOLE HUNG DUE TO CENTRONICS

WHEN SENSE SWITCH 1 OF THE CONTROL PANEL WAS RAISED WHILE THE CENTRONICS SERIAL PRINTER WAS PRINTING, THE SYSTEM CONSOLE REMAINED HUNG.

5.5 DOSSUB --EXTERNAL COMMANDS

WHEN EXTERNAL COMMANDS WERE GIVEN FROM THE SYSTEM CONSOLE, THEY OVERLAYED THE OPERATING SYSTEM, AND CAUSED THE SYSTEM TO CRASH. THE SYSTEM NOW RESPONDS WITH A NO OPR1 (NO OPERATOR PRIVILEGE)

5.6 PBDIOS --SECOND SERIAL PRINTER

THE SECOND CENTRONICS SERIAL PRINTER DID NOT WORK.

5.7 COMO$$ --CORRECTION TO PRWF$$

WHEN THE COMMAND COMO -CONTIN WAS GIVEN, THE SYSTEM CRASHED. PRWF$$ DID NOT WORK CORRETLY IF AN ATTEMPT WAS MADE TO POSITION IT AT END OF A FILE WHEN IT WAS ALREADY AT END OF FILE.

ABSTRACT

REVISION 15 OF PRIMOS IV EXTENDS THE PERFORMANCE IMPROVEMENTS MADE IN REVISION 14. NEW FUNCTIONALITY INCLUDES NETWORK SUPPORT FOR THE HSSMLC, MORE POWERFUL CONFIG PARAMETER HANDLING, AND SUPPORT FOR TWO URCS. PERFORMANCE HAS BEEN IMPROVED BY LOWERING THE WORKING SET OF PRIMOS, ALLOWING GREATER FILE SYSTEM CONCURRENCY, AND UNLOCKING THE RING 0 STACKS OF NON-LOGGED IN USERS. THIS DOCUMENT DESCRIBES ALL NEW REV 15 FEATURES AND IMPROVEMENTS.

REVISION 1 OF THIS DOCUMENT DESCIBES ALL NEW REV 15.1 FEATURES, IMPROVEMENTS, AND ERROR CORRECTIONS.

TABLE OF CONTENTS

1 CONFIGURATION AND OPERATIONAL MODIFICATIONS

1.1 BUILDING PRIMOS IV

THE BUILD PROCEDURES FOR PRIMOS IV, REV 15 ARE UNCHANGED FROM REV 14.

1.2 VERSIONS OF PRIMOS IV

PRIMOS IV IS DISTRIBUTED IN THREE VERSIONS -- 64-USER, 16-USER, AND A LARGE ADDRESS SPACE 16-USER VERSION. THE DEFAULT CHARACTERISTICS OF EACH VERSION ARE AS FOLLOWS:

VERSION	NUMBER SEGMENTS	SEGMENTS PER USER
64-USER	192	32
16-USER	144	8
16-USER-LARGE	320	256

AT REV 15, THE INCREASED FUNCTIONALITY OF PRIMENET HAS MADE IT NECESSARY TO BUILD THREE ADDITIONAL VERSIONS OF PRIMOS IV TO SUPPORT NETWORKING. AVAILABLE TO CUSTOMERS PURCHASING NETWORK FACILITIES, THESE THREE VERSIONS ARE THE THREE REGULARLY AVAILABLE VERSIONS WITH NETWORK SUPPORT INCLUDED. ALL REFERENCES TO NETWORKING IN THIS DOCUMENT REFER TO THESE THREE ADDITIONAL VERSIONS. ATTEMPTS TO CONFIGURE NETWORKS IN NON-NETWORKING VERSIONS WILL RESULT IN THE MESSAGE:

PRIMENET NOT AVAILABLE IN THIS VERSION

AND A MACHINE HALT AT COLD START. IN ALL OTHER RESPECTS, THE NETWORKING VERSIONS ARE THE SAME AS THEIR NON-NETWORKING COUNTERPARTS.

1.3 RUNNING PRIMOS IV

THE UFDS CONTAINING THE RUN FILES AND COMMAND FILES FOR THE THREE VERSIONS ARE IN PR4.64, PR4.16, AND PR4L16. AT INSTALLATIONS SUPPORTING PRIMENET, THE THREE NETWORKING PRIMOS VERSIONS ARE IN PRINET>NR4.64, PRINET>NR4.16, AND PRINET>NR4L16. TO RUN PRIMOS, ATTACH TO THE APPROPRIATE UFD AND TYPE 'R PRIMOS'.

NOTE: BREAK$ NOW MAINTAINS A QUIT INHIBIT COUNTER, RATHER THAN A SINGLE FLAG. EXTERNAL LOGIN PROGRAMS (WHICH ARE ENTERED WITH QUITS INHIBITED) SHOULD ENABLE BREAKS BEFORE CALLING EXIT. IN OTHER WORDS, THEY SHOULD ALWAYS ENABLE QUITS ONE TIME MORE THAN THE NUMBER OF INHIBITS, OTHERWISE THE USER WILL BE LOGGED IN WITH QUITS INHIBITED. (IF THIS HAPPENS, CAUSING ANY ERROR WILL RE-ENABLE QUITS.)

1.4 NUMBER OF SEGMENTS, PAGING SPACE

THE NUMBER OF SEGMENTS REQUIRED BY PRIMOS IS GIVEN BY:

$$NSEG = N + 9 + USERSEGS$$

WHERE N IS THE TOTAL NUMBER OF CONFIGURED USERS -- CONFIG PARAMETER 0 (NUMBER TERMINAL USERS) + CONFIG PARAMETER 6 (NUMBER PHANTOMS) + CONFIG PARAMETER 7 (NUMBER REMOTE USERS) -- AND USERSEGS IS THE TOTAL NUMBER OF SEGMENTS TO BE AVAILABLE TO USERS. NSEG MUST BE LESS THAN OR EQUAL TO 'NUMBER SEGMENTS' GIVEN IN THE ABOVE TABLE. IF IT IS DESIRED TO LIMIT NSEG TO A NUMBER LESS THAN 192, 144, OR 320 (TO PRESERVE PAGING SPACE, FOR EXAMPLE), THE NSEG, PAGDEV, AND ALTDEV CONFIGURATION COMMANDS CAN BE USED (SEE SECTION 6). IF NSEG IS NOT MODIFIED, USERSEGS DEFAULTS AS FOLLOWS:

VERSION	USERSEGS	
64-USER	119	(192 - 64 - 9)
16-USER	119	(144 - 16 - 9)
16-USER-LARGE	295	(320 - 16 - 9)

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

$$RECORDS = (64*USERSEGS + 8*N + 256) * RECORDS/PAGE$$

WHERE N IS AGAIN THE TOTAL NUMBER OF CONFIGURED USERS.

NOTE: IF IT IS DESIRED TO START WITH A SPECIFIED AMOUNT OF PRIMARY AND ALTERNATE PAGING SPACE, THE CALCULATION OF NSEG CAN BE PERFORMED AUTOMATICALLY BY USING THE <RECORDS> PARAMETER ON THE PAGDEV AND ALTDEV CONFIGURATION COMMANDS -- SEE SECTION 6.

1.5 NEW CONFIGURATION PROCEDURES

THE HANDLING OF THE CONFIG COMMAND HAS BEEN CONSIDERABLY EXPANDED AND MODIFIED. A COMPLETE DESCRIPTION OF THE NEW CONFIG HANDLING IS GIVEN IN SECTION 6. SOME OF THE NEW CAPABILITIES ARE AS FOLLOWS:

1.5.1 SYSTEM CONSOLE SPEED SELECTION

THE BAUD RATE OF THE SYSTEM CONSOLE CAN NOW BE SET TO 110, 300, 1200, OR 9600 BAUD. THE SELECTION CAN BE PERFORMED AS DESCRIBED UNDER THE ASRATE COMMAND IN SECTION 6 OR BY SETTING THE B-REGISTER OF *COLDS TO 110, 1010, 2010, OR 3410.

1.5.2 CONFIGURABLE TERMINAL I/O BUFFERS

IT IS NOW POSSIBLE TO CHANGE THE DEFAULT SIZES OF THE TERMINAL I/O BUFFERS BY USING THE AMLBUF AND ASRBUF COMMANDS. THE FORMAT OF THESE COMMANDS ARE DESCRIBED IN SECTION 6.

1.5.3 LOGIN INHIBIT OPTION

IT IS NOW POSSIBLE TO SUPPRESS EXECUTION OF THE LOGIN COMMAND FOR USERS THAT ARE CURRENTLY LOGGED IN. SEE THE DESCRIPTION OF THE LOGLOG COMMAND IN SECTION 6.

1.5.4 LOGIN/LOGOUT MESSAGE INHIBIT OPTION

IT IS NOW POSSIBLE TO SUPPRESS THE PRINTING OF LOGIN/LOGOUT MESSAGES ON THE SYSTEM CONSOLE. SEE THE DESCRIPTION OF THE LOGMSG COMMAND IN SECTION 6.

2 SUPPORT FOR NEW DEVICES

2.1 HSSMLC NETWORK SUPPORT

PRIMENET NOW SUPPORTS A FULL DUPLEX HSSMLC LINK. PRIMENET WILL SUPPORT A DIRECT CONNECTION OF UP TO 9600 BAUD BETWEEN HSSMLC'S AND A CONNECTION OVER A DEDICATED PHONE LINE USING FULL DUPLEX MODEMS UP TO 4800 BAUD.

2.2 SUPPORT FOR TWO UNIT RECORD CONTROLLERS

PRIMOS IV NOW SUPPORTS TWO UNIT RECORD CONTROLLERS AT DEVICE ADDRESSES 3 (FIRST CONTROLLER) AND 5 (SECOND CONTROLLER). NEW ASSIGNABLE DEVICE NAMES ARE CR1, PR2, AND PR3.

2.3 AMLC RECOGNITION OF X-OFF/X-ON CHARACTERS

SERIAL DEVICES THAT USE THE X-OFF (CONTROL-S, OCTAL 223) AND X-ON (CONTROL-Q, OCTAL 221) CHARACTERS CAN NOW BE SUPPORTED BY THE AMLC DRIVER. WHEN RECOGNITION OF THE X-OFF AND X-ON CHARACTERS IS ENABLED, SENDING AN X-OFF CHARACTER TO PRIMOS WILL INHIBIT TERMINAL OUTPUT UNTIL AN X-ON CHARACTER IS RECEIVED. DURING THE INHIBITED INTERVAL, CHARACTER DATA IN THE TERMINAL OUTPUT BUFFER IS NOT LOST.

TO TURN ON RECOGNITION OF THE X-OFF AND X-ON CHARACTERS FOR AN AMLC LINE, THE AMLC COMMAND CAN BE ISSUED FOR THE LINE:

```
AMLC <LINE> <PROTOCOL> <CONFIG-WORD> <LWORD>
```

RECOGNITION OF THE X-OFF AND X-ON CHARACTERS IS ENABLED BY SETTING BIT 3 OF <LWORD> TO 1. FOR EXAMPLE, TO ENABLE THIS OPTION FOR LINE 5 TO RUN AT 9600 BAUD, ISSUE THE COMMAND:

```
AMLC 5 TTYHS 2413 20007
```

REMEMBER THAT BITS 9-16 OF <LWORD> ARE THE BUFFER NUMBER ASSOCIATED WITH <LINE>, NORMALLY <LINE>+2.

X-OFF/X-ON RECOGNITION CAN ALSO BE TURNED ON PROGRAMMATICALLY BY CALLING THE DUPLX$ SUBROUTINE WITH A KEY WITH BIT 3 SET (:20000) (SEE ALSO NEW DUPLX$ FUNCTIONALLITY DESCRIBED BELOW). IN ADDITION, THE TERM COMMAND WILL ALLOW AN XOFF SPECIFICATION.

A LOGOUT OR QUIT FROM A USER TERMINAL WITH OUTPUT SUPPRESSED WILL TURN OUTPUT BACK ON.

NOTE: THE DETAILS OF THE IMPLEMENTATION OF THIS NEW FEATURE ARE SUBJECT TO CHANGE IN FUTURE REVISIONS OF PRIMOS IV.

3 NEW AND MODIFIED PRIMOS IV FACILITIES

3.1 PHANT$ -- START PHANTOM SVC

A NEW SVC (NUMBER '605) IS AVAILABLE TO START A PHANTOM USER. THE ROUTINE -- PHANT$ -- IS ALSO AVAILABLE AS A DIRECT-ENTRANCE CALL.

FUNCTION: TO START A PHANTOM USER.

CALLING SEQUENCE: CALL PHANT$(FILNAM,NAMLEN,UNIT,USER,CODE)
INTEGER FILNAM(1),NAMLEN,UNIT,USER,CODE

FILNAM	THE NAME OF THE COMMAND INPUT FILE TO BE RUN BY THE PHANTOM.
NAMLEN	THE LENGTH IN CHARACTERS OF 'FILNAM'.
UNIT	THE FILE UNIT ON WHICH TO OPEN 'FILNAM'. IF UNIT IS 0, UNIT 6 WILL BE USED.
USER	A VARIABLE RETURNED AS THE USER NUMBER OF THE PHANTOM.
CODE	THE RETURN CODE. IF 0, THE PHANTOM WAS INITIATED SUCCESSFULLY. IF CODE = E$NPHA, NO PHANTOMS WERE AVAILABLE. OTHER VALUES OF 'CODE' ARE FILE SYSTEM ERROR INDICATIONS.

3.2 PHANTOM TTY REQUEST MODIFICATION

THE 'PHANTOM TTY REQUEST' MESSAGE, SENT TO THE SYSTEM CONSOLE WHEN A PHANTOM USER REQUESTS TERMINAL INPUT, HAS BEEN MODIFIED TO:

USER NN: PHANTOM TTY REQUEST

WHERE 'NN' IS THE (DECIMAL) NUMBER OF THE PHANTOM USER.

3.3 FILE SYSTEM MODIFICATIONS

3.3.1 PRIVILEGE TO OPEN CURRENT UFD

WHEN THE CURRENT UFD IS BEING OPENED (AS VIA A CALL TO SRCH$$ WITH A NAME OF K$CURR) ON A NEW PARTITION, THE ACCESS PRIVILEGES HAVE BEEN CHANGED FROM <7 0> TO <7 1>. THE EFFECT OF THIS IS THAT NOW THE CURRENT UFD CAN BE OPENED FOR READING ON A NEW PARTITION WHEN ATTACHED AS NONOWNER. ON AN OLD PARTITION THE ACCESS PRIVILEGE REMAINS <7 0> (TO PREVENT DIVULGING THE UFD PASSWORDS TO A NONOWNER).

3.3.2 DIRECTORY SIZE RESTRICTION

THE SIZE OF SEGMENT DIRECTORIES AND USER FILE DIRECTORIES (UFDS) IS NOW LIMITED TO 65,536 WORDS. THIS RESTRICTION LIMITS A SEGMENT DIRECTORY TO 32,768 ENTRIES.

THE NUMBER OF ENTRIES PERMITTED IN A UFD IS DEPENDENT ON LENGTHS OF THE NAMES OF THE FILES IN THE DIRECTORY. IF ALL FILES HAVE 32 CHARACTER NAMES (MAXIMUM NAME LENGTH), A UFD WILL CONTAIN 2,339 ENTRIES.

IN ORDER TO ALLOW USERS WHO CURRENTLY HAVE DIRECTORIES OF GREATER LENGTH TO ADJUST THEIR PROGRAMS, THIS SIZE RESTRICTION IS NOT ENFORCED AT REV 15.

3.3.3 NEW SGDR$$ KEYS

TWO NEW KEYS ARE RECOGNIZED BY SGDR$$. THEY ARE SIMILAR TO K$SPOS. THE KEY K$FULL MOVES THE FILE POINTER TO THE NEXT NON-EMPTY ENTRY AND K$FREE MOVES TO THE NEXT VACANT ENTRY.

K$FULL — MOVE THE FILE POINTER OF FUNIT TO THE POSITION GIVEN BY THE VALUE OF ENTRYA. IF THE POSITION CONTAINS A FILE, SET ENTRYB TO THE VALUE OF ENTRYA. IF THE POSITION IS EMPTY, SEARCH FOR THE FIRST NON-EMPTY ENTRY FOLLOWING THE POSITION SPECIFIED. IF A NON-EMPTY ENTRY EXISTS, SET ENTRYB TO THE POSITION OF THAT ENTRY. IF THE EOF IS REACHED AND A ENTRY WITH A FILE HAS NOT BEEN FOUND, THEN RETURN -1 IN ENTRYB. IF EOF IS REACHED ON K$FULL, THE FILE POINTER IS LEFT AT EOF.

K$FREE — ACT IN THE SAME MANNER AS K$FULL, BUT FIND AN ENTRY THAT DOES NOT CONTAIN A FILE.

3.3.4 NEW RDEN$$ KEY

IT IS NOW POSSIBLE TO GET THE ENTRY INFORMATION FOR A SPECIFIED NAME USING RDEN$$.

K$NAME — POSITION TO THE START OF THE ENTRY SPECIFIED BY NAME AND NAMLEN. READ AS MUCH OF THE ENTRY AS WILL FIT INTO BUFFER. SET RNW TO THE NUMBER OF WORDS READ. IF THE ENTRY IS NOT IN THE DIRECTORY, THE CODE E$FNTF IS RETURNED.

3.3.5 SATR$$ MODIFICATIONS

THE HANDLING OF CHANGES TO THE DATE-TIME-MODIFIED FIELD AND THE DUMPED FLAG BY SATR$$ HAS BEEN CHANGED TO MATCH THE SUPERVISOR'S USE OF THESE FIELDS. WHEN THESE FIELDS ARE CHANGED FOR AN ENTRY, THE DATE-TIME-MODIFIED FIELD OF THE UFD CONTAINING THAT ENTRY WILL NOT BE CHANGED.

WHEN THE NAME OR THE PROTECTION ATTRIBUTES OF THE FILE ARE CHANGED, THE DATE-TIME-MODIFIED AND THE DUMPED BIT OF THE PARENT UFD ARE UPDATED. THE DUMPED BIT FOR THE FILE WILL ALSO BE TURNED OFF.

3.4 EVENT LOGGING MODIFICATIONS

THE NAME OF EACH DISK MENTIONED IN AN ADDISK OR STARTUP COMMAND IS NOW LOGGED IN THE LOGREC FILE (IN CMDNCO). (THE LOGPRT UTILITY HAS BEEN MODIFIED TO RECOGNIZE THE NEW ENTRY TYPE -- TYPE NAME IS DSKNAM.) THIS WILL ALLOW DETERMINATION OF THOSE DISK PACKS ON WHICH DISK ERRORS HAVE OCCURRED.

IT IS NOW POSSIBLE TO DISABLE THE EVENT RECORDING MECHANISM AND TO SUPPRESS THE 'EXCEEDING QUOTA ON LOGREC' MESSAGES. SEE THE LOGREC COMMAND IN SECTION 6.

3.5 DUPLX$ MODIFICATION

THE PRIMOS DUPLX$ SVC IS NOW A FUNCTION THAT RETURNS THE TERMINAL CONFIGURATION WORD AND INTERNAL BUFFER NUMBER AS THE VALUE OF THE FUNCTION. IN ADDITION, IF THE KEY PASSED TO DUPLX$ IS EQUAL TO -1, NO UPDATING OF THE CONFIGURATION WORD TAKES PLACE -- THE CURRENT VALUE IS JUST RETURNED. (REMEMBER TO DECLARE DUPLX$ AS AN INTEGER FUNCTION IF THE RETURNED VALUE IS TO BE USED.) THE CURRENT DEFINITION OF THE CONFIGURATION WORD IS AS FOLLOWS:

BIT	MASK	MEANING IF BIT IS ON
1	100000	HALF DUPLEX
2	040000	DO NOT ECHO LINE FEED AFTER CARRIAGE RETURN
3	020000	TURN ON X-OFF/X-ON CHARACTER RECOGNITION
4	010000	OUTPUT CURRENTLY SUPPRESSED (X-OFF RECEIVED)
5-8	007400	RESERVED
9-16	000377	INTERNAL BUFFER NUMBER (READ-ONLY)

3.6 SCHEDULER MODIFICATION

THE DEFAULT USER TIME-SLICE IS NOW 2 SECONDS. A USER WILL NOT, HOWEVER, REMAIN ON THE READY LIST FOR THIS INTERVAL. WHEN 1/3 SECOND OF CPU TIME HAS BEEN USED, A USER IS MOVED FROM THE READY LIST TO A NEW SCHEDULER QUEUE -- THE ELIGIBILITY QUEUE -- AND HIS TIME SLICE IS DECREMENTED BY 1/3

SECOND. THIS QUEUE IS CHECKED BY THE SCHEDULER AFTER CHECKING FOR INTERACTIVE USERS (ON THE HIGH PRIORITY QUEUE) AND BEFORE CHECKING THE LOW PRIORITY QUEUES. USERS CYCLE BETWEEN THE READY LIST AND THE ELIGIBILITY QUEUE UNTIL THEIR TIME SLICE IS EXHAUSTED, AT WHICH TIME HE ENTERS ONE OF THE LOW PRIORITY QUEUES. (SEE ALSO ELIGTS COMMAND BELOW.)

3.7 SECURITY ENHANCEMENT

AT REV 15.1 THE CONTENTS OF EVERY WORD OF UNITIALIZED PAGES OF USER'S VIRTUAL MEMORY WILL BE SET TO ZERO. THIS FEATURE CLOSES A SECURITY HOLE IN WHICH USERS WERE ABLE TO VIEW RANDOM DATA BELONGING TO OTHER USERS AND/OR RING-0 DATA. IT IS IMPORTANT TO NOTE THAT THE SYSTEM DOES NOT CLEAR MEMORY USED BY THE EXTERNAL LOGIN PROGRAM. THUS IF THE EXTERNAL LOGIN PROGRAM CONTAINS SENSITIVE INFORMATION IN ITS MEMORY IMAGE (SUCH AS MFD OR UFD PASSWORDS), THE PROGRAM SHOULD DESTROY THE SENSITIVE INFORMATION BEFORE CALLING EXIT.

4 INTERNAL COMMAND MODIFICATIONS AND ADDITIONS

4.1 REMOTE LOGIN

IT IS NOW POSSIBLE TO LOGIN INTO A UFD NOT LOCAL TO THE SYSTEM TO WHICH YOUR TERMINAL IS CONNECTED. THE FORMAT OF THE LOGIN COMMAND IS:

LOGIN <UFDNAME> -ON <NODENAME>

WHERE <NODENAME> IS THE NETWORK NAME OF THE SYSTEM ON WHICH THE UFD <UFDNAME> RESIDES. AT ANY TIME, ONLY A PRESET NUMBER OF USERS ARE ALLOWED TO BE LOGGED INTO A GIVEN REMOTE MACHINE. THIS NUMBER IS SET BY THE REMOTE MACHINE'S CONFIG PARAMETER 7/NUMBER-REMOTE-USERS OR THE NEW-STYLE CONFIG COMMAND NRUSR (SEE SECTION 6). THE PATHS TO BE USED IN ACCESSING THE REMOTE SYSTEM(S) CAN BE OPTIONALLY SPECIFIED WITH THE RLOGIN CONFIGURATION COMMAND. IF NO RLOGIN COMMANDS ARE ISSUED, THE -ON OPTION IS DISABLED. MORE INFORMATION ON REMOTE LOGIN IS CONTAINED IN PE-T-421.

4.2 PHANTOM COMMAND MODIFICATION

ON SUCCESSFUL INITIATION OF A PHANTOM USER, THE PHANTOM COMMAND WILL NOW RESPOND WITH A MESSAGE TO THE TERMINAL OF THE INITIATING USER:

PHANTOM IS USER NN

WHERE 'NN' IS THE (DECIMAL) NUMBER OF THE PHANTOM JUST INITIATED.

4.3 LOGOUT ALL OPTION

USER 1 CAN NOW OPTIONALLY SPECIFY 'ALL' ON A LOGOUT COMMAND:

LO ALL

THIS COMMAND WILL FORCE-LOGOUT ALL USERS EXCEPT USER 1. IT WILL ALSO PERFORM AN INTERNAL 'MAXUSR 1', THUS NOT ALLOWING SUBSEQUENT LOGINS (UNTIL MAXUSR HAS BEEN RAISED AGAIN). THIS COMMAND CAN BE USED JUST PRIOR TO A 'SH ALL' TO ALLOW A MORE ORDERLY SHUTDOWN OF PRIMOS (FOR EXAMPLE, ALL USERS WILL EXECUTE THE EXTERNAL LOGOUT PROGRAM).

4.4 SHUTDOWN-ALL PROMPT

A 'SH ALL' COMMAND FROM USER 1 WILL NOW RESULT IN THE PROMPT:

REALLY?

THE RESPONSE MUST BE 'YES' FOR THE SHUTDOWN TO TAKE PLACE.

4.5 DELAY COMMAND MODIFICATION

THE DELAY COMMAND CAN NOW ALSO BE ISSUED PRIOR TO LOGGING IN TO PRIMOS.

4.6 DELSEG COMMAND -- DELETE SEGMENT(S)

A NEW INTERNAL COMMAND -- DELSEG -- IS AVAILABLE TO A USER FOR THE PURPOSE OF FREEING (DELETING) HIS SEGMENTS. THE FORMAT OF THE COMMAND IS

```
DELSEG <SEGNO>
       ALL
```

WHERE <SEGNO> IS THE SEGMENT NUMBER OF THE SEGMENT TO BE FREED. <SEGNO> MUST BE 2000 (OCTAL) OR ABOVE AND NOT EQUAL TO 6000. SPECIFYING 'ALL' WILL DELETE ALL SEGMENTS BELONGING TO THE USER ISSUING THE COMMAND. A 'BAD PARAMETER' MESSAGE IS THE RESPONSE TO AN ILLEGAL SEGMENT NUMBER. DELETING AN ALREADY NONEXISTENT SEGMENT HAS NO EFFECT.

4.7 MAXSCH COMMAND -- SPECIFY SCHEDULING CONSTANT

THIS COMMAND IS USED TO SET THE VARIABLE MAXSCH (IN SUPCOM), WHICH CONTROLS THE CIRCUMSTANCES IN WHICH THE BACKSTOP PROCESS (THE SCHEDULER) ADDS PROCESSES TO THE READY LIST. THE FORMAT OF THE COMMAND IS:

```
MAXSCH <N>
```

THE DEFAULT VALUE OF MAXSCH IS 3.

4.8 ELIGTS COMMAND -- SET ELIGIBILITY TIME SLICE VALUE

THE ELIGTS COMMAND ALLOWS MODIFICATION OF THE ELIGIBILITY TIME SLICE. THE FORMAT OF THE COMMAND IS:

```
ELIGTS <TENTHS>
```

<TENTHS> SPECIFIES THE AMOUNT OF TIME IN TENTHS OF A SECOND THAT A USER WILL RUN BEFORE WAITING ON THE ELIGIBILITY SCHEDULER QUEUE. THE DEFAULT VALUE IS 3 (1/3 SECOND).

5 CORRECTED REV 14.1 & 14.2 PROBLEMS

5.1 SETIME CORRECTIONS

THE SETIME COMMAND AT REV 14.1 AND 14.2 ACCEPTED A YEAR LATER THAN 1977 IN THE OLD FORM, I.E., WITH A SINGLE DIGIT OF YEAR SPECIFIED. IT NOW REQUIRES TWO DIGITS IN ALL CASES.

1978 WAS DETERMINED INADVERTENTLY TO BE A LEAP YEAR.

5.2 SRCH$$ -- TYPE RETURNED INCORRECTLY ACROSS NETWORK

CALLS TO SRCH$$ THAT CAUSED A REMOTE REFERENCE RETURNED THE TYPE PARAMETER WITH AN INCORRECT VALUE.

5.3 RDEN$$ -- SPURIOUS E$IREM ERRORS

RDEN$$ WOULD ON OCCASION INCORRECTLY RETURN AN E$IREM (ILLEGAL REMOTE REFERENCE) ERROR.

5.4 ERRSET CORRECTION

A CALL TO ERRSET THAT RESULTED IN AN ALTERNATE RETURN BEING TAKEN RESULTED IN QUITS, IF INHIBITED, BEING ENABLED.

5.5 MAGTAPE CONTROLLER CORRECTIONS

WHEN MULTIPLE UNITS ON A MAGTAPE CONTROLLER WERE IN USE AND ONE USER REQUESTED TWO OPERATIONS IN SUCCESSION WITHOUT WAITING FOR COMPLETION OF THE FIRST OPERATION (MAGSAV OCCASIONALLY DOES THIS), ALL USERS OF THE CONTROLLER COULD HANG. THIS POSSIBILITY HAS BEEN REMOVED IN REV 15. ALSO, LOGGING OUT OR UNASSIGNING A TAPE UNIT WHILE AN OPERATION IS IN PROGRESS (E.G., A LONG SPACING OPERATION) WILL NOW CAUSE THE CONTROLLER TO BE RESET (AN OCP-INIT IS ISSUED), THUS FREEING IT FOR OPERATIONS ON OTHER UNITS.

5.6 SMLC CORRECTION

ATTEMPTING TO ASSIGN AN SMLC LINE WHEN THE SMLC HAD NOT BEEN CONFIGURED CAUSED AN INHIBITED PAGE-FAULT HALT.

5.7 PAPER TAPE READER

THE PAPER TAPE READER DID NOT WORK.

5.8 VERSATEC DRIVER

A STATUS REQUEST CAUSED A HALT AT 6/20000.

5.9 USER SEMAPHORE CALLS

CALLS TO SEM$TN CAUSED UNPREDICTABLE HALTS. THE USE OF SEMAPHORE NUMBERS ABOVE 16 ON 16- AND LARGE 16-USER VERSIONS ALSO CAUSED UNPREDICTABLE HALTS SINCE ONLY 16 USER SEMAPHORES WERE ALLOCATED. (64 USER SEMAPHORES ARE NOW AVAILABLE ON ALL VERSIONS.)

5.10 DSKRAT HANDLING

IT IS NOW POSSIBLE TO RENAME THE DSKRAT FILE ON NEW PARTITIONS. THE NEW NAME MUST BE NO LONGER THAN THE OLD NAME UNLESS THE PARTITION HAS BEEN MADE WITH REV 15 MAKE.

IT IS NO LONGER POSSIBLE TO OVERWRITE THE DSKRAT ON NEW PARTITIONS.

6 CORRECTED REV 15.0 PROBLEMS

6.1 SLEEP$ INACCURATE

CALLS TO SLEEP$ FOR LONG PERIODS OF TIME WERE INACCURATE IF SYSTEM USAGE WAS HEAVY.

6.2 CONFIG COMMAND NEEDED IN CONFIG FILE

THE CONFIG COMMAND WAS NOT OPTIONAL IF NETWORKS WERE CONFIGURED. THE SYSTEM WOULD FAIL TO COLD START.

6.3 DELAYED LOGIN

USERS LOGIN COMMAND WAS SOMTIMES DELAYED FOR AS LONG AS ONE MINUTE.

6.4 SECURITY PROBLEM

USER-RING (RING 3) PROGRAMS COULD USE THE RING 0 PRIVILEGED RETURN OF E$BPAS (BAD PASSWORD) FROM ATCH$$. THIS ERROR ALLOWED USERS TO WRITE A PROGRAM WHICH ITERATED THROUGH ALL POSSIBLE PASSWORDS IN FINITE AMOUNT OF TIME.

6.5 WRONG LINE NUMBER IN STATUS

THE STATUS COMMAND PRINTED THE INCORRECT LINE NUMBER WHEN THE USER'S AMLC LINE NUMBER WAS GREATER THAN THE NUMBER OF CONFIGURED TERMINAL USERS.

6.6 ATTACH-HOME FAILED

CALLS TO ATCH$$ TO ATTACH TO HOME DIRECTORY FAILED IF THE HOME DIRECTORY WAS ON A REMOTE DISK AND THE LOGICAL DEVICE ARGUMENT WAS K$ALLD (100000 OCTAL).

6.7 CARD READER-PUNCH FAILED

CALLS TO T$CMPC AND T$PMPC (CARD READER-PUNCH) COULD RESULT IN SPURIOUS 'NO MPC' ERROR MESSAGES.

6.8 GARBLED COLD START MESSAGE

AT COLD START, THE PRIMOS HEADER MESSAGE COULD SOMETIMES BE GARBLED DUE TO INCORRECT BAUD RATE SETTING.

6.9 9600 BAUD CONSOLE FAILED

SETTING THE BUAD RATE OF THE SYSTEM CONSOLE TO 9600 BAUD VIA THE B REGISTER SETTING OF *COLDS CAUSED SYSTEM CRASH AT COLDS START.

6.10 REMOTE LOGIN PROBLEM

FORCED LOGOUT OF A USER WHO WAS USING THE REMOTE LOGIN FEATURE CAUSED ANOMALOUS BEHAVIOUR.

7 NEW CONFIGURATION SPECIFICATION OPTIONS

AS AN ALTERNATIVE TO THE CURRENT CONFIG COMMAND REQUIRED BY THE PRELOADER ('PRIMOS'), CONFIG PARAMETERS ARE NOW ACCEPTED IN THE FORM OF A SERIES OF CONFIG COMMANDS. THESE COMMANDS ARE KEPT IN A DATA FILE IN CMDNCO AND ARE PROCESSED BY THE PRELOADER TO SET UP ALL THE SYSTEM PARAMETERS CURRENTLY SPECIFIED BY THE CONFIG COMMAND. IN ADDITION, THE NEW CONFIGURATION COMMANDS CAN BE USED TO SET VARIABLES IN FIGCOM AND TO OVERRIDE THE DEFAULT NETWORK CONFIGURATION.

NOTE THAT THE NEW CONFIGURATION FACILITIES IN NO WAY IMPACT EXISTING CONFIG COMMANDS. THESE WILL CONTINUE TO FUNCTION AS IN THE PAST. MANY NEW OPTIONS, HOWEVER, ARE AVAILABLE ONLY WITH THE NEW CONFIGURATION COMMANDS.

7.1 OVERVIEW OF PRELOADER ACTIONS

AS IS DONE CURRENTLY, THE PRELOADER ATTACHES TO CMDNCO AND LOOKS FOR THE FILE C_PRMO. IF THE FILE EXISTS, IT IS OPENED FOR COMMAND INPUT; IF IT DOESN'T, THE 'PLEASE ENTER CONFIG' PROMPT IS ISSUED. THE FIRST EXECUTABLE COMMAND IS READ (FROM THE TERMINAL OR FROM C_PRMO), AND A 'CO TTY' IS ISSUED. THE COMMAND IS EXAMINED TO ENSURE IT IS A CONFIG COMMAND.

N.B.: NOTE THAT COMMENTS -- LINES STARTING WITH '*' OR '/*' MAY NOW PRECEDE THE CONFIG COMMAND IN C_PRMO.

AT THIS POINT, THE NEW PRELOADER MAKES AN ADDITIONAL CHECK FOR THE KEYWORD '-DATA' AS THE FIRST NAME ON THE CONFIG COMMAND. IF THIS KEYWORD IS PRESENT, THE SECOND NAME FOLLOWING THE COMMAND IS TAKEN AS THE NAME OF A CONFIGURATION DATA FILE. THE FILE IS OPENED FOR INPUT, AND CONFIGURATION COMMANDS ARE PROCESSED AS DESCRIBED BELOW. A NEW-STYLE CONFIG COMMAND APPEARS AS:

```
CONFIG -DATA <CONFIGURATION-DATA-FILENAME>
```

NOTE: WHILE NO RESTRICTIONS ARE PLACED ON <CONFIGURATION-DATA-FILENAME> -- THE NAME OF THE CONFIGURATION DATA FILE -- IT IS SUGGESTED THAT THE NAME CONFIG BE ADAPTED AS A DEFACTO STANDARD.

7.2 CONFIGURATION COMMANDS

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 COMMANDS. THE FOLLOWING DESCRIBES ALL POSSIBLE CONFIGURATION COMMANDS IN ALPHABETICAL ORDER.

CORRESPONDENCE TO CURRENT CONFIG PARAMETERS IS NOTED WHERE APPROPRIATE. COMMAND NAMES (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 COMMAND IS NOT SPECIFIED OR IF A PARAMETER IS OMITTED, ARE UNDERLINED. THE CONFIGURATION COMMANDS CAN APPEAR IN THE CONFIGURATION DATA FILE IN ANY ORDER WITH THE EXCEPTION OF THE 'GO' COMMMAND, WHICH MUST BE THE LAST COMMAND IN THE CONFIGURATION DATA FILE.

ALL NUMERIC PARAMETERS ARE IN OCTAL UNLESS OTHERWISE SPECIFIED.

ALTDEV -- SPECIFY ALTERNATE PAGING DEVICE AND SIZE

ALTDEV <DVNO> [<RECORDS>]

<DVNO> IS THE DEVICE NUMBER OF THE DISK TO BE USED AS AN ALTERNATE PAGING DEVICE. A <DVNO> OF 0 IS NOW ACCEPTABLE. THIS COMMAND 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 COMMAND, 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 COMMAND.

AMLBUF -- SET TERMINAL I/O BUFFER SIZES

AMLBUF <LINE> [<IBUFSZ>] [<OBUFSZ>]

THE TERMINAL INPUT AND OUTPUT BUFFERS FOR AMLC LINE NUMBER <LINE> ARE SET TO THE NUMBER OF WORDS GIVEN BY <IBUFSZ> AND <OBUFSZ>. OMITTING <IBUFSZ>, <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 (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. THE DEFAULT BUFFER SIZES ARE 200 AND 300 (DECIMAL 128, 192).

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 COMMAND IS EQUIVALENT TO (AND WILL OVERRIDE) THE B-REGISTER SETTING OF *COLDS. THE DEFAULT VALUE IS 110. NOTE: IF THE ASRATE COMMAND IS USED, IT SHOULD APPEAR AT THE TOP OF THE CONFIGURATION DATA FILE IN ORDER TO ENSURE THAT ANY SUBSEQUENT CONFIGURATION ERROR MESSAGES ARE PRINTED AT THE APPROPRIATE SPEED. SOME ERROR MESSAGES (E.G., 'SEEK FAILURE ON PAGDEV') WILL BE PRINTED AT THE APPROPRIATE SPEED ONLY IF THE B-REGISTER SETTING OF *COLDS IS CORRECT.

ASRBUF -- SET ASR TERMINAL I/O BUFFER SIZE

ASRBUF <LINE> [<IBUFSZ>] [<OBUFSZ>]

THE TERMINAL INPUT AND OUTPUT BUFFERS FOR THE ASR ARE SET TO THE NUMBER OF WORDS GIVEN BY <IBUFSZ> AND <OBUFSZ>. OMITTING <IBUFSZ> OR <OBUFSZ> OR SPECIFYING 0 WILL RESULT IN NO CHANGE TO THE DEFAULT BUFFER SIZE. A 'TERMINAL I/O BUFFERS TOO LARGE' MESSAGE WILL BE PRINTED IF THE TOTAL SIZE OF THE I/O BUFFERS (INCLUDING AMLC BUFFERS) EXCEEDS 32K WORDS. A 'BAD LINE # IN ASRBUF CMND' MESSAGE WILL BE PRINTED IF <LINE> IS NOT 0. DEFAULT BUFFER SIZES ARE 200 AND 300 (DECIMAL 128 AND 192).

COMDEV -- SPECIFY COMMAND DEVICE

COMDEV <DVNO>

<DVNO> SPECIFIES THE DEVICE ON WHICH THE SYSTEM UFD CMDNCO RESIDES. THE COMMAND DEVICE MUSI BE SPECIFIED, EITHER WITH THE COMDEV COMMAND OR WITH A CONFIG COMMAND. THIS COMMAND CORRESPONDS TO CONFIG PARAMETER 2/<DVNO>.

CONFIG -- SPECIFY CONFIGURATION PARAMETERS

CONFIG [<NODE>] <NTUSR> <PAGDEV> <COMDEV> [<OTHER PARMS>]

AN OLD-STYLE CONFIG COMMAND 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 COMMAND.) A COMPLETE SPECIFICATION OF PARAMETERS FOR THE OLD-STYLE CONFIG COMMAND IS AS FOLLOWS:

<NODE>	NETWORK NODE NAME (6-CHARACTER MAXIMUM)
0/<NTUSR>	NUMBER OF TERMINAL USERS
1/<PAGDEV>	PAGING DEVICE
2/<COMDEV>	COMMAND DEVICE
3/<MAXPAG>	NUMBER PAGES PHYSICAL MEMORY TO USE
4/<ALTDEV>	ALTERNATE PAGING DEVICE
5/<NAMLC>	NUMBER ASSIGNABLE AMLC LINES
6/<NPUSR>	NUMBER PHANTOM USERS
7/<NRUSR>	NUMBER REMOTE USERS (NEW AT REV 15)
10/<SMLCON>	NON-ZERO => ENABLE SMLC

DISLOG -- SET DISCONNECT LOGOUT OPTION

DISLOG YES
NO

IF 'YES' IS SPECIFIED, A LOGOUT WILL BE PERFORMED WHEN DISCONNECT OCCURS ON AN AMLC LINE. THIS COMMAND IS USED TO SET THE FIGCOM VARIABLE DLOGOT. THE DEFAULT SETTING DOES NOT LOGOUT ON DISCONNECT.

ERASE -- SPECIFY SYSTEM DEFAULT ERASE CHARACTER

ERASE [<CHAR>] [<OCTAL-VAL>]

<CHAR> IS USED TO SET THE SYSTEM DEFAULT CHARACTER-ERASE CHARACTER. THE CHARACTER CAN OPTIONALLY BE SPECIFIED AS <OCTAL-VAL>. FOR EXAMPLE:

```
ERASE A          IS EQUIVALENT TO:
ERASE 301
```

THIS COMMAND IS USED TO SET THE FIGCOM VARIABLE DEFERA (DEFAULT VALUE IS '"').

FAM -- SPECIFY FAM NETWORK CONFIGURATION

FAM <NODENAME> <NETTYPE>

THE FAM COMMAND SPECIFIES WHICH NETWORK NODES ARE TO BE MADE AVAILABLE TO FAM. <NODENAME> SPECIFIES THE NAME OF THE REMOTE SYSTEM, <NETTYPE> SPECIFIES THE PATH TO BE USED -- IPC OR SMLC. THE DEFAULT FAM CONFIGURATION WOULD BE SPECIFIED BY THE COMMANDS:

```
FAM SYSA IPC
```

FAM SYSB IPC

FAM COMMANDS ARE IGNORED IF THE LOCAL NODE NAME IS NOT SPECIFIED WITH THE MYNAME COMMAND OR THE OLD-STYLE CONFIG <NODE> PARAMETER.

GO -- MARK END OF CONFIGURATION FILE

GO

THE GO COMMAND 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 COMMAND.

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 COMMAND IS USED TO SET THE FIGCOM VARIABLE DEFKIL. THE DEFAULT WOULD BE SPECIFIED AS:

```
KILL ?      OR EQUIVALENTLY:
KILL 277
```

LOGLOG -- ALLOW LOGINS WHILE LOGGED IN

LOGLOG YES
NO

IF 'YES' IS SPECIFIED, THE LOGIN COMMAND WILL BE PERMITTED WHILE A USER IS LOGGED IN. IF 'NO' IS SPECIFIED, THE LOGIN COMMAND WILL BE INHIBITED WHILE A USER IS LOGGED IN. THIS COMMAND IS USED TO SET THE FIGCOM VARIABLE LOGOVR. THE DEFAULT SETTING ALLOWS LOGINS WHILE LOGGED IN.

LOGMSG -- PRINT LOGIN/LOGOUT MESSAGES

LOGMSG YES
NO

THIS COMMAND 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 COMMAND IS USED TO SET THE FIGCOM VARIABLE NLGPRT.

LOGREC -- SPECIFY MAXIMUM SIZE OF LOGREC FILE

LOGREC <VAL>

<VAL>, IF POSITIVE, SPECIFIES THE NUMBER OF WORDS IN THE LOGREC FILE. WHEN LOGREC EXCEEDS <VAL> WORDS, THE 'EXCEEDING QUOTA ON LOGREC' MESSAGE IS PRINTED AS EACH NEW ENTRY IS ADDED TO LOGREC. SPECIFYING AN <VAL> OF 0 WILL INHIBIT THE QUOTA CHECK; NO MESSAGE WILL EVER BE PRINTED. SPECIFYING A NEGATIVE <VAL> WILL SUPPRESS ALL ATTEMPTS TO WRITE TO THE LOGREC FILE. (THIS WILL AVOID DISK WRITE ERRORS IF RUNNING ON A WRITE-PROTECTED DISK.) THE DEFAULT VALUE IS 10000 (4096 DECIMAL). THIS COMMAND IS USED TO SET THE VARIABLE LRQUOT IN FIGCOM.

LOUTQM -- SPECIFY INACTIVITY-LOGOUT QUANTUM

LOUTQM <MINS>

THIS COMMAND SPECIFIED 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 COMMAND IS USED TO SET THE FIGCOM VARIABLE LOUTQM. <MINS> MUST BE GREATER THAN ZERO.

MAXPAG -- SPECIFY NUMBER PAGES OF MEMORY TO VALIDATE

MAXPAG <NPAGES>

<NPAGES> IS THE NUMBER OF PAGES OF PHYSICAL MEMORY TO VALIDATE FOR USE. THE DEFAULT VALUE IS 400 (256 DECIMAL). THIS COMMAND CORRESPONDS TO THE OLD-STYLE CONFIG PARAMETER 3/<NPAGES>. (MEMORY VALIDATION OCCURS AT COLD START. EACH PAGE IS 1024 WORDS.)

MYNAME -- SPECIFY NETWORK NAME OF LOCAL NODE

MYNAME <NODENAME>

<NODENAME> (6 CHARACTERS MAX) IDENTIFIES THE NETWORK NODE NAME OF THE LOCAL SYSTEM (THE SYSTEM BEING COLD-STARTED). SPECIFICATION OF <NODENAME>, EITHER WITH THE MYNAME COMMAND OR BY SPECIFYING <NODE> ON AN OLD-STYLE CONFIG COMMAND, ENABLES PROCESSING OF THE FAM, NET, AND RLOGIN CONFIGURATION COMMANDS. IN THE ABSENCE OF ANY NETWORK NODENAME SPECIFICATION, THESE COMMANDS ARE IGNORED. (NOTE: FAILURE TO SPECIFY THE LOCAL NODENAME EFFECTIVELY DISABLES ALL NETWORK ACTIVITY. ATTEMPTING TO RUN FAM WITH THE NETWORK DISABLED WILL RESULT IN A 'BAD FAM SVC' ERROR.)

NAMLC -- SPECIFY NUMBER ASSIGNABLE AMLC LINES

NAMLC <NLINES>

<NLINES> SPECIFIES THE NUMBER OF ASSIGNABLE AMLC LINES IN THE SYSTEM. THIS COMMAND CORRESPONDS TO THE OLD-STYLE CONFIG PARAMETER 5/<NLINES>. THE DEFAULT VALUE IS 0.

NET -- SPECIFY NETWORK CONFIGURATION PARAMETERS

NET <NODENAME> IPC <NODENUMBER>
NET <NODENAME> SMLC <LINENUMBER> <CONFIG> <DSCTRL>

NET COMMANDS CAN BE USED TO OVERRIDE THE DEFAULT NETWORK CONFIGURATION BUILT INTO PRIMOS IV. EACH NET COMMAND IDENTIFIES A NODE IN THE NETWORK, THE NAME OF THAT NODE, AND THE IPC OR SMLC LINE NUMBER ASSOCIATED WITH THE NODE.

FOR AN SMLC NODE, <LINENUMBER> SPECIFIES THE LOGICAL LINE NUMBER (0 - 3) OVER WHICH THE CONNECTION TO <NODENAME> WILL BE MADE. THE MAPPING OF LOGICAL LINE NUMBER TO CONTROLLER NUMBER AND PHYSICAL LINE NUMBER IS DEFINED BY THE SMLC COMMAND (WHICH SEE). <CONFIG> SPECIFIES THE SMLC CONFIGURATION WORD FOR <LINENUMBER> AS DESCRIBED IN THE HSSMLC USER GUIDE, MAN2362 (PP. 5-11). <DSCTRL> SPECIFIES THE DATA SET CONTROL AND DATA SET STATUS EXPECTED FOR <LINENUMBER> AS FOLLOWS:

BITS 1-4: RECEIVER DATA SET STATUS EXPECTED:

100000 - RING INDICATOR
040000 - DATA CARRIER DETECT
020000 - CLEAR TO SEND
010000 - DATA SET READY

BITS 5-8: TRANSMITTER DATA SET STATUS EXPECTED:

004000 - RING INDICATOR
002000 - DATA CARRIER DETECT
001000 - CLEAR TO SEND
000400 - DATA SET READY

BITS 9-16: DATA SET CONTROL:

000370 - UNUSED
000004 - SPEED SELECT/SEND NEW SYNC
000002 - REQUEST TO SEND
000001 - DATA TERMINAL READY

THE RECEIVER DATA SET STATUS EXPECTED MUST CONTAIN AT LEAST THE SAME BITS AS THE TRANSMITTER DATA SET STATUS EXPECTED. THE DEFAULT <CONFIG> AND <DSCTRL> ARE 363 AND 50401. IF <NODENAME> IS THE SAME AS THE <NODE> SPECIFIED IN THE

MYNAME COMMAND, <LINENUMBER>, <CONFIG>, AND <DSCTRL> ARE IGNORED.

THE DEFAULT NETWORK CONFIGURATION IS A TWO-NODE IPC NETWORK THAT WOULD BE SPECIFIED AS:

```
NET SYSA IPC 1
NET SYSB IPC 2
```

NOTE: IF ANY NET COMMAND IS INCLUDED IN THE CONFIGURATION COMMAND FILE, THE ENTIRE NETWORK CONFIGURATION MUST BE SPECIFIED. NET COMMANDS ARE IGNORED IF THE LOCAL NODE NAME IS NOT SPECIFIED WITH THE MYNAME COMMAND OR THE OLD-STYLE CONFIG <NODE> PARAMETER. IN ADDITION, THE APPROPRIATE FAM AND RLOGIN COMMANDS MUST BE SPECIFIED.

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 COMMAND CORRESPONDS TO THE OLD-STYLE CONFIG PARAMETER 6/<N>. THE DEFAULT IS 0.

NRUSR -- SPECIFY NUMBER REMOTE USERS

NRUSR <N>

<N> SPECIFIES THE NUMBER OF PROCESSES TO BE RESERVED FOR REMOTE LOGINS (THE DEFAULT NUMBER IS 0). THE NRUSR COMMAND ALLOWS UP TO <N> CONCURRENT REMOTE USERS TO CONNECT TO THIS SYSTEM USING THE -ON KEYWORD OF THE LOGIN COMMAND (MAXIMUM VALUE IS 40 -- DECIMAL 32). THE NUMBER OF REMOTE USERS IS ADDED TO NPUSR AND NTUSR TO DETERMINE THE TOTAL NUMBER OF USERS ON THE SYSTEM.

NSEG -- SPECIFY NUMBER AVAILABLE SEGMENTS IN SYSTEM

NSEG <N>

<N> SPECIFIES THE TOTAL NUMBER OF SEGMENTS AVAILABLE IN THE SYSTEM (PRIMOS IV PLUS ALL USERS). THIS COMMAND IS USED TO SET THE VARIABLE NSEG IN SEGMENT 4. DEFAULT VALUES ARE 220 (16-USER), 300 (64-USER), AND 500 (LARGE 16-USER) -- DECIMAL 144, 192, 320. <N> MUST BE GREATER THAN TOTAL USERS + 9 AND LESS THAN THE DEFAULT VALUE. IF THE AMOUNT OF PAGING SPACE SPECIFIED IN THE PAGDEV AND ALTDEV COMMANDS 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 COMMANDS.)

NTUSR -- SPECIFY NUMBER OF TERMINAL USERS

NTUSR <N>

<N> SPECIFIES THE NUMBER OF TERMINAL USERS TO BE CONFIGURED. THE NUMBER OF USERS MUSI BE SPECIFIED, EITHER WITH THE NTUSR COMMAND OR WITH THE CONFIG COMMAND. THIS COMMAND CORRESPONDS TO THE OLD-STYLE CONFIG PARAMETER 0/<N>. NTUSR MUST BE GREATER THAN 0 AND LESS THAN OR EQUAL TO NUMBER OF USERS SUPPORTED BY THE VERSION (16- OR 64-USER) OF PRIMOS BEING RUN. NTUSR IS ADDED TO NPUSR AND NTUSR TO DETERMINE THE TOTAL NUMBER OF USERS ON THE SYSTEM.

PAGDEV -- SPECIFY PAGING DEVICE AND SIZE

PAGDEV <DVNO> [<RECORDS>]

<DVNO> SPECIFIES THE PHYSICAL DISK ON WHICH PAGING IS TO TAKE PLACE. THE PAGING DEVICE MUSI BE SPECIFIED, EITHER WITH THE PAGDEV COMMAND OR WITH THE CONFIG COMMAND. THIS COMMAND 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 COMMAND, IS USED TO LIMIT NSEG -- THE TOTAL NUMBER OF SEGMENTS IN THE SYSTEM. SECOND, IF AN ALTERNATE PAGING DEVICE HAS BEEN SPECIFIED (ALTDEV), <RECORDS> WILL DEFINE THE POINT AT WHICH PAGE SPACE ALLOCATION SWITCHES FROM THE PRIMARY TO THE ALTERNATE PAGING DEVICE.

NOTE: <RECORDS> CAN BE AS SMALL AS 1 TO FORCE ALMOST ALL PAGING TO OCCUR ON THE ALTERNATE PAGING DEVICE. THE PRIMARY DEVICE, HOWEVER, WILL ALWAYS BE USED TO PAGE THE SEGMENTS USED BY PRIMOS IV (SEGMENT NUMBERS 0-12 AND USER 1'S SEGMENT 6000).

PREPAG -- SPECIFY NUMBER OF PAGES TO PREPAGE

PREPAG <N>

<N> SPECIFIES THE NUMBER OF PAGES TO PREPAGE OUT WHEN A PAGE FAULT OCCURS. THE DEFAULT VALUE IS 3. THIS COMMAND SETS THE VARIABLE PREPGK IN PAGCOM.

RLOGIN -- SPECIFY REMOTE LOGIN NETWORK CONFIGURATION

RLOGIN <NODENAME> <NETTYPE>

THE RLOGIN COMMAND SPECIFIES WHICH NETWORK NODES ARE TO BE USED FOR REMOTE LOGINS FROM LOCAL TERMINALS USING THE -ON OPTION OF THE LOGIN COMMAND. <NODENAME> SPECIFIES THE NAME OF THE REMOTE SYSTEM, <NETTYPE> SPECIFIES THE PATH TO BE USED -- IPC OR SMLC. RLOGIN COMMANDS ARE IGNORED IF THE LOCAL NODE NAME IS NOT SPECIFIED WITH THE MYNAME COMMAND OR THE OLD-STYLE CONFIG <NODE> PARAMETER. THE DEFAULT PATHS USED BY REMOTE LOGIN WOULD BE SPECIFIED AS:

```
RLOGIN SYSA IPC
RLOBIN SYSB IPC
```

IF NO RLOGIN COMMANDS ARE ISSUED, REMOTE LOGINS ARE NOT POSSIBLE FROM THE LOCAL MACHINE.

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.

SMLC -- ENABLE AND CONFIGURE SMLC LINES

```
SMLC ON
SMLC CNTRLR <CTRLR-NUMBER> <DEVADR>
SMLC SMLCNN <CTRLR-NUMBER> <LINE-NUMBER>
```

SMLC COMMANDS 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-SMLC07) ONTO PHYSICAL CONTROLLERS AND LINE NUMBERS. <CTRLR-NUMBER> IS AS FOR THE THE SMLC CNTRLR COMMAND; <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.

AT REV 15.1 THERE EXISTS A KNOWN PROBLEM WITH THIS COMMAND. UNDER SOME CIRCOMSTANCES, PROCUDURE WHICH MUST BE MEMORY RESIDENT IS NOT MADE NON-PAGABLE. A SYSTEM CRASH WHEN USING THE SMLC IS POSSIBLE. THEREFORE IT IS RECOMENDED THAT THE OLD STYLE CONFIG COMMAND (10/1) BE USED. THIS PROBLEM WILL BE CORRECTED AT REV 15.2.

TYPOUT -- CONTROL PRINTING OF CONFIGURATION COMMANDS

TYPOUT YES
NO

PRINTING OF THE CONFIGURATION COMMANDS ON THE SYSTEM CONSOLE IS UNDER THE CONTROL OF THE TYPOUT COMMAND. SPECIFYING 'YES' WILL CAUSE THE COMMANDS TO BE PRINTED AS THEY ARE PROCESSED. THE DEFAULT OR ANY OTHER SPECIFICATION WILL CAUSE PRINTING OF THE COMMANDS TO BE SUPPRESSED. (SEVERAL TYPOUT COMMANDS CAN BE USED TO PRINT SELECTED CONFIGURATION COMMANDS.)

7.3 PRIMOS IV INITIALIZATION ERROR MESSAGES

THE FOLLOWING LISTS ALL ERROR MESSAGES GENERATED BY THE PRIMOS IV PRELOADER ('PRIMOS') AND THE PRIMOS IV INITIALIZATION SEQUENCE.

PRELOADER ('PRIMOS') ERROR MESSAGES

<FILE-SYSTEM-ERROR-MESSAGE> CMDNCO (PRIMOS)

A FILE SYSTEM ERROR WAS ENCOUNTERED BY THE PRELOADER WHILE ATTEMPTING TO ATTACH TO CMDNCO.

<FILE-SYSTEM-ERROR-MESSAGE> C_PRMO (PRIMOS)

A FILE SYSTEM ERROR (OTHER THAN FILE NOT FOUND) WAS ENCOUNTERED BY THE PRELOADER WHILE ATTEMPTING TO OPEN THE FILE C_PRMO FOR COMMAND INPUT.

FIRST COMMAND MUST BE CONFIG

THE COMMAND TYPED IN RESPONSE TO THE 'PLEASE ENTER CONFIG' PROMPT OR THE FIRST EXECUTABLE COMMAND IN C_PRMO IS NOT A CONFIG COMMAND.

<FILE-SYSTEM-ERROR-MESSAGE> <CONFIG-FILE> (PRIMOS)

A FILE SYSTEM ERROR WAS ENCOUNTERED BY THE PRELOADER WHILE ATTEMPTING TO OPEN THE CONFIGURATION FILE <CONFIG-FILE>.

MISSING NTUSR, PAGDEV, OR COMDEV

THE CONFIGURATION DATA FILE DID NOT SPECIFY THESE REQUIRED PARAMETERS.

ILLEGAL PAGDEV

THE DEVICE SPECIFIED FOR PAGING IS NOT A LEGAL PAGING DEVICE.

USE <DVNO> FOR PAGING?

THE DISK <DVNO> HAS BEEN SPECIFIED AS THE PAGING DEVICE, BUT IS FORMATTED AS A STANDARD PRIMOS DISK. A REPLY OF 'YES' IS REQUIRED TO ENABLE PAGING ACTIVITY ON <DVNO>.

ILLEGAL COMDEV

THE DEVICE SPECIFIED FOR THE COMMAND DEVICE IS NOT LEGAL.

ILLEGAL ALTDEV

THE DEVICE SPECIFIED AS THE ALTERNATE PAGING DEVICE IS NOT LEGAL.

<FILE-SYSTEM-ERROR-MESSAGE> PRNNNN (PRIMOS)

A FILE SYSTEM ERROR WAS ENCOUNTERED BY THE PRELOADER WHILE ATTEMPTING TO OPEN OR READ THE INDICATED PRNNNN FILE.

END OF FILE. MISSING 'GO' CMND (PRIMOS)

THE CONFIGURATION DATA FILE DOES NOT INCLUDE A GO COMMAND AS REQUIRED.

TPIOS ERROR

AN I/O ERROR OCCURRED WHILE PRELOADING THE PAGING DEVICE.

PRIMOS IV INITIALIZATION ERROR MESSAGES

NTUSR+NPUSR+NRUSR TOO BIG (AINIT)

THE NUMBER OF TERMINAL PLUS PHANTOM PLUS REMOTE USERS EXCEEDS THE MAXIMUM NUMBER OF CONFIGURABLE USERS (16 OR 64).

NRUSR INVALID (AINIT)

THE NUMBER OF REMOTE USERS SPECIFIED BY AN NRUSR COMMAND EXCEEDS THE MAXIMUM NUMBER OF CONFIGURABLE REMOTE USERS (40, DECIMAL 32).

SEEK FAILURE ON PAGDEV (AINIT)

THE INITIAL SEEK TO CYLINDER 0 ON THE PAGING DEVICE FAILED.

SEEK FAILURE ON ALTDEV (AINIT)

THE INITIAL SEEK TO CYLINDER 0 ON THE ALTERNATE PAGING DEVICE FAILED.

<FILE-SYSTEM-MESSAGE> CAN'T ATTACH TO CMDNC0 (AINIT)

A FILE SYSTEM ERROR WAS ENCOUNTERED WHILE ATTEMPTING TO ATTACH TO CMDNC0 FOR USER 1.

BAD CONFIG COMMAND: <CMND> (AINIT)

THE COMMAND <CMND> IN THE CONFIGURATION COMMAND FILE IS NOT A RECOGNIZED CONFIGURATION COMMAND.

BAD <CMND> PARAMETER (AINIT)

ONE OR MORE OF THE PARAMETERS SPECIFIED FOR THE CONFIGURATION COMMAND <CMND> IS INVALID.

BAD LINE # IN AMLBUF CMND (AINIT)

AN AMLBUF COMMAND SPECIFIES AN INVALID LINE NUMBER.

BAD DMQ AMLC CONFIGURATION (AINIT)

A DMQ BUFFER SIZE IN AN AMLBUF COMMAND WAS TOO LARGE OR NOT EQUAL TO A POWER OF 2.

BAD LINE # IN ASRBUF CMND (AINIT)

AN ASRBUF COMMAND SPECIFIED AN INVALID LINE NUMBER.

TERMINAL I/O BUFFERS TOO LARGE (AINIT)

THE TOTAL SIZE OF THE TERMINAL I/O BUFFERS EXCEEDS 32K WORDS.

SMLC CTRLR # OUT OF RANGE (AINIT)

AN SMLC COMMAND SPECIFIES AN INVALID CONTROLLER NUMBER.

SMLC LINE # OUT OF RANGE (AINIT)

AN SMLC COMMAND SPECIFIES AN INVALID LINE NUMBER.

NETWORK NOT CONFIGURED (AINIT)

A NODE NAME WAS SPECIFIED ON A SYSTEM NOT CONFIGURED FOR NETWORKS.

UNKNOWN NODE NAME (AINIT)

THE NETWORK NODE NAME SPECIFIED DOES NOT MATCH ANY NAME IN THE NETWORK CONFIGURATION TABLES.

RESTART PLEASE

THIS MESSAGE APPEARS FOLLOWING ANY ERROR MESSAGE PRINTED BY THE PRIMOS IV INITIALIZATION LOGIC (AINIT). THE SYSTEM WILL HALT THE LOCATION BOOTO_ IN SEGMENT 6. PRIMOS II MUST BE RELOADED. THE OFFENDING COMMAND IN THE CONFIGURATION DATA FILE MUST BE CORRECTED.

8 MODIFICATIONS TO INTERNAL LOGIC

THE FOLLOWING DESCRIBES THE MAJOR MODIFICATIONS THAT HAVE BEEN MADE TO THE INTERNAL LOGIC OF PRIMOS IV. THIS INFORMATION IS REQUIRED NORMALLY ONLY BY THOSE INVOLVED IN THE MODIFICATION OR MAINTENANCE OF PRIMOS IV.

8.1 PRIMOS LOAD ORDER

THE PRIMOS LOAD PROCEDURE (C_LOAD) NOW TAKES ADVANTAGE OF THE NEW SEG MIX COMMAND, WHICH ALLOWS INTERMIXING OF PROCEDURE AND LINK FRAMES. WITH TWO EXCEPTIONS, ALL PROCEDURES ARE NOW FOLLOWED IMMEDIATELY BY THEIR LINK FRAMES. THE EXCEPTIONS ARE DVDISK (AND ANY OTHER SEG0 PROCEDURES), WHOSE LINKS ARE LOADED INTO SEGMENT 6 AND SEG4, WHICH IS NOW REQUIRED TO GENERATE NO LINKS.

8.2 SYSTEM INITIALIZATION -- AINIT

THE PRELOADER PRIMOS NOW PROCESSES BOTH THE OLD-STYLE CONFIG COMMAND AS WELL AS THE NEW-STYLE CONFIGURATION SUBCOMMANDS CORRESPONDING THE OLD-STYLE PARAMETERS. ALL OTHER SUBCOMMANDS ARE LEFT FOR PROCESSING BY AINIT. THE NAME OF THE CONFIGURATION DATA FILE IS PASSED TO AINIT IN SEGMENT 1, WORD 0 BY THE PRELOADER.

AINIT IGNORES ALL SUBCOMMANDS THAT HAVE BEEN PROCESSED BY THE PRELOADER. ALL NEW CONFIGURATION COMMANDS CAUSE SETTING OF THE APPROPRIATE VARIABLES IN PRIMOS. ANY ERROR WILL CAUSE AN ERROR MESSAGE TO BE PRINTED, FOLLOWED BY A HALT VIA A CALL TO BOOT (SYMBOL BOOT0_ IN THE MAP).

PROCESSING OF THE NET, FAM, AND RLOGIN COMMANDS IS PERFORMED BY A NEW ROUTINE -- PRI400>NS>NETCLD -- THAT SETS UP THE NETWORK CONFIGURATION TABLES.

8.3 NEW FILE SYSTEM LOCKS

THE PURPOSE OF THE NEW FILE SYSTEM LOCKS IS TO ALLOW GREATER CONCURRENCY IN THE FILE SYSTEM AT REV 15. THE ACTUAL MEANS OF INTEGRATION OF THESE LOCKS INTO THE FILE SYSTEM WILL NOT BE DESCRIBED HERE, BUT IT BASICALLY CONSISTS OF CHANGING FILE SYSTEM MODULES TO UTILIZE A SET OF THESE LOCKS DURING THEIR OPERATION.

AN N-READERS-1-WRITER-LOCK (HEREAFTER SIMPLY "LOCK") FUNCTIONS AS A RESOURCE CONTROL WHICH CAN EITHER BE HELD JOINTLY BY N PROCESSES DOING READ OPERATIONS ON THE RESOURCE, OR BY A SINGLE PROCESS DOING A WRITE OPERATION, BUT NOT BOTH. THE UTILITY OF THIS KIND OF LOCK IN FILE SYSTEM APPLICATIONS IS READILY APPARENT. IT IS, HOWEVER, LIKELY THAT A WRITER WOULD EXPERIENCE LONG WAITS FOR A PARTICULAR LOCK, SINCE READ OPERATIONS PREDOMINATE IN THE FILE SYSTEM (ESPECIALLY FOR UFDS). THE SOLUTION TO THIS PROBLEM IS TO FORCE A NEW READER

PROCESS TO WAIT FOR THE LOCK IF EITHER (1) THE LOCK IS LOCKED FOR WRITING, OR (2) THERE IS AT LEAST ONE WRITER WAITING FOR THE LOCK. THIS ASSURES THAT THE MAXIMUM NUMBER OF PROCESSES THAT MUST RELINQUISH A LOCK BEFORE THE NEXT WRITER GETS CONTROL IS THE NUMBER OF READERS ALREADY OWNING THE LOCK WHEN THAT WRITER REQUESTED IT. THUS, THE WAIT TIME IS BOUNDED.

8.3.1 STRUCTURE AND OPERATION OF LOCKS

THE STRUCTURE OF A LOCK IS GIVEN BY THE FOLLOWING PL/I-ISH DECLARATION:

```
DCL 1 N1LOCK,
      2 SEM_R, /* READERS WAIT SEMAPHORE */
        3 COUNT FIXED BIN(15),
        3 WAITLIST OFFSET,
      2 SEM_W, /* WRITERS WAIT SEMAPHORE */
        3 COUNT FIXED BIN(15),
        3 WAITLIST OFFSET,
      2 UCOUNT FIXED BIN(15), /* USAGE COUNTER */
      2 PRIORITY BIT(16); /* FOR DEADLOCK AVOIDANCE */
```

THE SEMAPHORES ARE STANDARD P-400 SEMAPHORES. N1LOCK.UCOUNT IS AN INTEGER COUNTER WITH THE FOLLOWING POSSIBLE VALUES:

```
 0, IF THE LOCK IS FREE (INACTIVE),
+N, IF THERE ARE N ACTIVE READERS,
-1, IF THERE IS 1 ACTIVE WRITER.
```

N1LOCK.PRIORITY IS THE LOCK PRIORITY ('0001'B4 BEING TOPMOST). THESE PRIORITIES FORM A TOTAL ORDER ON LOCKS, AND ARE USED TO PREVENT DEADLOCKS. THE PRIORITIES ARE STATICALLY ASSIGNED IN THE PROGRAM TEXT WHERE THE LOCK VARIABLES ARE DEFINED.

IN ADDITION, THE LOCK PRIMITIVES MAINTAIN, FOR EACH PROCESS, A BITMAP OF THE LOCKS EACH PROCESS OWNS. THEREFORE, ATTEMPTS TO LOCK A LOCK WITH A LOWER PRIORITY NUMBER THAN ONE ALREADY OWNED, OR TO UNLOCK A LOCK NOT OWNED BY THE CALLER, ARE TRAPPED. THIS DATABASE ALSO PROVIDES THE MEANS TO FORCE A PROCESS TO UNLOCK ALL OF ITS LOCKS.

IN GENERAL, LOCKS ARE NOT RECURSIVELY LOCKABLE; AN ATTEMPT TO LOCK AGAIN A LOCK ALREADY HELD BY THE CALLING PROCESS IS DISALLOWED. BECAUSE OF THE STRUCTURE OF THE USER INTERFACE TO THE FILE SYSTEM, HOWEVER, IT WAS NECESSARY TO ALLOW THE TOPMOST LOCK (CALLED FSLOK FOR "FILE SYSTEM LOCK") TO BE RECURSIVELY LOCKED FOR READING. THE LOCK PRIMITIVES KEEP TRACK OF THE DEPTH OF THIS RECURSION FOR THE CALLER.

8.3.2 INTERACTION WITH THE PROCESS ABORT MECHANISM

IT IS DESIRED TO HOLD OFF ALL PROCESS ABORT FAULTS WHILE A PROCESS HAS ANY RING 0 LOCK SET (FOR READING OR WRITING), SINCE THERE IS NO WAY TO MASK THE FAULTS AT THE CPU LEVEL (EXCEPT BY INHIBITING INTERRUPTS), THE RING 0 PROCESS ABORT FAULT HANDLER HAS BEEN INSTRUCTED TO EXAMINE THE STATE OF A PROCESS' LOCKS DATABASE BEFORE CALLING THE ACTION ROUTINE ABORT PABORT. IF THE PROCESS HAS ANY RING 0 LOCK SET (AS EVIDENCED BY VARIABLE PUDCOM.LOKOWN BEING NONZERO), THE ABORT IS SAVED IN PUDCOM.ABSAVE AND THE FAULT IS RETURNED FROM. WHEN THE LAST LOCK IS RELEASED VIA A CALL TO UNLKN, UNLOCK, UNLKFS, OR UNLKF, THIS SAVED ABORT INFORMATION IS USED TO INVOKE THE DEFERRED PROCESS ABORT FAULT HANDLER.

HENCE, PROCESS ABORTS ARE DEFERRED WHENEVER PUDCOM.LOKOWN IS NONZERO.

THE LOCK DATABASE IN PUDCOM CONSISTS OF THE FOLLOWING VARIABLES:

PUDCOM.LOKOWN IS A BIT MAP LISTING ALL LOCKS OWNED BY THE PROCESS. FSLOK IS RECORDED IN BIT 16, UFDLOK IN BIT 15, AND SO ON.

PUDCOM.OWNFS IS THE RECURSION COUNTER FOR FSLOK, AND IS 0 WHEN FSLOK IS UNOWNED BY THE PROCESS. WHEN FSLOK IS OWNED BY THE PROCESS, IT IS ONE LESS THAN THE CURRENT LOCK RECURSION DEPTH.

8.4 MODIFICATIONS TO LOCATE AND ASSOCIATIVE BUFFER LOGIC

THE LOCATE MODULE, WHICH IS USED BY THE FILE SYSTEM TO READ AND MODIFY FILE SYSTEM RECORDS ON DISK, HAS BEEN MODIFIED EXTENSIVELY. PREVIOUSLY, ONLY ONE PROCESS AT A TIME WAS ABLE TO ACCESS ONE DISK RECORD AT A TIME. NOW, MANY PROCESSES MAY EACH ACCESS ONE DISK RECORD, AND, IN ADDITION, DISK RECORDS CAN BE SHARED AMONG PROCESSES. THE "ASSOCIATIVE BUFFER" CHARACTERISTICS OF THE LOCATE DISK BUFFERS ARE RETAINED, WITH EACH PROCESS HAVING ITS OWN "ASSOCIATIVE BUFFER WINDOW" THAT REPLACES THE ONE SYSTEM-WIDE WINDOW USED IN PREVIOUS VERSIONS.

8.5 UNLOCKING OF RING 0 STACKS FOR LOGGED OUT USERS

THE PER-USER RING 0 STACKS THAT WERE PERMANENTLY LOCKED AT REV 14 ARE NOW UNLOCKED WHEN A USER PROCESS IS NOT LOGGED IN. THIS CHECK IS PERFORMED IN C1IN, WHICH, WHEN A LOGGED OUT PROCESS REQUESTS TERMINAL INPUT, CALLS A NEW ROUTINE -- UNLOAD -- THAT UNLOCKS THE RING 0 STACK AND WAITS ON THE USER'S BUFFER SEMAPHORE. WHEN TERMINAL INPUT IS READY, THE STACK IS RELOCKED, AND, IF NECESSARY, THE ASSISTANCE OF USER 1 IS CALLED FOR TO BRING THE RING 0 STACK BACK INTO MEMORY.

8.6 PHANT$ ROUTINE

THE STARTUP OF PHANTOM USERS (VIA SVC OR PHANTOM COMMAND) IS NOW HANDLED BY A NEW ROUTINE -- PRI400>KS>PHANT$. PHANT$ CONTAINS THE LOGIC FORMERLY IN DOSSUB, LOCKS THE PHANTOM'S RING 0 STACK, AND RETURNS THE NUMBER OF THE PHANTOM USER. (SEE SECTION 3.1 FOR THE CALLING SEQUENCE FOR PHANT$.)

8.7 NEW MAGTAPE HANDLING

TWO SEMAPHORES ARE NOW ASSOCIATED WITH EACH MAGTAPE CONTROLLER -- ONE (MTOLCK OR MT1LCK) IS A QUEUE OF USERS WAITING FOR THE CONTROLLER; THE OTHER (MTOSEM OR MT1SEM) IS USED BY A PROCESS WAITING FOR THE COMPLETION OF A PREVIOUS OPERATION INITIATED BY ITSELF. THE MTDONE LOGIC NOW NOTIFIES ALL WAITING PROCESSES, NOT JUST THE FIRST ON THE MTNLCK QUEUE.

A NEW ROUTINE -- MAGONF -- IS CALLED WHEN A TAPE UNIT IS UNASSIGNED OR WHEN A USER OWNING A TAPE UNIT LOGS OUT. IF THAT USER CURRENTLY HAS A TAPE OPERATION IN PROGRESS, THE TAPE CONTROLLER IS REINITIALIZED TO ENSURE THAT OTHER USERS CAN GAIN IMMEDIATE ACCESS TO THE CONTROLLER.

8.8 NETWORK MODIFICATIONS

PRIMENET SUPPORTS COMMUNICATIONS AMONG PRIME PROCESSORS OVER TWO DIFFERENT COMMUNICATIONS MEDIA: THE IPC AND THE HSSMLC. WHEN SO CONFIGURED, PRIMOS PROVIDES NETWORK SERVICES OVER ANY OR ALL OF THESE MEDIA TO A COLLECTION OF REMOTE NODES. AS A RESULT OF THIS INCREASED SUPPORT, THE USER-AVAILABLE NETWORKING AND INTERPROCESS COMMUNICATION PRIMITIVES HAVE BEEN SLIGHTLY MODIFIED. (THESE PRIMITIVES WERE ORIGINALLY DESCRIBED IN PE-T-284.) THE PRIMITIVES AFFECTED ARE CONECT, RJCON, GETCON, AND NTSTAT. THE CHANGES TO EACH ARE BRIEFLY DESCRIBED BELOW. IN ADDITION TO THESE CHANGES, THERE IS NOW A VMODE LIBRARY TO ALLOW 64V FORTRAN USERS TO ACCESS NETWORKS. IT RESIDES IN UFD FAM>LIB AS VNETLB.

CONECT: THE CONECT PRIMITIVE OPTIONAL PARAMETER 'NUMTYP' IS NOW FURTHER DEFINED. AS BEFORE, WHEN NUMTYP IS NOT SPECIFIED, PRIMOS WILL ATTEMPT TO ESTABLISH A CONNECTION USING AN IPC LINK. WHEN NUMTYP IS SPECIFIED, IT REFERS TO THE TYPE OF COMMUNICATIONS PATH TO USE FOR THE CONNECTION, AND (IF AVAILABLE) WHICH OF THE MULTIPLE PATHS OF THAT TYPE TO USE. THE LOW BYTE OF NUMTYP IS THE NETWORK TYPE, AND THE HIGH BYTE IS RESERVED FOR THE LINE NUMBER. CURRENT LEGAL VALUES FOR NUMTYP ARE: 0 (IPC), 1 (HSSMLC).

RJCON: THE PRIMITIVE SUPPLIED TO REJECT A CONNECTION NOW ALSO TAKES A 'NUMTYP' AS AN ADDITIONAL ARGUMENT. NUMTYP IS USED EXACTLY AS FOR CONECT. THE REST OF THE ARGUMENTS ARE UNAFFECTED. THE NEW CALLING SEQUENCE FOR RJCON IS:

```
CALL RJCON (NODNAM, USER, STATUS, NUMTYP)
```

GETCON: THE STATUS PARAMETER TO A GETCON CALL IS NOW REQUIRED TO BE A TWO WORD ARRAY. THE FIRST WORD IS THE STATUS OF THE CALL (AS BEFORE). THE SECOND WORD IS NOW THE ANALOG TO THE NUMTYP PARAMETER IN THE CONECT CALL. THIS SECOND STATUS FIELD CONTAINS THE NUMBER THAT CORRESPONDS TO THE NETWORK TYPE OVER WHICH THE PENDING REQUEST CAME.

NISTAT: THE NTSTAT PRIMITIVE ALLOWS USERS TO OBTAIN CURRENT NETWORK STATUS INFORMATION. THE CHANGES MADE TO NTSTAT SIMPLY EXTEND THE STATUS REPORTING TO INCLUDE ALL TWO OF THE PRIMENET COMMUNICATIONS MEDIA. WHEN REQUESTING STATUS INFORMATION FOR A PARTICULAR NODE (NTSTAT KEYS 2 AND 3), THE USER MUST SPECIFY IN THE 'P1' PARAMETER THE LINE NUMBER/NETWORK TYPE OF THE APPROPRIATE COMMUNICATIONS PATH. (AGAIN, THE FORM THIS ARGUMENT TAKES IS THE SAME AS THE 'NUMTYP' IN CONECT.) FURTHER, CALLING NTSTAT WITH A KEY OF 4 NOW FILLS IN THE FOURTH, FIFTH, AND SIXTH WORDS OF THE DATA ARRAY WITH THE NUMBER OF IPC AND HSSMLC NETWORK LINES TO THE SPECIFIED NODE.

DATE: FEBRUARY 28, 1978

SUBJECT: REMOTE LOGIN

A NEW FEATURE HAS BEEN ADDED TO REV. 15.0 PRIMOS IV AND PRIMOS V CALLED REMOTE LOGIN. THIS MEMO DESCRIBES ITS USAGE, DESIGN AND IMPLEMENTATION.

I. PURPOSE

REMOTE LOGIN ALLOWS A USER AT ANY TERMINAL CONNECTED (HARD-WIRED OR DIAL-UP) TO ANY P400/P500 NODE IN A PRIMENET NETWORK TO LOG INTO AND INTERACT WITH ANY P400/P500 NODE IN THAT NETWORK. IT DOES NOT ALLOW A TERMINAL TO BE CONNECTED TO MORE THAN ONE ACTIVE PROCESS AT ANY ONE TIME.

II. USAGE

1. LOGGING_IN

THE LOGIN COMMAND NOW ACCEPTS ADDITIONAL ARGUMENTS. (BRACKETS [--] INDICATE OPTIONAL ARGUMENTS. IF PRESENT, ARGUMENTS MUST APPEAR IN THE ORDER SHOWN.)

```
LOGIN UFDNAME [PASSWORD] [-ON SYSNAME]
```

WHERE SYSNAME IS THE NAME OF A NODE IN PRIMENET. IF "-ON SYSNAME" IS OMITTED, AN ATTEMPT WILL BE MADE TO LOG INTO UFDNAME ON THE LOCAL SYSTEM ONLY. IF "-ON SYSNAME" IS SUPPLIED, THE TERMINAL WILL BE CONNECTED VIA PRIMENET TO A PROCESS ON NODE SYSNAME, IF ONE IS AVAILABLE. IF SYSNAME IS THE NAME OF THE LOCAL NODE, THE LOGIN ATTEMPT WILL BE MADE LOCALLY WITHOUT THE USE OF PRIMENET.

IF THE LOGIN COMMAND FAILS FOR ANY REASON (E.G., NOT FOUND, NO RIGHT, BAD PASSWORD), THE PRIMENET CONNECTION WILL BE BROKEN, AND THE TERMINAL RECONNECTED TO A LOCAL (NOT LOGGED-IN) PROCESS.

2. ERRORS

IN ADDITION TO NORMAL LOGIN ERRORS, THE FOLLOWING NEW ERRORS CAN OCCUR.

SYSNAME LINE DOWN

THE PRIMENET LINK TO NODE SYSNAME USED BY REMOTE LOGIN IS NOT OPERATIONAL. REMOTE LOGIN IS RESTRICTED TO USING A SINGLE TYPE OF PRIMENET LINK AT A TIME. THEREFORE THIS MESSAGE IS POSSIBLE EVEN IF AN ALTERNATE LINK TO SYSNAME IS OPERATIONAL. (SEE THE RLOGIN CONFIGURATION STATEMENT.)

INVALID SYSTEM NAME

THE SYSTEM NAME SPECIFIED IN THE LOGIN COMMAND IS NOT DEFINED.

NO MORE REMOTE TTY(S)

THE MAXIMUM NUMBER OF LOCAL TERMINALS ALLOWED TO LOGIN TO REMOTE SYSTEMS HAS BEEN REACHED. TRY AGAIN AT A LATER TIME.

NO FREE REMOTE USERS

THE MAXIMUM NUMBER OF PROCESSES CONFIGURED FOR USE BY REMOTE TERMINALS HAS BEEN REACHED ON THE INDICATED REMOTE SYSTEM. TRY AGAIN AT A LATER TIME.

LOGIN NOT ALLOWED TO REMOTE SYSTEM

LOCAL SYSTEM CONFIGURATION PARAMETERS HAVE DISALLOWED REMOTE LOGIN TO THE SPECIFIED SYSTEM.

LOGIN NOT SUPPORTED TO REMOTE SYSTEM

THE SPECIFIED REMOTE SYSTEM DOES NOT SUPPORT REMOTE LOGIN.

BAD LOGIN

OTHER SYNTAX OR SPELLING ERROR IN THE LOGIN COMMAND, OR MALFUNCTION.

3. LOGGING OUT

ISSUING THE COMMAND

```
LOGOUT
```

ON A TERMINAL LOGGED IN TO A REMOTE PROCESS, IN ADDITION TO LOGGING OUT THE PROCESS, WILL BREAK THE REMOTE CONNECTION OVER PRIMENET, AND RECONNECT THE TERMINAL TO ITS LOCAL PROCESS. DUE TO NETWORK DELAYS, ALL INPUT CHARACTERS TYPED BETWEEN THE LOGOUT COMMAND AND THE RESPONSE 'OK', ARE DISCARDED.

4. STATUS COMMAND

THE STATUS USERS COMMAND REPORTS BOTH LOCAL TERMINALS LOGGED INTO REMOTE PROCESSES, AND LOCAL PROCESSES SERVING REMOTE TERMINALS, AS FOLLOWS:

```
USER     NO    LIN    PDEVS

SMITH     6      4    (TO     SYSB USR#43)
JONES    49     75    (FROM   SYSB TTY#18)
```

THE LINE FOR USER SMITH SHOWS THAT HE IS LOGGED IN VIA TERMINAL 6 (CONNECTED TO AMLC LINE 4) TO USER PROCESS 43 ON REMOTE NODE SYSB. THE LINE FOR USER JONES SHOWS THAT HE IS LOGGED INTO LOCAL USER PROCESS 49. THE 75 UNDER LIN INDICATES THAT HIS TERMINAL I/O FLOWS OVER PRIMENET, TO TERMINAL 18 ON REMOTE NODE SYSB.

5. FORCED LOGOUT

THE COMMAND

```
LOGOUT -USERNO
```

WILL FORCE THE LOGOUT OF USERS CONNECTED VIA PRIMENET. USERNO IS THE NUMBER OF A LOCAL USER PROCESS, AS SHOWN IN THE NO COLUMN OF A STATUS USERS LISTING, WHETHER OR NOT THE USER IS ON A LOCAL OR REMOTE TERMINAL, USING A LOCAL OR REMOTE PROCESS.

IN ALL CASES, THE PROCESS IS LOGGED OUT AND RETURNED TO A POOL OF AVAILABLE REMOTE LOGIN SERVER PROCESSES, THE PRIMENET CONNECTION FOR THIS TERMINAL/PROCESS IS BROKEN, AND THE TERMINAL IS RECONNECTED TO ITS LOCAL PROCESS.

6. FORCED DISCONNECT

IN THE EVENT OF A MISHAP, THE CONNECTION OF A PROCESS TO A TERMINAL OVER PRIMENET MAY BE FORCIBLY BROKEN BY THE COMMAND

DISCON -USERNO

ISSUED FROM THE SUPERVISOR PROCESS ONLY, WHERE USERNO IS THE SAME AS FOR FORCED LOGOUT.

IF USERNO CORRESPONDS TO A LOCAL TERMINAL USING A REMOTE PROCESS, THE TERMINAL WILL BE RECONNECTED TO ITS LOCAL (PREVIOUSLY DORMANT) PROCESS. IF USERNO CORRESPONDS TO A LOCAL PROCESS IN USE BY A REMOTE TERMINAL, THE PROCESS WILL BE LOGGED OUT AND RETURNED TO THE POOL OF FREE REMOTE LOGIN SERVER PROCESSES.

THIS COMMAND SHOULD BE USED WITH GREAT CARE.

7. FAILURES

IF AN UNRECOVERABLE COMMUNICATIONS LINE FAILURE OCCURS, OR IF A REMOTE NODE CRASHES OR IS HALTED, THE LOCAL NODE WILL FORCE LOGOUT ALL LOCAL PROCESSES IN USE BY REMOTE TERMINALS ON THE FAILED LINE OR SYSTEM, AND WILL FORCE DISCONNECT ALL LOCAL TERMINALS THAT WERE LOGGED INTO PROCESSES ON THE REMOTE SYSTEM.

ON A WARM START AFTER A FAILURE, ALL LOCAL PROCESSES IN USE BY REMOTE TERMINALS WILL BE LOGGED OUT AND RETURNED TO THE POOL OF FREE REMOTE SERVER PROCESSES. ALSO, ALL LOCAL TERMINALS USING REMOTE PROCESSES WILL BE RECONNECTED TO THEIR LOCAL PROCESSES.

ON A COLD START OF A NODE, ALL TERMINALS ON REMOTE NODES THAT WERE USING PROCESSES ON THE COLD-STARTED NODE WILL BE DISCONNECTED AND RECONNECTED TO THEIR LOCAL PROCESSES. ALL PROCESSES ON REMOTE NODES THAT WERE IN USE BY TERMINALS ON THE COLD-STARTED NODE WILL BE LOGGED OUT.

III. CONFIGURATION CONTROL

AT PRESENT, TWO ADMINISTRATIVE CONTROLS GOVERN THE USE OF REMOTE LOGIN.

THE NUMBER OF LOCAL USER PROCESSES THAT ARE RESERVED FOR USE BY TERMINALS ON REMOTE NODES IS SPECIFIABLE BY THE EIGHTH NUMERICAL PARAMETER TO THE COLD START CONFIG COMMAND (SPECIFIED BY '7/), AND BY THE NRUSR CONFIGURATION STATEMENT IN THE COLD START CONFIGURATION FILE. (SEE PE-T-412.) THE NUMBER MUST BE LESS THAN OR EQUAL TO 32, THE MAXIMUM NUMBER OF LOCAL PROCESSES THAT MAY BE IN USE REMOTELY AT ANY ONE TIME. IF NOT SPECIFIED, THIS NUMBER DEFAULTS TO 0. THE NUMBER OF PROCESSES USED IS ADDED TO THE TOTAL NUMBER OF TERMINAL USERS SPECIFIED IN THE CONFIG

COMMAND. THESE PROCESSES DO NOT OCCUPY AMLC LINES. ASSIGNABLE AMLC LINES ARE ALLOCATED STARTING NUMERICALLY AFTER THE NUMBER OF LOCAL TERMINAL USERS SPECIFIED BY THE FIRST OCTAL PARAMETER TO THE CONFIG COMMAND.

THE COMMUNICATIONS LINE TYPE (IPC, RING, SMLC) TO BE USED FOR REMOTE LOGINS FROM TERMINALS ON THE LOCAL NODE TO PROCESSES ON A SPECIFIC REMOTE NODE IS SET BY THE RLOGIN STATEMENT IN THE COLD START CONFIGURATION FILE. (SEE PE-T-412.) IF THIS STATEMENT IS MISSING FROM THE CONFIGURATION FILE, REMOTE LOGINS WILL NOT BE ALLOWED TO THIS NODE.

IV. DESIGN AND IMPLEMENTATION

THIS SECTION DESCRIBES THE IMPLEMENTATION OF REMOTE LOGIN ON REV. 15.0 PRIMOS IV AND V ONLY. THESE DETAILS MAY CHANGE WITHOUT NOTICE.

1. GENERAL OPERATION

REMOTE LOGIN USES PRIMENET TO MOVE CHARACTERS BETWEEN THE LOW-SPEED TERMINAL I/O BUFFERS OF A TERMINAL ON ONE NODE AND THOSE OF A PROCESS ON ANOTHER NODE. INPUT CHARACTERS TYPED ON THE TERMINAL FIRST APPEAR IN AN INPUT BUFFER ON THE LOCAL NODE. THESE ARE MOVED QUICKLY (1/2 SECOND OR LESS) TO THE INPUT BUFFER OF THE USER PROCESS ON THE REMOTE NODE, WHERE THEY ARE CONSUMED NORMALLY. OUTPUT CHARACTERS PRODUCED BY THE REMOTE PROCESS ARE PLACED INITIALLY IN THE PROCESS' OUTPUT BUFFER IN THE REMOTE MODE. THEY ARE THEN MOVED TO THE OUTPUT BUFFER OF THE TERMINAL ON THE LOCAL MACHINE, WHERE THEY ARE DELIVERED TO THE TERMINAL.

ANY LOCAL TERMINAL (INCLUDING THE SYSTEM CONSOLE) MAY BE USED TO LOG INTO A REMOTE PROCESS. WHILE THE TERMINAL IS CONNECTED REMOTELY, ITS CORRESPONDING LOCAL PROCESS REMAINS INACTIVE UNTIL EXCEPTIONAL EVENTS OCCUR (E.G. QUIT, LOGOUT).

EACH PRIMENET NODE HAS A POOL OF FREE PROCESSES AVAILABLE FOR USE BY TERMINALS ON OTHER NODES. THESE PROCESSES ARE ALLOCATED ON DEMAND, AND FREED WHEN A REMOTE USER LOGS OUT.

2. TABLE STRUCTURE

REMOTE LOGIN USES TWO TABLES DEFINED IN COMDEF/NETCOM. RTTYT DESCRIBES LOCAL TERMINALS CURRENTLY USING REMOTE PROCESSES. RUSRT DESCRIBES LOCAL PROCESSES IN USE BY REMOTE TERMINALS. EACH TABLE CURRENTLY HAS 32 ENTRIES, 8 WORDS EACH. EACH COMPLETED CONNECTION BETWEEN A TERMINAL AND A PROCESS USES AN RTTYT ENTRY IN THE TERMINAL NODE AND AN RUSRT ENTRY IN THE PROCESS NODE. ENTRIES IN BOTH ARE SIMILAR:

WORD	1	TERMINAL # ON TERMINAL NODE (1-64)
	2	PROCESS # ON PROCESS NODE (1-64)
	3	PRIMENET VIRTUAL CIRCUIT ID (LUN ADDRESS)

4 ERROR CODE
5 SEQUENCE # OF LAST INPUT CHAR SENT/RECEIVED
6 SEQUENCE # OF LAST INPUT CHAR THAT MAY BE SENT/RECEIVED
7 SEQUENCE # OF LAST OUTPUT CHAR SENT/RECEIVED
8 SEQUENCE # OF LAST OUTPUT CHAR THAT MAY BE SENT/RECEIVED

ONLY THE LOCAL PROCESS OF A TERMINAL MODIFIES ITS ENTRY IN RTTYT, AT LOGIN, LOGOUT AND OTHER EVENTS CAUSING DISCONNECTION. ONLY A REMOTE LOGIN SERVER PROCESS, OR THE SUPERVISOR ACTING ON ITS BEHALF, UPDATES ITS ENTRY IN RUSRT. THUS, THESE TABLES ARE NOT REALLY SHARED RESOURCES AMONG PROCESSES. FOR THESE REASONS, THE TABLES NEED NOT BE LOCKED DURING SEARCH AND UPDATE OPERATIONS.

WORD 1 OF EACH ENTRY IS USED BOTH TO INDICATE AN ACTIVE CONNECTION, AND TO CONTROL TABLE SEARCHES. ITS VALUES ARE

-1 - ENTRY IS FREE, ACTIVE ENTRIES MAY FOLLOW
0 - ENTRY IS FREE, NO ACTIVE ENTIRES FOLLOW
>0 - ENTRY IS ACTIVE, VALUE IS TERMINAL # ON TERMINAL NODE

WORD 1 OF ALL ENTRIES IS INITIALLY 0. WHEN AN ENTRY IS USED, THE TERMINAL # (ALWAYS POSITIVE) IS SET INTO WORD 1. WHEN AN ENTRY IS FREED, WORD 1 IS SET TO -1. PERIODICALLY, ALL INACTIVE (0 OR -1) ENTRIES FOLLOWING THE HIGHEST NUMBERED ACTIVE ENTRY ARE SET TO 0. FOR EFFICIENCY, SEARCHES FOR ACTIVE ENTRIES PROCEED LINEARLY, TERMINATING AT THE FIRST 0 ENTRY.

3. FLOW CONTROL

WORDS 5-8 OF EACH TABLE ENTRY CONTROL THE FLOW OF CHARACTERS BETWEEN NODES. EACH CHARACTER FLOWING IN EACH DIRECTION ON EACH REMOTE LOGIN CONNECTION IS NUMBERED SEQUENTIALLY, MODULO 2**15. THE SENDING NODE MAINTAINS A PAIR OF COUNTERS INDICATING THE SEQUENCE NUMBER OF THE LAST CHARACTER SENT, AND THE CHARACTER NUMBER OF THE LAST CHARACTER THE RECEIVER HAS GIVEN PERMISSION TO SEND. AT ANY TIME, THE DIFFERENCE OF THESE NUMBERS IS THE NUMBER OF CHARACTERS THE SENDER MAY SEND. LIKEWISE, THE RECEIVING NODE MAINTAINS A PAIR OF COUNTERS OF THE NUMBER OF THE LAST CHARACTER IT ACTUALLY RECEIVED, AND THE NUMBER OF THE LAST CHARACTER IT HAS GIVEN THE SENDER PERMISSION TO TRANSMIT.

PERIODICALLY (EVERY 1/2 SECOND), THE RECEIVER COMPARES THE SPACE AVAILABLE IN ITS TERMINAL OUTPUT OR PROCESS INPUT BUFFER WITH THE DIFFERENCE OF ITS TWO COUNTERS. IF THERE HAS BEEN A CHANGE (THE PROCESS HAS CONSUMED MORE INPUT OR THE TERMINAL HAS TYPED SOME OUTPUT), THE RECEIVER UPDATES ITS PERMISSION TO SEND COUNTER AND TRANSMITS THE NEW VALUE TO THE SENDER.

EVERY 1/2 SECOND, OR WHEN THE SENDER OF CHARACTERS RECEIVES AN UPDATED PERMISSION TO SEND SEQUENCE NUMBER, THE SENDER EXAMINES ITS TERMINAL INPUT OR PROCESS OUTPUT BUFFER. IF THERE ARE ANY CHARACTERS PRESENT, AND IF THERE IS A "LARGE ENOUGH" WINDOW (DIFFERENCE BETWEEN LAST SENT AND LAST PERMISSION-TO-SEND COUNTERS) THE CHARACTERS ARE REMOVED FROM THEIR BUFFER AND SENT TO THE RECEIVER, WITH THE LAST SENT COUNTER UPDATED. PARAMETERS CONTROL HOW LARGE A WINDOW IS NECESSARY BEFORE CHARACTERS CAN BE SENT. THESE PARAMETERS CONTROL THE TRADEOFF BETWEEN FREQUENCY OF NETWORK MESSAGES AND MESSAGE SIZE, AND ARE NOT CURRENTLY ADJUSTABLE.

NOTE THAT NO SCHEDULING OR RUNNING OF THE LOCAL PROCESS OF A TERMINAL OCCURS DURING NORMAL I/O.

4. NETWORK_USE

REMOTE LOGIN IS IMPLEMENTED AT THE LEVEL OF NETMAN. THAT IS, REMOTE LOGIN MESSAGES ORIGINATE AND TERMINATE WITHIN THE PRIMOS KERNEL. REMOTE LOGIN USES THE KERNEL NETWORK PRIMITIVES GETBLK, RTNBLK, AND TRNMSG.

REMOTE LOGIN MESSAGES OCCUPY NETMAN MESSAGE CLASS 1 (CLASS 0 IS USED TO SUPPORT USER PROCESS VIRTUAL CIRCUITS.)

5. MODULES

THE MAJOR MODULES OF REMOTE LOGIN RESIDE IN THE SOURCE FILE RLOGIN. THE MODULES ARE AS FOLLOWS.

RLOGIN - HANDLES REMOTE LOGIN AND FORCED LOGOUT OF LOCAL TERMINALS. CALLED ONLY BY DOSSUB.

RLGETT - FUNCTION LOCATES RTTYT ENTRY OF A GIVEN LOCAL TERMINAL USING A REMOTE PROCESS.

RLGETU - FUNCTION LOCATES RUSRT ENTRY OF A GIVEN LOCAL PROCESS IN USE BY A REMOTE TERMINAL.

RLHDFL - FILLS THE REMOTE LOGIN MESSAGE HEADER IN A GIVEN MESSAGE BLOCK.

RLSEND - SENDS A SHORT (NON-DATA) REMOTE LOGIN MESSAGE.

RLSNDT - SENDS A SHORT (NON-DATA) MESSAGE FOR A SPECIFIC LOCAL TERMINAL.

RLSNDU - SENDS A SHORT (NON-DATA) MESSAGE FOR A SPECIFIC LOCAL PROCESS.

RLINOT - SENDS A LONG DATA MESSAGE, REMOVING CHARACTERS FROM A SPECIFIED I/O BUFFER.

RLPOLL - SENDS ALL PERMISSION-TO-SEND WINDOW MESSAGES. SENDS CHARACTER MESSAGES WHEN SUFFICIENT WINDOW EXISTS. CALLED EVERY 1/2 SECOND FROM PABORT..

RLSRVR - RECEIVES ALL REMOTE LOGIN (NETMAN CLASS 1) MESSAGES. SENDS CHARACTERS IF NEEDED WHEN WINDOW UPDATES ARRIVE. CALLED BY NETMAN WHEN A CLASS 1 MESSAGE IS RECEIVED.

6. LOGGING IN

A TERMINAL IS INITIALLY CONNECTED TO ITS LOCAL PROCESS, WITH DOSSUB PROCESSING COMMANDS. WHEN A -ON OPTION IS SUPPLIED TO THE LOGIN COMMAND, THE MODULE RLOGIN IS INVOKED. RLOGIN VALIDATES THE SYSTEM NAME ARGUMENT AND LOCATES THE USER 1 VIRTUAL CIRCUIT (LUN) TO THE INDICATED SYSTEM, USING THE DEFAULT OR SPECIFIED COMMUNICATIONS LINE TYPE. IT RE-READS THE TYPED COMMAND LINE FROM THE COMMAND BUFFER AND PLACES IT VERBATIM IN A MESSAGE THAT IS THEN SENT TO THE REMOTE SYSTEM.

THE MESSAGE IS RECEIVED BY NETMAN AND PASSED TO RLSRVR. IF A FREE PROCESS IS AVAILABLE IT IS ASSIGNED TO THE REQUESTING TERMINAL. (IF A PROCESS WAS IN USE BY THE REQUESTING TERMINAL, BUT WAS DISCONNECTED DUE FOR EXAMPLE, TO A CONTROL PANEL STOP/START SEQUENCE, IT IS FORCE LOGGED OUT.)

THE ORIGINAL LOGIN LINE IS INSERTED INTO THE INPUT BUFFER, FROM WHICH DOSSUB OBTAINS IT. THE REMOTE MACHINE THEN RETURNS A MESSAGE CONTAINING THE USER NUMBER OF THE PROCESS CHOSEN. AT THIS POINT, EACH MACHINE KNOWS OF THE BINDING BETWEEN TERMINAL AND USER. THE LOCAL PROCESS NOW WAITS ON ITS NTWAIT SEMAPHONE UNTIL AN EXCEPTIONAL EVENT OCCURS (QUIT, LOGOUT, LINE DOWN, ETC.). INPUT AND OUTPUT PROCEED AS DESCRIBED UNDER FLOW CONTROL.

IF THE LOGIN COMMAND IS UNSUCCESSFUL (DUE TO ERRORS SUCH AS NOT FOUND, NO RIGHT, OR BAD PASSWORD), OR IF NO REMOTE PROCESS IS AVAILABLE, THE REMOTE SYSTEM RETURNS A DISCONNECT MESSAGE WITH AN ERROR CODE INDICATING THE CAUSE OF ERROR, WHICH IS PRINTED ON THE USER'S TERMINAL.

7. QUIT

THE QUIT SIGNAL (CAUSED BY A USER HITTING BREAK OR CTRL/P ON THE TERMINAL) CAUSES A PROCESS ABORT IN THE LOCAL PROCESS. IF THE LOCAL TERMINAL IS LOGGED INTO A REMOTE PROCESS, A QUIT MESSAGE IS SENT, WHICH CAUSES A QUIT TO OCCUR IN THE REMOTE PROCESS. TO AVOID EXCESSIVE TERMINAL OUTPUT DUE TO NETWORK DELAYS, OUTPUT BUFFERS ARE CLEARED IN BOTH MACHINES ON A QUIT.

8. LOGGING OUT

THE LOGOUT COMMAND OPERATES IDENTICALLY ON LOCAL AND REMOTE PROCESS. (THE LOCAL PROCESS CORRESPONDING TO THE TERMINAL DOES NOT INTERPRET COMMANDS.) IN ADDITION, IF THE PROCESS WAS SERVING A TERMINAL ON A REMOTE NODE, A DISCONNECT MESSAGE IS RETURNED TO THE TERMINAL NODE, AND BOTH SIDES CLEAR THEIR TERMINAL/PROCESS BINDING .

THERE IS A 1-SECOND DELAY BEFORE THE REMOTE NODE SENDS THE DISCONNECT MESSAGE TO ALLOW RLPOLL TO SEND THE LOGOUT MESSAGE

OVER PRIMENET.

TYPING THE LOGIN COMMAND TO A LOGGED-IN PROCESS, EITHER LOCAL OR REMOTE, WILL RESULT IN AN IMPLICIT LOGOUT ONLY IF THE TARGET SYSTEM IS THE SAME AS THE CURRENT SYSTEM. THUS, WHEN LOGGED INTO SYSTEM SYSA, FROM EITHER A LOCAL OR REMOTE TERMINAL, THE FOLLOWING COMMANDS WILL PRODUCE AN IMPLICIT LOGOUT BEFORE LOGGING IN AGAIN.

```
LOGIN  UFDNAME
LOGIN  UFDNAME  -ON SYSA
```

A LOGIN COMMAND SPECIFYING ANY OTHER SYSTEM NAME WILL RESULT IN THE MESSAGE

```
PLEASE LOGOUT
```

THIS FORCES CONTROL OF THE TERMINAL TO ITS LOCAL SYSTEM BEFORE CHANGING NODES, AND IS DESIGNED TO PREVENT A USER FROM LOGGING IN THROUGH MULTIPLE NODES.

9. STOP/START AND WARM START

WHEN A REMOTE MACHINE HAS STOPPED RESPONDING TO NETWORK ACTIVITY, THERE IS NO WAY FOR THE LOCAL MACHINE TO DECIDE WHETHER THE REMOTE MACHINE WAS MANUALLY HALTED AND WILL BE RESTARTED IMMEDIATELY WITH NO CHANGES, OR WILL BE WARM- OR COLD-STARTED. THEREFORE, WHEN THE LOCAL MACHINE DETECTS AN INABILITY TO COMMUNICATE WITH A REMOTE NODE, ALL LOCAL PROCESSES IN USE BY TERMINALS ON THAT NODE ARE LOGGED OUT, AND ALL LOCAL TERMINALS LOGGED INTO PROCESSES ON THAT MACHINE ARE DISCONNECTED AND RECONNECTED TO THEIR LOCAL PROCESSES. THIS IS TANTAMOUNT TO ASSUMING THAT THE REMOTE NODE WILL BE COLD-STARTED. IT IS DONE PRIMARILY AS A FAIL-SAFE MEASURE, SO THAT HUMAN INTERVENTION AT THE SYSTEM CONSOLE IS NOT NECESSARY TO FREE UP TERMINALS AND PROCESSES CONNECTED TO A FAILED NODE.

IF A STOPPED NODE IS RESTARTED OR WARM STARTED, TERMINALS WILL BE RECONNECTED TO THEIR LOCAL PROCESSES, AND REMOTE LOGIN SERVER PROCESSES WILL LOG OUT.

SUBJECT: SEG FOR REV. 15

DEVELOPMENT EFFORTS ON SEG FOR REV. 15 HAVE BEEN CONCENTRATED IN 3 AREAS.

1) CONVERSION OF SOME OF SEG'S LOADER'S FUNCTIONALITY FROM PMA TO FORTRAN AND PRODUCTION OF A COMMON SET OF ROUTINES WHICH CAN BE USED BY BOTH SEG AND LOAD.

2) REDESIGN OF SOME OF SEG'S INTERNAL TABLES TO MAKE THE LOADING OF PROGRAMS TO SPECIFIC SEGMENTS EASIER.

3) FURTHER ENHANCEMENTS TO SEG'S FUNTIONALITY.

NOTE: IN ORDER TO PROVIDE FOR ENHANCED FUNCTIONALITY IN THE FUTURE AND TO ALLOW SUPPORT OF SHARED LIBRARIES AT REV. 15 WORDS 0 THROUGH 40 IN ALL PROCEDURE SEGMENTS MUST BE RESERVED FOR USE BY SEG AND THE OPERATING SYSTEM. THIS SHOULD NOT AFFECT MOST USERS, HOWEVER, IT DOES MEAN THAT THESE LOCATIONS CAN NOT BE USED AS CONSTANTS OR TEMPORARIES. SEGMENT 4000 WILL BE CONSIDERED A PROCEDURE SEGMENT IN THIS CONTEXT AND USERS PLACING DATA IN SEGMENT 4000 MUST TAKE THE RESTRICTION INTO CONSIDERATION.

ALL SEG RUN FILES CREATED PRIOR TO REV. 15 CAN BE ACCESSED WITH REV. 15 SEG.

USER VISIBLE CHANGES WHICH RESULT FROM THIS EFFORT INCLUDE:

SEG'S SYMBOL TABLE SEARCH HAS BEEN SPEEDED UP AGAIN. LARGER LOADS WITH MORE SYMBOLS WILL BENEFIT MORE THAN SMALL LOADS WITH FEW SYMBOLS. REDUCTIONS IN LOAD TIMES OF UP TO 50% HAVE BEEN OBSERVED IN LOADING PRIMOS IV.

SEG WILL NOW SUPPORT UP TO 256 SEGMENTS. SPLIT SEGMENTS NO LONGER OCCUPY TWO SEGMENT POSITIONS.

ALL COMMANDS WHICH LEAVE SEG'S LOADER (QUIT, RETURN AND EXECUTE) NOW PERFORM THE SAVE FUNCTION. THIS REPRESENTS NO LOSS OF FUNCTIONALITY FOR THE USER AND INSURES THAT SEG'S DELETE COMMAND WILL WORK ON ALL SEG RUN FILES.

QUIT AND ATTACH MAY NOW BE ABREVIATED TO Q AND A RESPECTIVELY.

SEG'S LOADER NOW KEEPS TRACK OF THE LENGTH OF COMMON BLOCKS UP TO ONE SEGMENT IN LENGTH. AN ATTEMPT TO REDEFINE A COMMON BLOCK TO A LONGER LENGTH WILL CAUSE AN ERROR. REDEFINITION TO A SHORTER LENGTH IS NOT FLAGGED.

SEVERAL ENHANCEMENTS HAVE BEEN ADDED TO THE MAP COMMAND. FIRST TWO NEW MAP COMMANDS HAVE BEEN ADDED:

MA ... 10 CAUSES ALL SYMBOLS TO BE PRINTED OUT ONE PER LINE IN ASCENDING ORDER BY ADDRESS.

MA ... 11 CAUSES ALL SYMBOLS TO BE PRINTED OUT IN ALPHABETICAL ORDER, ONE PER LINE.

IN BOTH CASES THE SYMBOL IS DESCRIBED BY TYPE (COMMON, DIRECT ENTRY CALL, ETC).

MAP FORMAT HAS CHANGED SLIGHTLY TO ALLOW 8 CHARACTER FILE NAMES AND THE COMMON BLOCK SECTION HAS BEEN REFORMATTED TO INCLUDE THE LENGTH OF THE COMMON BLOCK WHEN THIS IS KNOWN. THE LENGTH OF THE COMMON BLOCK IS PRINTED IN OCTAL.

FURTHER, THE MAP FORMAT FOR PROCEEDURE SYMBOLS HAS BEEN ENHANCED. SEG WILL NOW PRINT OUT THE LENGTH OF THE LINK FRAME AND THE SEGMENT OF THE LINK FRAME WHEN THESE ARE KNOWN.

FINALLY, *SYM NOW REPORTS THE NUMBER OF SYMBOLS IN THE SYMBOL TABLE (IN OCTAL).

IT IS NO LONGER NECESSARY TO WORRY ABOUT MIXING DEFAULT LOADING (SEG ASSIGNS THE SEGMENT NUMBERS) AND SPECIFIC LOADING (THE USER ASSIGNS THE SEGMENT NUMBERS). THE COMMAND FORMATS HAVE NOT CHANGED, BUT SEG WILL NOW PREVENT SEGMENT ASSIGNMENT CONFLICTS. IN ADDITION SEG KEEPS TRACK OF THE END OF THE PROCEDURE PORTION OF A SPLIT SEGMENT AND WILL NOT LOAD PROCEDURE INTO THE DATA PORTION. ALSO A SEGMENT MAY BE SPLIT ANYWHERE (NOT JUST ON :4000 WORD BOUNDRIES). IN ADDITION, PROGRAMS MAY NOW BE LOADED UNDER THE 'MI' OPTION (SEE BELOW) WHICH PERMITS MIXING OF DATA AND PROCEDURE IN THE SAME UNSPLIT SEGMENT. THIS FEATURE MAY MAKE SPLIT SEGMENTS OBSOLETE.

SEG WILL NOW SAFELY ASSIGN A STACK TO A SHARED PROCEDURE IN THE USER'S SEGMENTS (ABOVE :4000) RATHER THAN PLACING IT IN THE FIRST AVAILABLE PROCEDURE SEGMENT WHICH IS USUALLY BELOW :4000.

THE SEGMENT DIRECTORY IS NO LONGER FORCED TO 32 TIMES THE NUMBER OF SEGMENTS PERMITTED. INSTEAD "SLOTS" IN THE SEGMENT DIRECTORY ARE ASSIGNED AS NEEDED. THE RESULT IS A SHORTER SEGMENT DIRECTORY WHICH REDUCES THE LENGTH OF TIME REQUIRED TO INITIALIZE OR DELETE A SEG RUNFILE.

SEG IS NOW ABLE TO OPTIMIZE COMMON REFERENCES TO COMMON BLOCKS LOCATED IN EITHER THE SEGMENT OF THE PROCEDURE, OR THE SEGMENT OF THE LINK FRAME. AS A RESULT, PROGRAMS MAKING EXTENSIVE USE OF COMMON MAY BE SPEEDED UP.

FLEX MESSAGES CONTAIN THE ADDRESS OF THE OFFENDING INSTRUCTION.

SEG'S WORKING SET IS BIGGER BY 1000 TO 2000 (OCTAL) WORDS WHICH IS LESS THAN A 5% INCREASE. THE RUN FILE HAS DECREASED IN SIZE SLIGHTLY, HOWEVER, THIS IS IN PART BECAUSE THE UII PACKAGE IS NO LONGER LOADED. SEG AT REV. 15 - AS DELIVERED - WILL NOT RUN ON A P300. SEG NOW USES SEGMENT 4001 AS ITS BUFFERS FOR LOADING. THIS IS RESPONSIBLE FOR PART OF THE INCREASED SPEED OF SEG ON LARGE LOADS.

THERE IS A NEW COMMAND AT SEG COMMAND LEVEL WHICH ALLOWS THE USER TO DETERMINE THE REV. OF SEG. THE FORMAT FOR THIS COMMAND IS 'VE(RSION)'.

SEG'S LOADER HAS FOUR NEW COMMANDS. THESE ARE:

SE(T BASE) - CREATE A BASE AREA FOR DESECTORIZATION. THE FORMAT OF THE COMMAND IS:

```
SE SEGNO LENGTH
```

'SEGNO' IS THE SEGMENT IN WHICH THE BASE AREA IS TO BE LOCATED. IT MUST BE A PROCEDURE SEGMENT - OR UNDEFINED. 'LENGTH' IS THE LENGTH OF THE BASE AREA TO BE CREATED. THE BASE AREA IS CREATED AT THE CURRENT 'TOP' OF THE SEGMENT. THERE IS NO FACILITY FOR PLACING A BASE AREA AT A SPECIFIC LOCATION IN A SEGMENT.

SS (SAVE SYMBOLS) - DECLARE SYMBOLS AS PERMANENT FOR THE XPUNGE COMMAND. THIS COMMAND PREVENTS XPUNGE FROM DELETING THE SPECIFIED SYMBOLS. THE FORMAT OF THE COMMAND IS:

```
SS SYMBOL
```

'SYMBOL' IS THE NAME OF A DEFINED SYMBOL. TO PROTECT MULTIPLE SYMBOLS EACH MUST BE NAMED IN AN SS COMMAND SEPRATELY.

MI(XUP) - COMMAND TO PERMIT LOADING OF LINKAGE AND COMMON AREAS IN PROCEDURE SEGMENTS. THIS COMMAND CAUSES SEG TO MIX PROCEDURE AND DATA IN THE SAME SEGMENTS. WHEN MI HAS BEEN INVOKED ALL SEGMENTS WILL BE CREATED AS PROCEDURE SEGMENTS. THE FORMAT OF THE COMMAND IS:

```
MI [ON]
MI OFF
```

MI OR MI ON INVOKES THE MI FEATURE. MI OFF TURNS THE FEATURE OFF AND REGULAR LOADING INTO SEPARATE DATA AND PROCEDURE SEGMENTS WILL RESUME. THE FEATURE IS NOT RESET BY THE INITIATE COMMAND. IN THIS WAY USERS MAY ELECT TO SAVE A COPY OF SEG WITH MI TURNED ON AND MAKE THIS THE DEFAULT MODE OF LOADING. IN GENERAL LOADING UNDER THE MI OPTION WILL REDUCE THE NUMBER OF SEGMENTS REQUIRED FOR A PROGRAM. HOWEVER, DEBUGGING SUCH PROGRAMS MAY BE MORE DIFFICULT.

MV (MOVE) - COMMAND TO MOVE PORTIONS OF THE LOAD FILE. THIS COMMAND IS INTENDED PRIMARILY TO FACILITATE THE CREATION OF SHARED LIBRARIES. CODE OR DATA MOVED BY THE MV COMMAND CANNOT BE EXECUTED OR USED IN THE DESTINATION LOCATION AS THE LINKS TO THE MOVED AREA STILL REFLECT THE ORIGINAL ADDRESSES. THE FORMAT OF THE COMMAND IS:

```
MV SSYMBL MBLOCK DSEGNO
MV
```

THE SECOND FORM OF THE COMMAND CAUSES THE MV COMMAND TO ASK FOR FURTHER INPUT PERMITTING GREATER FLEXIBILITY. FOR THE FIRST FORM OF THE COMMAND, 'SSYMBL' IS THE NAME OF A SYMBOL INDICATING THE START OF THE MOVE. INFORMATION WILL BE MOVED FROM 'SSYMBL' TO THE CURRENT 'HIGH' IN THE SEGMENT.

'MBLOCK' IS A PREVIOUSLY DEFINED SYMBOL CORRESPONDING TO A 5 WORD BLOCK INTO WHICH INFORMATIOON CONCERNING THE MOVE WILL BE PLACED. 'MBLOCK' IS OPTIONAL. IF 'MBLOCK' IS SPECIFIED IT FORMAT AFTER THE MOVE WILL BE:

```
WORDS 1,2 ADDRESS TO WHICH MOVE WAS MADE
WORDS 3,4 ADDRESS FROM WHICH MOVE WAS MADE
WORD 5 NUMBER OF WORDS MOVED
```

'MBLOCK' MAY THUS BE USED TO RESTORE THE MOVED INFORMATION TO ITS ORIGINAL PLACE.

'DSEGNO' IS THE SEGMENT TO WHICH THE INFORMATION IS TO BE MOVED. 'DSEGNO' MAY BE EITHER A PROCEDURE OR DATA SEGMENT. IF THERE IS NOT ENOUGH ROOM IN THE SEGMENT THE NEXT SEGMENT WITH SUFFICIENT ROOM WILL BE USED.

FOR THE SECOND FORM OF THE COMMAND MV WILL RESPOND WITH THE FOLLOWING QUERIES:

```
START:
END:
DEST. SEGMENT:
IP VECTOR:
```

THE RESPONSE TO START MAY BE EITHER A DEFINED SYMBOL FROM THE SYMBOL TABLE OR THE SEGMENT NUMBER WORD ADDRESS DEFINING THE

BEGINNING OF THE MOVE. FOR EXAMPLE:

START: FOOBAR

OR

START: 4001_1232

END MAY ALSO BE SPECIFIED AS A SYMBOL OR NUMERIC VALUE. IF END IS SPECIFIED SYMBOLICALLY IT MUST BE IN THE SAME SEGMENT AS THAT DEFINED FOR THE START OF THE MOVE. IF END IS DEFINED NUMERICALLY IT MUST BE ONE NUMBER REPRESENTING THE FIRST LOCATION WHICH IS NOT TO BE MOVED IN THE SEGMENT. FOR EXAMPLE:

END: FOOEND

OR

END: 2000

IN THE EITHER CASE LOCATIONS UP TO BUT NOT INCLUDING SPECIFIED LOCATION WILL BE MOVED. A VALUE OF 0 OR NO VALUE (CRLF ONLY) WILL CAUSE MV TO MOVE LOCATIONS UP TO AND INCLUDING THE CURRENT 'HIGH' IN THE SEGMENT SPECIFIED BY START.

DEST. SEGMENT IS A SEGMENT NUMBER INTO WHICH THE BLOCK OF INFORMATION IS TO BE MOVED. IT MAY BE EITHER A PROCEDURE OR DATA SEGMENT. IF THERE IS NOT ENOUGH ROOM IN THE SEGMENT, THE NEXT SEGMENT WITH ENOUGH ROOM WILL BE USED.

IP VECTOR CORRESPONDS TO 'MBLOCK' ABOVE. IT IS ALSO OPTIONAL.

DATE: MARCH 15, 1978

SUBJECT: SHARED LIBRARIES FOR REV. 15

AT REV. 15 PRIME IS MAKING SOME SHARED V-MODE LIBRARIES AVAILABLE TO THE USER COMMUNITY. PRIME GUARANTEES SUPPORT FOR THE V-MODE RUN FILES WHICH USERS CREATE FOR AN INDEFINITE PERIOD. HOWEVER, USERS MAY NOT ADD THEIR OWN SHARED LIBRARIES TO THEIR SYSTEM BUT ARE RESTRICTED TO USING THE FOUR SUPPLIED BY PRIME. THE LIBRARIES WHICH ARE ALL OR PARTIALLY SHARABLE AT REV. 15 ARE THE FORTRAN LIBRARY (THE I/O ROUTINES ONLY) AND KI/DA FOR ALL USERS; AND COBOL AND/OR FORMS FOR THOSE USERS WHO HAVE PURCHASED THESE PACKAGES.

USERS MAY NOT CREATE THEIR OWN SHARED LIBRARIES AT REV. 15 BECAUSE THE MECHANISM WILL BE EVOLVING TO PROVIDE MORE GENERAL SUPPORT FOR SHARING PROCEDURES THE FUTURE. THE PRIME SUPPLIED SHARED LIBRARIES WILL BE KEPT IN STEP WITH THIS EVOLUTION SO THAT USER RUNFILES USING THE SHARED LIBRARIES WILL CONTINUE TO WORK. IT IS NOT POSSIBLE TO MAKE THE SAME ASSURANCE FOR USER CREATED SHARED LIBRARIES AT THIS TIME.

THE REMAINDER OF THIS DOCUMENT DESCRIBES THE USER INTERFACE TO SHARED LIBRARIES.

1 INSTALLING SHARED LIBRARIES

THE SHARED LIBRARIES OCCUPY SEGMENT 2014 AND MUST BE INSTALLED EACH TIME THE SYSTEM IS COLD STARTED. THE RUNFILES ARE RESIDENT IN UFD SYSTEM AND ARE MOST EASILY INSTALLED AT STARTUP TIME BY INCLUDING THE COMMAND FILE C_SHLB IN THE STARTUP COMMAND FILE. THE INVOCATION SHOULD BE:

```
CO C_SHLB SYSTEM
```

RUNNING THIS COMMAND FILE INSTALLS 8 MEMORY IMAGE FILES IN SEGMENT 2014 AND RUNS THE PROGRAMS REQUIRED TO INFORM THE OPERATING SYSTEM THAT SHARED LIBRARIES ARE ACTIVATED. ONCE THE LIBRARIES ARE INSTALLED, USERS WITH PROGRAMS LOADED USING THE SPECIAL SHARED LIBRARY OBJECT FILES MAY RUN V-MODE PROGAMS ACCESSING THESE SHARED LIBRARIES. IF THE SHARED LIBRARIES ARE NOT INSTALLED PROGRAMS EXPECTING THE SHARED LIBRARIES TO BE RESIDENT WILL GET A NOT FOUND MESSAGE FROM THE OPERATING SYSTEM WHENEVER AN ATTEMPT IS MADE TO ACCESS A SHARED LIBRARY ROUTINE.

2 MAKING USE OF THE SHARED LIBRARIES

USERS WISHING TO MAKE USE OF THE SHARED LIBRARIES MUST RELOAD THEIR PROGRAMS USING THE SPECIAL SHARED LIBRARY OBJECT FILES IN UFD LIB. FORMS USERS MUST ALSO MAKE A SOURCE CHANGE TO THEIR MAIN PROGRAM. EXCEPT FOR FORMS THERE ARE NO OTHER SPECIAL REQUIREMENTS. FOR PARTICULARS RELATING TO THE USE OF THE SHARED LIBRARIES, SEE THE REV. 15 DOCUMENTATION FOR EACH LIBRARY PACKAGE.

IF ONE OF THE SHARED LIBRARIES IS TO BE USED, ALL APPROPRIATE SHARED LIBRARIES MUST ALSO BE USED. THUS, IF THE USER WISHES TO USE THE SHARED FORTRAN LIBRARY AND ALSO REQUIRES KI/DA OR COBOL, THE SHARED KI/DA AND COBOL LIBRARIES MUST ALSO BE USED.

ONCE THE NEW V-MODE RUN FILE HAS BEEN CREATED, AND THE SHARED LIBRARIES INSTALLED - SEE ABOVE - THE USER'S PROGRAMS MAY BE RUN EXACTLY AS THEY HAVE BEEN RUN IN THE PAST.

IT IS ANTICIPATED THAT THE SHARED LIBRARIES WILL NOT BE DELIVERED AS THE DEFAULT LIBRARIES FOR REV. 15. HOWEVER THE SHARED LIBRARIES FILES ARE IN UFD LIB AS SFTNLB, SKDALB AND - FOR THOSE USERS PURCHASING COBOL AND/OR FORMS - SCOBLB AND SFORMS. IF THE SHARED LIBRARIES ARE TO BE USED SYSTEM WIDE, THE MOST APPROPRIATE ACTION IS TO RENAME THESE MODULES AS FOLLOWS:

```
SFTNLB TO PFTNLB
SKDALB TO VKDALB
SCOBLB TO VCOBLB - IF AVAILABLE ON THE SYSTEM
SFORMS TO VFORMS - IF AVAILABLE ON THE SYSTEM
```

IF THE SHARED LIBRARIES ARE NOT TO BE USED SYSTEM WIDE, THEN THOSE USERS PLANNING TO USE THEM MUST MODIFY THEIR COMMAND FILES TO USE THE SPECIAL LIBRARY FILES. IN PARTICULAR THE LOADER COMMAND:

LI SFTNLB

MUST BE INCLUDED RIGHT BEFORE THE USUAL 'LI' COMMAND.

3 ADVANTAGES AND DISADVANTAGES OF SHARED LIBRARIES

AS IMPLEMENTED AT REV. 15, EACH USER OF SHARED LIBRARY ROUTINES WILL AUTOMATICALLY MAKE USE OF PRIVATE SEGMENT 6001 IN ADDITION TO THE SEGMENTS OTHERWISE REQUIRED BY HIS PROGRAMS. SEGMENT 6001 IS USED FOR THE IMPURE PORTION OF THE SHARED LIBRARIES AND REPRESENTS A REDUCTION IN THE SIZE OF THE USER'S LOAD FILE BUT NOT IN THE SIZE OF THE SINGLE USER WORKING SET AT RUN TIME. THIS ADDITIONAL SEGMENT MAY BE COMPENSATED FOR BY A CORRESPONDING REDUCTION IN THE NUMBER OF SEGMENTS IN THE RUN FILE. USERS MAY BE ABLE TO REDUCE THE NUMBER OF SEGMENTS USED BY THEIR RUNFILES BY USING THE MI OPTION OF SEG'S LOADER. FOR DETAILS OF THIS FEATURE SEE THE REV. 15 DOCUMENTATION.

SEVERAL BENEFITS RESULT FROM USING THE SHARED LIBRARIES. IN THE FIRST PLACE, USER RUN FILES WILL BE SMALLER. THIS WILL NORMALLY REDUCE THE TIME REQUIRED TO RESTORE THE SEG RUNFILE AND IMPLIES THAT USER INTERACTION WITH THE PROGRAM WILL BEGIN SOONER.

FOR USERS WITH MANY LARGE V-MODE PROGRAMS MAKING EXTENSIVE USE OF THE SHARED LIBRARY ROUTINES, THE MOST IMPORTANT EFFECT WILL BE TO REDUCE THE LOAD PUT ON THE SYSTEM IN REGARD TO PRIVATE SEGMENTS AND PRIVATE MEMORY IMAGE SIZES. PROPERLY USED, THEY MAY REDUCE PAGING. USE OF THE SHARED LIBRARIES IS INDICATED WHEN THE MAJORITY OF USERS ON THE SYSTEM ARE NORMALLY USING THE LIBRARY ROUTINES WHICH ARE SHARED. SMALL SYSTEMS WITH FEW USERS AND ONLY ONE KI/DA USER OR ONE COBOL USER OR WHERE THE FORTRAN FORMATTED I/O ROUTINES ARE SELDOM USED, MAY SEE NO BENEFIT FROM SHARED LIBRARIES.

THE FINAL BENEFIT OF SHARED LIBRARIES IS THAT USERS INSTALLING A NEW REV. OF THE LIBRARY DO NOT NEED TO RELOAD THEIR PROGRAMS. INSTALLATION OF A REBUILT SHARED LIBRARY IS ALL THAT IS REQUIRED TO MAKE THE MODIFIED LIBRARY AVAILABLE TO ALL USERS OF THE SHARED LIBRARY.

4 REBUILDING AND REINSTALLING SHARED LIBRARIESS

EACH OF THE SHARED LIBRARIES IS REPRESENTED BY A SEPARATE SET OF RUNFILES AND A SEPARATE INSTALL PROGRAM. IN THE EVENT THAT ONE OF THE LIBRARIES MUST BE REPLACED IT IS ONLY NECESSARY TO REBUILD THAT LIBRARY. THE COMMAND FILES REQUIRED TO REBUILD EACH LIBRARY PACKAGE ARE DESCRIBED IN THE REV. 15 DOCUMENTATION FOR THAT PACKAGE. FOR EXAMPLE, IF IT IS NECESSARY TO REBUILD KI/DA, SEE THE KI/DA DOCUMENTATION. THESE COMMAND FILES PUT ALL THE NECESSARY FILES INTO UFD SYSTEM SO THAT INSTALLATION IS EASILY ACCOMPLISHED BY RUNNING THE COMMAND FILE C_SHLB IN THAT UFD.

IF A LIBRARY MUST BE REPLACED IT SHOULD NOT BE REPLACED WHILE USERS ARE

USING IT. AS PROGRAMS USING THE SHARED LIBRARIES EXECUTE, LINKS ARE MADE TO THE APPROPRIATE SHARED LIBRARY ROUTINES IN SUCH A WAY THAT ALTERING THE MEMORY IMAGE IN USE BY THE PROGRAM CAN CAUSE RANDOM AND UNPREDICTABLE BEHAVIOR. CHANGING A SHARED LIBRARY (REPLACING ITS MEMORY IMAGE IN SEGMENT 2014) HAS THE EFFECT OF MAKING JUST SUCH AN ALTERATION TO THE USER'S MEMORY IMAGE. THE BEST PLAN IS TO INSTALL NEW SHARED LIBRARIES ONLY WHEN BRINGING UP THE SYSTEM WITH A COLD START.

NOTE, HOWEVER, THAT IT IS PERFECTLY SAFE TO REPLACE THE MEMORY IMAGE FILES IN UFD SYSTEM AT ANY TIME AS THESE ARE ONLY LOADED INTO MEMORY WHEN THE EXPLICIT COMMANDS TO SHARE THEM ARE GIVEN.

DATE: NOVEMBER 30, 1977

SUBJECT: HASP & SPOOLER -- REV 15

THE HASP WORKSTATION (HWS) CONSOLE PROGRAM (P300 AND P400) HAS BEEN REWRITTEN IN A FEW CRITICAL AREAS AND THE RECV FUNCTION IS NOW 10 TIMES FASTER ON A P400 THAN AT REV 14. IT IS NOW SO FAST THAT IT IS UNLIKELY THAT P400 USERS WILL EVER SEE MORE THAN ONE TEMPORARY RECEIVE FILE IN THE HASP DIRECTORY.

IN ADDITION TO THE SPEED-UP, A FUNCTIONAL CHANGE WAS MADE. THE HASP CARRIAGE CONTROL IS NOW PASSED ONTO THE SPOOLER. THERE IS A NEW VERSION OF THE SPOOLER, THEREFORE, THAT HAS THE UPDATES TO UNDERSTAND THE HASP CONVENTIONS AS WELL AS TO PERMIT 132 DATA CHARACTERS INSTEAD OF 131. FOR SIMPLIFIED SOFTWARE MAINTENANCE, THE VARIOUS SPOOLERS (*SPPR0, *SPPR1, *SPCEN, *SPPLT) HAVE BEEN COMBINED INTO A SINGLE PROGRAM (*SPPR). THE CORRESPONDENCE IS AS FOLLOWS:

		OLD	NEW
MPC	PR0	R *SPPR0	R *SPPR
MPC	PR1	R *SPPR1	R *SPPR 3/1
CENTRONIX	CENPR	R *SPCEN	R *SPPR 2/1
VERSATEC/GOULD	PLOT	R *SPPLT	R *SPPR 2/2

THE VITAL FORM CONTROL PAPER TAPE IMAGES CAN BE CREATED BY THE PROGRAM *FCPRP AS BEFORE, BUT THE PROGRAM IS NOW IN NEWSPL. ALSO, THE CREATED IMAGES NOW BELONG IN THE SPOOLQ DIRECTORY, RATHER THAN IN THE HASP DIRECTORY.

DATE: JUNE 9, 1978

SUBJECT: TERM COMMAND

THE TERM COMMAND IS A USEFUL TOOL TO CONTROL THE DUPLEX OF A TERMINAL AS WELL AS SETTING THE KILL AND ERASE CHARACTERS AND ENABLING OR DISABLING THE BREAK KEY OR ENABLING THE X-ON/X-OFF OPTION. THE COMMAND LINE FOR THE REV. 15 TERM COMMAND WILL LOOK FOR OPTIONS TO BE PRECEDED BY A MINUS SIGN (-), THE OLD WAY (OPTIONS WITHOUT THE MINUS SIGN) WILL STILL WORK FOR COMPATIBILITY. THE REST OF THIS DOCUMENT WILL BE DEDICATED TO EXPLAINING THE DIFFERENT COMMAND LINE FORMATS FOR THE TERM COMMAND.

A.) TERM

TYPING TERM WITHOUT ANY OPTIONS WILL HAVE THE PROGRAM PRINT A GENERAL LIST OF POSSIBLE COMMAND LINE FORMATS.

B.) TERM -ERASE (CHAR)

THIS WILL SET THE ERASE CHARACTER FROM ITS CURRENT VALUE TO THAT OF CHAR WHICH IS SPECIFIED IN THE COMMAND LINE.

C. TERM -KILL (CHAR)

THIS WILL SET THE KILL CHARACTER FROM ITS CURRENT VALUE TO THAT OF CHAR WHICH IS SPECIFIED IN THE COMMAND LINE.

NOTE: CHAR MUST BE A SINGLE CHARACTER AND THE PARENTHESIS ARE NOT TO BE SPECFIED.

D.) TERM -BREAK ON

THIS ENABLES THE BREAK OR [CONTRL-P] KEY.

E.) TERM -BREAK OFF

THIS DISABLES THE BREAK OR [CONTRL-P] KEY.

F.) TERM -HALF -[XOFF OR NOXOFF] -[LF OR NOLF]

THE PARAMETERS IN THE BRACKETS ARE OPTIONAL. THE HALF DUPLEX KEY WILL NOT ECHO BACK INPUT FROM THE TERMINAL. THE NOLF WILL NOT ECHO A LINE FEED AFTER A CARRIAGE RETURN. A LF WILL ECHO A LINE FEED AFTER A CARRIAGE RETURN. AN XOFF WILL ENABLE THE X-OFF/X-ON FEATURE, A NOXOFF WILL DISABLE THE X-OFF/X-ON FEATURE. IF THE [XOFF OR NOXOFF] OPTION IS OMITTED THE TERM COMMAND WILL DEFAULT TO THE STATE OF THE X-OFF/X-ON THAT EXISTED BEFORE THE TERM COMMAND WAS INVOKED.

G.) TERM -FULL -[XOFF OR NOXOFF]

THE FULL DUPLEX KEY WILL ECHO BACK INPUT FROM THE KEYBOARD TO THE TERMINAL SCREEN. THE [XOFF OR NOXOFF] FEATURE WILL WORK AS DESCRIBED IN THE SECTION F.

H.) TERM -[XOFF OR NOXOFF]

THIS FORM WILL SET THE TERMINAL TO FULL DUPLEX (DEFAULT VALUE) AND ENABLE OR DISABLE THE X-OFF/X-ON ACCORDING TO THE SPECIFIED COMMAND IN THE COMMAND LINE.

I.) TERM -DISPLAY

THIS FORMAT WILL PRINT OUT THE TERMINAL'S KILL AND ERASE CHACACTERS AS WELL AS WHETHER THE TERMINAL IS IN FULL OR HALF DUPLEX OR IF THE X-ON/X-OFF FEATURE IS ENABLED, OR IF AN X-OFF (CONTRL-S) HAS BEEN RECIEVED.

DATE: MARCH 14, 1978

SUBJECT: PSD AND VPSD FOR REV. 15

THIS DOCUMENT DESCRIBES THE CHANGES TO VPSD (AND VPSD16) FOR REV 15. A COMPLETE DESCRIPTION OF REV. 14 VPSD IS CONTAINED IN SECTION 16 OF THE PMA PROGRAMMERS GUIDE, PDR3059.

R-MODE PSD (PSD, HPSD, AND PSD20) IS UNCHANGED FOR REV. 15. A FEATURE OF THE NEW LOAD, HOWEVER, IS OF INTEREST TO PSD USERS. THE COMMAND "MAP <FILENAME> 10" WILL PRODUCE A SYMBOL FILE SUITABLE FOR READING INTO PSD, ELIMINATING THE NEED FOR THE CNVTMA PROGRAM.

ENHANCEMENTS FOR REV. 15 ARE:

1. IT IS NOW POSSIBLE TO ENTER SHORT-FORM V-MODE INSTRUCTIONS USING THE # OPCODE SUFFIX. OPCODES NOT FOLLOWED BY EITHER # OR % WILL BE MADE LONG OR SHORT THE SAME WAY AS PMA WOULD DO IT.

2. THE FIELD ADDRESS INSTRUCTIONS ARE NOW SUPPORTED BY PSD WITH THE SAME SYNTAX USED IN PMA.

3. IN ALL APPROPRIATE PLACES THE SUFFIXES "+1C" AND "+NB" MAY BE USED TO INDICATE CHARACTER AND BIT OFFSETS.

4. STACK-RELATIVE INSTRUCTIONS MAY NOW BE USED IN R-MODE TYPEIN AND WILL BE PRODUCED ON TYPEOUT (IE LDA @+10).

5. THE AP PSEUDO-OP MAY NOW BE ENTERED, AND WILL BE DISPLAYED AS APPROPRIATE AFTER PCL INSTRUCTIONS. TO INVOKE AP TYPEOUT MODE WHEN VPSD HAS NOT SELECTED IT AUTOMATICALLY, USE THE :P SPECIFIER. TO TERMINATE AP MODE, USE :S OR ANY OTHER SPECIFIER.

6. LONG (32 BIT) OCTAL INTEGERS ARE NOW SUPPORTED, USE THE :L SPECIFIER.

7. TWO NEW COMMANDS HAVE BEEN ADDED TO LOOK AT THE FIELD ADDRESS AND LENGTH REGISTERS, "FA" AND "FL". EACH COMMAND TAKES A SINGLE ARGUMENT, WHICH IS THE REGISTER NUMBER TO LOOK AT. THE COMMANDS FUNCTION IN A SIMILAR MANNER TO THE ACCESS ("A") COMMAND. NEW VALUES MAY BE ENTERED TO REPLACE OLD ONES. CARRIGE RETURN ADVANCES TO THE "NEXT" REGISTER, AND "^" GOES BACK TO THE "PREVIOUS" ONE. A "(" WILL SWITCH TO ACCESS MODE AND DISPLAY THE LOCATION REFERANCED BY A FIELD ADDRESS REGISTER IN ASCII. A ")" WILL RETURN TO "FA" MODE.

8. IF VPSD IS ASSEMBLED WITH B-REGISTER BIT 10 SET, SEGMENT NUMBERS MAY BE SPECIFIED IN COMMAND LINES AS "<SEGMENT NUMBER>/". THE USE OF "/" IS THUS EXCLUDED AS AN "ESCAPE" CHARACTER, AND WILL GIVE AN

ERROR IF USED IMPROPERLY. QUESTION-MARK MAY STILL BE USED FOR THIS PURPOSE. BECAUSE THIS IS INCOMPATIBLE WITH PREVIOUS RELEASES OF VPSD, REV. 15 WILL BE RELEASED WITH THIS FEATURE DISABLED. NOTE THAT THE USE OF "/" IS IDENTICAL TO USING THE "SN" COMMAND: ITS EFFECT IS NOT LIMITED TO THE CURRENT COMMAND, AND COMMANDS WHICH COULD NOT PREVIOUSLY HANDLE CROSS-SEGMENT OPERATIONS (SUCH AS COPY) STILL CANNOT.

9. BASE REGISTERS MAY NOW BE USED IN COMMAND LINES, SUCH AS "A SB%+10". WARNINGS WHICH APPLY TO THE USE OF "/" AS A SEGMENT NUMBER SPECIFIER ALSO APPLY HERE.

10. THE CHARACTER "!" WILL CLOSE A LOCATION IN ACCESS MODE, SETTING IT TO A NEW VALUE IF ONE WAS SUPPLIED, AND RETURN TO COMMAND LEVEL. THIS FEATURE WAS PRESENT IN REV. 14 PSD BUT WAS NOT DOCUMENTED.

11. IF B-REGISTER BIT 11 (OCTAL 40) IS SET DURING ASSEMBLY, CODE IS INCLUDED TO SET UP THE SOC OR OPTION-A BOARDS ON A STAND-ALONE SYSTEM. DEFAULT IS 30CPS AND 6 FILLS AFTER THE LINEFEED CHARACTER. TO SET NEW TERMINAL CHARACTERISTICS, PATCH VPSD AS FOLLOWS:

```
<STARTING ADDRESS>+4    <OPTION-A CONTROL WORD>
                  +5    <SOC CONTROL WORD 1>
                  +6    <SOC CONTROL WORD 2>
                           +<NUMBER OF FILLS AFTER LINEFEED>
```

12. VPSD NOW USES SAVED REGSITER VALUES PROPERLY IN COMPUTING EFFECTIVE ADDRESSES. TYPING "=" AFTER AN INSTRUCTION WHICH REFERANCES THE LIVE REGISTERS WILL DISPLAY THE VALUE OF THE REGISTER.

13. VPSD FOR REV. 15, LIKE VPSD FOR REV. 13 AND UNLIKE VPSD FOR REV. 14, ONLY USES LOCATIONS 40-57 IN SEGMENT 4000 WHEN PROCESSING RUN ("R") COMMANDS.

VPSD FOR REV. 15 USES '11646 LOCATIONS, COMPARED WITH '7644 FOR REV. 14 VPSD.