

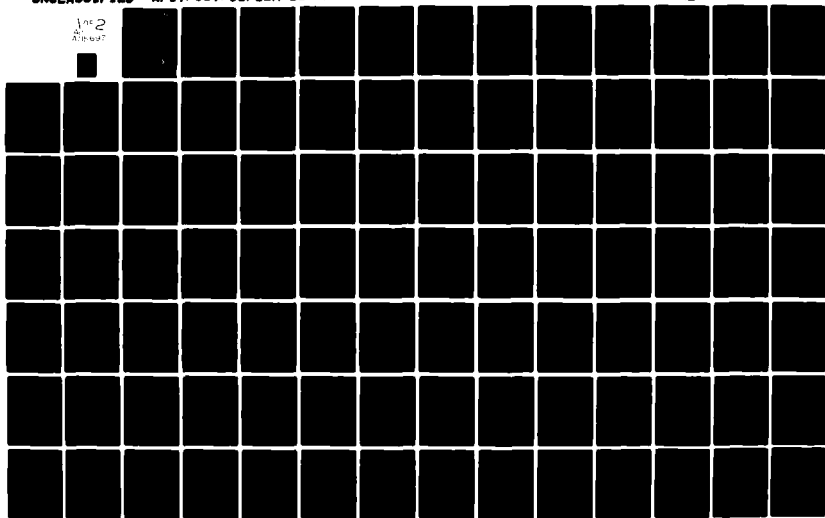
AD-A115 697

AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCH00--ETC F/G 15/7
THEATER WARFARE PROGRAMS AT AFIT: AN INSTRUCTIONAL AID.(U)
MAR 82 A WAIKANEN
AFIT/6ST/OS/82M-15

UNCLASSIFIED

NL

1 of 2
AFIT/6ST



AD A115697



THEATER WARFARE PROGRAMS AT AFIT:
AN INSTRUCTIONAL AID

THESIS

AFIT/GST/OS/82M-15

Anthony Waisanen
Captain USAF

DTIC
ELECTE
S JUN 18 1982 D

RECEIVED COPY

Approved for public release; distribution unlimited.

82 06 16 006

AFIT/GST/OS/82M-15

THEATER WARFARE PROGRAMS AT AFIT:
AN INSTRUCTIONAL AID

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
In Partial Fulfillment of the
Requirements for the Degree of
Master of Science

by

Anthony Waisanen

Captain USAF

Graduate Strategic and Tactical Sciences

March 1982

Approved for public release; distribution unlimited.



Accession For	
NTIS GRA&I	☒
DTIC TAB	☐
Unannounced	☐
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A	

Preface

One of the courses offered by the Air Force Institute of Technology (AFIT) is the Combined Warfare course (ST7.01) which is designed to teach concepts of warfare. The first year this course was offered, the class was sent to Maxwell AFB to participate in the playing of the Theater Warfare Exercise (TWX). This proved to be an invaluable addition to the course. Unfortunately, the exercise was designed solely for operation on a Honeywell computer. If the exercise were to run at AFIT, it would need to be compatible with at least one of the computers at Wright-Patterson AFB; the CDC or the Harris.

This report describes the modifications which were required to make the programs, data and operation compatible with the CDC computer.

I wish to thank Col Don Stevens for suggesting this topic. Hopefully, future students will derive as much insight from TWX as I have. I am indebted to Maj Dan Fox for his patience and suggestions during the many hours we spent discussing this problem. I could not have completed half of this thesis effort had he not introduced me to the UEDIT text editor and the PDP-11/60.

I also wish to thank the computer personnel at HQ AU/ACDY and AFIT/AD for taking the time to talk about how the TWX programs were designed and how they operate. In particular, I wish to thank Lt Ed Laugel for the many hours he spent explaining the origins of the TWX and the operation of Honeywell computers.

Capt Carl Lizza is also to be thanked for his assistance in the segmentation process and debugging of the programs. SrA John D. Long's help and knowledge of the CDC was greatly appreciated. AFIT would not have the TWX if it were not for these people.

I wish to thank my fellow classmates who suggested modifications, spent considerable effort in developing a test data base, and helped in so many other ways.

Finally, my wife Holly is to be thanked for her patience and understanding, and support through the many difficult periods during this project.

Anthony Waisanen

TABLE OF CONTENTS

	<u>Page</u>
Preface	ii
List of Abbreviations	vi
List of Tables	vii
Abstract	ix
I Introduction	1
Background	1
Problem Statement	2
Scope and Limitations	3
II Modifications	6
Data transfer	6
Program Modification	8
General	8
Compatible Modifications	8
Incompatible Modifications	10
Batch Programs	11
Interactive Programs	13
Utility Programs	15
Library Routines	17
Verification	17
III Segmentation	18
IV Operation of the Modified TWX Programs	21
General	22
Initialization	23
Execution	24
V Recommendations	26

Bibliography	29
Appendix A	30
INPTGO Procedure File/Program Listing	31
Appendix B	39
TWX Library Routines Listings	40
Appendix C	74
Segmentation Procedure File Listings	75
Appendix D	86
SETUP Listing	87
Appendix E	90
File Descriptions	91
Vita	97

List of Abbreviations

<u>Abbreviation</u>	<u>Definition</u>
AAFCE	Allied Air Forces, Central Europe
AFIT	Air Force Institute of Technology
AG	Air to Ground interface program
APE	AAFCE Planning Executive program
AR	Air battle simulation program
CAWC	Combined Air Warfare Course
CDC	Control Data Corporation (computer)
IP	Input Print program
LA	Logistics Analysis program
LB	Land battle simulation program
LI	Land order Input program
MA	Merge Actions program
MI	Mission order Input program
MR	Merge Records program (2ATAF and 4ATAF)
OR	Overrun program
SQ	Merge Records program (Blue and Red sides)
TWX	Theater Warfare Exercise
XX	Original main driver routine

List of Tables

<u>Table</u>		<u>Page</u>
I	COMMON Block Modifications	9
II	Deleted TWX JCL-type Subroutines	14
III	Read and Modification Utility Programs	16
IV	Daily File Modifications	22
V	TWX Object Decks and Libraries	23
VI	TWXDATA Files	93
VII	TWXPROG Files	94
IIX	TWXRUN Files	95
IX	TWXUTIL Files	96

Abstract

This thesis report details the processes and modifications that were required in order to execute the Theater Warfare Exercise (TWX) on the CDC computers at Wright-Patterson AFB. Prior to this effort, the TWX programs and data files could only be accessed and executed with Honeywell computers.

By modifying the data files, program coding, overlaying techniques, and operation, the TWX can now be run on any CDC computer. Any other computer with sufficient central memory and an ANSI standard FORTRAN-77 compiler can also execute the programs provided the operation methods (procedure files) are modified.

THEATER WARFARE PROGRAMS AT AFIT:
AN INSTRUCTIONAL AID

I Introduction

One of the objectives of a military institution like the Air Force Institute of Technology (AFIT) is to give its graduates the education necessary for them to be able "to understand their cultural and technological environment and to analyze and attempt to solve its problems (Ref 1:2+)." Ideally, the students should have the opportunity to apply what they have learned prior to graduating. Thus, their analyses could be critiqued by other analysts in a controlled setting. Such an interaction could instill in the student an understanding of the role and process of analyses in the military. Students in the School of Engineering have very full class schedules. Since non-computerized exercise would take too much time and effort to be useful, some computerization is needed. Thus, attention is directed to computerized exercises.

Background

One way of gaining this experience is to participate in the Theater Warfare Exercise (TWX) which is played as part of the Combined Air Warfare Course (CAWC) at Maxwell AFB. This is not a practical solution however considering the time-intensive class-

work required for all AFIT courses. Nor, due to TDY expenses, is it financially feasible since the TWX typically requires at least two days for introduction and preplanning and five days of execution for a minimum of seven days. This was done in February 1981 with the first class of Combined Air Warfare Seminar (ST7.01) students but there were only four students taking the course. Future classes are expected to number in excess of 15.

Another alternative would be to use a simulation in existence at AFIT such as STAG (Ref 4). However, STAG is only a two-player simulation with very limited realism and interaction. It was not designed to represent real-world results (Ref 4:13).

The most desirable option would be to perform the TWX at AFIT. This option offers a realistic exercise without excessive disruption to the schedules of the players and without requiring a large TDY budget. This would require the modification of the programs currently being used for the CAWC to a set of programs which could be run on either the Harris or CDC computers at Wright-Patterson AFB.

Problem Statement

Prior to this effort, the TWX programs could only be run on Honeywell computers. If these programs are to be used at AFIT, they must be able to execute on a variety of computers; specifically, the Harris and CDC computers. This required that the coding and data be transferred to one of these computers.

It was decided to convert the programs to ANSI standard

FORTRAN-77. This resulted in enhanced portability. FORTRAN-77 was selected since all of the TWX programs use character variables and ENCODE and DECODE statements (which are the FORTRAN-77 equivalent of internal file WRITES and READs).

A significant effort was required to develop job control language (JCL) to access the data files and execute the programs. In the original programs (Ref 7), data file access was accomplished with Honeywell machine dependent code in the programs and not with the JCL. Since file manipulation is strictly a function of the type of computer system being used, this was changed so that file manipulation would be completely external to the programs. Thus, transportation of the TWX programs from one computer system to another could be accomplished without requiring modification of the TWX source code.

Scope and Limitations

The purpose of this thesis effort was to create a set of theater level warfare exercise programs at AFIT that are functionally identical to those used in the CAWC at Maxwell AFB on August 1981. That is, from a player's and analyst's point of view the programs should appear to operate in identical fashion to those used in the CAWC. This way, players, analysts, and exercise administrators who are familiar with one set of the TWX will be able to function with the other set without lengthy learning periods. This also permits the use at AFIT of program documentation and exercise directions which have already been

developed for the exercise at Maxwell AFB. This effort was limited to the transfer and modification of the data files and program coding required to execute the programs at AFIT. None of the algorithms used in determining troop movement, losses, and others were modified unless the program would not otherwise run.

Because of its accessibility and large storage capacity, the CDC computer was used in making the modifications and developing the JCL.

Documentation of this thesis effort is was limited to the description of that coding which had to be modified. According to one of the TWX programmers (Ref 6), the original directives to the TWX programmers included the requirement for extensive documentation of every routine. Therefore, this documentation together with the documentation contained in this thesis should adequately describe the modified programs.

Finally, modifications to the coding were as limited as possible and no modifications were made to the data file formats. When system-specific intrinsic routines were found (for example, routines which determine binary file size), equivalent FORTRAN-77 coding was developed. The resulting code was placed in a separate library, not in the TWX programs themselves. This way, the translated TWX programs appear to be as similar as possible to the original and all system-specific coding is located only in the library. In those instances where a routine could not be recoded because of name similarity (for example, ENCODE), the program was modified by changing the offending code into comment statements and putting the equivalent coding directly after it.

Calls to some routines like FMEDIA and CREATE are required only by the Honeywell computers (Ref 5). These calls were changed to comment statements.

Along with the TWX programs, source code for 25 utility programs were also transferred. These programs are used at Maxwell AFB to manipulate or modify the data bases. Because they are highly system-specific and contain coding which performs JCL functions, they have not been converted and, in some cases, similar programs have been developed. Their code has been retained on file for future reference.

II Modifications

There were three major areas involved with modification; data transfer, coding modification, and verification of the modified coding. Data transfer involved the conversion of the extensive data bases used by the TWX programs from a Honeywell-specific format and data representation to a CDC-specific format and data representation. Coding modification involved the identification of Honeywell-specific coding and the development of equivalent code which is not specific to one computer. Verification was limited in that only those routines which were developed to replace the Honeywell intrinsic routines were thoroughly tested; the major TWX programs were verified to the extent that their operation was identical to the originals.

Data Transfer

Two types of data files are used in the TWX programs; sequential card image and random access binary (Ref 2). The transfer of the card image files was only a matter of reading the magnetic tape; the binary files required considerably more modification. Since the original data was stored as 36-bit/word binary, it had to be read in binary form and then written to tape as card image. The program used to do this read and wrote only real values. Thus, character variables such as the target names in file RLUM had to be input manually. Other than this problem,

the conversion into 60-bit/word binary form only required short programs that read card image and wrote binary. A different program was written for each format of data file. Each of these transfers was verified using a similar program that read binary and wrote card image. The output from these programs was compared to the card image listings obtained during the file transfers.

Program Modification

There are three types of programs in TWX; batch, interactive, and utility. These programs and the library routines used to replace Honeywell specific system routine calls differ greatly in both function and form and so will be discussed separately. First, however, is a general discussion of the major compatibilities and incompatibilities between the Honeywell FORTRAN compiler and the ANSI standard FORTRAN-77 compiler.

General. There were two types of modifications to the TWX programs; modifications which were compatible with both the Honeywell and CDC computers and modifications which were incompatible with the Honeywell FORTRAN compiler.

Compatible Modifications. One of the differences in compilers is the maximum allowable size of labels and variable names. In Honeywell FORTRAN, a maximum of 8 characters is allowed for variable, entry, subroutine, and program names. To minimize the dissimilarity between the original and modified versions, the seventh and eighth characters were deleted since only 6 characters are allowed by an ANSI standard FORTRAN-77 compiler. Only when a conflict would result were any of the other characters altered.

Another difference is that the Honeywell compiler allows character and numeric variables to be in the same COMMON block. This is not allowed in FORTRAN-77. When this occurred, the character variables were placed in a separate but similarly labelled COMMON block (Table I).

TABLE I

COMMON Block Modification

Program	Original Common	Modified Numeric	Commons Character
AG	ACCNTR	ACCNTR	ACCNTC
APE	LAINB	LAINB	LAINBC
AR	BUFFER	BUFFER	BUFFEC
	CNTR0L2	CNTRL2	CNTR2C
LA	ACDATA	ACDATA	ACDATC
	LGDATA	LGDATA	LGDATC
LB	PAGEDATA	PAGEDA	PAGECH
	TABLDATA	TABLDA	TABLCH
MI	CONTROL	CONTR0	CONTRC
	CYCLE	CYCLE	CYCLC
	READIN	READIN	READIC
OR	OUTPUT	OTPUT	OTPUTC
	TGOUT	TGOUT	TGOUTC

In some cases source code records extended past the 72nd column after being formatted for the FORTRAN-77 compiler. For example, a record is identified as a continuation by the Honeywell compiler if an ampersand is in the first through sixth column. The FORTRAN-77 compiler identifies a continuation record by the presence of any character in the sixth column. This meant that all continuation lines that did not begin in the sixth column had to be reformatted to be compatible with the FORTRAN-77 compiler which made some records more than 72 characters long, the maximum record length. In these cases, the extended coding was continued on the following line.

Probably the most troublesome difference between the computer systems is that the Honeywell loader initializes all variables to 0. Since the CDC initializes all variables to negative inde-

finite, all variables in labelled commons were set to 0 unless they were defined by the program.

Incompatible Modifications. Not all of the modifications made to the TTWX programs at AFIT are compatible with the Honeywell FORTRAN compiler. For example, free-field READ and PRINT statements in Honeywell FORTRAN are:

```
READ ,xxxx
```

```
PRINT ,xxxx
```

but in ANSI standard FORTRAN-77 they are:

```
READ *,xxxx
```

```
PRINT *,xxxx
```

Similarly, direct access files are read and written in Honeywell FORTRAN using the statements:

```
READ (lud`rec) xxx
```

```
WRITE (lud`rec) xxxx
```

where

lud = device number

rec = record number

The FORTRAN 77 compiler, however, uses the statements:

```
READ (lud,REC=rec) xxxx
```

```
WRITE (lud,REC=rec) xxxx
```

These statements occur infrequently only in the interactive programs so modification of the coding back to a form compatible with the H6000 compiler is not difficult.

The original programs made extensive use of intrinsic routines which would modify the bits of certain variables. Since this type of operation depends directly on the number of bits per

word being used by a given computer, the routines were recoded and placed in the library file. Instead of modifying the bits of a numeric variable, these new library routines modify the characters of character-type variables (with each variable consisting of 36 characters which represent the original 36 bits/word). This implementation allows the modified programs to run on any machine using ANSI standard FORTRAN-77 regardless of word length. This means that the modified use of Boolean variables and Honeywell intrinsic routines is not easy to modify back to the original.

All calls to the Honeywell system routines "LINK" and "LLINK" have been changed to comment statements since they operated the Honeywell overlay procedures.

Finally, every effort was made to exclude system-dependent coding and functions. However, some routines cannot realistically function without them. For example, subroutine DATIM, which returns the current date and time, uses two CDC intrinsic functions "DATE" and "CLOCK." These may require recoding if another system is to be used.

Batch Programs. The purpose of the batch programs is to perform the simulation of a theater-level war using the player-defined inputs. The players only see the results of these programs which they use in planning the following day's missions. Since these programs are never directly accessed by the players, data file access is performed by JCL rather than by program coding and no input/output routines required coding modification. One modification which was required was the use of logical vari-

ables.

One of the options indirectly available to the players is the number of reports that will be printed by the programs AR, LB, OR, and LA (words 1 through 19 on records 2 and 3 of file RMRx# control this). The option is indirect in that the values can be changed but a utility program must be used. These values are used as counters and sometimes as logicals with integer 0 being interpreted as "FALSE" by Honeywell FORTRAN. Since the CDC FORTRAN-77 compiler interprets all non-negative values to be "FALSE", the logical variables (words 1, 11, 12, 13 and 15) have been set to -1. Since words 8 and 9 are used by LB as logicals and integers, they have also been set to -1 and the coding of subroutines REPO8 and REPO9 have been modified to use their absolute values. The resulting coding will thus function on the CDC and Honeywell computers.

The air battle simulation program AR required more extensive modifications. Originally, it was designed to simulate tactical airlift (TAL), air force augmentation (AUG) and aircraft role change (RC) functions as well as air battle simulation. Since the TAL, AUG, and REC functions are now performed by APE, the subroutines DITAL, DUAUG, and DIRC are no longer needed according to CAWC personnel at Maxwell AFB (Ref 9). They have been commented out.

Finally, the use of maximums was modified in AR. In TWX, there is a maximum number of bases/side, aircraft types/side, cycles/day, munition types/side, corps/side, days/exercise, and seminars (words 13 through 20 of record 4 of RMRx#). First, the

array size of variable MAX was reduced from 9 to 8 since the FORTRAN-77 compiler does not permit the reading of more words per record than were specified when the file was declared. Second, the actual number of bases/side, aircraft types/side, and so forth were used as counters in place of the equivalent values of MAX.

Interactive Programs. Some of the important features of TWX are the allocation and apportionment of air resources, movement of materiel, and mission definition. These functions are performed by three interactive programs; APE (AAFCE Planning Executive program), MI (Mission Input program), and LI (Land Input program). All of these programs prompt the player for directions and modify the appropriate data files; hence, the term, interactive. These programs all had subroutines which determined the files to be accessed based upon the side, seminar number, and, for mission inputs, the ATAF number. After this information was input, the files were accessed using system dependent code within the programs. Since this function is performed with JCL on the CDC, these subroutines (Table II) were removed from both programs and the program INPTR was developed (Appendix A).

TABLE II

Deleted TWX JCL-type Subroutines

Program	Subroutines
APE	PIATCH, PILO, PILOPS, PILOSEM and PILOSIDE
LI	PILOSEM and PILOSIDE
MI	PIATCH, PILO, PILOPS, PILOSEM and PILOSIDE

It should be noted that INPTR, together with the JCL necessary to compile it and catalog the object deck, comprise the file INPTGO in the Indirect File System (IFS) file TWXRUN.

The purpose of INPTR is to interactively construct the procedure file XECUTE. If the player correctly inputs the information together with the appropriate password, XECUTE will execute either of three procedure files APEGO, MIGO, or LIGO (these are discussed in Chapter IV). Thus, converting the programs to operate on a different computer will only involve modifying these three procedure files and not any of the programs (assuming the other computer has a FORTRAN-77 compiler). If the player fails to input the correct password in three tries, XECUTE will copy a message for the player to get assistance and then terminate.

When a file is written or modified using a Honeywell computer, the new information is made permanent when information is read from the same area of the file. For this reason, the dummy array FLUSH is often read after a number of "writes". According to the programmers, this would allow continuation in the event of

accidental disruption without requiring a complete restart (Ref 6). This activity was replaced by the use of the JCL statements REPLACE and EXIT,S. If the program (APE, MI, or LI) runs successfully, the written file is put in an indirect access file. If the program is not successful and an abort condition is detected, the skipping of JCL is stopped by the EXIT,S statement and the same files are made permanent. This allows the player to continue from the point immediately prior to the error rather than requiring a total restart.

Utility Programs. Since 16 out of the 24 data files used by TWX are binary direct access, modifications to them can only be made by using interactive programs. Rather than have one program capable of using a variety of formats, a different program was written for each format. This resulted in ten read/read and modify programs (Table III). These programs are all capable of reading their applicable file(s) and many offer modification options. They all assume that the file to be read/modified is accessible as device 1 (in other words, "TAPE1").

TABLE III

Read and Modification Utility Programs

Program	Applicable File
RABREAD	RABx#
RACREAD	RACx
RAPREAD	RAPx#, RPPx# (no changes permitted)
RBLREAD	RBLx#, RRLx#, RMUX
RCRREAD	RCRx#
RLUREAD	RLUx#
RMRREAD	RMRx# (no changes permitted)
RPPREAD	RPPx#
RTGREAD	RTGx#
R2MREAD	R2Mx#, R4Mx#, RLGx and RMRx# (logicals are displayed as integers)

Using these programs to modify a file requires the use of the TWX Data Base Manual (Ref 2) since these programs print only values of the data, not meanings.

The operation of these utility programs is not automatic. Unlike the simulation programs which have procedure files to access the necessary tapes, the utility programs must be manually accessed, compiled and executed by the user with its appropriate input file also manually accessed by the user. Thus, deliberate and purposeful activities precede the modification of data files. This is to prevent accidental modification to the data files. All utility programs are stored in the IFS file TWXUTIL.

Library Routines

One of the goals of this effort was to minimize the modifications to the program code. Since all of the programs at some time or other use routines which are intrinsic only to a Honeywell computer, this meant that equivalent coding for those routines had to be developed and added to a TWX library called TWXLIB (Appendix B). Thus, instead of modifying the program coding, this library containing routines with the same names as Honeywell specific system routines is declared and the equivalent functions are performed.

Verification

There are two types of coding now in the TWX programs; modified coding and completely new routines (library routines) which were developed at AFIT. Since no documentation was available for the original set of programs, verification of the modified coding was limited to verifying that the programs executed like the originals. The library routines were all verified using test programs which would execute each aspect of each routine. Since these programs were only for testing purposes, they were retained until the verification was completed and, thus, are no longer on file.

III Segmentation

The major interactive programs APE, MI, and LI have two features in common: 1) a central memory requirement in excess of 65K (the maximum interactive core limit for the CDC) and 2) they were originally written in a highly structured form with each subroutine either modifying a particular set of arrays or checking and editing input data. This implied that these programs could be run where only the main driver and a very few subroutines would be in central memory at any one time without unduly increasing the run time. This, in turn, implied that some method of overlaying could be used to decrease the central memory requirement. Of all possible methods of overlaying, segmentation was chosen because of its transportability, greater flexibility (Ref 3:7-1), and because it does not require changes to the source programs.

The specific methods of segmentation are described in Chapter 7 of the CYBER LOADER VERSION 1 Reference Manual (Ref 3) but are not easily understood. Along with the specific rules for segmentation, the following rules have been found to be helpful:

1. In a TREE directive, the label field must not contain a program, subroutine, entry point, or function name that is in the object deck to be segment loaded. Also, this name must appear in the specification filed of another TREE directive. If it does not, it will be interpreted as another root segment.
2. A function or subroutine which is to be used by more

than one segment must be in the directive field or fields of one or more INCLUDE directives; it cannot be in both an INCLUDE directive field and a TREE directive field.

3. The label field of an INCLUDE field must be a function, subroutine, or program name and cannot be an arbitrary label or entry point name.

The first step taken in segmenting the TWX programs was to determine the subroutine interactions (that is, which subroutines call which other subroutines) and then construct a tree depicting those interactions.

The major tree (or "root segment") was constructed by using the main program as the root and the subroutines called by it as the branches. In the case where a subroutine in a branch would call other subroutines, another tree was made using this subroutine as its root and the subroutine name in the original tree was replaced with the label of this new tree. This procedure was used in segmenting APE, MI, and LI.

For example, the AAFCE Planning Executive Program, APE, calls subroutine PI (file initialization), OPTN (file modification option processor), and RAPUP (final data base modifications). The first two subroutines both call numerous subroutines and operate independently so both were declared as trees. This tree structure was converted into a set of SEGLOAD directives for processing by the CDC loader. In order to segment a program, the source is first compiled, the object deck is loaded and then

operated on by the SEGLOAD directives. The segmented object deck is then cataloged.

To automate this process, procedure files of JCL and SEGLOAD directives were developed to convert the source decks of APE, MI and LI into segmented object decks (Appendix C). The procedure files also catalog the resulting decks.

To execute any one of these procedure files, the command is

BEGIN,pname,lfn1,lfn2.

where:

pname = procedure name (APESEG, MISEG or LISEG)

lfn1 = local file name of the procedure file

lfn2 = local file name of the source deck (APE, MI, or LI)

If the local file name is the same as the procedure name, the procedure file is executed with the command:

pname,lfn2.

where:

pname = procedure name (APESEG, MISEG or LISEG)

lfn2 = local file name of the source deck (APE, MI, or LI)

IV Operation of the Modified TWX Programs

There are two groups of people who need to be involved whenever the TWX is executed: players and administrators. The players are those people who perform the analyses of the outputs, develop the air and land orders used in the simulation and input the data by executing the programs APE, MI, and LI. The activities of the players is described in the Theater Warfare Exercise (TWX) Players' Handbook (Draft Copy) (Ref 8).

The administrators are responsible for the initialization of the exercise and the direction of student activities. They also represent the "higher authority" capable of authorizing increases of forces and use of nuclear resources. These are the people responsible for the operation of the TWX, execution of the batch programs SQ, MR, IP, AR, AG, MA, LB, OR, and LA, and resetting of the file lock flags and exercise day indicators (Table IV). They are also responsible for the execution of the procedure SETUP (Appendix D) and the maintenance of the TWX files (Appendix E).

TABLE IV

Daily File Modifications

File	Variable Location		Reset to
	Record(s)	Word(s)	
RAPW#	5 - 48	1	-1 ("TRUE")
RBLW#	1	9	0.0
RRLW#	1	9	0.0
RLGW	2 - 3 2 - 3	1 - 2 16	Current Day 0
RLUW#	1	6 - 7	Current Day
RMRR#	1	7	Current Day
R2MW#	2 - 3	1 - 2	Current Day
	2 - 3	3-5,13,16-17	0
	4 - 265	1 - 20	0
R4MW#	2 - 3	1 - 2	Current Day
	2 - 3	3-5,13,16-17	0
	4 - 265	1 - 20	0

General

The operation of the TWX is in two phases: initialization and execution. Initialization is essentially the set-up of the necessary files and object decks. It also includes the running of programs to produce the data needed by the players for planning. The execution phase involves the cyclical execution of interactive and batch programs to simulate the daily publishing and execution of air and land orders. The initialization phase occurs any time prior to the exercise while the execution phase

lasts for a maximum of five simulated days.

Initialization

The first step in executing the TWX is the construction of all required object decks and libraries (Table V). This can be performed manually but the procedure file SETUP (Appendix D) has been developed for this purpose. To execute this procedure file, it is necessary to make a local copy using the IFS command "GET,SETUP,ID=TWXRUN" and then execute it with the statement "SETUP."

TABLE V

TWX Object Decks and Libraries

Programs	Object Decks
APE	APELGO
AR	ARLGO
LB	LBGO
LI	LIGO
MI	MIGO
Side	Libraries
BLUE	DATA11, DATA12, DATA13
RED	DATA21, DATA22, DATA23
Both	BACKUP, BATCHIN, MASTER

The next step is to execute the Weapon System Summary program (WSS). This too has been automated; the file WSSGO is re-

trieved from the indirect library file TWXRUN and then executed with the statement "WSSGO." The output from this program is stored in the indirect library file PRINT and is needed by the players for the duration of the exercise (it contains such information as the aircraft names, weapon loadings, and effectiveness indices).

After the initial introduction to the exercise, the players enter the preplanning stage. This stage requires that the batch programs AR, AG, MA, LB, OR, and LA be sequentially executed. The output files ARPLB1, ARPLR1, LBPLB1, LBPLR1, ORP2B1, ORP4B1, ORP2R1, ORP4R1, LAP2B1, LAP4B1, LAP2R1, and LAP4R1 are then retrieved from the indirect library file PRINT and given to the players for their preplanning. The execution phase of the exercise now begins.

Execution

The execution phase of the TWX begins with the players performing their mission planning and entering data with the interactive programs APE, MI and LI. When these programs have been executed, the batch programs SQ, MR, IP, AR, AG, MA, LB, OR, and LA are sequentially executed and the output files are given to the players for the next day's mission planning (each simulation day consists of a day and a night cycle). This phase continues until the exercise administrators terminate the exercise for a maximum of five simulation days.

In the original TWX, MR through LA were all major subrou-

tines of a single program, XX (the source for the main driver XX is retained as TWXSRCE in the indirect library file TWXPROG). These programs are run seperatedly to improve flexibility and to decrease turn-around time. In fact, only AR and LB require too much central memory to be executed in time sharing mode (it should be noted that OR does require segmentation to execute in time sharing mode). Rather than have an executive program which sequentially executes the batch routines, the JCL for the modified programs, beginning with AR, accesses the procedure file for the succeeding routine and executes it. These procedure files create and batch the JCL necessary to execute the succeeding program. Each of these procedure files has the suffix GOB and is stored in the indirect library file TWXRUN. SQ, MR, and IP, however, must be executed individually with procedure files SQGO, MRGO, and IPGO respectively.

After all batch programs have been executed, the appropriate data files are replaced and the cycle begins again with the players performing analysis and planning and again entering data using the interactive programs APE, MI and LI.

V RECOMMENDATIONS

Because of the amount of time which was required to translate the data and programs into a modifiable form, some problem areas could not be addressed with this thesis effort. This includes areas such as validation of the original algorithms, streamlining of the batch programs, modification of the land battle program, and complete and thorough documentation of all programs.

The validation and documentation of the original coding was not performed because the necessary analyst's manuals are either non-existent or are too scarce to be made available. The set of TWX programs used in the CAWC are continually being modified. Thus, it is recommended that the programs at AFIT be documented without reference to future manuals from Maxwell AFB.

The air and land battle programs (AR and LB respectively) both have central memory requirements in excess of 225K. Because of this and the large amounts of central processor and input/output time required, both jobs have very long turn-around times. One way of streamlining both programs would be to segment them. There are two procedure files in the indirect library file TWXRUN which have been designed to compile, segment, and catalog the segmented object decks of AR and LB. These procedure files are ARSEG and LBSEG respectively. Time did not permit the final verification of either of these two procedure files.

Currently, the land battle program and data files are not designed to provide information regarding future strategies.

This requires the players who are planning ground-support air sorties to guess what their ground forces will be doing during the next day of simulation. This clearly needs to be modified. A suggestion would be to modify the printout so it provides not only the location of the forces, but also its direction.

There is no automatic resetting of data files. Whenever APE, MI or LI is executed, some data file values are changed to indicate the status of the user. If the program terminates normally, these values (lock flags) are set to 1 indicating that the user had completed the desired modifications of those data bases. If those data bases are used by the same program, the user will be denied access to them until the lock flags are set to 0. Before another day of simulation can begin, the lock flags in files RBLW#, RRLW#, RLGW#, R2MW#, and R4MW# must be set to 0 and the exercise day updated in files RLGW#, RMRR#, R2MW#, and R4MW#. The program II (Input Initialization) was designed to perform this operation on the Honeywell, but II was not modified because of its highly specific coding. Currently, these file modifications are made manually using the utility programs described in Chapter II.

Finally, there is one unresolved problem. When either a segmented or unsegmented version of MI is executed and an invalid offensive counter air (OCA), battle area interdiction (BAI), or interdiction (IND) order is input, an error condition is detected and the player is asked if the order is to be changed. Regardless of whether the answer is "Y" or "N", a Mode 1 error occurs (an illegal address is specified). This condition appears to be

the result of the A0 register for subroutine EDITC1 being modified by subroutine CYFMT1. The addition of "PRINT" and "CONTINUE" statements has not changed the error condition; the error remains in the address for the variable, INTS02.

Bibliography

1. Air University. AFIT 1979-81 Catalogue, 18: (undated).
2. Air University. TWX Data Base Manual. Maxwell AFB: Alabama, undated.
3. Control Data Corporation. CYBER Loader Version 1 Reference Manual (Revision G). California: Control Data Corporation (1979).
4. Foley, John M. STAG: A Two Person Simulated Tactical Air War Game. Air Force Institute of Technology: Wright-Patterson AFB, Ohio, March 1980. (AFIT/GST/OS/80M-2).
5. Honeywell. Fortran. Computer manual. Minneapolis: Honeywell Corporation, January 1975.
6. Laugel, Edmund L. Conversations. Wright-Patterson AFB: Ohio (9 July 1981 to 9 March 1982).
7. Ritchey, Conrad. et. al. TWX computer program listings. Maxwell AFB, Alabama (12 August 1981).
8. Waisanen, Anthony. Theater Warfare Exercise (TWX) Players' Handbook (Draft Copy). School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, Ohio, 1982.
9. White, Fred. Telephone conversation. Maxwell AFB: Alabama (9 March 1982).

Appendix A

INPTGO Procedure File/Program Listing

```

1      .PROC,INPTGO.
2      .* THE PURPOSE OF THIS PROCEDURE FILE
3      .* IS TO COMPILE THE PROGRAM "INPTR" AND
4      .* THEN CATALOGUE THE OBJECT DECK AS "INPTRLGO"
5      .* UNDER THE USER'S ACCOUNT NUMBER.
6      .*
7      .* THE FIRST STEP IS TO RETURN ANY LOCAL FILE
8      .* NAMED "LGO."
9      RETURN,LGO.
10     .* THEN "LGO" IS REQUESTED.
11     REQUEST,LGO,*PF.
12     .* INPTR IS NOW COMPILED.
13     FTN5,I=PROG,LO=0,DB.
14     .* THEN THE OBJECT FILE IS CATALOGUED
15     CATALOG,LGO,INPTRLGO,XR=FOX,PW=FOX,RP=999.
16     .* THE LOCAL FILE AREA IS CLEANED.
17     RETURN,LGO,PROG.
18     .* CONTROL IS RETURNED TO THE USER.
19     REVERT.
20     .*
21     .* THIS LINE DEFINES LOCAL FILE "PROG"
22     .* WHICH WILL CONTAIN THE SOURCE OF INPTR
23     .DATA,PROG.

```

```

24      PROGRAM INPTR
25      COMMON/FILEDA/ICHOIS, ISEM, ISIDE, IATAF
26      COMMON/ITIMS/ITRY
27      DATA ICHOIS/0/, ISEM/0/, ISIDE/0/, IATAF/0/
28      OPEN(1,FILE='XECUTE',ACCESS='SEQUENTIAL')
29      REWIND 1
30      CX FIND OUT WHICH PROGRAM IS TO BE RUN
31      50 PRINT 12
32      READ *,ICHOIS
33      IF( ICHOIS .LT. 4 ) THEN
34          IF( ICHOIS .LT. 1 ) GO TO 50
35              ITRY = 0
36              CALL INTER
37          IF( ITRY .LT. 3 ) THEN
38              CALL BUILD
39              CALL INFORM
40          ELSE
41              CALL DONE
42          ENDIF
43      ELSE
44          CALL STOPR
45      ENDIF
46      STOP
47      12 FORMAT(' DO YOU WANT TO RUN THE:',/,
48      &' 1) AAFCE PLANNING PROGRAM',/,
49      &' 2) MISSION PLANNING PROGRAM',/,
50      &' 3) LAND MOVEMENT PROGRAM',/,
51      &' 4) STOP    >')
52      END

```

```

53      SUBROUTINE BUILD
54      COMMON/FILEDA/ICHOIS, ISEM, ISIDE, IATAF
55      CX THIS SUBROUTINE BUILDS THE APPROPRIATE JCL FILE
56      CX TO ACCESS ALL NECESSARY DATA FILES.
57      CX BRANCH ON TYPE OF PROGRAM
58      IF( ICHOIS .EQ. 1 ) THEN
59          WRITE(1,11) ISIDE, ISEM, IATAF
60      ELSE IF( ICHOIS .EQ. 2 ) THEN
61          WRITE(1,12) ISIDE, ISEM, IATAF
62      ELSE
63          WRITE(1,13) ISIDE, ISEM, IATAF
64      ENDIF
65      RETURN
66      11  FORMAT( '.PROC,XECUTE.',/,
67      &'APEGO,',I1,',',I1,',',I1,',',/,
68      &'EXIT,S.',/, 'REVERT.')
69      12  FORMAT( '.PROC,XECUTE.',/,
70      &'MIGO,',I1,',',I1,',',I1,',',/,
71      &'EXIT,S.',/, 'REVERT.')
72      13  FORMAT( '.PROC,XECUTE.',/,
73      &'LIGO,',I1,',',I1,',',I1,',',/,
74      &'EXIT,S.',/, 'REVERT.')
75      END

```

```

76          SUBROUTINE DONE
77
78 CX THIS SUBROUTINE BUILDS A FILE THAT WILL INFORM
79 CX THE USER OF HIS FAILURE TO PROPERLY ACCESS THE
80 CX THE FILES.
81      WRITE(1,12)
82      RETURN
83 12  FORMAT(' .PROC,XECUTE.',/, 'CONNECT,OUTPUT.',/,
84      &'COPY,KNUJ.',/, 'RETURN,KNUJ.',/,
85      &'RETURN,XECUTE,LGO.',/, 'REVERT.',/,
86      &' .DATA,KNUJ.',/, ' YOU ARE OBVIOUSLY HAVING PROBLEMS.',/,
87      &' PLEASE GET SOME HELP.',/, '.EOF')
      END

```

```

88          SUBROUTINE INFORM
89          COMMON/FILEDA/ICHOIS, ISEM, ISIDE, IATAF
90          P IN 12
91          12  FORMAT('      PLEASE SIT BACK FOR A FEW MINUTES WHILE',/,
92          &' YOUR FILES ARE ATTACHED.  THE PRINTOUT MAY BE',/,
93          &' RATHER MESSY BUT IS, UNFORTUNATELY, NECESSARY.',/,
94          &' DO NOT TOUCH ANY OF THE KEYS UNTIL THE PROGRAM',/,
95          &' REQUESTS SOME INPUT.  THANK-YOU FOR YOUR COOPERATION.')
96          RETURN
97          END

```

```

98          SUBROUTINE INTER
99          COMMON/FILEDA/ICHOIS, ISEM, ISIDE, IATAF
100         COMMON/ITIMS/ITRY
101         CX THE PURPOSE OF THIS SUBROUTINE IS TO INTERROGATE
102         CX THE STUDENT TO DETERMINE HIS SEMINAR # (ISEM), SIDE
103         CX (ISIDE, WHERE 1 => BLUE AND 2 => RED), AND ATAF
104         CX IF APPLICABLE.
105         CX
106         CHARACTER*4 INPT, IPASS(12), SIDE(2), IANS*1
107         DATA SIDE/'BLUE', 'RED' /
108         DATA IPASS/'PAS1','PAS2','PAS3','PAS4',
109         &          'PAS5','PAS6','PAS7','PAS8',
110         &          'PAS9','PAS0','PASA','PASB' /
111         CX NOW INTERROGATE FOR SIDE, SEMINAR, AND ATAF
112         10 PRINT *, 'WHICH SIDE ARE YOU ON, RED OR BLUE ? >'
113         READ 23, INPT
114         IF( INPT .NE. 'BLUE' .AND. INPT .NE. 'RED'
115         &   .AND. INPT .NE. 'RED' ) THEN
116             PRINT *, 'ONLY "BLUE" OR "RED" ALLOWED'
117             GO TO 10
118         ENDIF
119         ISIDE = 1
120         IF( INPT .NE. 'BLUE' ) ISIDE = 2
121         CX GET THE SEMINAR NUMBER
122         20 PRINT *, 'WHICH SEMINAR ARE YOU IN ( 1 OR 2 )? >'
123         READ *, ISEM
124         IF( ISEM .NE. 1 .AND. ISEM .NE. 2 ) THEN
125             PRINT *, 'ONLY 1 OR 2 ALLOWED'
126             GO TO 20
127         ENDIF
128         IF( ICHOIS .EQ. 2 ) THEN
129         CX GET THE ATAF NUMBER
130         30 PRINT *, 'WHICH ATAF ( 2 OR 4 )? >'
131         READ *, IATAF
132         IF( IATAF .NE. 2 .AND. IATAF .NE. 4 ) THEN
133             PRINT *, 'ONLY 2 OR 4 ALLOWED'
134             GO TO 30
135         ENDIF
136         ENDIF
137         CX CHECK IF INPUT DATA IS CORRECT SO FAR
138         IF( ICHOIS .NE. 2 ) THEN
139             IATAF = 0
140             PRINT 12, SIDE(ISIDE), ISEM
141         ELSE
142             PRINT 22, SIDE(ISIDE), ISEM, IATAF
143         ENDIF
144         READ 13, IANS
145         IF( IANS .NE. 'Y' ) GO TO 10
146         CX CALCULATE THE PASSWORD INDEX FOR THIS COMBINATION
147         INDEX = ISIDE * 6 + ISEM * 3 + IATAF/2 + 8

```

```

148      CX  ASK FOR THE PASSWORD
149      40  PRINT *, ' INPUT THE PASSWORD PLEASE >'
150      READ 23, INPT
151      IF( INPT .NE. IPASS(INDEX) ) THEN
152          ITRY = ITRY + 1
153          PRINT 32, INPT
154          IF( ITRY .LT. 3 ) GO TO 40
155      ELSE
156          PRINT *, ' PASSWORD IS CORRECT.'
157      ENDIF
158      RETURN
159      12  FORMAT( ' YOUR INPUTS ARE:',/, ' SIDE      : ',A4,/,
160      &      ' SEMINAR # : ',I2,/,/,
161      &      ' IS THIS CORRECT ( Y OR N ) >' )
162      22  FORMAT( ' YOUR INPUTS ARE:',/, ' SIDE      : ',A4,/,
163      &      ' SEMINAR # : ',I2,/, ' ATAF #      : ',I2,/,/,
164      &      ' IS THIS CORRECT ( Y OR N ) >' )
165      32  FORMAT( ' SO SORRY, BUT ',A4, ' IS NOT THE VALID PASSWORD.' )
166      13  FORMAT(A1)
167      23  FORMAT(A4)
168      END

```

```

169             SUBROUTINE STOPR
170 CX THIS SUBROUTINE SAYS GOOD-BYE TO THE STUDENT.
171 CX
172             WRITE(1,12)
173             RETURN
174             12  FORMAT(' .PROC,XECUTE.' ,/, 'CONNECT,OUTPUT.' ,/,
175 &'COPY,KNUJ.' ,/, 'RETURN,KNUJ.' ,/,
176 &'RETURN,XECUTE,LGO.' ,/, 'REVERT.' ,/,
177 &' .DATA,KNUJ.' ,/,
178 &' YOU REALLY SHOULD NOT HAVE CHOSEN THIS OPTION.' ,/,
179 &' IF THERE IS SOME QUESTION AS TO WHICH OPTION' ,/,
180 &' YOU SHOULD HAVE CHOSEN, PLEASE CONTACT THE' ,/,
181 &' GAME ADMINISTRATORS. THANK-YOU.' ,/,
182 &' .EOF' )
183             END
184 .EOF

```

Appendix B

TX Library Routines Listings

```

1      FUNCTION FLD( I, K, E )
2      CX*****
3      CX*
4      CX*      THE PURPOSE OF THIS FUNCTION
5      CX* IS TO RETURN K CHARACTERS OF WORD E
6      CX* BEGINNING WITH POSITION (I+1). THIS IS*
7      CX* THE FORTRAN 77 EQUIVALENT TO FUNCTION
8      CX* FLD AS DESCRIBED ON PAGE 6-7 OF THE
9      CX* HONEYWELL FORTRAN MANUAL. HOWEVER,
10     CX* ONLY THE FIRST FORM OF FLD IS HANDLED
11     CX* BY THIS FUNCTION. THE OTHER TYPE IS
12     CX* HANDLED BY FUNCTION FLD2.
13     CX*
14     CX***** C A U T I O N *****
15     CX*
16     CX* E MUST BE OF CHARACTER*36 TYPE,
17     CX* 0 .LE. I .LE. 35
18     CX* 1 .LE. K .LE. 36
19     CX*
20     CX*****
21     CX
22     CHARACTER *(*) E
23     J1 = I + 1
24     J2 = J1 + K - 1
25     FLD = FLOAT( ICH2N( E(J1:J2) ) )
26     RETURN
27     END

```

```

28          FUNCTION ICH2N( ICH )
29          C
30          C*****C
31          C
32          CX   AUTHOR: ANTHONY WAISANEN      C
33          CX           CAPT          USAF      C
34          CX   DECEMBER 1981                C
35          C                                  C
36          C   THE PURPOSE OF THIS FUNCTION   C
37          C   IS TO CONVERT A CHARACTER STRING TO C
38          C   AN INTEGER.                    C
39          C                                  C
40          C ***** C A U T I O N ***** C
41          C                                  C
42          C   YOU HAD BETTER BE AN INTEGER    C
43          C                                  C
44          C*****C
45          C
46          C   CHARACTER *(*) ICH
47          C
48          CX   FIND L (NUMBER OF CHARACTERS IN ICH)
49          C
50          C   L = LEN( ICH )
51          C
52          CX   CONVERT ICH TO AN INTEGER
53          C
54          C   ICH2N = 0
55          C   DO 15 I = 1, L
56          C   MOVE ICH2N OVER 1 DECIMAL PLACE
57          C   ICH2N = ICH2N * 10
58          C   ICHADD = ICHAR( ICH(I:I) ) * 16
59          CX   ADJUST IF OUTSIDE OF BOUNDS
60          C   IF( ICHADD .GT. 9 ) ICHADD = 0
61          CX
62          C   ICH2N = ICH2N + ICHADD
63          15  CONTINUE
64          C
65          C   RETURN
66          C   END

```

```

67      FUNCTION RAND( X )
68      CX*****
69      CX*
70      CX*   IN THE ORIGINAL PROGRAMS, "RAND" WAS *
71      CX* USED TO RETURN A UNIFORM RANDOM NUMBER *
72      CX* BETWEEN 0 AND 1. THIS IS EQUIVALENT TO*
73      CX* THE CDC FUNCTION "RANF," THEREFORE, *
74      CX* RANF WILL BE USED.
75      CX*
76      CX***** C A U T I O N *****
77      CX*
78      CX*   THIS FUNCTION MUST BE CHANGED IF A *
79      CX* DIFFERENT MAIN FRAME IS TO BE USED. *
80      CX*
81      CX*****
82      CX
83      Y = .273 E 03
84      Z = .1 E 01
85      X = AMOD( Y*X, Z )
86      RAND = X
87      RETURN
88      END

```

```

89             INTEGER FUNCTION ISETSW( N )
90             CX*****
91             CX*
92             CX*   THE PURPOSE OF THIS FUNCTION *
93             CX* IS TO SENSE THE VALUE OF THE NTH*
94             CX* BIT IN THE PROGRAM SWITCH WORD. *
95             CX* IN THE TWX PROGRAM, THE SWITCH *
96             CX* WORD IS ONLY USED TO SWITCH FROM*
97             CX* THEATER LEVEL SIMULATION ( WHERE*
98             CX* BIT #35 = 0 ) TO CORPS LEVEL *
99             CX* SIMULATION (BIT #35 = 1). SINCE*
100            CX* ONLY THEATER LEVEL SIMULATIONS *
101            CX* ARE DESIRED, THIS FUNCTION WILL *
102            CX* ALWAYS RETURN A 0. *
103            CX*
104            CX*****
105            CX
106                ISETSW = 0
107                RETURN
108                END

```

```

109      INTEGER FUNCTION KOMPCH( IWORD1, N1, IWORD2, N2, M )
110      CX*****
111      CX* THIS FUNCTION COMPARES M CHARACTERS OF IWORD1
112      CX* (BEGINNING WITH N1) WITH M CHARACTERS OF IWORD2
113      CX* (BEGINNING WITH N2). KOMPCH = 0 IF THEY ARE
114      CX* EQUAL 1 IF THEY ARE NOT.
115      CX*
116      CX***** C A U T I O N *****
117      CX* IWORD1 AND IWORD2 MUST BE CHARACTER TYPE
118      CX*
119      CX*****
120      CHARACTER *(*) IWORD1, IWORD2
121      LAST1 = N1 + M - 1
122      LAST2 = N2 + M - 1
123      CX INITIALIZE KOMPCH TO "FALSE"
124      KOMPCH = 1
125      CX COMPARE IWORD1 AND IWORD2
126      IF( IWORD1(N1:LAST1) .EQ. IWORD2(N2:LAST2) ) KOMPCH = 0
127      RETURN
128      END

```

```

129          LOGICAL FUNCTION CHKBIT(BIT,ARRAY,DIM)
130          C
131          C   CHECKS TO SEE IF BIT IS SET IN ARRAY - TRUE/FALSE
132          C   DEPENDING UPON BIT.
133          C   PIERCE.....1980
134          CX  WAISANEN.....1982
135          C
136          CX  INTEGER BIT,DIM,ARRAY,WORD,BIT1
137          CX  DIMENSION ARRAY(DIM)
138          INTEGER BIT,DIM,WORD,BIT1
139          CHARACTER*36 ARRAY(*)
140          C
141          WORD=((BIT-1)/36)+1
142          BIT1=MOD(BIT-1,36)
143          CHKBIT=(FLD(BIT1,1,ARRAY(WORD)).EQ.1)
144          RETURN
145          END

```

```

146          SUBROUTINE BCDASC( WORD1, WORD2, N )
147          CX*****
148          CX*
149          CX*      THE PURPOSE OF THIS SUBROUTINE
150          CX*      IS TO EQUATE WORD1 WITH WORD2.
151          CX*      IT IS A NECESSARY SUBROUTINE WHEN
152          CX*      USING A HONEYWELL COMPUTER SINCE NOT
153          CX*      ALL VARIABLES ARE KEPT AS ASCII. FOR
154          CX*      EXAMPLE, DATA STORED ON A DIRECT
155          CX*      ACCESS FILE IS BCD WHEREAS DATA READ
156          CX*      INTERACTIVELY IS ASCII.
157          CX*      SUCH IS NOT THE CASE WITH THE CDC
158          CX*      SO, SINCE THE PROGRAMS USE THIS
159          CX*      ROUTINE REGULARLY, THIS DUMMY ROUTINE
160          CX*      EXISTS RATHER THAN MAKING ALL CHANGES.*
161          CX*
162          CX***** C A U T I O N *****
163          CX*
164          CX*      WORD1 AND WORD2 MUST BE CHARACTER
165          CX*      TYPE. N IS A DUMMY ARGUMENT. IN THE
166          CX*      ORIGINAL VERSION, IT DETERMINED THE
167          CX*      NUMBER OF CHARACTERS TO BE CONVERTED.
168          CX*
169          CX*****
170          CX
171          CHARACTER *(*) WORD1, WORD2
172          WORD2 = WORD1
173          RETURN
174          END

```

```

175          SUBROUTINE BLNKOU( BUFFR )
176          CX
177          CX*****
178          CX*
179          CX*      THE PURPOSE OF THIS SUBROUTINE
180          CX*      IS TO "BLANK-OUT" A CHARACTER STRING.*
181          CX*
182          CX*****  C A U T I O N  *****
183          CX*
184          CX*      BUFFR MUST BE CHARACTER TYPE.
185          CX*
186          CX*****
187          CX
188          CHARACTER *(*) BUFFR
189          WRITE(BUFFR,'(A1)') ' '
190          RETURN
191          END

```

```

192          SUBROUTINE CALLSS( N )
193          CX*****
194          CX
195          CX  THIS SUBROUTINE IS USED BY THE HONEYWELL
196          CX  TO TRANSMIT JOB CONTROL LANGUAGE (JCL)
197          CX  FROM AN EXECUTING PROGRAM TO THE SYSTEM.
198          CX  THE FOLLOWING CODING IS JUST A DUMMY.
199          CX
200          CX*****
201          CX
202          CHARACTER *(*) N
203          RETURN
204          END

```

```

205          SUBROUTINE CONCAT(STR1,N1,STR2,N2,N3)
206          C
207          C*****C
208          C
209          CX  AUTHOR: ANTHONY WAISANEN
210          CX  CAPT      USAF
211          CX  DECEMBER 1981
212          C
213          C    THE PURPOSE OF THIS SUBROUTINE IS CONCATINATE
214          C    A NUMBER OF CHARACTERS (N3) OF A CHARACTER
215          C    STRING (STR1) BEGINNING WITH THE N1 CHARACTER
216          C    WITH N3 CHARACTERS OF ANOTHER CHARACTER STRING
217          C    (STR2) BEGINNING WITH ITS N2 CHARACTER.
218          C
219          C ***** C A U T I O N *****
220          C
221          C    N2 + N3 - 1 SHOULD NOT EXCEED THE NUMBER OF
222          C    CHARACTERS IS STR2 OR STRANGE THINGS WILL RESULT.
223          C
224          C*****C
225          C
226          CHARACTER * 1 STR1, STR2
227          LAST1 = N1 + N3 - 1
228          LAST2 = N2 + N3 - 1
229          C
230          STR1(N1:LAST1) = STR2(N2:LAST2)
231          C
232          RETURN
233          END

```

```

234          SUBROUTINE CREATE( LUD, ISIZE, N, ISTAT )
235          CX*****
236          CX* THE PURPOSE OF THIS SUBROUTINE IS *
237          CX* TO CREATE A LOCAL DIRECT ACCESS  *
238          CX* FILE. IT IS ESSENTIALLY THE EQUI- *
239          CX* VALENT OF THE HONEYWELL SUBROUTINE *
240          CX* OF THE SAME NAME.                  *
241          CX*****
242          CX
243              OPEN( UNIT=LUD, RECL=ISIZE, IOSTAT=ISTAT, ERR=10,
244                  & ACCESS='DIRECT' )
245              IF( ISTAT .NE. 0 ) THEN
246          10      PRINT*, ' TROUBLE IN CREATE WITH ISTAT =', ISTAT
247              ENDIF
248              RETURN
249              END

```

```

250          SUBROUTINE DATIM(DAYT,THYME)
251          C
252          C  **THIS ROUTINE RETURNS AN 8 CHARACTER DATE ('DAYT') FIELD AND A REAL
253          C  **VALUE OF TIME ('THYME') IN THE FORM OF 'MM/DD/YY' AND HH.XXXXXX
254          C  **(THYME IS A REAL VARIABLE).
255          CX  **AUTHOR: WAISANEN 2 DEC 81
256          CX  THIS SUBROUTINE IS COMPLETELY SYSTEM DEPENDENT
257          CX  BECAUSE OF THE WAY THE CURRENT DATE AND TIME
258          CX  ARE CALLED.  DEPENDING ON THE PARTICULAR SYSTEM
259          CX  BEING USED, THE CALLS TO "DATE" AND "TIME"
260          CX  SHOULD BE ALL THAT REQUIRE MODIFICATION.
261          C
262          CX  THIS SUBROUTINE HAS BEEN MODIFIED FOR THE
263          CX  CYBER/CDC MAINFRAME
264          CHARACTER*10 DATE, CLOCK, DUM, DAYT*8
265          REAL THYME
266          CX  FIRST, GET THE CURRENT DATE
267          DUM = DATE(0)
268          DAYT = DUM(2:9)
269          C  NOW FIND THE TIME
270          DUM = CLOCK(0)
271          C  THE CHARACTER STRING 'DUM' IS OF THE FORM:
272          C  'HH.MM.SS.
273          C  AND MUST BE CONVERTED TO THE REAL VARIABLE 'THYME'
274          THYME = FLOAT( ( ICHAR( DUM(2:2) ) - 16 ) * 10 +
275          &                ( ICHAR( DUM(3:3) ) - 16 ) ) +
276          &                FLOAT( ( ( ICHAR( DUM(5:5) ) - 16 ) * 10 +
277          &                ( ICHAR( DUM(6:6) ) - 16 ) ) * 60 +
278          &                ( ICHAR( DUM(8:8) ) - 16 ) * 10 +
279          &                ( ICHAR( DUM(9:9) ) - 16 ) ) / 3600.0
280          RETURN
281          END

```

```

282          SUBROUTINE DETACH( IFILE, ISTAT )
283          CX*****
284          CX*
285          CX* THIS SUBROUTINE IS THE FORTRAN 77
286          CX* EQUIVALENT OF THE HONEYWELL ROUTINE
287          CX* OF THE SAME NAME. UNLIKE THE HONEY-
288          CX* WELL ROUTINE, TAPE# IFILE IS NOT
289          CX* RETURNED BUT IS ONLY CLOSED.
290          CX* IF ISTAT .NE. 0, THEN AN ERROR HAS
291          CX* OCCURED AND TAPE# IFILE MAY NOT BE
292          CX* PROPERLY CLOSED.
293          CX*
294          CX***** C A U T I O N *****
295          CX*
296          CX* THERE ARE NO CAUTIONS WITH THIS
297          CX* ROUTINE.
298          CX*
299          CX*****
300          CLOSE( UNIT=IFILE, IOSTAT=ISTAT, ERR=10 )
301          CX ERR IS SPECIFIED TO FORCE THE PRINTING OF THE
302          CX ERROR STATEMENT.
303          10 IF( ISTAT .NE. 0 ) THEN
304             PRINT 12, ISTAT, IFILE
305             ENDIF
306             RETURN
307          12 FORMAT( ' DETACH ERROR # ',I3,' WITH TAPE ',I2)
308          END

```

```

309          SUBROUTINE FLD2( I, K, E, N )
310          CX*****
311          CX*
312          CX*      THE PURPOSE OF THIS SUBROUTINE
313          CX* IS TO SET K CHARACTERS OF WORD E TO # N*
314          CX* BEGINNING WITH POSITION (I+1). THIS IS*
315          CX* THE FORTRAN 77 EQUIVALENT TO SECOND
316          CX* FORM OF THE FUNCTION FLD AS DESCRIBED
317          CX* ON PAGE 6-7 OF THE HONEYWELL FORTRAN
318          CX* MANUAL.
319          CX*
320          CX***** C A U T I O N *****
321          CX*
322          CX* E MUST BE OF CHARACTER*36 TYPE,
323          CX* 0 .LE. I .LE. 35
324          CX* 1 .LE. K .LE. 36
325          CX*
326          CX*****
327          CX
328          CHARACTER *(*) E, NUM*1
329          CALL N2CH( NUM, N )
330          DO 15 J = 1, K
331             IPOS = J + I
332             E(IPOS:IPOS) = NUM
333          15 CONTINUE
334          RETURN
335          END

```

```

336      CX*#RUNH *#;FKADY09/D070/FKADYLB/OBJECT/ILLEGAL(NOGO)
337      CX      SUBROUTINE ILLEGAL(IPOS)
338      SUBROUTINE ILLEGA(IPOS)
339      C
340      C      WRITTEN BY CAPT. PIERCE.....5 DEC 1979
341      CX      MODIFIED BY CAPT WAISANEN....MAR 1982
342      C
343      CHARACTER*11 FMT
344      DATA FMT/'(XXX,--- --)'/'
345      C
346      IF(.NOT.(IPOS.LE.77))GO TO 10
347      C      THEN
348      CX      ENCODE(FMT,100)IPOS
349      WRITE(FMT,100)IPOS
350      WRITE(6,FMT)
351      GO TO 20
352      C      ELSE
353      10      CONTINUE
354      WRITE(6,200)IPOS
355      20      CONTINUE
356      C      ENDIF
357      CX100  FORMAT(T2,I2)
358      100  FORMAT('( ',I2,'X,--- --)' )
359      200  FORMAT(60X,'POSITION',I3,'->')
360      RETURN
361      END

```

```

362      CX  *#RUNH *#;FKADY01/SOURCE/TWXLIB/SLDSBUJ
363      CX  THE FOLLOWING SUBROUTINE HAS BEEN DETERMINED
364      CX  TO BE UNNECESSARY. THEREFORE, IT HAS BEEN DELETED AND
365      CX  REFERENCES TO IT HAVE BEEN MODIFIED TO 'MSORTY'.
366      CX
367      CX  SUBROUTINE MBSORTY(BLF,ITAC,SRATE,SFACT,ACSORT)
368      C
369      C  BLALOCK:9 DEC 77
370      C  CHITWOOD:6 APR 81
371      C  CALCULATE THE NORMAL SORTIES
372      C
373      CX  ACSORT=AIN(T(BLF*FLOAT(ITAC)*SRATE*SFACT)
374      CX  RETURN
375      CX  END
376      C  *#RUNH *#;FKADY01/CSTR/TWXLIB/CLDSGUJ(NOGO)
377      C  SUBROUTINE MSORTY(BLF,ITAC,SRATE,SFACT,ACSORT)
378      C
379      C  BLALOCK:9 DEC 77;CHITWOOD 6 APR 81
380      C  CALCULATE THE NORMAL SORTIES
381      C
382      C  ACSORT=AIN(T(BLF*FLOAT(ITAC)*SRATE*SFACT)
383      C  RETURN
384      C  END

```

```

385      SUBROUTINE N2CH( CHARA, I )
386      CX*****
387      CX*
388      CX*      THE PURPOSE OF THIS SUBROUTINE      *
389      CX*      IS TO CONVERT AN INTEGER INTO AN    *
390      CX*      EQUIVALENT CHARACTER STRING.  IT IS *
391      CX*      THE FORTRAN 77 EQUIVALENT OF AN     *
392      CX*      "ENCODE" STATEMENT.                *
393      CX*
394      CX*****  C A U T I O N  *****
395      CX*
396      CX*      THE INTEGER ARGUMENT MUST BE INTEGER *
397      CX*      AND LESS THAN 10**15.              *
398      CX*
399      CX*****
400      CX
401      CHARACTER *(*) CHARA
402      CHARA = ' '
403      INTGR = I
404      L = 1
405      NP10 = 1
406      CX IF I IS A ONE DIGIT NUMBER, SAVE CALCULATION
407      CX TIME BY SKIPPING.
408      IF( INTGR .LT. 10 ) GO TO 10
409      CX PERHAPS I IS A TWO DIGIT NUMBER?
410      L = 2
411      NP10 = 10
412      CX SKIP IF THE NUMBER HAS ONLY TWO DIGITS.
413      IF( INTGR .LT. 100 ) GO TO 10
414      CX CALCULATE THE NUMBER OF DIGITS (L-1) ONLY
415      CX IF THE NUMBER IS GREATER THAN 99.
416      L = IFIX( ALOG10( FLOAT( INTGR ) ) ) + 1
417      NP10 = 10 ** (L-1)
418      CX
419      CX NOW, PACK THE DIGITS INTO CHARA BEGINNING WITH
420      CX THE LEFT-MOST.
421      CX
422      10 DO 15 IPOS = 1, L
423      M = INTGR / NP10
424      CHARA(IPOS:IPOS) = CHAR( M + 16 )
425      INTGR = INTGR - M * NP10
426      NP10 = NP10 / 10
427      15 CONTINUE
428      RETURN
429      END

```

```

430          SUBROUTINE RANSIZ( IFILE, IRECSI, J )
431          CX*****
432          CX*
433          CX* THIS SUBROUTINE IS THE EQUIVALENT OF *
434          CX* A HONEYWELL FORTRAN ROUTINE OF THE *
435          CX* SAME NAME. ITS FUNCTION IS TO OPEN *
436          CX* A DIRECT ACCESS FILE (TAPE# IFILE) *
437          CX* WITH THE CORRECT NUMBER OF RECORDS *
438          CX* PER RECORD (IRECSI). J = 0 IMPLIES A *
439          CX* FORMATTED FORM OF FILE. J .NE. 0 IM- *
440          CX* PLIES AN UNFORMATTED, DIRECT ACCESS *
441          CX* FILE. *
442          CX*
443          CX***** C A U T I O N *****
444          CX* YOU HAD BETTER HAVE A LOCAL FILE BY *
445          CX* THE NAME OF "TAPE(IFILE)". *
446          CX*
447          CX*****
448          CX INITIALIZE ERROR INDICATOR ISTAT.
449          ISTAT = 0
450          CX BEFORE OPENNING A FILE, IT MUST BE CLOSED
451          CX BUT ONLY CLOSE THE FILE IF RECL > 1
452          IF( IRECSI .GT. 1 ) THEN
453          CX FILE HAS BEEN OPENPED BEFORE SO
454          CLOSE( UNIT=IFILE, IOSTAT=ISTAT, ERR=10, STATUS='KEEP' )
455          ENDIF
456          CX ERR IS SPECIFIED TO FORCE THE PRINTING OF THE ERROR STATEMENT.
457          10 IF( ISTAT .EQ. 0 ) THEN
458              IF( J .EQ. 0 ) THEN
459                  OPEN( UNIT=IFILE, IOSTAT=ISTAT, ERR=20, ACCESS='SEQUENTIAL',
460                      & FORM='FORMATTED' )
461              ELSE
462                  OPEN( UNIT=IFILE, IOSTAT=ISTAT, ERR=20, ACCESS='DIRECT',
463                      & FORM='UNFORMATTED', RECL=IRECSI )
464              ENDIF
465          20 IF( ISTAT .NE. 0 ) THEN
466              PRINT 22, ISTAT, IFILE
467          ENDIF
468          ELSE
469              PRINT 12, ISTAT, IFILE
470          ENDIF
471          RETURN
472          12 FORMAT(' CLOSE ERROR # ',I3,' WITH TAPE ',I2)
473          22 FORMAT(' OPENNING ERROR # ',I3,' WITH TAPE ',I2)
474          END

```

```

475          SUBROUTINE READBU( N, CHAR )
476          CX
477          CX*****
478          CX  AUTHOR: ANTHONY WAISANEN          C
479          CX          CAPT          USAF          C
480          CX
481          CX  THE PURPOSE OF THIS SUBROUTINE
482          CX  IS TO CHECK IF THE FIRST CHARACTER OF
483          CX  CHAR IS A '0'.  IF IT IS,
484          CX  N IS SET TO ZERO.  IF NOT, N IS
485          CX  SET TO 3.
486          CX*****
487          CX
488          CHARACTER *(*) CHAR
489          N = 3
490          IF( CHAR(1:1) .EQ. '0' ) N = 0
491          RETURN
492          END

```

```

493          SUBROUTINE SETB(ISET,ISETD,MIN,IST,LST,ISTAT,INC,ON)
494          C      ROUTINE TO SET BITS IN ISET CORRESPONDING TO NUMERIC INPUT.
495          C      YODIS.....13 JUL 78
496          C      PIERCE MODIFIED 5 DEC 79
497          CX     WAISANEN MODIFIED MAR L982
498          C
499          LOGICAL ON
500          CX     DIMENSION ISET(ISETD)
501          CHARACTER*36 ISET(*)
502          CX
503          C
504          JNUM=0
505          IF(ON)JNUM=1
506          IF(.NOT.(IST.GT.LST))GO TO 90
507          C      THEN FIX
508          M=IST
509          IST=LST
510          LST=M
511          90    CONTINUE
512          C      ENDIF
513          DO 110 NUM=IST,LST,INC
514          IBIT=NUM-MIN
515          JBIT=MOD(IBIT,36)
516          JWRD=IBIT/36+1
517          IF(JWRD.GT.ISETD)ISTAT=5
518          IF(.NOT.(ISTAT.EQ.99))GO TO 100
519          C      THEN SET BITS.
520          CX     FLD(JBIT,1,ISET(JWRD))=JNUM
521          CALL FLD2(JBIT,1,ISET(JWRD),JNUM)
522          100    CONTINUE
523          C      ENDIF
524          110    CONTINUE
525          C      ENDDO
526          INC=1
527          RETURN
528          END

```

```

529          SUBROUTINE SETBIT( IWORD, IBIT, ISW )
530          CX*****
531          CX*
532          CX*      THE PURPOSE OF THIS SUBROUTINE      *
533          CX* IS TO SET BIT # IBIT OF IWORD TO 1      *
534          CX* IF ISW = 1 AND BIT # IBIT OF IWORD TO*
535          CX* 0 IF ISW = 0.                          *
536          CX*      1 .LE. IBIT .LE. 36.              *
537          CX*
538          CX***** C A U T I O N *****
539          CX*
540          CX*      IWORD MUST BE 36 CHARACTERS LONG.  *
541          CX*      ISW = 0 OR 1 AND 0 < IBIT < 37    *
542          CX*
543          CX*****
544          CX
545          CHARACTER*36 IWORD
546          CX CHECK THE VALIDITY OF ISW AND IBIT
547          IF(ISW .EQ. 0 .OR. ISW .EQ. 1 .AND.
548          &      IBIT .GE. 1 .AND. IBIT .LE. 36) THEN
549          CX ISW AND IBIT IS VALID
550          CX SET CORRECT CHARACTER ('BIT') OF IWORD
551          IF( ISW .EQ. 1 ) IWORD(IBIT:IBIT) = '1'
552          IF( ISW .EQ. 0 ) IWORD(IBIT:IBIT) = '0'
553          ELSE
554          CX ERROR PROCESSING SECTION
555          PRINT *, ' ERROR IN SUBROUTINE SETBIT '
556          ENDIF
557          RETURN
558          END

```

```

559      C *#RUNH *#;FKADY09/DO70/FKADYLB/OBJECT/SEQIN(NOGO)
560      SUBROUTINE SEQIN(BUF1,IP,ISET,IDIM,MAX,MIN,ISTAT)
561      C ROUTINE TO DECODE A NUMERIC INPUT STRING OF THE FORM:
562      C '1,2,4-6,10,...' AND SET CORRESPONDING BITS OF ISET.
563      C YODIS.....14 JUL 78
564      C MODIFIED 14 DEC 78 BY: LT HARBICK
565      C BELLS AND WHISTLES ADDED 4 DEC 79 BY CAPT ROGER C PIERCE
566      CX IMPROVED TO WORK ON A CDC MARCH 1982 CAPT ANTHONY WAISANEN
567      C
568      CHARACTER BUF1*80
569      C
570      CHARACTER *15 CHARS
571      CX DIMENSION ISET(IDIM)
572      CHARACTER*36 ISET(*)
573      CX
574      LOGICAL DASH,ON,COMMA
575      C
576      C
577      CX
578      DATA CHARS/'0123456789-, AB'/
579      CX
580      ISETD=IDIM
581      DASH=.FALSE.
582      NUM=0
583      IBLANK=0
584      IOFFSE=IP-1
585      IP=IP-1
586      ISTAT=99
587      ON=.TRUE.
588      INC=1
589      COMMA=.TRUE.
590      C
591      C DOUNTIL (IP.GE.80 .OR. ISTAT.NE.99)
592      100 CONTINUE
593      IP=IP+1
594      ICHAR=16
595      DO 110 ICH=1,15
596      JCH=ICH
597      IF(KOMPCH(BUF1,IP,CHARS,JCH,1).EQ.0) ICHAR = JCH
598      110 CONTINUE
599      C ENDDO
600      IF(ICCHAR.GE.1 .AND. ICHAR.LE.10)ICTYPE=1
601      IF(ICCHAR.GT.10)ICTYPE=ICCHAR-9
602      C CASE ENTRY
603      GO TO (120,130,150,170,190,220,320)ICTYPE
604      C CASE 1. BUF1*IP IS A DIGIT.
605      120 CONTINUE
606      NUM=NUM*10+ICCHAR-1
607      COMMA=.FALSE.
608      GO TO 330

```

```

609      C      CASE 2. BUF1*IP IS A DASH.
610      130  CONTINUE
611          IF(NUM.LT.MIN)ISTAT=4
612          IF(NUM.GT.MAX)ISTAT=3
613          IF(COMMA)ISTAT=6
614          IF(KOMPCH(BUF1,IP+1,' ',1,1).EQ.0)ISTAT=6
615          IF(IP.EQ.1)ISTAT=6
616          IF(DASH)ISTAT=6
617          IF(.NOT.(ISTAT.EQ.99))GO TO 140
618      C      THEN SET 1ST DO PARAM AND DASH FLAG.
619          DASH=.TRUE.
620          ISTNUM=NUM
621          NUM=0
622      140  CONTINUE
623      C      ENDIF
624          COMMA=.FALSE.
625          GO TO 330
626      C      CASE 3. BUF1*IP IS A COMMA.
627      150  CONTINUE
628          IF(NUM.LT.MIN)ISTAT=4
629          IF(NUM.GT.MAX)ISTAT=3
630          IF(.NOT.(NUM.EQ.0.AND..NOT.DASH))GO TO 155
631      C      THEN ZERO VALUE TERMINATES STRING
632          ISTAT=0
633          LENIN=IP-1
634      155  CONTINUE
635      C      ENDIF
636          IF(KOMPCH(BUF1,IP+1,' ',1,1).EQ.0)ISTAT=6
637          IF(COMMA)ISTAT=6
638          IF(IP.EQ.1)ISTAT=6
639          IF(.NOT.(ISTAT.EQ.99))GO TO 160
640      C      THEN SET LAST DO PARAM AND SET BITS.
641          LSTNUM=NUM
642          NUM=0
643          IF(.NOT.DASH)ISTNUM=LSTNUM
644          DASH=.FALSE.
645          CALL SETB(ISET,ISETD,MIN,ISTNUM,LSTNUM,ISTAT,INC,ON)
646      160  CONTINUE
647      C      ENDIF
648          COMMA=.TRUE.
649          GO TO 330
650      C      CASE 4. BUF1*IP IS A SPACE.
651      170  CONTINUE
652          IBLANK=IBLANK+1
653          IF(.NOT.(IBLANK.EQ.(IP-IOFFSE)))GO TO 175
654      C      THEN CHECK TO SEE IF LEADING BLANK LIMIT EXCEEDED
655          IF(IBLANK.EQ.4)ISTAT=1
656          GO TO 177
657      C      ELSE PROCESS AS END OF STRING
658      175  CONTINUE

```

```

659             IF(NUM.LT.MIN)ISTAT=4
660             IF(NUM.GT.MAX)ISTAT=3
661             IF(NUM.EQ.0.AND..NOT.DASH)ISTAT=0
662             IF(.NOT.(ISTAT.EQ.99))GO TO 180
663     C          THEN SET LAST DO PARAM AND SET BITS.
664             LSTNUM=NUM
665             IF(.NOT.DASH)ISTNUM=LSTNUM
666             DASH=.FALSE.
667             CALL SETB(ISET,ISETD,MIN,ISTNUM,LSTNUM,ISTAT,INC,ON)
668             ISTAT=0
669     180        CONTINUE
670     C          ENDIF
671     177        CONTINUE
672     C          ENDIF
673             GO TO 330
674     C          CASE 5. BUF1*IP IS AN "A"
675     190        CONTINUE
676             IF(.NOT.(KOMPCH(BUF1,IP,"ALL",1,3).EQ.0))GO TO 200
677     C          THEN FILL ARRAY
678             ISTNUM=MIN
679             LSTNUM=MAX
680             CALL SETB(ISET,ISETD,MIN,ISTNUM,LSTNUM,ISTAT,INC,ON)
681             IP=IP+3
682             IF(KOMPCH(BUF1,IP,"",1,1).NE.0) ISTAT=6
683             IF(KOMPCH(BUF1,IP,"",1,1).EQ.0) ISTAT=0
684             GO TO 210
685     C          ELSE ILLEGAL CHARACTER
686     200        CONTINUE
687             ISTAT=2
688     210        CONTINUE
689     C          ENDIF
690             COMMA=.TRUE.
691             GO TO 330
692     C          CASE 6. BUF1*IP IS A "B"
693     220        CONTINUE
694             IF(.NOT.(KOMPCH(BUF1,IP,"BUT",1,4).EQ.0.AND.COMMA))GO TO 230
695     C          THEN TURN OFF ON
696             IF(.NOT.ON) ISTAT=8
697             ON=.FALSE.
698             IP=IP+3
699             GO TO 310
700     C          ELSE CHECK FOR BY
701     230        IF(.NOT.(KOMPCH(BUF1,IP,"BY",1,2).EQ.0.AND.COMMA))GO TO 290
702     C          THEN CHECK FOR INCREMENT
703             INC=11
704             DO 240 ICH=1,10
705                 JCH=ICH
706                 IF(KOMPCH(BUF1,IP+2,CHARS,JCH,1).EQ.0)INC=JCH-1
707     240        CONTINUE
708     C          ENDDO

```

```

709             IF(INC.EQ.0) INC=10
710             IF(.NOT.(INC.EQ.11))GO TO 250
711 C             THEN ERROR
712             ISTAT=7
713             IP=IP+2
714             GO TO 280
715 C             ELSE CHECK FOR COMMA
716             250 CONTINUE
717             IF(.NOT.(KOMPCH(BUF1,IP+3,',',1,1).EQ.0))GO TO 260
718 C             THEN OK
719             IP=IP+3
720             GO TO 270
721 C             ELSE DELIMETER ERROR
722             260 CONTINUE
723             ISTAT=6
724             270 CONTINUE
725 C             ENDIF
726             280 CONTINUE
727 C             ENDIF
728             GO TO 300
729 C             ELSE ILLEGAL CHARACTER
730             290 CONTINUE
731             ISTAT=2
732             IF(.NOT.COMMA)ISTAT=6
733             300 CONTINUE
734 C             ENDIF
735             310 CONTINUE
736 C             ENDIF
737             COMMA=.TRUE.
738             GO TO 330
739 C             CASE 7. BUF1*IP IS AN ILLEGAL CHAR.
740             320 CONTINUE
741             ISTAT=2
742             330 CONTINUE
743 C             ENDCASE.
744             IF(.NOT.(IP.GE.80 .OR. ISTAT.NE.99))GO TO 100
745 C             ENDDO
746             IF(ISTAT.EQ.4 .OR. ISTAT.EQ.3) IP=IP-1
747             RETURN
748             END

```

```

749          SUBROUTINE SLITE( LITE )
750          CX*****
751          CX*
752          CX*      THE PURPOSE OF THIS SUBROUTINE
753          CX* IS TO TURN ON A "SENSE LIGHT" WHICH IS
754          CX* USED AS A FLAG IN THE MISSION INPUT
755          CX* PROGRAMS TO INDICATE THE PRESENCE OF
756          CX* ERRORS.
757          CX* IF THE VALUE OF LITE IS 0, THEN ALL
758          CX* SENSE LIGHTS ARE TO BE TURNED OFF.
759          CX* IF THE VALUE OF LITE IS NOT 0, THEN
760          CX* SENSE LIGHT # LITE IS TURNED ON.
761          CX* THIS SUBROUTINE IS INTENDED TO
762          CX* REPLACE SUBROUTINE SLITE AS DESCRIBED
763          CX* ON PAGE 6-46 OF THE HONEYWELL FORTRAN
764          CX* MANUAL.
765          CX*
766          CX***** C A U T I O N *****
767          CX*
768          CX* THE VALUE OF LITE MUST LIE BETWEEN
769          CX* 1 AND 36 (THE NUMBER OF CHARACTERS IN
770          CX* THE SENSE LIGHT WORD, LIGHT).
771          CX*
772          CX*****
773          CX
774          CX DEFINE THE SENSE LIGHT WORD, LIGHT
775          COMMON /SENSE/LIGHT
776          CHARACTER*36 LIGHT
777          CX CHECK THE VALIDITY OF LITE
778          IF( LITE.GE.1 .AND. LITE.LE.35 ) THEN
779          CX LITE IS VALID SO OPERATE ACCORDINGLY
780          IF( LITE .LE. 0 ) THEN
781          CX      TURN OFF ALL SENSE LIGHTS
782          LIGHT = '0'
783          ELSE
784          CX      TURN ON LIGHT # LITE
785          LIGHT(LITE:LITE) = '1'
786          ENDIF
787          ELSE
788          CX ERROR HAS OCCURRED
789          PRINT 12,LITE
790          ENDIF
791          RETURN
792          12  FORMAT(1X,' ERROR IN SUBROUTINE SLITE. CALLED',
793          &' WITH LITE = ',I3)
794          END

```

```

795          SUBROUTINE SLITET( LITE, K )
796          CX*****
797          CX*      THE PURPOSE OF THIS SUBROUTINE      *
798          CX* IS TO SENSE IF A "SENSE LIGHT" IS ON. *
799          CX* IF THE VALUE OF LITE IS 0, THEN AN      *
800          CX* ERROR CONDITION EXISTS. K IS SET TO 0*
801          CX* AND AN ERROR MESSAGE IS PRINTED.      *
802          CX* IF THE VALUE OF LITE IS NOT 0, THEN *
803          CX* SENSE LIGHT # LITE IS CHECKED. IF IT *
804          CX* WAS ON (=1), K = 1. IF IT WAS OFF *
805          CX* (=0), K = 2. THE LIGHT IS TURNED OFF *
806          CX* AND CONTROL RETURNS TO THE CALLING *
807          CX* ROUTINE.                                *
808          CX* THIS SUBROUTINE IS INTENDED TO      *
809          CX* REPLACE SUBROUTINE SLITET AS DESCRIBED*
810          CX* ON PAGE 6-46 OF THE HONEYWELL FORTRAN *
811          CX* MANUAL.                                *
812          CX***** C A U T I O N *****
813          CX* THE VALUE OF LITE MUST LIE BETWEEN *
814          CX* 1 AND 36 (THE NUMBER OF CHARACTERS IN *
815          CX* THE SENSE LIGHT WORD, LIGHT).      *
816          CX*****
817          CX
818          CX DEFINE THE SENSE LIGHT WORD, LIGHT
819              COMMON /SENSE/LIGHT
820              CHARACTER*36 LIGHT
821          CX CHECK THE VALIDITY OF LITE
822              IF( LITE.GE.1 .AND. LITE.LE.35 ) THEN
823          CX LITE IS VALID SO OPERATE ACCORDINGLY
824          CX DEFINE A DEFAULT VALUE FOR K ("OFF" VALUE)
825              K = 2
826              IF( LIGHT(LITE:LITE) .EQ. '1' ) THEN
827          CX              LIGHT WAS ON
828                  K = 1
829          CX              NOW TURN IT OFF
830                  LIGHT(LITE:LITE) = '0'
831          CX              ELSE
832          CX              LIGHT WAS OFF SO USE DEFAULT VALUES.
833                  ENDIF
834              ELSE
835          CX ERROR HAS OCCURRED
836                  PRINT 12,LITE
837                  K = 0
838                  ENDIF
839                  RETURN
840          12  FORMAT(1X,' ERROR IN SUBROUTINE SLITET. CALLED',
841                  &' WITH LITE = ',I3)
842          END

```

```

843      C  *#RUNH *#;FKADY01/CSTR/TWXLIB/CLDBLUJ(NOGO)
844          SUBROUTINE TBLF(NAC,LOPT,BSTAT,TWXBLF)
845      C
846      C  BLALOCK: 17 APRIL 79
847      C  COMPUTE SORTIE SCALE FACTOR FROM BASE STAT AND LOAD FACTOR
848      C
849          X=FLOAT(NAC)/(FLOAT(LOPT)+1.0E-20)
850          X1=AMAX1(0.0,X-BSTAT)
851          C1=0.5*X1
852          C2=1.5*X1
853          TWXBLF=(AMIN1(X,BSTAT)+C1*EXP(-C2))/AMAX1(1.0E-20,X)
854      RETURN
855      CX  END
856      CX
857      CX  THE FOLLOWING SUBROUTINE HAS BEEN DETERMINED
858      CX  TO BE UNNECESSARY. THEREFORE, IT HAS BEEN DELETED AND
859      CX  REFERENCES TO IT HAVE BEEN MODIFIED TO 'TBLF'.
860      C  *#RUNH *#;FKADY01/CSTR/TWXLIB/CLDBBUJ(NOGO,BCD)
861      CX  SUBROUTINE TBBLF(NAC,LOPT,BSTAT,TWXBLF)
862      C
863      C  BLALOCK: 17 APRIL 79
864      C  COMPUTE SORTIE SCALE FACTOR FROM BASE
865      C  STAT AND LOAD FACTOR
866      C
867      CX  X=FLOAT(NAC)/(FLOAT(LOPT)+1.0E-20)
868      CX  X1=AMAX1(0.0,X-BSTAT)
869      CX  C1=0.5*X1
870      CX  C2=1.5*X1
871      CX  TWXBLF=(AMIN1(X,BSTAT)+C1*EXP(-C2))/AMAX1(1.0E-20,X)
872      CX  RETURN
873      CX  END

```

```

874 C  *#RUNH *:=;FKADY01/CSTR/TWXLIB/CLDDBUJ(NOGO,BCD)
875     SUBROUTINE TDATIM(DATETI)
876 C
877 C  **THIS ROUTINE RETURNS A 16 CHARACTER DATE TIME FIELD FOR
878 C  **PRINTOUT OF THE FORM 'YY MMM DD HH:MM'
879 C  **AUTHOR: ABBOTT     DATE: 12 DEC 77
880 CX **MODIFIER: WAISANEN 2 DEC 81
881 C
882 CX THIS SUBROUTINE HAS BEEN MODIFIED FOR THE
883 CX CYBER/CDC MAINFRAME
884     CHARACTER DATETI*16,DATE*8,MON(12)*3,HOURL*2,MIN*2
885     DATA MON/'JAN','FEB','MAR','APR','MAY','JUN','JUL','AUG','SEP',
886     &      'OCT','NOV','DEC'/
887 C
888     CALL DATIM(DATE,TIME)
889 CX
890 CX ORIGINAL CODING
891 CX     DECODE(DATE,500)MONTH
892 CX EQUIVALENT FORTRAN 77 CODING
893     READ(DATE,500)MONTH
894 CX
895     DATETI='
896     CALL CONCAT(DATETI,1,DATE,7,2)
897     CALL CONCAT(DATETI,4,MON(MONTH),1,3)
898     CALL CONCAT(DATETI,8,DATE,4,2)
899 CX
900     IH=TIME
901     IM=(TIME-FLOAT(IH))*60.0
902 CX
903 CX ORIGINAL CODING
904 CX     ENCODE(HOURL,500)IH
905 CX     ENCODE(MIN,502)IM/10
906 CX     ENCODE(MIN,501)IM-IM/10*10
907 CX EQUIVALENT FORTRAN 77 CODING
908 CX
909     WRITE(HOURL,500)IH
910     WRITE(MIN,502)IM/10
911     WRITE(MIN,501)IM-IM/10*10
912 CX
913     CALL CONCAT(DATETI,12,HOURL,1,2)
914     CALL CONCAT(DATETI,14,':',1,1)
915     CALL CONCAT(DATETI,15,MIN,1,2)
916 CX
917     RETURN
918     500 FORMAT(I2)
919     501 FORMAT(T2,I1)
920     502 FORMAT(T1,I1)
921 CX     END
922 CX
923 C  *#RUNH *:=;FKADY01/CSTR/TWXLIB/CLDDTUJ(NOGO)

```

```

924      CX
925      CX THE FOLLOWING SUBROUTINE HAS BEEN DETERMINED
926      CX TO BE UNNECESSARY. THEREFORE, IT HAS BEEN DELETED AND
927      CX REFERENCES TO IT HAVE BEEN MODIFIED TO 'TDATIM'.
928      CX SUBROUTINE TB DATIM(DATETIM)
929      C
930      C **THIS ROUTINE RETURNS A 16 CHARACTER DATE TIME FIELD FOR
931      C **PRINTOUT OF THE FORM 'YY MMM DD HH:MM'
932      C **AUTHOR: ABBOTT DATE: 12 DEC 77
933      C
934      CX CHARACTER DATETIM*16,DATE*8,MON*3(12),HOUR*2,MIN*2
935      CX DATA MON/'JAN','FEB','MAR','APR','MAY','JUN','JUL','AUG','SEP',
936      CX & 'OCT','NOV','DEC'/
937      C
938      CX CALL DATIM( DATE, TIME)
939      CX DECODE( DATE, 500) MONTH
940      CX DATETIM= ' '
941      CX CALL CONCAT( DATETIM, 1, DATE, 7, 2)
942      CX CALL CONCAT( DATETIM, 4, MON( MONTH), 1, 3)
943      CX CALL CONCAT( DATETIM, 8, DATE, 4, 2)
944      CX IH= TIME
945      CX IM=( TIME- FLOAT( IH)) * 60.0
946      CX ENCODE( HOUR, 500) IH
947      CX ENCODE( MIN, 502) IM/ 10
948      CX ENCODE( MIN, 501) IM+ IM/ 10* 10
949      CX CALL CONCAT( DATETIM, 12, HOUR, 1, 2)
950      CX CALL CONCAT( DATETIM, 14, ':', 1, 1)
951      CX CALL CONCAT( DATETIM, 15, MIN, 1, 2)
952      CX RETURN
953      CX500 FORMAT( I2)
954      CX501 FORMAT( T2, I1)
955      CX502 FORMAT( T1, I1)
956      END

```

```

957          SUBROUTINE TERMNO( CHAR )
958          CX*****
959          CX  AUTHOR: ANTHONY WAISANEN          C
960          CX      CAPT      USAF              C
961          CX
962          CX  THE PURPOSE OF THIS SUBROUTINE
963          CX  ORIGINALLY WAS TO RETURN THE 2 CHARACTER
964          CX  "NAME" OF THE TERMINAL BEING USED.
965          CX  THE ORIGINAL SUBROUTINE WAS A
966          CX  HONEYWELL SYSTEM ROUTINE.  THERE IS
967          CX  NO ANSI STANDARD ROUTINE.
968          CX  THIS, THEN, IS A DUMMY.
969          CX*****
970          CX
971          CHARACTER*2 CHAR
972          CHAR = '??'
973          RETURN
974          END

```

```

975 CX *#RUNH *=;FKADY01/CSTR/TWXLIB/CLDLTUJ(NOGO,BCD)
976 CX SUBROUTINE TLETTER(LINES,LETTERS,String,IFC)
977 SUBROUTINE TLETTE(LINES,LETTER,String,IFC)
978 C
979 C BLALOCK: MODIFIED 10 DEC 77
980 C NUMERIC - CHARACTER CONVERSION & PRINT
981 C
982 C /
983 IMPLICIT INTEGER(A-Z)
984 DIMENSION MAP(100),STRING(18,LINES)
985 CHARACTER*1 X,INDX(32)
986 CHARACTER*7 SYMB(100)
987 CHARACTER*5 BINY(32),PRNT(7,18)
988 DATA INDX/'O','1','2','3','4','5','6','7','8','9','A','B','C',
989 & 'D','E','F','G','H','I','J','K','L','M','N','O','P',
990 & 'Q','R','S','T','U','V'/
991 DATA BINY/' ','*',**,' ','*',**,' ','*',**,' ','*',**,' ',
992 & '***',' ','*',**,' ','*',**,' ','*',**,' ','*',**,' ',
993 & '***','*****',' ','*',**,' ','*',**,' ','*',**,' ','*',**,' ',
994 & '**','****','****','****','**','*','**','*','**','**','**',
995 & '***','*','****','*****','*****'/
996 DATA SYMB/'EHPLJHE','464444E','EHGE11V','EHGC6GH','8CA9V88',
997 & 'VF6GGHE','C21FHHE','VG8422Z','EHHEHHE','EHHUG86','EHHVVHH',
998 & 'FIIEIIF','EH111HE','FIIIIIIF','V11F11V','V11F111','U11THHE',
999 & 'HHHVHHH','E44444E','GGGGGHE','H95359H','111111V','HRLLHHH',
1000 & 'HJLPHHH','VHHHHHV','FHHF111','EHHL9M','FHHF59H','EH248HE',
1001 & 'V444444','HHHHHHE','HHHA444','HHHLLLA','HHA4AHH','HHA4444',
1002 & 'VG8421V','OARORAO','EHGMLD3','2552L9M','OOVOVOO','04EVE40',
1003 & '3J842PO','4AH0000','EH84404','044V440','01248G0','OG84210',
1004 & '00OV000','0000066','0006642','6606642','0660660','7444447',
1005 & 'S44444S','G84248G','1248421','1244421','G84448G','8420000',
1006 & '6666066','4444444','VVVVVVV','0000000','AA00000','EL5EKLE',
1007 & 'SOM9996','EHHE4E4','8088896','1195359','4444444','00ALLLL',
1008 & 'OODJHHH','OOEHHE','OOFHF11','OOE9E80','OODJ111','OOE1687',
1009 & '22F22IC','009999M','OOHHLLA','OOHA4AH','OOHHUGE','OOHHHA4',
1010 & 'OOV842V','006999M','11DJHJD','OOEH1HE','GGMPHPM','OOEHV1E',
1011 & '04U4444','OOE9E86','11DJHHH','4044444','000000V','002V200',
1012 & '8442448','2448442','00QDO00','2*' '/'
1013 DATA MAP/ 60, 61, 62, 63, 64, 65, 66, 67, 70, 71,101,102,103,104,
1014 & 105, 106,107,110,111,112,113,114,115,116,117,120,121,122,123,124,
1015 & 125,126,127,130,131,132, 43,100, 46, 75, 52, 45,136, 77, 53,
1016 & 134, 57, 55, 56, 54, 73, 72,135,133, 74, 76, 51, 50, 47, 41,
1017 & 174,200, 40, 42, 44,201,202,'52,153,154,155,156,157,160,161,
1018 & 162,163,164,165,167,170,171,166,172,141,142,143,144,145,146,
1019 & 147,150,151,137,140,173,175,176, 2*/
1020 DATA X/'/',DIFAUL/63/,D2FAUL/1/
1021 C
1022 DO 400 L=1,LINES
1023 DO 300 N=1,LETTER
1024 SEED=DIFAUL

```

```

1025      DO 100 K=1,100
1026      IF(STRING(N,L).EQ.MAP(K)) SEED=K
1027      100 CONTINUE
1028      DO 200 H=1,7
1029      HX=H
1030      CALL CONCAT(X,1,SYMB(SEED),HX,1)
1031      MX=D2FAUL
1032      DO 150 M=1,32
1033      IF(X.EQ.INDX(M)) MX=M
1034      150 CONTINUE
1035      PRNT(H,N)=BINY(MX)
1036      200 CONTINUE
1037      300 CONTINUE
1038      DO 350 H=1,7
1039      WRITE(IFC,600) (PRNT(H,N),N=1,LETTER)
1040      350 CONTINUE
1041      WRITE(IFC,601)
1042      400 CONTINUE
1043      RETURN
1044      600 FORMAT(1X,18(A5,2X))
1045      601 FORMAT(//)
1046      END

```

```

1047          SUBROUTINE UPRCAS( N, M )
1048          CX*****
1049          CX*
1050          CX* THIS IS A DUMMY SUBROUTINE. IN THE *
1051          CX* HONEYWELL SYSTEM, IT CONVERTS A LOWER *
1052          CX* CASE STRING OF M CHARACTERS, N, INTO *
1053          CX* AN EQUIVALENT UPPERCASE STRING, ALSO *
1054          CX* N. SINCE ASCII ONLY USES UPPERCASE, *
1055          CX* THIS SUBROUTINE HAS NO FUNCTION OTHER *
1056          CX* THAN TO INDICATE WHERE THE ORIGINAL *
1057          CX* TWX DATA HAD TO BE CONVERTED. *
1058          CX*
1059          CX***** C A U T I O N *****
1060          CX*
1061          CX* THERE ARE NO CAUTIONS ASSOCIATED WITH*
1062          CX* THIS SUBROUTINE. *
1063          CX*
1064          CX*****
1065          CHARACTER *(*) N
1066          RETURN
1067          END

```

Appendix C

Segmentation Procedure File Listings

```

1      .PROC,APESEG,FYLE=APE.
2      .* THIS PROCEDURE FILE CREATES A SEGMENTED OBJECT
3      .* DECK AND CATALOGS IT AS "APELGO."
4      .* THE SOURCE DECK "FYLE" MUST BE THE SOURCE DECK
5      .* FOR THE AAFCE PLANNING PROGRAM.
6      MAP,OFF.
7      RETURN,KNUJ.
8      ATTACH,KNUJ,APELGO,ID=T800855,PW=FOX.
9      .* SKIP IF NOT CATALOGED
10     SKIP,NONE.
11     EXIT,S.
12     ENDIF,NONE.
13     .* CONTINUE FROM HERE
14     REWIND,FYLE.
15     FTN5,I=FYLE,LO=0,B=APEB.
16     .* CHECK FOR TWXLIB PRESENCE
17     IFE,FILE(ZZZZZLW,LO.OR.PF).NE.1,GET1.
18     .* NOT HERE, SO MUST GET IT
19     ATTACH,ZZZZZLW,TWXLIB,ID=T800855,CY=1.
20     ENDIF,GET1.
21     .* DECLARE IT AND THE UEDIT LIBRARY
22     LIBRARY(ZZZZZLW,ZZZZZLA)
23     RETURN,SEGLGO.
24     REQUEST,SEGLGO,*PF.
25     SEGLOAD(I=DIRS,B=SEGLGO)
26     LOAD(APEB)
27     NOGO.
28     .* RECALL THE LIBRARIES
29     LIBRARY(ZZZZZLA,ZZZZZLB)
30     CATALOG,SEGLGO,APELGO,ID=T800855,XR=FOX,PW=FOX.
31     PURGE,KNUJ.
32     RENAME,SEGLGO,CY=1.
33     RETURN,KNUJ,SEGLGO.
34     RETURN,DIRS,APEB.
35     .* NO MATTER WHAT
36     EXIT,S.
37     LIBRARY(ZZZZZLA,ZZZZZLB)
38     RETURN,DIRS,APEB.
39     REVERT.

```

```

40      .DATA,DIRS.
41      *
42      * MAJOR TREE APE
43          TREE      APE-(IPT,OPTT,RAPUP)
44          INCLUDE    UTLLIN,BLKDAT
45      *
46      * SUBTREE IPT (FILE INITIALIZATION)
47      IPT  TREE      PI-(PIRANS,PIDS-PIDAYS,PIAC,PIAB,PINU,PILREC)
48      PI   INCLUDE    UTBLF
49      *
50      * SUBTREE OPTT ( OPTION HANDLING ROUTINES )
51      OPTT TREE      OPTN-(DPT,IPRC1,IPAG1,IPRS1,IPIV1,LAT)
52      OPTN INCLUDE    UTIMEO,UTLLIN,UTMLEF
53      *
54      * SUB-SUBTREE DPT ( DISPLAY ROUTINES )
55      DPT  TREE      DP-(DP12,DPAG,DPTS-DPTSOT,DP13,DPDR)
56      DP   INCLUDE    DPACIN,DPSETB,DPABNO
57      *
58      * SUBTREE OF DPT
59      DP12 TREE      DPBD-(DP2,DPBDOT)
60      DP2  TREE      DPACIN-DPACNA
61      DPACIN INCLUDE    DPSETB
62      *
63      * ANOTHER SUBTREE OF DPT
64      DP13 TREE      DPBL-(DP3,DPBLOT)
65      DP3  TREE      DPLGIN-DPLGNA
66      DPLGIN INCLUDE    DPSETB
67      *
68      * SUB-SUBTREE IPRC1
69      IPRC1 TREE      IPRC-(IPRCED-IPRCHC,IPRCUP)
70      IPRC INCLUDE    IPCHEC,UTNUM,UTIMEO,UTLLIN,UTMLEF
71      *
72      * SUB-SUBTREE IPAG1
73      IPAG1 TREE      IPAG-(IPAGED,IPAGUP)
74      IPAG INCLUDE    UTBLF,UTNUM,UTIMEO,UTLLIN,UTMLEF,IPCHEC
75      *
76      * SUB-SUBTREE IPRS1
77      IPRS1 TREE      IPRS-(IPRSED,IPRSUP)
78      IPRS INCLUDE    IPCHEC,UTNUM,UTIMEO,UTLLIN,UTMLEF
79      *
80      * SUB-SUBTREE IPIV1
81      IPIV1 TREE      IPIV-(IPIVED,IPIVCR,IPIVUP)
82      IPIV INCLUDE    IPCHEC,UTNUM,UTIMEO,UTLLIN,UTMLEF
83      *
84      * SUB-SUBTREE LAT
85      LAT  TREE      LA-(LAATAF,LA12,LA22)
86      LA   INCLUDE    LAINMS,LAWXSC,LAOTCA
87      *
88      * SUBTREE OF LAT
89      LA12 TREE      LAIN-(LANXQM,LA1)

```

```

90      *
91      *      SUBTREE OF LA12
92      LA1  TREE      LAINED-(LAINAC,LAINSR)
93      *
94      *      SUBTREE OF LA1
95      LA22 TREE      LAOT-(LAOTGR,LA2,LAOTPR)
96      *
97      *      SUBTREE OF LA22
98      LA2  TREE      LAOTAB-LAOTNO
99      LAOTAB INCLUDE  LAOTSE
100     *
101     * COMMON BLOCK DECLARATION
102     COMMON      ABDATA,ABLF,ACDATA,CTRL,IO,IOCH,
103     ,LAINB,LAINBC,LAIOBU,LAOTG,LAOTMU,LGDATA,LSTGD,MSNS,
104     ,NAMES,POLSPR
105     *
106     * APE DECLARED TO BE ENTRY POINT
107     END          APE

```

```

108      .PROC,LISEG,FYLE=LI.
109      RETURN,KNUJ.
110      ATTACH,KNUJ,LILGO,ID=T800855,PW=FOX.
111      SKIP,WHERE.
112      EXIT,S.
113      ENDIF,WHERE.
114      REWIND,FYLE.
115      FTN5,I=FYLE,B=LIB,LO=0,DB=0.
116      .* CHECK FOR TWXLIB PRESENCE
117      IFE,FILE(ZZZZZLW,LO.OR.PF).NE.1,GET1.
118      .* NOT HERE, SO MUST GET IT
119      ATTACH,ZZZZZLW,TWXLIB,ID=T800855,CY=1.
120      ENDIF,GET1.
121      .* DECLARE IT AND THE UEDIT LIBRARY
122      LIBRARY(ZZZZZLW,ZZZZZLA)
123      RETURN,SEGLGO.
124      REQUEST,SEGLGO,*PF.
125      SEGLOAD(I=DIRS,B=SEGLGO)
126      LOAD(LIB)
127      NOGO.
128      CATALOG,SEGLGO,LILGO,ID=T800855,PW=FOX,XR=FOX.
129      PURGE,KNUJ.
130      SKIP,NO.
131      EXIT,S.
132      ENDIF,NO.
133      RENAME,SEGLGO,CY=1.
134      RETURN,DIRS,XECUTE,KNUJ,SEGLGO.
135      .* RECALL THE LIBRARIES
136      LIBRARY(ZZZZZLA,ZZZZZLB)
137      .* NO MATTER WHAT
138      EXIT,S.
139      RETURN,DIRS,XECUTE.
140      LIBRARY(ZZZZZLA,ZZZZZLB)
141      REVERT.

```

```

142      .DATA,DIRS.
143          TREE      LI-(LINKA,OPTREE,WURAP)
144      *
145      LI      INCLUDE  BLKDAT
146      *
147      LINKA  TREE      PI-(PIAC,PIATCH,PILI,
148      ,PILO-(PILOPS-PILOCO),
149      ,PILOSE,PILOSI,PINIT,PIARST)
150      *
151      PINIT  INCLUDE  DATIM,FLD2,FP,RANSIZ,UTINAC,UTINCH,
152      ,UTINDA,UTINOK,UTINTG,UTINUT
153      *
154      OPTREE  TREE      OPTN-(LINKB,LINKC,LINKD,LINKE)
155      *
156      OPTN    INCLUDE  UTHELP
157      *
158      LINKB  TREE      DISPLA-(DISPDA,DISPPR,DISPUN-(DISPWX-DISPSV),
159      ,DISPMI,DISPAC,
160      ,DISPTG-(DISPTU,DISPTC,DISPTQ),
161      ,DISPAM,DISPAD,DISPRE)
162      *
163      DISPLA  INCLUDE  UTHELP
164      *
165      DISPAC  INCLUDE  CONCAT,DATIM,DISPWA,UTACVD,UTTOKE
166      *
167      DISPAD  INCLUDE  DATIM,DISPWA
168      *
169      DISPAM  INCLUDE  DATIM,DISPWA,UTHELP
170      *
171      DISPDA  INCLUDE  DATIM,DISPWL,UTHELP
172      *
173      DISPMI  INCLUDE  DATIM,DISPWL,UTHELP
174      *
175      DISPRE  INCLUDE  CONCAT,DATIM,DISPWA,DISPWL,FLD,
176      ,FLD2,ILLEGA,SEQIN,UTHASH,UTHELP
177      *
178      DISPTG  INCLUDE  CHKBIT,CONCAT,DATIM,DISPWA
179      *
180      DISPTC  INCLUDE  CONCAT,ILLEGA,SEQIN
181      *
182      DISPTQ  INCLUDE  CONCAT,ILLEGA,UTCoin,UTTOKE
183      *
184      DISPTU  INCLUDE  CONCAT,ILLEGA,SEQIN
185      *
186      DISPWX  INCLUDE  DISPWL
187      *
188      LINKC  TREE      MODIFY-(MODEL-MODDCH,CHGTREE)
189      *
190      MODIFY  INCLUDE  ILLEGA
191      MODEL   INCLUDE  FLD2

```

```

192      *
193      *
194      CHGTREE TREE      MODCHG-(MODWT1,MODWT2,MODWT3,
195      ,MODACF,MODCOR,MODCYC,MODDAY,MODPRI,MODQNT,
196      ,MODTGT,MODTUT,MODFUT,MODURQ,MODLLQ)
197      *
198      MODACF  INCLUDE    CONCAT,UTACVD,UTTOKE
199      *
200      MODCOR  INCLUDE    CONCAT,UTCOIN,UTTOKE
201      *
202      MODCYC  INCLUDE    UTDAVD
203      *
204      MODDAY  INCLUDE    CONCAT,UTCOIN,UTDAVD,UTTOKE
205      *
206      MODPRI  INCLUDE    CONCAT,UTCOIN,UTTOKE
207      *
208      MODQNT  INCLUDE    UTQUAN
209      *
210      MODTGT  INCLUDE    CONCAT,FLD,UTCOIN,UTTOKE
211      *
212      MODTUT  INCLUDE    CONCAT,UTCOIN,UTERR,UTTOKE
213      *
214      MODFUT  INCLUDE    CONCAT,UTCOIN,UTDAVD,UTERR,UTTGVD,UTTOKE
215      *
216      MODURQ  INCLUDE    CONCAT,UTCOIN,UTTOKE
217      *
218      MODLLQ  INCLUDE    CONCAT,UTCOIN,UTTOKE
219      *
220      LINKD   TREE      LANDIN-(LAATK,LADEF,LALAY,LAMOV,LAREI,LAWIT)
221      *
222      LAATK   INCLUDE    CONCAT,FLD,KOMPCH,LATGVD,LAUTCD,
223      ,UTABRT,UTCHK,UTCOIN,UTDAVD,UTDEC,UTERR,UTHASH,UTHELP,
224      ,UTINCH,UTTGVD,UTTOKE
225      *
226      LADEF   INCLUDE    CONCAT,FLD,KOMPCH,LATGVD,LAUTCD,
227      ,UTABRT,UTCHK,UTCOIN,UTDAVD,UTDEC,UTERR,UTHASH,UTHELP,
228      ,UTINCH,UTTGVD,UTTOKE
229      *
230      LALAY   INCLUDE    CONCAT,FLD,KOMPCH,LATGVD,LAUTCD,
231      ,UTABRT,UTCHK,UTCOIN,UTDAVD,UTDEC,UTERR,UTHASH,UTHELP,
232      ,UTINCH,UTQUAN,UTTGVD,UTTOKE
233      *
234      LAMOV   INCLUDE    CONCAT,FLD,KOMPCH,LATGVD,LAUTCD,
235      ,UTABRT,UTCHK,UTCOIN,UTDAVD,UTDEC,UTERR,UTHASH,UTHELP,
236      ,UTINCH,UTTGVD,UTTOKE
237      *
238      LAREI   INCLUDE    CONCAT,FLD,KOMPCH,LATGVD,LAUTCD,
239      ,UTABRT,UTCHK,UTCOIN,UTDAVD,UTDEC,UTERR,UTHASH,UTHELP,
240      ,UTINCH,UTQUAN,UTTGVD,UTTOKE
241      *

```

```

242      LAWIT   INCLUDE   CONCAT,FLD,KOMPCH,LATGVD,LAUTCD,
243      ,UTABRT,UTCHK,UTCoin,UTDAVD,UTDEC,UTERR,UTHASH,UTHELP,
244      ,UTINCH,UTQUAN,UTTGVD,UTTOKE
245      *
246      LINKE   TREE     AIRIN-(AIRPRI,AIRRES,AIRREC,AIRBAI,AIRCAS,AICYVD)
247      AIRIN   INCLUDE   UTHelp
248      *
249      AIRPRI  INCLUDE   CHKBIT,CONCAT,ILLEGA,SEQIN,UTHELP
250      *
251      AIRRES  INCLUDE   AICOVD,AIPIVD,CONCAT,UTABRT,UTACVD,UTCoin,
252      ,UTHASH,UTINCH,UTQUAN,UTTOKE
253      *
254      AIRREC  INCLUDE   AIQDVD,AIPIVD,AITGVD,CONCAT,FLD,UTABRT,
255      ,UTACVD,UTCoin,UTHASH,UTINCH,UTTOKE
256      *
257      AIRBAI  INCLUDE   AIPIVD,AITGVD,CONCAT,FLD,UTABRT,
258      ,UTACVD,UTCoin,UTHASH,UTINCH,UTQUAN,UTTOKE
259      *
260      AIRCAS  INCLUDE   AICOVD,AIPIVD,CONCAT,UTABRT,UTACVD,UTCoin,
261      ,UTHASH,UTINCH,UTQUAN,UTTOKE
262      *
263      AICYVD  INCLUDE   CONCAT
264      *
265      COMMON
266      END      LI

```

```

267      .PROC,MISEG,FYLE=MI,MAP=OFF.
268      RETURN,KNUJ,LGO.
269      ATTACH,KNUJ,MILGO,ID=T800855,PW=FOX.
270      .* SKIP IF NOT CATALOGED
271      SKIP,NONE.
272      EXIT,S.
273      ENDIF,NONE.
274      .* CONTINUE FROM HERE
275      REWIND,FYLE.
276      #MAP=MAP.
277      FTS5,I=FYLE,B=MIB,LO=0,DB=0.
278      .* CHECK FOR TWXLIB PRESENCE
279      IFE,FILE(ZZZZZLW,LO.OR.PF).NE.1,GET1.
280      .* NOT HERE, SO MUST GET IT
281      ATTACH,ZZZZZLW,TWXLIB,ID=T800855,CY=1.
282      ENDIF,GET1.
283      .* DECLARE IT AND THE UEDIT LIBRARY
284      LIBRARY(ZZZZZLW,ZZZZZLA)
285      RETURN,SEGLGO.
286      REQUEST,SEGLGO,*PF.
287      SEGLOAD(I=DIRS,B=SEGLGO)
288      LOAD(MIB)
289      NOGO.
290      CATALOG,SEGLGO,MILGO,ID=T800855,PW=FOX,XR=FOX.
291      PURGE,KNUJ.
292      .* SKIP TO "EXIT,S." STATEMENT IF FILE WAS NOT ATTACHED.
293      RENAME,SEGLGO,CY=1.
294      .* ALWAYS GO THROUGH THESE STATEMENTS
295      SKIP,NONE2.
296      EXIT,S.
297      ENDIF,NONE2.
298      RETURN,MIB,DIRS,XECUTE,KNUJ,SEGLGO.
299      .* RECALL THE LIBRARIES
300      LIBRARY(ZZZZZLA,ZZZZZLB)
301      .* NO MATTER WHAT
302      EXIT,S.
303      RETURN,MIB,DIRS,XECUTE.
304      LIBRARY(ZZZZZLA,ZZZZZLB)
305      REVERT.

```

```

306      .DATA, DIRS.
307      *
308      * MAIN TREE
309          TREE      MI-(ASKIT, CHANGE, CYCINPT, CYFMT1T,
310      , CYFMT2T, CYFMT3T, CYFMT4T, CYPRP1, DAYINP,
311      , DAYRAP, DELETE, ENTERT, INIT, INTERMT,
312      , MODIFYT, PRINTI, RESEQ, RESTAR)
313      *
314      MI      INCLUDE      CYOUT, CYPRP3, DAYOUT, ENCOD1, ENCOD2, ENCOD3,
315      , ENCOD4, PRNT1, SLITE, SLITET
316      *
317      * TREE FOR LINK A ( AKA "PI" )
318      ENTERT      TREE      ENTER-(INTWX, SETREC)
319      *
320      ENTER      INCLUDE      DETACH, PRNT1
321      INTWX      INCLUDE      CHKSYS, RANSIZ
322      SETREC      INCLUDE      RANSIZ
323      *
324      * TREE FOR LINK B ( AKA "RT" )
325      INTERMT      TREE      INTERM-YNC
326      *
327      INTERM      INCLUDE      FLD2, PRNT1
328      RESTAR      INCLUDE      ARRAYS, BCDASC, CALCSO, CYPRP3, SLITE
329      *
330      * TREE FOR LINK C ( AKA "DI" )
331      CYCINPT      TREE      CYCINP-(CYCPRE-(INACMA, INTGTT))
332      *
333      CYCINP      INCLUDE      CONCAT, ENCOD1, ENCOD2, ENCOD3, ENCOD4,
334      , PRNT1, RPRNT
335      INACMA      INCLUDE      FLD2, SETBIT
336      INTGTT      INCLUDE      FLD2
337      CYCPRE      INCLUDE      FLD, SLITE, SLITET
338      CYPRP1      INCLUDE      SLITE, SLITET
339      PRINTI      INCLUDE      PRNT1
340      *
341      * TREE FOR LINK D ( AKA "F1" )
342      CYFMT1T      TREE      CYFMT1-(FMT1A-(CKNUM1, EDIT1T))
343      *
344      CYFMT1      INCLUDE      FLD, PRNT1, RPRNT
345      FMT1A      INCLUDE      BLNKOU, CKCMA1, CONCAT, DETACH,
346      , ERR1, GREASE, SLITE, SLITET, UPDATE, UPRCAS
347      *
348      EDIT1T      TREE      EDIT1-(EDITA1, DECOD1, EDITB1,
349      , EDITC1, EDITD1)
350      EDIT1      INCLUDE      CONCAT, ERR1, PRNT1, SLITE,
351      , SLITET, UPRCAS
352      DECOD1      INCLUDE      SLITE
353      EDITA1      INCLUDE      ICHKAC, SLITE, SLITET
354      EDITB1      INCLUDE      CHKACM, FLD2, SLITE, SLITET, SETBIT
355      EDITC1      INCLUDE      SLITE, UPRCAS

```

```

356      EDITD1      INCLUDE      CHKTGH
357      *
358      * TREE FOR LINK E ( AKA "F2" )
359      CYFMT2T      TREE          CYFMT2-(FMT2A-(CKNUM2,EDIT4T))
360      *
361      CYFMT2      INCLUDE      FLD,PRNT1,RPRNT
362      FMT2A      INCLUDE      BLNKOU,CKCMA1,CONCAT,DETACH,ERR1,GREASE,
363      ,SLITE,SLITET,UPDATE,UPRCAS
364      *
365      EDIT4T      TREE          EDIT4-(DECOD2,EDITC2)
366      EDIT4      INCLUDE      EDITA2,EDITB2,FLD2,ICLKAC,SLITE,
367      ,SLITET,CHKACM,SETBIT
368      EDITC2      INCLUDE      CHKTGH,UPRCAS
369      *
370      * TREE FOR LINK F ( AKA "F3" )
371      CYFMT3T      TREE          CYFMT3-(FMT3A-(CKNUM3,DECOD3,EDITC3))
372      *
373      CYFMT3      INCLUDE      FLD,PRNT1,RPRNT
374      FMT3A      INCLUDE      BLNKOU,CHKACM,CKCMA1,CONCAT,EDITA2,EDITB2,
375      ,ERR1,FLD2,GREASE,ICLKAC,RPRNT,PRNT1,SETBIT,SLITE,
376      ,SLITET,UPDATE,UPRCAS
377      *
378      * TREE FOR LINK G ( AKA "F4" )
379      CYFMT4T      TREE          CYFMT4-(FMT4A-(CKNUM4,DECOD4,EDITC4T))
380      *
381      CYFMT4      INCLUDE      FLD,PRNT1,RPRNT
382      FMT4A      INCLUDE      BLNKOU,CHKACM,CKCMA1,CONCAT,EDITA2,EDITB2,
383      ,ERR1,FLD2,ICLKAC,GREASE,PRNT1,RPRNT,SETBIT,SLITE,SLITET,
384      ,UPDATE,UPRCAS
385      EDITC4T      TREE          EDITC4-(EDITD4,EDITF4)
386      EDITC4      INCLUDE      CHKTGH,ERR1,FLD,PRNT1,SLITE,SLITET
387      *
388      * TREE FOR LINK H ( AKA "CH" )
389      CHANGE      INCLUDE      FILESC,PRNT1,SLITE,UPRCAS
390      RESEQ      INCLUDE      SHUFFL
391      DELETE      INCLUDE      ARRAYS,CONCAT,EDIT1C,FILESC,PRNT1,SLITE
392      *
393      * TREE FOR LINK I ( AKA "PR" )
394      ASKIT      INCLUDE      PRNT1
395      *
396      * TREE FOR LINK K ( AKA "MO" )
397      MODIFYT      TREE          MODIFY-(FMAT1R,FMAT2R,FMAT3R,FMAT4R,FMAT1W,
398      ,FMAT2W,FMAT3W,FMAT4W)
399      *
400      MODIFY      INCLUDE      ARRAYS,CONCAT,EDIT1C,PRNT1,SLITE,UPRCAS
401      *
402      * TREE FOR LINK L ( AKA "WU" )
403      INIT      INCLUDE      BCDASC,CALCSO,CALLSS,TERMNO
404      DAYRAP      INCLUDE      DETACH,GREASE
405      *

```

```
406      * FINISH OF SEGMENTATION
407      COMMON      ACDATA, CONTRC, CONTRO, CYCLC, CYCLE, CYCLEC,
408      , FILEDA, FLAGS, READIN, READIC, SENSE, SORDAT
409      END          MI
410      .EOF
```

Appendix D

SETUP Listing

```

1      .PROC,SETUP.
2      .* THE PURPOSE OF THIS PROCEDURE FILE IS TO
3      .* BUILD ALL LIBRARIES AND OBJECT DECKS NECESSARY
4      .* FOR THE TWX. THE LIBRARIES WILL BE CREATED
5      .* INTERACTIVELY WHILE THE OBJECT DECKS WILL
6      .* BATCHED.
7      .* FIRST, THE JCL FOR THE BATCH JOB
8      .* IS SENT TO THE INPUT QUEUE.
9      RETURN,KNUJ.
10     REQUEST,KNUJ,*Q.
11     REWIND,JOB.
12     COPY,JOB,KNUJ.
13     BATCH,KNUJ,INPUT,HERE.
14     RETURN,JOB.
15     .* NOW, THE LIBRARIES ARE BUILT AND
16     .* FILLED WITH THE NECESSARY DATA FILES.
17     BUILD,ID=BACKUP.
18     BUILD,ID=BATCHIN.
19     BUILD,ID=MASTER.
20     BUILD,ID=PRINT.
21     GO,1.
22     GO,2.
23     GO,3.
24     GET,CAKM,ID=TWXDATA.
25     SAVE,CAKM=CAKW,ID=MASTER.
26     RETURN,CAKM.
27     GET,CCAM,ID=TWXDATA.
28     SAVE,CCAM=CCAW1,ID=BATCHIN.
29     RETURN,CCAM.
30     GET,CLAM,ID=TWXDATA.
31     SAVE,CLAM=CLAW1,ID=BATCHIN.
32     RETURN,CLAM.
33     GET,CLKM,ID=TWXDATA.
34     SAVE,CLKM=CLKW,ID=MASTER.
35     RETURN,CLKM.
36     GET,R2MM,ID=TWXDATA.
37     SAVE,R2MM=R2MW1,ID=MASTER.
38     SAVE,R2MM=R2MW1,ID=DATA12.
39     SAVE,R2MM=R2MW1,ID=DATA22.
40     RETURN,R2MM.
41     GET,R4MM,ID=TWXDATA.
42     SAVE,R4MM=R4MW1,ID=DATA12.
43     SAVE,R4MM=R4MW1,ID=DATA22.
44     RETURN,R4MM.
45     GET,RABM,ID=TWXDATA.
46     SAVE,RABM=RABW1,ID=DATA11.
47     SAVE,RABM=RABW1,ID=DATA21.
48     RETURN,RABM.
49     GET,RAPM,ID=TWXDATA.
50     SAVE,RAPM=RAPW1,ID=DATA11.

```

AD-A115 697 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL--ETC F/6 15/7
THEATER WARFARE PROGRAMS AT AFIT: AN INSTRUCTIONAL AID.(U)
MAR 82 A WAIKANEN
UNCLASSIFIED AFIT/OST/OS/82M-15 ML

242
242



END

DATE

FILMED

7-82

NTIC

51 SAVE,RAPM=RAPW1,ID=DATA21.
52 RETURN,RAPM.
53 GET,RBLM,ID=TWXDATA.
54 SAVE,RBLM=RBLW1,ID=DATA12.
55 RETURN,RBLM.
56 GET,RCRM,ID=TWXDATA.
57 SAVE,RCRM=RCRW1,ID=BATCHIN.
58 RETURN,RCRM.
59 GET,RLGM,ID=TWXDATA.
60 SAVE,RLGM=RLGW1,ID=BATCHIN.
61 RETURN,RLGM.
62 GET,RLUM,ID=TWXDATA.
63 SAVE,RLUM=RLUW1,ID=BATCHIN.
64 RETURN,RLUM.
65 GET,RACM,ID=TWXDATA.
66 SAVE,RACM=RACW,ID=MASTER.
67 RETURN,RACM.
68 GET,RMRM,ID=TWXDATA.
69 SAVE,RMRM=RMRR1,ID=DATA11.
70 SAVE,RMRM=RMRR1,ID=DATA21.
71 RETURN,RMRM.
72 GET,RMUM,ID=TWXDATA.
73 SAVE,RMUM=RMUW,ID=MASTER.
74 RETURN,RMUM.
75 GET,RPPM,ID=TWXDATA.
76 SAVE,RPPM=RPPW1,ID=BATCHIN.
77 RETURN,RPPM.
78 GET,RRLM,ID=TWXDATA.
79 SAVE,RRLM=RRLW1,ID=DATA22.
80 RETURN,RRLM.
81 GET,RSTM,ID=TWXDATA.
82 SAVE,RSTM=RSTW1,ID=BATCHIN.
83 RETURN,RSTM.
84 GET,RTGM,ID=TWXDATA.
85 SAVE,RTGM=RTGW1,ID=BATCHIN.
86 RETURN,RTGM.
87 GET,RWXM,ID=TWXDATA.
88 SAVE,RWXM=RWXW,ID=MASTER.
89 RETURN,RWXM.
90 REVERT.

91	.DATA,GO.
92	.PROC,GO,N.
93	BUILD,ID=DATA1#_N.
94	BUILD,ID=DATA2#_N.
95	REVERT.
96	.EOR
97	.DATA,JOB.
98	TWX,T500,I01000,CM100000. T800855 FOX
99	BEGIN,NOSFIL.
100	GET,TWXGO,ID=TWXRUN.
101	GET,TWXLIB,ID=TWXPROG.
102	TWXGO.
103	REQUEST,LGO,*PF.
104	GET,AR,ID=TWXPROG.
105	FTN5,I=AR,LO=0.
106	CATALOG,LGO,ARLGO,ID=T800855,XR=FOX,PW=FOX.
107	RETURN,LGO,AR.
108	GET,LB,ID=TWXPROG.
109	REQUEST,LGO,*PF.
110	FTN5,I=LG,LO=0.
111	CATALOG,LGO,LBLGO,ID=T800855,XR=FOX,PW=FOX.
112	RETURN,LGO.
113	GET,MISEG,ID=TWXRUN.
114	GET,MI,ID=TWXPROG.
115	MISEG.
116	GET,APESEG,ID=TWXRUN.
117	GET,APE,ID=TWXPROG.
118	APESEG.
119	GET,LISEG,ID=TWXRUN.
120	GET,LI,ID=TWXPROG.
121	LISEG.
122	GET,INPTGO,ID=TWXRUN.
123	INPTGO.
124	GET,PROCGO,ID=TWXRUN.
125	PROCGO.
126	.EOF

Appendix E

TWX File Descriptions

Appendix E

TWX File Descriptions

Four indirect library files are used for storing all the data files, procedure files, programs and utility routines required by the TWX at AFIT. These library files are TWXDATA, TWXRUN, TWXPROG and TWXUTIL, respectively. The primary reason behind separating the files was to emphasize the difference in the files as much as possible. This separation method is equivalent to the file-string identifiers used for the TWX files at Maxwell AFB.

TWXDATA

All data files are stored in the indirect library file TWXDATA (Table VI). There are two types of data files, sequential card image and direct access binary. The following nomenclature has been adopted to distinguish between them

ABBx#

where:

A = C if the file is sequential (card image)
R if the file is random (direct access)

BB = file identifier

x = M if the file is a master file
W if the file is a working file

= seminar number

TWXDATA contains only file masters. During initialization, copies are made of these files and are placed in the IFS files MASTER, DATA11, DATA12, DATA21, DATA22, and BATCHIN. MASTER contains those data files which are not modified during the execution phase. All IFS files with the prefix "DATA" contain the modifiable input files to APE, MI and LI where the name is of the form

DATAxy

where:

x = side (1 for Blue and 2 for Red)

y = 1, 2 or 3

1 = input files to APE and LI

2 = input files to MI (output from APE)

3 = output files from MI and LI

BATCHIN contains the input files to the batch jobs and the modified data files. Prior to the next day's run, those files which were originally in the DATA files (RAPW#, RABW#, R2MW#, R4MW#, RBLW# and RRLW#) must be modified with the appropriate utility programs and replaced.

TABLE VI

TWXDATA Files

Data File	Description
CAKM	Air constants file
CCAM	Combined actions file
CLAM	Land actions file
CLKM	Land constants file
RABM	Airbase file
RACM	Aircraft file
RAPM	AAFCE planning file
RBLM	Blue land file
RRLM	Red land file
RCRM	Combined reconnaissance file (not required for theater simulation)
RLGM	Logistics file
RLUM	Land units file
RMUM	Munition load file
RNRM	Mission reference file
R2MM	Mission working master file (2ATAF)
R4MM	Mission working master file (4ATAF)
RMWM	Mission working master file (ATAF flag not set)
RPPM	Preprogrammed control file (initially blank filled)
RSTM	Statistics file (initially blank filled)
RTGM	Target file
RWXM	Weather file

TWXPROG and TWXRUN

Two files are used for storing programs; TWXPROG and TWXRUN. TWXPROG contains only those programs (Table VII) which are totally source code (AR, APE, MI, LI, LB and TWXLIB). Programs in TWXRUN (Table IIIX) are preceded by a procedure file which compiles the program, accesses the necessary data files, executes the program, and replaces the modified data files. These pro-

grams are the relatively small batch programs SQ, MR, AG, MA, OR and LA. The purpose of this separation is to maximize clarity and minimize storage space. The primary function of TWXRUN is the storage of procedure files. Since the relatively small batch programs are prefixed with procedure files, they are stored in TWXRUN. The object decks for each of the small batch programs could be created once and stored on a disk file but then storage space would be required for two procedure files, the program source and its object deck for each of the programs. This way, only one procedure file and source is stored for each program.

TABLE VII

TWXPROG Files

File	Description
APE	AAFCE Planning Executive program
AR	Air battle simulation program
LB	Land battle simulation program
LI	Land order input program
MI	Mission order input program
MISC	Unmodified utility programs
TWXLIB	TWX Library routines
XXSRCE	Original main driver routine XX

TABLE IIX

TWXRUN Files

File	Description
AGGO, AGGOB	AG execution proc
APEGO	APE execution proc
APESEG	APE segmentation proc
ARGO	AR batch job spawning proc
ARSEG	AR segmentation proc (unvarified)
ARSEGO	AR segmentation spawning proc
INPTGO	INPTR compilation and cataloging proc
IPGO	IP execution proc
LAGO, LAGOB	LA execution proc
LBGO	LB execution spawning proc
LBSEG	LB segmentation spawning proc
LBSEG	LB segmentation proc (unvarified)
LBSEGO	LB segmentation spawning proc
LGGO	LG execution proc (no longer used)
LIGO	LI execution proc
LISEG	LI segmentation proc
MAGO, MAGOB	MA execution proc
MIGO	MI execution proc
MISEG	MI segmentation proc
MR	MR execution proc
ORGO, ORGOB	OR execution proc
SETUP	TWX file initialization proc
SQGO	SQ execution proc
STGO	ST execution proc
TWXJCL	original Honeywell JCL (archive file)
WSSGO	WSS execution proc

TWXUTIL

During the execution of the TWX, various data files must be modified, some directly, others indirectly. Unlike the sequential card image files which can be printed, edited, and replaced, direct access files require programs for reading and editing. Of the 22 data files used in TWX, 18 are direct access (these are identified by the first character; "R" for random). For these data files, programs have been developed to permit editing (Table IX).

TABLE IX

TWXUTIL Files

Program	File(s) read and Modified
RABREAD	RABx#
RACREAD	RACx
RAPREAD	RAPx#, RPPx# (no changes permitted)
RBLREAD	RBLx#, RRLx#, RMUx
RCRREAD	RCRx#
RLUREAD	RLUx#
RMRREAD	RMRx# (no changes permitted)
RPPREAD	RPPx#
RTGREAD	RTGx#
R2MREAD	R2Mx#, R4Mx#, RLGx and RMRx# (logicals are displayed as integers)

Vita

Anthony Waisanen was born in Warren, Minnesota on 7 January 1953 to Einard and Doris Waisanen. After graduating from Wadena Senior High School, Wadena, Minnesota in 1971, he enrolled at Concordia College, Moorhead, Minnesota. A year later, he enrolled at the University of Minnesota, Duluth where he received a commission in the Air Force through ROTC and Bachelor of Science degrees in Mathematics and Biology in June 1976. Following graduation, he was employed as a conventional weapon systems analyst at Eglin AFB, Florida from 24 October 1976 to 13 July 1980 before entering AFIT in September 1980. He is married to the former Holly A. Eifert of Wadena, Minnesota.

Permanent address: Box 116
Verndale, Minnesota 56481

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFIT/GST/OS/82M-15	2. GOVT ACCESSION NO. AD-A115697	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) THEATER WARFARE PROGRAMS AT AFIT: AN INSTRUCTIONAL AID		5. TYPE OF REPORT & PERIOD COVERED M.S. Thesis
7. AUTHOR(s) Anthony Waisanen Capt USAF		6. PERFORMING ORG. REPORT NUMBER
9. PERFORMING ORGANIZATION NAME AND ADDRESS Air Force Institute of Technology (AFIT/EN) Wright-Patterson AFB OH 45424		8. CONTRACT OR GRANT NUMBER(s)
11. CONTROLLING OFFICE NAME AND ADDRESS Air Force Institute of Technology (AFIT/EN) Wright-Patterson AFB OH 45424		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		12. REPORT DATE March 1982
		13. NUMBER OF PAGES 108
		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES 4 JUN 1982 LYNN E. WOLAVER Dean for Research and Professional Development AIR FORCE INSTITUTE OF TECHNOLOGY (ATC) WRIGHT-PATTERSON AFB, OH 45433 Approved for public release; IAW AFR 190-17		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Computerized simulation Operations research Programming techniques Central Europe scenario Gaming models		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) This thesis report details the processes and modifications that were required in order to execute the Theater Warfare Exercise (TWX) on the CDC computers at Wright-Patterson AFB. Prior to this effort, the TWX programs and data files could only be accessed and executed with Honeywell computers. By modifying the data files, program coding, overlaying techniques, and operation, the TWX can now be run on any CDC computer. Any other computer with sufficient central memory and an ANSI standard FORTRAN-77		

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

BLOCK 20.

→ compiler can also execute the programs provided the operation methods (procedure files) are modified).
↑

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

