

AD-A233 962

(1)

PROGRAMMER'S MANUAL
FOR THE
WARTIME PERSONNEL ASSESSMENT MODEL
(WARPAM)
(VERSION 1.0)

31 January 1991

DTIC
APR 01 1991

Prepared for:

TRADOC ANALYSIS COMMAND
Building 401B
Fort Benjamin Harrison, Indiana 46216-5000

Contract Number MDA903-88-D-1000
Task Order 0037

Prepared by:

James A. Wojcik
John A. Tenshaw
Beth A. White
Tanya L. Reaves

Science Applications International Corporation
1710 Goodridge Drive
McLean, Virginia 22102

Science Applications International Corporation
1710 Goodridge Drive
McLean, Virginia 22102

91 3 29 034

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188
Exp. Date: Jun 30, 1988

1a. REPORT SECURITY CLASSIFICATION Unclassified			1b. RESTRICTIVE MARKINGS None		
2a. SECURITY CLASSIFICATION AUTHORITY N/A			3. DISTRIBUTION / AVAILABILITY OF REPORT Unlimited		
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE N/A					
4. PERFORMING ORGANIZATION REPORT NUMBER(S) SAIC-90/1459			5. MONITORING ORGANIZATION REPORT NUMBER(S) To be assigned		
6a. NAME OF PERFORMING ORGANIZATION Science Applications International Corporation		6b. OFFICE SYMBOL (if applicable) N/A	7a. NAME OF MONITORING ORGANIZATION TRADOC Analysis Command-FT Benjamin Harrison		
6c. ADDRESS (City, State, and ZIP Code) 1710 Goodridge Drive, T1-7-2 McLean, VA 22102			7b. ADDRESS (City, State, and ZIP Code) ATTN: ATRC-B (BLDG 401-B) Fort Benjamin Harrison, IN 46216-5000		
8a. NAME OF FUNDING / SPONSORING ORGANIZATION TRAC-FBHN		8b. OFFICE SYMBOL (if applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER MDA903-88-D-1000		
8c. ADDRESS (City, State, and ZIP Code) ATTN: ATRC-B (BLDG 401B) Fort Benjamin Harrison, IN 46216-5000			10. SOURCE OF FUNDING NUMBERS		
			PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO. 0037
11. TITLE (Include Security Classification) Wartime Personnel Assessment Model (WARPAM)					
12. PERSONAL AUTHOR(S) James A. Wojcik, John A. Tenshaw, Beth A. White, Tanya L. Reaves					
13a. TYPE OF REPORT Programmer's Manual		13b. TIME COVERED FROM 6/30/89 TO 8/31/90		14. DATE OF REPORT (Year, Month, Day) 1991 January 31	
15. PAGE COUNT 241					
16. SUPPLEMENTARY NOTATION					
17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number) Personnel Replacements, Return-to-duty Personnel, CONUS Replacement Center, Replace t Battalion, Reclassification.		
FIELD	GROUP	SUB-GROUP			
19. ABSTRACT (Continue on reverse if necessary and identify by block number) The Wartime Personnel Assessment Model (WARPAM) is a skeletal model, designed for operation on a Sun workstation, links the outputs from several Army models and then through a series of simulations produces a comprehensive depiction of the Army wartime personnel replacement system. Specifically, WARPAM provides the capability to: forecast the personnel system's potential to satisfy projected requirements, link doctrinal concepts with output from current "stand alone" Army models, simulate the reclassification of return-to-duty personnel, generate logistical needs to support the personnel system and perform "What if" analysis regarding force structure or doctrinal changes. These capabilities enable TRAC-FBHN to provide quantitative input to the Army's macro-level decision-making process in regards to analyzing and evaluating force structure and personnel replacement doctrine and also satisfy the Army's requirements for micro-level modeling of replacement center activities.					
20. DISTRIBUTION / AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS			21. ABSTRACT SECURITY CLASSIFICATION Unclassified		
22a. NAME OF RESPONSIBLE INDIVIDUAL MAJ James Thomas			22b. TELEPHONE (Include Area Code) (317)543-6883		22c. OFFICE SYMBOL

<u>SECTION</u>	<u>PAGE</u>
1 GENERAL	
1.1 PURPOSE OF THE PROGRAMMER'S MANUAL.....	1
1.2 PRIMARY PROJECT REFERENCES.....	1
1.3 TERMS AND ABBREVIATIONS.....	2
2 SYSTEM SUMMARY	
2.1 SYSTEM APPLICATIONS.....	3
2.2 SECURITY.....	3
2.3 SYSTEM DESCRIPTION.....	4
2.4 SYSTEM OPERATION.....	4
3 ENVIRONMENT	
3.1 EQUIPMENT ENVIRONMENT.....	5
3.2 SUPPORT SOFTWARE.....	5
3.3 DATA BASES.....	5
3.3.1 OPERATIONAL ORGANIZATION.....	5
3.3.2 SSUB-DIRECTORY ORGANIZATION.....	6
3.4 INPUT FILE DATA BASES.....	7
3.4.1 AUTOREP.....	7
3.4.2 MOBMAN.....	7
3.4.3 CASUALTY STRATIFICATION MODEL (CSM II).....	7
3.4.4 MOBAPRINT.....	7
4 PREPROCESSOR	
4.1 GENERAL.....	8
4.2 INITIATION.....	8
4.3 AUTOREP MODULE.....	8
4.3.1 AUTOREP DBASE CONVERSION PROGRAMS.....	10
4.3.2 AUTOREP FORTRAN PROGRAMS.....	11
4.4 MOBMAN MODULE.....	28
4.4.1 GENERAL.....	28
4.4.2 MOBMAN FORTRAN PROGRAMS.....	29
4.5 CSM II MODULE....	51
4.5.1 GENERAL.....	51
4.5.2 CSM II FORTRAN PROGRAMS.....	52
4.6 MOBTNGBS MODULE.....	69
4.6.1 GENERAL.....	69
4.6.2 MOBTNGBS FORTRAN FILES.....	70
4.7 REQUIREMENTS/ASSETS GENERATOR MODULE.....	80
4.7.1 GENERAL.....	80
4.7.2 REQAST GENERATOR FORTRAN PROGRAMS.....	81

5 RECLASSIFICATION MODEL

5.1	GENERAL.....	93
5.2	INITIATION.....	93
5.3	INPUT FILES.....	93
5.4	INPUT VARIABLES.....	93
5.5	PROCESSING.....	94
5.6	OUTPUT REPORTS.....	94
5.7	RECLAS MODEL FORTRAN PROGRAMS.....	96

6 CONUS REPLACEMENT CENTER /OCONUS REPLACEMENT CO (CRC) MODEL

6.1	GENERAL.....	139
6.2	INITIATION.....	139
6.3	INPUT FILES.....	139
6.4	INPUT VARIABLES.....	139
6.5	PROCESSING.....	140
6.6	OUTPUT REPORTS.....	141
6.7	CRC/RPLBN MODEL--GENERAL FORTRAN PROGRAMS.....	143
6.8	CRC MODEL SPECIFIC PROGRAMS.....	170
6.8.1	FORTRAN PROGRAMS SUPPORTING SLAM II CRC PROGRAM - PROGRAM MAIN.....	170
6.8.2	CRC MODEL PROCESSING FLOW.....	172
6.8.3	CRC MODEL SLAM II PROGRAMS.....	181
6.9	RPL MODEL SPECIFIC PROGRAMS.....	191
6.9.1	FORTRAN PROGRAMS SUPPORTING SLAM II RPL PROGRAM - PROGRAM MAIN.....	191
6.9.2	RPL MODEL PROCESSING FLOW.....	200
6.9.3	RPL MODEL SLAM II PROGRAMS.....	201

7 TRANSPORTATION MODEL

7.1	GENERAL.....	205
7.2	INITIATION.....	205
7.3	INPUT FILES.....	205
7.4	INPUT VARIABLES.....	205
7.5	PROCESSING.....	206
7.6	OUTPUT REPORTS.....	206
7.7	TRANSPORTATION MODEL FORTRAN PROGRAMS.....	207

8 LOOK-UP TABLES

8.1	GENERAL.....	214
8.2	WARPAM BRANCH TABLE CODES.....	214
8.3	BRANCH AGGREGATION TABLE.....	215
8.4	WARPAM BRANCH PRIORITY TABLE.....	220
8.5	THEATER/REPLACEMENT TYPE TABLE.....	222
8.6	AUTOREP TIME PERIOD CONVERSION TABLE.....	223
8.7	OFFICER RECLASSIFICATION PERCENTAGE TABLE.....	224

8.8	ENLISTED RECLASSIFICATION PERCENTAGE TABLE.....	225
8.9	RECLASSIFICATION DELAY TABLE.....	226

9 REPORT GENERATOR

9.1	GENERAL.....	227
9.2	REQUIREMENTS/ASSETS REPORT.....	227
	9.2.1 CONVERSION PROGRAM.....	227
	9.2.2 OUTPUT FORMAT.....	228
9.3	RECLASSIFICATION FILE CONVERSION PROGRAMS.....	228
	9.3.1 CONVERSION PROGRAMS.....	228
	9.3.2 OUTPUT REPORT.....	229
9.4	CRC MODEL REPORT.....	229
	9.4.1 CONVERSION PROGRAMS.....	229
	9.4.2 OUTPUT REPORT.....	234
9.5	REPLACEMENT BN MODEL REPORT.....	234
	9.5.1 CONVERSION PROGRAMS.....	234
	9.5.2 OUTPUT REPORT.....	239
9.6	TRANSPORTATION MODEL REPORT.....	239
	9.6.1 CONVERSION PROGRAM.....	239
	9.6.2 OUTPUT REPORT.....	239

ANNEX

A	TERMS AND ABBREVIATIONS.....	A-1
B	SAMPLE FILE/OUTPUT FORMATS.....	B-1
C	OUTPUT REPORT FORMATS.....	C-1

Approved by	
NTIS Grant	
DHD Title	
Unpublished	
Justification	
By	
Distribution	
Available to	
Dist	Available to
A-1	

FIGURES

FIGURE 1:	MODULAR ARCHITECTURE.....	4
FIGURE 2:	OPERATIONAL ORGANIZATION.....	6
FIGURE 3:	AUTOREP FILE CONVERSION.....	9
FIGURE 4:	MOBMAN FILE CONVERSION.....	28
FIGURE 5:	CSM II FILE CONVERSION.....	51
FIGURE 6:	MOBTNGBS FILE CONVERSION.....	69
FIGURE 7:	REQUIREMENTS/ASSETS GENERATOR PROCESSING.....	80
FIGURE 8:	RECLASSIFICATION MODEL PROCESSING.....	95
FIGURE 9:	CRC FORTRAN PROCESSING.....	142
FIGURE 10:	CRC SLAM II PROCESSING	180
FIGURE 11:	RPL SLAM II PROCESSING.....	200

SECTION 1 GENERAL

1.1 PURPOSE OF THE PROGRAMMER'S MANUAL

The objective of the Programmer's Manual (PM) is to provide TRAC-FBHN programmers with the information necessary to effectively maintain WARPAM and, as required, effect minor program changes. When used in conjunction with the WARPAM Descriptive Documentation and User's Manual, the Programmer's Manual will allow TRAC-FBHN to maintain the system with internal personnel assets. The manual provides both overviews of the system architecture and the program code for the modules and models which form the WARPAM system.

1.2 PRIMARY PROJECT REFERENCES

The primary references upon which WARPAM is designed are listed below.

- o Wartime Personnel Assessment Model (WARPAM), Government Statement of Work, April 1989.
- o Personnel Service Support (PSS) in Army Models (Draft), TRADOC Analysis Command - Fort Benjamin Harrison, Major James Thomas, 1989.
- o Wartime Replacement System Study (WRSS), Soldier Support Center, Fort Benjamin Harrison, March 1987.
- o FM 12-6, Personnel Doctrine (Final Coordinating Draft), HQ, Department of the Army, August 1988.
- o TOE Number 12406L0, HHD, Personnel Replacement Battalion, HQ, Department of the Army, October 1987.
- o TOE Number 12407L0, Replacement Company, HQ, Department of the Army, October 1987.
- o FM 12-6, Personnel Doctrine (Final Approved Draft), HQ, Department of the Army, June 1989.
- o ARTEP Number 12-406-01-MTP, Personnel Replacement Battalion (GS/DS) (Coordinating Draft), HQ, Department of the Army, undated.
- o ARTEP Number 12-407-30-MTP, Replacement Company (GS/DS), HQ, Department of the Army, July 1989.
- o ARTEP Number 12-406-02-MTP, Personnel Replacement Battalion/Company (CRC) (Draft), HQ, Department of the Army, undated.

1.3 TERMS AND ABBREVIATIONS

Annex A contains a listing of terms, definitions, and acronyms unique to the development of WARPAM and subject to interpretation by the user of this document. This listing does not include data item names or codes which are discussed, as appropriate, within the body of the document.

SECTION 2 SYSTEM SUMMARY

2.1 SYSTEM APPLICATIONS

WARPAM is designed to resolve many of the US Army's modeling shortcomings associated with representing the flow of qualified replacements to the Airland Battlefield. This skeletal model, designed for operation on a Sun workstation, links the outputs from several Army models and then through a series of simulations produces a comprehensive depiction of the Army wartime personnel replacement system. Specifically, WARPAM provides the capability to: forecast the personnel system's potential to satisfy projected requirements, link doctrinal concepts with output from current "stand alone" Army models, simulate the reclassification of return-to-duty personnel, generate logistical and equipment requirements to support the personnel system and perform "What if" analysis in regards to force structure or doctrinal changes. These capabilities enable TRAC-FBHN to provide quantitative input to the Army's macro-level decision-making process in regards to analyzing and evaluating force structure and personnel replacement doctrine. Secondly, it satisfies the Army's requirements for micro-level modeling of replacement center activities enabling the analysis of contemplated changes prior to implementation. The following is a summary of WARPAM capabilities:

- o Comparison of requirements generated by other Army models.
- o Evaluation of the effects of proposed reclassification policy on replacement flow operations.
- o Micro-level modeling of replacement activity operations to include force structure evaluation and personnel policy.
- o What-If modeling of personnel policy and force structure with rapid response times.
- o Determination of transportation and support requirements.
- o Interface with other Army models to improve personnel modeling in the family of Army models.
- o Evaluation of the capability of active and reserve forces to support multiple theaters operations.

2.2 SECURITY

The data bases and tables used in developing the initial version of WARPAM are not classified. Other variations of these data bases (disaggregated to theater level) may be classified and care should be exercised when operating in the classified mode. Special precautions should be taken when the system is operated as designed in a LAN network configuration.

2.3 SYSTEM DESCRIPTION

The primary functions of WARPAM are the preparation of data from other Army models in a preprocessor phase, the reclassification of theater return-to-duty personnel, the time-phased processing of personnel through the replacement system, and the comparison of CONUS and OCONUS replacement activities. These function of each model or module is described in detail in later sections. The chart below shows the interrelationship of these modules and models.

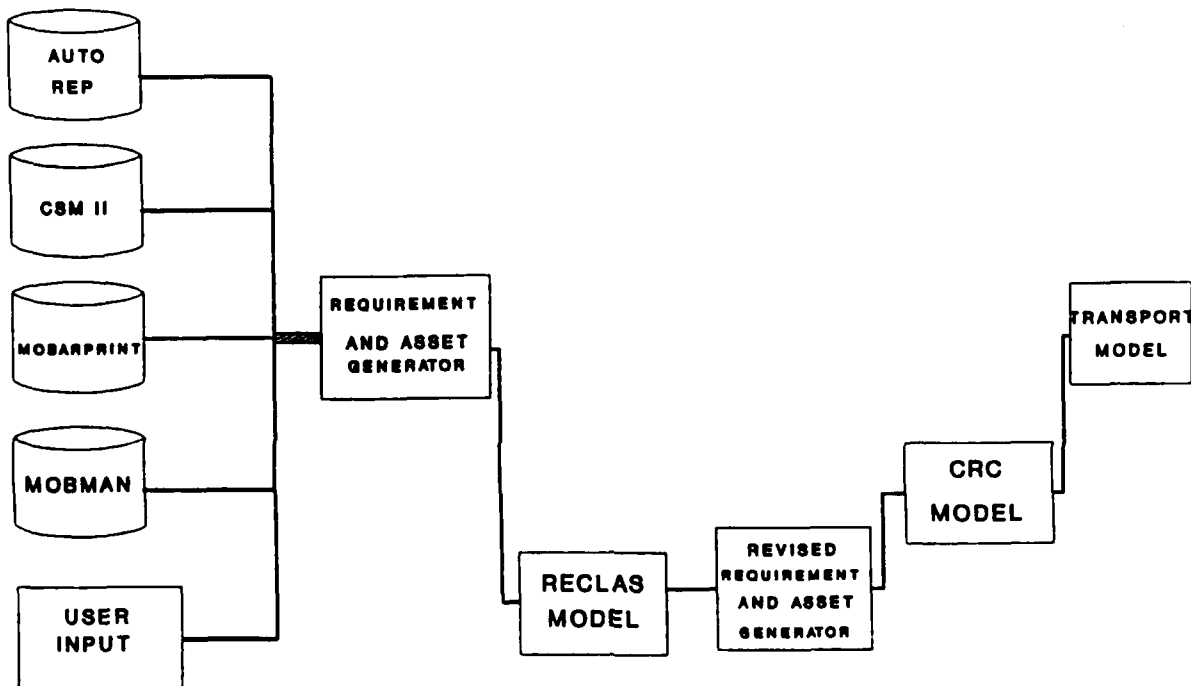


FIGURE 1: MODULAR ARCHITECTURE

2.4 PROGRAM DESCRIPTIONS

The individual programs and sub-routines are discussed in detail in the following sections.

SECTION 3 ENVIRONMENT

3.1 EQUIPMENT ENVIRONMENT

WARPAM is designed to operate on the TRAC-FBHN SUN 4/110-FCE-8 workstation with the following major components:

- o 16" color monitor
- o 32 MB memory
- o 327 MB hard disk
- o 60 MB 1/4" tape cartridge drive
- o Ethernet link to 5 1/4" diskette drive

3.2 SUPPORT SOFTWARE

All programs are heavily commented to afford ease of programming and maintenance.

WARPAM utilizes the following software:

- o SUN system "C" programming language: Executive Program
- o FORTRAN 77: All program routines except those written in SLAM II
- o SLAM II: CRC/RPL BN Model to replicate the internal operation of a replacement unit

3.3 DATA BASES

3.3.1 OPERATIONAL ORGANIZATION

WARPAM is organized with five major sub-systems. These are the preprocessor, reclassification model, CRC/RPL BN model, Transportation model and Report Generator. All but the report generator, which is performed on an IBM compatible PC, are run on the Sun workstation. The specific function of the major systems and their sub-components, as appropriate, are described in detail in subsequent sections. The chart on the following page depicts the WARPAM operational organization. Although the system appears to the user to be organized in this configuration, the actual data base organization is described in section 3.3.2 below.

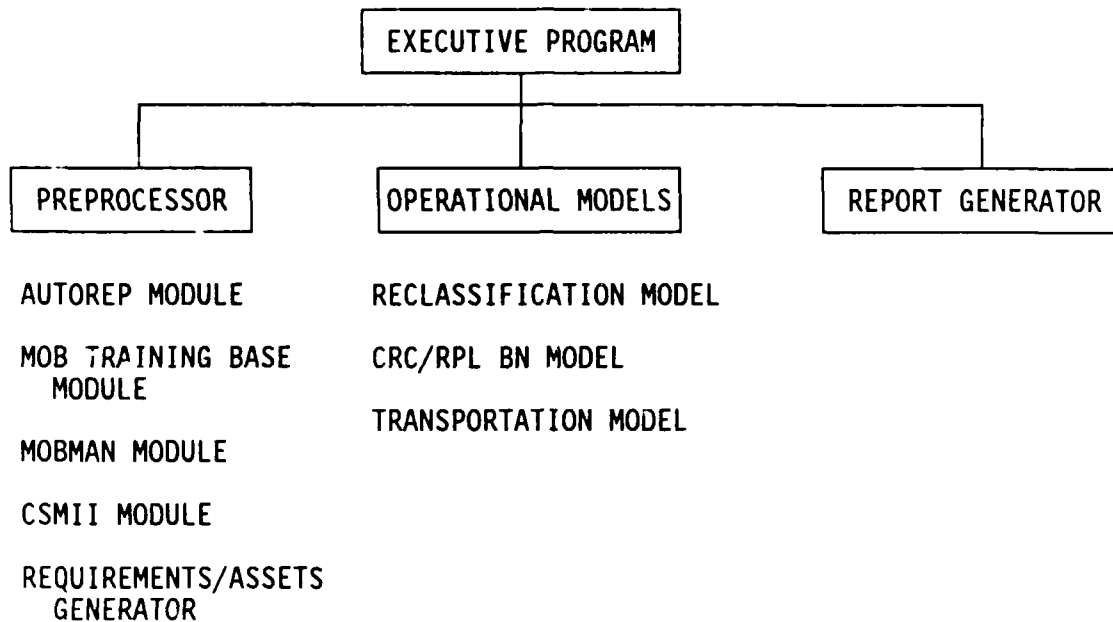


FIGURE 2: OPERATIONAL ORGANIZATION

3.3.2 SUB-DIRECTORY ORGANIZATION

WARPAM programs and files are organized on the Sun workstation in sub-directories based on the function of the program. These sub-directories are located in the WARPAM directory under the home directory on the TRAC-FBHN system. The sub-directories and their contents are:

- o FORTRAN All FORTRAN programs
- o SLAM All SLAM II programs
- o DBASE All DBASE programs if loaded on the Sun workstation.
 These may be found only on the PC linked by network to
 the workstation
- o IOFILES All data bases and look-up tables

3.4 INPUT FILE DATA BASES

3.4.1 AUTOREP

Source: USA PERSCOM "Shelf Requisition" files
Description: Individual theater requirements developed with CINC input at the ALO 2 level. Currently available for Europe and Korea.
Media: 5 1/4 floppy disk (10 low density disks)
Language: DBASE III Plus.
Format: Sample format at Annex B, page B-1

3.4.2 MOBMAN

Source: HQDA MOBMAN Model
Description: Multiple theater requirements consolidated to a single data base at the ALO 1 level
Media: 1/2" magnetic tape
Language: ASCII (must be requested from contractor)
Format: Sample format at Annex B, page B-2

3.4.3 CASUALTY STRATIFICATION MODEL II (CSM II)

Source: USA Soldier Support Center
Description: Individually developed casualty model with requirements at the theater or below level
Media: 5 1/4" floppy disk (one disk)
Language: ASCII developed from a DBASE III file
Format: Sample format at Annex B, page B-3

3.4.4 MOBAPRINT

Source: HQDA, ODCSPER
Description: Skill level one output from a constrained training base environment
Media: 5 1/4 floppy disk (one disk)
Language: ASCII
Format: Sample format at Annex B, page B-4

SECTION 4 PREPROCESSOR MODULES

4.1 GENERAL

The Preprocessor is designed to convert the output files of current military personnel mobilization models to a standard format and consolidate these into a single data base. To accomplish this, the preprocessor has five modules to convert the data, and a requirements/assets generator module to merge these converted files into a single data base. The files which WARPAM is currently configured to convert are described in the following sections. The input file and the converted file formats are at Annex B. This conversion process to a standard data base format includes the following steps:

- o Conversion to an ASCII format.
- o Aggregate occupational specialties into branch/grade groupings.
- o Prioritize branches.
- o Assign code numbers to each entry which represents the appropriate time period, branch priority and requirement or asset designator.

4.2 INITIATION

Each module of the preprocessor is initiated by user input from a Sun window which activates the FORTRAN program. This window is reached by using the WARPAM Executive Windows Program which allows the user to reach any module by simply using the workstation mouse to move the pointer over the appropriate window. THIS IS THE ONLY COMMAND REQUIRED TO RUN THE PREPROCESSOR PROGRAMS. Files produced from previous runs of the preprocessor should be stored in a different sub-directory or under a different file name prior to running the preprocessor modules. Any file of the same name in the IOFILE sub-directory on the Sun workstation will be overwritten by the new output file. After a conversion module is used to create a new file the requirements/assets generator program must also be run to bring this new file into the REQAST.TBL which is used by all the models in WARPAM. The individual files are NOT USED as separate entities by any program.

4.3 AUTOREP MODULE

This module converts the shelf requisition files created by US ARMY PERSCOM to standard WARPAM format. Multiple files may be received from PERSCOM for different theaters. These files can be combined to two, one for officers and one for enlisted. WARPAM is current configured to translate the files for Europe and Korea only. The new files from PERSCOM are received on 5 1/4" floppy disks and are loaded onto the Sun drive by way of the network and PC. Requirements created by this module are labeled as theater requirements AE1 for Europe and AK0 for Korea. As with all modules in the

preprocessor, AUTOREP is initiated by placing the workstation mouse arrow over the appropriate block. This module use two look-up tables, the Branch table and the Time Period tables to convert the MOS to branches and convert the time periods to standard WARPAM time periods. The total processing of the AUTOREP file encompasses first converting the DBASE III Plus file to an ASCII file using a DBASE III conversion program and then reformatting the data into the standard WARPAM format using a FORTRAN program. The DBASE conversion is accomplished on an IBM compatible PC, whereas the format conversion is accomplished on the Sun workstation. The program flow and interrelationship of the modules is shown in the figure 3, below.

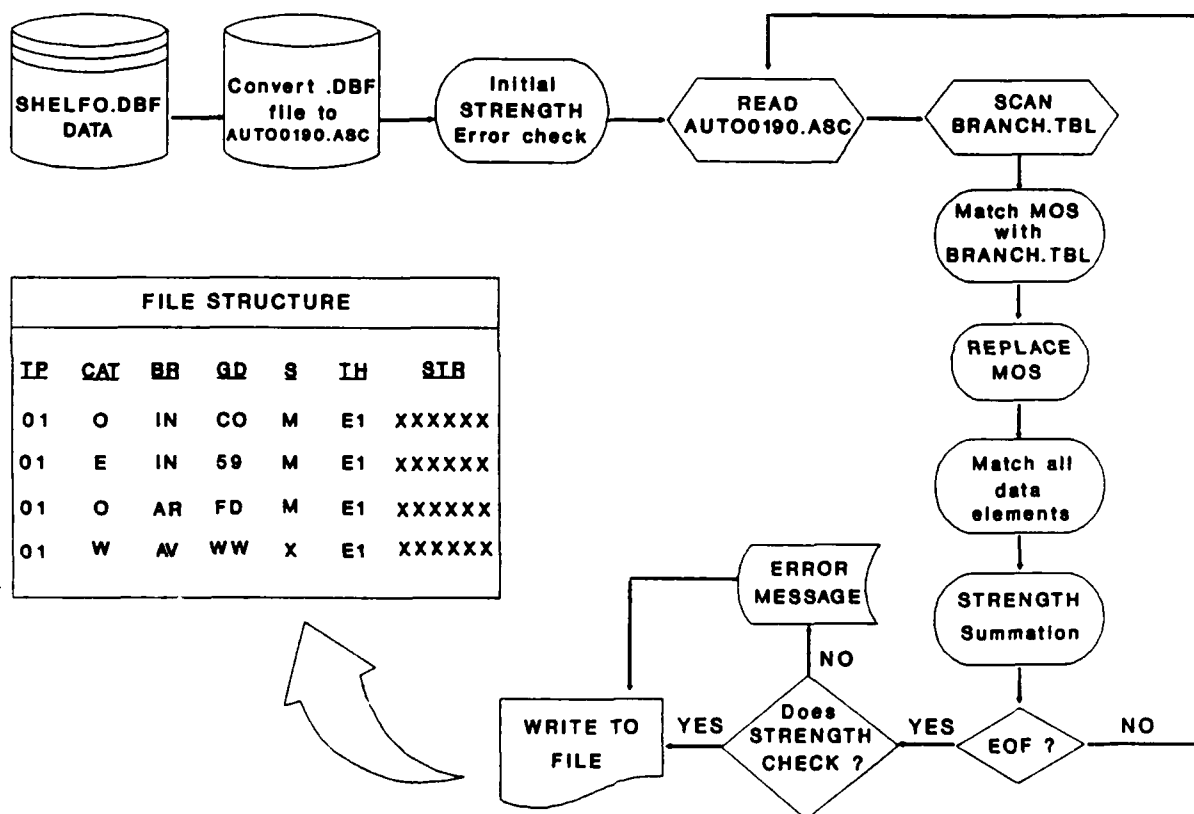


FIGURE 3: AUTOREP FILE CONVERSION

4.3.1 AUTOREP DBASE CONVERSION PROGRAMS

The DBASE programs can be used on either the single theater files and the files combined later or the files can be combined first. These programs may be modified by using DBASE III modified commands. The actual path for programs and data may be changed to accommodate the TRAC-FBHN PC configuration. The output files are designated AUTO0190.DAT and AUTE0190.DAT, for the officer and enlisted files respectively to designated file and program creation dates. Any change to these file names would require corresponding changes in the FORTRAN programs.

OFFICER CONVERSION PROGRAM

```
USE C:\SHELFO.DBF
COPY TO AUTO0190.DAT SDF FIELDS REQID, PERSCLASS, SPECCD, SKILID,
SKIL, GRADE, SEX, COMMAND, STRENGTH, RECORDID FOR RECORDID <>
'D2' .AND. RECORDID <> 'D3' .AND. RECORDID <> 'E2'^Z
```

ENLISTED CONVERSION PROGRAM

```
USE SHELFE.DBF
COPY TO AUTE0190.DAT SDF ALL FIELDS REQID, PERSCLASS, MOS, GRADE,
SEX, COMMAND, STRENGTH, RECORDID FOR RECORDID <> 'D2' .AND.
RECORDID <> 'D3' .AND. RECORDID <> 'E2'^Z
```


4.3.2 AUTOREP FORTRAN PROGRAMS

```
C*****
C
C Program Name:  AUTOOREP                      Date:  04-10-1990
C
C File Name:    AUTOREPO.FOR
C
C Programmer:   Beth White, SAIC, 749-8771
C
C Description:  Reads, extracts, and stores: [REQID] time period,
C              category identifier, {MOS} military occupation speciality
C              =[SPECCD] speciality + [SKILID] skill level identifier +
C              [SKIL] skill, [GRADE] grade, [SEX] gender, [COMMAND] command/
C              theater, and [STRENGTH] strength/casualties.
C              For each unique time period with the corresponding theater,
C              MOS, grade, and gender the strength/casualties are
C              summated.
C              An output file is created which represents: time period,
C              category identifier, branch, grade, sex, theater, and
C              strength.
C
C Input:        AUTO0190.DAT
C              AUTE0190.DAT
C
C Output:       AUTOREP.OUT
```

```
C*****
C Modifications: (STATUS: P - PROPOSED; R - REQUIRED ; C - COMPLETED)
C
C Number   Status   Date:           Description:           Initials
C -----
C    01      C      05/30/90  Modified directory changes.      BAW
```

```
C*****
```

PROGRAM AUTOOREP

C Global Variables

```
DIMENSION VSTOR(3000,6),TCHR(25),TPN(25),SLTP(8)
CHARACTER*1 CHRE(17),CATC,SKLC,GRDC,SEXC,SPEC1,SPEC2,SKLID
CHARACTER*2 TPC,TCHR,TPN,THRC,SPECC,GRDE,BRR,RCDID,VSTOR,SLTP
CHARACTER*3 MOSC,THR
CHARACTER*4 STRC
CHARACTER*17 LINE
```

LOGICAL THERE

```
INTEGER I,XX,NUM,STR,NRCD,NARAY,MAXARAY,STRNG(3000,1),IFLG,K,
&IFLAG,EXMAX
COMMON/CCTP/TCHR,TPN
```

COMMON/BRCH/CATC,MOSC,SPEC1,SPEC2,SPECC,SKLID,BRR
EQUIVALENCE (CHRE(1),LINE)

C Local Variables

NRCD = 0
NARAY = 0
MAXARAY = 0
IFLG = 0
EXMAX = 0

WRITE(6,90)
90 FORMAT(/////20X,'*****',
&/20X,'*****',//20X,
&' AUTOREP MODULE ',//20X,
&'*****',/20X,
&'*****',////////20X,
&'THE FOLLOWING FILES ARE NEEDED:',/30X,
&'AUTO0190.DAT',/30X,'AUTE0190.DAT',/30X,'TP.TBL',
&/30X,'BRANCH.TBL',//////////)

PAUSE

WRITE(6,91)
91 FORMAT(//////////)

C Checks to see if input files exists. If the input files
C do not exist, terminate the program.

INQUIRE(FILE='/home/warpam/iofiles/AUTO0190.DAT',EXIST=THE
IF (.NOT.THERE)THEN
WRITE(6,*)' ERROR - AUTO0190.DAT does not exist.'
GOTO 100
ENDIF

INQUIRE(FILE='/home/warpam/iofiles/AUTE0190.DAT',EXIST=THE
IF (.NOT.THERE)THEN
WRITE(6,*)' ERROR - AUTE0190.DAT does not exist.'
GOTO 100
ENDIF

INQUIRE(FILE='/home/warpam/iofiles/BRANCH.TBL',EXIST=THE
IF (.NOT.THERE)THEN
WRITE(6,*)' ERROR - BRANCH.TBL does not exist.'
GOTO 100
ENDIF

INQUIRE(FILE='/home/warpam/iofiles/TP.TBL',EXIST=THE
IF (.NOT.THERE)THEN
WRITE(6,*)' ERROR - TP.TBL does not exist.'
GOTO 100
ENDIF

C Checks to see if output file exists. If the output file
C exists, delete it.

```
INQUIRE(FILE='/home/warpam/iofiles/AUTOREP.OUT',EXIST=THEORE)  
IF (THEORE)THEN  
  OPEN(1,FILE='/home/warpam/iofiles/AUTOREP.OUT',STATUS='OLD')  
  CLOSE(1,STATUS='DELETE')  
ENDIF
```

C Begin AUTOOREP
C Calls subroutine TYMECODE which translates the time period code.
C Time period codes are placed into global variable arrays.

```
CALL TYMECODE
```

C Opening and reading input file: AUTO0190.DAT
C Extracts data into variables per line.

```
WRITE(6,*)' PROCESSING FILE: AUTO0190.DAT (OFFICER/WARRANT)'  
OPEN(3,FILE='/home/warpam/iofiles/AUTO0190.DAT',STATUS='OLD')  
20 READ(3,'(17(A1))',ERR=300,END=200)CHRE  
  
NRCD = NRCD + 1  
TPC = LINE(1:2)  
CATC = LINE(3:3)  
SPEC1= LINE(4:4)  
SPEC2= LINE(5:5)  
SKLID= LINE(6:6)  
SPECC= LINE(4:5)  
MOSC = LINE(4:6)  
SKLC = LINE(7:7)  
GRDC = LINE(8:8)  
SEXC = LINE(9:9)  
THRC = LINE(10:11)  
STRC = LINE(12:15)  
RCDID= LINE(16:17)
```

C Category verification [0 - officers, W - warrants]

```
IF ((CATC.EQ.'O').OR.(CATC.EQ.'W'))GOTO 66  
IF ((CATC.NE.'O').AND.(CATC.NE.'W'))GOTO 20
```

C Verify and translate Theater Code [THRC].
C Code descriptors: P1,P3,P8,NB,3A
C {AKO - Korea}, E1 {AE1 - Europe}.

```
66 IF ((THRC.NE.'E1').AND.(THRC.NE.'P1').AND.(THRC.NE.'P3').AND.  
&(THRC.NE.'P8').AND.(THRC.NE.'NB').AND.(THRC.NE.'3A'))GOTO 20  
IF (THRC.EQ.'E1')THEN  
  THRC='E1'  
  GOTO 67  
ENDIF
```

```

IF ((THRC.EQ.'P1').OR.(THRC.EQ.'P3').OR.(THRC.EQ.'P8'))THEN
  THRC='KO'
  GOTO 67
ENDIF
IF ((THRC.EQ.'NB').OR.(THRC.EQ.'3A'))THRC='KO'

```

C Verify and translate Time Period Code.

```

67 IFLAG = 0
DO 33 I = 1,25
  IF (TPC.EQ.TCHR(I))THEN
    IFLAG = 1
    TPC = TPN(I)
    GOTO 34
  ENDIF
33 CONTINUE
34 IF (IFLAG.EQ.0)GOTO 20

```

C Calls subroutine BRLOOKUP which translates the branch code.

```

CALL BRLOOKUP

```

C Translates SEX and GRADE codes.

```

IF (CATC.EQ.'W')THEN
  GRDE='WW'
  GOTO 79
ENDIF
IF (CATC.EQ.'O')THEN
  IF ((GRDC.EQ.'A').OR.(GRDC.EQ.'B'))THEN
    GRDE='FD'
    GOTO 78
  ENDIF
  IF ((GRDC.EQ.'C').OR.(GRDC.EQ.'D'))THEN
    GRDE='FD'
    GOTO 78
  ENDIF
  IF ((GRDC.EQ.'4').OR.(GRDC.EQ.'5').OR.(GRDC.EQ.'6'))THEN
    GRDE='FD'
    GOTO 78
  ENDIF
  IF ((GRDC.EQ.'7').OR.(GRDC.EQ.'8').OR.(GRDC.EQ.'9'))THEN
    GRDE='FD'
    GOTO 78
  ENDIF
  IF ((GRDC.EQ.'1').OR.(GRDC.EQ.'2').OR.(GRDC.EQ.'3'))THEN
    GRDE='CO'
    GOTO 78
  ENDIF
  IF ((GRDC.EQ.' ').OR.(GRDC.EQ.'E').OR.(GRDC.EQ.'F'))THEN
    GRDE='CO'
    GOTO 78
  ENDIF

```

```

        ENDIF
    ENDIF
78  IF ((CATC.EQ.'O').AND.(SEXC.EQ.' '))THEN
        IF ((SPECC.EQ.'11').OR.(SPECC.EQ.'12').OR.(SPECC.EQ.'18'))THEN
            SEXC='M'
        ELSE
            SEXC='X'
        ENDIF
        GOTO 80
    ENDIF
79  IF (SEXC.EQ.'Z')THEN
        SEXC='X'
        GOTO 80
    ENDIF
    IF (SEXC.EQ.'M')THEN
        SEXC='M'
        GOTO 80
    ENDIF
    IF ((SEXC.NE.'Z').AND.(SEXC.NE.'M'))SEXC='X'

```

C Converts strength [STRC] from character to a numeric value; such
C that the strengths may be summated.

```

80  STR = 0
    DO 22 I = 1,17
        IF (I.LT.12)GOTO 22
        IF (I.GT.15)GOTO 22
        XX = ICHAR(LINE(I:I))
        NUM = (79-(127-XX))
        IF (NUM.LT.0)NUM = 0
        IF (I.EQ.12)NUM = NUM * 1000
        IF (I.EQ.13)NUM = NUM * 100
        IF (I.EQ.14)NUM = NUM * 10
        IF (I.EQ.15)NUM = NUM * 1
        STR = STR + NUM
    22  CONTINUE

```

C Stores variables in array position for each time period (1-18) and
C for each theater (Europe and Korea).

```

    IF (NRCD.GT.1) GOTO 23
    IF (NRCD.EQ.1)THEN
        NARAY = NRCD
        VSTOR(NARAY,1) = TPC
        VSTOR(NARAY,2) = CATC
        VSTOR(NARAY,3) = BRR
        VSTOR(NARAY,4) = GRDE
        VSTOR(NARAY,5) = SEXC
        VSTOR(NARAY,6) = THRC
        STRNG(NARAY,1) = STR
        MAXARAY = NARAY
        GOTO 20
    ENDIF

```

```

ENDIF
23 IF (STR.EQ.0)GOTO 20
DO 24 I = 1,MAXARAY
  IF ((TPC.EQ.VSTOR(I,1)).AND.(CATC.EQ.VSTOR(I,2)))GOTO 30
  IFLG = 1
  GOTO 24
30 IF ((BRR.EQ.VSTOR(I,3)).AND.(GRDE.EQ.VSTOR(I,4)))GOTO 31
  IFLG = 1
  GOTO 24
31 IF ((SEXC.EQ.VSTOR(I,5)).AND.(THRC.EQ.VSTOR(I,6)))GOTO 32
  IFLG = 1
  GOTO 24
32 STRNG(I,1) = STR + STRNG(I,1)
  GOTO 20
24 CONTINUE
  IF (IFLG.EQ.1)THEN
    MAXARAY = MAXARAY + 1
    VSTOR(MAXARAY,1) = TPC
    VSTOR(MAXARAY,2) = CATC
    VSTOR(MAXARAY,3) = BRR
    VSTOR(MAXARAY,4) = GRDE
    VSTOR(MAXARAY,5) = SEXC
    VSTOR(MAXARAY,6) = THRC
    STRNG(MAXARAY,1) = STR
  ENDIF
  GOTO 20

```

C Writes message(s) to screen when end of file [EOF] is encountered
 C or when an error reading the file is encountered.
 C Close Input file: AUTO0190.DAT

```

300 WRITE(6,*)' ERROR DETECTED READING FILE.'
200 CLOSE(3,STATUS='KEEP')

```

C Opening output file: AUTOREP.OUT

```

OPEN(70,FILE='/home/warpam/iofiles/AUTOREP.OUT',STATUS='NEW')

DO 29 K = 1, MAXARAY
  IF (VSTOR(K,6).EQ.'E1')THEN
    THR = 'AE1'
  ELSE
    THR = 'AKO'
  ENDIF
  WRITE(70,46)VSTOR(K,1),VSTOR(K,2),VSTOR(K,3),VSTOR(K,4),
&VSTOR(K,5),THR,STRNG(K,1)

```

C Time period 14-18 are time period 13 requirements straight-lined
 C through 18 time periods.

```

IF ((VSTOR(K,1).EQ.'13').AND.(THR.EQ.'AKO'))THEN
  SLTP(1) = '14'

```

```

        SLTP(2) = '15'
        SLTP(3) = '16'
        SLTP(4) = '17'
        SLTP(5) = '18'
        DO 69 I = 1,5
            EXMAX = EXMAX + 1
            WRITE(70,46)SLTP(I),VSTOR(K,2),VSTOR(K,3),
&VSTOR(K,4),VSTOR(K,5),THR,STRNG(K,1)
69      CONTINUE
        ENDIF
        IF ((VSTOR(K,1).EQ.'10').AND.(THR.EQ.'AE1'))THEN
            SLTP(1) = '11'
            SLTP(2) = '12'
            SLTP(3) = '13'
            SLTP(4) = '14'
            SLTP(5) = '15'
            SLTP(6) = '16'
            SLTP(7) = '17'
            SLTP(8) = '18'
            DO 39 I = 1,8
                EXMAX = EXMAX + 1
                WRITE(70,46)SLTP(I),VSTOR(K,2),VSTOR(K,3),
&VSTOR(K,4),VSTOR(K,5),THR,STRNG(K,1)
39      CONTINUE
            ENDIF
46      FORMAT(2X,A2,3X,A1,2(A2),3X,A1,3X,A3,3X,I6)
29      CONTINUE
        MAXARAY = MAXARAY + EXMAX

```

C Temporarily close output file: AUTOREP.OUT. The AUTE0190.DAT
C results will be appended to the current output file.

```

        CLOSE(70,STATUS='KEEP')

```

C Recording input file validity.
C NRCD [Total record length]
C MAXARAY [Maximum number of processed records]

```

        WRITE(6,51)NRCD,MAXARAY
51      FORMAT(/15X,' INPUT FILE STATISTICS . . . AUTO0190.DAT',
&/8X,'Total No. of records in input file --->',I6,/8X,
&'Maximum No. of records processed --->',I6,/)

```

C Calls subroutine AUTOEREP which processes autorep file for
C enlisted officers by reading, extracting, and translating.

```

        CALL AUTOEREP

```

```

100  STOP
      END

```

C END AUTOEREP

SUBROUTINES

```

C *****
C
C Program Name:  TYMECODE                      Date:  04-17-1990
C
C File Name:    TYMEP.FOR
C
C Programmer:   Beth White, SAIC, 749-8771
C
C Description:  Reads and translates the time period codes.
C
C
C Input:        TP.TBL
C Output:       .
C
C *****
C Modifications: (STATUS:  P - PROPOSED; R - REQUIRED ; C - COMPLETED)
C
C Number   Status   Date:           Description:           Initials
C -----
C    01      C    05/30/90  Modified directory changes.      BAW
C
C *****

```

SUBROUTINE TYMECODE

C Global Variables

```

      DIMENSION TCHR(25),TPN(25)
      CHARACTER*2 TCHR,TPN,TC,TPX
      INTEGER L

```

```

      COMMON/CCTP/TCHR,TPN

```

C Local Variables

```

      L = 0

```

C BEGIN TYMECODE

C Opening input files: TP.TBL

```

      OPEN(60,FILE='/home/warpam/iofiles/TP.TBL',STATUS='OLD')
52  READ(60,16,ERR=88,END=99)TC,TPX
16  FORMAT(A2,1X,A2)
      L = L + 1
      TCHR(L) = TC
      TPN(L) = TPX
      GOTO 52
88  WRITE(6,*)' ERROR DETECTED READING FILE.'
```


C Close input file: TP.TBL, exit subroutine and return to main program.

```
99 CLOSE(60,STATUS='KEEP')  
    RETURN  
    END
```

C END TYMEP.FOR

```

C*****
C
C  Program Name:   BRLOOKUP                      Date: 04-17-1990
C
C  File Name:     BRNCH.FOR
C
C  Programmer:    Beth White, SAIC, 749-8771
C
C  Description:   Reads and extracts corresponding MOS {Military
C                 Occupation Speciality} code. The elements in the
C                 branch lookup table are in ascending order {low to high}
C                 for PERSCLASS/CATEGORY [ officer, warrant, enlisted].
C
C  Input:         BRANCH.TBL
C  Output:        .
C
C*****
C  Modifications: (STATUS: P - PROPOSED; R - REQUIRED ; C - COMPLETED)
C
C  Number   Status   Date:           Description:           Initials
C  -----
C    01      C      05/30/90  Modified directory changes.      BAW
C
C*****
C
C      SUBROUTINE BRLOOKUP
C
C      Global Variables
C
C          CHARACTER*1 BCHR(7),FRSTC,CATC,SPEC1,SPEC2,SKLID,BR1,BR2,BR3,BRNN
C          CHARACTER*2 SPECC,BRN,BRR
C          CHARACTER*3 MOSC,BRNUM
C          CHARACTER*7 BROW
C          INTEGER ICHK
C          COMMON/BRCH/CATC,MOSC,SPEC1,SPEC2,SPECC,SKLID,BRR
C          EQUIVALENCE (BCHR(1),BROW)
C
C      Local Variables
C
C          ICHK = 0
C
C      BEGIN BRLOOKUP
C
C      Opening input files:   BRANCH.TBL
C
C          OPEN(61,FILE='/home/warpam/iofiles/BRANCH.TBL',STATUS='OLD')
C          15 READ(61,'(7(A1))',ERR = 888,END = 999)BCHR
C
C      Extracts branch number code [BRNUM] and corresponding branch code
C      [BRCODE].
C
C          FRSTC = BROW(5:5)

```

```

IF (CATC.NE.FRSTC)GOTO 15
BR1 = BROW(1:1)
BR2 = BROW(2:2)
BR3 = BROW(3:3)
BRR = BROW(6:7)

```

```

IF ((BR2.NE.'*').AND.(BR3.NE.'*'))THEN
  BRNUM = BROW(1:3)
  IF(MOSC.EQ.BRNUM)THEN
    ICHK = ICHK + 1
    GOTO 16
  ENDIF
  GOTO 15
ENDIF

```

```

ENDIF
IF ((BR2.NE.'*').AND.(BR3.EQ.'*'))THEN
  BRN = BROW(1:2)
  IF(SPECC.EQ.BRN)THEN
    ICHK = ICHK + 1
    GOTO 16
  ENDIF
  GOTO 15
ENDIF

```

```

ENDIF
IF ((BR2.EQ.'*').AND.(BR3.EQ.'*'))THEN
  BRNN = BROW(1:1)
  IF(SPEC1.EQ.BRNN)THEN
    ICHK = ICHK + 1
    GOTO 16
  ENDIF
  GOTO 15
ENDIF
ENDIF

```

```

888 WRITE(6,*)' ERROR DETECTED READING FILE.'
999 IF (ICHK.EQ.0)THEN
  IF((CATC.EQ.'O').OR.(CATC.EQ.'E'))BRR = 'CS'
  IF(CATC.EQ.'W')BRR = 'CC'
ENDIF

```

```

C Close input file: BRANCH.TBL, exit subroutine and return to
C main program.

```

```

16 CLOSE(61,STATUS='KEEP')
RETURN
END

```

```

C END BRNCH.FOR

```

```

C*****
C
C  Program Name:  AUTOEREP                      Date:  04-10-1990
C
C  File Name:    AUTOREPE.FOR
C
C  Programmer:   Beth White, SAIC, 749-8771
C
C  Description:  Reads, extracts, and stores:  [REQID] time period,
C               [PERSCLASS]
C               category identifier, {MOS} military occupation speciality=
C               [SPECCD] speciality + [SKILID] skill level identifier +
C               [SKIL] skil, [GRADE] grade, [SEX] gender, [COMMAND] command/
C               theater, and [STRENGTH] strength/casualties.
C
C               For each unique time period with the corresponding theater,
C               MOS, grade, and gender the strength/casualties are
C               summated.
C               An output file is created which represents:  time period,
C               category identifier, branch, grade, sex, theater, and
C               strength.
C
C  Input:        AUTE0190.DAT
C
C  Output:       AUTOREP.OUT
C
C*****
C  Modifications: (STATUS:  P - PROPOSED; R - REQUIRED ; C - COMPLETED)
C
C  Number   Status   Date:           Description:           Initials
C  -----
C    01      C       05/30/90  Modified directory changes.      BAW
C
C*****

```

SUBROUTINE AUTOEREP

C Global Variables

```

DIMENSION VSTOR(3000,6),TCHR(25),TPN(25),SLTP(8)
CHARACTER*1 CHR(16),CATC,SKLC,GRDC,SEXC,SPEC1,SPEC2,SKLID
CHARACTER*2 TPC,TCHR,TPN,THRC,SPECC,GRDE,BRR,RCDID,VSTOR,SLTP
CHARACTER*3 MOSC,THR
CHARACTER*4 STRC
CHARACTER*16 LIN
INTEGER I,XX,NUM,STR,NROW,NARAY,MAXARAY,STRNG(3000,1),IFLG,K,
&IFLAG,EXMAX

```

```

COMMON/CCTP/TCHR,TPN
COMMON/BRCH/CATC,MOSC,SPEC1,SPEC2,SPECC,SKLID,BRR
EQUIVALENCE (CHR(1),LIN)

```

C Local Variables

```
NROW = 0
NARAY = 0
MAXARAY = 0
IFLG = 0
EXMAX = 0
```

C BEGIN AUTOEREP

C Passes global time period code translation from initial subroutine
C TYMECODE . The translation codes are made common to this
C subroutine: COMMON/CCTP/TCHR,TPN.

C Opening and reading input file: AUTE0190.DAT
C Extracts data into variables per line.

```
WRITE(6,*)' PROCESSING FILE: AUTE0190.DAT (ENLISTED)'  
OPEN(4,FILE='/home/warpam/iofiles/AUTE0190.DAT',STATUS='OLD')  
20 READ(4,'(16(A1))',ERR=888,END=500)CHR
```

```
NROW = NROW + 1  
TPC = LIN(1:2)  
CATC = LIN(3:3)  
SPEC1= LIN(4:4)  
SPEC2= LIN(5:5)  
SKLID= LIN(6:6)  
SPECC= LIN(4:5)  
MOSC = LIN(4:6)  
SKLC = ' '  
GRDC = LIN(7:7)  
SEXC = LIN(8:8)  
THRC = LIN(9:10)  
STRC = LIN(11:14)  
RCDID= LIN(15:16)
```

C Category verification [Only E - enlisted officers file]

```
IF (CATC.NE.'E')GOTO 20
```

C Verify and translate Theater Code [THRC].

C Code descriptors: P1,P3,P8,NB,3A
C {AKO - Korea}, E1 {AE1 - Europe}.

```
IF ((THRC.NE.'E1').AND.(THRC.NE.'P1').AND.(THRC.NE.'P3').AND.  
&(THRC.NE.'P8').AND.(THRC.NE.'NB').AND.(THRC.NE.'3A'))GOTO 20  
IF (THRC.EQ.'E1')THEN  
    THRC='E1'  
    GOTO 66  
ENDIF  
IF ((THRC.EQ.'P1').OR.(THRC.EQ.'P3').OR.(THRC.EQ.'P8'))THEN  
    THRC='KO'
```

```

        GOTO 66
    ENDIF
    IF ((THRC.EQ.'NB').OR.(THRC.EQ.'3A'))THRC='KO'

```

C Verify and translate Time Period Code.

```

66  IFLAG = 0
    DO 33 I = 1,25
        IF (TPC.EQ.TCHR(I))THEN
            IFLAG = 1
            TPC = TPN(I)
            GOTO 34
        ENDIF
    33  CONTINUE
    34  IF (IFLAG.EQ.0)GOTO 20

```

C Calls subroutine BRLOOKUP which translates the branch code.

```

        CALL BRLOOKUP

```

C Translates SEX and GRADE codes.

```

        IF ((GRDC.EQ.'1').OR.(GRDC.EQ.'2'))THEN
            GRDE='14'
            GOTO 88
        ENDIF
        IF ((GRDC.EQ.'3').OR.(GRDC.EQ.'4'))THEN
            GRDE='14'
            GOTO 88
        ENDIF
        IF ((GRDC.EQ.'5').OR.(GRDC.EQ.'6').OR.(GRDC.EQ.'7'))THEN
            GRDE='59'
            GOTO 88
        ENDIF
        IF ((GRDC.EQ.'8').OR.(GRDC.EQ.'9'))THEN
            GRDE='59'
            GOTO 88
        ENDIF
88  IF (SEXC.EQ.'M')THEN
        IF ((SPECC.EQ.'11').OR.(SPECC.EQ.'18').OR.(SPECC.EQ.'19'))THEN
            SEXC='M'
        ELSE
            SEXC='X'
        ENDIF
        GOTO 89
    ENDIF
    IF (SEXC.EQ.'M')THEN
        SEXC='M'
        GOTO 89
    ENDIF
    IF (SEXC.EQ.'Z')THEN
        SEXC='X'
    ENDIF

```

```

      GOTO 89
    ENDIF
    IF ((SEXC.NE.'Z').AND.(SEXC.NE.'M'))SEXC='X'

```

C Converts strength [STRC] from character to a numeric value; such
C that the strengths may be summated.

```

89  STR = 0
    DO 22 I = 1,16
      IF (I.LT.11)GOTO 22
      IF (I.GT.14)GOTO 22
      XX = ICHAR(LIN(I:I))
      NUM = (79-(127-XX))
      IF (NUM.LT.0)NUM = 0
      IF (I.EQ.11)NUM = NUM * 1000
      IF (I.EQ.12)NUM = NUM * 100
      IF (I.EQ.13)NUM = NUM * 10
      IF (I.EQ.14)NUM = NUM * 1
      STR = STR + NUM
    22 CONTINUE

```

C Stores variables in array position for each time period (1-18) and
C for each theater (Europe and Korea).

```

      IF (NROW.GT.1) GOTO 23
      IF (NROW.EQ.1)THEN
        NARAY = NROW
        VSTOR(NARAY,1) = TPC
        VSTOR(NARAY,2) = CATC
        VSTOR(NARAY,3) = BRR
        VSTOR(NARAY,4) = GRDE
        VSTOR(NARAY,5) = SEXC
        VSTOR(NARAY,6) = THRC
        STRNG(NARAY,1) = STR
        MAXARAY = NARAY
        GOTO 20
      ENDIF
23  IF (STR.EQ.0)GOTO 20
    DO 24 I = 1,MAXARAY
      IF ((TPC.EQ.VSTOR(I,1)).AND.(CATC.EQ.VSTOR(I,2)))GOTO 30
      IFLG = 1
      GOTO 24
30  IF ((BRR.EQ.VSTOR(I,3)).AND.(GRDE.EQ.VSTOR(I,4)))GOTO 31
      IFLG = 1
      GOTO 24
31  IF ((SEXC.EQ.VSTOR(I,5)).AND.(THRC.EQ.VSTOR(I,6)))GOTO 32
      IFLG = 1
      GOTO 24
32  STRNG(I,1) = STR + STRNG(I,1)
      GOTO 20
24  CONTINUE
    IF (IFLG.EQ.1)THEN

```

```

MAXARAY = MAXARAY + 1
VSTOR(MAXARAY,1) = TPC
VSTOR(MAXARAY,2) = CATC
VSTOR(MAXARAY,3) = BRR
VSTOR(MAXARAY,4) = GRDE
VSTOR(MAXARAY,5) = SEXC
VSTOR(MAXARAY,6) = THRC
STRNG(MAXARAY,1) = STR
ENDIF
GOTO 20

```

C Writes message(s) to screen when end of file [EOF] is encountered
 C or when an error reading the file is encountered.
 C Close Input file: AUTE0190.DAT

```

888 WRITE(6,*)' ERROR DETECTED READING FILE.'
500 CLOSE(4,STATUS='KEEP')

```

C Reopening output file : AUTOREP.OUT to append new output.

```

OPEN(70,FILE='/home/warpam/iofiles/AUTOREP.OUT',ACCESS='APPEND',
&STATUS='OLD')

```

```

DO 29 K = 1, MAXARAY
  IF (VSTOR(K,6).EQ.'E1')THEN
    THR = 'AE1'
  ELSE
    THR = 'AKO'
  ENDIF
  WRITE(70,46)VSTOR(K,1),VSTOR(K,2),VSTOR(K,3),VSTOR(K,4),
&VSTOR(K,5),THR,STRNG(K,1)

```

C Time period 14-18 are time period 13 requirements straight-lined
 C through 18 time periods.

```

  IF ((VSTOR(K,1).EQ.'13').AND.(THR.EQ.'AKO'))THEN
    SLTP(1) = '14'
    SLTP(2) = '15'
    SLTP(3) = '16'
    SLTP(4) = '17'
    SLTP(5) = '18'
    DO 69 I = 1,5
      EXMAX = EXMAX + 1
      WRITE(70,46)SLTP(I),VSTOR(K,2),VSTOR(K,3),
&VSTOR(K,4),VSTOR(K,5),THR,STRNG(K,1)
69    CONTINUE
  ENDIF
  IF ((VSTOR(K,1).EQ.'10').AND.(THR.EQ.'AE1'))THEN
    SLTP(1) = '11'
    SLTP(2) = '12'
    SLTP(3) = '13'
    SLTP(4) = '14'

```



```

        SLTP(5) = '15'
        SLTP(6) = '16'
        SLTP(7) = '17'
        SLTP(8) = '18'
        DO 39 I = 1,8
            EXMAX = EXMAX + 1
            WRITE(70,46)SLTP(I),VSTOR(K,2),VSTOR(K,3),
&VSTOR(K,4),VSTOR(K,5),THR,STRNG(K,1)
39      CONTINUE
        ENDIF
46      FORMAT(2X,A2,3X,A1,2(A2),3X,A1,3X,A3,3X,I6)
29      CONTINUE
        MAXARAY = MAXARAY + EXMAX

C   Close output file:  AUTOREP.OUT

        CLOSE(70,STATUS='KEEP')

C   Recording input file validity.
C   NROW [Total record length]
C   MAXARAY [Maximum number of processed records]

        WRITE(6,52)NROW,MAXARAY
52      FORMAT(/15X,' INPUT FILE STATISTICS . . . AUTE0190.DAT',
&/8X,'Total No. of records in input file --->',I6,/8X,
&'Maximum No. of records processed --->',I6,/)

C   Exit subroutine and return to main program.

        RETURN
        END

C   END AUTOOREP

```

4.4 MOBMAN

4.4.1 GENERAL

The MOBMAN module converts the output developed for the Mobilization Directorate of PERSCOM to standard format. The new file is received on 1/2" tape in an ASCII format and must be converted by programmer personnel utilizing a mainframe. All conversion programs are written in FORTRAN 77 and requires the Branch look-up table. The module generates two output files. The requirements file contains replacement requirements labeled "DEG" for Defense Guidance while the assets file entries are labeled: THS-active THS, IRR-initial ready reserve, STY-standby reserve and IMA and RET-category one retirees. MOBMAN generates these two output files with a single pass through the input data. This process and the interrelationship of the routines and sub-routines is shown in the figure below.

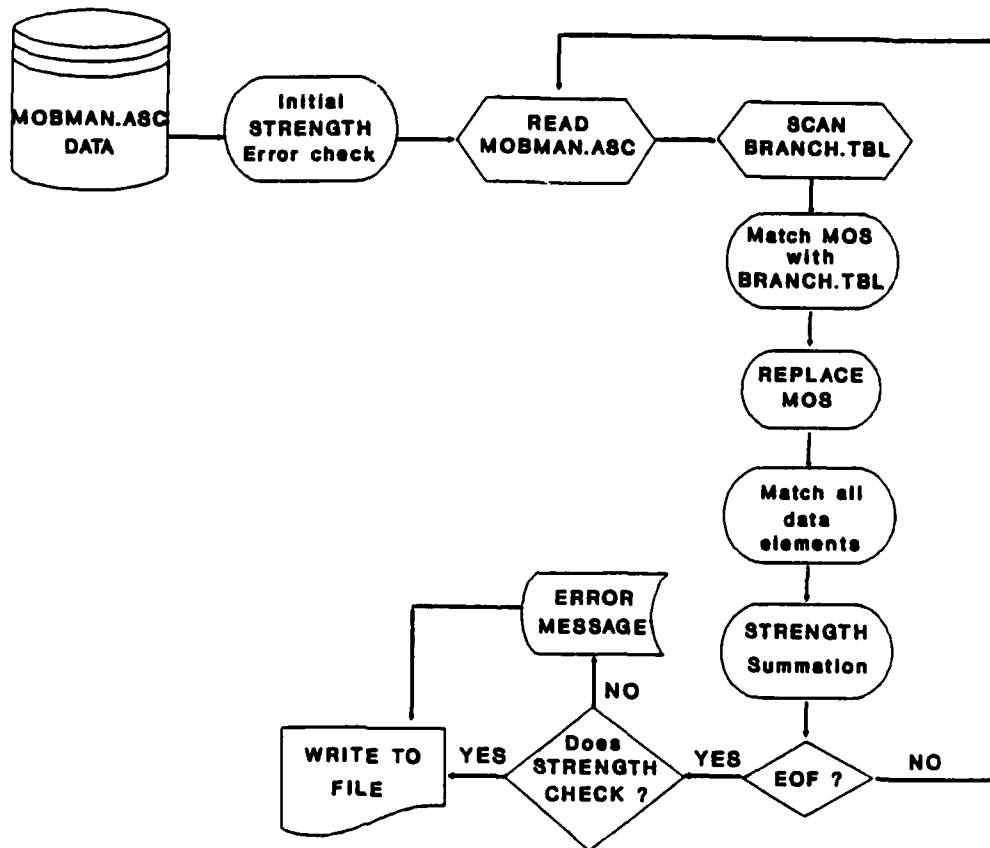


FIGURE 4: MOBMAN FILE CONVERSION

4.4.2 MOBMAN FORTRAN PROGRAMS

```
C*****
C
C Program Name:  MOBRA                      Date:  05-04-1990
C
C File Name:    MOBMANN.FOR
C
C Programmer:   Beth White, SAIC, 749-8771
C
C Description:  Reads, extracts, and stores: [REQID] time period,[PERSCLASS]
C              category identifier, {MOS} military occupation speciality
C              =[SPECCD] speciality + [SKILID] skill level identifier +
C              [SKIL] skil, [GRADE] grade, [SEX] gender, [COMMAND] command/
C              theater, and [STRENGTH] strength/casualties.
C              Note: Requirements and Assets
C
C              For each unique time period with the corresponding theater,
C              MOS, grade, and gender the str/casualties are summated.
C              An output file is created which represents: time period,
C              category identifier, branch, grade, sex, theater, and
C              strength.
C
C Input:        MOBMAN2.DAT
C
C Output:       TEMPA.OUT  ---| Temporary files
C              TEMPR.OUT  ---|
C
C              MOBMREQ.OUT
C              MOBMAST.OUT
C
C*****
C Modifications: (STATUS: P - PROPOSED; R - REQUIRED ; C - COMPLETED)
C
C Number   Status   Date:           Description:           Initials
C -----
C    01      C      05/30/90  Modified directory changes.      BAW
C                               Changed variable scan in
C                               input file from ACTIVE to
C                               THS.
C
C*****
C
C      PROGRAM MOBRA
C
C Global Variables
C
C      DIMENSION TPC(18),HOLD(18)
C      CHARACTER*1 CHR(125),CATC,CATHLD,GRDC,SPEC1,SPEC2,SKLID,SEXC,
C      &GRDHL
C      CHARACTER*2 SPECC,BRR,GRDE,TPC
C      CHARACTER*3 XMOS,MOSC,VARBL,THRC
```

CHARACTER*125 NCHR

LOGICAL THERE

INTEGER IFLG,IFND,NMOS,STR,K,I,MAXREQ,STRG1,STRG2,STRG3,
&STRG4,STRG5,STRG6,STRG7,STRG8,STRG9,STRG10,STRG11,STRG12,
&STRG13,STRG14,STRG15,STRG16,STRG17,STRG18,STR1N,STR2N,STR3N,
&STR4N,STR5N,STR6N,STR7N,STR8N,STR9N,STR10N,STR11N,STR12N,
&STR13N,STR14N,STR15N,STR16N,STR17N,STR18N,STRENG,HOLD,ASET,
&ASSET1,ASSET2,ASSET3,ASSET4,ASSET5,ASSET6,REQUIR,FILOPN,NUM,XX

COMMON/BRCH/CATC,MOSC,SPEC1,SPEC2,SPECC,SKLID,BRR
EQUIVALENCE (CHR(1),NCHR)

DATA TPC(1) //'01'//,TPC(2) //'02'//,TPC(3) //'03'//,TPC(4) //'04'//,
&TPC(5) //'05'//,TPC(6) //'06'//,TPC(7) //'07'//,TPC(8) //'08'//,
&TPC(9) //'09'//,TPC(10) //'10'//,TPC(11) //'11'//,TPC(12) //'12'//,
&TPC(13) //'13'//,TPC(14) //'14'//,TPC(15) //'15'//,TPC(16) //'16'//,
&TPC(17) //'17'//,TPC(18) //'18'//

C Local Variables

IFLG = 0
IFND = 0
NMOS = 0
REQUIR = 0
ASET = 0
ASSET1 = 0
ASSET2 = 0
ASSET3 = 0
ASSET4 = 0
ASSET5 = 0
ASSET6 = 0
MAXREQ = 0

90 WRITE(6,90)
FORMAT(////////20X,'*****',
&/20X,'*****',//20X,
&'MOBMAN MODULE',//20X,
&'*****',/20X,
&'*****',////////20X,
&'THE FOLLOWING FILES ARE NEEDED:',/30X,
&'MOBMAN2.DAT',/30X,'BRANCH.TBL',////////)

PAUSE

91 WRITE(6,91)
FORMAT(////////)

C Checks to see if input files exist. If input files do not exist;
C then write error message and terminate program.

```

INQUIRE(FILE='/home/warpam/iofiles/MOBMAN2.DAT',EXIST=THERE)
IF (.NOT.THERE)THEN
    WRITE(6,*)'ERROR - MOBMAN2.DAT does not exist.'
    GOTO 100
ENDIF

```

```

INQUIRE(FILE='/home/warpam/iofiles/BRANCH.TBL',EXIST=THERE)
IF (.NOT.THERE)THEN
    WRITE(6,*)' ERROR - BRANCH.TBL does not exist.'
    GOTO 100
ENDIF

```

C Checks to see if output files exists. If output files do exists;
C then old output file is deleted.

```

INQUIRE(FILE='/home/warpam/iofiles/TEMPR.OUT',EXIST=THERE)
IF (THERE)THEN
    OPEN(16,FILE='/home/warpam/iofiles/TEMPR.OUT',STATUS='OLD')
    CLOSE(16,STATUS='DELETE')
ENDIF

```

```

INQUIRE(FILE='/home/warpam/iofiles/TEMPA.OUT',EXIST=THERE)
IF (THERE)THEN
    OPEN(17,FILE='/home/warpam/iofiles/TEMPA.OUT',STATUS='OLD')
    CLOSE(17,STATUS='DELETE')
ENDIF

```

C Creates and opens temporary files for requirement and asset
C output. The files will become old such that new records
C may be appended the files.
C Temporary Files: TEMPR.OUT
C TEMPA.OUT

```

OPEN(16,FILE='/home/warpam/iofiles/TEMPR.OUT',STATUS='NEW')
CLOSE(16,STATUS='KEEP')

```

```

OPEN(17,FILE='/home/warpam/iofiles/TEMPA.OUT',STATUS='NEW')
CLOSE(17,STATUS='KEEP')

```

C Begin MOBRA

C Opening input file: MOBMAN2.DAT

```

WRITE(6,*)'PROCESSING INPUT FILE: MOBMAN2.DAT'
OPEN(2,FILE='/home/warpam/iofiles/MOBMAN2.DAT',STATUS='OLD')
10 READ(2,'(125(A1))',ERR=88,END=99)CHR
    XMOS = NCHR(2:4)
    IF ((IFND.EQ.1).AND.(XMOS.NE.'MOS'))GOTO 10
    IF (XMOS.EQ.'MOS')THEN
        GRDHL = NCHR(11:11)
        IF (GRDHL.EQ.'6')THEN
            CATHLD = NCHR(13:13)

```

```

        IF (CATHLD.EQ.'E')THEN
            IFND = 1
            GOTO 10
        ENDIF
    ENDIF
    IF ((GRDHLD.EQ.'0').OR.(GRDHLD.EQ.'0').OR.(GRDHLD.EQ.' '))THEN
        IFND = 1
        GOTO 10
    ELSE
        NMOS = NMOS + 1
        IF (NMOS.GT.1)THEN
C Case where REQUIR (requirement input line) does not exist.
C Strength is set to zero .
            IF (REQUIR.EQ.0)STRENG = 0

C Looks for cases where all 6 of the assets [THS,SEL RESERVE,
C IMA, IRR, STANDBY, RETIREES] were not found in the input file.
C If not found, the Strength (STRENG) = 0 .
C SPECIAL CASES: STANDBY and IMA were not found then
C Total Standby strength = 0
C
C STANDBY fails and IMA exists then
C Total Standby strength = IMA (HOLD(i)) + 0

            IF (ASET.LT.6)THEN
                OPEN(17,FILE='/home/warpam/iofiles/TEMPA.OUT',
&ACCESS='APPEND',STATUS='OLD')
                FILOPN = 17
            ENDIF
            DO 110 WHILE (ASET.LT.6)
                IF (ASSET1.EQ.0)THEN
C Input line 'THS' does not exist.
                    ASSET1 = 1
                    STRENG = 0
                    GOTO 110
                ENDIF
                IF (ASSET2.EQ.0)THEN
C Input line 'SEL RESERVE does not exist.
                    ASSET2 = 1
                    STRENG = 0
                    GOTO 110
                ENDIF
                IF ((ASSET4.EQ.0).AND.(ASSET3.EQ.0))THEN
C Input line 'STANDBY' does not exist
C and input line 'IMA' does not exist.
                    ASSET4 = 1
                    ASSET3 = 1
                    STRENG = 0
                    ASET = ASET + 1
                    GOTO 110
                ENDIF
                IF ((ASSET4.EQ.0).AND.(ASSET3.EQ.1))THEN

```

```

C          Input line 'STANDBY' does not exist
C          and input line 'IMA' exists.
          THRC = 'STY'
          ASSET4 = 1
          DO 55 I = 1,18
            STRENG = HOLD(I) + 0
            IF (STRENG.EQ.0)GOTO 55
            WRITE(17,49)TPC(I),CATC,BRR,GRDE,SEXC,THRC,STRENG
55          CONTINUE
          GOTO 110
        ENDIF
C          IF (ASSET5.EQ.0)THEN
            Input line 'IRR' does not exist.
            ASSET5 = 1
            STRENG = 0
            GOTO 110
          ENDIF
C          IF (ASSET6.EQ.0)THEN
            Input line 'RETIREEES' does not exist.
            ASSET6 = 1
            STRENG = 0
            GOTO 110
          ENDIF
110        ASET = ASET + 1
          IF (FILOPN.EQ.17)CLOSE(17,STATUS='KEEP')
        ENDIF

        IFND = 0
        REQUIR = 0
        ASSET1 = 0
        ASSET2 = 0
        ASSET3 = 0
        ASSET4 = 0
        ASSET5 = 0
        ASSET6 = 0
        ASET = 0
        DO 555 I = 1,18
555          HOLD(I) = 0
          SPEC1 = NCHR(6:6)
          SPEC2 = NCHR(7:7)
          SKLID = NCHR(8:8)
          SPECC = NCHR(6:7)
          MOSC = NCHR(6:8)
          GRDC = NCHR(11:11)
          CATC = NCHR(13:13)
          SEXC = ' '

```

C Verify and translate Grade Code [GRDC].

```

          IF (CATC.EQ.'W')THEN
            GRDE = 'WW'
            GOTO 23

```

```

ENDIF
IF (CATC.EQ.'E')THEN
  IF (GRDC.EQ.'1')THEN
    GRDE = '14'
    GOTO 23
  ENDIF
  IF ((GRDC.EQ.'2').OR.(GRDC.EQ.'3'))THEN
    GRDE = '59'
    GOTO 23
  ENDIF
  IF ((GRDC.EQ.'4').OR.(GRDC.EQ.'5'))THEN
    GRDE = '59'
    GOTO 23
  ENDIF
ENDIF
ENDIF
IF (CATC.EQ.'O')THEN
  IF (GRDC.EQ.'1')THEN
    GRDE = 'CO'
    GOTO 23
  ENDIF
  IF ((GRDC.EQ.'2').OR.(GRDC.EQ.'3'))THEN
    GRDE = 'CO'
    GOTO 23
  ENDIF
  IF (GRDC.EQ.'4')THEN
    GRDE = 'FD'
    GOTO 23
  ENDIF
  IF ((GRDC.EQ.'5').OR.(GRDC.EQ.'6'))THEN
    GRDE = 'FD'
    GOTO 23
  ENDIF
ENDIF
ENDIF

```

C Calls subroutine BRLOOKUP which translates the branch code.

```

23      CALL BRLOOKUP
        GOTO 10
      ENDIF
    ENDIF
    IF (XMOS.NE.'MOS')THEN
      VARBL = NCHR(6:8)
      IF ((VARBL.NE.'CAS').AND.(VARBL.NE.'THS').AND.(VARBL.NE.'SEL')
&.AND.(VARBL.NE.'IMA').AND.(VARBL.NE.'IRR').AND.(VARBL.NE.'STA')
&.AND.(VARBL.NE.'RET'))GOTO 10
    
```

C If input line = 'Casualty' ; then THRC = DEG

```

      IF (VARBL.EQ.'CAS')THEN
        THRC = 'DEG'
        REQUIR = 1
        GOTO 35
      ENDIF
    
```



```

C  If input line = 'Ths'          ; then THRC = THS
    IF (VARBL.EQ.'THS')THEN
        THRC = 'THS'
        ASSET1 = 1
        ASET = ASET + 1
        GOTO 35
    ENDIF

```

The Select Reserve input variable is inactivated as the preponderance of these personnel are in troop units. When the input file, MOBMAN, is capable of distinguishing between individual select reserve and personnel in troop units, this selection may be activated by replacing the "GOTO 10" LINE WITH "GOTO 35". This assumes that the input file only contains individual reserves. If it does not, than an additional discriminator must be used to eliminate the troop unit personnel as these should not be considered in filling the individual replacement requirements addressed in WARPAM.

```

C  If input line = 'Sel Reserve'; then THRC = SEL
    IF (VARBL.EQ.'SEL')THEN
        THRC = 'SEL'
        ASSET2 = 1
        ASET = ASET + 1
        GOTO 10
    ENDIF

```

```

C  If input line = 'Ima'          ; then THRC = STY
    IF (VARBL.EQ.'IMA')THEN
        DO 19 I = 1,18
19      HOLD(I) = 0
        THRC = 'STY'
        ASSET3 = 1
        ASET = ASET + 1
        GOTO 35
    ENDIF

```

```

C  If input line = 'Standby'      ; then THRC = STY
C  Note: STY = IMA + STY
    IF (VARBL.EQ.'STA')THEN
        THRC = 'STY'
        ASSET4 = 1
        ASET = ASET + 1
        GOTO 35
    ENDIF

```

```

C  If input line = 'Irr'          ; then THRC = IRR
    IF (VARBL.EQ.'IRR')THEN
        THRC = 'IRR'
    ENDIF

```

```

        ASSET5 = 1
        ASET = ASET + 1
        GOTO 35
    ENDIF

C   If input line = 'Retirees' ; then THRC = RET
    IF (VARBL.EQ.'RET')THEN
        THRC = 'RET'
        ASSET6 = 1
        ASET = ASET + 1
        GOTO 35
    ENDIF
ENDIF

C   Converts strength [STRC] from character to a numeric value; such
C   that the strengths may be summated.

35   STR = 0
    K = 0
    DO 22 I = 1,125
        IF (I.LT.40)GOTO 22
        IF ((I.GT.44).AND.(I.LT.49))GOTO 22
        IF ((I.GT.53).AND.(I.LT.58))GOTO 22
        IF ((I.GT.62).AND.(I.LT.67))GOTO 22
        IF ((I.GT.71).AND.(I.LT.76))GOTO 22
        IF ((I.GT.80).AND.(I.LT.85))GOTO 22
        IF ((I.GT.89).AND.(I.LT.94))GOTO 22
        IF ((I.GT.98).AND.(I.LT.103))GOTO 22
        IF ((I.GT.107).AND.(I.LT.112))GOTO 22
        IF ((I.GT.116).AND.(I.LT.121))GOTO 22
        XX = ICHAR(NCHR(I:I))
        NUM = (79-(127-XX))
        IF (NUM.LT.0)NUM = 0
        IF ((I.GT.39).AND.(I.LT.45))THEN
            K = K + 1
            IF (I.EQ.40)NUM = NUM * 10000
            IF (I.EQ.41)NUM = NUM * 1000
            IF (I.EQ.42)NUM = NUM * 100
            IF (I.EQ.43)NUM = NUM * 10
            IF (I.EQ.44)NUM = NUM * 1
            STR = STR + NUM
            IF (K.EQ.5)THEN
                STRG1 = STR
                IF (VARBL.EQ.'IMA')THEN
                    HOLD(1) = STRG1
                    IF (HOLD(1).LT.0)HOLD(1) = 0
                ENDIF
                K = 0
                STR = 0
            ENDIF
        ENDIF
        GOTO 22
    ENDIF

```

```

IF ((I.GT.48).AND.(I.LT.54))THEN
  K = K + 1
  IF (I.EQ.49)NUM = NUM * 10000
  IF (I.EQ.50)NUM = NUM * 1000
  IF (I.EQ.51)NUM = NUM * 100
  IF (I.EQ.52)NUM = NUM * 10
  IF (I.EQ.53)NUM = NUM * 1
  STR = STR + NUM
  IF (K.EQ.5)THEN
    STRG2 = STR
    IF (VARBL.EQ.'IMA')THEN
      HOLD(2) = STRG2 - STRG1
      IF (HOLD(2).LT.0)HOLD(2) = 0
    ENDIF
    K = 0
    STR = 0
  ENDIF
  GOTO 22
ENDIF
IF ((I.GT.57).AND.(I.LT.63))THEN
  K = K + 1
  IF (I.EQ.58)NUM = NUM * 10000
  IF (I.EQ.59)NUM = NUM * 1000
  IF (I.EQ.60)NUM = NUM * 100
  IF (I.EQ.61)NUM = NUM * 10
  IF (I.EQ.62)NUM = NUM * 1
  STR = STR + NUM
  IF (K.EQ.5)THEN
    STRG3 = STR
    IF (VARBL.EQ.'IMA')THEN
      HOLD(3) = STRG3 - STRG2
      IF (HOLD(3).LT.0)HOLD(3) = 0
    ENDIF
    K = 0
    STR = 0
  ENDIF
  GOTO 22
ENDIF
IF ((I.GT.66).AND.(I.LT.72))THEN
  K = K + 1
  IF (I.EQ.67)NUM = NUM * 10000
  IF (I.EQ.68)NUM = NUM * 1000
  IF (I.EQ.69)NUM = NUM * 100
  IF (I.EQ.70)NUM = NUM * 10
  IF (I.EQ.71)NUM = NUM * 1
  STR = STR + NUM
  IF (K.EQ.5)THEN
    STRG4 = STR
    IF (VARBL.EQ.'IMA')THEN
      HOLD(4) = STRG4 - STRG3
      IF (HOLD(4).LT.0)HOLD(4) = 0
    ENDIF
  ENDIF

```

```

        K = 0
        STR = 0
    ENDIF
    GOTO 22
ENDIF
IF ((I.GT.75).AND.(I.LT.81))THEN
    K = K + 1
    IF (I.EQ.76)NUM = NUM * 10000
    IF (I.EQ.77)NUM = NUM * 1000
    IF (I.EQ.78)NUM = NUM * 100
    IF (I.EQ.79)NUM = NUM * 10
    IF (I.EQ.80)NUM = NUM * 1
    STR = STR + NUM
    IF (K.EQ.5)THEN
        STRG5 = STR
        IF (VARBL.EQ.'IMA')THEN
            HOLD(5) = STRG5 - STRG4
            IF (HOLD(5).LT.0)HOLD(5) = 0
        ENDIF
        K = 0
        STR = 0
    ENDIF
    GOTO 22
ENDIF
IF ((I.GT.84).AND.(I.LT.90))THEN
    K = K + 1
    IF (I.EQ.85)NUM = NUM * 10000
    IF (I.EQ.86)NUM = NUM * 1000
    IF (I.EQ.87)NUM = NUM * 100
    IF (I.EQ.88)NUM = NUM * 10
    IF (I.EQ.89)NUM = NUM * 1
    STR = STR + NUM
    IF (K.EQ.5)THEN
        STRG6 = STR
        STRG7 = STR
        STRG8 = STR
        IF (VARBL.EQ.'IMA')THEN
            HOLD(6) = NINT((STRG6 - STRG5)/3.0)
            HOLD(7) = NINT((STRG7 - STRG6)/3.0)
            HOLD(8) = NINT((STRG8 - STRG7)/3.0)
            IF (HOLD(6).LT.0)HOLD(6) = 0
            IF (HOLD(7).LT.0)HOLD(7) = 0
            IF (HOLD(8).LT.0)HOLD(8) = 0
        ENDIF
        K = 0
        STR = 0
    ENDIF
    GOTO 22
ENDIF
IF ((I.GT.93).AND.(I.LT.99))THEN
    K = K + 1
    IF (I.EQ.94)NUM = NUM * 10000

```

```

IF (I.EQ.95)NUM = NUM * 1000
IF (I.EQ.96)NUM = NUM * 100
IF (I.EQ.97)NUM = NUM * 10
IF (I.EQ.98)NUM = NUM * 1
STR = STR + NUM
IF (K.EQ.5)THEN
  STRG9 = STR
  STRG10 = STR
  STRG11 = STR
  IF (VARBL.EQ.'IMA')THEN
    HOLD(9) = NINT((STRG9 - STRG8)/3.0)
    HOLD(10) = NINT((STRG10 - STRG9)/3.0)
    HOLD(11) = NINT((STRG11 - STRG10)/3.0)
    IF (HOLD(9).LT.0)HOLD(9) = 0
    IF (HOLD(10).LT.0)HOLD(10) = 0
    IF (HOLD(11).LT.0)HOLD(11) = 0
  ENDIF
  K = 0
  STR = 0
ENDIF
GOTO 22
ENDIF
IF ((I.GT.102).AND.(I.LT.108))THEN
  K = K + 1
  IF (I.EQ.103)NUM = NUM * 10000
  IF (I.EQ.104)NUM = NUM * 1000
  IF (I.EQ.105)NUM = NUM * 100
  IF (I.EQ.106)NUM = NUM * 10
  IF (I.EQ.107)NUM = NUM * 1
  STR = STR + NUM
  IF (K.EQ.5)THEN
    STRG12 = STR
    STRG13 = STR
    STRG14 = STR
    IF (VARBL.EQ.'IMA')THEN
      HOLD(12) = NINT((STRG12 - STRG11)/3.0)
      HOLD(13) = NINT((STRG13 - STRG12)/3.0)
      HOLD(14) = NINT((STRG14 - STRG13)/3.0)
      IF (HOLD(12).LT.0)HOLD(12) = 0
      IF (HOLD(13).LT.0)HOLD(13) = 0
      IF (HOLD(14).LT.0)HOLD(14) = 0
    ENDIF
    K = 0
    STR = 0
  ENDIF
  GOTO 22
ENDIF
IF ((I.GT.111).AND.(I.LT.117))THEN
  K = K + 1
  IF (I.EQ.112)NUM = NUM * 10000
  IF (I.EQ.113)NUM = NUM * 1000
  IF (I.EQ.114)NUM = NUM * 100

```

```

        IF (I.EQ.115)NUM = NUM * 10
        IF (I.EQ.116)NUM = NUM * 1
        STR = STR + NUM
        IF (K.EQ.5)THEN
            STRG15 = STR
            STRG16 = STR
            STRG17 = STR
            IF (VARBL.EQ.'IMA')THEN
                HOLD(15) = NINT((STRG15 - STRG14)/3.0)
                HOLD(16) = NINT((STRG16 - STRG15)/3.0)
                HOLD(17) = NINT((STRG17 - STRG16)/3.0)
                IF (HOLD(15).LT.0)HOLD(15) = 0
                IF (HOLD(16).LT.0)HOLD(16) = 0
                IF (HOLD(17).LT.0)HOLD(17) = 0
            ENDIF
            K = 0
            STR = 0
        ENDIF
        GOTO 22
    ENDIF
    IF (I.GT.120)THEN
        K = K + 1
        IF (I.EQ.121)NUM = NUM * 10000
        IF (I.EQ.122)NUM = NUM * 1000
        IF (I.EQ.123)NUM = NUM * 100
        IF (I.EQ.124)NUM = NUM * 10
        IF (I.EQ.125)NUM = NUM * 1
        STR = STR + NUM
        IF (K.EQ.5)THEN
            STRG18 = STR
            IF (VARBL.EQ.'IMA')THEN
                HOLD(18) = STRG18 - STRG17
                IF (HOLD(18).LT.0)HOLD(18) = 0
            ENDIF
            K = 0
            STR = 0
        ENDIF
        GOTO 22
    ENDIF
22  CONTINUE
    IF (VARBL.EQ.'IMA')GOTO 10

    STR1N = STRG1
    IF (STR1N.LT.0)STR1N = 0

    STR2N = STRG2 - STRG1
    IF (STR2N.LT.0)STR2N = 0

    STR3N = STRG3 - STRG2
    IF (STR3N.LT.0)STR3N = 0

    STR4N = STRG4 - STRG3

```

IF (STR4N.LT.0)STR4N = 0

STR5N = STRG5 - STRG4

IF (STR5N.LT.0)STR5N = 0

STR6N = NINT((STRG6 - STRG5)/3.0)

IF (STR6N.LT.0)STR6N = 0

STR7N = STR6N

STR8N = STR6N

STR9N = NINT((STRG9 - STRG8)/3.0)

IF (STR9N.LT.0)STR9N = 0

STR10N = STR9N

STR11N = STR9N

STR12N = NINT((STRG12 - STRG11)/3.0)

IF (STR12N.LT.0)STR12N = 0

STR13N = STR12N

STR14N = STR12N

STR15N = NINT((STRG15 - STRG14)/3.0)

IF (STR15N.LT.0)STR15N = 0

STR16N = STR15N

STR17N = STR15N

STR18N = STRG18 - STRG17

IF (STR18N.LT.0)STR18N = 0

C Opens output files: TEMPR.OUT [Requirements output] and
C TEMPA.OUT [Assets output].

IF (VARBL.EQ.'CAS')THEN

OPEN(16,FILE='/home/warpam/iofiles/TEMPR.OUT',ACCESS='APPEND',
&STATUS='OLD')

FILOPN = 16

ELSE

OPEN(17,FILE='/home/warpam/iofiles/TEMPA.OUT',ACCESS='APPEND',
&STATUS='OLD')

FILOPN = 17

ENDIF

DO 14 I = 1,18

IF (I.EQ.1)STRENG = STR1N

IF (I.EQ.2)STRENG = STR2N

IF (I.EQ.3)STRENG = STR3N

IF (I.EQ.4)STRENG = STR4N

IF (I.EQ.5)STRENG = STR5N

IF (I.EQ.6)STRENG = STR6N

IF (I.EQ.7)STRENG = STR7N

IF (I.EQ.8)STRENG = STR8N

IF (I.EQ.9)STRENG = STR9N

IF (I.EQ.10)STRENG = STR10N

```

        IF (I.EQ.11)STRENG = STR11N
        IF (I.EQ.12)STRENG = STR12N
        IF (I.EQ.13)STRENG = STR13N
        IF (I.EQ.14)STRENG = STR14N
        IF (I.EQ.15)STRENG = STR15N
        IF (I.EQ.16)STRENG = STR16N
        IF (I.EQ.17)STRENG = STR17N
        IF (I.EQ.18)STRENG = STR18N

C   Case where input lines:  STANDBY and IMA exist.
C   If both exist; the Standby strength = current standby strength
C   plus IMA (HOLD array) for the current time period.
C   Equation:  STRENG = HOLD(I) + STRENG

C   Case where input line STANDBY exist and IMA fails.
C   If Standby exist and Ima fails; the Standby strength =
C   current standby strength + 0 (no IMA)

        IF ((VARBL.EQ.'STA').AND.(ASSET3.EQ.1))THEN
            STRENG = HOLD(I) + STRENG
        ENDIF
        IF ((VARBL.EQ.'STA').AND.(ASSET3.EQ.0))THEN
            ASSET3 = 1
            STRENG = STRENG + 0
        ENDIF
        IF (STRENG.EQ.0)GOTO 14
        IF (FILOPN.EQ.16)THEN
            WRITE(16,49)TPC(I),CATC,BRR,GRDE,SEXC,THRC,STRENG
        ELSE
            WRITE(17,49)TPC(I),CATC,BRR,GRDE,SEXC,THRC,STRENG
        ENDIF
49    FORMAT(2X,A2,2X,A1,2X,A2,2X,A2,2X,A1,2X,A3,2X,I6)
14    CONTINUE
        IF (FILOPN.EQ.16)CLOSE(16,STATUS='KEEP')
        IF (FILOPN.EQ.17)CLOSE(17,STATUS='KEEP')
        GOTO 10

88    WRITE(6,*)' ERROR - Reading File.'
99    CLOSE(2,STATUS='KEEP')
        CLOSE(16,STATUS='KEEP')
        CLOSE(17,STATUS='KEEP')

C   Writing results to output file.
        CALL MOBREQ
        CALL MOBAST

100   STOP
        END

C   END MOBMANN.FOR

```



```

C*****
C*****
C
C
C          SUBROUTINES
C*****

```

```

C*****

```

```

C
C Program Name:   BRLOOKUP                      Date: 04-17-1990
C
C File Name:     BRNCH.FOR
C
C Programmer:    Beth White, SAIC, 749-8771
C
C Description:   Reads and extracts corresponding MOS {Military
C               Occupation Speciality} code. The elements in the
C               branch lookup table are in ascending order {low to high}
C               for PERSCLASS/CATEGORY [ officer, warrant, enlisted].
C
C Input:         BRANCH.TBL
C Output:        .
C

```

```

C*****

```

```

C Modifications: (STATUS: P - PROPOSED; R - REQUIRED ; C - COMPLETED)
C
C Number   Status   Date:           Description:           Initials
C -----
C    01      C    05/30/90 Modified directory changes.      BAW
C

```

```

C*****

```

SUBROUTINE BRLOOKUP

```

C Global Variables

```

```

CHARACTER*1 BCHR(7),FRSTC,CATC,SPEC1,SPEC2,SKLID,BR1,BR2,BR3,BRNN
CHARACTER*2 SPECC,BRN,BRR
CHARACTER*3 MOSC,BRNUM
CHARACTER*7 BROW
INTEGER ICHK
COMMON/BRCH/CATC,MOSC,SPEC1,SPEC2,SPECC,SKLID,BRR
EQUIVALENCE (BCHR(1),BROW)

```

```

C Local Variables

```

```

ICHK = 0

```

```

C BEGIN BRLOOKUP

```

```

C Opening input files:   BRANCH.TBL

```

```

OPEN(61,FILE='/home/warpam/iofiles/BRANCH.TBL',STATUS='OLD')

```

```

15 READ(61,'(7(A1))',ERR = 888,END = 999)BCHR
C   Extracts branch number code [BRNUM] and corresponding branch code
    [BRCODE].

    FRSTC = BROW(5:5)
    IF (CATC.NE.FRSTC)GOTO 15
    BR1 = BROW(1:1)
    BR2 = BROW(2:2)
    BR3 = BROW(3:3)
    BRR = BROW(6:7)

    IF ((BR2.NE.'*').AND.(BR3.NE.'*'))THEN
        BRNUM = BROW(1:3)
        IF(MOSC.EQ.BRNUM)THEN
            ICHK = ICHK + 1
            GOTO 16
        ENDIF
        GOTO 15
    ENDIF
    IF ((BR2.NE.'*').AND.(BR3.EQ.'*'))THEN
        BRN = BROW(1:2)
        IF(SPECC.EQ.BRN)THEN
            ICHK = ICHK + 1
            GOTO 16
        ENDIF
        GOTO 15
    ENDIF
    IF ((BR2.EQ.'*').AND.(BR3.EQ.'*'))THEN
        BRNN = BROW(1:1)
        IF(SPEC1.EQ.BRNN)THEN
            ICHK = ICHK + 1
            GOTO 16
        ENDIF
        GOTO 15
    ENDIF

888 WRITE(6,*)' ERROR DETECTED READING FILE.'
999 IF (ICLK.EQ.0)THEN
    IF((CATC.EQ.'O').OR.(CATC.EQ.'E'))BRR = 'CS'
    IF(CATC.EQ.'W')BRR = 'CC'
ENDIF

C   Close input file: BRANCH.TBL, exit subroutine and return to
C   main program.

16 CLOSE(61,STATUS='KEEP')
   RETURN
   END

C   END BRNCH.FOR

```

```

C*****
C
C Program Name:  MOBREQ                      Date:  05-14-1990
C
C File Name:    MOBRSLT1.FOR
C
C Programmer:   Beth White, SAIC, 749-8771
C
C Description:  Writes MOBMAN2.DAT output to a file which is a
C               requirement file.
C
C Input:        TEMPR.OUT
C
C Output:       MOBMREQ.OUT
C*****
C Modifications: (STATUS:  P - PROPOSED; R - REQUIRED ; C - COMPLETED)
C
C Number      Status  Date:           Description:           Initials
C -----
C      01         C    05/30/90  Modified directory changes.      BAW
C*****
C
C      SUBROUTINE MOBREQ
C
C      Global Variables
C
C      DIMENSION VSTOR(7000,6),STRNG(7000,1)
C      CHARACTER*1 CATC,SEXC
C      CHARACTER*2 BRR,GRDE,TP
C      CHARACTER*3 THRC,VSTOR
C      LOGICAL THERE
C      INTEGER II,STRENG,STRNG,MAXARAY
C
C      Local Variables
C
C      II = 0
C      MAXARAY = 0
C
C      Checks to see if output file exists.  If output file exists;
C      then delete old output file and create a new one.
C
C      INQUIRE(FILE='/home/warpam/iofiles/MOBMREQ.OUT',EXIST=THERE)
C      IF (THERE)THEN
C          OPEN(81,FILE='/home/warpam/iofiles/MOBMREQ.OUT',STATUS='OLD')
C          CLOSE(81,STATUS='DELETE')
C      ENDIF
C
C      WRITE(6,*)' GENERATING OUTPUT FILE:  MOBMREQ.OUT '
C      OPEN(16,FILE='/home/warpam/iofiles/TEMPR.OUT',STATUS='OLD')
500  READ(16,87,ERR=888,END=999)TP,CATC,BRR,GRDE,SEXC,THRC,STRENG

```

```

87  FORMAT(2X,A2,2X,A1,2X,A2,2X,A2,2X,A1,2X,A3,2X,I6)
    IFLG = 0
    II = II + 1
    IF (II.GT.1)GOTO 501
    IF (II.EQ.1)THEN
        MAXARAY = II
        VSTOR(II,1) = TP
        VSTOR(II,2) = CATC
        VSTOR(II,3) = BRR
        VSTOR(II,4) = GRDE
        VSTOR(II,5) = SEXC
        VSTOR(II,6) = THRC
        STRNG(II,1) = STRENG
        GOTO 500
    ENDIF
501  DO 503 J = 1,MAXARAY
        IF ((TP.EQ.VSTOR(J,1)).AND.(CATC.EQ.VSTOR(J,2)))GOTO 510
        IFLG = 1
        GOTO 503
510  IF ((BRR.EQ.VSTOR(J,3)).AND.(GRDE.EQ.VSTOR(J,4)))GOTO 511
        IFLG = 1
        GOTO 503
511  IF ((SEXC.EQ.VSTOR(J,5)).AND.(THRC.EQ.VSTOR(J,6)))GOTO 512
        IFLG = 1
        GOTO 503
512  STRNG(J,1) = STRENG + STRNG(J,1)
        GOTO 500
503  CONTINUE
    IF (IFLG.EQ.1)THEN
        MAXARAY = MAXARAY + 1
        VSTOR(MAXARAY,1) = TP
        VSTOR(MAXARAY,2) = CATC
        VSTOR(MAXARAY,3) = BRR
        VSTOR(MAXARAY,4) = GRDE
        VSTOR(MAXARAY,5) = SEXC
        VSTOR(MAXARAY,6) = THRC
        STRNG(MAXARAY,1) = STRENG
        GOTO 500
    ENDIF

888  WRITE(6,*)' ERROR - Reading File.'
999  CLOSE(16,STATUS='KEEP')

    OPEN(81,FILE='/home/warpam/iofiles/MOBMREQ.OUT',STATUS='NEW')
    DO 797 I = 1,MAXARAY
        WRITE(81,444)VSTOR(I,1),VSTOR(I,2),VSTOR(I,3),VSTOR(I,4),
&VSTOR(I,5),VSTOR(I,6),STRNG(I,1)
444  FORMAT(2X,A2,2X,A1,2X,A2,2X,A2,2X,A1,2X,A3,2X,I6)
797  CONTINUE
    CLOSE(81,STATUS='KEEP')

    OPEN(16,FILE='/home/warpam/iofiles/TEMPR.OUT',STATUS='OLD')

```

CLOSE(16,STATUS='DELETE')

C Recording input file validity.

C II [Total record length]

C MAXARY [Maximum number of processed records]

WRITE(6,51)II,MAXARAY

51 FORMAT(/15X,' INPUT FILE STATISTICS . . . MOBMAN2.DAT',
&/8X,'Total No. of records in input file --->',I6,/8X,
&'Maximum No. of records processed --->',I6,/)

RETURN

END

C END MOBRSLT1.FOR

```

C*****
C
C Program Name:  MOBAST                      Date:  05-14-1990
C
C File Name:    MOBRSLT2.FOR
C
C Programmer:   Beth White, SAIC, 749-8771
C
C Description:  Writes MOBMAN2.DAT output to a file which is an
C               asset file.
C
C Input:        TEMPA.OUT
C
C Output:       MOBMAST.OUT
C
C*****
C Modifications: (STATUS: P - PROPOSED; R - REQUIRED ; C - COMPLETED)
C
C Number   Status   Date:           Description:           Initials
C -----
C    01      C      05/30/90  Modified directory changes.      BAW
C
C*****

```

SUBROUTINE MOBAST

C Global Variables

```

    DIMENSION VSTOR(7000,6),STRNG(7000,1)
    CHARACTER*1 CATC,SEXC
    CHARACTER*2 BRR,GRDE,TP
    CHARACTER*3 THRC,VSTOR
    LOGICAL THERE
    INTEGER II,STRENG,STRNG,MAXARAY

```

C Local Variables

```

    II = 0
    MAXARAY = 0

```

```

C Checks to see if output file exists.  If output file exists;
C then delete old output file and create a new one.

```

```

    INQUIRE(FILE='/home/warpam/iofiles/MOBMAST.OUT',EXIST=THERE)
    IF (THERE)THEN
        OPEN(3,FILE='/home/warpam/iofiles/MOBMAST.OUT',STATUS='OLD')
        CLOSE(3,STATUS='DELETE')
    ENDIF

```

```

    WRITE(6,*)' GENERATING OUTPUT FILE: MOBMAST.OUT '
    OPEN(17,FILE='/home/warpam/iofiles/TEMPA.OUT',STATUS='OLD')
500  READ(17,87,ERR=888,END=999)TP,CATC,BRR,GRDE,SEXC,THRC,STRENG

```

```

87  FORMAT(2X,A2,2X,A1,2X,A2,2X,A2,2X,A1,2X,A3,2X,I6)
    IFLG = 0
    II = II + 1
    IF (II.GT.1)GOTO 501
    IF (II.EQ.1)THEN
        MAXARAY = II
        VSTOR(II,1) = TP
        VSTOR(II,2) = CATC
        VSTOR(II,3) = BRR
        VSTOR(II,4) = GRDE
        VSTOR(II,5) = SEXC
        VSTOR(II,6) = THRC
        STRNG(II,1) = STRENG
        GOTO 500
    ENDIF
501  DO 503 J = 1,MAXARAY
        IF ((TP.EQ.VSTOR(J,1)).AND.(CATC.EQ.VSTOR(J,2)))GOTO 510
        IFLG = 1
        GOTO 503
510  IF ((BRR.EQ.VSTOR(J,3)).AND.(GRDE.EQ.VSTOR(J,4)))GOTO 511
        IFLG = 1
        GOTO 503
511  IF ((SEXC.EQ.VSTOR(J,5)).AND.(THRC.EQ.VSTOR(J,6)))GOTO 512
        IFLG = 1
        GOTO 503
512  STRNG(J,1) = STRENG + STRNG(J,1)
        GOTO 500
503  CONTINUE
    IF (IFLG.EQ.1)THEN
        MAXARAY = MAXARAY + 1
        VSTOR(MAXARAY,1) = TP
        VSTOR(MAXARAY,2) = CATC
        VSTOR(MAXARAY,3) = BRR
        VSTOR(MAXARAY,4) = GRDE
        VSTOR(MAXARAY,5) = SEXC
        VSTOR(MAXARAY,6) = THRC
        STRNG(MAXARAY,1) = STRENG
        GOTO 500
    ENDIF

888  WRITE(6,*)' ERROR - Reading File.'
999  CLOSE(17,STATUS='KEEP')

    OPEN(3,FILE='/home/warpam/iofiles/MOBBMAST.OUT',STATUS='NEW')
    DO 797 I = 1,MAXARAY
        WRITE(3,444)VSTOR(I,1),VSTOR(I,2),VSTOR(I,3),VSTOR(I,4),
&VSTOR(I,5),VSTOR(I,6),STRNG(I,1)
444  FORMAT(2X,A2,2X,A1,2X,A2,2X,A2,2X,A1,2X,A3,2X,I6)
797  CONTINUE
    CLOSE(3,STATUS='KEEP')

    OPEN(17,FILE='/home/warpam/iofiles/TEMPA.OUT',STATUS='OLD')

```

```

        CLOSE(17,STATUS='DELETE')

C  Recording input file validity.
C  II [ Total record length]
C  MAXARY [ Maximum number of processed records]

        WRITE(6,51)II,MAXARAY
51  FORMAT(/15X,' INPUT FILE STATISTICS . . . MOBMAN2.DAT',
        &/8X,'Total No. of records in input file --->',I6,/8X,
        &'Maximum No. of records processed --->',I6,/)

        RETURN
        END

C  END MOBRSLT2.FOR

```


4.5 CASUALTY STRATIFICATION MODEL II (CSM II) MODULE

4.5.1 GENERAL

This module converts the CSMII model output created by Soldiers Support Center to usable WARPAM configuration. The new file is received on 5 1/4" floppy disks. The file should be requested in ASCII format. The input files are loaded onto the Sun drive by way of the network and PC. This module requires the Branch look-up table. Conversion of this file results in the creation of two requirement files labeled, CSMT for the total casualty requirement and CSMB for the battle casualty only portion of the output. As CSM II is operated by an office in the immediate vicinity of TRAC-FBHN and as the level of combat in CSM II can be easily varied, different levels of command modeled utilizing WARPAM could be easily varied using CSM II model outputs. The input file from the CSM II model must be configured in the format shown immediately below.

M	S	S	CASUALTIES	
T P	K E		BATTLE	DNBI
P C MOS	L X			

01 E 00B	* *		0	0

LEGEND:
 TP--TIME PERIOD
 MPC--MIL PERSONNEL CATEGORY
 SKL--GRADE

The processing flow of the CSMII module is shown below at Figure 5.

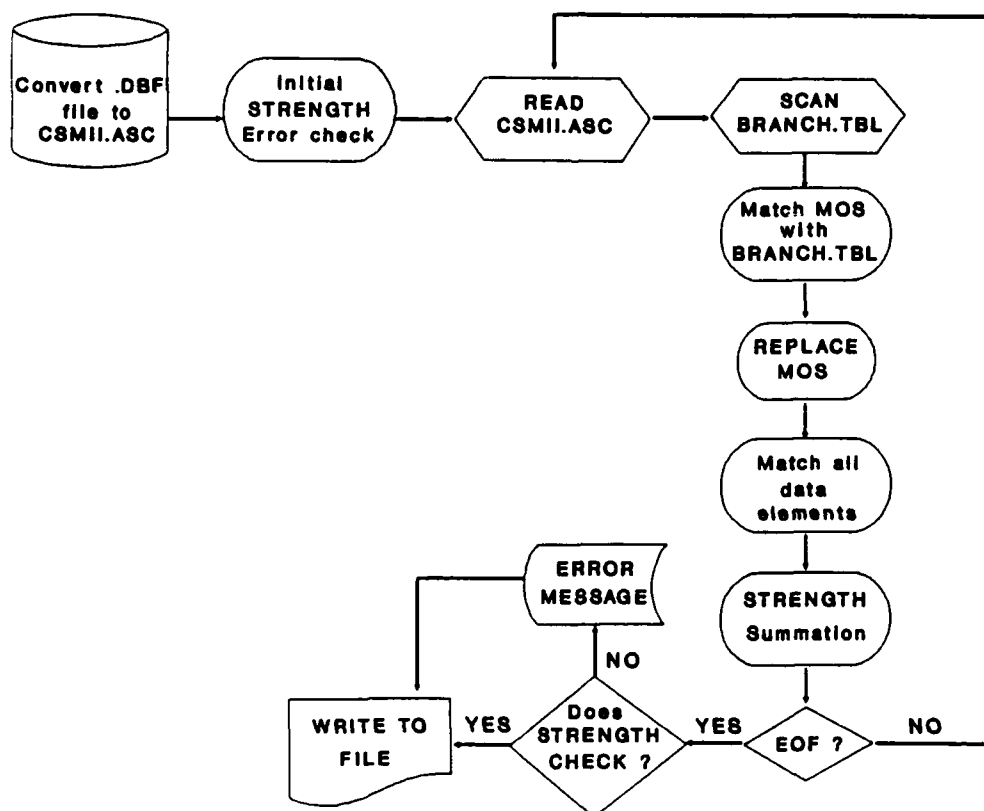


FIGURE 5: CSM II FILE CONVERSION

4.5.2 CSM II FORTRAN PROGRAMS

```
*****
*
* PROGRAMMER : JOHN A. TENSCHAW
* COMPANY    : SAIC
* ADDRESS    : 1710 GOODRIDGE DR. MS-T-1-7-2, MCLEAN, VA. 22102
* PHONE      : 703-734-5584
* DATE       : 25 APRIL 90
*
```

```
*****
*
* PROGRAM CSMII
* This program reads the csmii.dat file, condenses the info according
* to mos groupings, skill, and sex, and then creates an output file
* to store the total num. of casualties.
*
```

```
* DEFINITION OF VARIABLES :
* NUME, NUMO, NUMW - the number of rows in the enlisted, officer,
*                   and warrant officer matrices.
* PERIOD - the number of time periods.
* EBMAT, ENBMAT - the enlisted battle/non-battle casualty matrices.
* OBMAT, ONBMAT - the officer battle/non-battle casualty matrices.
* WBMAT, WNBMAT - the warrant off battle/non-battle casualty matrices
* TP - the time period specified on the input line; chars. 1 + 2
*     of input line.
* BAT, NBAT - the battle/non-battle casualties specified on the
*             input line.
* MPC - either o/w/e; the 4th char. of the input line.
* MTAG1, MTAG2 - the last 2 chars. in mos, blank except for warrant
*               officers; chars. 8 + 9 of input line.
* SKL - the skill number or *; char. 12 of input line.
* SEX - the sex of the personnel, either m/x/*; char.15 of input line
* CTR1, CTR2 = integer counters
* COUNT = the num of lines of input read in
* THERE = a logical variable; does the file exist or not
```

```
* VIEW OF THE OFFICER MATRIX :
* - EACH MATRIX ELEMENT IS A SIX DIGIT INTEGER
* - EACH MOS TYPE ACTUALLY CONSISTS OF 4 CONSECUTIVE ROWS;
*   WITH THE ROW NUMS IN PARENTHESIS

*
*                   TIME PERIODS
*      1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18
* OAD (1-4)
* OAR (5-8)
* OAV (9-12)
```

* OCE (13-16) AN EXAMPLE MOS SET OF 4 ROWS :
 * OCM (17-20) 1 - MOS - SEX=M - SKILL=14
 * M OCS (21-24) 2 - MOS - SEX=*/X - SKILL=14
 * O OFA (25-28) 3 - MOS - SEX=M - SKILL=59
 * S OIN (29-32) 4 - MOS - SEX=*/X - SKILL=FD
 * OMC (33-36)
 * OMI (37-40)
 * OMP (41-44)
 * OOD (45-48)
 * OQM (49-52)
 * OSC (53-56)
 * OTC (57-60)

* VIEW OF THE WARRANT OFFICER MATRIX :
 * - EACH MATRIX ELEMENT IS A SIX DIGIT INTEGER
 * - EACH MOS TYPE ACTUALLY CONSISTS OF 2 CONSECUTIVE ROWS;
 * WITH THE ROW NUMS IN PARENTHESIS

* TIME PERIODS
 * 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
 * M WCB (1-2) AN EXAMPLE MOS SET OF 2 ROWS :
 * O WCC (3-4) 1 - MOS - SEX=MALE - SKILL=WW
 * S WCS (5-6) 2 - MOS - SEX=*/X - SKILL=WW

* VIEW OF THE ENLISTED MATRIX :
 * - EACH MATRIX ELEMENT IS A SIX DIGIT INTEGER
 * - EACH MOS TYPE ACTUALLY CONSISTS OF 4 CONSECUTIVE ROWS;
 * WITH THE ROW NUMS IN PARENTHESIS

* TIME PERIODS
 * 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
 * EAD (1-4)
 * EAR (5-8)
 * EAV (9-12) AN EXAMPLE MOS SET OF 4 ROWS :
 * M ECE (13-16) 1 - MOS - SEX=M - SKILL=CO
 * O ECM (17-20) 2 - MOS - SEX=*/X - SKILL=CO
 * S ECS (21-24) 3 - MOS - SEX=M - SKILL=FD
 * EFA (25-28) 4 - MOS - SEX=*/X - SKILL=59
 * EIN (29-32)
 * EMC (33-36)
 * EMI (37-40)
 * EMM (41-44)
 * EMP (45-48)
 * EOD (49-52)
 * EQM (53-56)
 * ESC (57-60)
 * ESM (61-64)
 * ETC (65-68)

```

INTEGER NUME, NUMO, NUMW, PERIOD
PARAMETER (NUME = 68, NUMO = 60, NUMW = 6, PERIOD = 18)
INTEGER EBMAT, OBMAT, WBMAT, ENBMAT, ONBMAT, WNBMAT
DIMENSION EBMAT (NUME, PERIOD), ENBMAT (NUME, PERIOD)
DIMENSION OBMAT (NUMO, PERIOD), ONBMAT (NUMO, PERIOD)
DIMENSION WBMAT (NUMW, PERIOD), WNBMAT (NUMW, PERIOD)
INTEGER TP, BAT, NBAT
CHARACTER MPC, MTAG1, MTAG2, SKL, SEX
INTEGER CTR1, CTR2, COUNT
LOGICAL THERE

```

```

* Initialize the ematrix to 0
DO 6, CTR1 = 1, NUME
  DO 5, CTR2 = 1, PERIOD
    EBMAT (CTR1, CTR2) = 0
    ENBMAT (CTR1, CTR2) = 0
  5   CONTINUE
  6   CONTINUE

* Initialize the omatrix to 0
DO 8, CTR1 = 1, NUMO
  DO 7, CTR2 = 1, PERIOD
    OBMAT (CTR1, CTR2) = 0
    ONBMAT (CTR1, CTR2) = 0
  7   CONTINUE
  8   CONTINUE

* Initialize the wmatrix to 0
DO 10, CTR1 = 1, NUMW
  DO 9, CTR2 = 1, PERIOD
    WBMAT (CTR1, CTR2) = 0
    WNBMAT (CTR1, CTR2) = 0
  9   CONTINUE
  10  CONTINUE

* If input file does not exist, stop
INQUIRE (FILE = '/home/warpam/CSMII.DAT',EXIST = THERE)
IF (.NOT. THERE) THEN
  PRINT *, ' ERROR - CSMII.DAT DOES NOT EXIST'
  GO TO 110
END IF

* If output file exists, delete it
INQUIRE (FILE = '/home/warpam/CSMII.OUT',EXIST = THERE)
IF (THERE) THEN
  OPEN(UNIT = 2,FILE = '/home/warpam/CSMII.OUT',
  $    STATUS = 'OLD')
  CLOSE (2, STATUS = 'DELETE')
END IF

```

```

* Open the input and output files
  OPEN(UNIT = 1, FILE = '/home/warpam/CSMII.DAT',
    $   STATUS = 'OLD')
  OPEN(UNIT = 2, FILE = '/home/warpam/CSMII.OUT',
    $   STATUS = 'NEW')

* Initialize count = 1
  COUNT = 1

* Skip first three lines
  15  READ (1, '(A1)', END = 100) SEX
      IF (COUNT .LE. 4) THEN
        COUNT = COUNT + 1
        GO TO 15
      END IF

* Read input line and check for errors
  20  READ(1, 40, END = 80) TP, MPC, MOSNM, MTAG1, MTAG2, SKL,
    $   SEX, BAT, NBAT

  40  FORMAT(I2,1X,A1,1X,I2,A1,A1,2X,A1,2X,A1,1X,I6,3X,I6)

* Find correct row and modify matrices
      IF (MPC .EQ. 'O') THEN
        CALL OROW (OBBMAT, ONBBMAT, MOSNM, NUMO, PERIOD, SKL, SEX, TP,
    $           BAT, NBAT)
      ELSE IF (MPC .EQ. 'W') THEN
        CALL WROW (WBBMAT, WNBMAT, MOSNM, NUMW, PERIOD, SEX, TP,
    $           BAT, NBAT)
      ELSE
        CALL EROW (EBBMAT, ENBBMAT, MOSNM, MTAG1, NUME, PERIOD, SKL, SEX,
    $           TP, BAT, NBAT)
      ENDIF

* Increment count +1
      COUNT = COUNT + 1

* Go back to start of loop
      GO TO 20

  80  CALL PRINTMATRIX (OBBMAT, ONBBMAT, NUMO, WBBMAT, WNBMAT, NUMW,
    $           EBBMAT, ENBBMAT, NUME, PERIOD)

* Close files and exit program
  100 CLOSE(1, STATUS = 'KEEP')
      CLOSE(2, STATUS = 'KEEP')
  110 END

```

***** SUBROUTINES - IN ALPHABETICAL ORDER *****

```

SUBROUTINE EROW (ARRAY1, ARRAY2, MOSNM, MT1, NUMP, PER, SKILL,
$              SX, TIME, BT, NBT)
*****
*
* SUBROUTINE EROW
* This subroutine finds the correct 4 rows in the enlisted matrix, +
* passes them to 'modifyemat' to update the battle and non-battle
* matrix quantities.
*
*****

CHARACTER SKILL, SX, MT1
INTEGER NUMP, MOSNM, PER, TIME, BT, NBT, ARRAY1, ARRAY2
DIMENSION ARRAY1 (NUMP, PER), ARRAY2 (NUMP, PER)

* ECE
IF (((MOSNM .EQ. 41) .AND. (MT1 .EQ. 'B')) .OR.
$   ((MOSNM .EQ. 52) .AND. ((MT1 .EQ. 'E') .OR.
$   (MT1 .EQ. 'G')))) THEN
CALL MODIFYEMAT (ARRAY1, ARRAY2, NUMP, PER, SKILL, SX, TIME,
$              BT, NBT, 13, 14, 15, 16)

* EFA
ELSE IF (((MOSNM .EQ. 21) .AND. (MT1 .EQ. 'G')) .OR.
$   ((MOSNM .EQ. 82) .AND. (MT1 .EQ. 'C')) .OR.
$   ((MOSNM .EQ. 93) .AND. (MT1 .EQ. 'F')))) THEN
CALL MODIFYEMAT (ARRAY1, ARRAY2, NUMP, PER, SKILL, SX, TIME,
$              BT, NBT, 25, 26, 27, 28)

* EMC
ELSE IF (((MOSNM .EQ. 71) .AND. (MT1 .EQ. 'G')) .OR.
$   ((MOSNM .EQ. 76) .AND. (MT1 .EQ. 'J')) .OR.
$   ((MOSNM .EQ. 94) .AND. (MT1 .EQ. 'F')))) THEN
CALL MODIFYEMAT (ARRAY1, ARRAY2, NUMP, PER, SKILL, SX, TIME,
$              BT, NBT, 33, 34, 35, 36)

* EMM
ELSE IF (((MOSNM .EQ. 25) .AND. (MT1 .EQ. 'L')) .OR.
$   ((MOSNM .EQ. 46) .AND. (MT1 .EQ. 'N')))) THEN
CALL MODIFYEMAT (ARRAY1, ARRAY2, NUMP, PER, SKILL, SX, TIME,
$              BT, NBT, 41, 42, 43, 44)

* EOD
ELSE IF ((MOSNM .EQ. 62) .AND. (MT1 .EQ. 'B')) THEN
CALL MODIFYEMAT (ARRAY1, ARRAY2, NUMP, PER, SKILL, SX, TIME,
$              BT, NBT, 49, 50, 51, 52)

* ESM
ELSE IF ((MOSNM .EQ. 35) .AND. (MT1 .EQ. 'H')) THEN
CALL MODIFYEMAT (ARRAY1, ARRAY2, NUMP, PER, SKILL, SX, TIME,
$              BT, NBT, 61, 62, 63, 64)

* EAD

```

```

ELSE IF (MOSNM .EQ. 16) THEN
CALL MODIFYEMAT (ARRAY1, ARRAY2, NUMP, PER, SKILL, SX, TIME,
$               BT, NBT, 1, 2, 3, 4)
* EAR
ELSE IF (MOSNM .EQ. 19) THEN
CALL MODIFYEMAT (ARRAY1, ARRAY2, NUMP, PER, SKILL, SX, TIME,
$               BT, NBT, 5, 6, 7, 8)
* EAV
ELSE IF ((MOSNM .EQ. 66) .OR. (MOSNM .EQ. 67) .OR.
$       (MOSNM .EQ. 68) .OR. (MOSNM .EQ. 93)) THEN
CALL MODIFYEMAT (ARRAY1, ARRAY2, NUMP, PER, SKILL, SX, TIME,
$               BT, NBT, 9, 10, 11, 12)
* ECE
ELSE IF ((MOSNM .EQ. 12) .OR. (MOSNM .EQ. 51) .OR.
$       (MOSNM .EQ. 62) .OR. (MOSNM .EQ. 81) .OR.
$       (MOSNM .EQ. 82)) THEN
CALL MODIFYEMAT (ARRAY1, ARRAY2, NUMP, PER, SKILL, SX, TIME,
$               BT, NBT, 13, 14, 15, 16)
* ECM
ELSE IF (MOSNM .EQ. 54) THEN
CALL MODIFYEMAT (ARRAY1, ARRAY2, NUMP, PER, SKILL, SX, TIME,
$               BT, NBT, 17, 18, 19, 20)
* EFA
ELSE IF ((MOSNM .EQ. 13) .OR. (MOSNM .EQ. 15) .OR.
$       (MOSNM .EQ. 17)) THEN
CALL MODIFYEMAT (ARRAY1, ARRAY2, NUMP, PER, SKILL, SX, TIME,
$               BT, NBT, 25, 26, 27, 28)
* EIN
ELSE IF ((MOSNM .EQ. 11) .OR. (MOSNM .EQ. 18)) THEN
CALL MODIFYEMAT (ARRAY1, ARRAY2, NUMP, PER, SKILL, SX, TIME,
$               BT, NBT, 29, 30, 31, 32)
* EMC
ELSE IF ((MOSNM .EQ. 1) .OR. (MOSNM .EQ. 35) .OR.
$       (MOSNM .EQ. 42) .OR. (MOSNM .EQ. 91) .OR.
$       (MOSNM .EQ. 92)) THEN
CALL MODIFYEMAT (ARRAY1, ARRAY2, NUMP, PER, SKILL, SX, TIME,
$               BT, NBT, 33, 34, 35, 36)
* EMI
ELSE IF ((MOSNM .EQ. 5) .OR. (MOSNM .EQ. 96) .OR.
$       (MOSNM .EQ. 97) .OR. (MOSNM .EQ. 98)) THEN
CALL MODIFYEMAT (ARRAY1, ARRAY2, NUMP, PER, SKILL, SX, TIME,
$               BT, NBT, 37, 38, 39, 40)
* EMM
ELSE IF ((MOSNM .EQ. 21) .OR. (MOSNM .EQ. 24) .OR.
$       (MOSNM .EQ. 26) .OR. (MOSNM .EQ. 27)) THEN
CALL MODIFYEMAT (ARRAY1, ARRAY2, NUMP, PER, SKILL, SX, TIME,
$               BT, NBT, 41, 42, 43, 44)
* EMP
ELSE IF (MOSNM .EQ. 95) THEN
CALL MODIFYEMAT (ARRAY1, ARRAY2, NUMP, PER, SKILL, SX, TIME,
$               BT, NBT, 45, 46, 47, 48)
* EOD

```

```

        ELSE IF ((MOSNM .EQ. 41) .OR. (MOSNM .EQ. 44) .OR.
$           (MOSNM .EQ. 45) .OR. (MOSNM .EQ. 52) .OR.
$           (MOSNM .EQ. 63)) THEN
        CALL MODIFYEMAT (ARRAY1, ARRAY2, NUMP, PER, SKILL, SX, TIME,
$           BT, NBT, 49, 50, 51, 52)
* EQM
        ELSE IF ((MOSNM .EQ. 43) .OR. (MOSNM .EQ. 55) .OR.
$           (MOSNM .EQ. 57) .OR. (MOSNM .EQ. 76) .OR.
$           (MOSNM .EQ. 77) .OR. (MOSNM .EQ. 94)) THEN
        CALL MODIFYEMAT (ARRAY1, ARRAY2, NUMP, PER, SKILL, SX, TIME,
$           BT, NBT, 53, 54, 55, 56)
* ESC
        ELSE IF ((MOSNM .EQ. 29) .OR. (MOSNM .EQ. 31) .OR.
$           (MOSNM .EQ. 36) .OR. (MOSNM .EQ. 72)) THEN
        CALL MODIFYEMAT (ARRAY1, ARRAY2, NUMP, PER, SKILL, SX, TIME,
$           BT, NBT, 57, 58, 59, 60)
* ESM
        ELSE IF ((MOSNM .EQ. 33) .OR. (MOSNM .EQ. 39)) THEN
        CALL MODIFYEMAT (ARRAY1, ARRAY2, NUMP, PER, SKILL, SX, TIME,
$           BT, NBT, 61, 62, 63, 64)
* ETC
        ELSE IF (MOSNM .EQ. 88) THEN
        CALL MODIFYEMAT (ARRAY1, ARRAY2, NUMP, PER, SKILL, SX, TIME,
$           BT, NBT, 65, 66, 67, 68)
* ECS
        ELSE
        CALL MODIFYEMAT (ARRAY1, ARRAY2, NUMP, PER, SKILL, SX, TIME,
$           BT, NBT, 21, 22, 23, 24)
        ENDIF

* Exit subroutine
RETURN
END

```



```

      SUBROUTINE GRADESEX (TYPE, ROW, GRADE, SEX)
      *****
      *
      * SUBROUTINE GRADESEX
      * This subroutine determines the grade and sex given the matrix row
      * number.
      *
      *****

      INTEGER ROW
      CHARACTER TYPE, SEX, GRADE*2

      * Officer
      IF (TYPE .EQ. 'O') THEN
      IF (MOD (ROW,4) .EQ. 0) THEN
        SEX = 'X'
        GRADE = 'FD'
      ELSE IF (MOD (ROW,4) .EQ. 3) THEN
        SEX = 'M'
        GRADE = 'FD'
      ELSE IF (MOD (ROW,4) .EQ. 2) THEN
        SEX = 'X'
        GRADE = 'CO'
      ELSE
        SEX = 'M'
        GRADE = 'CO'
      ENDIF
      * Enlisted
      ELSE
      IF (MOD (ROW,4) .EQ. 0) THEN
        SEX = 'X'
        GRADE = '59'
      ELSE IF (MOD (ROW,4) .EQ. 3) THEN
        SEX = 'M'
        GRADE = '59'
      ELSE IF (MOD (ROW,4) .EQ. 2) THEN
        SEX = 'X'
        GRADE = '14'
      ELSE
        SEX = 'M'
        GRADE = '14'
      ENDIF
      END IF

      * Exit subroutine
      RETURN
      END

```

```

      SUBROUTINE OROW (ARRAY1, ARRAY2, MOSNM, NUMP, PER, SKILL, SX,
$      TIME, BT, NBT)
*****
*
* SUBROUTINE OROW
* This subroutine finds the correct 4 rows in the officer matrix, and
* passes them to 'modifyomat' to update the battle and non-battle
* matrix quantities.
*
*****

```

```

      CHARACTER SKILL, SX
      INTEGER NUMP, MOSNM, PER, TIME, BT, NBT, ARRAY1, ARRAY2
      DIMENSION ARRAY1 (NUMP, PER), ARRAY2 (NUMP, PER)

```

```

* OAD
  IF (MOSNM .EQ. 14) THEN
    CALL MODIFYOMAT (ARRAY1, ARRAY2, NUMP, PER, SKILL, SX, TIME,
$    BT, NBT, 1, 2, 3, 4)
* OAR
  ELSE IF (MOSNM .EQ. 12) THEN
    CALL MODIFYOMAT (ARRAY1, ARRAY2, NUMP, PER, SKILL, SX, TIME,
$    BT, NBT, 5, 6, 7, 8)
* OAV
  ELSE IF (MOSNM .EQ. 15) THEN
    CALL MODIFYOMAT (ARRAY1, ARRAY2, NUMP, PER, SKILL, SX, TIME,
$    BT, NBT, 9, 10, 11, 12)
* OCE
  ELSE IF (MOSNM .EQ. 21) THEN
    CALL MODIFYOMAT (ARRAY1, ARRAY2, NUMP, PER, SKILL, SX, TIME,
$    BT, NBT, 13, 14, 15, 16)
* OCM
  ELSE IF (MOSNM .EQ. 74) THEN
    CALL MODIFYOMAT (ARRAY1, ARRAY2, NUMP, PER, SKILL, SX, TIME,
$    BT, NBT, 17, 18, 19, 20)
* OFA
  ELSE IF (MOSNM .EQ. 13) THEN
    CALL MODIFYOMAT (ARRAY1, ARRAY2, NUMP, PER, SKILL, SX, TIME,
$    BT, NBT, 25, 26, 27, 28)
* OIN
  ELSE IF (MOSNM .EQ. 11 .OR. MOSNM .EQ. 18) THEN
    CALL MODIFYOMAT (ARRAY1, ARRAY2, NUMP, PER, SKILL, SX, TIME,
$    BT, NBT, 29, 30, 31, 32)
* OMC
  ELSE IF ((MOSNM .GE. 60 .AND. MOSNM .LE. 68) .OR.
$    MOSNM .EQ. 74) THEN
    CALL MODIFYOMAT (ARRAY1, ARRAY2, NUMP, PER, SKILL, SX, TIME,
$    BT, NBT, 33, 34, 35, 36)
* OMI
  ELSE IF (MOSNM .EQ. 35) THEN
    CALL MODIFYOMAT (ARRAY1, ARRAY2, NUMP, PER, SKILL, SX, TIME,
$    BT, NBT, 37, 38, 39, 40)

```

```

* OMP
  ELSE IF (MOSNM .EQ. 31) THEN
    CALL MODIFYOMAT (ARRAY1, ARRAY2, NUMP, PER, SKILL, SX, TIME,
$                      BT, NBT, 41, 42, 43, 44)
* OOD
  ELSE IF (MOSNM .EQ. 91) THEN
    CALL MODIFYOMAT (ARRAY1, ARRAY2, NUMP, PER, SKILL, SX, TIME,
$                      BT, NBT, 45, 46, 47, 48)
* OQM
  ELSE IF (MOSNM .EQ. 92) THEN
    CALL MODIFYOMAT (ARRAY1, ARRAY2, NUMP, PER, SKILL, SX, TIME,
$                      BT, NBT, 49, 50, 51, 52)
* OSC
  ELSE IF (MOSNM .EQ. 25) THEN
    CALL MODIFYOMAT (ARRAY1, ARRAY2, NUMP, PER, SKILL, SX, TIME,
$                      BT, NBT, 53, 54, 55, 56)
* OTC
  ELSE IF (MOSNM .EQ. 88 .OR. MOSNM .EQ. 95) THEN
    CALL MODIFYOMAT (ARRAY1, ARRAY2, NUMP, PER, SKILL, SX, TIME,
$                      BT, NBT, 57, 58, 59, 60)
* OCS
  ELSE
    CALL MODIFYOMAT (ARRAY1, ARRAY2, NUMP, PER, SKILL, SX, TIME,
$                      BT, NBT, 21, 22, 23, 24)
  END IF

* Exit subroutine
  RETURN
  END

```

```

      SUBROUTINE MODIFYEMAT (ARRAY1, ARRAY2, NUMP, PER, SKILL, SX,
$      TIME, BT, NBT, R1, R2, R3, R4)
*****
*
* SUBROUTINE MODIFYEMAT
* This subroutine determines the skill and sex of the officer, and
* updated the matrix rows accordingly.
*
*****

      CHARACTER SKILL, SX
      INTEGER NUMP, PER, TIME, BT, NBT, R1, R2, R3, R4, ARRAY1, ARRAY2
      DIMENSION ARRAY1 (NUMP, PER), ARRAY2 (NUMP, PER)

      IF (SKILL .EQ. '*') THEN
      IF (SX .EQ. '*' .OR. SX .EQ. 'X') THEN
        ARRAY1 (R2, TIME) = ARRAY1 (R2, TIME) + NINT (BT * 0.7)
        ARRAY1 (R4, TIME) = ARRAY1 (R4, TIME) + NINT (BT * 0.3)
        ARRAY2 (R2, TIME) = ARRAY2 (R2, TIME) + NINT (NBT * 0.7)
        ARRAY2 (R4, TIME) = ARRAY2 (R4, TIME) + NINT (NBT * 0.3)
      ELSE
        ARRAY1 (R1, TIME) = ARRAY1 (R1, TIME) + NINT (BT * 0.7)
        ARRAY1 (R3, TIME) = ARRAY1 (R3, TIME) + NINT (BT * 0.3)
        ARRAY2 (R1, TIME) = ARRAY2 (R1, TIME) + NINT (NBT * 0.7)
        ARRAY2 (R3, TIME) = ARRAY2 (R3, TIME) + NINT (NBT * 0.3)
      ENDIF
      ELSE IF (SKILL .EQ. '1') THEN
      IF (SX .EQ. '*' .OR. SX .EQ. 'X') THEN
        ARRAY1 (R2, TIME) = ARRAY1 (R2, TIME) + BT
        ARRAY2 (R2, TIME) = ARRAY2 (R2, TIME) + NBT
      ELSE
        ARRAY1 (R1, TIME) = ARRAY1 (R1, TIME) + BT
        ARRAY2 (R1, TIME) = ARRAY2 (R1, TIME) + NBT
      ENDIF
      ELSE
      IF (SX .EQ. '*' .OR. SX .EQ. 'X') THEN
        ARRAY1 (R4, TIME) = ARRAY1 (R4, TIME) + BT
        ARRAY2 (R4, TIME) = ARRAY2 (R4, TIME) + NBT
      ELSE
        ARRAY1 (R3, TIME) = ARRAY1 (R3, TIME) + BT
        ARRAY2 (R3, TIME) = ARRAY2 (R3, TIME) + NBT
      ENDIF
      ENDIF

      * Exit subroutine
      RETURN
      END

```

```

SUBROUTINE MODIFYOMAT (ARRAY1, ARRAY2, NUMP, PER, SKILL, SX,
$                      TIME, BT, NBT, R1, R2, R3, R4)
*****
*
* SUBROUTINE MODIFYOMAT
* This subroutine determines the skill and sex of the officer, and
* updated the matrix rows accordingly.
*
*****

CHARACTER SKILL, SX
INTEGER NUMP, PER, TIME, BT, NBT, R1, R2, R3, R4, ARRAY1, ARRAY2
DIMENSION ARRAY1 (NUMP, PER), ARRAY2 (NUMP, PER)

IF (SKILL .EQ. '*') THEN
IF (SX .EQ. '*' .OR. SX .EQ. 'X') THEN
ARRAY1 (R2, TIME) = ARRAY1 (R2, TIME) + NINT (BT * 0.7)
ARRAY1 (R4, TIME) = ARRAY1 (R4, TIME) + NINT (BT * 0.3)
ARRAY2 (R2, TIME) = ARRAY2 (R2, TIME) + NINT (NBT * 0.7)
ARRAY2 (R4, TIME) = ARRAY2 (R4, TIME) + NINT (NBT * 0.3)
ELSE
ARRAY1 (R1, TIME) = ARRAY1 (R1, TIME) + NINT (BT * 0.7)
ARRAY1 (R3, TIME) = ARRAY1 (R3, TIME) + NINT (BT * 0.3)
ARRAY2 (R1, TIME) = ARRAY2 (R1, TIME) + NINT (NBT * 0.7)
ARRAY2 (R3, TIME) = ARRAY2 (R3, TIME) + NINT (NBT * 0.3)
ENDIF
ELSE IF (SKILL .EQ. '3' .OR. SKILL .EQ. '2' .OR.
$ SKILL .EQ. '1') THEN
IF (SX .EQ. '*' .OR. SX .EQ. 'X') THEN
ARRAY1 (R2, TIME) = ARRAY1 (R2, TIME) + BT
ARRAY2 (R2, TIME) = ARRAY2 (R2, TIME) + NBT
ELSE
ARRAY1 (R1, TIME) = ARRAY1 (R1, TIME) + BT
ARRAY2 (R1, TIME) = ARRAY2 (R1, TIME) + NBT
ENDIF
ELSE
IF (SX .EQ. '*' .OR. SX .EQ. 'X') THEN
ARRAY1 (R4, TIME) = ARRAY1 (R4, TIME) + BT
ARRAY2 (R4, TIME) = ARRAY2 (R4, TIME) + NBT
ELSE
ARRAY1 (R3, TIME) = ARRAY1 (R3, TIME) + BT
ARRAY2 (R3, TIME) = ARRAY2 (R3, TIME) + NBT
ENDIF
ENDIF
ENDIF

* Exit subroutine
RETURN
END

```

```

      SUBROUTINE PRINTMATRIX (OM1, OM2, ONUM, WM1, WM2, WNUM,
$                               EM1, EM2, ENUM, PER)
*****
*
* SUBROUTINE PRINTMATRIX
* This subroutine prints the officer/warrant officer/enlisted matrix
* info for battle/non-battle casualties.
*
*****

      INTEGER ONUM, WNUM, ENUM, PER
      INTEGER OM1, OM2, WM1, WM2, EM1, EM2
      DIMENSION OM1 (ONUM, PER), OM2 (ONUM, PER)
      DIMENSION WM1 (WNUM, PER), WM2 (WNUM, PER)
      DIMENSION EM1 (ENUM, PER), EM2 (ENUM, PER)
      INTEGER CNT1, CNT2
      CHARACTER HEAD0*47, HEAD1*46, HEAD2*48, CATBRG*3, GRADE*2, SEX

* Print Header info.
      HEAD0 = '
      HEAD1 = ' TP   CATBRGD   S   BATTLE   NON-BATTLE   TOTAL'
      HEAD2 = '-----STR-----STR-----STR'
*
      BXXBBBBXXXXXBBBBXBBBBXXXXXBBBBBXXXXXBBBBBXXXXXX
      WRITE(2,500) HEAD0, HEAD1, HEAD2

* The header format
      500 FORMAT(1X,A47/1X,A46,/1X,A48)

* Go thru matrix and print qty if qty != 0
      DO 560, CNT1 = 1, PER
* Print officers
      DO 550, CNT2 = 1, ONUM
      IF (OM1 (CNT2, CNT1) .NE. 0 .OR.
$          OM2 (CNT2, CNT1) .NE. 0) THEN
      IF (CNT2 .GE. 1 .AND. CNT2 .LE. 4) THEN
      CATBRG = 'OAD'
      ELSE IF (CNT2 .GE. 5 .AND. CNT2 .LE. 8) THEN
      CATBRG = 'OAR'
      ELSE IF (CNT2 .GE. 9 .AND. CNT2 .LE. 12) THEN
      CATBRG = 'OAV'
      ELSE IF (CNT2 .GE. 13 .AND. CNT2 .LE. 16) THEN
      CATBRG = 'OCE'
      ELSE IF (CNT2 .GE. 17 .AND. CNT2 .LE. 20) THEN
      CATBRG = 'OCM'
      ELSE IF (CNT2 .GE. 21 .AND. CNT2 .LE. 24) THEN
      CATBRG = 'OCS'
      ELSE IF (CNT2 .GE. 25 .AND. CNT2 .LE. 28) THEN
      CATBRG = 'OFA'
      ELSE IF (CNT2 .GE. 29 .AND. CNT2 .LE. 32) THEN
      CATBRG = 'OIN'
      ELSE IF (CNT2 .GE. 33 .AND. CNT2 .LE. 36) THEN
      CATBRG = 'OMC'

```

```

ELSE IF (CNT2 .GE. 37 .AND. CNT2 .LE. 40) THEN
CATBRG = 'OMI'
ELSE IF (CNT2 .GE. 41 .AND. CNT2 .LE. 44) THEN
CATBRG = 'OMP'
ELSE IF (CNT2 .GE. 45 .AND. CNT2 .LE. 48) THEN
CATBRG = 'OOD'
ELSE IF (CNT2 .GE. 49 .AND. CNT2 .LE. 52) THEN
CATBRG = 'OQM'
ELSE IF (CNT2 .GE. 53 .AND. CNT2 .LE. 56) THEN
CATBRG = 'OSC'
ELSE
CATBRG = 'OTC'
ENDIF

```

```

CALL GRADESEX ('O', CNT2, GRADE, SEX)

```

* Determine appropriate write format type; if cnt1 <= 9 you must add
* a 0 to the front

```

IF (CNT1 .LE. 9) THEN
WRITE(2,570) CNT1, CATBRG, GRADE, SEX, OM1 (CNT2,CNT1),
$ OM2 (CNT2,CNT1),
$ OM1 (CNT2,CNT1) + OM2 (CNT2,CNT1)
ELSE
WRITE(2,580) CNT1, CATBRG, GRADE, SEX, OM1 (CNT2,CNT1),
$ OM2 (CNT2,CNT1),
$ OM1 (CNT2,CNT1) + OM2 (CNT2,CNT1)
ENDIF
ENDIF
550 CONTINUE

```

* Print warrant officers

```

GRADE = 'WW'
DO 553, CNT2 = 1, WNUM
IF (WM1 (CNT2, CNT1) .NE. 0 .OR.
$ WM2 (CNT2, CNT1) .NE. 0) THEN
IF (CNT2 .EQ. 1) THEN
CATBRG = 'WCB'
SEX = 'M'
ELSE IF (CNT2 .EQ. 2) THEN
CATBRG = 'WCB'
SEX = 'X'
ELSE IF (CNT2 .EQ. 3) THEN
CATBRG = 'WCC'
SEX = 'M'
ELSE IF (CNT2 .EQ. 4) THEN
CATBRG = 'WCC'
SEX = 'X'
ELSE IF (CNT2 .EQ. 5) THEN
CATBRG = 'WCS'
SEX = 'M'
ELSE
CATBRG = 'WCS'

```

SEX = 'X'
END IF

- * Determine appropriate write format type; if cnt1 <= 9 you must add
- * a 0 to the front

```
IF (CNT1 .LE. 9) THEN
WRITE(2,570) CNT1, CATBRG, GRADE, SEX, WM1 (CNT2,CNT1),
$                               WM2 (CNT2,CNT1),
$                               WM1 (CNT2,CNT1) + WM2 (CNT2,CNT1)
ELSE
WRITE(2,580) CNT1, CATBRG, GRADE, SEX, WM1 (CNT2,CNT1),
$                               WM2 (CNT2,CNT1),
$                               WM1 (CNT2,CNT1) + WM2 (CNT2,CNT1)
ENDIF
ENDIF
553 CONTINUE
```

- * Print enlisted

```
DO 555, CNT2 = 1, ENUM
IF (EM1 (CNT2, CNT1) .NE. 0 .OR.
$   EM2 (CNT2, CNT1) .NE. 0) THEN
IF (CNT2 .GE. 1 .AND. CNT2 .LE. 4) THEN
CATBRG = 'EAD'
ELSE IF (CNT2 .GE. 5 .AND. CNT2 .LE. 8) THEN
CATBRG = 'EAR'
ELSE IF (CNT2 .GE. 9 .AND. CNT2 .LE. 12) THEN
CATBRG = 'EAV'
ELSE IF (CNT2 .GE. 13 .AND. CNT2 .LE. 16) THEN
CATBRG = 'ECE'
ELSE IF (CNT2 .GE. 17 .AND. CNT2 .LE. 20) THEN
CATBRG = 'ECM'
ELSE IF (CNT2 .GE. 21 .AND. CNT2 .LE. 24) THEN
CATBRG = 'ECS'
ELSE IF (CNT2 .GE. 25 .AND. CNT2 .LE. 28) THEN
CATBRG = 'EFA'
ELSE IF (CNT2 .GE. 29 .AND. CNT2 .LE. 32) THEN
CATBRG = 'EIN'
ELSE IF (CNT2 .GE. 33 .AND. CNT2 .LE. 36) THEN
CATBRG = 'EMC'
ELSE IF (CNT2 .GE. 37 .AND. CNT2 .LE. 40) THEN
CATBRG = 'EMI'
ELSE IF (CNT2 .GE. 41 .AND. CNT2 .LE. 44) THEN
CATBRG = 'EMM'
ELSE IF (CNT2 .GE. 45 .AND. CNT2 .LE. 48) THEN
CATBRG = 'EMP'
ELSE IF (CNT2 .GE. 49 .AND. CNT2 .LE. 52) THEN
CATBRG = 'EOD'
ELSE IF (CNT2 .GE. 53 .AND. CNT2 .LE. 56) THEN
CATBRG = 'EQM'
ELSE IF (CNT2 .GE. 57 .AND. CNT2 .LE. 60) THEN
CATBRG = 'ESC'
ELSE IF (CNT2 .GE. 61 .AND. CNT2 .LE. 64) THEN
```



```

CATBRG = 'ESM'
ELSE
CATBRG = 'ETC'
ENDIF

```

```

CALL GRADESEX ('E', CNT2, GRADE, SEX)

```

* Determine appropriate write format type; if cnt1 <= 9 you must add
 * a 0 to the front

```

      IF (CNT1 .LE. 9) THEN
        WRITE(2,570) CNT1, CATBRG, GRADE, SEX, EM1 (CNT2,CNT1),
$                                     EM2 (CNT2,CNT1),
$                                     EM1 (CNT2,CNT1) + EM2 (CNT2,CNT1)
      ELSE
        WRITE(2,580) CNT1, CATBRG, GRADE, SEX, EM1 (CNT2,CNT1),
$                                     EM2 (CNT2,CNT1),
$                                     EM1 (CNT2,CNT1) + EM2 (CNT2,CNT1)
      ENDIF
    ENDIF
555    CONTINUE
560    CONTINUE

```

* The write format

```

570  FORMAT (2X,'0',I1,4X,A3,A2,4X,A1,3X,I6,4X,I6,4X,I6)
580  FORMAT (2X,I2,4X,A3,A2,4X,A1,3X,I6,4X,I6,4X,I6)

```

* Exit subroutine

```

  RETURN
  END

```

```

      SUBROUTINE WROW (ARRAY1, ARRAY2, MOSNM, NUMP, PER, SX,
$      TIME, BT, NBT)
*****
*
* SUBROUTINE WROW
* This subroutine finds the correct 4 rows in the warrant officer
* matrix, and updates the battle and non-battle matrix quantities.
*
*****

      CHARACTER SX
      INTEGER NUMP, MOSNM, PER, TIME, BT, NBT, ARRAY1, ARRAY2
      DIMENSION ARRAY1 (NUMP, PER), ARRAY2 (NUMP, PER)

* WCB
      IF (MOSNM .GE. 10 .AND. MOSNM .LE. 19) THEN
      IF (SX .EQ. '*' .OR. SX .EQ. 'X') THEN
          ARRAY1 (2, TIME) = ARRAY1 (2,TIME) + BT
          ARRAY2 (2, TIME) = ARRAY2 (2,TIME) + NBT
      ELSE
          ARRAY1 (1, TIME) = ARRAY1 (1,TIME) + BT
          ARRAY2 (1, TIME) = ARRAY2 (1,TIME) + NBT
      ENDIF

* WCC
      ELSE IF ((MOSNM .GE. 20 .AND. MOSNM .LE. 39) .OR.
$      (MOSNM .GE. 60 .AND. MOSNM .LE. 69)) THEN
      IF (SX .EQ. '*' .OR. SX .EQ. 'X') THEN
          ARRAY1 (6, TIME) = ARRAY1 (6,TIME) + BT
          ARRAY2 (6, TIME) = ARRAY2 (6,TIME) + NBT
      ELSE
          ARRAY1 (5, TIME) = ARRAY1 (5,TIME) + BT
          ARRAY2 (5, TIME) = ARRAY2 (5,TIME) + NBT
      ENDIF

* WCS
      ELSE
      IF (SX .EQ. '*' .OR. SX .EQ. 'X') THEN
          ARRAY1 (4, TIME) = ARRAY1 (4,TIME) + BT
          ARRAY2 (4, TIME) = ARRAY2 (4,TIME) + NBT
      ELSE
          ARRAY1 (3, TIME) = ARRAY1 (3,TIME) + BT
          ARRAY2 (3, TIME) = ARRAY2 (3,TIME) + NBT
      ENDIF
      END IF

* Exit subroutine
      RETURN
      END

```

4.6 MOBTNGBS (MOBARPRINT) MODULE

4.6.1 GENERAL

The MOBTNGBS (Mobilization Training Base) module converts the output file generated from the MOBARPRINT program produced for HQDA, ODCSPER to standard WARPAM format. The incoming file is supplied by the support contractor on a single 5 1/4" low-density floppy disk in ASCII format. This module requires that the Branch.Tbl be present for processing. The output from this conversion is an asset file of skill level one training base assets which receive the asset code "TRN". The processing flow through the MOBTNGBS module is shown below at Figure 6.

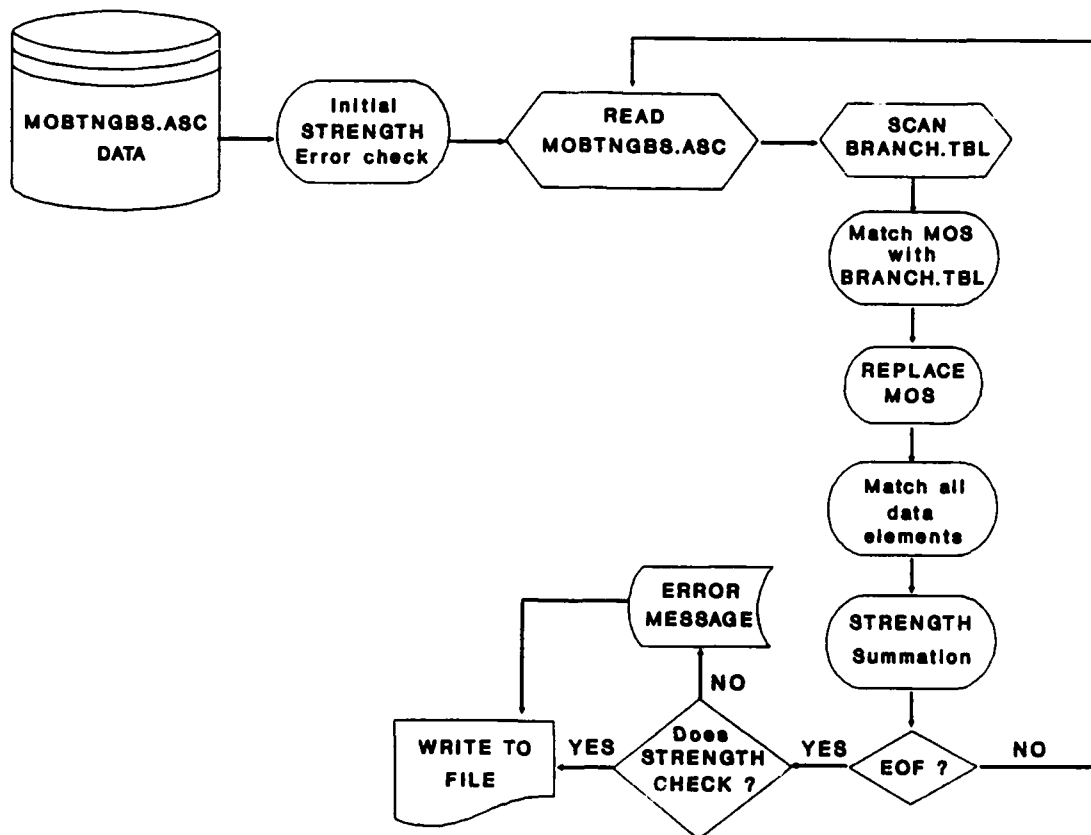


FIGURE 6: MOBTNGBS FILE CONVERSION

4.6.2 MOBTNGBS FORTRAN PROGRAMS

```
*****
*
* PROGRAMMER : JOHN A. TENSCHAW
* COMPANY   : SAIC
* ADDRESS    : 1710 GOODRIDGE DR. MS-T-1-7-2, MCLEAN, VA. 22102
* PHONE      : 703-734-5584
* DATE       : 20 APRIL 90
*
*****
```

```
*****
*
* Program MOBTNGBS
* This program reads the MOBTNGBS file and condenses the info.
* according to mos types and prints out the results per time period
*
*****
```

```
* Define variables
* NUMMOS = the num. of diff. mos types; the num of rows (ie. EAD)
* PERIOD = the 18 time periods; the num of cols
* MATRIX = a NUMMOS x PERIOD array representing mos and time periods
* ROW = the row index
* LABEL = first 2 MOS numbers (1st two chars on input line)
* TAG = the third mos number (3rd char on input line)
* TEMP = extra char that must be read in but not used
*        (4th char on input line)
* V1...V18 = the 18 time period qtys. and total listed separate
*           (items 5 - 112 on input line)
* CTR1, CTR2 = integer counters
* COUNT = the num of lines of input read in
```

```
* VIEW OF MATRIX :
*
*           TIME PERIODS
*           1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18
* A1
* EAD
* EAR
* EAV      * EACH MATRIX ELEMENT IS A SIX DIGIT INTEGER
* M  ECE
* O  ECM      * A1 = LINE 1 OF INPUT; INPUTTED TIME PERIOD TOTALS
* S  ECS      * Z1 = PROGRAM SUMMED TIME PERIOD TOTALS
*     EFA
*     EIN
*     EMC
*     EMI
*     EMM
*     EMP
```

```

*   EOD
*   EQM
*   ESC
*   ESM
*   ETC
*   Z1

```

```

INTEGER NUMMOS, PERIOD
PARAMETER (NUMMOS = 19, PERIOD = 18)
INTEGER MATRIX
DIMENSION MATRIX (NUMMOS, PERIOD)
INTEGER ROW
CHARACTER TAG, TEMP
CHARACTER*4 FIRST
INTEGER LABEL, V1, V2, V3, V4, V5, V6, V7, V8, V9, V10
INTEGER V11, V12, V13, V14, V15, V16, V17, V18
INTEGER CTR1, CTR2, COUNT
LOGICAL THERE

```

```

*****

```

```

* Initialize the matrix to 0
  DO 6, CTR1 = 1, NUMMOS
    DO 5, CTR2 = 1, PERIOD
      MATRIX (CTR1, CTR2) = 0
    5    CONTINUE
  6    CONTINUE

* If input file does not exist, stop
  INQUIRE (FILE = '/home/warpam/MOBTNGBS.DAT',
$    EXIST = THERE)
  IF (.NOT. THERE) THEN
    PRINT *, 'ERROR - MOBTNGBS.DAT DOES NOT EXIST'
    GO TO 110
  END IF

* If output file exists, delete it
  INQUIRE (FILE = '/home/warpam/MOBTNGBS.OUT',
$    EXIST = THERE)
  IF (THERE) THEN
    OPEN(UNIT = 2, FILE = '/home/warpam/MOBTNGBS.OUT',
$    STATUS = 'OLD')
    CLOSE (2, STATUS = 'DELETE')
  END IF

* Open the input and output files
  OPEN(UNIT = 1, FILE = '/home/warpam/MOBTNGBS.DAT',
$    STATUS = 'OLD')
  OPEN(UNIT = 2, FILE = '/home/warpam/MOBTNGBS.OUT',
$    STATUS = 'NEW')

```

```

* initialize count = 1
  COUNT = 1

* Read input line and check for errors
10  IF (COUNT .EQ. 1) THEN
    READ(1, 40, END = 80) FIRST, V1, V2, V3, V4, V5,
    $      V6,V7,V8,V9,V10,V11,V12,V13,V14,V15,V16,V17,V18
    ELSE
    READ(1, 50, END = 80) LABEL, TAG, TEMP, V1, V2, V3, V4, V5,
    $      V6,V7,V8,V9,V10,V11,V12,V13,V14,V15,V16,V17,V18
    ENDIF

40  FORMAT(A4,18(I6))
50  FORMAT(I2,2(A),18(I6))

* If count = 1, you have the input total line, so just modify row 1,
* otherwise find correct row.
  ROW = 1
  IF (COUNT .NE. 1) THEN
    CALL FINDROW (LABEL, TAG, ROW)
  ENDIF

* Modify that row in MATRIX
  CALL MATRIXADD (MATRIX, NUMMOS, PERIOD, ROW, V1, V2, V3, V4, V5, V6, V7, V8,
  $      V9, V10, V11, V12, V13, V14, V15, V16, V17, V18)

* Increment count +1
  COUNT = COUNT + 1

* Go back to start of loop
  GO TO 10

80  CALL CHECKSUMS (MATRIX, NUMMOS, PERIOD)
    CALL PRINTMATRIX (MATRIX, NUMMOS, PERIOD)

* Close files and exit program
100 CLOSE(1, STATUS = 'KEEP')
    CLOSE(2, STATUS = 'KEEP')
110 END

```

***** SUBROUTINES - IN ALPHABETICAL ORDER *****

SUBROUTINE CHECKSUMS (ARRAY, MOS, PER)

```
*****
*
* SUBROUTINE CHECKSUMS
* This subroutine checks to make sure that the sum for each time
* period matches the sums on the first input line, else an error
* message will appear for each mismatch.
*
*****
```

```
INTEGER MOS, PER
INTEGER ARRAY
DIMENSION ARRAY (MOS, PER)
INTEGER CNT
```

```
DO 150, CNT = 1, PER
IF (ARRAY (1, CNT) .NE. ARRAY (MOS, CNT)) THEN
WRITE (*,130) CNT
```

```
130 FORMAT (1X, 'ERROR - TIME PERIOD ',I2,' TOTALS DON'T MATCH')
ENDIF
150 CONTINUE
```

```
* Exit subroutine
RETURN
END
```

```

SUBROUTINE FINDROW (LABL, LETTER, RW)
*****
*
* SUBROUTINE FINDROW
* This subroutine finds the correct row num in MATRIX in which to
* add the new time period qtys. to. ROW will then be the desired row
* number.
*
*****

```

```

INTEGER LABL, RW
CHARACTER LETTER

```

```

IF (((LABL .EQ. 41) .AND. (LETTER .EQ. 'B')) .OR.
$ ((LABL .EQ. 52) .AND. ((LETTER .EQ. 'E') .OR.
$ (LETTER .EQ. 'G')))) THEN
RW = 4
ELSE IF (((LABL .EQ. 21) .AND. (LETTER .EQ. 'G')) .OR.
$ ((LABL .EQ. 82) .AND. (LETTER .EQ. 'C')) .OR.
$ ((LABL .EQ. 93) .AND. (LETTER .EQ. 'F')))) THEN
RW = 8
ELSE IF (((LABL .EQ. 71) .AND. (LETTER .EQ. 'G')) .OR.
$ ((LABL .EQ. 76) .AND. (LETTER .EQ. 'J')) .OR.
$ ((LABL .EQ. 94) .AND. (LETTER .EQ. 'F')))) THEN
RW = 10
ELSE IF (((LABL .EQ. 25) .AND. (LETTER .EQ. 'L')) .OR.
$ ((LABL .EQ. 46) .AND. (LETTER .EQ. 'N')))) THEN
RW = 12
ELSE IF ((LABL .EQ. 62) .AND. (LETTER .EQ. 'B')) THEN
RW = 14
ELSE IF ((LABL .EQ. 35) .AND. (LETTER .EQ. 'H')) THEN
RW = 15
ELSE IF (LABL .EQ. 16) THEN
RW = 2
ELSE IF (LABL .EQ. 19) THEN
RW = 3
ELSE IF ((LABL .EQ. 66) .OR. (LABL .EQ. 67) .OR.
$ (LABL .EQ. 68) .OR. (LABL .EQ. 93)) THEN
RW = 4
ELSE IF ((LABL .EQ. 12) .OR. (LABL .EQ. 51) .OR.
$ (LABL .EQ. 62) .OR. (LABL .EQ. 81) .OR.
$ (LABL .EQ. 82)) THEN
RW = 5
ELSE IF (LABL .EQ. 54) THEN
RW = 6
ELSE IF ((LABL .EQ. 13) .OR. (LABL .EQ. 15) .OR.
$ (LABL .EQ. 17)) THEN
RW = 8
ELSE IF ((LABL .EQ. 11) .OR. (LABL .EQ. 18)) THEN
RW = 9
ELSE IF ((LABL .EQ. 1) .OR. (LABL .EQ. 35) .OR.
$ (LABL .EQ. 42) .OR. (LABL .EQ. 91) .OR.

```



```

$      (LABL .EQ. 92)) THEN
  RW = 10
  ELSE IF ((LABL .EQ. 5) .OR. (LABL .EQ. 96) .OR.
$      (LABL .EQ. 97) .OR. (LABL .EQ. 98)) THEN
  RW = 11
  ELSE IF ((LABL .EQ. 21) .OR. (LABL .EQ. 24) .OR.
$      (LABL .EQ. 26) .OR. (LABL .EQ. 27)) THEN
  RW = 12
  ELSE IF (LABL .EQ. 95) THEN
  RW = 13
  ELSE IF ((LABL .EQ. 41) .OR. (LABL .EQ. 44) .OR.
$      (LABL .EQ. 45) .OR. (LABL .EQ. 52) .OR.
$      (LABL .EQ. 63)) THEN
  RW = 14
  ELSE IF ((LABL .EQ. 43) .OR. (LABL .EQ. 55) .OR.
$      (LABL .EQ. 57) .OR. (LABL .EQ. 76) .OR.
$      (LABL .EQ. 77) .OR. (LABL .EQ. 94)) THEN
  RW = 15
  ELSE IF ((LABL .EQ. 29) .OR. (LABL .EQ. 31) .OR.
$      (LABL .EQ. 36) .OR. (LABL .EQ. 72)) THEN
  RW = 16
  ELSE IF ((LABL .EQ. 33) .OR. (LABL .EQ. 39)) THEN
  RW = 17
  ELSE IF (LABL .EQ. 88) THEN
  RW = 18
  ELSE
  RW = 7
  ENDIF

```

```

* Exit subroutine
  RETURN
  END

```

```

SUBROUTINE MATRIXADD (ARRAY,MOS,PER,NUM,A,B,C,D,E,F,G,H,I,J,K,L,
$                      M,N,O,P,Q,R)
*****
*
* SUBROUTINE MATRIXADD
* This subroutine adds the input time period qtys (V1...V18) to
* to index time periods 1...18 of the correct row in MATRIX
*
*****

```

```

INTEGER MOS, PER
INTEGER ARRAY
DIMENSION ARRAY (MOS, PER)
INTEGER NUM,A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R

```

```

* modify the mos row
  ARRAY (NUM, 1) = ARRAY (NUM, 1) + A
  ARRAY (NUM, 2) = ARRAY (NUM, 2) + B
  ARRAY (NUM, 3) = ARRAY (NUM, 3) + C
  ARRAY (NUM, 4) = ARRAY (NUM, 4) + D
  ARRAY (NUM, 5) = ARRAY (NUM, 5) + E
  ARRAY (NUM, 6) = ARRAY (NUM, 6) + F
  ARRAY (NUM, 7) = ARRAY (NUM, 7) + G
  ARRAY (NUM, 8) = ARRAY (NUM, 8) + H
  ARRAY (NUM, 9) = ARRAY (NUM, 9) + I
  ARRAY (NUM, 10) = ARRAY (NUM, 10) + J
  ARRAY (NUM, 11) = ARRAY (NUM, 11) + K
  ARRAY (NUM, 12) = ARRAY (NUM, 12) + L
  ARRAY (NUM, 13) = ARRAY (NUM, 13) + M
  ARRAY (NUM, 14) = ARRAY (NUM, 14) + N
  ARRAY (NUM, 15) = ARRAY (NUM, 15) + O
  ARRAY (NUM, 16) = ARRAY (NUM, 16) + P
  ARRAY (NUM, 17) = ARRAY (NUM, 17) + Q
  ARRAY (NUM, 18) = ARRAY (NUM, 18) + R

```

```

* modify the summed totals, the last row, if row != 1
  IF (NUM .NE. 1) THEN
    ARRAY (MOS, 1) = ARRAY (MOS, 1) + A
    ARRAY (MOS, 2) = ARRAY (MOS, 2) + B
    ARRAY (MOS, 3) = ARRAY (MOS, 3) + C
    ARRAY (MOS, 4) = ARRAY (MOS, 4) + D
    ARRAY (MOS, 5) = ARRAY (MOS, 5) + E
    ARRAY (MOS, 6) = ARRAY (MOS, 6) + F
    ARRAY (MOS, 7) = ARRAY (MOS, 7) + G
    ARRAY (MOS, 8) = ARRAY (MOS, 8) + H
    ARRAY (MOS, 9) = ARRAY (MOS, 9) + I
    ARRAY (MOS, 10) = ARRAY (MOS, 10) + J
    ARRAY (MOS, 11) = ARRAY (MOS, 11) + K
    ARRAY (MOS, 12) = ARRAY (MOS, 12) + L
    ARRAY (MOS, 13) = ARRAY (MOS, 13) + M
    ARRAY (MOS, 14) = ARRAY (MOS, 14) + N
    ARRAY (MOS, 15) = ARRAY (MOS, 15) + O
  
```

```
ARRAY (MOS, 16) = ARRAY (MOS, 16) + P  
ARRAY (MOS, 17) = ARRAY (MOS, 17) + Q  
ARRAY (MOS, 18) = ARRAY (MOS, 18) + R  
ENDIF
```

```
* Exit subroutine  
RETURN  
END
```

```

      SUBROUTINE PRINTMATRIX (ARRAY, MOS, PER)
      *****
      *
      * SUBROUTINE PRINTMATRIX
      * This subroutine prints the entire matrix according to time periods.
      *
      *****

      INTEGER MOS, PER
      INTEGER ARRAY
      DIMENSION ARRAY (MOS, PER)
      INTEGER CNT1, CNT2
      CHARACTER HEAD1*30, HEAD2*31, CATBRG*5

      * Print Header info.
      HEAD1 = ' TP   CATBRGD   S   TYPE   STR'
      HEAD2 = '-----'
      *
      *          BXXBBBBBXXXXBBBBBBBBBBXXXXBBXXXXXX
      WRITE(2,500) HEAD1, HEAD2

      * The header format
      500  FORMAT(1X,A30,/1X,A31)

      * Go thru matrix and print qty if qty != 0
      DO 560, CNT1 = 1, PER
      DO 550, CNT2 = 2, MOS-1
      IF (ARRAY (CNT2, CNT1) .NE. 0) THEN
      IF (CNT2 .EQ. 2) THEN
      CATBRG = 'EAD14'
      ELSE IF (CNT2 .EQ. 3) THEN
      CATBRG = 'EAR14'
      ELSE IF (CNT2 .EQ. 4) THEN
      CATBRG = 'EAV14'
      ELSE IF (CNT2 .EQ. 5) THEN
      CATBRG = 'ECE14'
      ELSE IF (CNT2 .EQ. 6) THEN
      CATBRG = 'ECM14'
      ELSE IF (CNT2 .EQ. 7) THEN
      CATBRG = 'ECS14'
      ELSE IF (CNT2 .EQ. 8) THEN
      CATBRG = 'EFA14'
      ELSE IF (CNT2 .EQ. 9) THEN
      CATBRG = 'EIN14'
      ELSE IF (CNT2 .EQ. 10) THEN
      CATBRG = 'EMC14'
      ELSE IF (CNT2 .EQ. 11) THEN
      CATBRG = 'EMI14'
      ELSE IF (CNT2 .EQ. 12) THEN
      CATBRG = 'EMM14'
      ELSE IF (CNT2 .EQ. 13) THEN
      CATBRG = 'EMP14'

```

```

ELSE IF (CNT2 .EQ. 14) THEN
CATBRG = 'EOD14'
ELSE IF (CNT2 .EQ. 15) THEN
CATBRG = 'EQM14'
ELSE IF (CNT2 .EQ. 16) THEN
CATBRG = 'ESC14'
ELSE IF (CNT2 .EQ. 17) THEN
CATBRG = 'ESM14'
ELSE
CATBRG = 'ETC14'
ENDIF

```

* Determine appropriate write format type; if cnt1 <= 9 you must add
 * a 0 to the front

```

IF (CNT1 .LE. 9) THEN
WRITE(2,520) CNT1, CATBRG, ARRAY (CNT2, CNT1)
ELSE
WRITE(2,530) CNT1, CATBRG, ARRAY (CNT2, CNT1)
ENDIF

```

* The write format

```

520 FORMAT (2X,'0',I1,4X,A5,9X,'TRN',2X,I6)
530 FORMAT (2X,I2,4X,A5,9X,'TRN',2X,I6)
ENDIF

```

```

550     CONTINUE
560     CONTINUE

```

* Exit subroutine
 RETURN
 END

4.7 REQUIREMENT/ASSET GENERATOR (REQAST GEN) MODULE

4.7.1 GENERAL

This module merges the converted input files into a single data base, assigns branch priorities and a unique code number, and then sorts the file by this code number. The entire program is written in FORTRAN 77. The module utilizes two look-up tables, the WARPAM Branch Priority table and the Theater/Replacement Type table for branch priorities and code number development, respectively. This output file, titled REQAST.TBL is the basis of all subsequent WARPAM modeling and can be viewed by using the REQAST DBASE program. The processing flow for REQAST GEN module is shown at figure 7, below.

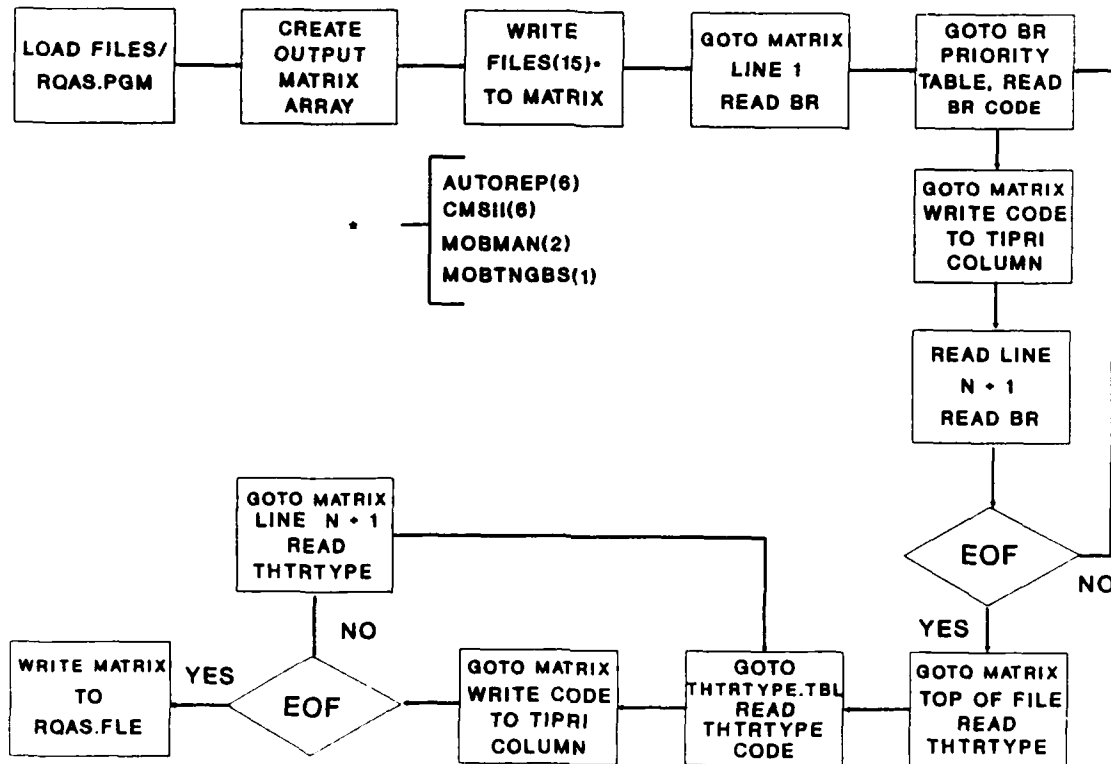


FIGURE 7: REQUIREMENTS/ASSETS GENERATOR PROCESSING

4.7.2 REQAST GENERATOR FORTRAN PROGRAMS

```
*****
*
* PROGRAMMER : JOHN A. TENSCHAW
* COMPANY : SAIC
* ADDRESS : 1710 GOODRIDGE DR. MS-T-1-7-2, MCLEAN, VA. 22102
* PHONE : 703-734-5584
* DATE : 5 MAY 90
*
*****
```

```
INTEGER MAXNUM, MAXTHR, MAXPTY
PARAMETER (MAXNUM = 37000, MAXTHR = 30, MAXPTY = 100)
CHARACTER CODE(MAXTHR)*4, ABBR(MAXTHR)*3, OUTSTR(MAXNUM)*24
CHARACTER PRIOR(MAXPTY)*5
INTEGER COUNT, NUM, LINE, STAT1
CHARACTER IN11*1, IN12*1, IN21*2, IN22*2, IN23*2, IN3*3, IN4*4, IN5*5
CHARACTER IN61*6, IN62*6, IN63*6
LOGICAL THERE
```

```
WRITE (6, 5)
5 FORMAT(//////////////////20X,'*****',
$/20X,'*****',//24X,
$'WARPAM REQ/ASSETS GENERATOR',//20X,
$'*****',/20X,
$'*****',////////20X
$'THE FOLLOWING FILES ARE NEEDED TO RUN :',/30X,
$'THTRTYPE.TBL',/30X,'WARPRI.TBL',/30X,'MAX.TBL',/30X,
$'MOBTNGBS.OUT',/30X,'CSMII.OUT',/30X,'AUTOREP.OUT',/30X,
$'MOBMREQ.OUT',/30X,'MOBMAST.OUT',//////////)

```

PAUSE

```
WRITE (6, 6)
6 FORMAT (//////////////////////////
$10X,'PLEASE WAIT ----- REQ/ASSETS GENERATOR RUNNING')
```

```
* If input file does not exist, stop
  INQUIRE (FILE = '/home/warpam/iofiles/THTRTYPE.TBL',EXIST = THERE)
  IF (.NOT. THERE) THEN
    PRINT *, ' ERROR - THTRTYPE.TBL DOES NOT EXIST'
    GO TO 500
  END IF

* Open the THTRTYPE.TBL file and read the info into CODE and ABBR
  OPEN(UNIT = 1,FILE = '/home/warpam/iofiles/THTRTYPE.TBL',
    $ STATUS = 'OLD', IOSTAT = STAT1)

* Initialize count = 1
  COUNT = 1
```

```

* Read input line and check for errors
20  READ(1, '(A4,1X,A3)', END = 25) IN4, IN3
    IF (COUNT .LE. MAXTHR) THEN
        CODE (COUNT) = IN4
        ABBR (COUNT) = IN3
        COUNT = COUNT + 1
        GO TO 20
    ELSE IF (STAT1 .LT. 0) THEN
        GO TO 25
    ELSE
        PRINT *, ' ERROR - TOO MANY INPUT LINES IN THTRTYPE.TBL'
        PRINT *, '          MODIFY THE MAXTHR VARIABLE IN REQST.F'
        PRINT *, '          ACCORDINGLY.'
        GO TO 500
    END IF

* Close THTRTYPE.TBL file
25  CLOSE(1, STATUS = 'KEEP')

* If input file does not exist, stop
    INQUIRE (FILE = '/home/warpam/iofiles/WARPRI.TBL', EXIST = THERE)
    IF (.NOT. THERE) THEN
        PRINT *, ' ERROR - WARPRI.TBL DOES NOT EXIST'
        GO TO 500
    END IF

* Open the WARPRI.TBL file and read the info into PRIOR
    OPEN(UNIT = 1, FILE = '/home/warpam/iofiles/WARPRI.TBL',
        $   STATUS = 'OLD', IOSTAT = STAT1)

* Initialize count = 1
    COUNT = 1

* Read input line and check for errors
30  READ(1, '(I2,1X,A5)', END = 35) NUM, IN5
    IF (COUNT .LE. MAXPTY) THEN
        PRIOR (COUNT) = IN5
        COUNT = COUNT + 1
        GO TO 30
    ELSE IF (STAT1 .LT. 0) THEN
        GO TO 35
    ELSE
        PRINT *, ' ERROR - TOO MANY INPUT LINES IN WARPRI.TBL'
        PRINT *, '          MODIFY THE MAXPTY VARIABLE IN REQST.F'
        PRINT *, '          ACCORDINGLY.'
        GO TO 500
    END IF

* Close WARPRI.TBL file
35  CLOSE(1, STATUS = 'KEEP')

```



```

* Initialize count = 1 + line = 1
  COUNT = 1
  LINE = 1

* If input file does not exist, stop
  INQUIRE (FILE = '/home/warpam/iofiles/MAX.TBL',EXIST = THERE)
  IF (.NOT. THERE) THEN
    PRINT *, ' ERROR - MAX.TBL DOES NOT EXIST'
    GO TO 500
  END IF

* Open the MAX.TBL file and read the info into OUTSTR
  OPEN(UNIT = 1, FILE = '/home/warpam/iofiles/MAX.TBL',
    $   STATUS = 'OLD', IOSTAT = STAT1)

* Skip first three rows of input because it is just header info.
39  READ(1, '(A1)', END = 55) IN11
    IF (LINE .LT. 3) THEN
      LINE = LINE + 1
      GO TO 39
    END IF

* Read input line and check for errors
40  READ(1, 50, END = 55) IN21, IN5, IN3, IN61

* Insert input data into OUTSTR and modify COUNT
  OUTSTR (COUNT)(1:5) = IN5
  OUTSTR (COUNT)(6:6) = 'X'
  OUTSTR (COUNT)(7:9) = IN3
  OUTSTR (COUNT)(10:11) = IN21
  OUTSTR (COUNT)(19:24) = IN61
  COUNT = COUNT + 1
  GO TO 40

50  FORMAT(1X,A2,3X,A5,4X,A3,4X,A6)

* Close MAX.TBL file
55  CLOSE(1, STATUS = 'KEEP')

* Initialize line = 1
  LINE = 1

* If input file does not exist, stop
  INQUIRE (FILE = '/home/warpam/iofiles/MOBTNGBS.OUT',EXIST = THERE)
  IF (.NOT. THERE) THEN
    PRINT *, ' ERROR - MOBTNGBS.OUT DOES NOT EXIST'
    GO TO 500
  END IF

```

```

* Open the MOBTNGBS.OUT file and read the info into OUTSTR
  OPEN(UNIT = 1, FILE = '/home/warpam/iofiles/MOBTNGBS.OUT',
    $   STATUS = 'OLD', IOSTAT = STAT1)

* Skip first two rows of input because it is just header info.
59  READ(1, '(A1)', END = 75) IN11
    IF (LINE .LT. 2) THEN
      LINE = LINE + 1
      GO TO 59
    END IF

* Read input line and check for errors
60  READ(1, 70, END = 75) IN21, IN5, IN3, IN61

* Insert input data into OUTSTR and modify COUNT
  OUTSTR (COUNT)(1:5) = IN5
  OUTSTR (COUNT)(6:6) = 'X'
  OUTSTR (COUNT)(7:9) = IN3
  OUTSTR (COUNT)(10:11) = IN21
  OUTSTR (COUNT)(19:24) = IN61
  COUNT = COUNT + 1
  GO TO 60

70  FORMAT(2X,A2,4X,A5,9X,A3,2X,A6)

* Close MOBTNGBS.OUT file
75  CLOSE(1, STATUS = 'KEEP')

* Initialize line = 1
  LINE = 1

* If input file does not exist, stop
  INQUIRE (FILE = '/home/warpam/iofiles/CSMII.OUT', EXIST = THERE)
  IF (.NOT. THERE) THEN
    PRINT *, ' ERROR - CSMII.OUT DOES NOT EXIST'
    GO TO 500
  END IF

* Open the CSMII.OUT file and read the info into OUTSTR
  OPEN(UNIT = 1, FILE = '/home/warpam/iofiles/CSMII.OUT',
    $   STATUS = 'OLD', IOSTAT = STAT1)

* Skip first three rows of input because it is just header info.
79  READ(1, '(A1)', END = 95) IN11
    IF (LINE .LT. 3) THEN
      LINE = LINE + 1
      GO TO 79
    END IF

```

```

* Read input line and check for errors
80  READ(1, 90, END = 95) IN21, IN5, IN11, IN61, IN62, IN63

* Insert input data into OUTSTR and modify COUNT
  OUTSTR (COUNT)(1:5) = IN5
  OUTSTR (COUNT)(6:6) = IN11
  OUTSTR (COUNT)(7:9) = 'CST'
  OUTSTR (COUNT)(10:11) = IN21
  OUTSTR (COUNT)(19:24) = IN63
  COUNT = COUNT + 1
  OUTSTR (COUNT)(1:5) = IN5
  OUTSTR (COUNT)(6:6) = IN11
  OUTSTR (COUNT)(7:9) = 'CSB'
  OUTSTR (COUNT)(10:11) = IN21
  OUTSTR (COUNT)(19:24) = IN61
  COUNT = COUNT + 1
  GO TO 80

90  FORMAT(2X,A2,4X,A5,4X,A1,3X,A6,4X,A6,4X,A6)

* Close CSMII.OUT file
95  CLOSE(1, STATUS = 'KEEP')

* If input file does not exist, stop
  INQUIRE (FILE = '/home/warpam/iofiles/AUTOREP.OUT', EXIST = THERE)
  IF (.NOT. THERE) THEN
    PRINT *, ' ERROR - AUTOREP.OUT DOES NOT EXIST'
    GO TO 500
  END IF

* Open the AUTOREP.OUT file and read the info into OUTSTR
  OPEN(UNIT = 1, FILE = '/home/warpam/iofiles/AUTOREP.OUT',
    $   STATUS = 'OLD', IOSTAT = STAT1)

* Read input line and check for errors
100 READ(1, 110, END = 115) IN21, IN5, IN11, IN3, IN61

* Insert input data into OUTSTR and modify COUNT
  IF (COUNT .LE. MAXNUM) THEN
    OUTSTR (COUNT)(1:5) = IN5
    OUTSTR (COUNT)(6:6) = IN11
    OUTSTR (COUNT)(7:9) = IN3
    OUTSTR (COUNT)(10:11) = IN21
    OUTSTR (COUNT)(19:24) = IN61
    COUNT = COUNT + 1
    GO TO 100
  ELSE
    PRINT *, ' ERROR - OUTSTR LIMIT EXCEEDED WHILE READING '
    PRINT *, ' AUTOREP.OUT. INCREASE MAXNUM LIMIT'
    PRINT *, ' TO ARRAY OUTSTR ACCORDINGLY.'
  
```

GO TO 500
END IF

110 FORMAT(2X,A2,3X,A5,3X,A1,3X,A3,3X,A6)

* Close AUTOREP.OUT file

115 CLOSE(1, STATUS = 'KEEP')

* If input file does not exist, stop

INQUIRE (FILE = '/home/warpam/iofiles/MOBMREQ.OUT', EXIST = THERE)
IF (.NOT. THERE) THEN
PRINT *, ' ERROR - MOBMREQ.OUT DOES NOT EXIST'
GO TO 500
END IF

* Open the MOBMAN.OUT file and read the info into OUTSTR

OPEN(UNIT = 1, FILE = '/home/warpam/iofiles/MOBMREQ.OUT',
\$ STATUS = 'OLD', IOSTAT = STAT1)

* Read input line and check for errors

130 READ(1, 140, END = 145) IN21, IN11, IN22, IN23, IN12, IN3, IN61

* Insert input data into OUTSTR and modify COUNT

IF (COUNT .LE. MAXNUM) THEN
OUTSTR (COUNT)(1:1) = IN11
OUTSTR (COUNT)(2:3) = IN22
OUTSTR (COUNT)(4:5) = IN23

IF (IN12 .EQ. 'M') THEN

OUTSTR (COUNT)(6:6) = 'M'

ELSE IF (IN22 .EQ. 'IN' .OR.

\$ IN22 .EQ. 'AR' .OR. IN22 .EQ. 'FA') THEN

OUTSTR (COUNT)(6:6) = 'M'

ELSE

OUTSTR (COUNT)(6:6) = 'X'

END IF

OUTSTR (COUNT)(7:9) = IN3

OUTSTR (COUNT)(10:11) = IN21

OUTSTR (COUNT)(19:24) = IN61

COUNT = COUNT + 1

GO TO 130

ELSE

PRINT *, ' ERROR - OUTSTR LIMIT EXCEEDED WHILE READING '

PRINT *, ' MOBMREQ.OUT. INCREASE MAXNUM LIMIT'

PRINT *, ' TO ARRAY OUTSTR ACCORDINGLY.'

GO TO 500

END IF

140 FORMAT(2X,A2,2X,A1,2X,A2,2X,A2,2X,A1,2X,A3,2X,A6)

* Close MOBMREQ.OUT file

145 CLOSE(1, STATUS = 'KEEP')

* If input file does not exist, stop
INQUIRE (FILE = '/home/warpam/iofiles/MOBMAST.OUT', EXIST = THERE)
IF (.NOT. THERE) THEN
PRINT *, ' ERROR - MOBMAST.OUT DOES NOT EXIST'
GO TO 500
END IF

* Open the MOBMAN.OUT file and read the info into OUTSTR
OPEN(UNIT = 1, FILE = '/home/home/iofiles/MOBMAST.OUT',
\$ STATUS = 'OLD', IOSTAT = STAT1)

* Read input line and check for errors
150 READ(1, 160, END = 165) IN21, IN11, IN22, IN23, IN12, IN3, IN61

* Insert input data into OUTSTR and modify COUNT
IF (COUNT .LE. MAXNUM) THEN
OUTSTR (COUNT)(1:1) = IN11
OUTSTR (COUNT)(2:3) = IN22
OUTSTR (COUNT)(4:5) = IN23

IF (IN12 .EQ. 'M') THEN
OUTSTR (COUNT)(6:6) = 'M'
ELSE IF (IN22 .EQ. 'IN' .OR.
\$ IN22 .EQ. 'AR' .OR. IN22 .EQ. 'FA') THEN
OUTSTR (COUNT)(6:6) = 'M'
ELSE
OUTSTR (COUNT)(6:6) = 'X'
END IF

OUTSTR (COUNT)(7:9) = IN3
OUTSTR (COUNT)(10:11) = IN21
OUTSTR (COUNT)(19:24) = IN61
COUNT = COUNT + 1
GO TO 150
ELSE
PRINT *, ' ERROR - OUTSTR LIMIT EXCEEDED WHILE READING '
PRINT *, ' MOBREQ.OUT. INCREASE MAXNUM LIMIT '
PRINT *, ' TO ARRAY OUTSTR ACCORDINGLY.'
GO TO 500
END IF

160 FORMAT(2X, A2, 2X, A1, 2X, A2, 2X, A2, 2X, A1, 2X, A3, 2X, A6)

165 COUNT = COUNT - 1
CALL MODIFYARRAY (OUTSTR, MAXNUM, CODE, ABBR, MAXTHR, PRIOR, MAXPTY,
\$ COUNT)
CALL SORTARRAY (OUTSTR, MAXNUM, COUNT)
CALL PRINTARRAY (OUTSTR, MAXNUM, COUNT)

WRITE (6, 250)

```
250  FORMAT (////////10X,'REQ/ASSETS GENERATOR DONE')
```

```
* Close file and exit program  
500  CLOSE(1, STATUS = 'KEEP')  
      END
```

***** SUBROUTINES - IN ALPHABETICAL ORDER *****

SUBROUTINE MODIFYARRAY (OS, MN, CD, AB, MT, PR, MP, CT)

```
*****
*
* SUBROUTINE MODIFYARRAY
* This subroutine goes thru the OUTSTR array and fills in the
* elements 12-18 with the correct priorities and codes.
*
*****
```

```
INTEGER CT, MT, MP, MN
CHARACTER CD(MT)*4, AB(MT)*3, OS(MN)*24
CHARACTER PR(MP)*5
INTEGER CTR1, CTR2
LOGICAL MORE
```

```
DO 690, CTR1 = 1, CT
```

* Find the correct replacement type and modify OS accordingly

```
MORE = .TRUE.
CTR2 = 1
600 IF (CTR2 .LE. MT .AND. MORE) THEN
    IF (OS(CTR1)(7:9) .EQ. AB(CTR2)) THEN
        OS(CTR1)(15:18) = CD (CTR2)
        MORE = .FALSE.
    ELSE
        CTR2 = CTR2 + 1
        GO TO 600
    END IF
END IF
```

* Find the correct priority num. and modify OS accordingly

```
MORE = .TRUE.
CTR2 = 1
610 IF (CTR2 .LE. MP .AND. MORE) THEN
    IF (OS(CTR1)(1:5) .EQ. PR(CTR2)) THEN
        IF (CTR2 .GT. 99) THEN
            OS(CTR1)(12:12) = CHAR (CTR2 / 100 + 48)
            OS(CTR1)(13:13) = CHAR ((CTR2 / 100) / 10 + 48)
            OS(CTR1)(14:14) = CHAR (MOD (CTR2, 10) + 48)
        ELSE
            OS(CTR1)(12:12) = '0'
            OS(CTR1)(13:13) = CHAR (CTR2 / 10 + 48)
            OS(CTR1)(14:14) = CHAR (MOD (CTR2, 10) + 48)
        END IF
        MORE = .FALSE.
    END IF
```

```
ELSE  
  CTR2 = CTR2 + 1  
  GO TO 610  
END IF  
END IF
```

```
690 CONTINUE
```

```
* Exit subroutine  
  RETURN  
END
```



```

SUBROUTINE PRINTARRAY (OS, MN, CT)
*****
*
* SUBROUTINE PRINTARRAY
* This subroutine prints OS to an output file.
*
*****

INTEGER MN, CT
CHARACTER OS(MN)*24
INTEGER STAT2, CNT1
CHARACTER HEAD0*40, HEAD1*40, HEAD2*41
LOGICAL THERE

*If output file exists, delete it
INQUIRE (FILE = '/home/warpam/iofiles/RECAST.OUT',EXIST = THERE)
IF (THERE) THEN
OPEN(UNIT = 2,FILE = '/home/          ifiles/RECAST.OUT',
$      STATUS = 'OLD', IOSTAT = STAT2)
CLOSE (2, STATUS = 'DELETE')
END IF

* Open the RECAST.OUT file and read the OUTSTR info. into it
OPEN(UNIT = 2,FILE = '/home/dnna/iofiles/RECAST.OUT',
$      STATUS = 'NEW', IOSTAT = STAT2)

* Print Header info.
HEAD0 = ' CAT/BR      REQ/  TIME PER/  REQ''T/'
HEAD1 = ' GRADE      S   TYPE   PRIORITY  ASSETS'
HEAD2 = '-----'
*
BXXXXXBBBBBXXXXXXBBBBXXXXXXXXXXXXBBBBXXXXX
WRITE(2,700) HEAD0, HEAD1, HEAD2

* The header format
700  FORMAT(1X,A39/1X,A39,/1X,A40)

* Go thru matrix and print info to outfile
DO 750, CNT1 = 1, CT
WRITE(2,770) OS(CNT1)(1:5), OS(CNT1)(6:6), OS(CNT1)(7:9),
$      OS(CNT1)(10:18), OS(CNT1)(19:24)
750  CONTINUE

770  FORMAT(2X,A5,4X,A1,4X,A3,3X,A9,3X,A6)

* Close RECAST.OUT file
CLOSE(2, STATUS = 'KEEP')

* Exit subroutine
RETURN
END

```

```

      SUBROUTINE SORTARRAY (OS, MN, CT)
*****
*
* SUBROUTINE SORTARRAY
* This subroutine uses a shell sort to sort OUTSTR by
* time per/priority (elem 10-18) in ascending order.
*
*****

      INTEGER MN, CT
      CHARACTER OS(MN)*24, TEMP*24
      INTEGER CTR, NDELTA
      LOGICAL INORDR

      NDELTA = CT
800  IF (NDELTA .GT. 1) THEN
      NDELTA = NDELTA/2
810  INORDR = .TRUE.
      DO 820, CTR = 1, CT - NDELTA
      IF (OS(CTR)(10:18) .GT. OS(CTR + NDELTA)(10:18)) THEN
      TEMP = OS(CTR)
      OS(CTR) = OS(CTR + NDELTA)
      OS(CTR + NDELTA) = TEMP
      INORDR = .FALSE.
      END IF
820  CONTINUE
      IF (.NOT. INORDR) GO TO 810
      GO TO 800
      END IF

* Exit subroutine
      RETURN
      END

```

SECTION 5 RECLASSIFICATION MODEL

5.1 GENERAL

The Reclassification Model is designed to return a percentage of the casualties (requirements) sustained within a theater during a time period back to duty in a number of new branches over several later time periods to simulate the effects of hospitalization and reclassification actions. The model allows the user through the control of various input variables and user created tables to simulate current personnel policy or conduct "What If" analysis.

5.2 INITIATION

The Reclassification Model is initiated through user input from a Sun window which activates the Reclassification FORTRAN program. To proceed, the user must type "go" on the response line to advance to the first input variable. This input line ONLY ACCEPTS the word "go" in small case letters. Files produced from previous runs of the preprocessor should be stored in a different sub-directory or under a different file name (as with the date run) prior to running the preprocessor modules. Any file of the same name in the IOFILE sub-directory on the Sun workstation will be overwritten by the new output file.

5.3 INPUT FILES

The files required for each routine and sub-routine are listed at the beginning of the programs.

5.4 INPUT VARIABLES

The user is prompted by the input screen to input the desired value of the following variables on a response line: (input variables from previous runs are shown on the input screen prior to the first response)

Requirement File: Which of the various requirement files does the user desire to use for this run of the model. The available requirement files are listed at the input line. The Reclassification Model will not accept the MAX requirement file as an input and will run the DEG file in its place.

Time Periods: A time period is 10 days. The model will only run for the time periods chosen. The user must input both the start time period and the end time period for the run. If start time period is "1" and end time period is "10" the model will run time periods "1-10" inclusive. Due to the configuration of current input data (MOBMAN has all assets in time period one), the model MUST BE STARTED WITH TIME PERIOD ONE.

Branch: Branch represents the specialties/MOS and grade combinations which have been grouped together in the preprocessor. These branches are then prioritized in the Branch Look-Up Table and given a priority number. The user should consult the current table in the preprocessor to determine the priority code for specific branches. The model can be run with from one or up to the maximum number of branches which were created in the preprocessor. The initial version of WARPAM has 67 branch/grade combinations.

Return-to-Duty Rate: This is the percentage of casualties which the user desires to return to duty within the theater. The model will accept either a rate (decimal) or percentage (whole number) ranging from .1% (.001) to 99.99% (.9999). Based on 1989 CAA estimates the recommended rate for current policy is 20%.

5.5 PROCESSING

Casualties (requirements) are redesignated as new branches specified in the officer and enlisted reclassification tables. This is accomplished by reading the requirements line from the REQAST.TBL for the specified requirement into the model. Then through a series of calculations the requirements are transformed into a reduced number of assets in new branches based on the data found in the reclassification look-up tables. The current model then distributes these reclassified personnel over six time periods according to percentages found in the reclassification delay table. Following the reclassification processing the model appends these results to the REQAST.TBL, sorts the file, and relabels the new file as MODRQAST.TBL which is used in subsequent models. This processing flow is depicted in figure 8.

5.6 OUTPUT REPORTS

The modified requirements/assets file (MODRQAST.TBL) is produced from each run of the reclassification model. This file is automatically replicated as a DOS file which the user is allowed to view using DBASE III. However, any changes made by the user will not effect the subsequent models as these changes are not recorded on the UNIX addition of the file which is used by the CRC/RPLCO model. The intent of reviewing the MODRQAST.TBL is detect any catastrophic errors in the file prior to using it in other models. To make changes in the UNIX version of the file, the programmer must use an editor program directly with this file version.

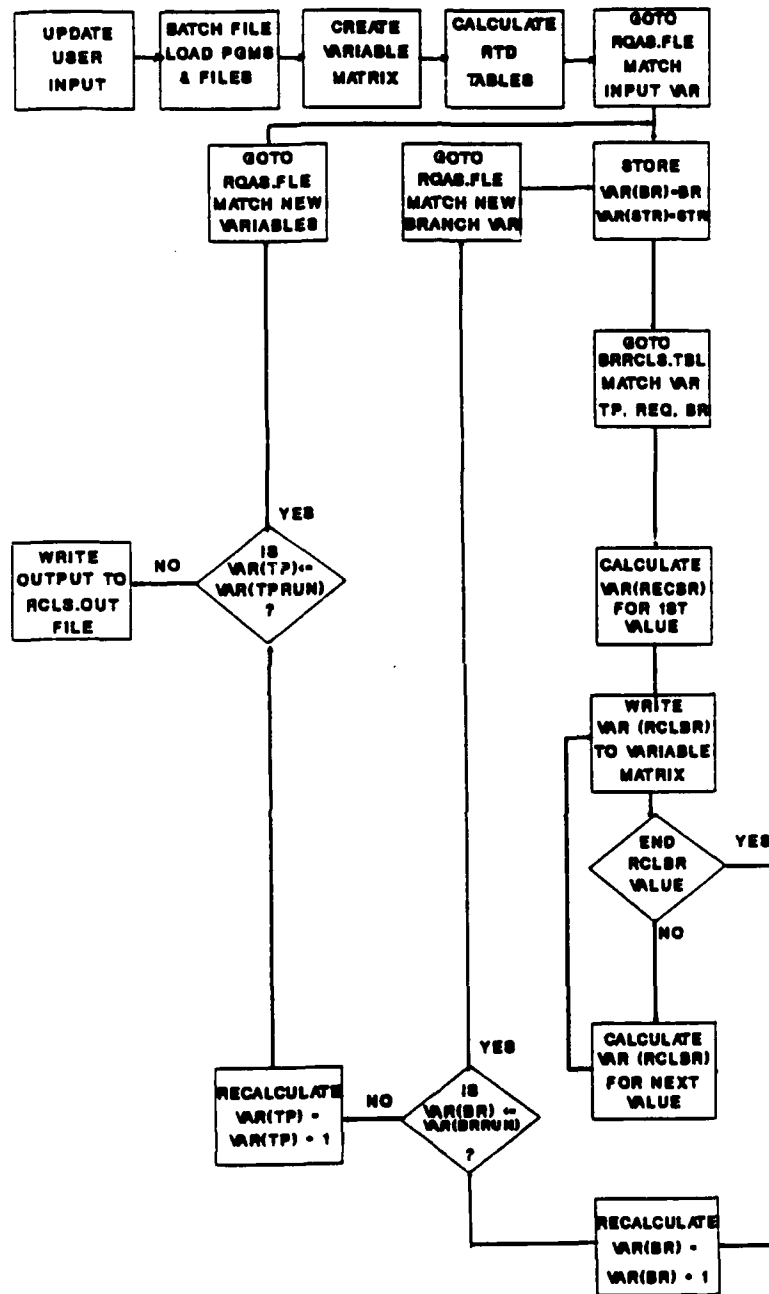


FIGURE 8: RECLASSIFICATION MODEL PROCESSING

5.7 RECLAS MODEL FORTRAN PROGRAMS

```

C*****
C
C Program Name:  MODRECL                      Date:  05-21-1990
C
C File Name:    MODRCLS.FOR
C
C Programmer:   Beth White, SAIC, 749-8771
C
C Description:  Modifies the Requirements/Assets file by appending
C              TRD [Theater Return to Duties] to the file output.
C
C Input:        REQAST.OUT
C              ORCLSPER.TBL
C              ERCLSPER.TBL
C              WARPRI.TBL
C              RCLSDLY.TBL
C
C Output:       MODRQAST.OUT
C
C*****
C Modifications: (STATUS: P - PROPOSED; R - REQUIRED ; C - COMPLETED)
C
C Number   Status   Date:           Description:           Initials
C -----
C    01      C      05/31/90  Modified directory changes.      BAW
C
C*****

```

PROGRAM MODRECL

C Global Variables

```

    DIMENSION ORCL(18,18),ORCLBR(18),ERCL(17,17),ERCLBR(17),
&WPRNUMB(67),WPRCODE(67),RDLAY(18,6),STOR(5000,4),
&STORST(5000),OUTSTR(40000)
    CHARACTER*1 RQCHR(40),UNPT(55),CAT,SEX,XTP1,TABL
    CHARACTER*2 ORCLBR,ERCLBR,WPRNUMB,BR,GRD,TP,NEWBR,NBRN,CHNTP
    CHARACTER*3 RDREQ,SREQ,TYPE,VREQ,VBRR,CBG1,NWBRCH
    CHARACTER*4 REQASET
    CHARACTER*5 WPRCODE,NEWCBG,CTPNB
    CHARACTER*6 VSTOR
    CHARACTER*8 FLNAM1
    CHARACTER*9 TPBRQA,STOR
    CHARACTER*11 FLNAM2
    CHARACTER*21 DIRNAM
    CHARACTER*24 fdate,OUTSTR
    CHARACTER*32 FNAME
    CHARACTER*40 RACHR
    CHARACTER*55 USRPT

```

```

LOGICAL THERE
REAL RDPCNT, RETDP, ORCL, ERCL, RDLAY

```

```

INTEGER ICHK, USRNPOT, STARTP, ENDP, STARTBR, ENDBR, NEWSTR,
&VSTRNG, VTP, CURNTP, NCOLUMN, ADDSTR, SUMRET, TEMPSTR, CURBR,
&LOOP, COUNT, MAXCOUNT, IFLG, STORST, VBRNH, CNT1, KL

```

```

COMMON/PASRTD/RETD
COMMON/OCST/ORCL, ORCLBR
COMMON/ECLST/ERCL, ERCLBR
COMMON/WARPR/WPRNUMB, WPRCODE
COMMON/RCDLY/RDLAY
COMMON/CVRTT/RQCHR, VTP, VBRNH, VSTRNG
COMMON/JTSRT/OUTSTR, CNT1

```

```

EQUIVALENCE (RQCHR(1), RACHR)
EQUIVALENCE (UNPT(1), USRPT)

```

C Local Variables

```

ICHK = 0
COUNT = 0
MAXCOUNT = 0
IFLG = 0
VTP = 0
VBRNH = 0
VSTRNG = 0
SEX = 'X'
TYPE = 'TRD'
REQASET = '0100'

```

```

90 WRITE(6,90)
   FORMAT(//////20X, '*****',
&/20X, '*****', //20X,
&'WARPAM RECLAS MODIFICATION MODULE', //20X,
&'*****', /20X,
&'*****', //20X,
&'THE FOLLOWING FILES ARE NEEDED:', /30X,
&'REQAST.OUT', /30X, 'RCLSDLY.TBL', /30X, 'ORCLSPER.TBL',
&/30X, 'ERCLSPER.TBL', /30X, 'WARPRI.TBL', //20X)

```

```

PAUSE

```

```

91 WRITE(6,91)
   FORMAT(////////////////////////////////////)

```

C Checks to see if input files exists. If input files do not exist
C the write error message and terminate program.

```

INQUIRE(FILE='/home/warpam/iofiles/REQAST.OUT', EXIST=TRUE)
IF (.NOT. TRUE) THEN

```

```

        WRITE(6,*)'ERROR - REQAST.OUT does not exist.'
        GOTO 310
    ENDIF

    INQUIRE(FILE='/home/warpam/iofiles/RCLSDLY.TBL',EXIST=THE)
    IF (.NOT.THERE)THEN
        WRITE(6,*)'ERROR - RCLSDLY.TBL does not exist.'
        GOTO 310
    ENDIF

    INQUIRE(FILE='/home/warpam/iofiles/ORCLSPER.TBL',EXIST=THE)
    IF (.NOT.THERE)THEN
        WRITE(6,*)'ERROR - ORCLSPER.TBL does not exist.'
        GOTO 310
    ENDIF

    INQUIRE(FILE='/home/warpam/iofiles/ERCLSPER.TBL',EXIST=THE)
    IF (.NOT.THERE)THEN
        WRITE(6,*)'ERROR - ERCLSPER.TBL does not exist.'
        GOTO 310
    ENDIF

    INQUIRE(FILE='/home/warpam/iofiles/WARPRI.TBL',EXIST=THE)
    IF (.NOT.THERE)THEN
        WRITE(6,*)'ERROR - WARPRI.TBL does not exist.'
        GOTO 310
    ENDIF

    INQUIRE(FILE='/home/warpam/iofiles/REQAST.TMP',EXIST=THE)
    IF (THERE)THEN
        OPEN(10,FILE='/home/warpam/iofiles/REQAST.TMP',STATUS='OLD')
        CLOSE(10,STATUS='DELETE')
    ENDIF

    OPEN(10,FILE='/home/warpam/iofiles/REQAST.TMP',STATUS='NEW')
    OPEN(915,FILE='/home/warpam/iofiles/REQAST.OUT',STATUS='OLD')
920  READ(915,'(40(A1))',ERR = 310,END = 951)RQCHR
    WRITE(10,922)RQCHR
922  FORMAT(40(A1))
    GOTO 920
951  CLOSE(915,STATUS='KEEP')
    CLOSE(10,STATUS='KEEP')

```

C Checks to see if output file exist. If output file does exist,
 C the old output file is deleted. File: XXX.OUT is the temporary
 C file which stores the unsorted TRD's. File: MODRQAST.OUT is
 C the final sorted modified requirements asset file with the
 C new TRD's. File: USRNPOT.OUT is the user input file that stores
 C all the user variable declarations.

```

    INQUIRE(FILE='/home/warpam/iofiles/XXX.OUT',EXIST=THE)
    IF (THERE)THEN

```



```

        OPEN(202,FILE='/home/warpam/iofiles/XXX.OUT',STATUS='OLD')
        CLOSE(202,STATUS='DELETE')
    ENDIF

    INQUIRE(FILE='/home/warpam/iofiles/MODRQAST.OUT',EXIST=THRE)
    IF (THRE)THEN
        OPEN(15,FILE='/home/warpam/iofiles/MODRQAST.OUT',STATUS='OLD')
        CLOSE(15,STATUS='DELETE')
    ENDIF

    INQUIRE(FILE='/home/warpam/iofiles/USRNPUT.OUT',EXIST=THRE)
    IF (THRE)THEN
        WRITE(6,299)
299    FORMAT(5X,'The USRNPUT.OUT file already exists.',/5X,
        &'The following screen shows the previous input values.',/))
        OPEN(2,FILE='/home/warpam/iofiles/USRNPUT.OUT',STATUS='OLD')
300    READ(2,'(55(A1))',ERR=301,END=302)UNPT
        WRITE(6,350)UNPT
350    FORMAT(55(A1))
        GOTO 300
301    WRITE(6,*)'ERROR READING FILE:  USRNPUT.OUT'
302    CLOSE(2,STATUS='KEEP')
        OPEN(2,FILE='/home/warpam/iofiles/USRNPUT.OUT',
        &ACCESS='APPEND',STATUS='OLD')
    ELSE
        OPEN(2,FILE='/home/warpam/iofiles/USRNPUT.OUT',STATUS='NEW')
    ENDIF

```

C Begin MODRECL

C User Input Variables

C Input Variable (Start Time Period and End Time Period)

```

        WRITE(6,10)
10    FORMAT(/10X,'ENTER START TIME PERIOD (1-18)')
        READ(*,*)USRNPUT
        IF ((USRNPUT.GT.0).AND.(USRNPUT.LT.19))THEN
            STARTP = USRNPUT
        ELSE
            STARTP = 1
            WRITE(6,12)
12    FORMAT(/15X,'ERROR - INVALID TIME PERIOD',/15X,
            &'DEFAULT START TIME PERIOD WILL BE USED:  STARTP = 1')
        ENDIF

        WRITE(6,101)
101    FORMAT(/10X,'ENTER END TIME PERIOD (1-18)')
        READ(*,*)USRNPUT
        IF ((USRNPUT.GT.0).AND.(USRNPUT.LT.19))THEN
            ENDTP = USRNPUT
        ELSE
            ENDTP = 18

```

```

        WRITE(6,121)
121    FORMAT(/15X,'ERROR - INVALID END TIME PERIOD',/15X,
        & 'DEFAULT END TIME PERIOD WILL BE USED:  ENDTP = 18')
        ENDIF

```

C Input Variable (Start Requirement)

```

9    WRITE(6,13)
13   FORMAT(/10X,'ENTER START REQUIREMENT',/10X,
        & '(MAX,DEG,AE1,AKO,ASW,CST,CSB)')
        READ(*,*)RDREQ
        IF ((RDREQ.EQ.'MAX').OR.(RDREQ.EQ.'max'))THEN
            WRITE(6,*)'                ERROR - NOT ACCEPTED'
            GOTO 9
        ENDIF
        IF ((RDREQ.EQ.'DEG').OR.(RDREQ.EQ.'deg'))THEN
            ICHK = 1
            SREQ = 'DEG'
            GOTO 32
        ENDIF
        IF ((RDREQ.EQ.'AE1').OR.(RDREQ.EQ.'ae1'))THEN
            ICHK = 1
            SREQ = 'AE1'
            GOTO 32
        ENDIF
        IF ((RDREQ.EQ.'AKO').OR.(RDREQ.EQ.'ako'))THEN
            ICHK = 1
            SREQ = 'AKO'
            GOTO 32
        ENDIF
        IF ((RDREQ.EQ.'ASW').OR.(RDREQ.EQ.'asw'))THEN
            ICHK = 1
            SREQ = 'ASW'
            GOTO 32
        ENDIF
        IF ((RDREQ.EQ.'CST').OR.(RDREQ.EQ.'cst'))THEN
            ICHK = 1
            SREQ = 'CST'
            GOTO 32
        ENDIF
        IF ((RDREQ.EQ.'CSB').OR.(RDREQ.EQ.'csb'))THEN
            ICHK = 1
            SREQ = 'CSB'
            GOTO 32
        ENDIF
        IF (ICHK.EQ.0)THEN
            SREQ = 'DEG'
            WRITE(6,33)
33    FORMAT(/15X,'ERROR - INVALID REQUIREMENT',/15X,
            & 'DEFAULT START REQUIREMENT WILL BE USED:  SREQ = DEG')
        ENDIF

```

C Input Variable (Return To Duty Percent)

```
32  WRITE(6,14)
14  FORMAT(/10X,'ENTER RTD PERCENT(%)',/10X,
&'(PERCENT WILL BE TRANSLATED TO RATE IN DECIMAL FORM)',
&/10X,'(NORMAL RTD PERCENT IS 20%)')
    READ(*,*)RDPCNT
    IF ((RDPCNT.GT.0).AND.(RDPCNT.LT.1.0))RETDP = RDPCNT
    IF (RDPCNT.GE.1.0)RETDP = (RDPCNT)/100.0
```

C Input Variable (Start Branch and End Branch)

```
    WRITE(6,115)
115  FORMAT(/10X,'ENTER START BRANCH (1-67)')
    READ(*,*)USRNPUT
    IF ((USRNPUT.GT.0).AND.(USRNPUT.LT.68))THEN
        STARTBR = USRNPUT
    ELSE
        STARTBR = 1
        WRITE(6,34)
34    FORMAT(/15X,'ERROR - INVALID START BRANCH NO.',/15X,
&'DEFAULT START BRANCH NO. WILL BE USED:  STARTBR = 1')
    ENDIF

    WRITE(6,16)
16  FORMAT(/10X,'ENTER END BRANCH (1-67)')
    READ(*,*)USRNPUT
    IF ((USRNPUT.GT.0).AND.(USRNPUT.LT.68))THEN
        ENDBR = USRNPUT
    ELSE
        ENDBR = 67
        WRITE(6,334)
334  FORMAT(/15X,'ERROR - INVALID END BRANCH NO.',/15X,
&'DEFAULT END BRANCH NO. WILL BE USED:  ENDBR = 67')
    ENDIF
```

C Write user inputs to screen and file: USRNPUT.OUT

```
    WRITE(2,81)STARTP,ENDTP,SREQ,RETDP,STARTBR,ENDBR,fdate()
81  FORMAT(2X,I2,2X,I2,2X,A3,2X,F4.2,2X,I2,2X,I2,3X,A24)
    CLOSE(2,STATUS='KEEP')

    WRITE(6,80)fdate(),STARTP,ENDTP,SREQ,RETDP,STARTBR,ENDBR
80  FORMAT(/10X,'USER INPUT(S): ',/12X,A24,/12X,
&'START TIME PERIOD ---> ',I2,/12X,
&'END TIME PERIOD ---> ',I2,/12X,
&'START REQUIREMENT ---> ',A3,/12X,
&'RTD RATE ---> ',F4.2,/12X,
&'START BRANCH NO. ---> ',I2,/12X,
&'END BRANCH NO. ---> ',I2)
```

C Subroutines globally store ORCLSPER.TBL, ERCLSPER.TBL,

C RCLSDLY.TBL, and WARPRI.TBL.

```
CALL ORCT
CALL ERCT
CALL RDLY
CALL WARPRIT
```

C Initializes I line counter to zero and initializes the
C CURNTP [Current time period] and CURBR [Current branch].

```
I = 0
CURNTP = STARTP
CURBR = STARTBR
```

C Prepares a separate file name for the TRD's using the
C requirement name plus an extension. i.e.[DEGRCLS.OUT]

```
DIRNAM = '/home/warpam/iofiles/'
FLNAM1 = 'RCLS.OUT'
FLNAM2 = SREQ // FLNAM1
FNAME = DIRNAM // FLNAM2
```

C Read input file: REQAST.TMP.

```
OPEN(10,FILE='/home/warpam/iofiles/REQAST.TMP',STATUS='OLD')
666 READ(10,'(40(A1))',ERR=499,END=500)RQCHR
I = I + 1
IF (I.LT.4)GOTO 666
VREQ = RACHR(17:19)
IF (VREQ.NE.SREQ)GOTO 666
TP = RACHR(23:24)
```

C Subroutine converts Time Period from characters to a numeric
C value [VTP].

```
CALL CNVRTP

IF (CURNTP.EQ.18)GOTO 500
IF ((VTP.NE.CURNTP).AND.(VTP.GT.ENDTP))GOTO 500
IF ((VTP.NE.CURNTP).AND.(VTP.LE.ENDTP))THEN
  CURNTP = VTP
  CURBR = STARTBR
  VBRR = RACHR(25:27)
```

C Converts variable branch [VBRR] from character to a numeric
C value [VBRNH].

```
CALL CNVRBR
IF ((VBRNH.NE.CURBR).AND.(VBRNH.GT.ENDBR))GOTO 500
IF ((VBRNH.NE.CURBR).AND.(VBRNH.LE.ENDBR))THEN
  CURBR = VBRNH
  CAT = RACHR(3:3)
```

```

        BR = RACHR(4:5)
        GRD = RACHR(6:7)
        VSTOR = RACHR(35:40)

C  Converts variable strength [VSTR] from character to an
C  numeric value [VSTRNG].

        CALL CNVRSTR
        GOTO 200
    ENDIF

    IF ((VBRNH.EQ.CURBR).AND.(VBRNH.GT.ENDBR))GOTO 500
    IF ((VBRNH.EQ.CURBR).AND.(VBRNH.LE.ENDBR))THEN
        CAT = RACHR(3:3)
        BR = RACHR(4:5)
        GRD = RACHR(6:7)
        VSTOR = RACHR(35:40)

C  Converts variable strength [VSTR] from character to an
C  numeric value [VSTRNG].

        CALL CNVRSTR
        GOTO 200
    ENDIF
ENDIF

    IF ((VTP.EQ.CURNTP).AND.(VTP.LE.ENDTP))THEN
        VBRR = RACHR(25:27)

C  Converts variable branch [VBRR] from character to a numeric
C  value [VBRNH].

        CALL CNVRBR

        IF ((VBRNH.NE.CURBR).AND.(VBRNH.GT.ENDBR))GOTO 500
        IF ((VBRNH.NE.CURBR).AND.(VBRNH.LE.ENDBR))THEN
            CURBR = VBRNH
            CAT = RACHR(3:3)
            BR = RACHR(4:5)
            GRD = RACHR(6:7)
            VSTOR = RACHR(35:40)

C  Converts variable strength [VSTR] from character to an
C  numeric value [VSTRNG].

            CALL CNVRSTR
            GOTO 200
        ENDIF

        IF ((VBRNH.EQ.CURBR).AND.(VBRNH.GT.ENDBR))GOTO 500
        IF ((VBRNH.EQ.CURBR).AND.(VBRNH.LE.ENDBR))THEN
            CAT = RACHR(3:3)

```

```
BR = RACHR(4:5)
GRD = RACHR(6:7)
VSTR = RACHR(35:40)
```

C Converts variable strength [VSTR] from character to an
C numeric value [VSTRNG].

```
CALL CNVRSTR
GOTO 200
ENDIF
ENDIF
```

C Category : OFFICER, WARRANT, ENLISTED

```
200 IF (CAT.EQ.'O')THEN
    NCOLUMN = 18
    TABL = 'O'
    GOTO 250
```

```
ENDIF
IF (CAT.EQ.'W')THEN
    NCOLUMN = 18
    TABL = 'W'
    GOTO 250
```

```
ENDIF
IF (CAT.EQ.'E')THEN
    NCOLUMN = 17
    TABL = 'E'
    GOTO 250
ENDIF
```

```
250 DO 251 LL = 1,NCOLUMN
    IF (TABL.EQ.'O')THEN
        IF (BR.EQ.ORCLBR(LL))GOTO 252
        GOTO 251
    ENDIF
    IF (TABL.EQ.'W')THEN
        IF (LL.LT.16)GOTO 251
        IF (BR.EQ.ORCLBR(LL))GOTO 252
        GOTO 251
    ENDIF
    IF (TABL.EQ.'E')THEN
        IF (BR.EQ.ERCLBR(LL))GOTO 252
        GOTO 251
    ENDIF
251 CONTINUE
```

C Defines new branch using correct table file.

```
252 DO 253 JJ = 1,NCOLUMN
    IF (CAT.EQ.'O')THEN
        IF (JJ.GT.15)GOTO 253
        IF (JJ.EQ.1)NEWBR = 'IN'
```

```

      IF (JJ.EQ.2)NEWBR = 'AR'
      IF (JJ.EQ.3)NEWBR = 'FA'
      IF (JJ.EQ.4)NEWBR = 'AD'
      IF (JJ.EQ.5)NEWBR = 'AV'
      IF (JJ.EQ.6)NEWBR = 'CE'
      IF (JJ.EQ.7)NEWBR = 'SC'
      IF (JJ.EQ.8)NEWBR = 'MP'
      IF (JJ.EQ.9)NEWBR = 'MI'
      IF (JJ.EQ.10)NEWBR = 'MC'
      IF (JJ.EQ.11)NEWBR = 'CM'
      IF (JJ.EQ.12)NEWBR = 'TC'
      IF (JJ.EQ.13)NEWBR = 'OD'
      IF (JJ.EQ.14)NEWBR = 'QM'
      IF (JJ.EQ.15)NEWBR = 'CS'
      GOTO 254
ENDIF
IF (CAT.EQ.'W')THEN
      IF (JJ.LT.16)GOTO 253
      IF (JJ.EQ.16)NEWBR = 'CB'
      IF (JJ.EQ.17)NEWBR = 'CS'
      IF (JJ.EQ.18)NEWBR = 'CC'
      GOTO 254
ENDIF
IF (CAT.EQ.'E')THEN
      IF (JJ.EQ.1)NEWBR = 'AR'
      IF (JJ.EQ.2)NEWBR = 'AV'
      IF (JJ.EQ.3)NEWBR = 'IN'
      IF (JJ.EQ.4)NEWBR = 'FA'
      IF (JJ.EQ.5)NEWBR = 'AD'
      IF (JJ.EQ.6)NEWBR = 'CE'
      IF (JJ.EQ.7)NEWBR = 'CM'
      IF (JJ.EQ.8)NEWBR = 'MI'
      IF (JJ.EQ.9)NEWBR = 'MP'
      IF (JJ.EQ.10)NEWBR = 'SC'
      IF (JJ.EQ.11)NEWBR = 'MC'
      IF (JJ.EQ.12)NEWBR = 'TC'
      IF (JJ.EQ.13)NEWBR = 'MM'
      IF (JJ.EQ.14)NEWBR = 'OD'
      IF (JJ.EQ.15)NEWBR = 'QM'
      IF (JJ.EQ.16)NEWBR = 'SM'
      IF (JJ.EQ.17)NEWBR = 'CS'
      GOTO 254
ENDIF

```

C Extracts new branch and concatenates category, new branch,
 C and grade. Translates New Branch code using WARPRI.TBL
 C matrix. Then, spreads new strength over n-new time periods
 C [where maximum n = 6]. Furthermore, the new time period
 C [NTP] is converted to characters and the new time period,
 C branch, and requirement/asset are concatenated.

254 CBG1 = CAT // NEWBR

```

NEWCBG = CBG1 // GRD

DO 255 KBR = 1,67
  IF (WPRCODE(KBR).EQ.NEWCBG)THEN
    XTP1 = '0'
    NBRN = WPRNUMB(KBR)
    NWBRCH = XTP1 // NBRN
    GOTO 256
  ENDIF
255 CONTINUE

256 IF (CURNTP.LT.13)LOOP = 6
    IF (CURNTP.EQ.13)LOOP = 5
    IF (CURNTP.EQ.14)LOOP = 4
    IF (CURNTP.EQ.15)LOOP = 3
    IF (CURNTP.EQ.16)LOOP = 2
    IF (CURNTP.EQ.17)LOOP = 1

SUMRET = 0
TEMPSTR = NINT(ORCL(LL,JJ) * VSTRNG)
DO 257 NN = 1,LOOP
  NTP = CURNTP + NN
  NEWSTR = NINT((ORCL(LL,JJ) * VSTRNG) * RDLAY(CURNTP,NN))
  SUMRET = NEWSTR + SUMRET
  IF (NN.EQ.LOOP)THEN
    ADDSTR = TEMPSTR - SUMRET
    NEWSTR = NEWSTR + ADDSTR
    SUMRET = SUMRET + ADDSTR
  ENDIF
  IF (NEWSTR.LT.0)NEWSTR = 0
  IF (NEWSTR.EQ.0)GOTO 257

  IF (NTP.EQ.1)CHNTP = '01'
  IF (NTP.EQ.2)CHNTP = '02'
  IF (NTP.EQ.3)CHNTP = '03'
  IF (NTP.EQ.4)CHNTP = '04'
  IF (NTP.EQ.5)CHNTP = '05'
  IF (NTP.EQ.6)CHNTP = '06'
  IF (NTP.EQ.7)CHNTP = '07'
  IF (NTP.EQ.8)CHNTP = '08'
  IF (NTP.EQ.9)CHNTP = '09'
  IF (NTP.EQ.10)CHNTP = '10'
  IF (NTP.EQ.11)CHNTP = '11'
  IF (NTP.EQ.12)CHNTP = '12'
  IF (NTP.EQ.13)CHNTP = '13'
  IF (NTP.EQ.14)CHNTP = '14'
  IF (NTP.EQ.15)CHNTP = '15'
  IF (NTP.EQ.16)CHNTP = '16'
  IF (NTP.EQ.17)CHNTP = '17'
  IF (NTP.EQ.18)CHNTP = '18'

CTPNB = CHNTP // NWBRCH

```


TPBRQA = CTPNB // REQASET

C Stores new time period, category, branch, requirement/asset.
C branch code, and strength to array [STOR] & [STORST].

```
      COUNT = COUNT + 1
      IFLG = 0
      IF (COUNT.GT.1)GOTO 258
      IF (COUNT.EQ.1)THEN
        STOR(COUNT,1) = NEWCBG
        STOR(COUNT,2) = SEX
        STOR(COUNT,3) = TYPE
        STOR(COUNT,4) = TPBRQA
        STORST(COUNT) = NEWSTR
        MAXCOUNT = COUNT
        GOTO 257
      ENDIF
258      DO 259 NJ = 1,MAXCOUNT
          IF ((NEWCBG.EQ.STOR(NJ,1)).AND.
            &(SEX.EQ.STOR(NJ,2)))GOTO 260
          IFLG = 1
          GOTO 259
260      IF ((TYPE.EQ.STOR(NJ,3)).AND.
            &(TPBRQA.EQ.STOR(NJ,4)))GOTO 261
          IFLG = 1
          GOTO 259
261      STORST(NJ) = NEWSTR + STORST(NJ)
          GOTO 257
259      CONTINUE
          IF (IFLG.EQ.1)THEN
            MAXCOUNT = MAXCOUNT + 1
            STOR(MAXCOUNT,1) = NEWCBG
            STOR(MAXCOUNT,2) = SEX
            STOR(MAXCOUNT,3) = TYPE
            STOR(MAXCOUNT,4) = TPBRQA
            STORST(MAXCOUNT) = NEWSTR
          ENDIF
257      CONTINUE
253      CONTINUE
```

C Proceeds to next Branch (CURBR = CURBR + 1).

```
      IF (CURBR.LE.ENDBR)THEN
        CURBR = CURBR + 1
        GOTO 666
      ENDIF
499  WRITE(6,*)'ERROR READING FILE:  REQAST.TMP'
500  CLOSE(10,STATUS='KEEP')
```

C Appends results to file: REQAST.TMP and creates a separate
C TRD file.

```
      OPEN(202,FILE='/home/warpam/iofiles/XXX.OUT',STATUS='NEW')
      OPEN(10,FILE='/home/warpam/iofiles/REQAST.TMP',ACCESS='APPEND',
&STATUS='OLD')
      DO 470 I = 1,MAXCOUNT
        WRITE(202,777)STOR(I,1),STOR(I,2),STOR(I,3),STOR(I,4),STORST(I)
        WRITE(10,777)STOR(I,1),STOR(I,2),STOR(I,3),STOR(I,4),STORST(I)
777   FORMAT(2X,A5,4X,A1,4X,A3,3X,A9,3X,I6)
470   CONTINUE
      CLOSE(202,STATUS='KEEP')
      CLOSE(10,STATUS='KEEP')
```

C Stores into array to be sorted.

```
      I = 0
      CNT1 = 0

      OPEN(15,FILE='/home/warpam/iofiles/MODRQAST.OUT',STATUS='NEW')
      OPEN(10,FILE='/home/warpam/iofiles/REQAST.TMP',STATUS='OLD')
825   READ(10,'(40(A1))',ERR=810,END=811)RQCHR
      I = I + 1
      IF (I.LE.3)THEN
        WRITE(15,701)RQCHR
701   FORMAT(40(A1))
        GOTO 825
      ENDIF
      CNT1 = CNT1 + 1
      OUTSTR(CNT1)(1:5) = RACHR(3:7)
      OUTSTR(CNT1)(6:6) = RACHR(12:12)
      OUTSTR(CNT1)(7:9) = RACHR(17:19)
      OUTSTR(CNT1)(10:18) = RACHR(23:31)
      OUTSTR(CNT1)(19:24) = RACHR(35:40)
      GOTO 825

810   WRITE(6,*)'ERROR READING FILE: REQAST.TMP'
811   CLOSE(10,STATUS='KEEP')
      CLOSE(15,STATUS='KEEP')
```

C Calls subroutine SORTARRAY to resort file and then write
C to output file: MODRQAST.OUT.

```
      CALL SORTARRAY

      OPEN(15,FILE='/home/warpam/iofiles/MODRQAST.OUT',
&ACCESS='APPEND',STATUS='OLD')
      DO 905 NR = 1,CNT1
        WRITE(15,877)OUTSTR(NR)(1:5),OUTSTR(NR)(6:6),
&OUTSTR(NR)(7:9),OUTSTR(NR)(10:18),OUTSTR(NR)(19:24)
877   FORMAT(2X,A5,4X,A1,4X,A3,3X,A9,3X,A6)
905   CONTINUE
```

CLOSE(15,STATUS='KEEP')

C Stores into array to be sorted.

CNT1 = 0

```
OPEN(202,FILE='/home/warpam/iofiles/XXX.OUT',STATUS='OLD')
901 READ(202,'(40(A1))',ERR=918,END=919)RQCHR
    CNT1 = CNT1 + 1
    OUTSTR(CNT1)(1:5) = RACHR(3:7)
    OUTSTR(CNT1)(6:6) = RACHR(12:12)
    OUTSTR(CNT1)(7:9) = RACHR(17:19)
    OUTSTR(CNT1)(10:18) = RACHR(23:31)
    OUTSTR(CNT1)(19:24) = RACHR(35:40)
    GOTO 901
```

```
918 WRITE(6,*)'ERROR READING TEMP FILE: XXX.OUT'
919 CLOSE(202,STATUS='KEEP')
```

C Calls subroutine SORTARRAY to resort file and the write
C to output file: requirement + RCLS.OUT.

CALL SORTARRAY

C Checks to see if the separate sorted TRD file exists. If
C the file exists, then the old version of the file is deleted
C and a new TRD file is created.

```
INQUIRE(FILE=FNAME,EXIST=THERE)
IF (THERE)THEN
    OPEN(201,FILE=FNAME,STATUS='OLD')
    CLOSE(201,STATUS='DELETE')
ENDIF

OPEN(201,FILE=FNAME,STATUS='NEW')
DO 925 KL = 1,CNT1
    WRITE(201,923)OUTSTR(KL)(1:5),OUTSTR(KL)(6:6),
&OUTSTR(KL)(7:9),OUTSTR(KL)(10:18),OUTSTR(KL)(19:24)
923   FORMAT(2X,A5,4X,A1,4X,A3,3X,A9,3X,A6)
925 CONTINUE
    CLOSE(201,STATUS='KEEP')
```

C Deletes file: XXX.OUT temporary files which stored the
C unsorted TRD's

```
OPEN(202,FILE='/home/warpam/iofiles/XXX.OUT',STATUS='OLD')
CLOSE(202,STATUS='DELETE')
310 WRITE(6,*)' RECLAS MODIFICATION MODEL COMPLETED '
    STOP
    END
```

C END MODRCLS.FOR

```

C*****
C
C                      SUBROUTINES
C*****

```

```

C*****

```

```

C
C Program Name:  ORCT                      Date:  05-23-1990
C
C File Name:    MODRCLS.FOR
C
C Programmer:   Beth White, SAIC, 749-8771
C
C Description:  Read and stores Officers Reclas Table to an array.
C
C Input:       ORCLSPER.TBL
C
C Output:      .
C

```

```

C*****

```

```

C Modifications: (STATUS: P - PROPOSED; R - REQUIRED ; C - COMPLETED)
C
C Number   Status   Date:           Description:           Initials
C -----
C    01      C      05/31/90  Modified directory changes.      BAW
C
C*****

```

SUBROUTINE ORCT

C Global Variables

```

    DIMENSION ORCL(18,18),ORCLBR(18)
    CHARACTER*1 OCHR(110)
    CHARACTER*2 ORCLBR
    CHARACTER*110 ORCHR
    REAL ORCL,RETDP
    INTEGER I,L,M

```

```

    COMMON/PASRTD/RETDP
    COMMON/OCLST/ORCL,ORCLBR
    EQUIVALENCE (OCHR(1),ORCHR)

```

C Local Variables

```

    I = 0
    L = 0
    M = 0
    SUM = 0

```

```

    10 OPEN(5,FILE='/home/warpam/iofiles/ORCLSPER.TBL',STATUS='OLD')
    READ(5,'(110(A1))',ERR=99,END=100)OCHR
    I = I + 1

```

```

IF (I.LT.3)GOTO 10
M = M + 1
ORCLBR(M) = ORCHR(1:2)
DO 11 J = 1,110
  IF (J.LT.5)GOTO 11
  IF ((J.GT.8).AND.(J.LT.11))GOTO 11
  IF ((J.GT.14).AND.(J.LT.17))GOTO 11
  IF ((J.GT.20).AND.(J.LT.23))GOTO 11
  IF ((J.GT.26).AND.(J.LT.29))GOTO 11
  IF ((J.GT.32).AND.(J.LT.35))GOTO 11
  IF ((J.GT.38).AND.(J.LT.41))GOTO 11
  IF ((J.GT.44).AND.(J.LT.47))GOTO 11
  IF ((J.GT.50).AND.(J.LT.53))GOTO 11
  IF ((J.GT.56).AND.(J.LT.59))GOTO 11
  IF ((J.GT.62).AND.(J.LT.65))GOTO 11
  IF ((J.GT.68).AND.(J.LT.71))GOTO 11
  IF ((J.GT.74).AND.(J.LT.77))GOTO 11
  IF ((J.GT.80).AND.(J.LT.83))GOTO 11
  IF ((J.GT.86).AND.(J.LT.89))GOTO 11
  IF ((J.GT.92).AND.(J.LT.95))GOTO 11
  IF ((J.GT.98).AND.(J.LT.101))GOTO 11
  IF ((J.GT.104).AND.(J.LT.107))GOTO 11
  XX = ICHAR(ORCHR(J:J))
  NUM = (79-(127-XX))
  IF (NUM.LT.0)NUM = 0
  IF ((J.GT.4).AND.(J.LT.9))THEN
    L = L + 1
    IF (J.EQ.5)THEN
      AD = NUM
      GOTO 11
    ENDIF
    IF (J.EQ.6)GOTO 11
    IF (J.EQ.7)NUM = NUM * 10
    IF (J.EQ.8)NUM = NUM * 1
    SUM = SUM + NUM
    IF (L.EQ.4)THEN
      L = 0
      ORCL(M,1) = (AD+(SUM/100.0))*RETD
      SUM = 0
    ENDIF
    GOTO 11
  ENDIF
  IF ((J.GT.10).AND.(J.LT.15))THEN
    L = L + 1
    IF (J.EQ.11)THEN
      AD = NUM
      GOTO 11
    ENDIF
    IF (J.EQ.12)GOTO 11
    IF (J.EQ.13)NUM = NUM * 10
    IF (J.EQ.14)NUM = NUM * 1
    SUM = SUM + NUM

```

```

      IF (L.EQ.4)THEN
        L = 0
        ORCL(M,2) = (AD+(SUM/100.0))*RETDP
        SUM = 0
      ENDIF
      GOTO 11
    ENDIF
  IF ((J.GT.16).AND.(J.LT.21))THEN
    L = L + 1
    IF (J.EQ.17)THEN
      AD = NUM
      GOTO 11
    ENDIF
    IF (J.EQ.18)GOTO 11
    IF (J.EQ.19)NUM = NUM * 10
    IF (J.EQ.20)NUM = NUM * 1
    SUM = SUM + NUM
    IF (L.EQ.4)THEN
      L = 0
      ORCL(M,3) = (AD+(SUM/100.0))*RETDP
      SUM = 0
    ENDIF
    GOTO 11
  ENDIF
  IF ((J.GT.22).AND.(J.LT.27))THEN
    L = L + 1
    IF (J.EQ.23)THEN
      AD = NUM
      GOTO 11
    ENDIF
    IF (J.EQ.24)GOTO 11
    IF (J.EQ.25)NUM = NUM * 10
    IF (J.EQ.26)NUM = NUM * 1
    SUM = SUM + NUM
    IF (L.EQ.4)THEN
      L = 0
      ORCL(M,4) = (AD+(SUM/100.0))*RETDP
      SUM = 0
    ENDIF
    GOTO 11
  ENDIF
  IF ((J.GT.28).AND.(J.LT.33))THEN
    L = L + 1
    IF (J.EQ.29)THEN
      AD = NUM
      GOTO 11
    ENDIF
    IF (J.EQ.30)GOTO 11
    IF (J.EQ.31)NUM = NUM * 10
    IF (J.EQ.32)NUM = NUM * 1
    SUM = SUM + NUM
    IF (L.EQ.4)THEN

```

```

        L = 0
        ORCL(M,5) = (AD+(SUM/100.0))*RETD
        SUM = 0
    ENDIF
    GOTO 11
ENDIF
IF ((J.GT.34).AND.(J.LT.39))THEN
    L = L + 1
    IF (J.EQ.35)THEN
        AD = NUM
        GOTO 11
    ENDIF
    IF (J.EQ.36)GOTO 11
    IF (J.EQ.37)NUM = NUM * 10
    IF (J.EQ.38)NUM = NUM * 1
    SUM = SUM + NUM
    IF (L.EQ.4)THEN
        L = 0
        ORCL(M,6) = (AD+(SUM/100.0))*RETD
        SUM = 0
    ENDIF
    GOTO 11
ENDIF
IF ((J.GT.40).AND.(J.LT.45))THEN
    L = L + 1
    IF (J.EQ.41)THEN
        AD = NUM
        GOTO 11
    ENDIF
    IF (J.EQ.42)GOTO 11
    IF (J.EQ.43)NUM = NUM * 10
    IF (J.EQ.44)NUM = NUM * 1
    SUM = SUM + NUM
    IF (L.EQ.4)THEN
        L = 0
        ORCL(M,7) = (AD+(SUM/100.0))*RETD
        SUM = 0
    ENDIF
    GOTO 11
ENDIF
IF ((J.GT.46).AND.(J.LT.51))THEN
    L = L + 1
    IF (J.EQ.47)THEN
        AD = NUM
        GOTO 11
    ENDIF
    IF (J.EQ.48)GOTO 11
    IF (J.EQ.49)NUM = NUM * 10
    IF (J.EQ.50)NUM = NUM * 1
    SUM = SUM + NUM
    IF (L.EQ.4)THEN
        L = 0

```

```

        ORCL(M,8) = (AD+(SUM/100.0))*RETD
        SUM = 0
    ENDIF
    GOTO 11
ENDIF
IF ((J.GT.52).AND.(J.LT.57))THEN
    L = L + 1
    IF (J.EQ.53)THEN
        AD = NUM
        GOTO 11
    ENDIF
    IF (J.EQ.54)GOTO 11
    IF (J.EQ.55)NUM = NUM * 10
    IF (J.EQ.56)NUM = NUM * 1
    SUM = SUM + NUM
    IF (L.EQ.4)THEN
        L = 0
        ORCL(M,9) = (AD+(SUM/100.0))*RETD
        SUM = 0
    ENDIF
    GOTO 11
ENDIF
IF ((J.GT.58).AND.(J.LT.63))THEN
    L = L + 1
    IF (J.EQ.59)THEN
        AD = NUM
        GOTO 11
    ENDIF
    IF (J.EQ.60)GOTO 11
    IF (J.EQ.61)NUM = NUM * 10
    IF (J.EQ.62)NUM = NUM * 1
    SUM = SUM + NUM
    IF (L.EQ.4)THEN
        L = 0
        ORCL(M,10) = (AD+(SUM/100.0))*RETD
        SUM = 0
    ENDIF
    GOTO 11
ENDIF
IF ((J.GT.64).AND.(J.LT.69))THEN
    L = L + 1
    IF (J.EQ.65)THEN
        AD = NUM
        GOTO 11
    ENDIF
    IF (J.EQ.66)GOTO 11
    IF (J.EQ.67)NUM = NUM * 10
    IF (J.EQ.68)NUM = NUM * 1
    SUM = SUM + NUM
    IF (L.EQ.4)THEN
        L = 0
        ORCL(M,11) = (AD+(SUM/100.0))*RETD

```



```

        SUM = 0
    ENDIF
    GOTO 11
ENDIF
IF ((J.GT.70).AND.(J.LT.75))THEN
    L = L + 1
    IF (J.EQ.71)THEN
        AD = NUM
        GOTO 11
    ENDIF
    IF (J.EQ.72)GOTO 11
    IF (J.EQ.73)NUM = NUM * 10
    IF (J.EQ.74)NUM = NUM * 1
    SUM = SUM + NUM
    IF (L.EQ.4)THEN
        L = 0
        ORCL(M,12) = (AD+(SUM/100.0))*RETD
        SUM = 0
    ENDIF
    GOTO 11
ENDIF
IF ((J.GT.76).AND.(J.LT.81))THEN
    L = L + 1
    IF (J.EQ.77)THEN
        AD = NUM
        GOTO 11
    ENDIF
    IF (J.EQ.78)GOTO 11
    IF (J.EQ.79)NUM = NUM * 10
    IF (J.EQ.80)NUM = NUM * 1
    SUM = SUM + NUM
    IF (L.EQ.4)THEN
        L = 0
        ORCL(M,13) = (AD+(SUM/100.0))*RETD
        SUM = 0
    ENDIF
    GOTO 11
ENDIF
IF ((J.GT.82).AND.(J.LT.87))THEN
    L = L + 1
    IF (J.EQ.83)THEN
        AD = NUM
        GOTO 11
    ENDIF
    IF (J.EQ.84)GOTO 11
    IF (J.EQ.85)NUM = NUM * 10
    IF (J.EQ.86)NUM = NUM * 1
    SUM = SUM + NUM
    IF (L.EQ.4)THEN
        L = 0
        ORCL(M,14) = (AD+(SUM/100.0))*RETD
        SUM = 0
    ENDIF

```

```

ENDIF
GOTO 11
ENDIF
IF ((J.GT.88).AND.(J.LT.93))THEN
  L = L + 1
  IF (J.EQ.89)THEN
    AD = NUM
    GOTO 11
  ENDIF
  IF (J.EQ.90)GOTO 11
  IF (J.EQ.91)NUM = NUM * 10
  IF (J.EQ.92)NUM = NUM * 1
  SUM = SUM + NUM
  IF (L.EQ.4)THEN
    L = 0
    ORCL(M,15) = (AD+(SUM/100.0))*RETD
    SUM = 0
  ENDIF
  GOTO 11
ENDIF
IF ((J.GT.94).AND.(J.LT.99))THEN
  L = L + 1
  IF (J.EQ.95)THEN
    AD = NUM
    GOTO 11
  ENDIF
  IF (J.EQ.96)GOTO 11
  IF (J.EQ.97)NUM = NUM * 10
  IF (J.EQ.98)NUM = NUM * 1
  SUM = SUM + NUM
  IF (L.EQ.4)THEN
    L = 0
    ORCL(M,16) = (AD+(SUM/100.0))*RETD
    SUM = 0
  ENDIF
  GOTO 11
ENDIF
IF ((J.GT.100).AND.(J.LT.105))THEN
  L = L + 1
  IF (J.EQ.101)THEN
    AD = NUM
    GOTO 11
  ENDIF
  IF (J.EQ.102)GOTO 11
  IF (J.EQ.103)NUM = NUM * 10
  IF (J.EQ.104)NUM = NUM * 1
  SUM = SUM + NUM
  IF (L.EQ.4)THEN
    L = 0
    ORCL(M,17) = (AD+(SUM/100.0))*RETD
    SUM = 0
  ENDIF

```

```

        GOTO 11
    ENDIF
    IF ((J.GT.106).AND.(J.LT.111))THEN
        L = L + 1
        IF (J.EQ.107)THEN
            AD = NUM
            GOTO 11
        ENDIF
        IF (J.EQ.108)GOTO 11
        IF (J.EQ.109)NUM = NUM * 10
        IF (J.EQ.110)NUM = NUM * 1
        SUM = SUM + NUM
        IF (L.EQ.4)THEN
            L = 0
            ORCL(M,18) = (AD+(SUM/100.0))*RETD
            SUM = 0
        ENDIF
        GOTO 11
    ENDIF
11  CONTINUE
    GOTO 10

99  WRITE(6,*)'ERROR READING FILE:  ORCLSPER.TBL'
100 CLOSE(5,STATUS='KEEP')

C   Exit subroutine and return to main program.

    RETURN
    END

C   END ORCLS.FOR

```

```

C*****
C
C Program Name:  ERCT                      Date:  05-23-1990
C
C File Name:     MODRCLS.FOR
C
C Programmer:    Beth White, SAIC, 749-8771
C
C Description:    Read and stores Enlisted Reclas Table toa an array.
C
C Input:         ERCLSPER.TBL
C
C Output:        .
C
C*****
C Modifications: (STATUS:  P - PROPOSED; R - REQUIRED ; C - COMPLETED)
C
C Number   Status   Date:           Description:           Initials
C -----
C    01      C      05/31/90  Modified directory changes.      BAW
C
C*****

```

SUBROUTINE ERCT

C Global Variables

```

    DIMENSION ERCL(17,17),ERCLBR(17)
    CHARACTER*1 ECHR(104)
    CHARACTER*2 ERCLBR
    CHARACTER*104 ERCHR
    REAL ERCL,RETD
    INTEGER I,L,M

```

```

    COMMON/PASRTD/RETD
    COMMON/ECLST/ERCL,ERCLBR
    EQUIVALENCE (ECHR(1),ERCHR)

```

C Local Variables

```

    I = 0
    L = 0
    M = 0
    SUM = 0

    OPEN(4,FILE='/home/warpam/iofiles/ERCLSPER.TBL',STATUS='OLD')
10  READ(4,'(104(A1))',ERR=99,END=100)ECHR
    I = I + 1
    IF (I.LT.3)GOTO 10
    M = M + 1
    ERCLBR(M) = ERCHR(1:2)
    DO 11 J = 1,104

```

```

IF (J.LT.5)GOTO 11
IF ((J.GT.8).AND.(J.LT.11))GOTO 11
IF ((J.GT.14).AND.(J.LT.17))GOTO 11
IF ((J.GT.20).AND.(J.LT.23))GOTO 11
IF ((J.GT.26).AND.(J.LT.29))GOTO 11
IF ((J.GT.32).AND.(J.LT.35))GOTO 11
IF ((J.GT.38).AND.(J.LT.41))GOTO 11
IF ((J.GT.44).AND.(J.LT.47))GOTO 11
IF ((J.GT.50).AND.(J.LT.53))GOTO 11
IF ((J.GT.56).AND.(J.LT.59))GOTO 11
IF ((J.GT.62).AND.(J.LT.65))GOTO 11
IF ((J.GT.68).AND.(J.LT.71))GOTO 11
IF ((J.GT.74).AND.(J.LT.77))GOTO 11
IF ((J.GT.80).AND.(J.LT.83))GOTO 11
IF ((J.GT.86).AND.(J.LT.89))GOTO 11
IF ((J.GT.92).AND.(J.LT.95))GOTO 11
IF ((J.GT.98).AND.(J.LT.101))GOTO 11
XX = ICHAR(ERCHR(J:J))
NUM = (79-(127-XX))
IF (NUM.LT.0)NUM = 0
IF ((J.GT.4).AND.(J.LT.9))THEN
  L = L + 1
  IF (J.EQ.5)THEN
    AD = NUM
    GOTO 11
  ENDIF
  IF (J.EQ.6)GOTO 11
  IF (J.EQ.7)NUM = NUM * 10
  IF (J.EQ.8)NUM = NUM * 1
  SUM = SUM + NUM
  IF (L.EQ.4)THEN
    L = 0
    ERCL(M,1) = (AD+(SUM/100.0))*RETD
    SUM = 0
  ENDIF
  GOTO 11
ENDIF
IF ((J.GT.10).AND.(J.LT.15))THEN
  L = L + 1
  IF (J.EQ.11)THEN
    AD = NUM
    GOTO 11
  ENDIF
  IF (J.EQ.12)GOTO 11
  IF (J.EQ.13)NUM = NUM * 10
  IF (J.EQ.14)NUM = NUM * 1
  SUM = SUM + NUM
  IF (L.EQ.4)THEN
    L = 0
    ERCL(M,2) = (AD+(SUM/100.0))*RETD
    SUM = 0
  ENDIF

```

```

      GOTO 11
ENDIF
IF ((J.GT.16).AND.(J.LT.21))THEN
  L = L + 1
  IF (J.EQ.17)THEN
    AD = NUM
    GOTO 11
  ENDIF
  IF (J.EQ.18)GOTO 11
  IF (J.EQ.19)NUM = NUM * 10
  IF (J.EQ.20)NUM = NUM * 1
  SUM = SUM + NUM
  IF (L.EQ.4)THEN
    L = 0
    ERCL(M,3) = (AD+(SUM/100.0))*RETD
    SUM = 0
  ENDIF
  GOTO 11
ENDIF
IF ((J.GT.22).AND.(J.LT.27))THEN
  L = L + 1
  IF (J.EQ.23)THEN
    AD = NUM
    GOTO 11
  ENDIF
  IF (J.EQ.24)GOTO 11
  IF (J.EQ.25)NUM = NUM * 10
  IF (J.EQ.26)NUM = NUM * 1
  SUM = SUM + NUM
  IF (L.EQ.4)THEN
    L = 0
    ERCL(M,4) = (AD+(SUM/100.0))*RETD
    SUM = 0
  ENDIF
  GOTO 11
ENDIF
IF ((J.GT.28).AND.(J.LT.33))THEN
  L = L + 1
  IF (J.EQ.29)THEN
    AD = NUM
    GOTO 11
  ENDIF
  IF (J.EQ.30)GOTO 11
  IF (J.EQ.31)NUM = NUM * 10
  IF (J.EQ.32)NUM = NUM * 1
  SUM = SUM + NUM
  IF (L.EQ.4)THEN
    L = 0
    ERCL(M,5) = (AD+(SUM/100.0))*RETD
    SUM = 0
  ENDIF
  GOTO 11

```

```

ENDIF
IF ((J.GT.34).AND.(J.LT.39))THEN
  L = L + 1
  IF (J.EQ.35)THEN
    AD = NUM
    GOTO 11
  ENDIF
  IF (J.EQ.36)GOTO 11
  IF (J.EQ.37)NUM = NUM * 10
  IF (J.EQ.38)NUM = NUM * 1
  SUM = SUM + NUM
  IF (L.EQ.4)THEN
    L = 0
    ERCL(M,6) = (AD+(SUM/100.0))*RETD
    SUM = 0
  ENDIF
  GOTO 11
ENDIF
IF ((J.GT.40).AND.(J.LT.45))THEN
  L = L + 1
  IF (J.EQ.41)THEN
    AD = NUM
    GOTO 11
  ENDIF
  IF (J.EQ.42)GOTO 11
  IF (J.EQ.43)NUM = NUM * 10
  IF (J.EQ.44)NUM = NUM * 1
  SUM = SUM + NUM
  IF (L.EQ.4)THEN
    L = 0
    ERCL(M,7) = (AD+(SUM/100.0))*RETD
    SUM = 0
  ENDIF
  GOTO 11
ENDIF
IF ((J.GT.46).AND.(J.LT.51))THEN
  L = L + 1
  IF (J.EQ.47)THEN
    AD = NUM
    GOTO 11
  ENDIF
  IF (J.EQ.48)GOTO 11
  IF (J.EQ.49)NUM = NUM * 10
  IF (J.EQ.50)NUM = NUM * 1
  SUM = SUM + NUM
  IF (L.EQ.4)THEN
    L = 0
    ERCL(M,8) = (AD+(SUM/100.0))*RETD
    SUM = 0
  ENDIF
  GOTO 11
ENDIF
ENDIF

```

```

IF ((J.GT.52).AND.(J.LT.57))THEN
  L = L + 1
  IF (J.EQ.53)THEN
    AD = NUM
    GOTO 11
  ENDIF
  IF (J.EQ.54)GOTO 11
  IF (J.EQ.55)NUM = NUM * 10
  IF (J.EQ.56)NUM = NUM * 1
  SUM = SUM + NUM
  IF (L.EQ.4)THEN
    L = 0
    ERCL(M,9) = (AD+(SUM/100.0))*RETD
    SUM = 0
  ENDIF
  GOTO 11
ENDIF
IF ((J.GT.58).AND.(J.LT.63))THEN
  L = L + 1
  IF (J.EQ.59)THEN
    AD = NUM
    GOTO 11
  ENDIF
  IF (J.EQ.60)GOTO 11
  IF (J.EQ.61)NUM = NUM * 10
  IF (J.EQ.62)NUM = NUM * 1
  SUM = SUM + NUM
  IF (L.EQ.4)THEN
    L = 0
    ERCL(M,10) = (AD+(SUM/100.0))*RETD
    SUM = 0
  ENDIF
  GOTO 11
ENDIF
IF ((J.GT.64).AND.(J.LT.69))THEN
  L = L + 1
  IF (J.EQ.65)THEN
    AD = NUM
    GOTO 11
  ENDIF
  IF (J.EQ.66)GOTO 11
  IF (J.EQ.67)NUM = NUM * 10
  IF (J.EQ.68)NUM = NUM * 1
  SUM = SUM + NUM
  IF (L.EQ.4)THEN
    L = 0
    ERCL(M,11) = (AD+(SUM/100.0))*RETD
    SUM = 0
  ENDIF
  GOTO 11
ENDIF
IF ((J.GT.70).AND.(J.LT.75))THEN

```



```

L = L + 1
IF (J.EQ.71)THEN
  AD = NUM
  GOTO 11
ENDIF
IF (J.EQ.72)GOTO 11
IF (J.EQ.73)NUM = NUM * 10
IF (J.EQ.74)NUM = NUM * 1
SUM = SUM + NUM
IF (L.EQ.4)THEN
  L = 0
  ERCL(M,12) = (AD+(SUM/100.0))*RETD
  SUM = 0
ENDIF
GOTO 11
ENDIF
IF ((J.GT.76).AND.(J.LT.81))THEN
  L = L + 1
  IF (J.EQ.77)THEN
    AD = NUM
    GOTO 11
  ENDIF
  IF (J.EQ.78)GOTO 11
  IF (J.EQ.79)NUM = NUM * 10
  IF (J.EQ.80)NUM = NUM * 1
  SUM = SUM + NUM
  IF (L.EQ.4)THEN
    L = 0
    ERCL(M,13) = (AD+(SUM/100.0))*RETD
    SUM = 0
  ENDIF
  GOTO 11
ENDIF
IF ((J.GT.82).AND.(J.LT.87))THEN
  L = L + 1
  IF (J.EQ.83)THEN
    AD = NUM
    GOTO 11
  ENDIF
  IF (J.EQ.84)GOTO 11
  IF (J.EQ.85)NUM = NUM * 10
  IF (J.EQ.86)NUM = NUM * 1
  SUM = SUM + NUM
  IF (L.EQ.4)THEN
    L = 0
    ERCL(M,14) = (AD+(SUM/100.0))*RETD
    SUM = 0
  ENDIF
  GOTO 11
ENDIF
IF ((J.GT.88).AND.(J.LT.93))THEN
  L = L + 1

```

```

        IF (J.EQ.89)THEN
            AD = NUM
            GOTO 11
        ENDIF
        IF (J.EQ.90)GOTO 11
        IF (J.EQ.91)NUM = NUM * 10
        IF (J.EQ.92)NUM = NUM * 1
        SUM = SUM + NUM
        IF (L.EQ.4)THEN
            L = 0
            ERCL(M,15) = (AD+(SUM/100.0))*RETDP
            SUM = 0
        ENDIF
        GOTO 11
    ENDIF
    IF ((J.GT.94).AND.(J.LT.99))THEN
        L = L + 1
        IF (J.EQ.95)THEN
            AD = NUM
            GOTO 11
        ENDIF
        IF (J.EQ.96)GOTO 11
        IF (J.EQ.97)NUM = NUM * 10
        IF (J.EQ.98)NUM = NUM * 1
        SUM = SUM + NUM
        IF (L.EQ.4)THEN
            L = 0
            ERCL(M,16) = (AD+(SUM/100.0))*RETDP
            SUM = 0
        ENDIF
        GOTO 11
    ENDIF
    IF ((J.GT.100).AND.(J.LT.105))THEN
        L = L + 1
        IF (J.EQ.101)THEN
            AD = NUM
            GOTO 11
        ENDIF
        IF (J.EQ.102)GOTO 11
        IF (J.EQ.103)NUM = NUM * 10
        IF (J.EQ.104)NUM = NUM * 1
        SUM = SUM + NUM
        IF (L.EQ.4)THEN
            L = 0
            ERCL(M,17) = (AD+(SUM/100.0))*RETDP
            SUM = 0
        ENDIF
        GOTO 11
    ENDIF
11  CONTINUE
    GOTO 10

```

```
99 WRITE(6,*)'ERROR READING FILE: ERCLSPER.TBL'  
100 CLOSE(4,STATUS='KEEP')
```

```
C Exit subroutine and return to main program.
```

```
RETURN  
END
```

```
C END ERCLS.FOR
```

```

C*****
C
C Program Name:  RDLY                      Date:  05-23-1990
C
C File Name:    MODRCLS.FOR
C
C Programmer:   Beth White, SAIC, 749-8771
C
C              Reads and stores time periods to delay return of
C              TRD and percentage into each time period.
C
C Input:        RCLSDLY.TBL
C
C Output:       .
C
C*****
C Modifications: (STATUS:  P - PROPOSED; R - REQUIRED ; C - COMPLETED)
C
C Number   Status   Date:           Description:           Initials
C -----
C    01      C      05/31/90  Modified directory changes.      BAW
C
C*****

```

SUBROUTINE RDLY

C Global Variables

```

    DIMENSION RDLAY(18,6)
    CHARACTER*1 RCHR(62)
    CHARACTER*62 RCCHR
    REAL RDLAY
    INTEGER I,J,L,M

    COMMON/RCDLY/RDLAY
    EQUIVALENCE (RCHR(1),RCCHR)

```

C Local Variables

```

    I = 0
    J = 0
    L = 0
    M = 0
    SUM = 0

    OPEN(3,FILE='/home/warpam/iofiles/RCLSDLY.TBL',STATUS='OLD')
10  READ(3,'(142(A1))',ERR=99,END=100)RCHR
    I = I + 1
    IF (I.LT.3)GOTO 10
    M = M + 1
    DO 30 K = 1,62
        IF (K.LT.9)GOTO 30

```

```

IF ((K.GT.12).AND.(K.LT.19))GOTO 30
IF ((K.GT.22).AND.(K.LT.29))GOTO 30
IF ((K.GT.32).AND.(K.LT.39))GOTO 30
IF ((K.GT.42).AND.(K.LT.49))GOTO 30
IF ((K.GT.52).AND.(K.LT.59))GOTO 30
XX = ICHAR(RCCHR(K:K))
NUM = (79-(127-XX))
IF ((K.GT.8).AND.(K.LT.13))THEN
  L = L + 1
  IF (K.EQ.9)THEN
    AD = NUM
    GOTO 30
  ENDIF
  IF (K.EQ.10)GOTO 30
  IF (K.EQ.11)NUM = NUM * 10
  IF (K.EQ.12)NUM = NUM * 1
  SUM = SUM + NUM
  IF (L.EQ.4)THEN
    L = 0
    SUM = AD + (SUM/100.0)
    RDLAY(M,1) = SUM
    SUM = 0
  ENDIF
  GOTO 30
ENDIF
IF ((K.GT.18).AND.(K.LT.23))THEN
  L = L + 1
  IF (K.EQ.19)THEN
    AD = NUM
    GOTO 30
  ENDIF
  IF (K.EQ.20)GOTO 30
  IF (K.EQ.21)NUM = NUM * 10
  IF (K.EQ.22)NUM = NUM * 1
  SUM = SUM ÷ NUM
  IF (L.EQ.4)THEN
    L = 0
    SUM = AD + (SUM/100.0)
    RDLAY(M,2) = SUM
    SUM = 0
  ENDIF
  GOTO 30
ENDIF
IF ((K.GT.28).AND.(K.LT.33))THEN
  L = L + 1
  IF (K.EQ.29)THEN
    AD = NUM
    GOTO 30
  ENDIF
  IF (K.EQ.30)GOTO 30
  IF (K.EQ.31)NUM = NUM * 10
  IF (K.EQ.32)NUM = NUM * 1

```

```

SUM = SUM + NUM
IF (L.EQ.4)THEN
  L = 0
  SUM = AD + (SUM/100.0)
  RDLAY(M,3) = SUM
  SUM = 0
ENDIF
GOTO 30
ENDIF
IF ((K.GT.38).AND.(K.LT.43))THEN
  L = L + 1
  IF (K.EQ.39)THEN
    AD = NUM
    GOTO 30
  ENDIF
  IF (K.EQ.40)GOTO 30
  IF (K.EQ.41)NUM = NUM * 10
  IF (K.EQ.42)NUM = NUM * 1
  SUM = SUM + NUM
  IF (L.EQ.4)THEN
    L = 0
    SUM = AD + (SUM/100.0)
    RDLAY(M,4) = SUM
    SUM = 0
  ENDIF
  GOTO 30
ENDIF
IF ((K.GT.48).AND.(K.LT.53))THEN
  L = L + 1
  IF (K.EQ.49)THEN
    AD = NUM
    GOTO 30
  ENDIF
  IF (K.EQ.50)GOTO 30
  IF (K.EQ.51)NUM = NUM * 10
  IF (K.EQ.52)NUM = NUM * 1
  SUM = SUM + NUM
  IF (L.EQ.4)THEN
    L = 0
    SUM = AD + (SUM/100.0)
    RDLAY(M,5) = SUM
    SUM = 0
  ENDIF
  GOTO 30
ENDIF
IF ((K.GT.58).AND.(K.LT.63))THEN
  L = L + 1
  IF (K.EQ.59)THEN
    AD = NUM
    GOTO 30
  ENDIF
  IF (K.EQ.60)GOTO 30

```

```

        IF (K.EQ.61)NUM = NUM * 10
        IF (K.EQ.62)NUM = NUM * 1
        SUM = SUM + NUM
        IF (L.EQ.4)THEN
            L = 0
            SUM = AD + (SUM/100.0)
            RDLAY(M,6) = SUM
            SUM = 0
        ENDIF
        GOTO 30
    ENDIF
30  CONTINUE
    GOTO 10

99  WRITE(6,*)'ERROR READING FILE:  RCLSDLY.TBL'
100 CLOSE(3,STATUS='KEEP')

C    Exit subroutine and return to main program.

    RETURN
    END

C    END RCDLY.FOR

```

```

C*****
C
C Program Name:  WARPRI                      Date:  05-22-1990
C
C File Name:    MODRCLS.FOR
C
C Programmer:   Beth White, SAIC, 749-8771
C
C Description:  Read and stores Warpam Branch Priority Table to
C               an array.
C
C Input:        WARPRI.TBL
C
C Output:       .
C
C*****
C Modifications: (STATUS:  P - PROPOSED; R - REQUIRED ; C - COMPLETED)
C
C Number   Status   Date:           Description:           Initials
C -----
C    01      C      05/31/90  Modified directory changes.      BAW
C
C*****

```

SUBROUTINE WARPRI

C Global Variables

```

    DIMENSION WPRNUMB(67),WPRCODE(67)
    CHARACTER*1 WCHR(8)
    CHARACTER*2 WNUM,WPRNUMB
    CHARACTER*5 WBRH,WPRCODE
    CHARACTER*8 XCHR
    INTEGER I

```

```

    COMMON/WARPR/WPRNUMB,WPRCODE
    EQUIVALENCE (WCHR(1),XCHR)

```

C Local Variables

```

    I = 0

    OPEN(5,FILE='/home/warpam/iofiles/WARPRI.TBL',STATUS='OLD')
10  READ(5,'(8(A1))',ERR=99,END=100)WCHR
    I = I + 1
    WNUM = XCHR(1:2)
    WBRH = XCHR(4:8)
    WPRNUMB(I) = WNUM
    WPRCODE(I) = WBRH
    GOTO 10

99  WRITE(6,*)'ERROR READING FILE:  WARPRI.TBL'

```


100 CLOSE(5,STATUS='KEEP')

C Exit subroutine and return to main program.

RETURN
END

C END WARP.FOR

```

C*****
C
C Program Name:  CNVRTP                      Date:  06-05-1990
C
C File Name:     MODRCLS.FOR
C
C Programmer:    Beth White, SAIC, 749-8771
C
C Description:    Converts time period from characters to an numeric
C                  value [VTP].
C
C Input:         .
C
C Output:        .
C
C*****
C Modifications: (STATUS:  P - PROPOSED; R - REQUIRED ; C - COMPLETED)
C
C Number   Status  Date:           Description:           Initials
C -----
C
C*****

```

SUBROUTINE CNVRTP

C Global Variables

```

CHARACTER*1 RQCHR(40)
CHARACTER*40 RACHR

INTEGER VTP, IC, II, XX, NUM

COMMON/CVRTT/RQCHR, VTP, VBRNH, VSTRNG

EQUIVALENCE (RQCHR(1), RACHR)

```

C Local Variables

```

VTP = 0
NUM = 0
II = 22

```

C Begin Subroutine CNVRTP

```

DO 100 IC = 1, 2
  II = II + 1
  XX = ICHAR(RACHR(II:II))
  NUM = (79 - (127 - XX))
  IF (NUM.LT.0) NUM = 0
  IF (IC.EQ.1) NUM = NUM * 10
  IF (IC.EQ.2) NUM = NUM * 1

```

VTP = VTP + NUM
100 CONTINUE

C Exit subroutine and return to main program.

RETURN
END

C END SUBROUTINE CNVRTP

```

C*****
C
C Program Name:  CNVRBR                      Date:  06-05-1990
C
C File Name:    MODRCLS.FOR
C
C Programmer:    Beth White, SAIC, 749-8771
C
C Description:   Converts variable branch [VBRR] from character to an
C               numeric value [VBRNH].
C
C Input:        .
C
C Output:       .
C
C*****
C Modifications: (STATUS:  P - PROPOSED; R - REQUIRED ; C - COMPLETED)
C
C Number   Status   Date:           Description:           Initials
C -----
C
C
C*****
C
C      SUBROUTINE CNVRBR
C
C      Global Variables
C
C      CHARACTER*1 RQCHR(40)
C      CHARACTER*40 RACHR
C
C      INTEGER VBRNH, IC, II, XX, NUM
C
C      COMMON/CVRTT/RQCHR, VTP, VBRNH, VSTRNG
C
C      EQUIVALENCE (RQCHR(1), RACHR)
C
C      Local Variables
C
C      VBRNH = 0
C      NUM = 0
C      II = 24
C
C      Begin Subroutine CNVRBR
C
C      DO 100 IC = 1,3
C        II = II + 1
C        XX = ICHAR(RACHR(II:II))
C        NUM = (79-(127-XX))
C        IF (NUM.LT.0)NUM = 0
C        IF (IC.EQ.1)NUM = NUM * 100
C        IF (IC.EQ.2)NUM = NUM * 10

```

```
        IF (IC.EQ.3)NUM = NUM * 1  
        VBRNH = VBRNH + NUM  
100  CONTINUE  
  
C    Exit subroutine and return to main program.  
  
        RETURN  
        END  
  
C    END SUBROUTINE CNVRBR
```

```

C *****
C
C Program Name:  CNVRSTR                      Date:  06-05-1990
C
C File Name:    MODRCLS.FOR
C
C Programmer:   Beth White, SAIC, 749-8771
C
C Description:  Converts variable strength [VSTR] from character to
C               an numeric value [VSTRNG].
C
C Input:       .
C
C Output:      .
C
C *****
C Modifications: (STATUS:  P - PROPOSED; R - REQUIRED ; C - COMPLETED)
C
C Number   Status   Date:           Description:           Initials
C -----
C
C *****

```

SUBROUTINE CNVRSTR

C Global Variables

```

CHARACTER*1 RQCHR(40)
CHARACTER*40 RACHR

INTEGER VSTRNG, IC, II, XX, NUM

COMMON/CVRTT/RQCHR, VTP, VBRNH, VSTRNG

EQUIVALENCE (RQCHR(1), RACHR)

```

C Local Variables

```

VSTRNG = 0
NUM = 0
II = 34

```

C Begin Subroutine CNVRSTR

```

DO 100 IC = 1, 6
  II = II + 1
  XX = ICHAR(RACHR(II:II))
  NUM = (79 - (127 - XX))
  IF (NUM.LT.0) NUM = 0
  IF (IC.EQ.1) NUM = NUM * 100000
  IF (IC.EQ.2) NUM = NUM * 10000

```

```
      IF (IC.EQ.3)NUM = NUM * 1000
      IF (IC.EQ.4)NUM = NUM * 100
      IF (IC.EQ.5)NUM = NUM * 10
      IF (IC.EQ.6)NUM = NUM * 1
      VSTRNG = VSTRNG + NUM
100  CONTINUE
```

C Exit subroutine and return to main program.

```
      RETURN
      END
```

C END SUBROUTINE CNVRSTR

```

C*****
C*****
C
C PROGRAMMER:  JOHN A. TENSHAW
C COMPANY      :  SAIC
C ADDRESS      :  1710 GOODRIDGE DR. MS-T-1-7-2, MCLEAN, VA. 22102
C PHONE        :  703-734-5584
C DATE         :  5 MAY 90
C
C SUBROUTINE SORTARRAY
C This subroutine uses a shell sort to sort OUTSTR BY time per/
C priority (elem 10-18) in ascending order.
C
C*****

```

SUBROUTINE SORTARRAY

```

DIMENSION OUTSTR(40000)
CHARACTER*24 OUTSTR,TEMP

```

```

LOGICAL INORDR

```

```

INTEGER CNT1,CTR,NDELTA

```

```

COMMON/JTSRT/OUTSTR,CNT1

```

```

NDELTA = CNT1
800 IF (NDELTA.GT.1)THEN
    NDELTA = NDELTA/2
810  INORDR = .TRUE.
    DO 820, CTR = 1, CNT1 - NDELTA
        IF (OUTSTR(CTR)(10:18).GT.OUTSTR(CTR+NDELTA)(10:18))THEN
            TEMP = OUTSTR(CTR)
            OUTSTR(CTR) = OUTSTR(CTR+NDELTA)
            OUTSTR(CTR+NDELTA) = TEMP
            INORDR = .FALSE.
        ENDIF
820  CONTINUE
    IF (.NOT.INORDR)GOTO 810
    GOTO 800
ENDIF

```

```

C Exit subroutine
  RETURN
  END

```


SECTION 6 CONUS REPLACEMENT CENTER / OCONUS REPLACEMENT CO. (CRC) MODEL

6.1 GENERAL

The CRC Model is designed to represent the flow of personnel replacements through a CONUS CRC or OCONUS Replacement Battalion. The model, designed in FORTRAN and SLAM II, depicts the micro-level flow of personnel through the various stations in the replacement facilities over a number of time periods to meet a specific requirement designated by the user. Statistics are provided for both the operation of the replacement facility and the macro-level flow through the system. The first time period of the model is designed to represent the buildup of personnel in the system. Accordingly, there is no output from the system until the first person or groups has completed processing the entire system. Time periods 2 through 18 are designed to represent a steady-state operation. Under these conditions, personnel exit the process as soon as the time period begins to represent those personnel in the system at the end of the last time period.

6.2 INITIATION

The CRC Model is initiated through user input from a Sun window which activates the SLAM II and FORTRAN programs. Programs may be initiated, viewed or modified by bypassing the sunview option and accessing the programs directly using the Sun VI editor. If the required files are in place and the user is prepared to proceed, the user must type "go" on the response line to advance to the first input variable. This input line ONLY ACCEPTS the word "go" in lower case letters.

6.3 INPUT FILES

Files required to operate the specific routines and sub-routines are listed in the programs below.

6.4 INPUT VARIABLES

The user is prompted by the input screen to input the following input variables on a response line: (input variables from previous runs are shown on the input screen prior to the first response)

Requirement File: Which of the various requirement files does the user desire to use for this run of the model. The available requirement files are listed at the input line. The Reclassification Model will not accept the MAX requirement file as an input and will run the DEG file in its place.

End Time Period: A time period is 10 days. The model will only run for the time periods inputted. The user must input the end time period for each run. The model always starts with time period one due to the input data (all

MOBMAN assets are in time period one at the beginning of the model run). If end time period is "10" is selected the model will run time periods "1-10" inclusive. WARPAM will produce a statistical analysis of the processing times and queues for each of these time periods.

Branch: Branch represents the specialties/MOS and grade combinations which have been grouped together in the preprocessor. These branches are then prioritized in the Branch Look-Up Table and given a priority number. The user should consult the current table in the preprocessor to determine the priority code for specific branches. The model can be run with one up to the maximum number of branches which were created in the preprocessor. The initial version of WARPAM has 67 branch/grade combinations.

CRC or Replacement Co: The user must select either a CRC or an OCONUS Replacement Company operation to model. The CRC model does not process Theater Return-To-Duty personnel, but reduces the requirement by an equal amount to account for these personnel being supplied from within a theater. The CRC model also increases the requirement for each branch to offset transient casualties based on a user inputted attrition factor.

Attrition Factor: When operated as a CRC model, the requirement for each branch for each time period is increased to account for transient replacement casualties. The user is asked to enter a rate (percentage) which the model uses to calculate this increase. The response line will accept a percentage ranging from .1% (.001) to 99.9% (.999). This response should be typed in decimal format (eg .04). However, the model will accept integers and transforms these to decimal input internally.

SLAM II Variables: The Programmers manual should be consulted for desired changes in the SLAM section of the model. Specific variables which must be considered in operating the model which are situated with SLAM are:

Time Constraint: The CRC model will suspend operation for a time period if the specified time for a time period has elapsed.

Transportation Constraint: The CRC model will suspend operation for a time period if the specified transportation assets for a time period have been expended.

6.5 PROCESSING

The CRC model is the most intricate of the WARPAM models. The program is initiated through a FORTRAN routine which prompts a second routine to produce an assets file based on the requirements file selected by the user. This program reads the MODRQAST.TBL, determines the requirement and then builds a file with line entries consisting of single asset types. The assets file consists of as many asset type entries as necessary to sum to the exact requirements for each branch per time period. SLAM terminates the processing when either the time limit for the period is reached, there are no entities in the system, or, if transportation constraints have been invoked, there are no remaining transportation assets. At the completion of the SLAM processing

cycle a set of statistics for the time period labeled either CRC(REQ FILE NAME)(TIME PERIOD).OUT or RPL with the same format (eg. crcl.out, rpl2.out). An example of this format is at Annex C. The model then has a build-in delay, currently set at approximately nine minutes to allow this stats table to be built prior to the next time period run. The next FORTRAN program prompted is the UPDATE sub-routine which encompasses shifting unfilled requirements and used assets to the next time period so that neither is lost through the process. When completed and there is a remaining time period, the opening FORTRAN routine is prompted and the total cycle begins again. This processing flow through the FORTRAN routines is shown in figure 9. The SLAM processing flow is depicted in a later section.

6.6 OUTPUT REPORTS

The output file from the CRC/RPLCO model is the MODREQAST.TBL modified by the addition of two columns. The first column is the number of requirements satisfied and is equal to the sum of the second column (assets used) for each branch per time period. Output reports from the CRC model may be view by using either the VI editor on the Sun or Dbase III Plus on a standard PC in a LAN configuration with the workstation or in the Sun DOS window program as described in section 9. The system was designed to have the output programs translated to Dbase III formats, thereby allowing the user to manipulate the data as desired.

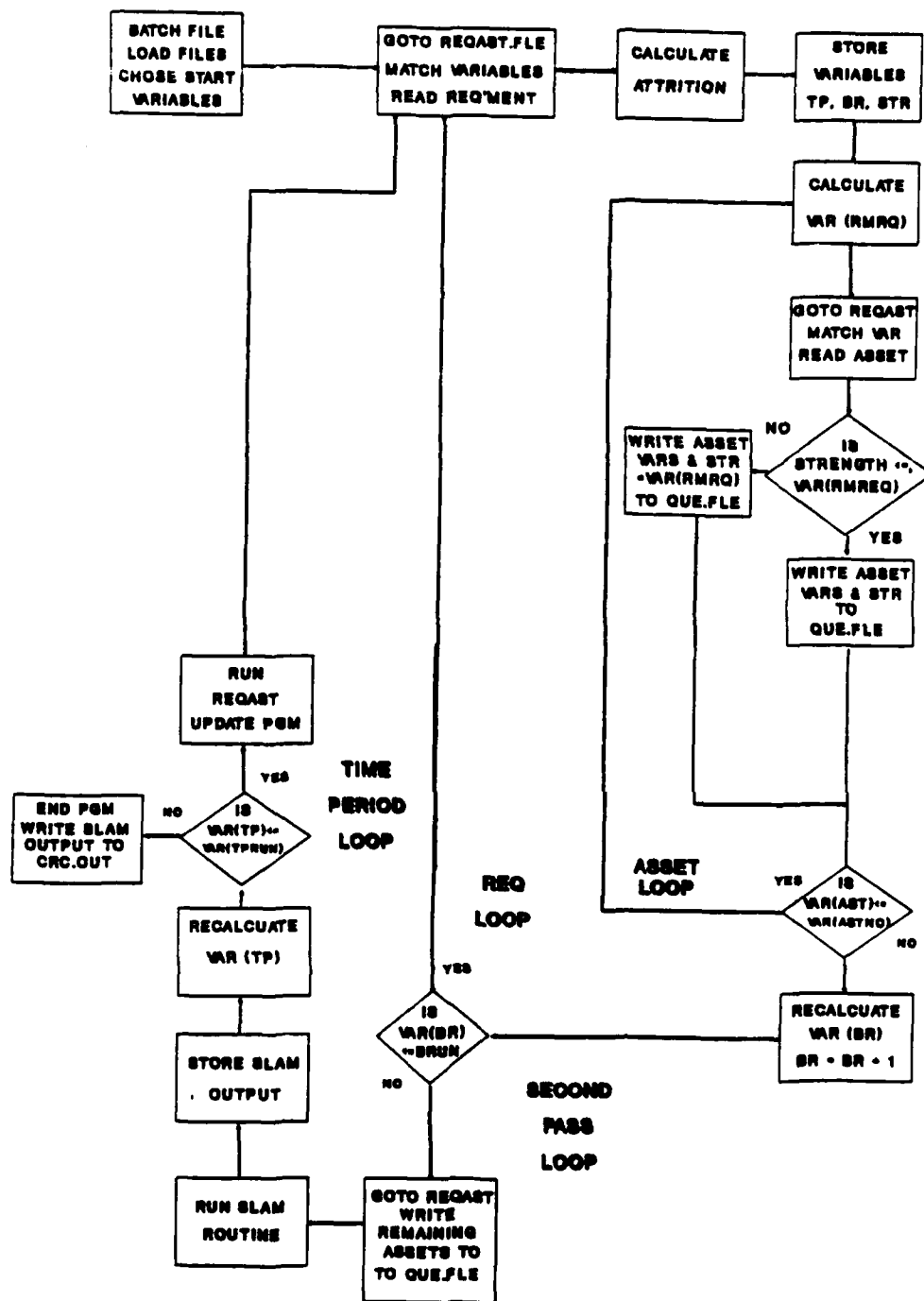


FIGURE 9: CRC FORTRAN PROCESSING

6.7 CRC/RPLBN MODEL--GENERAL FORTRAN PROGRAMS

```
*****
*      Program Name:      CRCRUN                      Date:
*                                                                06-06-1990
*
*      File Name:        CRC.FOR
*
*      Programmer:       John A. Tenshaw, SAIC, 703-734-5584
*                        Beth A. White Wilson, SAIC, 703-749-8771
*
*      Description:      Reads output file: MODRQAST.OUT and queues in all
*                        assets associated with the entered requirement for
*                        n - number of time periods and branches.
*
*      Input:            MODRQAST.OUT / MODRQAST.TMP {Temp file}
*
*      Output:           CRC.OUT
*                        RPL.OUT      Blanks filled with VAR
*                        CRCUSR.OUT   (requirement input)
*
*****
```

```
*****
*      Modifications:    (STATUS: P - PROPOSED; R - REQUIRED; C - COMPLETED)
*      Number Status Date:      Description:      Initials
*
*      01      C      12-18-90 Modified subroutine DROPLINE BAW
*                        and UPDATEFILE.
*      02      C      01-30-91 Added system call statements BAW
*                        to call SLAM for execution
*                        CRC processing.
*
*****
```

PROGRAM CRCRUN

```
*      Global Variables
```

```
DIMENSION CRCTRANS(18,67,7,8),CBGTRANS(18,67,7)
```

```
CHARACTER*1 UNPT(55),RQCHR(40),SEXX
CHARACTER*24 fdate
```

```
CHARACTER RDRESP*3, SREQ*3, IFR*3, ROM*3, VBRR*3 CHARACTER TYP*3, XF1*6,
ORGSTR*6, NEWSTR*6,CURNTP*2,STARTP*2 CHARACTER TP*2, XTEND*4, CATBRG*5,
TPRI*9, XF2*10, ENDTP*2 CHARACTER XTND*21, FLNAM*31, RACHR*40,
USRPT*55,MAXTP*2 CHARACTER HEAD0*59, HEAD1*55,
HEAD2*58,STARTBR*2,ENDBR*2
*      CHARACTER LINE1*64,LINE2*41
```

```
REAL RATTR,ATRFAC,CRC1TIM,CRC2TIM,CRC3TIM,CRC4TIM,CRC5TIM,
&CRC6TIM,CRC7TIM,CRC8TIM,CRC1DLY,CRC2DLY,CRC3DLY,CRC4DLY,
&CRC5DLY,CRC6DLY,CRC7DLY,CRC8DLY,C1AVGT1M,C2AVGT1M,C3AVGT1M,
&C4AVGT1M,C5AVGT1M,C6AVGT1M,C7AVGT1M,C8AVGT1M,C1AVGDLY,
&C2AVGDLY,C3AVGDLY,C4AVGDLY,C5AVGDLY,C6AVGDLY,C7AVGDLY,
```

&C8AVGDLY

INTEGER ASETSU,ASTCNT,USR1,USR2,CRCTRANS,CBGTRANS,CRC1CNT,

&CRC2CNT,CRC3CNT,CRC4CNT,CRC5CNT,CRC6CNT,CRC7CNT,CRC8CNT
LOGICAL THERE

COMMON/CRARY/CRCTRANS,CBGTRANS,CRC1CNT,CRC2CNT,CRC3CNT,CRC4CNT,
&CRC5CNT,CRC6CNT,CRC7CNT,CRC8CNT,CRC1TIM,CRC2TIM,CRC3TIM,CRC4TIM,
&CRC5TIM,CRC6TIM,CRC7TIM,CRC8TIM,CRC1DLY,CRC2DLY,CRC3DLY,CRC4DLY,
&CRC5DLY,CRC6DLY,CRC7DLY,CRC8DLY,C1AVGTIM,C2AVGTIM,C3AVGTIM,
&C4AVGTIM,C5AVGTIM,C6AVGTIM,C7AVGTIM,C8AVGTIM,C1AVGDLY,C2AVGDLY,
&C3AVGDLY,C4AVGDLY,C5AVGDLY,C6AVGDLY,C7AVGDLY,C8AVGDLY
EQUIVALENCE (UNPT(1),USRPT)
EQUIVALENCE (RQCHR(1),RACHR)

* Initialize Variables

CURNTP = '00'
ENDTP = '00',
STARTBR = '00'
ENDBR = '00',
ASTCNT = 0

DO 3 I=1,18
DO 4 J=1,67
DO 5 K=1,7
CBGTRANS(I,J,K)=0
DO 7 L=1,8
CRCTRANS(I,J,K,L)=0
7 CONTINUE
5 CONTINUE
4 CONTINUE
3 CONTINUE

* Begin Menu Screen

WRITE(6,10)
10 FORMAT(/////20X, '*****',
&/20X, '*****',//20X,
&' WARPAM CRC MODEL ',//20X,
&'*****',/20X,
&'*****',/////20X,
&,'THE FOLLOWING FILES ARE NEEDED:',/30X,
&'MODRQAST.OUT',/////////))

PAUSE

WRITE(6,11)
11 FORMAT(//////////)

* Checks to see if input and output files exist. If input files
* does not exist; an error message is written and the program is
* terminated. If output file exists; the old output file is deleted.

* If output file: CRCUSR.OUT does not exist, the new user inputs
 * will be appended to the CRCUSR.OUT file and displayed at the
 * screen.

```
INQUIRE(FILE=,/home/warpam/iofiles/MODRQAST.OUT',EXIST=THERE)
IF (.NOT.THERE)THEN
  WRITE(6,*)' ERROR - MODRQAST.OUT does not exist.'
```

```
GOTO 999
```

```
ENDIF
```

```
INQUIRE(FILE=,/home/warpam/iofiles/MODRQAST.TMP',EXIST=THERE)
IF (THERE)THEN
  OPEN(80,FILE=,/home/warpam/iofiles/MODRQAST.TMP',STATUS='OLD')
  CLOSE(80,STATUS='DELETE,')
ENDIF
```

```
INQUIRE(FILE=,/home/warpam/iofiles/CRCUSR.OUT',EXIST=THERE)
IF (THERE)THEN
  WRITE(6,12)
12  FORMAT(/5X,'The CRCUSR.OUT files already exists.',/5X,
  &'The following screen shows the previous input values.',/))
  OPEN(81,FILE=,/home/warpam/iofiles/CRCUSR.OUT',STATUS='OLD')
13  READ(81,'(55(A1))',ERR=15,END=16)UNPT
  WRITE(6,14)UNPT
14  FORMAT(55(A1))
  GOTO 13
15  WRITE(6,*,)' ERROR READING FILE: CRCUSR.OUT,
16  CLOSE(81,STATUS='KEEP,')
  OPEN(81,FILE=,/home/warpam/iofiles/CRCUSR.OUT',ACCESS='APPEND',
  &STATUS='OLD')
  ELSE
    OPEN(81,FILE=,/home/warpam/iofiles/CRCUSR.OUT',STATUS='NEW')
  ENDF
```

* Input Variable (Start Requirement).

* The default requirement is [DEG].

```
WRITE(6,17)
17  FORMAT(/10X,,ENTER START REQUIREMENT,,/10X,
  &'REQUIREMENTS: (MAX,DEG,AE1,AKO,ASW,CST,CSB)')
  SREQ = 'XXX'
  READ(*,*)RDRESP
```

```
IF ((RDRESP.EQ.'MAX').OR.(RDRESP.EQ.'max'))SREQ = 'MAX'
IF ((RDRESP.EQ.'DEG').OR.(RDRESP.EQ.'deg'))SREQ = 'DEG'
IF ((RDRESP.EQ.'AE1').OR.(RDRESP.EQ.'ae1'))SREQ = 'AE1'
IF ((RDRESP.EQ.'AKO').OR.(RDRESP.EQ.'ako'))SREQ = 'AKO'
IF ((RDRESP.EQ.'ASW').OR.(RDRESP.EQ.'asw'))SREQ = 'ASW'
IF ((RDRESP.EQ.'CST').OR.(RDRESP.EQ.'cst'))SREQ = 'CST'
IF ((RDRESP.EQ.'CSB').OR.(RDRESP.EQ.'csb'))SREQ = 'CSB'
IF (SREQ.EQ.,XXX')THEN
```

```

        SREQ = 'DEG'
        WRITE(6,18)
18      FORMAT(/15X,'ERROR - INVALID REQUIREMENT',/15X,
      & 'DEFAULT START REQUIREMENT WILL BE USED: SREQ = DEG')
      ENDIF

*   Input Variable (Start Time Period and End Time Period).

      STARTP = ,01'
      USR1 = 1

      WRITE(6,22)
22     FORMAT(/10X,'ENTER END TIME PERIOD (1-18) )

      READ(*,*)USR2
      IF ((USR2.GT.0).AND.(USR2.LT.19).AND.(USR1.LE.USR2))THEN
        CALL INT2STR (USR2, ENDTP)
      ELSE
        ENDTP = '18'
        WRITE(6,23)
23     FORMAT(/15X,'ERROR - INVALID END TIME PERIOD',/15X,
      & ,DEFAULT END TIME PERIOD WILL BE USED: ENDTP = 18')
      ENDIF

*   Input Variable (Start Branch and End Branch).
      WRITE(6,24)
24     FORMAT(/10X,'ENTER START BRANCH (1-67),)
      READ(*,*)USR1
      IF ((USR1.GT.0).AND.(USR1.LT.68))THEN
        CALL INT2STR (USR1, STARTBR)
      ELSE
        STARTBR = '01'
        WRITE(6,25)
25     FORMAT(/15X,'ERROR - INVALID START BRANCH NO.,,/15X,
      & 'DEFAULT START BRANCH NO. WILL BE USED:
      ENDIF STARTBR = 1')

      WRITE(6,26)
26     FORMAT(/10X,'ENTER END BRANCH (1-67),)
      READ(*,*)USR2
      IF ((USR2.GT.0).AND.(USR2.LT.68).AND.(USR1.LE.USR2)) THEN
        CALL INT2STR (USR2, ENDBR)
      ELSE
        ENDBR = ,67'
        WRITE(6,27)
27     FORMAT(/15X,'ERROR - INVALID END BRANCH NO.,,/15X,
      & 'DEFAULT END BRANCH NO. WILL BE USED: ENDBR = 67')
      ENDIF

*   Input Replacement Facility.
28     WRITE(6,29)

```



```

29  FORMAT(/10X,,ENTER REPLACEMENT FACILITY (CRC OR RPL),)
    1FR = ,XXX'
    READ(*,*)RDRESP

    IF ((RDRESP.EQ.'CRC,').OR.(RDRESP.EQ.,crc,))1FR = ,CRC,
    IF ((RDRESP.EQ.'RPL,').OR.(RDRESP.EQ.,rp1'))1FR = ,RPL'
    IF (1FR.EQ.,XXX')THEN
        WRITE(6,30)
30    FORMAT(/10X,,ERROR:  INVALID ENTRY')
        GOTO 28
    ENDIF

* Declaration of variable to satisfy requirement
  ROM = 'REQ'

* Input Variable Attrition.
  WRITE(6,35)
35  FORMAT(/10X,'ENTER ATTRITION FACTOR',/10X,
    &'(PERCENT WILL BE TRANSLATED TO RATE IN DECIMAL FORM)',
    &/10X,,(NORMAL ATTRITION IS LESS THAN 4%))'
    READ(*,*)RATTR
    IF ((RATTR.GT.0.0).AND.(RATTR.LT.1.0))ATRFAC = RATTR
    IF ((RATTR.GT.0.0).AND.(RATTR.LT.101.0))ATRFAC = (RATTR)/100.0

* Write user inputs to screen and file:  CRCUSR.OUT
  WRITE(81,36)STARTP,ENDTP,SREQ,ATRFAC,STARTBR,ENDBR,1FR,ROM,fdate()
36  FORMAT(2X,A2,2X,A2,2X,A3,2X,F4.2,2X,A2,2X,A2,3X,A3,3X,A3,3X,A24)
    CLOSE(81,STATUS='KEEP,')

    WRITE(6,37)fdate(),STARTP,ENDTP,SREQ,ATRFAC,STARTBR,ENDBR,1FR,ROM
37  FORMAT(/10X,,USER INPUT(S): ',//12X,A24,//12X,
    &'START TIME PERIOD ---> ',A2,/12X,
    &'END TIME PERIOD ---> ',A2,/12X,
    &'START REQUIREMENT ---> ',A3,/12X,
    &'ATTRITION FACTOR ---> ',F4.2,/12X,
    &'START BRANCH NO. ---> ',A2,/12X,
    &'END BRANCH NO. ---> ',A2,/12X,
    &'REPLMNT FACILITY ---> ',A3,/12X,
    &'REQ or MAX ---> ',A3,////)

* Initialize CURNTP [current time period].
  CURNTP = STARTP

* Checks to see if output file exists.  If output file does exist,
* the old output file is deleted.  The name is created by
* concatenatin input facility requirement plus the start requirement
* and '.OUT'.  i.e.  CRCDEG.OUT, RPLDEG.OUT
* Note:  The file is created and the proper headin is appended.
  XTEND = '.OUT'
  XTND = '/home/warpam/iofiles/'
  XF1 = 1FR // SREQ
  XF2 = XF1 // XTEND

```

```

FLNAM = XTND // XF2

* Create parameters for the SYSTEM calls at the end of the program
  LINE1 = 'dos2unix ,// FLNAM // ' ' // XTND // XF1 // '.DOS'
*   LINE2 = 'chmod 777 // XTND // XF1 // ,.DOS'

  INQUIRE(FILE=FLNAM,EXIST=THEIR)
  IF (THEIR)THEN
    OPEN(78,FILE=FLNAM,STATUS='OLD')
    CLOSE(78,STATUS='DELETE')
  ENDF

* Header Information
  HEAD0=' CAT/BR   REQ/  TIME PER/  REQ',T/  REQ,,T   ASSET
  &S'
  HEAD1='  GRADE  S TYPE  PRIORITY  ASSETS  FILLED  USED'
  HEAD2='
  &'
*   BXXXXXBBBBBXXXXBBBXXXXXXXXXXXXBBBXXXXXXXXBBBXXXXXXXXBBBXXXXXX
  OPEN(78,FILE=FLNAM,STATUS='NEW')
  WRITE(78,38)HEAD0,HEAD1,HEAD2
38  FORMAT(1X,A57,/1X,A55,/1X,A58)
  CLOSE(78,STATUS='KEEP,')

*   Backing up original file: MODRQAST.OUT.
*   MODRQAST.OUT is copied into MODRQAST.TMP which will be read.
*   Backup file will contain only time periods from start time period
*   to end time period + 1 and maximum branches [ENDBR].
  WRITE(6,39)
39  FORMAT(5X,,PLEASE WAIT - Backing up MODRQAST.OUT')

  OPEN(82,FILE='/home/warpam/iofiles/MODRQAST.OUT',STATUS=,OLD')
  OPEN(80,FILE=,/home/warpam/iofiles/MODRQAST.TMP',STATUS='NEW')

  MAXTP = ENDTP
  CALL MODIFYTP (MAXTP)

*   Read MODRQAST.OUT header info.
  READ(82,'(3(A1))',ERR=42,END=43)SEXX
  READ(82,'(3(A1))',ERR=42,END=43)SEXX
  READ(82,,(3(A1))',ERR=42,END=43)SEXX

40  READ(82,'(40(A1))',ERR=42,END=43)RQCHR

  ASETSU = 0
  CATBRG = RACHR(3:7)
  SEXX = RACHR(12:12)
  TYP = RACHR(17:19)
  TPR1 = RACHR(23:31)
  NEWSTR = RACHR(35:40)
  ORGSTR = '  0'
  TP = RACHR(23:24)

```

```

        VBRR = RACHR(25:27)

        IF ((TP.LE.MAXTP).AND.(VBRR.LE.ENDBR))THEN
            IF ((TYP.EQ.SREQ).OR.(TYP.EQ.'TRD').OR.(TYP.EQ.,THS,)
&.OR.(TYP.EQ.'SEL').OR.(TYP.EQ. 1RR,).OR.(TYP.EQ.'STY')
&.OR.(TYP.EQ.,RET').OR.(TYP.EQ.,TRN'))THEN
                WRITE(80,41)CATBRG,SEXX,TYP,TPRI,NEWSTR,ORGSTR,ASETSU
41          FORMAT(2X,A5,4X,A1,4X,A3,3X,A9,3X,A6,3X,A6,3X,16)
                GOTO 40
            ENDIF
        ENDIF
        GOTO 40
42  WRITE(6,*)' ERROR READING FILE:  MODRQAST.OUT'
43  CLOSE(82,STATUS='KEEP')
    CLOSE(80,STATUS='KEEP')

    WRITE(6,49)
49  FORMAT(////////////////////////////////////)

*  Checks to see if ASETS.TMP or ASETS2.TMP (assets file) exists.
44  INQUIRE(FILE='/home/warpam/iofiles/ASETS.TMP',EXIST=THERE)
    IF (THERE)THEN
        OPEN(83,FILE='/home/warpam/iofiles/ASETS.TMP',STATUS='OLD')
        CLOSE(83,STATUS='DELETE,')
    ENDIF
    INQUIRE(FILE=,'/home/warpam/iofiles/ASETS2.TMP',EXIST=THERE)
    IF (THERE)THEN
        OPEN(31,FILE='/home/warpam/iofiles/ASETS2.TMP',STATUS='OLD')
        CLOSE(31,STATUS='DELETE,')
    ENDIF

    WRITE(6,50)
50  FORMAT(/),
    WRITE(6,*)  PROCESSING TIME PERIOD , ,CURNTP

*  Initialization of computed variables.
    CRC1CNT = 0
    CRC2CNT = 0
    CRC3CNT = 0
    CRC4CNT = 0
    CRC5CNT = 0
    CRC6CNT = 0
    CRC7CNT = 0
    CRC8CNT = 0
    CRC1TIM = 0.0
    CRC2TIM = 0.0
    CRC3TIM = 0.0
    CRC4TIM = 0.0
    CRC5TIM = 0.0
    CRC6TIM = 0.0
    CRC7TIM = 0.0
    CRC8TIM = 0.0

```

```

CRC1DLY = 0.0
CRC2DLY = 0.0
CRC3DLY = 0.0
CRC4DLY = 0.0
CRC5DLY = 0.0
CRC6DLY = 0.0
CRC7DLY = 0.0
CRC8DLY = 0.0
C1AVGTIM = 0.0
C2AVGTIM = 0.0
C3AVGTIM = 0.0
C4AVGTIM = 0.0
C5AVGTIM = 0.0
C6AVGTIM = 0.0
C7AVGTIM = 0.0
C8AVGTIM = 0.0
C1AVGDLY = 0.0
C2AVGDLY = 0.0
C3AVGDLY = 0.0
C4AVGDLY = 0.0
C5AVGDLY = 0.0
C6AVGDLY = 0.0
C7AVGDLY = 0.0
C8AVGDLY = 0.0

```

- * Subroutine CRCAST generates a ASETS.TMP (temp file) with
- * all the assets for the current time for n-number of branches.

```
CALL CRCAST(SREQ,CURNTP,ASTCNT,ATRFAC,IFR,ROM,STARTBR,ENDBR)
```

- * Verification of ASETS.TMP file. If there were no assets found
- * and written to the file for that time period, then you go to
- * the next time period.

```
IF (ASTCNT .EQ. 0) GOTO 998
```

- * This System Call statement calls slam for execution of slam
- * CRC processing and produces n-number of crc#.out/rpl#.out
- * files. i.e. crc1.out-crc8.out/rpl1.out-rpl8.out

```

IF (IFR.EQ.'CRC')THEN
  IF (CURNTP.EQ.'01')CALL SYSTEM("rs1amb crc1 crcexe")
  IF (CURNTP.EQ.'02')CALL SYSTEM("rs1amb crc2 crcexe")
  IF (CURNTP.EQ.'03')CALL SYSTEM("rs1amb crc3 crcexe")
  IF (CURNTP.EQ.'04')CALL SYSTEM("rs1amb crc4 crcexe")
  IF (CURNTP.EQ.'05')CALL SYSTEM("rs1amb crc5 crcexe")
  IF (CURNTP.EQ.'06')CALL SYSTEM("rs1amb crc6 crcexe")
  IF (CURNTP.EQ.'07')CALL SYSTEM("rs1amb crc7 crcexe")
  IF (CURNTP.EQ.'08')CALL SYSTEM("rs1amb crc8 crcexe")
  IF (CURNTP.EQ.'09')CALL SYSTEM("rs1amb crc9 crcexe")
  IF (CURNTP.EQ.'10')CALL SYSTEM("rs1amb crc10 crcexe")
  IF (CURNTP.EQ.'11')CALL SYSTEM("rs1amb crc11 crcexe")

```

```

IF (CURNTP.EQ.'12')CALL SYSTEM("rslamb crc12 crcexe")
IF (CURNTP.EQ.'13')CALL SYSTEM("rslamb crc13 crcexe")
IF (CURNTP.EQ.'14')CALL SYSTEM("rslamb crc14 crcexe")
IF (CURNTP.EQ.'15')CALL SYSTEM("rslamb crc15 crcexe")
IF (CURNTP.EQ.'16')CALL SYSTEM("rslamb crc16 crcexe")
IF (CURNTP.EQ.'17')CALL SYSTEM("rslamb crc17 crcexe")
IF (CURNTP.EQ.'18')CALL SYSTEM("rslamb crc18 crcexe")
GOTO 997
ELSE
IF (CURNTP.EQ.'01')CALL SYSTEM("rslamb rpl1 rplexe")
IF (CURNTP.EQ.'02')CALL SYSTEM("rslamb rpl2 rplexe")
IF (CURNTP.EQ.'03')CALL SYSTEM("rslamb rpl3 rplexe")
IF (CURNTP.EQ.'04')CALL SYSTEM("rslamb rpl4 rplexe")
IF (CURNTP.EQ.'05')CALL SYSTEM("rslamb rpl5 rplexe")
IF (CURNTP.EQ.'06')CALL SYSTEM("rslamb rpl6 rplexe")
IF (CURNTP.EQ.'07')CALL SYSTEM("rslamb rpl7 rplexe")
IF (CURNTP.EQ.'08')CALL SYSTEM("rslamb rpl8 rplexe")
IF (CURNTP.EQ.'09')CALL SYSTEM("rslamb rpl9 rplexe")
IF (CURNTP.EQ.'10')CALL SYSTEM("rslamb rpl10 rplexe")
IF (CURNTP.EQ.'11')CALL SYSTEM("rslamb rpl11 rplexe")
IF (CURNTP.EQ.'12')CALL SYSTEM("rslamb rpl12 rplexe")
IF (CURNTP.EQ.'13')CALL SYSTEM("rslamb rpl13 rplexe")
IF (CURNTP.EQ.'14')CALL SYSTEM("rslamb rpl14 rplexe")
IF (CURNTP.EQ.'15')CALL SYSTEM("rslamb rpl15 rplexe")
IF (CURNTP.EQ.'16')CALL SYSTEM("rslamb rpl16 rplexe")
IF (CURNTP.EQ.'17')CALL SYSTEM("rslamb rpl17 rplexe")
IF (CURNTP.EQ.'18')CALL SYSTEM("rslamb rpl18 rplexe")
GOTO 997
ENDIF
*
* The IJ loop is a program execution time delay operator
* which allows each crc#.out/rpl#.out to be produced before
* a new SLAM call.

997 DO 600 IJ = 1,900000000
600 CONTINUE

*
* Call subroutine UPDATEFILE which updates the ASETS.TMP file
* by creating a temporary file to show which assets were used.

CALL UPDATEFILE (SREQ, MAXTP)

* Increment time period and then verify if current time is
* greater than or less than ENDTP [end time period]. If
* end time period is reached, the final output file is created
* [CRC/RPL---.OUT]. The IJ loop is a program execution time
* delay operator which allows each crc#.out/rpl#.out to be
* produced before a new SLAM call.

998 CALL MODIFYTP(CURNTP)
IF (CURNTP .LE. ENDTP) THEN

```

```

        GOTO 44
    ELSE
        CALL CROUT (FLNAM,SREQ)
    ENDIF

*   Copy the UNIX output file into a DOS format file (.DOS; LINE1) and
*   remove all protections from that file (LINE2).
*       CALL SYSTEM(LINE1)
*       CALL SYSTEM(LINE2)

*   CALL CRCSTATS (1FR,CURNTP,XTEND,XTND)
999  WRITE(6,45)
    45  FORMAT(////5X,,**** CRC MODEL COMPLETED PROCESSING ****') STOP
    END

```

SUBROUTINES - in alphabetical order

*

* Subroutine: CRCAST

*

* Description: CRC model dumps assets to a file based on the user
* input requirement/type, input replacement facility,
* and the variable to satisfy the requirement or max.
* time and transportation.

* Input: MODRQAST.TMP

*

* Output: ASETS.TMP

*

* Modifications: (STATUS: P - PROPOSED; R - REQUIRED; C - COMPLETED)

* Number Status Date: Description: Initials

*

*

*

SUBROUTINE CRCAST (RQ,CURTP,TOTAL,ATTRIT,REP,RM,STARTB,ENDB)

* Global Variables

CHARACTER RQ*3,CURTP*2,REP*3,RM*3,ARR(4000)*18,STARTB*2,ENDB*2 CHARACTER
CATBRG*5,SX*1,RTYPE*3,TPPRI*9,OLDBR*3

REAL ATTRIT

INTEGER TOTAL,DEGTOT,ASTTOT,NEWTOT,STR,I1,I2,OLDSTR,INDEX
INTEGER OVER, OVRNUM(4000)

LOGICAL THERE

TOTLSTR = 0
TOTAL = 0
INDEX = 0
OLDBR = ,XXX'

OPEN(80,FILE=,/home/warpam/iofiles/MODRQAST.TMP',STATUS='OLD,)
OPEN(83,FILE=,/home/warpam/iofiles/ASETS.TMP',STATUS='NEW')

* Read MODRQAST.TMP file

400 READ(80,410,END=500)CATBRG,SX,RTYPE,TPPRI,STR,I1,I2
410 FORMAT (2X,A5,4X,A1,4X,A3,3X,A9,3X,I6,3X,I6,3X,I6)

* Go to current TP

IF (TPPRI(1:2) .EQ. CURTP .AND.
\$ TPPRI(4:5) .GE. STARTB .AND.

```

$   TPPRI(4:5) .LE. ENDB) THEN
*   Input is a requirement line
    IF (RTYPE .EQ. RQ) THEN
        OLDSTR = STR
        DEGTOT = NINT (REAL(STR)/REAL(1.0 - ATTRIT))
        OLDBR(1:3) = TPPRI(3:5)
        ASTTOT = 0
    ELSE IF (TPPRI(3:5) .EQ. OLDBR .AND.
$       ASTTOT .LT. DEGTOT .AND.
$       RTYPE .EQ. 'TRD' .AND.
$       REP .EQ. ,CRC') THEN
        TOTAL = TOTAL + 1
        DEGTOT = NINT (REAL(OLDSTR - STR)/REAL(1.0 - ATTRIT))
        IF (DEGTOT .LT. 0) THEN
            DEGTOT = 0
        ENDIF
    ELSE IF (TPPRI(3:5) .EQ. OLDBR .AND.
$       ASTTOT .LT. DEGTOT .AND.
$       ((REP .EQ. ,CRC, .AND.
$       RTYPE .NE. ,TRD') .OR.
$       REP .EQ. ,RPL')) THEN
        TOTAL = TOTAL + 1
        ASTTOT = ASTTOT + STR
        IF (ASTTOT .LE. DEGTOT) THEN
*           WRITE (6,430)CATBRG,SX,RTYPE,TPPRI(1:2),TPPRI(3:5),
*           $           TPPRI(6:9),STR
$           WRITE (83,430)CATBRG,SX,RTYPE,TPPRI(1:2),TPPRI(3:5),
$           $           TPPRI(6:9),STR
        ELSE
*           NEWTOT = DEGTOT - (ASTTOT - STR)
*           WRITE (6,430)CATBRG,SX,RTYPE,TPPRI(1:2),TPPRI(3:5),
$           $           CPPRI(6:9),NEWTOT
$           WRITE (83,430)CATBRG,SX,RTYPE,TPPRI(1:2),TPPRI(3:5),
$           $           TPPRI(6:9),NEWTOT
            IF (RM .EQ. ,MAX,) THEN
                OVER = STR - NEWTOT
                INDEX = INDEX + 1
                ARR(INDEX)(1:5) = CATBRG
                ARR(INDEX)(6:6) = SX
                ARR(INDEX)(7:9) = RTYPE
                ARR(INDEX)(10:11) = TPPRI
                ARR(INDEX)(12:14) = TPPRI
                ARR(INDEX)(15:18) = TPPRI
                OVRNUM(INDEX) = OVER
            ENDIF
        ENDIF
    ELSE IF (((REP .EQ. ,CRC, .AND.
$       RTYPE .NE. ,TRD') .OR.
$       REP .EQ. ,RPL,) .AND.
$       RM .EQ. ,MAX') THEN
        INDEX = INDEX + 1
        ARR(INDEX)(1:5) = CATBRG

```



```

        ARR(INDEX)(6:6) = SX
        ARR(INDEX)(7:9) = RTYPE
        ARR(INDEX)(10:11) = TPPRI
        ARR(INDEX)(12:14) = TPPRI
        ARR(INDEX)(15:18) = TPPRI
        OVRNUM(INDEX) = STR
    ENDIF
    GOTO 400

```

```

* Input lines TP is less than the current TP, so read in another line
  ELSE IF (TPPRI(1:2) .LE. CURTP) THEN
    GOTO 400

```

```

* Input lines TP is greater than the current TP, so stop and output

```

```

* ARR contents to file ASETS.TMP

```

```

  ELSE
    IF (RM .EQ. 'MAX') THEN
      DO 420 OVER = 1, INDEX
        WRITE (83,430)ARR(OVER)(1:5),
          $      ARR(OVER)(6:6),ARR(OVER)(7:9),
          $      ARR(OVER)(10:11),ARR(OVER)(12:14),
          $      ARR(OVER)(15:18),OVRNUM(OVER)
420    CONTINUE
      ENDIF
      GOTO 500
    ENDIF

```

```

430  FORMAT (2X,A5,4X,A1,4X,A3,3X,A2,3X,A3,3X,A4,3X,I6)

```

```

500  CLOSE(80,STATUS='KEEP,')
     CLOSE(83,STATUS='KEEP')

```

```

* Maximizes the number of assets allowed to 75,000.
* a temporary file, BTMP.TMP is created to store the
* maximum number fo assets (75,000). Then the BTMP.TMP
* file overwrites the ASETS.TMP.

```

```

    INQUIRE(FILE='/home/warpam/iofiles/BTMP.TMP',STATUS='OLD')
    IF (THERE)THEN
      OPEN(183,FILE='/home/warpam/iofiles/BTMP.TMP ,STATUS='OLD')
      CLOSE(183,STATUS='DELETE')
    ENDIF

```

```

    OPEN(83,FILE='home/warpam/iofiles/ASETS.TMP,STATUS='OLD')
    OPEN(183,FILE='home/warpam/iofiles/BTMP.TMP,STATUS='NEW')

```

```

180  READ (83,184,END=186)CATBRG,SX,RTYPE,TPPRI(1:2),TPPRI(3:5)
     &TPPRI(6:9), STR

```

```

184  FORMAT(2X,A5,4X,A1,4X,A3,3X,A2,3X,A3,3X,A4,3X,I6)

```

```

    TOTLSTR = TOTLSTR + STR
    IF (TOTLSTR.GT.75000)GOTO 186
    IF (TOTLSTR.LE.75000)THEN

```

```

        WRITE (183,184)CATBRG,SX,RTYPE,TPPRI(1:2),TPPRI(3:5),
$TPPRI(6:9),STR
        GOTO 180
    ENDIF

186  CLOSE(83,STATUS='DELETE')
    CLOSE(183,STATUS='KEEP')

    CALL SYSTEM("cp /home/warpam/iofiles/BTMP.TMP /home/warpam/iofiles
&/ASETS2.TMP")

    OPEN(31,FILE='/home/warpam/iofiles/ASETS2.TMP',STATUS='NEW')
    CLOSE(31,STATUS='KEEP')

    RETURN
    END

```

*

* Subroutine: CROUT

*

* Description: Writes the final results to the CRC/RPL output.

*

* Input: MODRQAST.TMP

*

* Output: CRC/RPL____.OUT

*

* Modifications: (STATUS: P - PROPOSED; R - REQUIRED; C - COMPLETED)

*

* Number Status Date: Description: Initials

*

*

*

SUBROUTINE CROUT (FILENM,REQ)

* Global Variables

CHARACTER FILENM*31,REQ*3,CAT*5,SEX*1,TYPE*3,TYPPRI*9

INTEGER INT1,INT2,INT3

* Writes final results to file: CRC/RPL---.OUT.

OPEN(78,FILE=FILENM,ACCESS='APPEND',STATUS='OLD')

OPEN(85,FILE=,/home/warpam/iofiles/MODRQAST.TMP',STATUS='OLD')

600 READ(85,610,ERR=699,END=700)CAT,SEX,TYPE,TYPPRI,INT1,INT2,INT3

610 FORMAT (2X,A5,4X,A1,4X,A3,3X,A9,3X,I6,3X,I6,3X,I6)

IF (TYPE .EQ. REQ) THEN

WRITE (78,620)CAT,SEX,TYPE,TYPPRI,INT1,INT2,'

ELSE

WRITE (78,630)CAT,SEX,TYPE,TYPPRI,INT1,' ',INT3

ENDIF

GOTO 600

620 FORMAT (2X,A5,4X,A1,4X,A3,3X,A9,3X,I6,3X,I6,3X,A6)

630 FORMAT (2X,A5,4X,A1,4X,A3,3X,A9,3X,I6,3X,A6,3X,I6)

699 WRITE(6,*)'ERROR READING FILE: MODRQAST.TMP'

700 CLOSE(85,STATUS='KEEP,')

CLOSE(78, STATUS = ,KEEP')

RETURN

END

```

*****
*
* Subroutine:    DROPLINE
*
* Description:   This subroutine looks for a certain TP/BR/PRIORITY
*                string within th TP + 1 data of the MODRQAST.TMP
*                file. If found, it updates the assets/req. number;
*                otherwise it inserts it accordingly into the file.
*
* Input:        MODRQAST.TMP file array and STR arrays
*
* Output:       updated MODRQAST.TMP file array and STR arrays
*****
* Modifications: (STATUS: P - PROPOSED; R - REQUIRED; C - COMPLETED)
*
* Number Status Date:           Description:           Initials
*
*    01    C           12-18-90    Modified subroutine DROPLINE    BAW
*****

      SUBROUTINE DROPLINE(STRNG,OUT2,STR2,STR3,STR4,LOOP,NEWTP,DELTA)

      CHARACTER OUT2(150000)*18,STRNG*18,NEWTP*2
      INTEGER STR2(150000),STR3(150000),STR4(150000)
      INTEGER SETT,LOOP,DELTA

      SETT = 0
100  SETT = SETT + 1

      IF (SETT.GT.LOOP)GOTO 400

      IF (OUT2 (SETT) (10 :11).NE.NEWTP)GOTO 100

      IF (OUT2(SETT)(10:11).EQ.NEWTP)THEN

* Modify quantity for NEWTP.

      IF (STRNG(12:18).GT.OUT2(SETT)(12:18))GOTO 100

      IF (STRNG(12:18).EQ.OUT2(SETT)(12:18))THEN
        STR2(SETT) = STR2(SETT) + DELTA
        GOTO 400
      ENDIF

* Inserting element for NEWTP.

      IF (STRNG(12:18).LT.OUT2(SETT)(12:18))THEN
        DO 150 I = LOOP,SETT,-1
          OUT2 (I+1) (1: 18) = OUT2 (I) (1 : 18)
          STR2(I+1) = STR2 (I)

```

```

11 READ(31,12,ERR=17,END=18)CBG,SEX,ASETTYPE,TP,BRC,TYPE,STR
12 FORMAT(2X,A5,4X,A1,4X,A3,3X,A2,A3,A4,3X,16)

NTP=0
NBRC=0
NTYPE=0

DO 13 JJ=1,2
  II=ICHAR(TP(JJ:JJ))
  NUM=(79-(127-II))
  IF(JJ.EQ.1)NUM=NUM*10
  IF(JJ.EQ.2)NUM=NUM*1
  NTP=NTP+NUM
13 CONTINUE

DO 14 KK=1,3
  IF(KK.EQ.1)GOTO 14
  11=ICHAR(BRC(KK:KK))
  NUM=(79-(127-11))
  IF(KK.EQ.2)NUM=NUM*10
  IF(KK.EQ.3)NUM=NUM*1
  BRC=NBRC+NUM
14 CONTINUE

DO 15 LL=1,2
  IF(LL.EQ.1)GOTO 15
  11=ICHAR(TYPE(LL:LL))
  NUM=(79-(127-11))
  IF(LL.EQ.2)NTYPE=NUM
15 CONTINUE

* WRITE(6,16)CBG,SEX,ASETTYPE,TP,BRC,TYPE,STR,
* &CBGTRANS(NTP,NBRC,NTYPE)
  WRITE(32,16)CBG,SEX,ASETTYPE,TP,BRC,TYPE,STR,
  &CBGTRANS(NTP,NBRC,NTYPE)
16 FORMAT(2X,A5,4X,A1,4X,A3,3X,A2,A3,A4,3X,16,3X,16)

GOTO 11

17 WRITE(6,*)'ERROR READING FILE: ASETS2.TMP'
18 CLOSE(31,STATUS='KEEP')
  CLOSE(32,STATUS='KEEP')

* Calculate average processing time and average delay
* for each CRC (CRC1 thru CRC8).
  C1AVGTIM = CRC1TIM/CRC1CNT
  C1AVGDLY = CRC1DLY/CRC1CNT

  C2AVGTIM = CRC2TIM/CRC2CNT
  C2AVGDLY = CRC2DLY/CRC2CNT

  C3AVGTIM = CRC3TIM/CRC3CNT

```

```

C*****
C
C      Subroutine:  OPUT
C
C      Description: Generates an output file called TRSLT.TMP.
C
C*****

```

SUBROUTINE OPUT

```

      DIMENSION      CRCTRANS(18,67,7,8),CBGTRANS(18,67,7)      1INCLUDE
      'PARAM.1NC'

```

```

      INTEGER NDEX,STR,NTP,NBRC,NTYPE,CRCTRANS,CBGTRANS,
&CRC1CNT,CRC2CNT,CRC3CNT,CRC4CNT,CRC5CNT,CRC6CNT,CRC7CNT,
&CRC8CNT

```

```

      REAL CRC1TIM,CRC2TIM,CRC3TIM,CRC4TIM,CRC5TIM,CRC6TIM,
&CRC7TIM,CRC8TIM,CRC1DLY,CRC2DLY,CRC3DLY,CRC4DLY,CRC5DLY,
&CRC6DLY,CRC7DLY,CRC8DLY,C1AVGTIM,C2AVGTIM,C3AVGTIM,C4AVGTIM,
&C5AVGTIM,C6AVGTIM,C7AVGTIM,C8AVGTIM,C1AVGDLY,C2AVGDLY,
&C3AVGDLY,C4AVGDLY,C5AVGDLY,C6AVGDLY,C7AVGDLY,C8AVGDLY

```

```

      CHARACTER CBG*5,SEX*1,ASETTYPE*3,TP*2,BRC*3,TYPE*4

```

```

      COMMON/SCOM1/ATR1B(MATRB),DD(MEQT),DDL(MEQT),DTNOW,1I,
1MFA,MSTOP,NCLNR, NCRDR, NPRNT, NNRUN, NNSET, SS(MEQT),
2SSL(MEQT),TNEXT,TNOW, XX(100)

```

```

      COMMON/UCOM1/CBG,SEX,ASETTYPE,TP,BRC,TYPE,STR,NDEX
COMMON/CRARY/CRCTRANS,CBGTRANS,CRC1CNT,CRC2CNT,CRC3CNT,
&CRC4CNT,CRC5CNT,CRC6CNT,CRC7CNT,CRC8CNT,CRC1TIM,CRC2TIM,
&CRC3TIM,CRC4TIM,CRC5TIM,CRC6TIM,CRC7TIM,CRC8TIM,CRC1DLY,
&CRC2DLY,CRC3DLY,CRC4DLY,CRC5DLY,CRC6DLY,CRC7DLY,CRC8DLY,
&C1AVGTIM,C2AVGTIM,C3AVGTIM,C4AVGTIM,C5AVGTIM,C6AVGTIM,
&C7AVGTIM,C8AVGTIM,C1AVGDLY,C2AVGDLY,C3AVGDLY,C4AVGDLY,
&C5AVGDLY,C6AVGDLY,C7AVGDLY,C8AVGDLY

```

```

      LOGICAL THERE

```

- * Checks to see if output file: TRSLT.TMP exists. If the
- * file exist, then the old file is deleted and a new file
- * is then opened.

```

      INQUIRE(FILE=,/home/warpam/iofiles/TRSLT.TMP,,EXIST=THERE)
      IF (THERE)THEN
        OPEN(32,FILE='/home/warpam/iofiles/TRSLT.TMP,,STATUS='OLD')
        CLOSE(32,STATUS='DELETE')
      ENDIF

```

```

      OPEN(32,FILE='/home/warpam/iofiles/TRSLT.TMP',STATUS='NEW')
      OPEN(31,FILE='/home/warpam/iofiles/ASETS2.TMP',STATUS='OLD')

```

```
CRC7CNT = CRC7CNT + 1
CRC7TIM = CRC7TIM + XTIM
CRC7DLY = CRC7DLY + XDLY
ENDIF
IF (NDEX.EQ.8)THEN
  CRC8CNT = CRC8CNT + 1
  CRC8TIM = CRC8TIM + XTIM
  CRC8DLY = CRC8DLY + XDLY
ENDIF
RETURN
END
```

9999 RETURN

* Begins processing EVENT(I) a second time and stores shipped
* persons into an array.

```
2  XTP = ATRIB(4)
   XBRC= ATRIB(9)
   XTYPE=ATRIB(10)

   ITP = (NINT(XTP))
   IBRC =(NINT(XBRC))
   ITYPE=(NINT(XTYPE))

   NDEX=ATRIB(11)

   CRCTRANS(ITP,IBRC,ITYPE,NDEX)=CRCTRANS(ITP,IBRC,ITYPE,NDEX)+1
   CBGTRANS(ITP,IBRC,ITYPE)=CBGTRANS(ITP,IBRC,ITYPE)+1

   XTIM = ATRIB(7)
   XDLY = ATRIB(8)

   IF (NDEX.EQ.1)THEN
     CRC1CNT = CRC1CNT + 1
     CRC1TIM = CRC1TIM + XTIM
     CRC1DLY = CRC1DLY + XDLY
   ENDIF
   IF (NDEX.EQ.2)THEN
     CRC2CNT = CRC2CNT + 1
     CRC2TIM = CRC2TIM + XTIM
     CRC2DLY = CRC2DLY + XDLY
   ENDIF
   IF (NDEX.EQ.3)THEN
     CRC3CNT = CRC3CNT + 1
     CRC3TIM = CRC3TIM + XTIM
     CRC3DLY = CRC3DLY + XDLY
   ENDIF
   IF (NDEX.EQ.4)THEN
     CRC4CNT = CRC4CNT + 1
     CRC4TIM = CRC4TIM + XTIM
     CRC4DLY = CRC4DLY + XDLY
   ENDIF
   IF (NDEX.EQ.5)THEN
     CRC5CNT = CRC5CNT + 1
     CRC5TIM = CRC5TIM + XTIM
     CRC5DLY = CRC5DLY + XDLY
   ENDIF
   IF (NDEX.EQ.6)THEN
     CRC6CNT = CRC6CNT + 1
     CRC6TIM = CRC6TIM + XTIM
     CRC6DLY = CRC6DLY + XDLY
   ENDIF
   IF (NDEX.EQ.7)THEN
```



```
15    WRITE(31,15)CBG,SEX,ASETYP,TP,BRC,TYPE,STR  
      FORMAT(2X,A5,4X,A1,4X,A3,3X,A2,A3,A4,3X,16)
```

* Assign entity attribute Values.

```
      ATRIB(1)=CBG  
      ATRIB(2)=SEX  
      ATRIB(3)=ASETYP
```

* Converts character string into an integer Value and
* assigns it to the appropriate attribute ATRIB().

```
      NTP=0  
      NBRC=0  
      NTYPE=0
```

```
      DO 16 IJ=1,2  
        I1=ICHAR(TP(IJ:IJ))  
        NUM=(79-(127-I1))  
        IF(IJ.EQ.1)NUM=NUM*10  
        IF(IJ.EQ.2)NUM=NUM*1  
        NTP=NTP+NUM  
16    CONTINUE  
      ATRIB(4)=NTP
```

```
      ATRIB(5)=STR
```

```
      DO 17 J=1,3  
        IF(J.EQ.1)GOTO 17  
        I1=ICHAR(BRC(J:J))  
        NUM=(79-(127-I1))  
        IF(J.EQ.2)NUM=NUM*10  
        IF(J.EQ.3)NUM=NUM*1  
        NBRC=NBRC+NUM  
17    CONTINUE  
      ATRIB(9)=NBRC
```

```
      DO 18 K=1,2  
        IF(K.EQ.1)GOTO 18  
        I1=ICHAR(TYPE(K:K))  
        NUM=(79-(127-I1))  
        IF(K.EQ.2)NTYPE=NUM  
18    CONTINUE  
      ATRIB(10)=NTYPE
```

```
      GOTO 9999
```

```
20    WRITE(6,*),ERROR READING FILE: ASETS.TMP'  
25    CLOSE(30,STATUS='KEEP')  
      CLOSE(31,STATUS='KEEP,')  
      XX(1)= 1.0
```

```

C*****
C
C Subroutine: EVENT
C
C Description: Reads in all data from ASETS.TMP file.
C
C*****

```

```

SUBROUTINE EVENT(I)

```

```

DIMENSION CRCTRANS(18,67,7,8),CBGTRANS(18,67,7)
INCLUDE 'PARAM.INC'

```

```

INTEGER NDEX,STR,CRCTRANS,CBGTRANS,CRC1CNT,CRC2CNT,CRC3CNT,
&CRC4CNT,CRC5CNT,CRC6CNT,CRC7CNT,CRC8CNT

```

```

REAL CRC1TIM,CRC2TIM,CRC3TIM,CRC4TIM,CRC5TIM,CRC6TIM,
&CRC7TIM,CRC8TIM,CRC1DLY,CRC2DLY,CRC3DLY,CRC4DLY,CRC5DLY,
&CRC6DLY,CRC7DLY,CRC8DLY,C1AVGTIM,C2AVGTIM,C3AVGTIM,
&C4AVGTIM,C5AVGTIM,C6AVGTIM,C7AVGTIM,C8AVGTIM,C1AVGDLY,
&C2AVGDLY,C3AVGDLY,C4AVGDLY,C5AVGDLY,C6AVGDLY,C7AVGDLY,
&C8AVGDLY

```

```

CHARACTER CBG*5,SEX*1,ASETTYPE*3,TP*2,BRC*3,TYPE*4

```

```

COMMON/SCOM1/ATTRIB(MATRB), DD(MEQT), DDL(MEQT), DTNOW, II,
IMFA,MSTOP,NCLNR, NCRDR, NPRNT, NNRUN, NNSET, NTAPE, SS(MEQT),
2SSL(MEQT),TNEXT,TNOW, XX(100)

```

```

COMMON/UCOM1/CBG,SEX,ASETTYPE,TP,BRC,TYPE,STR,NDEX
COMMON/CRARY/CRCTRANS,CBGTRANS,CRC1CNT,CRC2CNT,CRC3CNT,
&CRC4CNT,CRC5CNT,CRC6CNT,CRC7CNT,CRC8CNT,CRC1TIM,CRC2TIM,
&CRC3TIM,CRC4TIM,CRC5TIM,CRC6TIM,CRC7TIM,CRC8TIM,CRC1DLY,
&CRC2DLY,CRC3DLY,CRC4DLY,CRC5DLY,CRC6DLY,CRC7DLY,CRC8DLY,
&C1AVGTIM,C2AVGTIM,C3AVGTIM,C4AVGTIM,C5AVGTIM,C6AVGTIM,
&C7AVGTIM,C8AVGTIM,C1AVGDLY,C2AVGDLY,C3AVGDLY,C4AVGDLY,
&C5AVGDLY,C6AVGDLY,C7AVGDLY,C8AVGDLY

```

```

GOTO (1,2),I

```

```

* Open input file: ASETS.TMP, read entire file.
* Open output file: ASETS2.TMP and stores the ASETS.TMP such
* that the time period, branhh and type are concatenated to
* a nine character code.

```

```

1 OPEN(30,FILE='/home/warpam/iofiles/ASETS.TMP',STATUS='OLD')
OPEN(31,FILE='/home/warpam/iofiles/ASETS2.TMP',STATUS='OLD')

```

```

14 READ(30,14,ERR=20,END=25)CBG,SEX,ASETTYPE,TP,BRC,TYPE,STR
FORMAT(2X,A5,4X,A1,4X,A3,3X,A2,3X,A3,3X,A4,3X,I6)

```

```

* WRITE(6,14)CBG,SEX,ASETTYPE,TP,BRC,TYPE,STR

```

```

C*****
C
C Subroutine: INTLC
C
C Description: Initializes SLAM II variables.
C
C*****

```

```

SUBROUTINE INTLC

```

```

INCLUDE 'PARAM.INC,

```

```

CHARACTER*2 TP

```

```

INTEGER TTFIN

```

```

REAL ATRIB(5),ATR1B(12)
COMMON/SCOM1/ATRIB(MATRB), DD(MEQT), DDL(MEQT), DTNOW, II,
1MFA,MSTOP,NCLNR, NCRDR, NPRNT, NNRUN, NNSET, NTAPE, SS(MEQT),
2SSL(MEQT),TNEXT, TNOW, XX(100)

```

```

* Initialize SLAM II variables.

```

```

ATR1B(1)=0.
ATR1B(2)=0.
ATR1B(3)=0.
ATR1B(4)=0.
ATR1B(5)=0.
ATR1B(9)=0.
ATR1B(10)=0.
ATR1B(11)=0.
ATR1B(12)=0.

```

```

IF (TP.EQ. 01')THEN

```

```

TTFIN = 7200

```

```

GOTO 999

```

```

ELSE

```

```

TTFIN = 7680

```

```

* MONTR,CLEAR,480;

```

```

ENDIF

```

```

999 CALL ENTER(1,ATRIB)

```

```

RETURN

```

```

END

```

```

$ KVANCO=7,KVASTO=8,KVAMNO=9,KVATLU=10,KVAPFE=11,KVAPLE=12,
$ KVCSPR=1,KVCNWL=2,KVCPFE=3,KVCPLE=4,KVCMXL=5,KVCPPL=6,
$ KVCCRC=7,KVCNTE=8,KVCCST=9,KVCSTB=10,KVCTLU=11,KVCVCO=12,
$ KVCRRC=13,KVCCPI=14,KVSESP=1,KVSLSP=2,KVSACC=3,KVSDEC=4,KVSLEN=5,
$ KVSBUF=6,KVSCKZ=7,KVSIFL=8,KVSJRQ=9,KVSINI=10,KVSREP=11,
$ KVSNTL=12,KVSNTU=13,KVSNUL=14,KVSNUU=16,KVSNU=18,KVSNUF=20,
$ KV8NUC=22,KVSNU=24,KVSTE=26,KVSSTF=27,KVSTLU=28,KVSPFE=29,
$ KVSPL=30,KVSPAL=31,KVSNV32,KVUPV8=1,KVUCSG=2,KVUCCP=3,
$ KVICP=4,KVUDCP=5,KVUPCL=6,KVUVM0=7,KVUCSI=8,KVUSPD=9,KVUCBF=10,
$ KVUCBT=11,KVUTLU=12,KVUSCP=13,KVUSHT=14,KVUNTL=15,KVUSPT=16,
$ KVFVSI=4,KVWIFL=4,KVWVSI=5,KVWVRQ=6,KVWVRE=7,
$ KWCPI=8,KVMCPI=4,KVANWF=13,KVCNWF=15,KVFNWF=5,KVMNWF=4,
$ KVSNUF=33,KVUNWF=15,KVWNWF=8)
  PARAMETER (MXMSG=250)
  PARAMETER (MXPOUT=6)
  INCLUDE 'PARAM.INC'
  COMMON/SCOM1/ATRIB(MATRB), DD(MEQT), DDL(MEQT), DTNOW, II,
1MFA,MSTOP,NCLNR, NCRDR, NPRNT, NNRUN, NNSET, NTAPE, SS(MEQT),
2SSL(MEQT),TNEXT, TNOW, XX(100)
  COMMON QSET(1800000)
  EQUIVALENCE (NSET (1),QSET(1))
  NNSET=1800000
  NCRDR=5
  NPRNT=6
  NTAPE=7
  OPEN(UNIT=NCRDR,FILE='fort.5')
  OPEN(UNIT=NPRNT,FILE='fort.6')
  CALL SLAM
  STOP
  END

```

6.8 CRC MODEL SPECIFIC PROGRAMS

6.8.1 FORTRAN PROGRAMS SUPPORTING SLAM II CRC PROGRAM - PROGRAM MAIN

C*****

C Program Name: CRCEXE Date: 12-03-1990

C

C File Name: CRCNPUT.F

C

C Programmer: Beth A. Wilson, SAIC, (703)749-8771

C

C Description: FORTRAN programs supporting SLAM II CRC program
C Program MAIN which includes CRC Subroutine INTLC,
C Subroutine EVENT, Subroutine OPUT.

C

C Input: ASETS.TMP

C

C Output: TRSLT.TMP

C

C*****

C Modifications: (STATUS: P - PROPOSED; R - REQUIRED;
C - COMPLETED)

C

C Number Status Date: Description: Initials

C

C 01 12-03-90 Modified Subroutine EVENT such BAW
C that all data records in the
C file: ASETS.TMP would be read
C and the accumulated strencth
C would be calculated and passed
C back to the SLAM II program
C QUEUE for unbatching and process-
C ing in the CRC model.

C

C 02 01-30-91 Documentation of arious calcula- BAW
C tions and procedures.

C

C*****

PROGRAM MAIN

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

DIMENSION NSET(1800000)

PARAMETER (KVACP1=1,KVACP2=2,KVALEN=3,KVACAP=4,KVADIR=5,KVANTX=6,

```

        CALL MODIFYTP (NEWTP)
        DELTA = STR2(KK) - STR1(MAX)
        STRNG = OUT2(KK)(1:18)
        CALL DROPLINE(STRNG,OUT2,STR2,STR3,STR4,LOOP,
&                                NEWTP,DELTA)
        MAX = MAX + 1
        GOTO 20
    ENDIF
ENDIF
IF (OUT2(KK)(1:18).NE.OUT1(MAX)(1:18))THEN
    STR3(KK) = 0
    STR4(KK) = 0
    STR3(REQLOC) = STR3(REQLOC) + STR4(KK)
    NEWTP = OLDTP
    CALL MODIFYTP (NEWTP)
    DELTA = STR2(KK) - STR4(KK)
    STRNG = OUT2(KK)(1:18)
    CALL DROPLINE(STRNG,OUT2,STR2,STR3,STR4,LOOP,
&                                NEWTP,DELTA)
    GOTO 20
ENDIF
ENDIF
ENDIF
ENDIF

```

```

50 WRITE(6,*)'WRITING UPDATES TO FILE:  MODRQAST.TMP'
* Write new output to MODRQA8T.TMP.

```

```

OPEN(80,FILE=,/home/warpam/iofiles/MODRQAST.TMP',STATUS='NEW')

```

```

    DO 601 J = 1,LOOP
        IF (OUT2(J)(10:11) .NE. LASTTP) THEN
*           WRITE(6,602)OUT2(J)(1:5), OUT2(J)(6:6), OUT2(J)(7:9),
*           $           OUT2(J)(10:18), STR2(J), STR3(J), STR4(J)
        WRITE(80,602)OUT2(J)(1:5), OUT2(J)(6:6), OUT2(J)(7:9),
        $           OUT2(J)(10:18), STR2(J), STR3 (J), STR4 (J)
602     FORMAT(2X,A5,4X,A1,4X,A3,3X,A9,3X,I6,3X,I6,3X,I6)
        ENDIF
601 CONTINUE
    CLOSE(80,STATUS=,KEEP,)

```

```

* Exit subroutine
  RETURN
  END

```

```

      IF (STR3(REQLOC).LT.STR2(REQLOC))THEN
        NEWTP = OLDTP
        CALL MODIFYTP (NEWTP)
        DELTA = STR2(REQLOC) - STR3(REQLOC)
        STRNG = OUT2(REQLOC)(1:18)
        CALL DROPLINE(STRNG,OUT2,STR2,STR3,STR4,LOOP,
&                      NEWTP,DELTA)
      GOTO 50
    ENDIF
  ENDIF
  IF (OUT2(KK)(10:11).NE.OLDTP)GOTO 20
  IF (OUT2(KK)(10:11).EQ.OLDTP)THEN
    IF (OUT2(KK)(7:9).EQ.RQMT)THEN
      RQCNT = RQCNT + 1
      IF (RQCNT.EQ.1)THEN
        REQLOC = KK
        STR3(REQLOC) = 0
        STR4(REQLOC) = 0
        GOTO 20
      ENDIF
      IF (RQCNT.GT.1)THEN
        IF (STR3(REQLOC).EQ.STR2(REQLOC))THEN
          RELOC = KK
          STR3(REQLOC) = 0
          STR4(REQLOC) = 0
          GOTO 20
        ENDIF
        IF (STR3(REQLOC).LT.STR2(REQLOC))THEN
          NEWTP = OLDTP
          CALL MODIFYTP (NEWTP)
          DELTA = STR2(REQLOC) - STR3(REQLOC)
          STRNG = OUT2(REQLOC)(1: 18)
          CALL DROPLINE(STRNG,OUT2,STR2,STR3,STR4,LOOP,
&                      NEWTP,DELTA)
          REQLOC = KK
          STR3(REQLOC) = 0
          STR4(REQLOC) = 0
          GOTO 20
        ENDIF
      ENDIF
    ENDIF
  ENDIF
  IF (OUT2(KK)(7:9).NE.RQMT)THEN
    IF (OUT2(KK)(1:18).EQ.OUT1(MAX)(1:18))THEN
      STR3(KK) = 0
      STR4(KK) = STR1(MAX)
      STR3(REQLOC) = STR3(REQLOC) + STR1(MAX)
      IF (STR1(MAX).EQ.STR2(KK))THEN
        MAX = MAX + 1
        GOTO 20
      ENDIF
      IF (STR1(MAX).LT.STR2(KK))THEN
        NEWTP = OLDTP

```

```

        OUT1(INDEX)(7:9) = TTYPE
        OUT1(INDEX)(10:11) = TP
        OUT1(INDEX)(12:18) = BRAST
        STR1(INDEX) = SHIPPED
        GOTO 11

13  WRITE(6,*),ERROR READING FILE:  TRSLT.TMP'
14  CLOSE(32,STATUS=,KEEP')

*
* Stores file:  MODRQAST.TMP into arrays OUT2, STR2, STR3, STR4
*
        LOOP = 0

OPEN(80,FILE='/home/warpam/iofiles/MODRQAST.TMP',STATUS='OLD')

16  READ(80,17,ERR=18,END=19)CATBRG,SEXX,TYP,TPRI,NEWSTR,
    $                                ORGSTR,ASETSU
17  FORMAT(2X,A5,4X,A1,4X,A3,3X,A9,3X,I6,3X,I6,3X,I6)

        LOOP = LOOP + 1
        OUT2(LOOP)(1:5) = CATBRG
        OUT2(LOOP)(6:6) = SEXX
        OUT2(LOOP)(7:9) = TYP
        OUT2(LOOP)(10:18) = TPRI
        STR2(LOOP) = NEWSTR
        STR3(LOOP) = ORGSTR
        STR4(LOOP) = ASETSU

        GOTO 16
18  WRITE(6,*),' ERROR READING FILE:    MODRQAST.TMP,
19  CLOSE(80,STATUS='DELETE')

*
* Modify MODRQAST.TMP arrays (OUT2,STR2,STR3,STR4) *

        WRITE(6,*)'BEGIN MODIFYING MODRQAST.TMP arrays'
        WRITE(6,*)'No. OF TRSLT.TMP RECORDS ... INDEX ' INDEX
        WRITE(6,*)'No. OF MODRQAST.TMP RECORDS .. LOOP, LOOP
        WRITE(6,*)'TIME PERIOD ... OLDTP ', OLDTP

        KK = 0
        MAX = 1
        RQCNT = 0
        REQLOC = 0

20  KK = KK + 1

        IF (KK.GT.LOOP)GOTO 50

        IF (MAX.GT.INDEX)THEN
            IF (STR3(REQLOC).EQ.STR2(REQLOC))GOTO 50

```



```

*****
*
* Subroutine:    UPDATEFILE
*
* Description:   Updates the ASETS.TMP by creating a temporary file
*               to show which assets were used.
*
* Input:        TRSLT.TMP
*               ASETS.TMP
*
* Output:       Modified MODRQAST.TMP
*
*****
* Modifications: (STATUS: P - PROPOSED; R - REQUIRED; C - COMPLETED)
*
* Number Status Date:           Description:           Initials
* -----
*      01      C      12-18-90   Modified subroutine   BAW
*                                UPDATEFILE
*
*****

```

SUBROUTINE UPDATEFILE (RQMT, LASTTP)

```

* Global Variables
  CHARACTER RQMT*3, LASTTP*2, NEWTP*2, STRNG*18
  CHARACTER TSEX*1, SEX*1, TP*2, TTYPE*3, TYP*3, OLDTP*2
  CHARACTER CBG*5, CATBRG*5, BRAST*7, TPRI*9
  CHARACTER OUT1(15000)*18, OUT2(150000)*18

  INTEGER STR1(15000), STR2(150000), STR3(150000), STR4(150000)
  INTEGER STR, SHIPPED, NEWSTR, ORGSTR, ASETSU, LOOP, KK, MAX, INDEX
  INTEGER REQLOC, DELTA, RQCNT

* Initialize Variables

  INDEX = 0

* Opens and reads input file: TRSLT.TMP.
* The OUT1 and STR1 arrays will hold assets used (shipped)
* along with their characteristics.

  OPEN(32, FILE='/home/warpam/iofiles/TRSLT.TMP', STATUS='OLD')

11  READ(32, 12, ERR=13, END=14) TCBG, TSEX, TTYPE, TP, BRAST, STR, SHIPPED
12  FORMAT(2X, A5, 4X, A1, 4X, A3, 3X, A2, A7, 3X, 16, 3X, I6)

  OLDTP = TP
  INDEX = INDEX + 1
  OUT1(INDEX)(1:5) = TCBG
  OUT1(INDEX)(6:6) = TSEX

```

```
TIME = '15'  
ELSE IF (TIME .EQ. '15') THEN  
TIME = '16'  
ELSE IF (TIME .EQ. '16') THEN  
TIME = '17'  
ELSE IF (TIME .EQ. '17') THEN  
TIME = '18'  
ELSE IF (TIME .EQ. '18') THEN  
TIME = '19'  
ENDIF
```

```
* Exit subroutine  
RETURN  
END
```

```

*****
*
* Subroutine:    MODIFYTP
*
* Description:   This subroutine increments the TP string by 1
*
* Input:        TP
*
* Output:       TP + 1
*
*****
* Modifications: (STATUS: P - PROPOSED; R - REQUIRED; C - COMPLETED)
*
* Number Status Date:           Description:           Initials
* -----
*
*****

```

SUBROUTINE MODIFYTP (TIME)

CHARACTER TIME*2

```

IF (TIME .EQ. '00') THEN
TIME = '01'
ELSE IF (TIME .EQ. '01') THEN
TIME = '02'
ELSE IF (TIME .EQ. '02') THEN
TIME = '03'
ELSE IF (TIME .EQ. '03') THEN
TIME = '04'
ELSE IF (TIME .EQ. '04') THEN
TIME = '05'
ELSE IF (TIME .EQ. '05') THEN
TIME = '06'
ELSE IF (TIME .EQ. '06') THEN
TIME = '07'
ELSE IF (TIME .EQ. '07') THEN
TIME = '08'
ELSE IF (TIME .EQ. '08') THEN
TIME = '09'
ELSE IF (TIME .EQ. '09') THEN
TIME = '10'
ELSE IF (TIME .EQ. '10') THEN
TIME = '11'
ELSE IF (TIME .EQ. '11') THEN
TIME = '12'
ELSE IF (TIME .EQ. '12') THEN
TIME = '13'
ELSE IF (TIME .EQ. '13') THEN
TIME = '14'
ELSE IF (TIME .EQ. '14') THEN

```

```
STR = '66'  
ELSE IF (NUM .EQ. 67) THEN  
STR = '67'  
ELSE IF (NUM .EQ. 68) THEN  
STR = '68'  
ENDIF
```

```
* Exit subroutine  
RETURN  
END
```

```
STR = '40'  
ELSE IF (NUM .EQ. 41) THEN  
STR = '41'  
ELSE IF (NUM .EQ. 42) THEN  
STR = '42'  
ELSE IF (NUM .EQ. 43) THEN  
STR = '43'  
ELSE IF (NUM .EQ. 44) THEN  
STR = '44'  
ELSE IF (NUM .EQ. 45) THEN  
STR = '45'  
ELSE IF (NUM .EQ. 46) THEN  
STR = '46'  
ELSE IF (NUM .EQ. 47) THEN  
STR = '47'  
ELSE IF (NUM .EQ. 48) THEN  
STR = '48'  
ELSE IF (NUM .EQ. 49) THEN  
STR = '49'  
ELSE IF (NUM .EQ. 50) THEN  
STR = '50'  
ELSE IF (NUM .EQ. 51) THEN  
STR = '51'  
ELSE IF (NUM .EQ. 52) THEN  
STR = '52'  
ELSE IF (NUM .EQ. 53) THEN  
STR = '53'  
ELSE IF (NUM .EQ. 54) THEN  
STR = '54'  
ELSE IF (NUM .EQ. 55) THEN  
STR = '55'  
ELSE IF (NUM .EQ. 56) THEN  
STR = '56'  
ELSE IF (NUM .EQ. 57) THEN  
STR = '57'  
ELSE IF (NUM .EQ. 58) THEN  
STR = '58'  
ELSE IF (NUM .EQ. 59) THEN  
STR = '59'  
ELSE IF (NUM .EQ. 60) THEN  
STR = '60'  
ELSE IF (NUM .EQ. 61) THEN  
STR = '61'  
ELSE IF (NUM .EQ. 62) THEN  
STR = '62'  
ELSE IF (NUM .EQ. 63) THEN  
STR = '63'  
ELSE IF (NUM .EQ. 64) THEN  
STR = '64'  
ELSE IF (NUM .EQ. 65) THEN  
STR = '65'  
ELSE IF (NUM .EQ. 66) THEN
```

```
STR = '14'  
ELSE IF (NUM .EQ. 15) THEN  
STR = '15'  
ELSE IF (NUM .EQ. 16) THEN  
STR = '16'  
ELSE IF (NUM .EQ. 17) THEN  
STR = '17'  
ELSE IF (NUM .EQ. 18) THEN  
STR = '18'  
ELSE IF (NUM .EQ. 19) THEN  
STR = '19'  
ELSE IF (NUM .EQ. 20) THEN  
STR = '20'  
ELSE IF (NUM .EQ. 21) THEN  
STR = '21'  
ELSE IF (NUM .EQ. 22) THEN  
STR = '22'  
ELSE IF (NUM .EQ. 23) THEN  
STR = '23'  
ELSE IF (NUM .EQ. 24) THEN  
STR = '24'  
ELSE IF (NUM .EQ. 25) THEN  
STR = '25'  
ELSE IF (NUM .EQ. 26) THEN  
STR = '26'  
ELSE IF (NUM .EQ. 27) THEN  
STR = '27'  
ELSE IF (NUM .EQ. 28) THEN  
STR = '28'  
ELSE IF (NUM .EQ. 29) THEN  
STR = '29'  
ELSE IF (NUM .EQ. 30) THEN  
STR = '30'  
ELSE IF (NUM .EQ. 31) THEN  
STR = '31'  
ELSE IF (NUM .EQ. 32) THEN  
STR = '32'  
ELSE IF (NUM .EQ. 33) THEN  
STR = '33'  
ELSE IF (NUM .EQ. 34) THEN  
STR = '34'  
ELSE IF (NUM .EQ. 35) THEN  
STR = '35'  
ELSE IF (NUM .EQ. 36) THEN  
STR = '36'  
ELSE IF (NUM .EQ. 37) THEN  
STR = '37'  
ELSE IF (NUM .EQ. 38) THEN  
STR = '38'  
ELSE IF (NUM .EQ. 39) THEN  
STR = '39'  
ELSE IF (NUM .EQ. 40) THEN
```

```

*****
*
* Subroutine:      INT2STR
*
* Description:     Converts an integer to a character string
*
* Input:           INTEGER VARIABLE
*
* Output:          CHARACTER STRING
*
*****
* Modifications:  (STATUS: P - PROPOSED; R - REQUIRED; C - COMPLETED)
*
* Number Status  Date:           Description:           Initials
* -----
*
*****

```

SUBROUTINE INT2STR (NUM, STR)

INTEGER NUM

CHARACTER STR*2

```

IF (NUM .EQ. 1) THEN
  STR = '01'
ELSE IF (NUM .EQ. 2) THEN
  STR = '02'
ELSE IF (NUM .EQ. 3) THEN
  STR = '03'
ELSE IF (NUM .EQ. 4) THEN
  STR = '04'
ELSE IF (NUM .EQ. 5) THEN
  STR = '05'
ELSE IF (NUM .EQ. 6) THEN
  STR = '06'
ELSE IF (NUM .EQ. 7) THEN
  STR = '07'
ELSE IF (NUM .EQ. 8) THEN
  STR = '08'
ELSE IF (NUM .EQ. 9) THEN
  STR = '09'
ELSE IF (NUM .EQ. 10) THEN
  STR = '10'
ELSE IF (NUM .EQ. 11) THEN
  STR = '11'
ELSE IF (NUM .EQ. 12) THEN
  STR = '12'
ELSE IF (NUM .EQ. 13) THEN
  STR = '13'
ELSE IF (NUM .EQ. 14) THEN

```

```

        STR3(I+1) = STR3(I)
        STR4(I+1) = STR4(I)
150  CONTINUE
        OUT2(SETT)(1:9) = STRNG(1:9)
        OUT2(SETT)(10:11) = NEWTP
        OUT2(SETT)(12:18) = STRNG(12:18)
        STR2(SETT) = DELTA
        STR3(SETT) = 0
        STR4(SETT) = 0

        LOOP = LOOP + 1
        GOTO 400
    ENDIF
ENDIF
400  RETURN
    END

```



```

C3AVGDLY = CRC3DLY/CRC3CNT

C4AVGTIM = CRC4TIM/CRC4CNT
C4AVGDLY = CRC4DLY/CRC4CNT

C5AVGTIM = CRC5TIM/CRC5CNT
C5AVGDLY = CRC5DLY/CRC5CNT

C6AVGTIM = CRC6TIM/CRC6CNT
C6AVGDLY = CRC6DLY/CRC6CNT

C7AVGTIM = CRC7TIM/CRC7CNT
C7AVGDLY = CRC7DLY/CRC7CNT

C8AVGTIM = CRC8TIM/CRC8CNT
C8AVGDLY = CRC8DLY/CRC8CNT

20  WRITE(6,20)
    FORMAT(/15X,'AVERAGE TIME FOR THIS TIME PERIOD',//3X,
    &' CRC',4X,'COUNT',8X,'AVG. PROCESSING TIME',8X,
    &'AVG. DELAY TIME',/)

21  WRITE(6,21)CRC1CNT,C1AVGTIM,C1AVGDLY
    FORMAT(3X,'I ',4X,I5,10X,F10.1,16X,F10.1)

22  WRITE(6,22)CRC2CNT,C2AVGTIM,C2AVGDLY
    FORMAT(3X,'II ',4X,I5,10X,F10.1,16X,F10.1)

23  WRITE(6,23)CRC3CNT,C3AVGTIM,C3AVGDLY
    FORMAT(3X,'III ',4X,I5,10X,F10.1,16X,F10.1)

24  WRITE(6,24)CRC4CNT,C4AVGTIM,C4AVGDLY
    FORMAT(3X,'IV ',4X,I5,10X,F10.1,16X,F10.1)

25  WRITE(6,25)CRC5CNT,C5AVGTIM,C5AVGDLY
    FORMAT(3X,'V ',4X,I5,10X,F10.1,16X,F10.1)

26  WRITE(6,26)CRC6CNT,C6AVGTIM,C6AVGDLY
    FORMAT(3X,'VI ',4X,I5,10X,F10.1,16X,F10.1)

27  WRITE(6,27)CRC7CNT,C7AVGTIM,C7AVGDLY
    FORMAT(3X,'VII ',4X,I5,10X,F10.1,16X,F10.1)

28  WRITE(6,28)CRC8CNT,C8AVGTIM,C8AVGDLY
    FORMAT(3X,'VIII ',4X,I5,10X,F10.1,16X,F10.1)

    RETURN
    END

```

6.8.2 CRC MODEL PROCESSING FLOW

Figure 10 depicts the processing flow within the SLAM II portion of the CRC model.

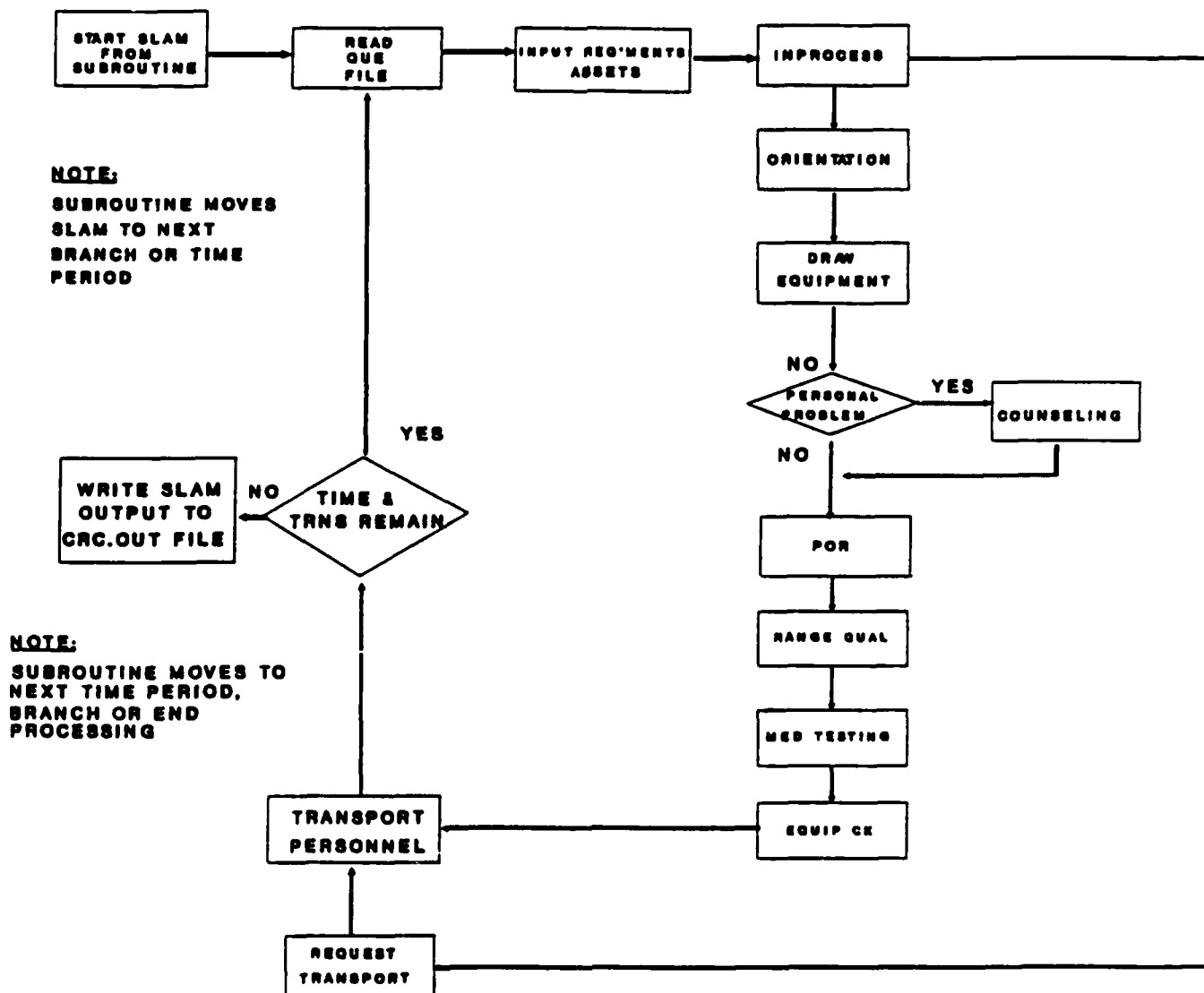


FIGURE 10: CRC SLAM II PROCESSING

6.8.3 CRC MODEL SLAM II PROGRAMS

```

;*****
; SLAM II VARIABLES
; ATRIB(1) = CAT/BRANCH AND GRADE ACROYMN
; ATRIB(2) = MALE OR FEMALE
; ATRIB(3) = ASSET TYPE
; ATRIB(4) = TIME PERIOD
; ATRIB(5) = REQUIREMENT ASSET QUANTITY
; ATRIB(6) = ARRIVAL TIME TO CRC
; ATRIB(7) = TOTAL TIME IN THE CRC
; ATRIB(8) = EXCESS TRANSPORTATION TIME OF (72 hours +- 40 hours)
; ATRIB(9) = BRANCH INDEX
; ATRIB(10)= TYPE ASSET INDEX
; ATRIB(11)= CRC INDEX
; ATRIB(12)= COUNTER TO COLLECT STATISTICS
;

```

```

;*****

```

```

GEN,WILSON\WOJCIK\FRAME,WARPAM\VERSION II.1/30/91,,N,N,Y/N,N,Y/F,72;
LIMITS,90,14,80000;
EQUIVALENCE/ATRIB(1),CBG;
EQUIVALENCE/ATRIB(2),SEX;
EQUIVALENCE/ATRIB(3),ASETTYPE;
EQUIVALENCE/ATRIB(4),TP;
EQUIVALENCE/ATRIB(5),STR;
EQUIVALENCE/ATRIB(9),BRC;
EQUIVALENCE/ATRIB(10),TYPE;

```

LIMIT STATEMENT SETS THE
NUMBER OF QUEUES WHICH MAY
BE CREATED: SET AT 90-MAX
IS 100; 14 ATTRIBUTES PER
ENTITY AND 80000 ENTITIES
MAX IN THE SYSTEM

```

; The INIT statement sets the length of the simulation for CRC
; and time period 1. It is set in minutes at 7919 (11 days)
; (actually one minute less) to represent 1 zero day and 10
; processing days.
;
; For CRC and time periods 2-18, it is set at 10059 representing
; the same 11 days plus the time required to fill the system for
; steady state operation which for version 1.0 of WARPAM is 3 days
; (2160 min) which is the time the first entities would be available
; to exit the system.
;
; For RPL and time period 1, it is set at 7200 representing 10 days.
;
; For RPL and time period 2-18, it is set at 9009 representing 13 days
; (10 processing days and 1110 min., the time required to complete
; processing the 6 stations in RPL version 1.0).

```

```

INIT,,7919;
NETWORK;

```

INIT FOR CRC1.DAT (TIME PERIOD 1) IS 7919
INIT FOR CRC2.DAT-CRC18.DAT IS 10059

*** REPLACEMENT OPERATION BEGINS!!!! **

```

ENTER,1;
; EVENT 1 clones the incoming entities. One entity moves through the
; normal system and one is returned to the FORTRAN program where it
; causes the next line to be read and then is destroyed.
;
LOOP EVENT,1;
GOON,2;
ACT,,XX(1).EQ.0.0;
ACT,,XX(1).EQ.0.0,LOOP; READ IN NEXT LINE
ACT,,XX(1).EQ.1.0,T2; TERMINATE DUPLICATE OF LAST ASSET LINE
;
GOON;
ACT/89; NUMBER OF ENTITIES
GOON;
UNBATCH,5;
;
GOON;
; ASSIGN,TRIB(6)=TNOW;
GOON,1;
;
; This section may be activated to route certain branches to specific
; CRC Branch numbers are found in the WARPRI.TBL.
;
; ACT/90,,TRIB(9).EQ.3.OR.TRIB(9).EQ.8,C1; INFANTRY OFFICIERS
; ACT/91,,TRIB(9).EQ.18.OR.TRIB(9).EQ.23,C1; INFANTRY ENLISTED
; ACT/92,,TRIB(9).EQ.2.OR.TRIB(9).EQ.7.OR.TRIB(9).EQ.15,C1; AVN OFF
; ACT/93,,TRIB(9).EQ.17.OR.TRIB(9).EQ.22,C1; AVIATION ENL
; ACT/94,,TRIB(9).EQ.1.OR.TRIB(9).EQ.6,C2; ARMOR OFF
; ACT/95,,TRIB(9).EQ.16.OR.TRIB(9).EQ.21,C2; ARMOR ENL
; ACT/96,,H3; OTHER MOS
;
; This section may be activated to route a segregated branch to a certain
; CRC. WARPAM version 1.0 routes nearly equally to all eight CRC.
;H3 GOON;
; ACT,,.19,C2;
; ACT,,.81,C3;
;
; ACT,,.12,C1;
; ACT,,.13,C2;
; ACT,,.12,C3;
; ACT,,.13,C4;
; ACT,,.12,C5;
; ACT,,.13,C6;
; ACT,,.12,C7;
; ACT,,.13,C8;
;
; ***** FORT BENNING (CRC I) *****
;
; C1 GOON;
;
; Zero day allows number of personnel shown to enter a CRC.
;

```

EVENT 1 READS IN THE ASSETS FILE

ASSET LINES ARE UNBATCHED TO ENTITIES

THIS SECTION ROUTES ENTITIES
TO THE CRC. THE SUM MUST BE
100%. TO A CRC(S) MAKE THE
PERCENTAGE ZERO.


```

;
;   QUEUE(15),,30000,BLOCK;
;   ACT(150)/15,RNORM(135,13.5);          CRC II  WPN ZERO
;
;   QUEUE(16),,30000,BLOCK;
;   ACT(75)/16,RNORM(53,5.3);          CRC II  MASK TEST
;
;   QUEUE(17),,30000,BLOCK;
;   ACT(300)/17,RNORM(240,24);          CRC II  MEDICAL
;
;   GOON,1;
;   QUEUE(18),,30000,BLOCK;
;   ACT(60)/18,RNORM(48,0),0.2;          CRC II  OPTICAL
;   ACT/19,,0.8;
;
;   QUEUE(19),,30000,BLOCK;
;   ACT(3000)/20,RNORM(360,36),,EXIT; CRC II  MISC/MEALS
;
; *****          FORT JACKSON (CRC III)          *****
GOON;
;   ASSIGN,ATRI(11)=3;
;   QUEUE(21),,30000,BLOCK;
;   ACT(400)/21,RNORM(720,0);          CRC III DAY ZERO
;
;   ASSIGN,ATRI(6)=TNOW;
;
;   QUEUE(22),,30000,BLOCK;
;   ACT(32)/22,RNORM(4,0.4);          CRC III INPROCESS
;
;   QUEUE(23),,30000,BLOCK;
;   ACT(32)/23,RNORM(16,1.6);          CRC III  POR
;
;   QUEUE(24),,30000,BLOCK;
;   ACT(50)/24,RNORM(90,9);          CRC III  OCIE
;
;   QUEUE(25),,30000,BLOCK;
;   ACT(200)/25,RNORM(165,16.5);          CRC III WPN ZERO
;
;   QUEUE(26),,30000,BLOCK;
;   ACT(100)/26,RNORM(68,6.8);          CRC III MASK TEST
;
;   QUEUE(27),,30000,BLOCK;
;   ACT(100)/27,RNORM(80,8);          CRC III MEDICAL;
;
;   GOON,1;
;   QUEUE(28),,30000,BLOCK;
;   ACT(80)/28,RNORM(48,4.8),0.2;          CRC III OPTICAL
;   ACT/29,,0.8;
;
;   QUEUE(29),,30000,BLOCK;
;   ACT(3000)/30,RNORM(360,36),,EXIT;CRC III MISC/MEALS

```

```

;
;
;***** FORT SILL (CRC IV)*****
;
C4      GOON;
;
;      Zero day allows number of personnel shown to enter a CRC.
;
;      ASSIGN, ATRIB(11)=4;
;      QUEUE(31),,30000,BLOCK;
;      ACT(300)/31,RNORM(720,0);
;                                     CRC IV DAY ZERO
;
;      ASSIGN, ATRIB(6)=TNOW;
;
;      QUEUE(32),,30000,BLOCK;
;      ACT(24)/32,RNORM(4,0.4);
;                                     CRC IV INPROCESS/INTER
;
;      QUEUE(33),,30000,BLOCK;
;      ACT(24)/33,RNORM(16,1.6);
;                                     CRC IV POR
;
;      QUEUE(34),,30000,BLOCK;
;      ACT(45)/34,RNORM(90,9);
;                                     CRC IV OCIE
;
;      QUEUE(35),,30000,BLOCK;
;      ACT(150)/35,RNORM(135,13.5);
;                                     CRC IV WPN ZERO
;
;      QUEUE(36),,30000,BLOCK;
;      ACT(75)/36,RNORM(53,5.3);
;                                     CRC IV MASK TEST
;
;      QUEUE(37),,30000,BLOCK;
;      ACT (300)/37,RNORM(240,24);
;                                     CRC IV MEDICAL
;
;      GOON,1;
;      QUEUE(38),,30000,BLOCK;
;      ACT(60)/38,RNORM(48,0),0.2;
;                                     CRC IV OPTICAL
;      ACT/39,,0.8;
;
;      QUEUE(39),,30000,BLOCK;
;      ACT(3000)/40,RNORM(360,36),,EXIT; CRC IV MISC/MEALS
;

```

```

;***** FORT LEWIS (CRC V) *****
;

```

```

C5      GOON;
;
;      Zero day allows number of personnel shown to enter a CRC.
;
;      ASSIGN, ATRIB(11)=5;
;      QUEUE(41),,30000,BLOCK;
;      ACT(300)/41,RNORM(720,0);
;                                     CRC V DAY ZERO
;
;      ASSIGN, ATRIB(6)=TNOW;
;

```



```

;
;   QUEUE(57),,30000,BLOCK;
;   ACT(300)/57,RNORM(240,24);                                CRC VI MEDICAL
;
;   GOON,1;
;   QUEUE(58),,30000,BLOCK;
;   ACT(60)/58,RNORM(48,0),0.2;                                CRC VI OPTICAL
;   ACT/59,,0.8;
;
;   QUEUE(59),,30000,BLOCK;
;   ACT(3000)/60,RNORM(360,36),,EXIT;  CRC VI MSC/MEALS
;
;*****      FORT DIX  (CRC VII)      *****
;
C7      GOON;
;
;   Zero day allows number of personnel shown to enter a CRC.
;
;   ASSIGN,ATRI(11)=7;
;   QUEUE(61),,30000,BLOCK;
;   ACT(300)/61,RNORM(720,0);                                CRC VII DAY ZERO
;
;   ASSIGN,ATRI(6)=TNOW;
;
;   QUEUE(62),,30000,BLOCK;
;   ACT(24)/62,RNORM(4,0.4);                                CRC VII INPROCESS/INTER
;
;   QUEUE(63),,30000,BLOCK;
;   ACT(24)/63,RNORM(16,1.6);                                CRC VII POR
;
;   QUEUE (64) , ,30000,BLOCK;
;   ACT(45)/64,RNORM(90,9);                                CRC VII OCIE
;
;   QUEUE(65),,30000,BLOCK;
;   ACT(150)/65,RNORM(135,13.5);                                CRC VII WPN ZERO
;
;   QUEUE(66),,30000,BLOCK;
;   ACT(75)/66,RNORM(53,5.3);                                CRC VII MASK TEST
;
;   QUEUE(67),,30000,BLOCK;
;   ACT(300)/67,RNORM(240,24);                                CRC VII MEDICAL
;
;   GOON,1;
;   QUEUE(68),,30000,BLOCK;
;   ACT(60)/68,RNORM(48,0),0.2;                                CRC VII OPTICAL
;   ACT/69,,0.8;
;
;   QUEUE(69),,30000,BLOCK;
;   ACT(3000)/70,RNORM(360,36),,EXIT;  CRC VII MISC/MEALS
;
;*****      FORT LEONARD WOOD (CRC VIII)      *****
;

```

```

C8      GOON;
;
;      Zero day allows number of personnel shown to enter a CRC.
;
;      ASSIGN, ATRIB(11)=8;
;      QUEUE(71),,30000,BLOCK;
;      ACT(300)/71,RNORM(720,0);          CRC VIII DAY ZERO
;
;      ASSIGN, ATRIB(6)=TNOW;
;
;      QUEUE(72),,30000,BLOCK;
;      ACT(24)/72,RNORM(4,0.4);          CRC VIII INPROCESS/INTERVIEW
;
;      QUEUE(73),,30000,BLOCK;
;      ACT(24)/73,RNORM(16,1.6);          CRC VIII POR
;
;      QUEUE(74),,30000,BLOCK;
;      ACT(45)/74,RNORM(90,9);           CRC VIII OCIE
;
;      QUEUE(75),,30000,BLOCK;
;      ACT(150)/75,RNORM(135,13.5);      CPC VIII WPN ZERO
;
;      QUEUE(76),,30000,BLOCK;
;      ACT(75)/76,RNORM(53,5.3);          CRC VIII MASK TEST
;
;      QUEUE(77),,30000,BLOCK;
;      ACT(300)/77,RNORM(240,24);          CRC VIII MEDICAL
;
;      GOON,1;
;      QUEUE(78),,30000,BLOCK;
;      ACT(60)/78,RNORM(48,0),0.2;        CRC VIII OPTICAL
;      ACT/79,,0.8;
;
;      QUEUE(79),,30000,BLOCK;
;      ACT(3000)/80,RNORM(360,36),,EXIT;  CRC VIII MISC/MEALS
;

```

```

;      TIME IN SYSTEM CALCULATION & EXCESSIVE TRANSPORT TIME CALCULATION
;

```

ENTITIES ARE HELD FOR 3 DAYS, THE REQUIRED TIME TO RESERVE AN AIR TRANSPORTATION SEAT. IF PROCESSING IS GREATER THAN REQUIRED DELAY ENTITIES ARE NOT DELAYED
--

```

EXIT  ASSIGN, ATRIB(7)=TNOW-ATRIB(6);
;      ASSIGN, ATRIB(8)=RNORM(2160,120)-ATRIB(7);
;      TRANSPORTATION RATE OF(3 DAYS +-2 HOURS)-TIME IN SYSTEM=EXCESS TRANSPORT
;
;
;      Personnel are held at this activity until the air transportation
;      waiting requirement has been satisfied.
;

```

```

;
GOON,1;
ACT/81;    COMPLETED PROCESSING
;
; *****TRANSPORTATION CAPACITY CHECK*****
;
; Version 1.0 limits available air transportation to 32,000 seats per
; time period.
;
GOON,1;
ACT/82,,NNCNT(86).LE.32000,G1;GET TRANSPORTED
ACT/83,,NNCNT(86).GT.32000,T1;RAN OUT OF TRANSPORTATION
;
; *****PROCESS EXCESS TRANSPORTATION TIME IF ANY*****
;
G1 GOON,1;
ACT/84,TRIB(8),TRIB(8).GT.0.0;    WAITING    TRANSPORTATION
ACT/85,,TRIB(8).LE.0.0;          NOT WAITING
;
GOON;
ACT/86;    COMPLETE TRANSPORT
;
; EVENT 2 loops through the FORTRAN programs to store the SLAM
; output in an array.
;
EVENT,2;
;
;
; T2 statement allows simulation to continue.
T2 TERM;
;
; EVENT 2 loops through the FORTRAN programs to store the SLAM
; output in an array.
;
EVENT,2;
;
;
; T2 statement allows simulation to continue.
T2 TERM;
;
;
; T1 statement terminates program.
T1 TERM,1;
ENDNETWORK;
;
; The MONTR statement is set differently for time period and type
; processing operation.
;

```

```
; For CRC and time period 1, set at 720 (one 12 hour day) for the
; zero day.
;
; For CRC and time period 2-18, set at the time it takes for the first
; entity to exit the system. For Version 1.0, this is 2160 (air trns
; wait time).
;
; For RPL and time period 1, comment out MONTR statement. It is not
; required.
;
; For RPL and time period 2-18, set at 1110, the time required to process
; all the processing stations.
;
MONTR,CLEAR,720;
FIN;
```

MONTR CLEARS STATS AT 720 FOR CRC1 AND AT 2160 FOR CRC2.DAT - CRC18. THIS ALLOWS THE SYSTEM TO REACH A STEADY STATE FOR 2-18
--

6.9 RPL MODEL SPECIFIC PROGRAMS

6.9.1 FORTRAN PROGRAMS SUPPORTING SLAM II RPL CO PROGRAMS - PROGRAM MAIN

C*****

C Program Name: RPLEXE Date: 12-03-1990

C

C File Name: RPLNPUT.F

C

C Programmer: Beth A. Wilson, SAIC, (703)749-8771

C

C Description: FORTRAN programs supporting SLAM II CRC program
C Program MAIN which includes CRC Subroutine INTLC,
C Subroutine EVENT, Subroutine OPUT.

C

C Input: ASETS.TMP

C

C Output: TRSLT.TMP

C

C*****

C Modifications: (STATUS: P - PROPOSED; R - REQUIRED;
C - COMPLETED)

C

C Number Status Date: Description: Initials

C

C 01 12-03-90 Modified Subroutine EVENT such BAW
C that all data records in the
C file: ASETS.TMP would be read
C and the accumulated strencth
C would be calculated and passed
C back to the SLAM II program
C QUEUE for unbatching and process-
C ing in the CRC model.

C

C 02 01-30-91 Documentation of arious calcula BAW
C tions and procedures.

C

C*****

PROGRAM MAIN

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

*
* SLAM II VERSION 4.03 *
*

DIMENSION NSET(1800000)

```

PARAMETER (KVACP1=1,KVACP2=2,KVALEN=3,KVACAP=4,KVADIR=5,KVANTx=6,
$ KVANCO=7,KVASTO=8,KVAMNO=9,KVATLU=10,KVAPFE=11,KVAPLE=12,
$ KVCSPR=1,KVCNWL=2,KVCPFE=3,KVCPLE=4,KVCMXL=5,KVCPPL=6,
$ KVCCRC=7,KVCNTE=8,KVCCST=9,KVCSTB=10,KVCTLU=11,KVCVCO=12,
$ KVCRRC=13,KVCCPI=14,KVSESP=1,KVSLSP=2,KVSACC=3,KVSDEC=4,KVSLN=5,
$ KVSBUF=6,KVSCKZ=7,KVSIFL=8,KVSJRQ=9,KVSINI=10,KVSREP=11,
$ KVSNTL=12,KVSNTU=13,KVSNU=14,KVSNUU=16,KVSNU=18,KvSNUF=20,
$ KV8NUC=22,KVSNU=24,KVSTE=26,KVSSTF=27,KVSTLU=28,KVSPFE=29,
$ KVSPL=30,KVSPAL=31,KVSNV32,KVUPV8=1,KVUCSG=2,KVUCCP=3,
$ KVUICP=4,KVUDCP=5,KVUPCL=6,KVUVM0=7,KVUCSI=8,KVUSPD=9,KVUCBF=10,
$ KVUCBT=11,KVUTLU=12,KVUSCP=13,KVUSHT=14,KVUNTL=15,KVUSPT=16,
$ KVFVSI=4,KVWIFL=4,KVWVSI=5,KVWVRQ=6,KVWVRE=7,
$ KVMCPI=8,KVMCPI=4,KVANWF=13,KVCNWF=15,KVFNWF=5,KVMNWF=4,
$ KVSNUF=33,KVUNWF=15,KVWNWF=8)
PARAMETER (MXMSG=250)
PARAMETER (MXPOUT=6)
INCLUDE 'PARAM.INC'
COMMON/SCOM1/ATRIB(MATRB), DD(MEQT), DDL(MEQT), DTNOW, II,
IMFA,MSTOP,NCLNR, NCRDR, NPRNT, NNRUN, NNSET, NTAPE, SS(MEQT),
2SSL(MEQT),TNEXT, TNOW, XX(100)
COMMON QSET(1800000)
EQUIVALENCE (NSET (1),QSET(1))
NNSET=1800000
NCRDR=5
NPRNT=6
NTAPE=7
OPEN(UNIT=NCRDR,FILE='fort.5')
OPEN(UNIT=NPRNT,FILE='fort.6')
CALL SLAM
STOP
END

```

```

C*****
C
C Subroutine: INTLC
C
C Description: Initializes SLAM II variables.
C
C*****

```

```

SUBROUTINE INTLC

```

```

INCLUDE 'PARAM.INC,

```

```

CHARACTER*2 TP

```

```

INTEGER TTFIN

```

```

REAL ATRIB(5),ATRIB(12)
COMMON/SCOM1/ATRIB(MATRB), DD(MEQT), DDL(MEQT), DTNOW, II,
1MFA,MSTOP,NCLNR, NCRDR, NPRNT, NNRUN, NNSET, NTAPE, SS(MEQT),
2SSL(MEQT),TNEXT, TNOW, XX(100)

```

```

* Initialize SLAM II variables.

```

```

ATRIB(1)=0.
ATRIB(2)=0.
ATRIB(3)=0.
ATRIB(4)=0.
ATRIB(5)=0.
ATRIB(9)=0.
ATRIB(10)=0.
ATRIB(11)=0.
ATRIB(12)=0.

```

```

IF (TP.EQ. 01') THEN

```

```

    TTFIN = 7200

```

```

    GOTO 999

```

```

ELSE

```

```

    TTFIN = 7680

```

```

* MONTR,CLEAR,480;

```

```

ENDIF

```

```

999 CALL ENTER(1,ATRIB)

```

```

RETURN

```

```

END

```

```

C*****
C
C Subroutine:  EVENT
C
C Description: Reads in all data from ASETS.TMP file.
C
C*****

```

```

SUBROUTINE EVENT(I)

```

```

    DIMENSION CRCTRANS(18,67,7,8),CBGTRANS(18,67,7)
    INCLUDE 'PARAM.INC'

```

```

    INTEGER NDEX,STR,CRCTRANS,CBGTRANS,CRC1CNT,CRC2CNT,CRC3CNT,
&CRC4CNT,CRC5CNT,CRC6CNT,CRC7CNT,CRC8CNT

```

```

    REAL CRC1TIM,CRC2TIM,CRC3TIM,CRC4TIM,CRC5TIM,CRC6TIM,
&CRC7TIM,CRC8TIM,CRC1DLY,CRC2DLY,CRC3DLY,CRC4DLY,CRC5DLY,
&CRC6DLY,CRC7DLY,CRC8DLY,C1AVGTIM,C2AVGTIM,C3AVGTIM,
&C4AVGTIM,C5AVGTIM,C6AVGTIM,C7AVGTIM,C8AVGTIM,C1AVGDLY,
&C2AVGDLY,C3AVGDLY,C4AVGDLY,C5AVGDLY,C6AVGDLY,C7AVGDLY,
&C8AVGDLY

```

```

    CHARACTER CBG*5,SEX*1,ASETTYPE*3,TP*2,BRC*3,TYPE*4

```

```

    COMMON/SCOM1/ATRIB(MATRB), DD(MEQT), DDL(MEQT), DTNOW, II,
1MFA,MSTOP,NCLNR, NCRDR, NPRNT, NNRUN, NNSET, NTAPE, SS(MEQT),
2SSL(MEQT),TNEXT,TNOW, XX(100)

```

```

    COMMON/UCOM1/CBG,SEX,ASETTYPE,TP,BRC,TYPE,STR,NDEX
    COMMON/CRARY/CRCTRANS,CBGTRANS,CRC1CNT,CRC2CNT,CRC3CNT,
&CRC4CNT,CRC5CNT,CRC6CNT,CRC7CNT,CRC8CNT,CRC1TIM,CRC2TIM,
&CRC3TIM,CRC4TIM,CRC5TIM,CRC6TIM,CRC7TIM,CRC8TIM,CRC1DLY,
&CRC2DLY,CRC3DLY,CRC4DLY,CRC5DLY,CRC6DLY,CRC7DLY,CRC8DLY,
&C1AVGTIM,C2AVGTIM,C3AVGTIM,C4AVGTIM,C5AVGTIM,C6AVGTIM,
&C7AVGTIM,C8AVGTIM,C1AVGDLY,C2AVGDLY,C3AVGDLY,C4AVGDLY,
&C5AVGDLY,C6AVGDLY,C7AVGDLY,C8AVGDLY

```

```

    GOTO (1,2),I

```

```

* Open input file: ASETS.TMP, read entire file.
* Open output file: ASETS2.TMP and stores the ASETS.TMP such
* that the time period, branch and type are concatenated to
* a nine character code.

```

```

1  OPEN(30,FILE='/home/warpam/iofiles/ASETS.TMP',STATUS='OLD')
   OPEN(31,FILE='/home/warpam/iofiles/ASETS2.TMP',,STATUS='OLD')

```

```

14  READ(30,14,ERR=20,END=25)CBG,SEX,ASETTYPE,TP,BRC,TYPE,STR
    FORMAT(2X,A5,4X,A1,4X,A3,3X,A2,3X,A3,3X,A4,3X,I6)

```

```

*  WRITE(6,14)CBG,SEX,ASETTYPE,TP,BRC,TYPE,STR

```



```

15  WRITE(31,15)CBG,SEX,ASETTYPE,TP,BRC,TYPE,STR
    FORMAT(2X,A5,4X,A1,4X,A3,3X,A2,A3,A4,3X,16)

```

* Assign entity attribute Values.

```

    ATRIB(1)=CBG
    ATRIB(2)=SEX
    ATRIB(3)=ASETTYPE

```

* Converts character string into an integer Value and
 * assigns it to the appropriate attribute ATRIB().

```

    NTP=0
    NBRC=0
    NTYPE=0

    DO 16 IJ=1,2
        I1=ICHAR(TP(IJ:IJ))
        NUM=(79-(127-I1))
        IF(IJ.EQ.1)NUM=NUM*10
        IF(IJ.EQ.2)NUM=NUM*1
        NTP=NTP+NUM
16  CONTINUE
    ATRIB(4)=NTP

    ATRIB(5)=STR

    DO 17 J=1,3
        IF(J.EQ.1)GOTO 17
        I1=ICHAR(BRC(J:J))
        NUM=(79-(127-I1))
        IF(J.EQ.2)NUM=NUM*10
        IF(J.EQ.3)NUM=NUM*1
        NBRC=NBRC+NUM
17  CONTINUE
    ATRIB(9)=NBRC

    DO 18 K=1,2
        IF(K.EQ.1)GOTO 18
        I1=ICHAR(TYPE(K:K))
        NUM=(79-(127-I1))
        IF(K.EQ.2)NTYPE=NUM
18  CONTINUE
    ATRIB(10)=NTYPE

    GOTO 9999

20  WRITE(6,*),ERROR READING FILE: ASETS.TMP'
25  CLOSE(30,STATUS='KEEP')
    CLOSE(31,STATUS='KEEP,')
    XX(1)= 1.0

```

9999 RETURN

- * Begins processing EVENT(I) a second time and stores shipped
- * persons into an array.

```
2  XTP = ATRIB(4)
   XBRC = ATRIB(9)
   XTYPE = ATRIB(10)

   ITP = (NINT(XTP))
   IBRC = (NINT(XBRC))
   ITYPE = (NINT(XTYPE))

   NDEX = ATRIB(11)

   CRCTRANS(ITP, IBRC, ITYPE, NDEX) = CRCTRANS(ITP, IBRC, ITYPE, NDEX) + 1
   CBGTRANS(ITP, IBRC, ITYPE) = CBGTRANS(ITP, IBRC, ITYPE) + 1

   XTIM = ATRIB(7)
   XDLY = ATRIB(8)

   IF (NDEX.EQ.1) THEN
     CRC1CNT = CRC1CNT + 1
     CRC1TIM = CRC1TIM + XTIM
     CRC1DLY = CRC1DLY + XDLY
   ENDIF

   RETURN
   END
```

```

C*****
C
C      Subroutine:  OPUT
C
C      Description: Generates an output file called TRSLT.TMP.
C
C*****

```

SUBROUTINE OPUT

```

      DIMENSION      CRCTRANS(18,67,7,8),CBGTRANS(18,67,7)      INCLUDE
      'PARAM.1NC'

```

```

      INTEGER NDEX,STR,NTP,NBRC,NTYPE,CRCTRANS,CBGTRANS,
&CRC1CNT,CRC2CNT,CRC3CNT,CRC4CNT,CRC5CNT,CRC6CNT,CRC7CNT,
&CRC8CNT

```

```

      REAL CRC1TIM,CRC2TIM,CRC3TIM,CRC4TIM,CRC5TIM,CRC6TIM,
&CRC7TIM,CRC8TIM,CRC1DLY,CRC2DLY,CRC3DLY,CRC4DLY,CRC5DLY,
&CRC6DLY,CRC7DLY,CRC8DLY,C1AVGTIM,C2AVGTIM,C3AVGTIM,C4AVGTIM,
&C5AVGTIM,C6AVGTIM,C7AVGTIM,C8AVGTIM,C1AVGDLY,C2AVGDLY,
&C3AVGDLY,C4AVGDLY,C5AVGDLY,C6AVGDLY,C7AVGDLY,C8AVGDLY

```

```

      CHARACTER CBG*5,SEX*1,ASETTYPE*3,TP*2,BRC*3,TYPE*4

```

```

      COMMON/SCOM1/ATR1B(MATRB),DD(MEQT),DDL(MEQT),DTNOW,II,
1MFA,MSTOP,NCLNR, NCRDR, IPRNT, NNRUN, NNSET, SS(MEQT),
2SSL(MEQT),TNEXT,TNOW, XX(100)

```

```

      COMMON/UCOM1/CBG,SEX,ASETTYPE,TP,BRC,TYPE,STR,NDEX
      COMMON/CRARY/CRCTRANS,CBGTRANS,CRC1CNT,CRC2CNT,CRC3CNT,
&CRC4CNT,CRC5CNT,CRC6CNT,CRC7CNT,CRC8CNT,CRC1TIM,CRC2TIM,
&CRC3TIM,CRC4TIM,CRC5TIM,CRC6TIM,CRC7TIM,CRC8TIM,CRC1DLY,
&CRC2DLY,CRC3DLY,CRC4DLY,CRC5DLY,CRC6DLY,CRC7DLY,CRC8DLY,
&C1AVGTIM,C2AVGTIM,C3AVGTIM,C4AVGTIM,C5AVGTIM,C6AVGTIM,
&C7AVGTIM,C8AVGTIM,C1AVGDLY,C2AVGDLY,C3AVGDLY,C4AVGDLY,
&C5AVGDLY,C6AVGDLY,C7AVGDLY,C8AVGDLY

```

LOGICAL THERE

- * Checks to see if output file: TRSLT.TMP exists. If the
- * file exist, then the old file is deleted and a new file
- * is then opened.

```

      INQUIRE(FILE= '/home/warpam/iofiles/TRSLT.TMP',,EXIST=THERE)
      IF (THERE)THEN
        OPEN(32,FILE= '/home/warpam/iofiles/TRSLT.TMP',,STATUS='OLD')
        CLOSE(32,STATUS='DELETE')
      ENDIF

```

```

      OPEN(32,FILE= '/home/warpam/iofiles/TRSLT.TMP',STATUS='NEW')
      OPEN(31,FILE= '/home/warpam/iofiles/ASETS2.TMP',STATUS='OLD')

```

```

11  READ(31,12,ERR=17,END=18)CBG,SEX,ASETTYPE,TP,BRC,TYPE,STR
12  FORMAT(2X,A5,4X,A1,4X,A3,3X,A2,A3,A4,3X,16)

      NTP=0
      NBRC=0
      NTYPE=0

      DO 13 JJ=1,2
        II=ICHAR(TP(JJ:JJ))
        NUM=(79-(127-II))
        IF(JJ.EQ.1)NUM=NUM*10
        IF(JJ.EQ.2)NUM=NUM*1
        NTP=NTP+NUM
13  CONTINUE

      DO 14 KK=1,3
        IF(KK.EQ.1)GOTO 14
        II=ICHAR(BRC(KK:KK))
        NUM=(79-(127-II))
        IF(KK.EQ.2)NUM=NUM*10
        IF(KK.EQ.3)NUM=NUM*1
        BRC=NBRC+NUM
14  CONTINUE

      DO 15 LL=1,2
        IF(LL.EQ.1)GOTO 15
        II=ICHAR(TYPE(LL:LL))
        NUM=(79-(127-II))
        IF(LL.EQ.2)NTYPE=NUM
15  CONTINUE

*      WRITE(6,16)CBG,SEX,ASETTYPE,TP,BRC,TYPE,STR,
*      &CBGTRANS(NTP,NBRC,NTYPE)
*      WRITE(32,16)CBG,SEX,ASETTYPE,TP,BRC,TYPE,STR,
*      &CBGTRANS(NTP,NBRC,NTYPE)
16  FORMAT(2X,A5,4X,A1,4X,A3,3X,A2,A3,A4,3X,16,3X,16)

      GOTO 11

17  WRITE(6,*)'ERROR READING FILE:  ASETS2.TMP'
18  CLOSE(31,STATUS='KEEP')
   CLOSE(32,STATUS='KEEP')

*  Calculate average processing time and average delay
*  for each CRC (CRC1 thru CRC8).
      CIAVGTIM = CRC1TIM/CRC1CNT
      CIAVGDLY = CRC1DLY/CRC1CNT

      WRITE(6,20)
20  FORMAT(//15X,'AVERAGE TIME FOR THIS TIME PERIOD',//3X,
&' CRC',4X,'COUNT',8X,'AVG. PROCESSING TIME',8X,
&'AVG. DELAY TIME',/)

```

```
21  WRITE(6,21)CRC1CNT,C1AVGTIM,C1AVGDLY  
    FORMAT(3X,'I ',4X,I5,10X,F10.1,16X,F10.1)
```

```
    RETURN  
    END
```

6.9.2

RPLBN MODEL PROCESSING FLOW

Figure 11 depicts the processing flow within the SLAM II portion of the RPLBN model.

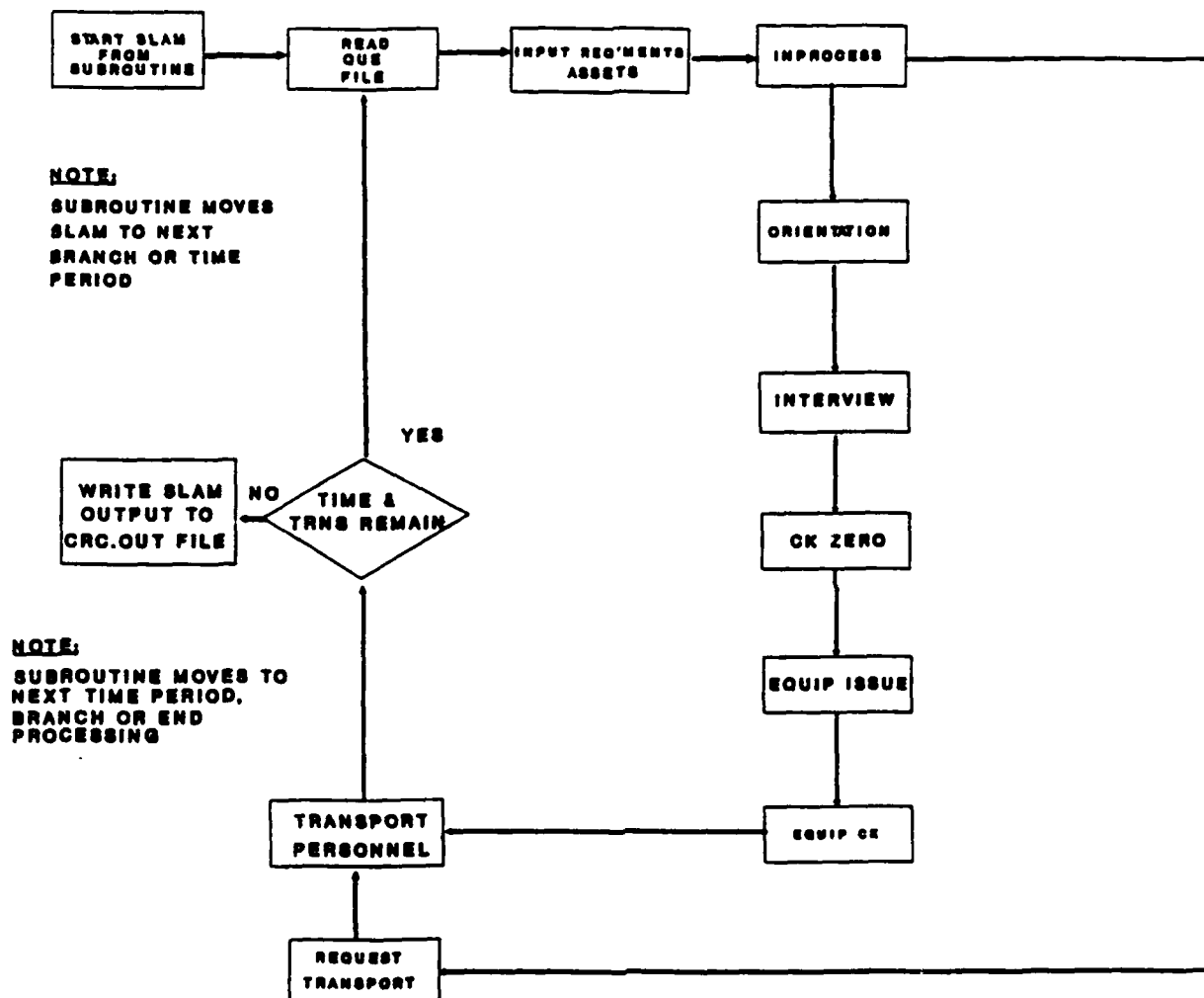


FIGURE 11: RPLBN SLAM II PROCESSING

THE RPLBN SIMULATION IS CONFIGURED VERY SIMILAR TO THE CRC MODEL, BUT WITH ONLY 6 STATIONS WITH DIFFERENT ACTIVITIES AND DURATIONS. EACH STATION IS CONFIGURED TO ALLOW CONTINUOUS FLOW WITHOUT REQUIRING A BATCH MOVEMENT FROM ONE STATION TO THE NEXT.

THE RPLBN MODEL HAS TIME AND TRANSPORTATION CONSTRAINTS, BUT THE TRANSPORTATION CONSTRAINT HAS BEEN IN ESSENCE INACTIVATED BY SETTING IT AT 99,999 CAPACITY. THERE IS NO FUNCTION FOR TRANSPORTATION DELAY.

```

;*****
;   SLAM II VARIABLES
;   ATRIB(1) = CAT/BRANCH AND GRADE ACROYMN
;   ATRIB(2) = MALE OR FEMALE
;   ATRIB(3) = ASSET TYPE
;   ATRIB(4) = TIME PERIOD
;   ATRIB(5) = REQUIREMENT ASSET QUANTITY
;   ATRIB(6) = ARRIVAL TIME TO CRC
;   ATRIB(7) = TOTAL TIME IN THE CRC
;   ATRIB(8) = EXCESS TRANSPORTATION TIME OF (72 hours +/- 40 hours)
;   ATRIB(9) = BRANCH INDEX
;   ATRIB(10)= TYPE ASSET INDEX
;   ATRIB(11)= CRC INDEX
;   ATRIB(12)= COUNTER TO COLLECT STATISTICS
;*****
;
;GEN,WILSONIWOJCIK1FRAME,WARPAM1VERSION II. 1/30/91,,N,N,Y/N,N,Y/F,72;
;LIMITS,15,14,80000;
;EQUIVALENCE/ATRIB(1),CBG;
;EQUIVALENCE/ATRIB(2),SEX;
;EQUIVALENCE/ATRIB(3),ASETYPE;
;EQUIVALENCE/ATRIB(4),TP;
;EQUIVALENCE/ATRIB(5),STR;
;EQUIVALENCE/ATRIB(9),BRC;
;EQUIVALENCE/ATRIB(10),TYPE;
;
;   The INIT statement sets the length of the simulation for CRC
;   and time period 1. It is set in minutes at 7919 (11 days)
;   (actually one minute ;less) to represent 1 zero day and 10
;   processing days.
;
;   For CRC and time periods 2-18, it is set at 10059 representing
;   the same 11 days plus the time required to fill the system for
;   steady state operation which for version 1.0 of WARPAM is 3 days
;   (2160 min) which is the time the first entities would be available
;   to exit the system.
;
;   For RPL and time period 1, it is set at 7200 representing 10 days.
;

```

```
; For RPL and time period 2-18, it is set at 9009 representing 13 days
; (10 processing days and 1110 min., the time required to complete
; processing the 6 stations in RPL version 1.0).
```

```
INIT,,7200;
NETWORK;
```

```
INIT SET AT 7200 FOR RPL1.DAT
INIT SET AT 7590 FOR RPL2.DAT - RPL18.DAT
```

```
*** REPLACEMENT OPERATION BEGINS!!!! ***
```

```
; ENTER,1;
; EVENT 1 clones the incoming entities. One entity moves through the
; normal system and one is returned to the FORTRAN program where it
; causes the next line to be read and then is destroyed.
```

```
LOOP EVENT,1;
GOON,2;
ACT,,XX(1).EQ.0.0;
ACT,,XX(1).EQ.0.0,LOOP; READ IN NEXT LINE
ACT,,XX(1).EQ.1.0,T2; TERMINATE DUPLICATE OF LAST ASSET LINE
```

```
; GOON;
; ACT/15; NUMBER OF ENTITIES
; GOON;
; UNBATCH,5;
```

```
; GOON;
; ASSIGN,ATRI(6)=TNOW;
; GOON;
```

```
;*****
```

```
OCONUS REPLACEMENT ACTIVITY
```

```
*****
```

QUEUE(1),,2,BLOCK;	
ACT(2)/1,RNORM(30,5);	RPL INPROCESSING/INTERVIEW
QUEUE(2),,50,BLOCK;	
ACT(50)/2,RNORM(60,10);	RPL ORIENTATION
QUEUE(3),,50,BLOCK;	
ACT(2)/3,RNORM(30,5);	RPL EQUIP/DRAW
QUEUE(4),,100,BLOCK;	
ACT(100)/4,RNORM(480,20);	RPL CHECK ZERO
QUEUE(5),,50,BLOCK;:	
ACT(50)/5,RNORM(480,20);	RPL EQUIPMENT ISSUE
QUEUE(6),,50,BLOCK;	
ACT(4)/6,RNORM(30,5);	RPL EQUIP/CHECK

PROGRAMMER'S NOTE: THE FOLLOWING SECTION REGARDING A MINIMUM PROCESSING TIME HAS BEEN RETAINED, BUT COMMENTED OUT SHOULD TRAC-FBHN DESIRE TO REINSTATE A DELAY. THE TRANSPORTATION CAPACITY SECTION WAS NOT COMMENTED OUT, BUT SET AT 99,000 WHICH EFFECTS NO CONSTRAINT. A TRANS CONSTRIANT CAN BE ADDED BY CHANGING THIS NUMBER TO LOWER FIGURE. THE MONTR STATEMENT WAS ALSO COMMENTED OUT FOR THE SAME REASON

```

;
;   TIME IN SYSTEM CALCULATION & EXCESSIVE TRANSPORT TIME CALCULATION
;EXIT ASSIGN,TRIB(7)=TNOW-TRIB(6);
;   ASSIGN,TRIB(8)=RNORM(2160,120)-TRIB(7);
;   TRANSPORTATION RATE OF(3 DAYS +-2 HOURS)-TIME IN SYSTEM=EXCESS TRANSPORT
;
;
;   Personnel are held at this activity until the air transportation
;   waiting requirement has been satisfied.
;
;   GOON,1;
;   ACT/8;      COMPLETED PROCESSING
;
; *****TRANSPORTATION CAPACITY CHECK*****
;
;
;   Version 1.0 limits available air transportation to 99,999 seats per
;   time period.
;
;   GOON,1;
;   ACT/9,,NNCNT(13).LE.99999,G1;GET TRANSPORTED
;   ACT/10,,NNCNT(13).GT.99999,T1;RAN OUT OF TRANSPORTATION
;
; *****PROCESS EXCESS TRANSPORTATION TIME IF ANY*****
;G1  GOON,1;
;   ACT/11,TRIB(8),TRIB(8).GT.0.0; WAITING TRANSPORTATION
;   ACT/12,,TRIB(8).LE.0.0;      NOT WAITING
;
;   GOON;
;G1  ACT/13: COMPLETE TRANSPORT
;
;   EVENT 2 loops through the FORTRAN programs to store the SLAM
;   output in an array.
;
;   EVENT,2;
;
;
;   T2 statement allows simulation to continue.
;
;T2  TERM;
;
;
;   T1 statement terminates program.

```

```

;
T1 TERM,1;
ENDNETWORK;

;
; The MONTR statement is set differently for time period and type
; processing operation.
;
; For CRC and time period 1, set at 720 (one 12 hour day) for the
; zero day.
;
; For CRC and time period 2-18, set at the time it takes for the first
; entity to exit the system. For Version 1.0, this is 2160 (air trns
; wait time).
;
; For RPL and time period 1, comment out MONTR statement. It is not
; required.
;
; For RPL and time period 2-18, set at 1110, the time required to process
; all the processing stations.
;
; MONTR,CLEAR,720;
FIN;

```

MONTR CLEAR SET INACTIVE FOR RPL1.DAT FOR RPL2.DAT - RPL18.DAT IT IS SET AT 390 MIN, THE MEAN TIME TO PROCESS STATIONS
--

SECTION 7 TRANSPORTATION MODEL

7.1 GENERAL

The Transportation Model is designed to represent the macro-level flow of personnel replacements through the CRC(s) and a specified OCONUS Replacement battalion (RPLBN). The model matches the replacement flow through the CRC and RPLBN to determine if these organizations can meet the requirements for a theater and if the flow is balanced through the two facilities. Statistics are provided regarding the replacement requirement satisfied and the difference in flow capacities of these facilities. The model uses the output files from a CRC model run and a RPLBN model run. The files selected should be based on the same requirement file and number of time periods to produce meaningful results. Currently, the user can select the any of the single theater requirement files: Europe (AE1), Korea (AKO), maximum flow (MAX) or either of the CSM II files (CST or CSB).

7.2 INITIATION

The Transportation Model is initiated through user input from a Sun window which activates the FORTRAN program. This window is reached by using the WARPAM Executive Windows Program which allows the user to reach any module by simply placing the workstation mouse over the appropriate window. To initiate the model, the user must type "go" on the response line to advance to the first input variable. This input line ONLY ACCEPTS the word "go" in small case letters.

7.3 INPUT FILES

The files which must be present on the workstation before the model can be run are listed with the code description below.

7.4 INPUT VARIABLES

The user is prompted by the input screen to input the desired value of the following variables on a response line: (input variables from previous runs are shown on the input screen prior to the first response)

CRC: The user must select a CRC output file from those stored in the IOFILE sub-directory.

REPL BN: The user must select a Replacement Battalion output file from those stored in the IOFILE sub-directory.

7.5 PROCESSING

The Transportation Model, written in FORTRAN, reads the output files from designated CRC and RPL BN runs and writes portions of these to a file with the MODRQAST.TBL. The result is an output which allows the user to compare the flow through a CRC and a RPL BN. The model also computes statistics on the systems ability to meet the demand based on the RPL BN flow and on the difference between the CRC and RPL BN flows. To accomplish this the model reads the required data from the two output files and appends these to the MODRQAST.TBL. Statistics are calculated following these reads.

7.6 OUTPUT REPORTS

Output reports are generated for each run of the model. These reports which are automatically translated to DOS by the TRAC-FBHN system are available to the user through DBASE III on a standard PC in a LAN configuration with the workstation or in the Sun DOS window program.

7.7 TRANSPORTATION MODEL FORTRAN PROGRAMS

```
C*****
C****
C
C Program Name:  TRANSP                      Date:  06-15-1990
C
C File Name:    TRANSPRT.FOR
C
C Programmer:   Beth White, SAIC, 749-8771
C
C Description:  Reads input files:  CRC---.OUT and RPF---.OUT
C               (i.e. CRCDEG.OUT , RPLAE1.OUT) and produces
C               a transportation output file. Calculations
C               performed a flow difference and theater percent.
C
C Input:        CRC---.OUT
C               RPL---.OUT
C
C Output:       TRANS.OUT
C
C*****
C Modifications: (STATUS:  P - PROPOSED; R - REQUIRED ; C - COMPLETED)
C
C Number   Status   Date:           Description:           Initials
C -----
C
C*****
```

PROGRAM TRANSP

C Global Variables

```
DIMENSION VARHLD(1300,7)
CHARACTER*1 XTCHR(58),SEX
CHARACTER*3 TYP,RDRESP,SREQ,XCRC,XRPL,XFIL
CHARACTER*4 XTEND
CHARACTER*5 CATBRG
CHARACTER*6 REQT,REQFIL,CRCXTN,RPLXTN
CHARACTER*9 TP,VARHLD
CHARACTER*10 CRCF,RPLF
CHARACTER*21 FILH
CHARACTER*31 CRCFIL,RPLFIL
CHARACTER*58 XCHR
CHARACTER*69 HEAD1
CHARACTER*70 HEAD0
CHARACTER*72 HEAD2
```

LOGICAL THERE

REAL THTRP

INTEGER I,J,LL,NN,NCHK,ICLK,MAXCNT,RQMT,CRCFLO,RPLFLO,DELFLO

EQUIVALENCE (XTCHR(1),XCHR)

C Local Variables

```
I = 0
J = 0
NN = 0
NCHK = 0
ICLK = 0
XTEND = '.OUT'
FILH = '/home/warpam/iofiles/'
```

C Begin TRANSPRT.FOR

C Begin Menu Screen

```
WRITE(6,9)
9  FORMAT(//////20X,'*****',
&/20X,'*****',
&'      WARPAM TRANSPORTATION MODEL      ',//20X,
&'*****',/20X,
&'*****',////////20X,
&'THE FOLLOWING FILES ARE NEEDED:',/30X,
&'CRC---.OUT   {i.e. CRCDEG.OUT}',/30X,
&'RPL---.OUT   {i.e. RPLAE1.OUT}',/////////))
```

PAUSE

```
WRITE(6,10)
10  FORMAT(//////////)
```

C End Menu Screen

C Input Variable (CRC File: CRC---.OUT)

```
111 WRITE(6,11)
11  FORMAT(/10X,'ENTER CRC REQUIREMENT FILE:',/10X,
&'REQUIREMENTS: (MAX,DEG,AE1,AKO,ASW,CST,CSB)')
SREQ = 'XXX'
XFIL = 'CRC'
READ(*,*)RDRESP

IF ((RDRESP.EQ.'MAX').OR.(RDRESP.EQ.'max'))SREQ = 'MAX'
IF ((RDRESP.EQ.'DEG').OR.(RDRESP.EQ.'deg'))SREQ = 'DEG'
IF ((RDRESP.EQ.'AE1').OR.(RDRESP.EQ.'ael'))SREQ = 'AE1'
IF ((RDRESP.EQ.'AKO').OR.(RDRESP.EQ.'ako'))SREQ = 'AKO'
IF ((RDRESP.EQ.'ASW').OR.(RDRESP.EQ.'asw'))SREQ = 'ASW'
IF ((RDRESP.EQ.'CST').OR.(RDRESP.EQ.'cst'))SREQ = 'CST'
IF ((RDRESP.EQ.'CSB').OR.(RDRESP.EQ.'csb'))SREQ = 'CSB'
IF (SREQ.EQ.'XXX')GOTO 111
```

```

XCRC = SREQ
CRCXTN = XFIL // SREQ
CRCF = CRCXTN // XTEND
CRCFIL = FILH // CRCF

```

C Input Variable (RPL File: RPL---.OUT)

```

112 WRITE(6,12)
12  FORMAT(/10X,'ENTER RPL REQUIREMENT FILE:',/10X,
&'REQUIREMENTS: (MAX,DEG,AE1,AKO,ASW,CST,CSB)')
SREQ = 'XXX'
XFIL = 'RPL'
READ(*,*)RDRESP

IF ((RDRESP.EQ.'MAX').OR.(RDRESP.EQ.'max'))SREQ = 'MAX'
IF ((RDRESP.EQ.'DEG').OR.(RDRESP.EQ.'deg'))SREQ = 'DEG'
IF ((RDRESP.EQ.'AE1').OR.(RDRESP.EQ.'ae1'))SREQ = 'AE1'
IF ((RDRESP.EQ.'AKO').OR.(RDRESP.EQ.'ako'))SREQ = 'AKO'
IF ((RDRESP.EQ.'ASW').OR.(RDRESP.EQ.'asw'))SREQ = 'ASW'
IF ((RDRESP.EQ.'CST').OR.(RDRESP.EQ.'cst'))SREQ = 'CST'
IF ((RDRESP.EQ.'CSB').OR.(RDRESP.EQ.'csb'))SREQ = 'CSB'
IF (SREQ.EQ.'XXX')GOTO 112

XRPL = SREQ
RPLXTN = XFIL // SREQ
RPLF = RPLXTN // XTEND
RPLFIL = FILH // RPLF

```

C Checks to see if input files exist. If input files do not
C exist, a message is written to inform user that file was not
C found and terminates the program.

```

INQUIRE(FILE=CRCFIL,EXIST=THEIR)
IF (.NOT.THERE)THEN
  NCHK = 1
  WRITE(6,13)CRCF
13  FORMAT(///5X,'ERROR - CRC file: ',1X,A10,1X,'was not found.')
ENDIF

INQUIRE(FILE=RPLFIL,EXIST=THEIR)
IF (.NOT.THERE)THEN
  NCHK = 1
  WRITE(6,14)RPLF
14  FORMAT(///5X,'ERROR - RPL file: ',1X,A10,1X,'was not found.')
ENDIF

IF (NCHK.EQ.1)THEN
  WRITE(6,113)
113 FORMAT(////////5X,'***** TRANSPORTATION ANALYSIS TERMINATED *****')
&')
  GOTO 999
ENDIF

```

```

        IF (XRPL.NE.XCRC)THEN
            WRITE(6,114)
114    FORMAT(/////5X,'You have chosen two different requirement file
        &s.',/5X,'The requirements which will be used in the',/5X,
        &'output will be from the CRC file. The Theater',/5X,
        &'Percentage will also be calculated using the CRC file.',
        &////////)

        PAUSE

        WRITE(6,115)
115    FORMAT(//////////)

        ENDIF

```

C Checks to see if input and output files exist. If input files
C does not exist; an error message is written and the program is
C terminated. If output file exists; the old output file is deleted.

```

        INQUIRE(FILE='/home/warpam/iofiles/TRANS.TMP',EXIST=THERE)
        IF (THERE)THEN
            OPEN(52,FILE='/home/warpam/iofiles/TRANS.TMP',STATUS='OLD')
            CLOSE(52,STATUS='DELETE')
        ENDIF

        INQUIRE(FILE='/home/warpam/iofiles/TRANS.OUT',EXIST=THERE)
        IF (THERE)THEN
            OPEN(53,FILE='/home/warpam/iofiles/TRANS.OUT',STATUS='OLD')
            CLOSE(53,STATUS='DELETE')
        ENDIF

        OPEN(50,FILE=CRCFIL,STATUS='OLD')
15    READ(50,'(58(A1))',ERR=16,END=17)XCHR
        I = I + 1
        IF (I.LT.4)GOTO 15

        CATBRG = XCHR(3:7)
        SEX = XCHR(12:12)
        TYP = XCHR(17:19)
        TP = XCHR(23:24)
        REQT = XCHR(35:40)
        REQFIL = XCHR(44:49)

        IF ((TYP.NE.'MAX').AND.(TYP.NE.'DEG').AND.(TYP.NE.'AE1')
        &.AND.(TYP.NE.'AKO').AND.(TYP.NE.'ASW').AND.(TYP.NE.'CST')
        &.AND.(TYP.NE.'CSB'))GOTO 15

        J = J + 1
        VARHLD(J,1) = TP
        VARHLD(J,2) = CATBRG
        VARHLD(J,3) = SEX

```



```

    VARHLD(J,4) = TYP
    VARHLD(J,5) = REQ
    VARHLD(J,6) = REQFIL
    VARHLD(J,7) = '      0'

    GOTO 15

16  WRITE(6,*) 'ERROR READING OUTPUT FILE.'
17  CLOSE(50,STATUS='KEEP')

    MAXCNT = J
    I = 0

    OPEN(51,FILE=RPLFIL,STATUS='OLD')
18  READ(51,'(58(A1))',ERR=21,END=22)XCHR
    I = I + 1
    IF (I.LT.4)GOTO 18

    CATBRG = XCHR(3:7)
    SEX = XCHR(12:12)
    TYP = XCHR(17:19)
    TP = XCHR(23:24)
    REQ = XCHR(35:40)
    REQFIL = XCHR(44:49)

    IF ((TYP.NE.'MAX').AND.(TYP.NE.'DEG').AND.(TYP.NE.'AE1')
    &.AND.(TYP.NE.'AKO').AND.(TYP.NE.'ASW').AND.(TYP.NE.'CST')
    &.AND.(TYP.NE.'CSB'))GOTO 18

    ICHK = 0
    DO 19 LL = 1,MAXCNT
        IF ((TP.EQ.VARHLD(LL,1)).AND.(CATBRG.EQ.VARHLD(LL,2)))THEN
            VARHLD(LL,7) = REQFIL
            ICHK = 1
            GOTO 20
        ENDIF
19  CONTINUE
20  IF (ICHK.EQ.1)GOTO 18

    IF (ICHK.EQ.0)THEN
        MAXCNT = MAXCNT + 1
        VARHLD(MAXCNT,1) = TP
        VARHLD(MAXCNT,2) = CATBRG
        VARHLD(MAXCNT,3) = SEX
        VARHLD(MAXCNT,4) = TYP
        VARHLD(MAXCNT,5) = REQ
        VARHLD(MAXCNT,6) = '      0'
        VARHLD(MAXCNT,7) = REQFIL
        GOTO 18
    ENDIF

```

```

21 WRITE(6,*)'ERROR READING FILE'
22 CLOSE(51,STATUS='KEEP')

```

C Stores initial maxtrix to file: TRANS.TMP

```

OPEN(52,FILE='/home/warpam/iofiles/TRANS.TMP',STATUS='NEW')

DO 24 N = 1,MAXCNT
  WRITE(52,23)VARHLD(N,1),VARHLD(N,2),VARHLD(N,3),
&VARHLD(N,4),VARHLD(N,5),VARHLD(N,6),VARHLD(N,7)
23   FORMAT(2X,A2,3X,A5,3X,A1,3X,A3,3X,A6,3X,A6,3X,A6)
24   CONTINUE

CLOSE(52,STATUS='KEEP')

```

C Header Information

```

HEAD0='          CAT/BR          CRC          RPL          THE
&ATER  FLOW'

HEAD1=' TP  GRADE  S  THEATER  REQ  FLOW  FLOW  PER
&      DIF'

HEAD2='-----
&-----'
C      BXXBBBXXXXXBBBBBXXXXXBBBBBXXXXXBBBBBXXXXXBBBB
C      &XXXXXXBBBXXXXXX

```

C Read file: TRANS.TMP and calculate theater percentage and
C delta flow.

```

OPEN(53,FILE='/home/warpam/iofiles/TRANS.OUT',STATUS='NEW')
OPEN(52,FILE='/home/warpam/iofiles/TRANS.TMP',STATUS='OLD')

WRITE(53,25)HEAD0,HEAD1,HEAD2
25   FORMAT(1X,A70,/1X,A69,/1X,A72)

26   READ(52,27,ERR=29,END=30)TP,CATBRG,SEX,TYP,RQMT,CRCFLO,RPLFLO
27   FORMAT(2X,A2,3X,A5,3X,A1,3X,A3,3X,I6,3X,I6,3X,I6)

```

C Calculate Theater Percentage.

```

THTRP = REAL(RPLFLO) / REAL(RQM) * 100.0

```

C Calculate Delta Flow.

```

DELFLO = CRCFLO - RPLFLO

```

C Writes results to output file: TRANS.OUT

```

WRITE(53,28)TP,CATBRG,SEX,TYP,RQMT,CRCFLO,RPLFLO,THTRP,DELFLO

```

```
28  FORMAT(2X,A2,3X,A5,4X,A1,5X,A3,5X,I6,3X,I6,3X,I6,3X,F5.1,'% ',
      &3X,I6)
      GOTO 26
29  WRITE(6,*)'ERROR READING FILE:  TRANS.TMP'
30  CLOSE(53,STATUS='KEEP')
      CLOSE(52,STATUS='DELETE')
      WRITE(6,31)
31  FORMAT(/////5X,'***** TRANSPORTATION ANALYSIS COMPLETED *****')
999  STOP
      END
```

```
C  END TRANSPRT.FOR
```

SECTION 8 LOOK-UP TABLES

8.1 GENERAL

WARPAM was designed to enable the user and programmer to easily update the many look-up tables. This updating process is accomplished using Lotus or Symphony spreadsheets and then storing the actual table portion of the spreadsheet as ASCII files. This conversion is accomplished using the standard Lotus commands to store a spreadsheet to a file vice the printer. When entering the name the extension .tbl must be used so that the FORTRAN tables can recognize it. These tables are then stored in the IOFILES sub-directory with the other programs. Each table has a unique structure which can not be altered without also changing the FORTRAN programs which utilize it.

8.2 WARPAM BRANCH TABLE CODES:

The following is an overview of the coding system used in WARPAM to create the standard data format used throughout the system. The translation of specific data elements are accomplished by the look-up tables described in later sections.

1ST DIGIT: CATEGORY IDENTIFIER

O-OFFICER
W-WARRANT
E-ENLISTED

2ND & 3RD DIGITS: BRANCH IDENTIFIER

IN-INFANTRY	AR-ARMOR	FA-FIELD ARTILLERY
AD-AIR DEFENSE	AV-AVIATION	CE-ENGINEER
SC-SIGNAL/COMMO	MP-MIL POLICE	MC-MEDICAL
MI-MIL INTEL	CM-CHEMICAL	TC-TRANSPORTATION
OD-ORDINANCE	QM-QUARTERMASTER	SM-SIGNAL MAINT
MM-MECHANICAL MAINT		
CS-OTHER COMBAT SERVICE SUPPORT		

4TH & 5TH DIGIT: GRADE

FD-FIELD GRADE OFFICER (04 THRU 09)
CO-COMPANY GRADE OFFICER (01 THRU 03)
WW-ALL WARRANT OFFICER GRADES (W1 THRU W4)
59-ENLISTED NCO GRADES (E5 THRU E9)
14-ENLISTED SKILL LEVEL ONE GRADES (E1-E4)

EXAMPLE

OINCO	GRADE:	COMPANY	GRADE
	BRANCH:	INFANTRY	
	CATEGORY:	OFFICER	

8.3 BRANCH AGGREGATION TABLE

FILE NAME: BRANCH.WR1

Location: The Lotus/Symphony spreadsheet is stored on a standard PC. The table which is created must be stored on the sun workstation in the IOFILE sub-directory.

Use: The Branch Table converts officer and enlisted MOS to the standard branches used in WARPAM. This table is used by all the FORTRAN conversion programs in the preprocessor.

Structure: As MOS are created or redesignated, the user may desire to update the file by changes branch groupings or creating new branches. However, many changes may not be required as the file is designed with "wild card" designators, noted by the *. which denotes that the branch includes any MOS with the first two characters. In those cases when an MOS could not be placed in a general category, it is shown individually. When this occurs the general category MOS will read all MOS with the first two digits as shown, except the MOS listed individually (eg all enlisted MOS 76 are grouped in quartermaster (EQM), but the special case MOS 76J is grouped with medical corps (EMC)). The file is structured to have two digits, a single character (letter or *), one space and three characters for the branch code. To update the file, the desired changes should be entered manually and the file sorted on the MOS field to restore the numerical order of the file. When all changes are entered and the worksheet saved, the actual table used by the WARPAM models is extracted to file by the method described below.

Note: Wherever feasible MOS have been grouped into CMF equivalents. The table contains several MOS which have been entered twice with one entry containing the most common errors as with OOR vs OOR (ZERO ZERO ROMEO).

Conversion to table: To convert this worksheet to a table format for use in the preprocessor, the block containing the MOS and code only are saved to a print file with a .tbl extension using the normal Lotus structured commands. During the print command sequence, the user must select the Lotus command to save the table to "file" (disk) vice the normal printer command. When complete the file is loaded in the sub-directory containing the other Lotus tables.

BRANCH AGGREGATION TABLES

COMMISSIONED OFFICER

CATEGORY/BRANCH CODE
SPECIFIC Specialties

BRANCHES (Specialties)
INCLUDED IN CODED BRANCH

OIN 11A 11B 11C 11X 18A	BR 11 & 18
OAR 12A 12B 12C 12X	BR 12
OFA 13A 13B 13C 13D 13E	BR 13
OAD 14A 14B 14C 14D 14E	BR 14
OAV 15A 15B 15C 15D 15E	BR 15
OCE 21A 21B 21C 21D	BR 21
OSC 25A 25B 25C 25D 25E	BR 25
OMP 31A 31B 31C 31D	BR 31
OMI 35A 35B 35C 35D 35E 35F 35G	BR 35
OMC 60A 60B 60C 60D 60F 60G 60H 60J 60K 60L 60M 60N 60P 60Q 60R 60S 60T 60U 60V 60W 61A 61B 61C 61D 61E 61F 61G 61H 61J 61K 61L 61M 61N 61P 61Q 61R 61U 61W 61Z 62A 62B 63A 63B 63D 63E 63F 63H 63K 63M 63N 63P 63R 64A 64B 64C 64D 64E 64F 65A 65B 65C 66A 66B 66C 66D 66E 66F 66G 66H 66J 67A 67B 67C 67D 67E 67F 67G 67H 67J 67K 67L 68A 68B 68C 68D 68E 68F 68G 68H 68J 68K 68L 68M 68N 68P 68R 68S 68T 68U	BR 60 (60-68)
OCM 74A 74B 74C	BR 74
OTC 88A 88B 88C 88D 88E	BR 88
OOD 91A 91B 91C 91D 91E	BR 91
OQM 92A 92B 92D 92F 92G	BR 92

OCS ALL LESS THOSE LISTED ABOVE
 00B 01A 02A 03A 04A 38A 39A 39B 39C 41A 42A 42B 42C 42E 44A 45A 45B 46A 46B 47A
 47B 48A 48B 48C 48D 48E 48F 48G 48H 48I 48J 49A 49B 49C 49D 49E 49W 49X 50A 51A
 51B 51C 51D 52A 52B 53A 53B 53C 54A 55A 55B 56A 56D 97A 97B 97C

WARRANT OFFICER

CATEGORY/BRANCH CODE	BRANCHES (MOS)
SPECIFIC MOS	INCLUDED IN CODED BRANCH

WCB	ALL 130, 140, 150, 180
-----	------------------------

130A 130B 131A 131B 132A 140A 140B 140C 140D 140E 150A 151A 152B 152C 152D 152F
 152G 153A 153B 153C 153D 154A 154B 154C 155A 155D 155E 156A 180A

WCS	ALL 210, 213, 215, 250, 251, 252, 256 311, 600, 640, 670 210A
-----	---

213A 215A 215B 215C 215D 250A 250B 251A 252A 256A 311A 600A 640A 670A

WCC	ALL LESS WCB & WCS
-----	--------------------

350B 350D 350L 351B 351C 351E 352C 352D 352G 352H 352J 352K 353A 420A 420C 420D
 550A 880A 881A 910A 911A 912A 913A 914A 915A 915B 915C 915D 915E 920A 920B 921A
 922A

ENLISTED

CATEGORY/BRANCH CODE	CMF INCLUDED IN EACH
SPECIFIC MOS	CODED BRANCH

EIN	CMF 11 & 18
11B 11C 11H 11M 11Z 18B 18C 18D 18E 18F 18Z	

EAR	CMF 19
19D 19E 19K 19Z	

EFA	CMF 13
13B 13C 13E 13F 13M 13N 13P 13R 13T 13Z 15E 17B 21G 82C 93F	

EAD	CMF 16
16D 16E 16F 16G 16H 16J 16P 16R 16S 16T 16Z	

EAV CMF 67 & 93
66G 66H 66J 66N 66R 66S 66T 66U 66V 66X 66Y 67G 67H 67N 67R 67S 67T 67U 67V 67X
67Y 67Z 68B 68D 68F 68G 68H 68J 68K 68L 68N 68P 68Q 68R 93B 93C 93D 93P

ECE CMF 12, 51, 81
00B 12B 12C 12F 12Z 41B 51B 51G 51H 51K 51M 51R 51T 51Z 52E 52G 62E 62F 62G 62H
62J 62N 81B 81C 81Q 81Z 82B 82D 83E 83F

ESC CMF 31
31C 31D 31F 31G 31K 31L 31M 31N 31Q 31V 31W 31Y 31Z 36L 36M 72E 72G

EMP CMF 95
95B 95C 95D

EMI CMF 96 & 98
05D 05H 05K 96B 96D 96F 96H 96R 96Z 97B 97E 97G 97Z 98C 98G 98J 98Z

EMC CMF 91
01H 35G 35U 42C 42D 42E 71G 76J 91A 91B 91C 91D 91E 91F 91G 91H 91J 91L 91N 91P
91Q 91R 91S 91T 91U 91V 91W 91X 91Y 92B 92E 94F

ECM
54B

ETC CMF 88
88H 88K 88L 88M 88N 88P 88Q 88R 88S 88T 88U 88V 88W 88X 88Y 88Z

EOD CMF 63
41C 44B 44E 45B 45D 45E 45G 45K 45L 45N 45T 45Z 52C 52D 52F 52X 62B 63B 63D 63E
63G 63H 63J 63N 63S 63T 63W 63Y 63Z

EQM CMF 55, 76, 77, 94
43E 43M 55B 55D 55G 55R 55X 55Z 57E 57F 76C 76P 76V 76X 76Y 76Z 77F 77L 77W 94B

EMM CMF 24 & 27
24C 24G 24M 24N 24R 24S 24T 24U 25L 26H 21L 24H 24K 27B 27C 27D 27E 27F 27G 27H
27J 27K 27L 27M 27N 27V 27Z 46N

ESM CMF 29 & 33
29E 29F 29J 29M 29N 29P 29S 29T 29V 29W 29X 29Y 29Z 35H 39B 39C 39D 39E 39G 39L
39V 39W 39X 39Y 33M 33P 33Q 33R 33T 33V 33Z

ECS CMF 25, 46, 71, 74, 79 97 PLUS ANY MOS NOT LISTED ABOVE
00E 00R 00Z 02B 02C 02D 02E 02F 02G 02H 02J 02K 02L 02M 02N 02S 02T 02U 02Z 25P
25Q 25R 25S 25Z 46Q 46R 46Z 71C 71D 71E 71L 71M 73C 73D 73Z 74D 74F 74Z 75B 75C
75D 75E 75F 75Z 79D

BRANCH TABLE
(ACTUAL Lotus TABLE)

00* OCS	41* OCS	67* OMC	02* ECS	39* ESM	72* ESC
00* OCS	42* OCS	68* OMC	05* EMI	41B ECE	73* ECS
00* OCS	44* OCS	74* OCM	05* EMI	41* EOD	74* ECS
00* OCS	45* OCS	88* OTC	11* EIN	42* EMC	75* ECS
01* OCS	46* OCS	91* OOD	12* ECE	43* EQM	76J EMC
01* OCS	47* OCS	92* OQM	13* EFA	44* EOD	76* EQM
02* OCS	48* OCS	95* OTC	15* EFA	45* EOD	77* EQM
02* OCS	49* OCS	97* OCS	16* EAD	46N EMM	79D ECS
03* OCS	50* OCS	1** WCB	17* EFA	46* ECS	81* ECE
03* OCS	50* OCS	2** WCS	18* EIN	51* ECE	82C EFA
04* OCS	51* OCS	3** WCS	19* EAR	52E ECE	82* ECE
04* OCS	52* OCS	4** WCC	21G EFA	52G ECE	88* ETC
11* OIN	53* OCS	5** WCC	21* EMM	52* EOD	91* EMC
12* OAR	54* OCS	6** WCS	24* EMM	54* ECM	92* EMC
13* OFA	55* OCS	7** WCC	25L EMM	55* EQM	93F EFA
14* OAD	56* OCS	8** WCC	25* ECS	57* EQM	93* EAV
15* OAV	60* OMC	9** WCC	26* EMM	62B EOD	94F EMC
18* OIN	60* OMC	00* ECS	27* EMM	62* ECE	94* EQM
21* OCE	61* OMC	00* ECS	29* ESC	63* EOD	95* EMP
25* OSC	62* OMC	00* ECS	31* ESC	66* EAV	96* EMI
31* OMP	63* OMC	00* ECS	33* ESM	67* EAV	97* EMI
35* OMI	64* OMC	01* EMC	35H ESM	68* EAV	98* EMI
38* OCS	65* OMC	01* EMC	35* EMC	71G EMC	
39* OCS	66* OMC	02* ECS	36* ESC	71* ECS	

NOTE: * DESIGNATES THAT ALL SPECIALTIES/MOS WITH THE FIRST TWO DIGITS ARE GROUPED IN THIS BRANCH, EXCEPT WHERE INDIVIDUAL MOS ARE LISTED SEPARATELY.

8.4 WARPAM BRANCH PRIORITY TABLE

FILENAME: WARPRI.WR1

Location: THE Lotus/SYMPHONY file is stored on a standard PC. The table created from this worksheet is stored in the Sun workstation "IOFILE" sub-directory.

Use: Used to construct the officer and enlisted branch priority table for use by the REQAST.FOR program. This look-up table supplies the program with the priority of each of the branch/grade combinations found in the current table. To update the file the priorities are manually changed and the file is then resorted in ascending order using the Lotus sort command.

Structure: The file consists of a two digit number, a space and the five letter code for each branch/grade combination.

Conversion to table: The block consisting of the priority and code letter only is copied to a file (not printer) with the file name "WARPRI.TBL using the standard Lotus print commands.

WARPRI TABLE
(ACTUAL Lotus TABLE)

01 OARFD	35 OMCCO
02 OAVFD	36 OMICO
03 OINFD	37 OMPCO
04 OFAFD	38 OCMCO
05 OADFD	39 OTCCO
06 OARCO	40 EMI59
07 OAVCO	41 EMC59
08 OINCO	42 EMP59
09 OFACO	43 ECM59
10 OADCO	44 ETC59
11 OCEFD	45 EOD59
12 OSCFD	46 EMI14
13 OCECO	47 EMC14
14 OSCCO	48 EMP14
15 WCBWW	49 ECM14
16 EAR59	50 ETC14
17 EAV59	51 EOD14
18 EIN59	52 OODFD
19 EFA59	53 OQMFD
20 EAD59	54 OCSFD
21 EAR14	55 WCSWW
22 EAV14	56 WCCWW
23 EIN14	57 OODCO
24 EFA14	58 OQMCO
25 EAD14	59 OCSCO
26 ECE59	60 EQM59
27 ESC59	61 EMM59
28 ECE14	62 ESM59
29 ESC14	63 ECS59
30 OMCFD	64 EQM14
31 OMIFD	65 EMM14
32 OMPFD	66 ESM14
33 OCMFD	67 ECS14
34 OTCFD	

8.5 THEATER/REPLACEMENT TYPE TABLE

FILENAME: THTRTYPE.WR1

LOCATION: The worksheet is stored on a standard PC. The table created from this worksheet is stored on the Sun workstation in the "IOFILE" sub-directory.

Use: The table is used in the requirements/assets table construction program. The table supplies the code numbers corresponding to the letter code for each type of requirement (by theater) and asset.

Structure: The file may be updated by manually changing either the coded number and corresponding letter code or by adding a new line. The spacing of the file containing the coded numbers and letters may not be changed.

Conversion to table: The block consisting of the coded number and letters is saved to a file named THTRTYPE.TBL using the standard Lotus print commands. Old files may be saved by simply renaming these to a different filename in DOS. Only one file may be present on the Sun workstation with the designated name.

	0001	MAX	MAXIMUM FLOW
	0010	DEG	DEFENSE GUIDANCE
	0021	AE1	AUTOREP--EUROPE
	0022	AKO	AUTOREP--KOREA
	0023	ASW	AUTOREP--SW ASIA
	0031	CST	CSMII--TOTAL
	0032	CSB	CSMII--BATTLE ONLY
	0100	TRD	THEATER RETURN TO DUTY
	0200	THS	ACTIVE THS
*	0300	SEL	SELECT RESERVES
	0400	IRR	INITIAL READY RESERVE
	0500	STY	STANDBY BY & IMA
	0600	RET	RETIREEES
	0700	TRN	TRAINING BASE

* - Not used in current version. The MOBBMAN data base cannot distinguish individual select reserves from units.

8.6 AUTOREP TIME PERIOD CONVERSION TABLE

FILE NAME: TP.WR1

LOCATION: THE Lotus/SYMPHONY worksheet is stored on a standard PC. The table which is created must be stored on the sun workstation in the IOFILE sub-directory.

Use: Used to construct the time periods conversion table for use in the AUTOREP FORTRAN program. This table converts the coded time periods in the input file to 10 day standard format time periods. The TP.Tbl file must be present to run the AUTOREP.FOR program.

Structure: This file should not have to be updated unless there is a change in the AUTOREP input file structure. The table is designed to start at the left MOS margin and consist of two letters, a space and two digits.

Warning: This table structure must not be altered unless the FORTRAN programs which utilizes it is also altered.

Conversion to table: The block consisting of coded letters and conversion numbers below without any header information is saved as a print file with a .TBL extension. To accomplish this in Lotus, print this block to file vice printer in the Lotus structured commands. The file name must be TP.Tbl. Several files may be created, but only one can be present with this specific name in the Sun sub-directory.

CA 01	AM 08
AA 02	AN 08
AB 02	AO 09
AC 03	AP 09
AD 03	AQ 10
AE 04	AR 10
AF 04	AS 11
AG 05	AT 11
AH 05	AU 12
AI 06	AV 12
AJ 06	AW 13
AK 07	AX 13
AL 07	

8.7 OFFICER RECLASSIFICATION PERCENTAGE TABLE

FILE: ORCLSPER.WR1

LOCATION: THE Lotus/SYMPHONY worksheet is stored on a standard PC. The table which is created must be stored on the sun workstation in the IOFILE sub-directory.

USE: Used to construct the reclassification table for the RECLAS module. This table prescribes what percentage of the old branch is reclassified into the new branches.

Structure: This file may be easily updated by manually inserting new percentages into each line. However, the total of the line must be 100%. This may be accomplished on this worksheet by having the CS column equal the difference between 100% and the sum of the other columns. If another method is used than a check column with the sum of the percentages in the line should be used to verify the sum.

Note: The column spacing may not be changed.

Conversion to table: The portion of the file containing the actual branch codes and percentages should be copied without headers or other the standard Lotus print commands.

OFFICER RECLASSIFICATION PERCENTAGES

	NEW MOS																	
OLD MOS	IN	AR	FA	AD	AV	CE	SC	MP	MI	MC	CM	TC	OD	QM	CS	CB	CS	CC
IN	20								2					5	74			
AR		20				1			1			3	3	5	67			
FA			20						2			2	2	3	72			
AD				20		1			1			2	2	3	71			
AV					20	1	2		7			2	2	4	62			
CE						30					1	2	2	3	63			
SC							30		1					1	68			
MP								40	4		1	1	1	4	51			
MI							1	1	40						58			
MC										100								
CM											50	5	5	1	39			
TC											1	50		6	43			
OD											1		70	6	23			
QM												6	6	70	18			
CS															100			
CB																20	40	40
CS																	80	20
CC																		100

8.8 ENLISTED RECLASSIFICATION PERCENTAGE TABLE

FILE: ERCLSPER.WR1

LOCATION: THE Lotus/SYMPHONY worksheet is stored on a standard PC. The table which is created must be stored on the sun workstation in the IOFILE sub-directory.

USE: Used to construct the Reclassification table for the RECLAS module. This table prescribes what percentage of the old branch is reclassified into the new branches.

Structure: This file may be easily updated by manually inserting new percentages into each line. However, the total of the line must be 100%. This may be accomplished on this worksheet by having the CS column equal the difference between 100% and the sum of the other columns. If another method is used, then a check column with the sum of the percentages in the line should be used to verify the sum.

Note: The column spacing may not be changed.

Conversion to table: The portion of the file containing the actual branch codes and percentages should be copied without headers or other the standard Lotus print commands.

ENLISTED RECLASSIFICATION PERCENTAGES

	AR	AV	IN	FA	AD	CE	CM	MI	MP	SC	MC	TC	MM	OD	QM	SM	CS
AR	20								10	10		10	10	10	10		20
AV		20							10	10		10	20	20	10		
IN			20							10		20	10	10	10		20
FA				20					10	10		10	10	20	10		10
AD					20				10	10		10	10	10	10	10	10
CE						20			10	10		20	10	20	10		
CM							30	10		10			10	20	10		10
MI								30	10	20			10		10	10	10
MP									30	10		20	10	10	10		10
SC										30		10	10	10	10	20	10
MC											100						
TC												40	20	20	20		
MM													50	20	20		10
OD														60	20		20
QM															90		10
SM																100	
CS																	100

8.9 RECLASSIFICATION DELAY TABLE

FILE NAME: RCLSDLY.WR1

Location: The worksheet is stored on a standard PC. The table created from this worksheet is stored on the Sun workstation in the "IOFILE" sub-directory.

Use: Used to construct the reclassification delay table (RCLSDLY.TBL) used in the reclassification module. This table distributes the reclassified personnel after they have been given a new branch into one of six time periods after the current time period. The percentage in each time period is based on information provided by Soldier Support Center. To update the table these percentages are changed manually by the user and re-saved with the same name. Although all time periods contain the same percentages in this developmental model, these may be changed to a different percentage for each time period.

Structure: Although the file may be updated, the structure of the file as the width of rows, may not be changed. A change in structure will cause the FORTRAN program to read the file incorrectly.

Conversion to table: The block consisting of the data only without any header information is copied to "file" vice printer using the standard Lotus print commands and given a extension of.TBL. Previous tables may be saved by simply renaming the old file with standard DOS commands.

TIME PERIODS TO DELAY RETURN OF TRD AND PERCENTAGE INTO EACH TIME PERIOD

TP	1	2	3	4	5	6
01	0.12	0.27	0.31	0.23	0.05	0.02
02	0.12	0.27	0.31	0.23	0.05	0.02
03	0.12	0.27	0.31	0.23	0.05	0.02
04	0.12	0.27	0.31	0.23	0.05	0.02
05	0.12	0.27	0.31	0.23	0.05	0.02
06	0.12	0.27	0.31	0.23	0.05	0.02
07	0.12	0.27	0.31	0.23	0.05	0.02
08	0.12	0.27	0.31	0.23	0.05	0.02
09	0.12	0.27	0.31	0.23	0.05	0.02
10	0.12	0.27	0.31	0.23	0.05	0.02
11	0.12	0.27	0.31	0.23	0.05	0.02
12	0.12	0.27	0.31	0.23	0.05	0.02
13	0.12	0.27	0.31	0.23	0.05	0.02
14	0.12	0.27	0.31	0.23	0.05	0.02
15	0.12	0.27	0.31	0.23	0.05	0.02
16	0.12	0.27	0.31	0.23	0.05	0.02
17	0.12	0.27	0.31	0.23	0.05	0.02
18	0.12	0.27	0.31	0.23	0.05	0.02

SECTION 9 REPORT GENERATOR CODE (DBASE III)

9.1 GENERAL

The WARPAM Report Generator is designed to allow the user easy access to the WARPAM output files and provide a flexible system to develop both standard format and specially designed reports. The preprocessor and models of WARPAM generate output files in the standard UNIX format which are automatically translated to a DOS file when the reports are copied to a PC via the TRAC-FBHN network. The purpose of the Report Generator programs is to translate these DOS (ASCII) files to DBASE III Plus format. This is accomplished in DBASE III Plus from the dot prompt command line. The user or programmer need only enter "DO filename.PRG" to execute the individual conversion routines. Once, this is accomplished the user may proceed to the assist system and produce reports using the new file which will be named for the program run with the standard DBASE III extension for a data base--.DBF. To modify the program, the programmer may use the DBASE III modify command processor or any editor as Sidekick to modify the programs. As these conversion programs are designed to read the WARPAM output file formats, any change to the FORTRAN programs which results in a change in the output file must be accompanied by a change in the appropriate conversion program. The User's Manual should be consulted for specific steps to initiate each program.

9.2 REQUIREMENT/ASSET REPORT

9.2.1 CONVERSION PROGRAM

**** REQAST.PRG ****

SET ECHO OFF
SET TALK OFF
CLEAR

ERASE REQAST.DBF
COPY FILE REQBLNK.DBF TO REQAST.DBF
USE REQAST
APPEND FROM REQAST.OUT TYPE DELIMITED WITH BLANK
GO TOP
DELETE NEXT 3
PACK

SET TALK ON
SET ECHO ON

*******SUBROUTINES*******

**** REQAST2.PRG ****

SET ECHO OFF
SET TALK OFF

USE REQAST

SET ALTERNATE TO K:\KAK\REQAST.TXT

SET ALTERNATE ON

??	CAT/BR	REQ/	TIME PER/	REQ'T/	"
? "	GRADE	S	TYPE	PRIORITY	ASSETS "
? "	-----"				

DO WHILE .NOT. EOF()

? " ",CAT_BR_GRD," ",SEX," ",REQ_TYPE," ",TP_PRIORITY," ",REQT_ASSET

SKIP

ENDDO

SET ALTERNATE TO
CLOSE ALTERNATE

SET TALK ON
SET ECHO ON

9.2.2 OUTPUT FORMAT

SEE ANNEX C, PAGE C-1.

9.3 RECLASSIFICATION (MODIFIED REQUIREMENT ASSETS) FILE CONVERSION PROGRAMS

9.3.1 CONVERSION PROGRAM

**** MODRQAST.PRG ****

SET ECHO OFF
SET TALK OFF
CLEAR

ERASE MODRQAST.DBF

COPY FILE MODRBLNK.DBF TO MODRQAST.DBF

USE MODRQAST

APPEND FROM MODRQAST.OUT TYPE DELIMITED WITH BLANK

GO TOP

DELETE NEXT 3

PACK

SET TALK ON
SET ECHO ON

9.3.2 OUTPUT FILE

SEE ANNEX C, PAGE C-2.

9.4 CRC MODEL REPORT (INDIVIDUAL PROGRAMS FOR EACH REQUIREMENT FILE)

9.4.1 CONVERSION PROGRAMS

BASE PROGRAM

**** CRC.PRG ****

SET ECHO OFF
SET TALK OFF
CLEAR

ERASE CRC.DBF
COPY FILE CRCBLNK.DBF TO CRC.DBF
USE CRC
APPEND FROM CRC.OUT TYPE DELIMITED WITH BLANK
GO TOP
DELETE NEXT 3
PACK
GO TOP

DO WHILE .NOT. EOF()

IF TEMP <> 0

IF VAL(SUBSTR(TP_PRIORIT,7,3)) < 100

REPLACE REQT_FILLD WITH TEMP

ELSE

REPLACE ASSET_USED WITH TEMP

ENDIF

ENDIF

SKIP

ENDDO

SET TALK ON
SET ECHO ON

**** CRCMAX.PRG ****

SET ECHO OFF
SET TALK OFF
CLEAR

ERASE CRCMAX.DBF
COPY FILE CRCBLNK.DBF TO CRCMAX.DBF
USE CRCMAX
APPEND FROM CRCMAX.OUT TYPE DELIMITED WITH BLANK
GO TOP
DELETE NEXT 3
PACK
GO TOP

DO WHILE .NOT. EOF()

IF TEMP <> 0

IF VAL(SUBSTR(TP_PRIORIT,7,3)) < 100

REPLACE REQ_T_FILLD WITH TEMP

ELSE

REPLACE ASSET_USED WITH TEMP

ENDIF

ENDIF

SKIP

ENDDO

SET TALK ON
SET ECHO ON

**** CRCDEG.PRG ****

SET ECHO OFF
SET TALK OFF
CLEAR

ERASE CRCDEG.DBF
COPY FILE CRCBLNK.DBF TO CRCDEG.DBF
USE CRCDEG
APPEND FROM CRCDEG.OUT TYPE DELIMITED WITH BLANK
GO TOP
DELETE NEXT 3
PACK
GO TOP

```

DO WHILE .NOT. EOF()
  IF TEMP <> 0
    IF VAL(SUBSTR(TP_PRIORIT,7,3)) < 100
      REPLACE REQT_FILLD WITH TEMP
    ELSE
      REPLACE ASSET_USED WITH TEMP
    ENDIF
  ENDIF
  SKIP
ENDDO

SET TALK ON
SET ECHO ON

** CRCAE1.PRG **

SET ECHO OFF
SET TALK OFF
CLEAR

ERASE CRCAE1.DBF
COPY FILE CRCBLNK.DBF TO CRCAE1.DBF
USE CRCAE1
APPEND FROM CRCAE1.OUT TYPE DELIMITED WITH BLANK
GO TOP
DELETE NEXT 3
PACK
GO TOP

DO WHILE .NOT. EOF()
  IF TEMP <> 0
    IF VAL(SUBSTR(TP_PRIORIT,7,3)) < 100
      REPLACE REQT_FILLD WITH TEMP
    ELSE
      REPLACE ASSET_USED WITH TEMP
    ENDIF
  
```

ENDIF

SKIP

ENDDO

SET TALK ON

SET ECHO ON

**** CRCAKO.PRG ****

SET ECHO OFF

SET TALK OFF

CLEAR

ERASE CRCAKO.DBF

COPY FILE CRCBLNK.DBF TO CRCAKO.DBF

USE CRCAKO

APPEND FROM CRCAKO.OUT TYPE DELIMITED WITH BLANK

GO TOP

DELETE NEXT 3

PACK

GO TOP

DO WHILE .NOT. EOF()

IF TEMP <> 0

IF VAL(SUBSTR(TP_PRIORIT,7,3)) < 100

REPLACE REQ_T_FILLD WITH TEMP

ELSE

REPLACE ASSET_USED WITH TEMP

ENDIF

ENDIF

SKIP

ENDDO

SET TALK ON

SET ECHO ON

**** CRCCST.PRG ****

SET ECHO OFF
SET TALK OFF
CLEAR

ERASE CRCCST.DBF
COPY FILE CRCBLNK.DBF TO CRCCST.DBF
USE CRCCST
APPEND FROM CRCCST.OUT TYPE DELIMITED WITH BLANK
GO TOP
DELETE NEXT 3
PACK
GO TOP

DO WHILE .NOT. EOF()

IF TEMP <> 0

IF VAL(SUBSTR(TP_PRIORIT,7,3)) < 100

REPLACE REQ_T_FILLD WITH TEMP

ELSE

REPLACE ASSET_USED WITH TEMP

ENDIF

ENDIF

SKIP

ENDDO

SET TALK ON
SET ECHO ON

**** CRCCSB.PRG ****

SET ECHO OFF
SET TALK OFF
CLEAR

ERASE CRCCSB.DBF
COPY FILE CRCBLNK.DBF TO CRCCSB.DBF
USE CRCCSB
APPEND FROM CRCCSB.OUT TYPE DELIMITED WITH BLANK
GO TOP
DELETE NEXT 3
PACK

```

GO TOP
DO WHILE .NOT. EOF()
  IF TEMP <> 0
    IF VAL(SUBSTR(TP_PRIORIT,7,3)) < 100
      REPLACE REQT_FILLD WITH TEMP
    ELSE
      REPLACE ASSET_USED WITH TEMP
    ENDIF
  ENDIF
SKIP
ENDDO

```

```

SET TALK ON
SET ECHO ON

```

9.4.2 OUTPUT REPORT

SEE ANNEX C, PAGE C-3.

9.5 REPLACEMENT CO MODEL REPORT FILE CONVERSION PROGRAMS

9.5.1 CONVERSION PROGRAMS (INDIVIDUAL PROGRAM FOR EACH REQUIEMENT FILE)

**** RPLMAX.PRG ****

```

SET ECHO OFF
SET TALK OFF
CLEAR

```

```

ERASE RPLMAX.DBF
COPY FILE RPLBLNK.DBF TO RPLMAX.DBF
USE RPLMAX
APPEND FROM RPLMAX.OUT TYPE DELIMITED WITH BLANK
GO TOP
DELETE NEXT 3
PACK
GO TOP

```

```

DO WHILE .NOT. EOF()
  IF TEMP <> 0

```



```

    IF VAL(SUBSTR(TP_Priorit,7,3)) < 100
        REPLACE REQT_FILLD WITH TEMP
    ELSE
        REPLACE ASSET_USED WITH TEMP
    ENDIF
ENDIF
SKIP
ENDDO

SET TALK ON
SET ECHO ON

** RPLDEG.PRG **

SET ECHO OFF
SET TALK OFF
CLEAR

ERASE RPLDEG.DBF
COPY FILE RPLBLNK.DBF TO RPLDEG.DBF
USE RPLDEG
APPEND FROM RPLDEG.OUT TYPE DELIMITED WITH BLANK
GO TOP
DELETE NEXT 3
PACK
GO TOP

DO WHILE .NOT. EOF()

    IF TEMP <> 0
        IF VAL(SUBSTR(TP_Priorit,7,3)) < 100
            REPLACE REQT_FILLD WITH TEMP
        ELSE
            REPLACE ASSET_USED WITH TEMP
        ENDIF
    ENDIF
ENDIF
SKIP

```

ENDDO

SET TALK ON
SET ECHO ON

**** RPLAE1.PRG ****

SET ECHO OFF
SET TALK OFF
CLEAR

ERASE RPLAE1.DBF
COPY FILE RPLBLNK.DBF TO RPLAE1.DBF
USE RPLAE1
APPEND FROM RPLAE1.OUT TYPE DELIMITED WITH BLANK
GO TOP
DELETE NEXT 3
PACK
GO TOP

DO WHILE .NOT. EOF()

IF TEMP <> 0

IF VAL(SUBSTR(TP_PRIORIT,7,3)) < 100

REPLACE REQT_FILLD WITH TEMP

ELSE

REPLACE ASSET_USED WITH TEMP

ENDIF

ENDIF

SKIP

ENDDO

SET TALK ON
SET ECHO ON

**** RPLAKO.PRG ****

SET ECHO OFF
SET TALK OFF
CLEAR

ERASE RPLAKO.DBF

COPY FILE RPLBLNK.DBF TO RPLAKO.DBF
USE RPLAKO
APPEND FROM RPLAKO.OUT TYPE DELIMITED WITH BLANK
GO TOP
DELETE NEXT 3
PACK
GO TOP

DO WHILE .NOT. EOF()

IF TEMP <> 0

IF VAL(SUBSTR(TP_PRIORIT,7,3)) < 100

REPLACE REQT_FILLD WITH TEMP

ELSE

REPLACE ASSET_USED WITH TEMP

ENDIF

ENDIF

SKIP

ENDDO

SET TALK ON
SET ECHO ON

**** RPLCST.PRG ****

SET ECHO OFF
SET TALK OFF
CLEAR

ERASE RPLCST.DBF
COPY FILE RPLBLNK.DBF TO RPLCST.DBF
USE RPLCST
APPEND FROM RPLCST.OUT TYPE DELIMITED WITH BLANK
GO TOP
DELETE NEXT 3
PACK
GO TOP

DO WHILE .NOT. EOF()

IF TEMP <> 0

IF VAL(SUBSTR(TP_PRIORIT,7,3)) < 100

```

        REPLACE REQT_FILLD WITH TEMP
    ELSE
        REPLACE ASSET_USED WITH TEMP
    ENDIF
ENDIF
SKIP
ENDDO

SET TALK ON
SET ECHO ON

** RPLCSB.PRG **

SET ECHO OFF
SET TALK OFF
CLEAR

ERASE RPLCSB.DBF
COPY FILE RPLBLNK.DBF TO RPLCSB.DBF
USE RPLCSB
APPEND FROM RPLCSB.OUT TYPE DELIMITED WITH BLANK
GO TOP
DELETE NEXT 3
PACK
GO TOP

DO WHILE .NOT. EOF()

    IF TEMP <> 0

        IF VAL(SUBSTR(TP_PRIORIT,7,3)) < 100
            REPLACE REQT_FILLD WITH TEMP
        ELSE
            REPLACE ASSET_USED WITH TEMP
        ENDIF
    ENDIF
SKIP
ENDDO

```

SET TALK ON
SET ECHO ON

9.5.2 OUTPUT FILE

SEE ANNEX C, PAGE C-3.

9.6 TRANSPORTATION MODEL REPORT CONVERSION PROGRAMS

9.6.1 CONVERSION PROGRAM

** TRANS.PRG **

SET ECHO OFF
SET TALK OFF
CLEAR

ERASE TRANS.DBF
COPY FILE TRNSBLNK.DBF TO TRANS.DBF
USE TRANS
APPEND FROM TRANS.OUT TYPE DELIMITED WITH BLANK
GO TOP
DELETE NEXT 3
PACK

SET TALK ON
SET ECHO ON

9.6.2 OUTPUT REPORT

SEE ANNEX C, PAGE C-4

ANNEX A

TERMS & ABBREVIATIONS

- ASSET:** Personnel inventory used to satisfy requirements. There are seven classes of assets: TRD-Theater Return-To-Duty, THS-active duty transients, holdees, students and hospital, SEL-Select Reserve, IRR-Initial Ready Reserve, STY-Stand By and IMA, RET-retirees, TRN-skill level one trainees.
- AUTOREP:** US ARMY PERSCOM shelf requestion system.
- Branch:** Branch represents the specialties/MOS and grade combinations which have been grouped together in the preprocessor. These branches are then prioritized in the Branch Look-Up Table and given a priority number. The initial version of WARPAM has 67 branch/grade combinations.
- CSM II:** Soldier Support Center casualty stratification model.
- MOBARPRINT:** HQDA, ODCSPER system for the projection of skill level one training base output. MOBTNGBS is used interchangeable in WARPAM.
- MOBMAN:** US ARMY PERSCOM model to project defense guidance level requirements and personnel assets.
- Return-to-Duty Rate:** This is the percentage of casualties which the user desires to return to duty within the theater. The model will accept either a rate (decimal) or percentage (whole number) ranging from .1% (.001) to 99.99% (.9999). Based on 1989 CAA estimates the recommended rate for current policy is 20%.
- Requirements:** Personnel requirements in a theater caused by either a shortage of personnel or by casualties. Requirements are derived from other military model outputs and are found in the requirement/assets file.
- Requirements/Assets Generator:** This module merges the files derived from other military models into a single file, assigns branch priorities, assigns a unique code number, and sorts the file by code number. The output of this module is the REQAST.TBL.
- Time Periods:** A time period is 10 days.

ANNEX B: SAMPLE FILE/OUTPUT FORMATS

B.1 INPUT FILES

B.1.1 AUTOREP INPUT FILE

AAW100AX NB0001A2
ABW100AX NB0001A2
ACW100AX NB0002A2
ADW100AX NB0003A2
AEW100AX NB0004A2
AFW100AX NB0006A2
AGW100AX NB0009A2
AHW100AX NB0009A2
AIW100AX NB0009A2
AJW100AX NB0009A2
AKW100AX NB0009A2
ALW100AX NB0009A2
AMW100AX NB0009A2
ANW100AX NB0009A2
AOW100AX NB0009A2
APW100AX NB0005A2
AQW100AX NB0005A2
ARW100AX NB0005A2
ASW100AX NB0005A2
ATW100AX NB0005A2
AUW100AX NB0005A2
AVW100AX NB0005A2
AWW100AX NB0005A2
AXW100AX NB0005A2
AGW100BX NB0001A2
AHW100BX NB0001A2
AIW100BX NB0001A2
AJW100BX NB0001A2
AKW100BX NB0001A2
ALW100BX NB0001A2
AMW100BX NB0001A2
ANW100BX NB0001A2
AOW100BX NB0001A2
AEW100CX NB0001A2
AFW100CX NB0001A2
AGW100CX NB0001A2
AHW100CX NB0001A2
AIW100CX NB0001A2
AJW100CX NB0001A2
AKW100CX NB0001A2
ALW100CX NB0002A2
AMW100CX NB0002A2

B.1.2 NOBMAN INPUT FILE

MOBMAN 1322 REPORT												
REQUIREMENTS AND ASSETS FOR FY 91 AT MOS LEVEL												
	PEACE	M-DAY	M-10	M-20	M-30	M-40	M-50	M-60	M-90	M-120	M-150	M-180
RETIREES	25	25	25	25	25	25	25	25	25	25	25	25
TRAINEES	0	0	0	0	0	0	0	0	0	0	0	0
SUPPLY TOTAL	908	908	908	908	912	915	918	921	930	931	931	931
CUM BALANCE	108	98	95	97	108	108	110	113	123	124	123	122
-MOS 02C E												
WAR REQUIRED	254	259	259	259	259	259	259	259	259	259	259	259
CASUALTIES	0	0	0	0	0	0	0	0	0	0	0	0
TNS	5	5	5	5	2	2	2	2	2	2	2	2
REQ'D TOTAL	259	264	264	264	261	261	261	261	261	261	261	261
ACTIVE	105	105	105	105	105	105	105	105	105	105	105	105
SEL RESERVE	136	136	136	136	136	136	136	136	136	136	136	136
IMA	0	0	0	0	0	0	0	0	0	0	0	0
IRR	16	16	16	16	16	16	16	16	16	16	16	16
STANDBY	0	0	0	0	0	0	0	0	0	0	0	0
RETIRES	6	6	6	6	6	6	6	6	6	6	6	6
TRAINEES	0	0	1	2	3	4	5	6	8	8	8	8
SUPPLY TOTAL	268	268	264	265	266	267	268	269	271	271	271	271
CUM BALANCE	4	-1	0	1	5	6	7	8	10	10	10	10
IMOSCDRX												
-MOS 02D E												
WAR REQUIRED	402	403	403	403	403	403	403	403	403	403	403	403
CASUALTIES	0	0	0	0	0	0	0	0	0	0	0	0
TNS	7	7	7	7	6	6	6	3	3	3	3	3
REQ'D TOTAL	409	410	410	410	409	409	409	406	406	406	406	406
ACTIVE	136	136	136	136	136	136	136	136	136	136	136	136
SEL RESERVE	203	203	203	203	203	203	203	203	203	203	203	203
IMA	0	0	0	0	0	0	0	0	0	0	0	0
IRR	28	28	28	28	28	28	28	28	28	28	28	28
STANDBY	0	0	0	0	0	0	0	0	0	0	0	0
RETIRES	9	9	9	9	9	9	9	9	9	9	9	9
TRAINEES	0	0	1	2	3	4	5	6	7	14	14	14
SUPPLY TOTAL	376	376	377	378	379	380	381	382	385	390	390	390
CUM BALANCE	-33	-34	-33	-32	-30	-29	-28	-24	-21	-16	-16	-16
-MOS 02E E												
WAR REQUIRED	518	518	518	518	518	518	518	518	518	518	518	518
CASUALTIES	0	0	0	0	0	1	1	1	1	1	1	1
TNS	10	10	10	10	7	7	7	4	3	3	3	3
REQ'D TOTAL	528	528	528	528	525	526	526	523	522	522	522	522
ACTIVE	212	212	212	212	212	212	212	212	212	212	212	212
SEL RESERVE	282	282	282	282	282	282	282	282	282	282	282	282
IMA	2	2	2	2	2	2	2	2	2	2	2	2
IRR	44	44	44	44	44	44	44	44	44	44	44	44
STANDBY	0	0	0	0	0	0	0	0	0	0	0	0
RETIRES	15	15	15	15	15	15	15	15	15	15	15	15
TRAINEES	0	0	1	2	3	4	5	6	9	13	13	13
SUPPLY TOTAL	555	555	556	557	558	559	560	561	564	568	568	568
CUM BALANCE	32	27	28	29	33	38	34	38	42	46	46	46
-MOS 02F E												
WAR REQUIRED	376	377	377	377	377	377	377	377	377	377	377	377
CASUALTIES	0	0	0	0	0	0	0	0	0	0	0	0
TNS	6	6	6	6	6	6	6	3	3	3	3	3
REQ'D TOTAL	382	383	383	383	383	383	383	380	380	380	380	380
ACTIVE	136	136	136	136	136	136	136	136	136	136	136	136
SEL RESERVE	208	208	208	208	208	208	208	208	208	208	208	208
IMA	1	1	1	1	1	1	1	1	1	1	1	1
IRR	34	34	34	34	34	34	34	34	34	34	34	34
STANDBY	0	0	0	0	0	0	0	0	0	0	0	0

B.1.3 CSM II INPUT FILE

09 0 60T	*	*	0	0
09 0 60V	*	*	0	0
09 0 60W	*	*	0	0
09 0 61F	*	*	1	0
09 0 61H	*	*	1	0
09 0 61J	*	*	2	0
09 0 61K	*	*	0	0
09 0 61M	*	*	1	0
09 0 61N	*	*	0	0
09 0 61S	*	*	0	0
09 0 61U	*	*	0	0
09 0 61Z	*	*	0	0
09 0 62A	*	*	1	0
09 0 63A	*	*	1	0
09 0 63B	*	*	0	0
09 0 63G	*	*	0	0
09 0 63H	*	*	0	0
09 0 63N	*	*	0	0
09 0 63R	*	*	0	0
09 0 64A	*	*	0	0
09 0 64B	*	*	0	0
09 0 64E	*	*	0	0
09 0 65A	*	*	0	0
09 0 65B	*	*	0	0
09 0 65C	*	*	1	0
09 0 66A	*	*	0	0
09 0 66C	*	*	0	0
09 0 66E	*	*	2	0
09 0 66F	*	*	2	0
09 0 66G	*	*	0	0
09 0 66H	*	*	12	0
09 0 66J	*	*	4	0
09 0 67A	*	*	0	0
09 0 67B	*	*	3	0
09 0 67C	*	*	0	0
09 0 67D	*	*	0	0
09 0 67E	*	*	1	0
09 0 67F	*	*	0	0
09 0 67G	*	*	0	0
09 0 67H	*	*	0	0
09 0 67J	*	*	0	0
09 0 67K	*	*	1	0
09 0 67X	*	*	0	0
09 0 68A	*	*	0	0
09 0 68B	*	*	0	0
09 0 68C	*	*	0	0

[illegible]

B.2 CONVERTED INPUT FILES

B.2.1 AUTOREP FILE

02	WCBWW	X	AKO	24
03	WCBWW	X	AKO	85
04	WCBWW	X	AKO	93
05	WCBWW	X	AKO	104
06	WCBWW	X	AKO	140
07	WCBWW	X	AKO	180
08	WCBWW	X	AKO	226
09	WCBWW	X	AKO	255
10	WCBWW	X	AKO	204
11	WCBWW	X	AKO	64
12	WCBWW	X	AKO	68
13	WCBWW	X	AKO	74
14	WCBWW	X	AKO	74
15	WCBWW	X	AKO	74
16	WCBWW	X	AKO	74
17	WCBWW	X	AKO	74
18	WCBWW	X	AKO	74

B.2.2 MOBMAN REQUIREMENTS FILE

01	E	CS	59	DEG	191
02	E	CS	59	DEG	138
03	E	CS	59	DEG	183
04	E	CS	59	DEG	273
05	E	CS	59	DEG	214
06	E	CS	59	DEG	44
07	E	CS	59	DEG	44
08	E	CS	59	DEG	44
09	E	CS	59	DEG	74
10	E	CS	59	DEG	74
11	E	CS	59	DEG	74
12	E	CS	59	DEG	54
13	E	CS	59	DEG	54
14	E	CS	59	DEG	54
15	E	CS	59	DEG	63
16	E	CS	59	DEG	63
17	E	CS	59	DEG	63
18	E	CS	59	DEG	177
01	E	CS	14	DEG	285
02	E	CS	14	DEG	202
03	E	CS	14	DEG	260
04	E	CS	14	DEG	390
05	E	CS	14	DEG	307
18	E	CS	14	DEG	276

B.2.3 MOBMAN ASSETS FILE

01	E	CS	59	IRR	4046
01	E	CS	59	THS	1274
01	E	CS	59	STY	4044
01	E	CS	59	RET	13211
01	E	CS	14	THS	1071
01	E	CS	14	IRR	14530
01	E	CS	14	STY	1734
01	E	CS	14	RET	96
01	E	MI	14	THS	188
01	E	MI	59	THS	336
01	E	MI	59	IRR	1621
01	E	MI	59	STY	332
01	E	MI	59	RET	2559
01	E	MI	14	IRR	2275
01	E	IN	14	THS	1883
01	E	IN	14	IRR	23949
01	E	IN	14	STY	63
01	E	IN	59	THS	1630
01	E	IN	59	IRR	5674
01	E	IN	59	STY	829
01	E	IN	59	RET	10042
01	E	CE	14	THS	1128
01	E	CE	14	IRR	13920
01	E	CE	14	STY	104
01	E	CE	14	RET	91
01	E	CE	59	THS	531
01	E	CE	59	IRR	1836
01	E	CE	59	STY	582
01	E	CE	59	RET	3088
01	E	FA	14	THS	1067
01	E	FA	14	IRR	13441
01	E	FA	14	STY	29

B.2.4 CSM II FILE

TP	CATBRGD	S	BATTLE STR	NON-BATTLE STR	TOTAL STR
01	OADCO	X	34	0	34
01	OADFD	X	15	0	15
01	OARCO	X	141	0	141
01	OARFD	X	60	0	60
01	OAVCO	X	13	0	13
01	OAVFD	X	5	0	5
01	OCECO	X	40	0	40
01	OCEFD	X	17	0	17
01	OCMCO	X	6	0	6
01	OCMFD	X	3	0	3
01	OCSCO	X	14	0	14
01	OCSFD	X	3	0	3
01	OFACO	X	83	0	83
01	OFAFD	X	36	0	36
01	OINCO	X	184	0	184
01	OINFD	X	78	0	78
01	OMCCO	X	27	0	27
01	OMCFD	X	10	0	10
01	OMICO	X	7	0	7
01	OMIFD	X	2	0	2
01	OMPCO	X	8	0	8
01	OMPFD	X	4	0	4
01	OODCO	X	26	0	26
01	OODFD	X	11	0	11
01	OQMCO	X	7	0	7
01	OQMFD	X	2	0	2
01	OSCCO	X	24	0	24
01	OSCFD	X	11	0	11
01	OTCCO	X	20	0	20
01	OTCFD	X	10	0	10
01	WCBWW	X	40	0	40
01	WCCWW	X	52	0	52
01	WCSWW	X	109	0	109
01	EAD14	X	386	0	386
01	EAD59	X	165	0	165
01	EAR14	X	1428	0	1428
01	EAR59	X	612	0	612
01	EAV14	X	21	0	21
01	EAV59	X	8	0	

B.2.5 MOBTNGBS FILE

TP	CATBRGD	S	TYPE	STR
01	EAD14		TRN	140
01	EAR14		TRN	250
01	EAV14		TRN	302
01	ECE14		TRN	515
01	ECM14		TRN	68
01	ECS14		TRN	758
01	EFA14		TRN	561
01	EIN14		TRN	763
01	EMC14		TRN	831
01	EMI14		TRN	235
01	EMM14		TRN	68
01	EMP14		TRN	273
01	EOD14		TRN	920
01	EQM14		TRN	1401
01	ESC14		TRN	582
01	ESM14		TRN	38
01	ETC14		TRN	659

ANNEX C: OUTPUT REPORT FORMATS

C.1 PREPROCESSOR OUTPUT (RECAST FILE)

CAT/BR GRADE	S	REQ/ TYPE	TIME PER/ PRIORITY	REQ'T/ ASSETS
OARFD	M	DEG	010010010	46
OARFD	M	IRR	010010400	671
OARFD	M	STY	010010500	390
OARFD	M	RET	010010600	515
OAVFD	X	DEG	010020010	10
OAVFD	X	IRR	010020400	415
OAVFD	X	STY	010020500	340
OAVFD	X	RET	010020600	305
OINFD	M	DEG	010030010	78
OINFD	M	IRR	010030400	1462
OINFD	M	STY	010030500	1143
OINFD	M	RET	010030600	1340
OFAFD	M	DEG	010040010	41
OFAFD	M	IRR	010040400	796
OFAFD	M	STY	010040500	427
OFAFD	M	RET	010040600	417
OADFD	X	DEG	010050010	9
OADFD	X	IRR	010050400	209
OADFD	X	STY	010050500	127
OADFD	X	RET	010050600	87
OARCO	M	DEG	010060010	379
OARCO	M	IRR	010060400	1758
OARCO	M	STY	010060500	166
OARCO	M	RET	010060600	87
OAVCO	X	DEG	010070010	39
OAVCO	X	IRR	010070400	483
OAVCO	X	STY	010070500	61

C.2 RECLASSIFICATION MODEL OUTPUT (MODRQAST FILE)

CAT/BR GRADE	S	REQ/ TYPE	TIME PER/ PRIORITY	REQ'T/ ASSETS
OARFD	M	DEG	010010010	46
OARFD	M	THS	010010200	60
OARFD	M	IRR	010010400	671
OARFD	M	STY	010010500	390
OARFD	M	RET	010010600	515
OAVFD	X	DEG	010020010	10
OAVFD	X	THS	010020200	84
OAVFD	X	IRR	010020400	415
OAVFD	X	STY	010020500	340
OAVFD	X	RET	010020600	305
OINFD	M	DEG	010030010	78
OINFD	M	THS	010030200	132
OINFD	M	IRR	010030400	1462
OINFD	M	STY	010030500	1143
OINFD	M	RET	010030600	1340
OFAFD	M	DEG	010040010	41
OFAFD	M	THS	010040200	72
OFAFD	M	IRR	010040400	796
OFAFD	M	STY	010040500	427
OFAFD	M	RET	010040600	417
OADFD	X	DEG	010050010	9
OADFD	X	THS	010050200	34
OADFD	X	IRR	010050400	209
OADFD	X	STY	010050500	127
OADFD	X	RET	010050600	87
OARCO	M	DEG	010060010	379
OARCO	M	THS	010060200	291
OARCO	M	IRR	010060400	1758
OARCO	M	STY	010060500	166
OARCO	M	RET	010060600	87
OAVCO	X	DEG	010070010	39
OAVCO	X	THS	010070200	187
OAVCO	X	IRR	010070400	483
OAVCO	X	STY	010070500	61

C.3 CRC/RPL CO MODEL OUTPUT FILE (CRC_*.OUT/RPL_*.OUT---* THREE LTR
REQUIREMENT CODE)

CAT/BR GRADE	S	REQ/ TYPE	TIME PER/ PRIORITY	REQ'T/ ASSETS	REQ'T FILLED	ASSET USED
OARFD	M	DEG	010010010	46	48	
OARFD	M	THS	010010200	60		48
OARFD	M	IRR	010010400	671		0
OARFD	M	STY	010010500	390		0
OARFD	M	RET	010010600	515		0
OAVFD	X	DEG	010020010	10	10	
OAVFD	X	THS	010020200	84		10
OAVFD	X	IRR	010020400	415		0
OAVFD	X	STY	010020500	340		0
OAVFD	X	RET	010020600	305		0
OINFD	M	DEG	010030010	78	80	
OINFD	M	THS	010030200	132		80
OINFD	M	IRR	010030400	1462		0
OINFD	M	STY	010030500	1143		0
OINFD	M	RET	010030600	1340		0
OFAFD	M	DEG	010040010	41	42	
OFAFD	M	THS	010040200	72		42
OFAFD	M	IRR	010040400	796		0
OFAFD	M	STY	010040500	427		0
OFAFD	M	RET	010040600	417		0
OADFD	X	DEG	010050010	9	9	
OADFD	X	THS	010050200	34		9
OADFD	X	IRR	010050400	209		0
OADFD	X	STY	010050500	127		0
OADFD	X	RET	010050600	87		0
OARCO	M	DEG	010060010	379	379	
OARCO	M	THS	010060200	291		291
OARCO	M	IRR	010060400	1758		0
OARCO	M	STY	010060500	166		0
OARCO	M	RET	010060600	87		0

C.4 TRANSPORTATION MODEL OUTPUT (TRANS.OUT)

TP	CAT/BR GRADE	S	THEATER	REQ	CRC FLOW	RPL FLOW	THEATER PER	FLOW DIF
01	OARFD	M	DEG	46	48	46	100.0%	2
01	OAVFD	X	DEG	10	10	10	100.0%	0
01	OINFD	M	DEG	78	81	78	100.0%	3
01	OFAFD	M	DEG	41	43	41	100.0%	2
01	OADFD	X	DEG	9	9	9	100.0%	0
01	GARCO	M	DEG	379	395	379	100.0%	16
01	OAVCO	X	DEG	39	41	39	100.0%	2
01	OINCO	M	DEG	514	535	514	100.0%	21
01	OFACO	M	DEG	207	216	207	100.0%	9
01	OADCO	X	DEG	67	70	67	100.0%	3
01	OCEFD	X	DEG	4	4	4	100.0%	0
01	OSCFD	X	DEG	21	22	21	100.0%	1
01	OCECO	X	DEG	1	1	1	100.0%	0
01	OSCCO	X	DEG	73	76	73	100.0%	3
01	EAR59	M	DEG	2138	1449	1511	70.7%	-62
01	EAV59	X	DEG	112	0	0	0.0%	0
01	EIN59	M	DEG	2690	0	0	0.0%	0
01	EFA59	M	DEG	1010	0	0	0.0%	0
01	EAD59	X	DEG	356	0	0	0.0%	0
01	EAR14	M	DEG	2803	0	0	0.0%	0
01	EAV14	X	DEG	133	0	0	0.0%	0
01	EIN14	M	DEG	4985	0	0	0.0%	0
01	EFA14	M	DEG	1702	0	0	0.0%	0
01	EAD14	X	DEG	483	0	0	0.0%	0
01	ECE59	X	DEG	449	0	0	0.0%	0
01	ESC59	X	DEG	652	0	0	0.0%	0
01	ECE14	X	DEG	1213	0	0	0.0%	0
01	ESC14	X	DEG	639	0	0	0.0%	0
01	OMCFD	X	DEG	59	0	0	0.0%	0
01	OMIFD	X	DEG	16	0	0	0.0%	0
01	OMPFD	X	DEG	7	0	0	0.0%	0
01	OCMFD	X	DEG	7	0	0	0.0%	0
01	OMCCO	X	DEG	139	0	0	0.0%	0
01	OMICO	X	DEG	53	0	0	0.0%	0
01	OMPCO	X	DEG	29	0	0	0.0%	0
01	OCMCO	X	DEG	29	0	0	0.0%	0
01	EMI59	X	DEG	192	0	0	0.0%	0
01	EMC59	X	DEG	487	0	0	0.0%	0
01	EMP59	X	DEG	86	0	0	0.0%	0