ARI Contractor Report 96-86

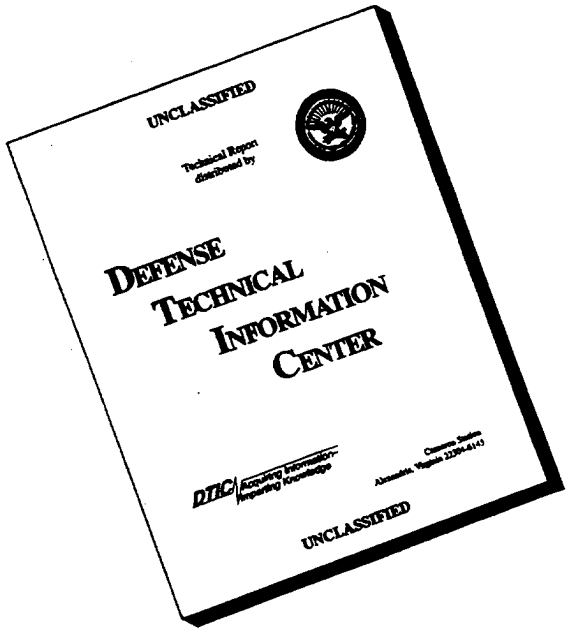
Final Report: Maintenance of the National Training Center (NTC) Mission Database and Replay Program During FY 95

William E. Walsh
BDM Federal, Inc.

September 1996

United States Army Research Institute for the Behavioral and Social Sciences

DISCLAIMER NOTICE



THIS DOCUMENT IS BEST QUALITY AVAILABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.

U.S. ARMY RESEARCH INSTITUTE FOR THE BEHAVIORAL AND SOCIAL SCIENCES

A Field Operating Agency Under the Jurisdiction
of the Deputy Chief of Staff for Personnel

EDGAR M. JOHNSON
Director

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	September 1996	Final Report 01/05/95 - 07/21/95

4. TITLE AND SUBTITLE
Final Report: Maintenance of the National Training Center (NTC) Mission Database and Replay Program During FY95

5. FUNDING NUMBERS
MDA903-92-D-0075-0034
2131
R12
MIPR from TRADOC

6. AUTHOR(S)
[redacted] Walsh, William E.

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
BDM FEDERAL INC.
BOD CENTER MONTEREY BAY
400 GIGLING ROAD
SEASIDE, CA 93955

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
U.S. ARMY RESEARCH INSTITUTE FOR THE
BEHAVIORAL AND SOCIAL SCIENCES
5001 EISENHOWER AVENUE
ALEXANDRIA, VA 22333-5600

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
Contractor Report 96-86

11. SUPPLEMENTARY NOTES
The COR is Michael R. McCluskey. This report is published to meet legal and contractual requirements and may not meet ARI's scientific or professional standards for publication.

12a. DISTRIBUTION/AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE;
DISTRIBUTION IS UNLIMITED.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

The purpose of this report is to document the work performed while maintaining the NTC Mission Database and Replay Program during the first six months of FY95. This project started in October 1994 and included a six month period of performance. This report includes a brief description of the NTC Mission Database, a copy of the User's Guide for the ARI-NTC Mission Database, a listing of the rotations and missions which are currently contained in the Mission Database, and the current C and FoxPro source code with documentation.

The Department of Defense, through AR 11-33, has designated the Combat Training Center (CTC) Archive as the repository for all unit generated and unit performance data derived from training exercises conducted at the combat training centers. Archive data are used to assess unit performance/readiness and to validate unit training practices. CTC training data are stored in several digital databases within the CTC Archive to facilitate access and research by military analysts. The Archive Research Databases provide powerful capabilities for using and sharing the CTC training data.

14. SUBJECT TERMS
Automated Finder's Guide (AFG), Battle Damage Assessment (BDA) Database, Graphics Database, Mission Database, Mission Replay and Battle Trace, Take Home Package (THP) Database

15. NUMBER OF PAGES

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UNLIMITED

CONTRACT #: MDA 903-92-D-0075
DELIVERABLE #: 0034-0034AC
July 21, 1995

FINAL REPORT

MAINTENANCE OF THE NATIONAL TRAINING CENTER (NTC) MISSION DATABASE AND REPLAY PROGRAM DURING FY95

William E.
~~Bill~~ Walsh
BDM Federal, *Inc.*

Submitted by: Mr. Michael R. McCluskey, Acting Chief
Unit-Collective Training Research Unit
and Dr. Zita M. Simutis, Director
Personnel and Training Research Division

Mr. Michael R. McCluskey, Contracting Officer's Representative

August 17, 1995

U.S. Army Research Institute

FINAL REPORT

Contents

FINAL REPORT

Contents

Introduction

The Combat Training Centers (CTCs) provide superb combined arms training experiences, excellent readiness feedback to the training units and rich sources of training data for research and analysis purposes. The Department of Defense, through AR 11-33, has designated the Combat Training Center (CTC) Archive as the repository for all unit generated and unit performance data derived from training exercises conducted at the combat training centers. Archive data are used to assess unit performance/readiness and to validate unit training practices. Currently, the CTC Archive contains training data from the National Training Center (NTC), the Joint Readiness Training Center (JRTC), and the Combat Maneuver Training Center (CMTC). It is anticipated that the CTC Archive Research Databases will be expanded during FY 96 to include training data from the Battle Command Training Program (BCTP).

CTC training data are stored in several digital databases within the CTC Archive to facilitate access and research by military analysts. The Archive Research Databases provide powerful capabilities for using and sharing the CTC training data. The CTC Archive Digital Databases and Automated Research Tools include:

Automated Finder's Guide (AFG)
Battle Damage Assessment (BDA) Database
Graphics Database
Mission Database
Mission Replay and Battle Trace
Take Home Package (THP) Database

The Mission Database is a relational database which contains data derived from the Core Instrumentation Subsystem (CIS) at the NTC. CIS data are collected on SUN Work Stations and stored in a real time format. The Mission Database is a collection of battle/mission research databases - a separate research database is generated for each mission conducted at the NTC during a training rotation. The individual battle/mission research databases within the Mission Database contain direct fire event data, indirect fire mission data, minefield data, communications data, ground and air player position location data, fratricide data, and maneuver control measure data.

The Mission Database consists of a collection of relational tables which are organized by fiscal year, rotation, and mission. Twelve control measure tables exist for each rotation and each battle/mission research database contains sixteen relational database tables. The tables are defined as either "event" or "static" tables, depending on the type of data that they contain. "Event" data describes an occurrence during a training mission, while "static" data defines the condition of the battlefield and players during the rotation and during the training mission. Appendix A, Mission Database Structure, to Annex A, User's Guide for the ARI-NTC Mission Database contains detailed descriptions of the relational tables. The elements of the tables are defined in terms of name, description, and format.

The Mission Replay and Battle Trace Automated Research Tool uses data directly from the Mission Database tables to graphically portray unit maneuvers, direct and indirect firing

events, unit task organizations, results of battlefield engagements, and unit maneuver control measures. In addition, this software tool provides a trace of the battle over a map background which contains realistic NTC terrain features. The source code files for the Mission Replay and Battle Trace are contained on the enclosed compact discs within the ARCHIVE\SRC\REPLAY subdirectory.

The instrumentation hardware and software systems at the NTC were revised and new systems were installed during the first six months of FY 95. The new systems at the NTC will be completely operational by July 1995. As a result of these changes at the NTC, software upgrades to the CALL systems will be necessary in order to continue the processing of the NTC instrumentation data and the construction of mission databases in the future.

Objective

The purpose of this report is to document the work performed while maintaining the NTC Mission Database and Replay Program during the first six months of FY 95. This project started in October 1994 and included a six month period of performance. This report includes a brief description of the NTC Mission Database, a copy of the User's Guide for the ARI-NTC Mission Database, a listing of the rotations and missions which are currently contained in the Mission Database, and the current C and FoxPro source code with documentation.

Project Objectives for Delivery Order 34

To maintain the NTC Mission Database

The ARI, UCTRU received and processed digital data tapes which were generated by the instrumentation systems at the NTC during field training exercises. The instrumentation data from each rotation were processed into database format and used to construct mission databases. Software support was required to process the raw NTC instrumentation data and to maintain the NTC Mission Database at ARI, UCTRU during the first half of FY 95.

To maintain and enhance the NTC Replay Program

The NTC Replay Program uses the data directly from the mission databases to provide analysts with the ability to examine unit movements and engagements during NTC battles. Software support was required to maintain the NTC Replay Program at ARI, UCTRU during the first half FY 95.

Scope of Effort

The following four tasks were accomplished in order to satisfy the two project objectives defined above:

a. Task 1: Prepare a plan for the execution of the effort. The Contractor prepared an overall plan to guide the work. The plan described how the NTC Mission Database and Replay Program would be maintained at ARI, UCTRU during the first six months of FY 95. The plan also described the deliverable products and provided a schedule of project deliverables. This plan was reviewed and approved by ARI, UCTRU prior to the start of the project.

b. Task 2: Provide data processing and software support for the maintenance of the NTC Mission Database at ARI, UCTRU. The Contractor used the appropriate data sources (THPs, Rotational Summary Documents and End of Mission AAR Summaries) to develop the necessary time lines and mission cuts. The Contractor provided the software support that was required for the creation of research mission databases from the instrumentation data for each NTC rotation during the first six months of FY 95. The Contractor incorporated the NTC instrumentation data into the Mission Database and provided programming support for data retrieval.

c. Task 3: Provide software support for the maintenance of the NTC Replay Program at ARI, UCTRU. The Contractor continued to maintain the Replay Program's ability to access and use data directly from the NTC Mission Database.

d. Task 4: Prepare a final project report. This report included current software documentation and source code for the NTC Mission Database.

Description of the NTC Mission Database

The NTC Mission Database consists of battle/mission database tables and rotational database tables. These tables are organized by fiscal year, rotation, and mission within the Mission Database. The battle/mission database tables are FoxPro tables which contain both "static" and "event" data which define unit status and mission events. The rotational database tables are also FoxPro tables, however, these tables are composed entirely of "static" data which define unit control measures for the entire rotation. The table types were designed to provide the maximum amount of relevant data to support analysis of maneuver training. See Appendix A, Mission Database Structure, to Annex A, User's Guide for the ARI-NTC Mission Database, for the complete descriptions of the relational database tables.

The Mission Database rotational and battle/mission tables are included along with the other CTC Archive Data and Software for FY 92, FY 93, FY 94, and FY 95 (partial) on the enclosed Compact Discs (CDs). The following table provides a high level description of the subdirectory structure for the CDs containing the CTC Archive Data and Program Software:

Subdirectory	Description
\ARCHIVE\AFG	Database tables for use with the Automated Finder's Guide
\ARCHIVE\BDA	CTC Battle Damage Assessment Database tables
\ARCHIVE\BIN	Executable code for Archive Tools
\ARCHIVE\GRAPHICS	CTC mission graphics by rotation
\ARCHIVE\MISSION	NTC mission database tables by year, rotation, and mission
\ARCHIVE\SRC	Source code for executable files in \ARCHIVE\BIN
\ARCHIVE\THP	CTC Take Home Packages by rotation

The \ARCHIVE\MISSION subdirectory contains two additional subdirectory levels. Within each fiscal year (FY 92, FY 93, FY 94, and FY 95 (partial)) the ARCHIVE\MISSION subdirectory defined above contains a rotational subdirectory level with a rotational subdirectory for each training rotation that occurred during the specified year, e.g. subdirectory ARCHIVE\MISSION\N931 exists for NTC rotation 93_01. The FoxPro database files which contain the control measure tables for each rotation have been placed in the corresponding rotational subdirectory. In addition, at the rotational subdirectory level, each rotational subdirectory contains a mission subdirectory level with a mission subdirectory for each mission that occurred during the specified rotation, e.g. subdirectory ARCHIVE\MISSION\N931\N931CV16 exists for mission N931CV16 of NTC rotation 93_01. The FoxPro database files which contain the battle/mission tables for each mission have been placed in the corresponding mission subdirectory.

The following representation defines a "segment" of the directory/subdirectory path listing for the Mission Database files/tables which are contained on the enclosed compact discs:

```
CD Drive:\
        Archive\
                AFG\
                BDA\
                BIN\
                GRAPHICS\
                MISSION\
                    N931\
                            N931CV16\
```

```
                    aplt.dbf,  ct.dbf, esit.dbf, esut.dbf, fet.dbf, frat.dbf,
                    gplt.dbf, ifct.dbf, ifmf.dbf, mct.dbf, mid.dbf, pet.dbf
                    platform.dbf, symbol.dbf, task_org.dbf, weapon.dbf
            N931C_09\
            N931C_11\
            N931C_13\
            N931_V04\
            N931_V06\
            N931_V08\
            arc.dbf
            circle.dbf
            cm_maste.dbf
            ellipse.dbf
            ifgt.dbf
            iftt.dbf
            line.dbf
            point.dbf
            polygon.dbf
            polyline.dbf
            rectangl.dbf
            text.dbf
        N932\
        N933\
SRC\
        AFG\
        REPLAY\
THP\
```

The creation of a battle/mission research database is a two step process. First, FoxPro rotational tables are built and loaded with the control measure data for the entire training rotation. The control measure tables for each rotation of FY 92, FY 93, FY 94, and FY 95 (partial) have been placed in the appropriate rotational subdirectories on the enclosed compact discs, e.g. \ARCHIVE\MISSION\N931 for NTC rotation 93_01. The second step in the Mission Database creation process involves the construction and loading of the FoxPro tables that constitute a particular battle/mission database. These tables define unit status and mission events during individual missions within a rotation. The battle/mission database tables for each mission of each rotation of FY 92, FY 93, FY 94, and FY 95 (partial) have been placed in the appropriate mission subdirectories on the enclosed compact discs, e.g. \ARCHIVE\MISSION\N931\N931CV16 for mission N931CV16 of NTC rotation 93_01.

ANNEX A

USER'S GUIDE FOR THE ARI-NTC MISSION DATABASE
(REVISED FOR NTC ROTATIONS 92-01 THROUGH 95-04)

JACK D. BALDWIN
BDM INTERNATIONAL, INC.

USER'S GUIDE FOR THE NTC MISSION DATABASE

Contents

LIST OF TABLES

LIST OF FIGURES

Purpose

This document is intended for users of the ARI-NTC mission database. Its purpose is to familiarize the user with the current database structure and provide examples of Structured Query Language (SQL) routines for the researcher when doing analysis with the training data. These examples may help the reader understand how to construct their own queries once they have established an understanding of the mission databases. Appendix A contains a description of the NTC mission database formats. If the reader is not familiar with the structure and content of the mission databases, it would be of value for the reader to refer to them now, before working through the exercises.

We shall explore the principal linkages between the relational tables in the data base, that is the Player Identification Number (PID), Logical Player Number (LPN) and the Time of an event. It should be noted that our data bases are 'event driven', because all the entries in the Fire Event Table (FET), the Pairing Event Table (PET), and the Indirect Fire Missions Fired (IFMF) result from actions initiated in the training exercise.

It is our desire to provide a template for use when the researcher designs a plan of analysis for a specific research issue. It is assumed that the reader will be familiar with using SQL and have a knowledge of the particular training rotation they wish to investigate.

Gaining access to ARI databases

If you are a workshop participant, you can access the ARI-POM research databases through the analyst workstation. Use the remote access menu and select the databases for 92-01 rotations forward. All accounts have been constructed for your use, and no personal password are required.

A list of all current research databases are provided with each menu selection. Any of these can be accessed by the researcher.

The Player: PID and LPN

We begin our discussion with an explanation of the player identification number, the PID. (Note here that the title 'player' actually refers to a specific tank crew, a TOW squad or a squad in an armored personnel carrier). It is a three character acronym for a player's company and platoon. Consider the player name 'SA13'; Character one denotes the Observer-Controller team which is training the player. The second character is alphabetic, and therefore we may assume the player belongs to 'A' company. If the player name were designated as 'S13A', we can be led to believe that the player is from a different battalion (i.e. a cross attached unit). The position within the name of the alpha character describes that player's task force association. The '1' in SA13 describes the players platoon. The '3' identifies the squad within the platoon. With this information, we see that 'A13' stands for 'A' company, first platoon, third squad.

The rules are:

first character - Observer/Controller team - SA13

B - broncos
C - cobras
S - scorpions
T - tarantulas
W - werewolves

for indigenous task force - SA13

A = company
1 = platoon
3 = squad

for cross attached units - S13A

1 = platoon
3 = squad
A = company

There are a number of notable exceptions to the above rules, which will be addressed now. Leaders will have a six in their name as in SH66 (battalion commander), SH65 (second in charge), SB66 (B company commander), SB65 (B company second in charge) and so on. Anti-tank squads (using TOWs) are given names with an 'E'. A partial list of anti-tank player names would be SE11, SE12, and so forth. Player identifications are not unique, as both Opfor and Bluefor players can have the same PID (as in SH66, the commander of each force). In order to uniquely determine a player, we use the Logical Player Number or LPN, which is the player's sequence number in the Element State Initialization Table (ESIT). Each player in the ESIT is assigned a unique sequence number, and can be tracked with this number throughout the mission.

The Tables: Event data vs. Static data

Lets begin our discussion here with a distinction between 'event' and 'static' data. Events are those data that occurred in the field of play, such as a main gun round being fired. Think of them as 'what's going on' during a training mission. Static data defines the condition of the field and players during a training mission. Think of static data as 'what it looks like'. We will now expand on these concepts.

A good rule of thumb for recognizing an 'event' table is the presence of a time variable. In our database, the following tables have a time stamp with each event:

1) Fire Event Table (FET)
2) Pairing Event Table (PET)
3) Indirect Fire Missions Fired (IFMF)
4) Indirect Fire Casualties Table (IFCT)
5) Minefield Casualties Table (MCT)
6) Communications Table (CT)
7) Ground Player Location Table (GLPT)

8) Air Player Location Table (APLT)
9) Element State Update Table (ESUT)
10) Fratricide Table (FRAT)

All of the above tables can be linked logically by their time variable, and can give us a chronological view of the training exercise. The exception is the Control Measure Master Table (CM_MASTER) and the Mission Identification Table (MID). These two tables have a start time and end time, which bracket the events within a training exercise.

Static tables support the event tables. They contain the condition and or state of the troops at the start of the training exercise. The following is a list of the database's static tables:

1) Element State Initialization Table (ESIT)
2) Weapon Table (WEAPON)
3) Platform Table (PLATFORM)
4) Symbol Table (SYMBOL)
5) Indirect Fire Target Table (IFTT)
6) Indirect Fire Group Table (IFGT)
7) Control Measure Master Table (CM_MASTER)
8) Control Measure Arc Table (ARC)
9) Control Measure Circle Table (CIRCLE)
10) Control Measure Ellipse Table (ELLIPSE)
11) Control Measure Point Table (POINT)
12) Control Measure Line Table (LINE)
13) Control Measure Polyline Table (POLYLINE)
14) Control Measure Polygon Table (POLYGON)
15) Control Measure Rectangle Table (RECTANGLE)
16) Control Measure Text Table (TEXT)

These tables help give the researcher an idea of the physical condition of the players and the battlefield during the training mission.

With a better understanding of Event and Static table types, let us now investigate how the researcher can utilize the information in the different tables to their advantage. We will begin with some elementary examples of querying the database and progress to move complex techniques.

For a first example, let us determine the vehicle type of a target (person being fired at) in the PET. We will need to look at the LPN of the target in the PET and match it to the LPN in the ESIT (See Appendix A for a description of the variables within each database table).

Examples: The Pairing Event Table.

The Pairing Event Table (PET) is considered to be the heart of the NTC research database. It is where the direct fire assessments are recorded on a player by player basis. Our first set of query examples will center around using the PET. We hope that these examples will provide a useful guideline for researchers.

Example one merely matches the PET to the ESIT using the targets' logical player number as the thread between the two tables.

Example 1: Finding the vehicle description of the Target

```
1> /*****************************************************************/
2> /* PET example # 1:  determine the vehicle type of a target from */
3> /*                a pairing event in the PET                     */
4> /*****************************************************************/
5> select  p.time,      /* columns to be returned... */
6> p.tpid,
7> p.result,
8> p.tside,
9> t.platform_desc
10> from    PET p,      /* tables to be used in script */
11> ESIT e,
12> PLATFORM t
13> where   p.tlpn = e.lpn     /* match tables on lpn  */
14> and    e.platform = t.platform_type
15> order by p.time            /* sort by time         */
```

time	tpid	result	tside	platform_desc
10-Sep-1992 05:37:20	CD21	Hit	B	M1A1_TANK
10-Sep-1992 05:37:46	CD21	Near Miss	B	M1A1_TANK
10-Sep-1992 05:38:28	CD21	Hit	B	M1A1_TANK
10-Sep-1992 05:39:37	CD21	Near Miss	B	M1A1_TANK
10-Sep-1992 05:39:37	CD21	Near Miss	B	M1A1_TANK
10-Sep-1992 05:39:56	CD21	Near Miss	B	M1A1_TANK
10-Sep-1992 05:40:01	CD21	Near Miss	B	M1A1_TANK
10-Sep-1992 05:40:30	SE42	Near Miss	B	M901_AT_APC
10-Sep-1992 05:42:23	CA31	Near Miss	B	M1A1_TANK
10-Sep-1992 05:43:20	212	Near Miss	O	BMP1
10-Sep-1992 05:48:26	CD11	Near Miss	B	M1A1_TANK
10-Sep-1992 05:49:34	CA34	Kill	B	M1A1_TANK
10-Sep-1992 05:51:13	CB21	Hit	B	M1A1_TANK
10-Sep-1992 05:54:26	C60C	Kill	B	M2_IFV
10-Sep-1992 05:57:06	CD24	Near Miss	B	M1A1_TANK
10-Sep-1992 05:57:09	C60C	Hit		M2_IFV
10-Sep-1992 05:58:02	CD23	Hit	B	M1A1_TANK
10-Sep-1992 06:06:33	271	Near Miss	O	BMP1
10-Sep-1992 06:07:12	C60C	Kill	B	M2_IFV
10-Sep-1992 06:09:20	B14	Near Miss	O	T72_TANK
10-Sep-1992 06:10:52	CB21	Hit	B	M1A1_TANK
10-Sep-1992 06:16:38	271	Near Miss	O	BMP1
10-Sep-1992 06:18:08	271	Kill	O	BMP1
10-Sep-1992 06:18:14	E927	Kill	B	UH_1
10-Sep-1992 06:19:30	233	Hit	O	BMP1
10-Sep-1992 06:20:45	CH66	Near Miss	B	M1A1_TANK
10-Sep-1992 06:22:14	272	Near Miss	O	BMP1
10-Sep-1992 06:22:26	CH66	Hit	B	M1A1_TANK
10-Sep-1992 06:22:52	CH66	Near Miss	B	M1A1_TANK
10-Sep-1992 06:23:13	710	Near Miss	O	BRDM2_AT5
10-Sep-1992 06:23:52	CH66	Hit	B	M1A1_TANK
10-Sep-1992 06:23:56	710	Near Miss	O	BRDM2_AT5
10-Sep-1992 06:24:07	710	Near Miss	O	BRDM2_AT5
10-Sep-1992 06:25:34	CH66	Near Miss	B	M1A1_TANK
10-Sep-1992 06:26:34	CS9	Near Miss	B	MANPACK_TOW

The fields that are displayed are in the same order as they appear in the 'SELECT' statement of the query. 'Time' refers to the time of the pairing event. 'TPID' is the targets player identification (the target). 'RESULT' is the outcome of the engagement. 'TSIDE' is the force code of the target.

The last line of the query is a sort directive, which keeps the data in the same order as it occurred during the training exercise.

The next example builds upon the first in that we've added a third table to our query. Let's see if we can determine the weapon system that caused the pairing to occur, by matching the weapon code to the weapon table and extracting a weapon description.

Example 2:

```
1> /*****************************************************************************/
2> /* PET example # 2:  determine the vehicle type of a target from          */
3> /*                   a pairing event in the PET and also, get a     */
4> /*                   description of the weapon type causing the*/
5> /*                   pairing to occur.                    */
6> /*****************************************************************************/
7> select  p.time,      /* columns to be returned... */
8> p.tpid,
9> p.result,
10> p.tside,
11> t.platform_desc,
12> p.fpid,
13> w.weapon_desc
14> from    PET p,     /* tables to be used in script */
15> ESIT e,
16> PLATFORM t,
17> WEAPON w
18> where   p.tlpn = e.lpn    /* match tables on lpn  */
19> and    e.platform = t.platform_type /* and vehicle */
20> and    p.weapon = w.weapon_type
21> order by p.time          /* sort by time        */
```

time	tpip	result	tside	platform_desc	fpip	weapon_desc
16-Jul-1992 18:20:12	S34B	Near Miss	B	M1A1_TANK		WEAPON UNDEFINED
16-Jul-1992 18:20:35	S34B	Near Miss	B	M1A1_TANK		WEAPON UNDEFINED
16-Jul-1992 18:32:22	S22B	Near Miss	B	M1A1_TANK		105MM TANK MAIN GUN
16-Jul-1992 18:32:24	S22B	Near Miss	B	M1A1_TANK		105MM TANK MAIN GUN
16-Jul-1992 18:32:25	S22B	Near Miss	B	M1A1_TANK		105MM TANK MAIN GUN
16-Jul-1992 18:32:27	S22B	Near Miss	B	M1A1_TANK		105MM TANK MAIN GUN
16-Jul-1992 18:37:04	090	Kill	O	BRDM		120MM TANK MAIN GUN
16-Jul-1992 18:43:30	090	Hit	O	BRDM		WEAPON UNDEFINED
16-Jul-1992 18:43:30	090	Hit	O	BRDM		WEAPON UNDEFINED
16-Jul-1992 18:43:34	090	Hit	O	BRDM		30MM AIRBORNE
16-Jul-1992 19:02:57	CA65	Kill	B	M1A1_TANK		30MM AIRBORNE
16-Jul-1992 19:03:33	CA65	Hit	B	M1A1_TANK		30MM AIRBORNE
16-Jul-1992 19:03:36	CA65	Hit	B	M1A1_TANK		VIPER LAW
16-Jul-1992 19:04:05	CA65	Hit	B	M1A1_TANK		120MM TANK MAIN GUN
16-Jul-1992 19:04:07	CA65	Hit	B	M1A1_TANK		152MM TANK MAIN GUN

Example three turns our attention to the firer, the player who pulled the trigger to cause the pairing. Because the NTC uses the MILES laser system for simulating engagements, many times the specific firer is unknown to the SUN Workstations. This is reflected in the data, as only approximately twenty percent of the pairing events have a known firer. When the firer is known, his LPN is recorded in the field 'flpn' of the PET, otherwise the flpn is zero. We can use this fact to find those 'matched' pairing events in the PET. Example three is similar to example one, except now we seek information about the firer.

Example 3:

```
 1> /*****************************************************************************/
 2> /* PET example # 3: determine the vehicle type of a target from     */
 3> /*                  a pairing event in the PET and also, get a      */
 4> /*                  description of the weapon type causing the      */
 5> /*                  pairing to occur, and only select those         */
 6> /*                  events in which the firer is known.             */
 7> /*****************************************************************************/
 8> select  p.time,      /* select columns to be returned... */
 9> p.tpid,
10> p.result,
11> p.tside,
12> t.PLATFORM_desc,
13> p.fpid,
14> w.weapon_desc,
15> p.distance
16> from    PET p,     /* tables to be used in script */
17> ESIT e,
18> PLATFORM t,
19> WEAPON w
20> where   p.flpn > 0
21> and   p.tlpn = e.lpn    /* match tables on lpn  */
22> and   e.PLATFORM = t.PLATFORM_type /*nd vehicle */
23> and   p.weapon = w.weapon_type
24> order by p.time          /* sort by time        */
```

time	tpid	result	tside	platform_desc	fpid	weapon_desc	distance
17-Jul-1992 06:11:34	610	Kill	O	BRDM	TD53	TOW	927
17-Jul-1992 07:58:33	617	Kill	O	BMP1	TD21	TOW	1063

Our next example will explore uses of the indirect fire tables, the Indirect Fire Missions Fired (IFMF) and the Indirect Fire Casualty Table (IFCT). The following example shows a method to link the two tables by time and plan id.

Example 4:

```
 1> /*******************************************************/
 2> /* Example 4:     Matching an indirect fire casualty  */
 3> /*              event with the indirect fire mission */
 4> /*******************************************************/
 5> select m.time,              /* columns to be returned */
 6>        m.lpn,
 7>        m.pid,
 8>        m.plan_id,
 9>        m.side,
10>        c.lpn,
11>        c.pid,
12>        c.side,
```

```
13>       dist_from_impact = sqrt((m.impact_x - c.x)**2 +
14>                               (m.impact_y - c.y)**2)
15> from   ifmf m,              /* Tables to use..        */
16>        ifct c
17> where  c.time = m.time      /* search conditions      */
18> and    c.plan_id = m.plan_id
```

```
+------------------------+--------+------------------+-----------------+-------+--------+------------+-------+-------------+
|time                    |lpn     |pid               |plan_id          |side   |lpn     |pid         |side|dist_from_i |
+------------------------+--------+------------------+-----------------+-------+--------+------------+-------+-------------+
|16-Jul-1992 05:28:26 |659     |/1/A/1 /319       |W-03-FOF  |B      |1503          |TD21|B|105.475|
|16-Jul-1992 06:08:54 |659     |/1/A/1 /319       |W-05-FOF  |B      |1584          |TEN3|B|168.048|
+------------------------+--------+------------------+-----------------+-------+--------+------------+-------+-------------+
```

The last example shows an example of using some INGRES provided numerics function to compute the average engagement range,the number of engagements by weapon system, the minimum engagement range and the maximum engagement range for matched pairing events. Other numeric functions such as summation and trigonometric functions are provided by INGRES and are described in Volume One of the INGRES manuals.

Example Five: Computing summary statistics.

```
1> select distinct
2> q.weapon,
3> w.weapon_desc,
4> e.platform,
5> p.platform_desc,
6> ave_range = avg(q.distance),
7> sample_size = count(q.distance),
8> min_distance = min(q.distance),
9> max_distance = max(q.distance)
10> from   PET q,
11> WEAPON w,
12> PLATFORM p,
13> ESIT e
14> where  q.flpn > 0
15> and    q.weapon = w.weapon_type
16> and    q.flpn = e.lpn
17> and    e.platform = p.platform_type
18> group by q.weapon,
19> w.weapon_desc,
20> e.platform,
21> p.platform_desc
```

```
-------------------------------------------------------------------------------------------------------------------------
|weapon |weapon_desc                |platfo |platform_desc  |ave_range     |sample_size  |min_distance  |max_distance |
-------------------------------------------------------------------------------------------------------------------------
|31     |120MM TANK MAIN GUN |32     |M1A1_TANK  |2043.600      |5            |416           |2822|
|172    |AT5                        |171    |BMP2           |2099.000      |2            |2011          |2187|
-------------------------------------------------------------------------------------------------------------------------
```

Now, suppose we wished to collect this same data across multiple mission databases and perform summary statistics on them. We could use either a statistical package or a spreadsheet program on a microcomputer to help us get the data into a presentation form.

First, we would need to output the data from one mission database into a file with a format acceptable to a statistical package or spreadsheet program, and then combine these files into a single file for input into our summary software. To do this, we use the Report Writer function of INGRES.

Using Report Writer to Cross Multiple Mission Databases

Note that the 'sort' option is removed from the query and placed with the report. The output file is defined with the '.OUT' statement and is called 'range.dat' in our example. The '.PRINT' line defines our output format and includes special characters used to delimit the fields of the records. Other delimiters may be used based on the needs of your summary software. The last command, '.NL' tells the report writer to end each record with a new line character. The following is a listing of an INGRES report:

```
.NAME matchpair
.OUT  matchpair.dat
.PRINT          "\"","DB name","\"",",",
        "\"","Mission Type","\"",",",
        "\"","Tgt Side","\"",",",
        "\"","Event Time","\"",",",
        "\"","Target Description","\"",",",
        "\"","Result","\"",",",
        "\"","Firer Side","\"",",",
        "\"","Weapon Description","\"",",",
        "\"","Distance","\"" .NL
.QUERY
select dbname = dbmsinfo('database'), m.phase_type,
      p.tside, p.time,
      s.platform_desc,
      p.result, p.fside,
      w.weapon_desc,
      p.distance
from mid m, pet p, platform s, esit e, weapon w
where p.tlpn = e.lpn
and e.platform = s.platform_type
and p.weapon = w.weapon_type
and p.distance > 0
.DETAIL
.PRINT          "\"",dbname (c8),"\"",",",
        "\"",phase_type (c20),"\"",",",
        "\"",tside (c1),"\"",",",
        "\"",time (c20),"\"",",",
        "\"",platform_desc (c20),"\"",",",
        "\"",result (c9),"\"",",",
        "\"",fside (c1),"\"",",",
        "\"",weapon_desc (c20),"\"",",",
        ,distance (n6) .NL
```

Two steps are needed to execute the above report. The first is to check the syntax of our report and install it in the necessary mission database:

```
$ sreport *dbname* matchpair.rw
```

where *dbname* is one of the mission database names and 'matchpair.rw' is the name given to the file containing our report writer commands. The next step is to run the report, and that is done by the following:

$ report *dbname* matchpair

where 'matchpair' is the name given to the report on the '.NAME' line of the report definition. The output is a file named 'matchpair.dat' in your current working directory. The output file looks like the listing on the next page.

For each mission database that we apply the report against, we will get a file named matchpair.dat. Each of these files need to be renamed, so the next time the report is executed, it will not write over the existing file. The following UNIX command will accomplish this:

mv matchpair.dat 'newfilename.dat'

where 'newfilename.dat' is the name of a file of your own choosing. Be systematic in the naming of your files. for example, you should name the files like 'N923AM04.out' [N923AM04 being the database name].

Note that the first line of the output has field names in it. This will allow you to import the data into a spreadsheet, and have cell names at the top of each column. This will create a problem with the numeric fields in the file, as the spreadsheet will think that the numeric fields are of type text. You may modify this in the spreadsheet once you have imported the data.

```
"DB name","Mission Type","Tgt Side","Event Time","Target Description","Result","Firer Side","Weapon Description","Distance"
"n923am04","MOVEMENT TO CONTACT ","O","04-Dec-1991 10:59:53","BMP1          ","Near Miss","B","105MM TANK MAIN GUN ", 1537
"n923am04","MOVEMENT TO CONTACT ","O","04-Dec-1991 10:57:13","T72_TANK      ","Near Miss","B","105MM TANK MAIN GUN ", 1575
"n923am04","MOVEMENT TO CONTACT ","O","04-Dec-1991 11:16:29","T72_TANK      ","Near Miss","B","105MM TANK MAIN GUN ", 2274
"n923am04","MOVEMENT TO CONTACT ","O","04-Dec-1991 11:14:38","T72_TANK      ","Near Miss","B","105MM TANK MAIN GUN ", 2850
"n923am04","MOVEMENT TO CONTACT ","O","04-Dec-1991 11:09:31","BMP1          ","Near Miss","B","25MM IFV MAIN GUN  ", 689
"n923am04","MOVEMENT TO CONTACT ","O","04-Dec-1991 07:46:51","BMP1          ","Hit      ","B","25MM IFV MAIN GUN  ", 1410
"n923am04","MOVEMENT TO CONTACT ","O","04-Dec-1991 09:01:08","BMP1          ","Near Miss","B","TOW             ", 1095
"n923am04","MOVEMENT TO CONTACT ","O","04-Dec-1991 09:01:42","T72_TANK      ","Kill     ","B","TOW             ", 1512
"n923am04","MOVEMENT TO CONTACT ","B","04-Dec-1991 12:06:49","M2_IFV        ","Kill     ","O","125MM TANK MAIN GUN ", 166
"n923am04","MOVEMENT TO CONTACT ","B","04-Dec-1991 12:07:00","M2_IFV        ","Near Miss","O","125MM TANK MAIN GUN ", 166
"n923am04","MOVEMENT TO CONTACT ","B","04-Dec-1991 11:04:41","M1_TANK      ","Kill     ","O","125MM TANK MAIN GUN ", 291
"n923am04","MOVEMENT TO CONTACT ","B","04-Dec-1991 10:55:58","M1_TANK      ","Near Miss","O","125MM TANK MAIN GUN ", 420
"n923am04","MOVEMENT TO CONTACT ","B","04-Dec-1991 10:56:31","M1_TANK      ","Near Miss","O","125MM TANK MAIN GUN ", 420
"n923am04","MOVEMENT TO CONTACT ","B","04-Dec-1991 10:57:56","M1_TANK      ","Near Miss","O","125MM TANK MAIN GUN ", 420
"n923am04","MOVEMENT TO CONTACT ","B","04-Dec-1991 11:03:03","M1_TANK      ","Near Miss","O","125MM TANK MAIN GUN ", 487
"n923am04","MOVEMENT TO CONTACT ","B","04-Dec-1991 12:08:25","M901_AT_APC  ","Near Miss","O","125MM TANK MAIN GUN ", 1317
"n923am04","MOVEMENT TO CONTACT ","B","04-Dec-1991 12:08:41","M2_IFV        ","Near Miss","O","125MM TANK MAIN GUN ", 1397
```

Appendix

Mission Database Structure (Revised July 1993)

This Appendix documents the format of the NTC Mission Database. These data are collected on the SUN Workstations and stored in a real time format.

The RDMS (Range Data Measurement System) log is the raw events coming out of the field from the players' Micro-B units. This is the data stream that feeds the SUN Workstations.

Tables marked with an '*' are new tables from past versions of the mission databases. They replace or supersede previous tables.

Each Mission Database contains 16 FOXPRO tables:

(1) Mission Identification Table (MID),
* (2) Element State Initialization Table (ESIT),
* (3) Element State Update Table (ESUT),
* (4) Task Organization Table (TASK-ORG),
* (5) Symbol type table (SYMBOL),
* (6) Platform type table (PLATFORM),
* (7) Weapon type table (WEAPON),
(8) Firing Event Table (FET),
(9) Pairing Event Table (PET),
(10) Communication Table (CT),
(11) Ground Player Position Location Table (GPLT),
(12) Air Player Position Location Table (APLT),
* (13) Fratricide Table (FRAT),
(14) IFCAS Missions Fired Table (IFMF),
(15) IFCAS Casualties Table (IFCT),
(16) Minefield Casualties Table (MCT),

The following twelve tables exist for each NTC Rotation:

* (17) Control Measure Master Table (MASTER),
* (18) Arc's used in control measure graphic (ARC),
* (19) Circle's used in control measure graphic (CIRCLE),
* (20) Ellipse's used in control measure graphic (ELLIPSE),
* (21) Point's used in control measure graphic (POINT),
* (22) Line's used in control measure graphic (LINE),
* (23) Polyline's used in control measure graphic (POLYLINE),
* (24) Polygon's used in control measure graphic (POLYGON),
* (25) Rectangle's used in control measure graphic (RECTANGLE) and
* (26) Text used in control measure graphic (TEXT).
(27) IFCAS Target Table(IFTT),
(28) IFCAS Target Group Table (IFGT),

The table types and their compositions were chosen to allow for the inclusion of the maximum amount of information in a format that will facilitate access for the kinds of research issues that have been defined to date. The table descriptions have been purposely kept as simple as possible to allow review of the structure and content without overwhelming the reviewer with reams of documentation.

A rotation at the NTC is a three week period of time when a task force (Armored and Mechanized units) trains at Ft. Irwin, California. These segments are After Action Reviews (AARs), battles such as Deliberate Attacks, Defend in Sector, Defend Battle Position and Movement to Contact, etc. Each Mission-level segment is a candidate for a unique research database.

A separate database is generated for each mission segment. The database name is an eight-character code constructed as follows:

Character 1 - For the mission databases derived from the National Training Center at Ft. Irwin, Ca, 'N'. For mission databases derived from data from the Joint Readiness Training Center at Ft. Chaffee, Ark. 'J'.

Characters 2, 3 - Year of the Rotation.

Character 4 - A single hexadecimal digit representing the Rotation number. It ranges from 1 to D (for the twelve rotations usually scheduled in a fiscal year period)

Characters 5, 6 - One of the following codes representing the type of task force:

A_ Armor 1
R_ Armor 2
M_ Mech Inf 1
E_ Mech Inf 2
I_ Infantry 1
N_ Infantry 2
C_ Cavalry 1
V_ Cavalry 2
T_ Air Assault
L_ Light / Ranger
S_ Special Forces
AM Armor / Mech
IM Infantry / Mech
AC Armor / Cavalry
ZA More than 2 TF's
_A Armor 1
_R Armor 2
_M Mech Inf 1
_E Mech Inf 2
_I Infantry 1
_N Infantry 2
_C Cavalry 1

_V Cavalry 2
_T Air Assault
_L Light / Ranger
_S Special Force

Characters 7, 8 - Day of the month of the exercise began on.

Description of NTC Mission Database Tables

This section describes the contents of each table in the Mission Database. It includes the explicit layout, element by element, for each of the 27 tables.

Table A-1

Mission Identification Table (MID)

Name	Description	Format
phase_name	NTC assigned training exercise name	20 Char
phase_type	Pre-determined training mission type	30 Char
Starting	date-time at start of mission	20 Char
ending	date-time at end of mission	20 Char
log_rate	time in seconds between ground player position/location records	Integer
dbname	database name	10 Char

phase_name	phase_type	starting	ending	log_ra	dbname
T_5_DATK_IV	MOVEMENT TO CONTACT	10-Sep-1992 05:35:00	10-Sep-1992 09:47:39	300	N92CAL10

Table A-2

CT (communications table)

Name	Description	Format
time	date-time of communications exception	20 Char
lpn	logical player number of radio user	Integer
pid	player bumper number	8 Char
side	side of radio user	1 Char
x	UTM x coordinate	Integer
y	UTM y coordinate	Integer
duration	number of seconds radio button held down	Integer
net	radio net in use (1 or 2)	Integer
transmission		Integer

time	lpn	pid	side	x	y	durati	net	transmission
10-Sep-1992 07:22:10	1173	CD23	B	58020	113748	147	2	=> 55 seconds

Table A-3

APLT (air player position / location table)

Name	Description	Format
time	date-time of p/l reading	20 Char
lpn	logical player number	Integer
pid	player bumper number	8 Char
x	UTM x coordinate	Integer
y	UTM y coordinate	Integer
z	elevation in feet	Integer

time	lpn	pid	x	y	z
10-Sep-1992 05:35:13	1056	E459	32800	102896	5
10-Sep-1992 05:35:26	1056	E459	32788	102880	5
10-Sep-1992 05:35:33	1056	E459	32800	102896	4
10-Sep-1992 05:35:36	1056	E459	32796	102880	2
10-Sep-1992 05:35:44	1056	E459	32804	102900	2
10-Sep-1992 05:35:55	1056	E459	32808	102916	11
10-Sep-1992 05:36:00	1056	E459	32804	102932	19
10-Sep-1992 05:36:03	1056	E459	32816	102948	19
10-Sep-1992 05:36:08	1056	E459	32824	102964	9
10-Sep-1992 05:36:09	1056	E459	32824	102980	11
10-Sep-1992 05:36:13	1056	E459	32828	102980	57
10-Sep-1992 05:36:14	1056	E459	32832	102988	90
10-Sep-1992 05:36:16	1056	E459	32844	103012	90
10-Sep-1992 05:36:18	1056	E459	32848	103032	125
10-Sep-1992 05:36:20	1056	E459	32872	103068	157

Table A-4

GPLT (ground player position / location table)

Name	Description	Format
time	date-time of p/l reading	20 Char
lpn	logical player number	Integer
pid	player bumper number	8 Char
x	UTM x coordinate (meters)	Integer
y	UTM y coordinate (meters)	Integer

time	lpn	pid	x	y
10-Sep-1992 05:45:00	530	CS6	56348	112100
10-Sep-1992 05:45:00	535	613	52204	117700
10-Sep-1992 05:45:00	537	A23	29855	101250
10-Sep-1992 05:45:00	538	HQ50	4230	86875
10-Sep-1992 05:45:00	548	SS4	45430	131125
10-Sep-1992 05:45:00	568	127	29328	102616

Table A-5

FET (fire event table)

Name	Description	Format
time	date-time of fire event	20 Char
lpn	logical player number	Integer
pid	player bumper number	8 Char
side	side of firer	1 Char
x	UTM x coordinate (meters)	Integer
y	UTM y coordinate (meters)	Integer
weapon	type of weapon fired (see WEAPON table for definitions)	Integer

time	lpn	pid	side	x	y	weapon
10-Sep-1992 05:40:23	1293	SE42	B	37376	127688	60
10-Sep-1992 05:48:18	1184	CD55	B	46184	112092	31
10-Sep-1992 05:48:40	1184	CD55	B	46184	112092	31
10-Sep-1992 05:50:17	1184	CD55	B	46184	112092	31
10-Sep-1992 05:59:02	1240	CB66	B	49488	110044	3
10-Sep-1992 05:59:16	1240	CB66	B	49488	110044	3
10-Sep-1992 06:02:36	1195	C23C	B	48424	112060	40
10-Sep-1992 06:02:49	1195	C23C	B	48424	112060	52
10-Sep-1992 06:03:03	1195	C23C	B	48424	112060	52

Table A-6

PET (pairing event table)

Name	Description	Format
time	date-time of pairing event	20 Char
tlpn	target logical player number	Integer
tpid	target player bumper number	8 char
tside	target side	1 char
tx	target UTM x coordinate (meters)	Integer
ty	target UTM y coordinate (meters)	Integer
pair_type	type of pairing	Integer
weapon	weapon type of firer	Integer
flpn	firer logical player number	Integer
fpid	firer player bumper number	8 Char
fside	firer side	1 Char
fx	firer UTM x coordinate (meters)	Integer
fy	firer UTM y coordinate (meters)	Integer
result	pairing result	10 Char
distance	distance in meters of engagement Format	Integer

time	tlpn	tpid	tside	tx	ty	pair_t	weapon	flpn	fpid	fside	fx	fy	result	distance
10-Sep-1992 08:31:04	1410	S33B	B	36148	129156	23	31	0			0	0	Hit	0
10-Sep-1992 08:31:28	1410	S33B	B	36148	129156	23	31	0			0	0	Hit	0
10-Sep-1992 08:33:16	1208	CB33	B	61004	119164	4	140	858	B14	O	61008	117788	Hit	1376
10-Sep-1992 08:33:45	858	B14	O	61008	117788	23	3	0			0	0	Hit	0
10-Sep-1992 08:33:50	858	B14	O	61008	117788	23	3	0			0	0	Hit	0
10-Sep-1992 08:34:01	1240	CB66	B	60924	119212	4	140	858	B14	O	61008	117788	Hit	1426
10-Sep-1992 08:34:07	1240	CB66	B	60924	119212	23	140	0			0	0	Kill	0
10-Sep-1992 08:34:13	1410	S33B	B	36148	129156	25	31	0			0	0	Kill	0
10-Sep-1992 08:34:16	1091	C50C	B	60960	119200	25	3	0			0	0	Near Miss	0

Table A-7

IFMF (indirect fire missions fired)

Name	Description	Format
time	date-time of fire mission	20 Char
lpn	logical player number of firing battery	Integer
pid	player bumper number of firing battery	30 Char
target	target name	8 Char
plan_id	name of firing plan	8 Char
side	side of firing battery	1 Char
battery_x	UTM x coordinate (meters)	Integer
battery_y	UTM y coordinate (meters)	Integer
weapon	weapon code of firing battery	Integer
shell	type of shell fired	15 Char
fuse	type of fuse used	15 Char
impact_x	UTM x coordinate (meters)	Integer
impact_y	UTM y coordinate (meters)	Integer

time	lpn	pid	target	plan_id	side	battery_x	battery_y	weapon	shell	fuse	impact_x	impact_y
10-Sep-1992 05:42:44	575	/1/B/1 /3	~~~~~~	W-05-FOF	B	46530	116500	21	High Explosive	Variable time	55200	115600
10-Sep-1992 06:48:10	550	F BATT MRLS	~~~~~~	W-05-FOF	B	43530	103200	24	High Explosive	Point Detonator	61300	120100
10-Sep-1992 06:48:52	550	F BATT MRLS	~~~~~~	W-05-FOF	B	43530	103200	24	High Explosive	Point Detonator	61300	120500
10-Sep-1992 06:55:00	550	F BATT MRLS	~~~~~~	W-05-FOF	B	43530	103200	24	High Explosive	Point Detonator	63300	119700
10-Sep-1992 06:55:03	550	F BATT MRLS	~~~~~~	W-05-FOF	B	43530	103200	24	High Explosive	Point Detonator	61600	119900
10-Sep-1992 06:55:10	550	F BATT MRLS	~~~~~~	W-05-FOF	B	43530	103200	24	High Explosive	Point Detonator	61300	120500

Table A-8

IFCT (indirect fire casualty table)

Name	Description	Format
time	date-time of indirect fire casualty	20 Char
lpn	logical player number of casualty	Integer
pid	player bumper number of casualty	8 Character
target	pre-planned target name	8 Character
plan_id	plan-id of indirect fire mission	8 Character
side	side of casualy	1 Char
x	UTM x coordinate (meters)	Integer
y	UTM y coordinate (meters)	Integer

time	lpn	pid	target	plan_id	side	x	y
05-May-1992 07:57:43	636	MP19	~~~~~~	W03-FOF	O	48716	97572

Table A-9

ESIT (element state initialization table)

Name	Description	Format
lpn	logical player number	Integer
bunit	Micro-B transmitter number	Integer
player_type	type of element	10 Char
	player bumper number or unit designation	30 Char
nhlu	next higher line unit	Integer
nhe	next higher element	Integer
nle	next lower element	Integer
sibling	sibling or sister element	Integer
instrument	Instrumented or Un-instrumented	1 Char
pl_status	Position/Location status	5 Char
rdms	player type for range data measurement system	15 Char
battle_status	element status at start of training exercise	15 Char
side	element's side code	1 Char
echelon	elements echelon	10 Char
weapon_1	weapon code # 1 (see weapon code table)	Integer
fic_1	firer's laser code # 1	Integer
weapon_2	Weapon code # 2 (see weapon code table)	Integer
fic_2	firer's laser code # 2	Integer
weapon_3	weapon code # 3 (see weapon code table)	Integer
fic_3	firer's laser code # 3	Integer
platform	vehicle platform weapons are mounted on (see platform table)	Integer
mopp_level	one through four	6 Char
symbol	display symbol used for element	Integer

Element State Initialization Table

lpn	bunit	player_typid	pid	nhlu	nhe	nle	siblin	instru	pl_sta	rdms	battle_status	side	echelon	weapon	fic_1	weapon	fic_2	weapon	fic_3	platfo	mopp_l	symbol
577	-1	Unit	1-23rd CAV	-1	-1	572	1589	U	Found	don't track	Operational	B	Battalion	3	0	3	0	3	0	1	MOPP1	2
578	-1	Undefined	DA2	581	579	0	0	U	Found	ground vehicle	Operational	B	Undefined	21	0	3	0	3	0	101	MOPP 1	2
579	-1	Unit	/2/A/1 /3	588	588	578	575	U	Found	don't track	Operational	B	Section	3	0	3	0	3	0	1	MOPP 1	2
580	-1	Undefined	DA1	581	581	0	0	U	Found	ground vehicle	Operational	B	Undefined	21	0	3	0	3	0	101	MOPP 1	2
581	-1	Unit	/1/A/1 /3	588	588	580	579	U	Found	don't track	Operational	B	Section	3	0	3	0	3	0	1	MOPP 1	2
582	619	Undefined	137	1023	989	0	670	I	Lost	ground vehicle	Operational	O	Undefined	150	0	161	0	170	0	170	MOPP 1	2
583	-1	Undefined	RADAR2	588	588	0	581	U	Found	ground vehicle	Operational	B	Undefined	23	0	3	0	3	0	103	MOPP 1	2
584	999	Undefined	SEV2	1158	1158	0	1211	U	Found	ground vehicle	Operational	B	Undefined	50	0	3	0	3	0	42	MOPP 1	2
585	-1	Unit	RFLD	588	640	0	638	U	Found	don't track	Operational	B	Section	3	0	3	0	3	0	1	MOPP 1	2
586	-1	Unit	RCBT	640	640	0	585	U	Found	don't track	Operational	B	Section	3	0	3	0	3	0	1	MOPP 1	2
587	-1	Unit	DSTOC	588	588	0	583	U	Found	don't track	Operational	B	Section	3	0	3	0	3	0	1	MOPP 1	2
588	-1	Unit	1-3 FA	692	692	587	552	U	Found	don't track	Operational	B	Battalion	3	0	3	0	3	0	1	MOPP 1	2
589	353	Undefined	21I	-1	-1	0	0	I	Lost	ground vehicle	Operational	W	Undefined	3	0	3	0	3	0	12	MOPP 1	2
590	-1	Undefined	RC2	591	591	0	0	U	Found	ground vehicle	Operational	B	Undefined	21	0	3	0	3	0	101	MOPP 1	2
591	-1	Unit	/2/C/2 /637	640	640	590	0	U	Found	don't track	Operational	B	Section	3	0	3	0	3	0	1	MOPP 1	2
592	882	Undefined	M9275	577	577	0	538	I	Lost	ground vehicle	Operational	B	Undefined	31	0	52	0	50	0	32	MOPP 1	2
593	-1	Undefined	RC1	594	594	0	0	U	Found	ground vehicle	Operational	B	Undefined	21	0	3	0	3	0	101	MOPP 1	2
594	-1	Unit	/1/C/2 /637	640	640	593	591	U	Found	don't track	Operational	B	Section	3	0	3	0	3	0	1	MOPP 1	2
595	-1	Undefined	RB2	598	598	0	0	U	Found	ground vehicle	Operational	B	Undefined	21	0	3	0	3	0	101	MOPP 1	2
596	924	Undefined	21U	-1	-1	0	602	I	Found	ground vehicle	Operational	W	Undefined	3	0	3	0	3	0	12	MOPP 1	2
597	-1	Undefined	LOA14	618	618	0	0	U	Found	ground vehicle	Operational	O	Undefined	124	0	3	0	3	0	214	MOPP 1	2
598	-1	Unit	/2/B/2 /637	640	640	595	594	U	Found	don't track	Operational	B	Section	3	0	3	0	3	0	1	MOPP 1	2
599	-1	Undefined	RB1	611	611	0	0	U	Found	ground vehicle	Operational	B	Undefined	21	0	3	0	3	0	101	MOPP 1	2
600	962	Undefined	21J	-1	-1	0	571	I	Lost	ground vehicle	Operational	W	Undefined	3	0	3	0	3	0	12	MOPP 1	2
601	571	Undefined	21G	-1	-1	0	661	I	Lost	ground vehicle	Operational	W	Undefined	3	0	3	0	3	0	12	MOPP 1	2
602	556	Undefined	21V	-1	-1	0	807	I	Found	ground vehicle	Operational	W	Undefined	3	0	3	0	3	0	12	MOPP 1	2
603	921	Undefined	21E	-1	-1	0	809	I	Lost	ground vehicle	Operational	W	Undefined	3	0	3	0	3	0	12	MOPP 1	2
604	1003	Undefined	627	948	948	0	885	I	Lost	ground vehicle	Operational	O	Undefined	150	0	161	0	170	0	170	MOPP 1	2
605	-1	Undefined	730	998	615	0	617	U	Found	ground vehicle	Operational	O	Undefined	172	0	3	0	3	0	174	MOPP 1	2
606	998	Undefined	626	948	948	0	604	I	Found	ground vehicle	Operational	O	Undefined	150	0	161	0	170	0	170	MOPP 1	2
607	-1	Unit	2S1 Sec 2	562	562	609	610	U	Found	don't track	Operational	O	Section	3	0	3	0	3	0	1	MOPP 1	2
608	956	Undefined	617	948	948	0	648	I	Lost	ground vehicle	Operational	O	Undefined	150	0	161	0	170	0	170	MOPP 1	2
609	-1	Undefined	532	607	607	0	0	U	Found	ground vehicle	Operational	O	Undefined	120	0	3	0	3	0	210	MOPP 1	2
610	-1	Unit	2S1 Sec 3	562	562	614	621	U	Found	don't track	Operational	O	Section	3	0	3	0	3	0	1	MOPP 1	2
611	-1	Unit	/1/B/2 /637	640	640	599	598	U	Found	don't track	Operational	B	Section	3	0	3	0	3	0	1	MOPP 1	2
612	-1	Undefined	RA2	628	628	0	0	U	Found	ground vehicle	Operational	B	Undefined	21	0	3	0	3	0	101	MOPP 1	2
613	-1	Undefined	8INCH2	539	539	0	0	U	Found	ground vehicle	Operational	B	Undefined	23	0	3	0	3	0	103	MOPP 1	2
614	-1	Undefined	533	610	610	0	0	U	Found	ground vehicle	Operational	O	Undefined	120	0	3	0	3	0	210	MOPP 1	2
615	-1	Unit	3rd Plt/AT Co	619	884	605	0	U	Found	don't track	Operational	O	Platoon	3	0	3	0	3	0	1	MOPP 1	2
617	991	Undefined	731	615	615	0	944	I	Lost	ground vehicle	Operational	O	Undefined	172	0	3	0	3	0	174	MOPP 1	2
618	-1	Unit	122mm MRL LOA14	818	818	597	1082	U	Found	don't track	Operational	O	Battalion	3	0	3	0	3	0	1	MOPP 1	2
619	-1	Undefined	E682	1538	1267	0	536	U	Found	helicopter	Operational	B	Undefined	3	0	3	0	3	0	115	MOPP 1	2
620	551	Undefined	21D	-1	-1	0	603	I	Lost	ground vehicle	Operational	W	Undefined	3	0	3	0	3	0	12	MOPP 1	2
621	-1	Unit	2S1 Sec 4	562	562	624	625	U	Found	don't track	Operational	O	Section	3	0	3	0	3	0	1	MOPP 1	2
622	874	Undefined	625	948	948	0	606	I	Found	ground vehicle	Operational	O	Undefined	150	0	161	0	170	0	170	MOPP 1	2
623	888	Undefined	624	948	948	0	622	I	Lost	ground vehicle	Operational	O	Undefined	150	0	161	0	170	0	170	MOPP 1	2
624	-1	Undefined	534	621	621	0	0	U	Found	ground vehicle	Operational	O	Undefined	120	0	3	0	3	0	210	MOPP 1	2
625	-1	Unit	2S1 Sec 5	562	562	708	712	U	Found	don't track	Operational	O	Section	3	0	3	0	3	0	1	MOPP 1	2
626	-1	Undefined	Z15	1412	805	0	1483	U	Found	manpack	Operational	O	Undefined	161	0	3	0	3	0	192	MOPP 1	2
627	-1	Undefined	Z14	1412	805	0	626	U	Found	manpack	Operational	O	Undefined	161	0	3	0	3	0	192	MOPP 1	2
628	-1	Unit	/2/A/2 /637	640	640	612	611	U	Found	don't track	Operational	B	Section	3	0	3	0	3	0	1	MOPP 1	2
629	691	Undefined	225	992	992	0	976	I	Found	ground vehicle	Operational	O	Undefined	150	0	161	0	170	0	170	MOPP 1	2
630	820	Undefined	21B	-1	-1	0	620	I	Lost	ground vehicle	Operational	W	Undefined	3	0	3	0	3	0	12	MOPP 1	2
631	-1	Undefined	RA1	635	635	0	0	U	Found	ground vehicle	Operational	B	Undefined	21	0	3	0	3	0	101	MOPP 1	2
632	760	Undefined	351	997	997	0	0	I	Lost	ground vehicle	Operational	O	Undefined	150	0	161	0	170	0	170	MOPP 1	2
633	536	Undefined	451	771	771	0	0	I	Lost	ground vehicle	Operational	O	Undefined	151	0	161	0	172	0	171	MOPP 1	2
634	-1	Undefined	LOA6	817	817	0	0	U	Found	ground vehicle	Operational	O	Undefined	120	0	3	0	3	0	210	MOPP 1	2
635	-1	Unit	/1/A/2 /637	640	640	631	628	U	Found	don't track	Operational	B	Section	3	0	3	0	3	0	1	MOPP 1	2

Table A-10

Element State Update Table

Name	Description	Format
time	date-time of element update	20 Char
lpn	logical player number	Integer
bunit	Micro-B identification number	Integer
player_type	element type	10 Char
pid	element bumper number	30 Char
nhlu	next higher line unit	Integer
nhe	next higher element	Integer
nle	next lower element	Integer
sibling	sibling (sister) element	Integer
instrument	Instrumented or Un-instrumented	1 Char
pl_status	position/location status	5 Char
rdms	player type for range data measurement status	15 Char
battle_status	element's current status	15 Char
side	element side	1 Char
echelon	echelon of element	10 Char
weapon_1	player weapon # 1 (see weapon table for description)	Integer
fic_1	players laser code for weapon # 1	Integer
weapon_2	player weapon # 2 (see weapon table for description)	Integer
fic_2	player laser code for weapon # 2	Integer
weapon_3	player weapon # 3 (see weapon table for description)	Integer
fic_3	player laser code for weapon # 3	Integer
platform	vehicle type of element (see platform table for descriptions)	6 Char
mopp_level	MOPP level 1 through 4	Integer
symbol	symbol used for displaying element on AHSWS	

Element State Update Table

time	lpn	bunit	player_typid	pid	nhlu	nhe	nle	siblin	instru	pl_sta	rdms	battle_status	side	echelon	weapon	fic_1	weapon	fic_2	weapon	fic_3	platfo	mopp_l	symbol
10-Sep-1992 05:35:18	1196	563	Undefined	C21C	1197	1197	0	1194	U	Found	ground vehicle	Operational	B	Undefined	40	0	52	0	60	0	40	MOPP 1	2
10-Sep-1992 05:35:24	1532	931	Undefined	TCF	1121	1121	0	0	I	Found	air defense	Operational	B	Undefined	72	0	3	0	3	0	51	MOPP 1	2
10-Sep-1992 05:35:24	1533	785	Undefined	TBF	1131	1131	0	1124	I	Lost	air defense	Operational	B	Undefined	72	0	3	0	3	0	51	MOPP 1	2
10-Sep-1992 05:36:12	1168	622	Undefined	CD33	1171	1171	0	1167	U	Found	ground vehicle	Operational	B	Undefined	31	0	52	0	50	0	32	MOPP 1	2
10-Sep-1992 05:36:24	756	641	Undefined	GO6	1388	1388	0	1535	I	Lost	manpack	Operational	B	Undefined	3	0	3	0	3	0	60	MOPP 1	2
10-Sep-1992 05:36:42	1532	931	Undefined	TCF	1121	1121	0	0	I	Found	air defense	Operational	B	Undefined	72	0	3	0	3	0	51	MOPP 1	2
10-Sep-1992 05:36:54	1234	148	Undefined	CB13	1237	1237	0	1233	I	Found	ground vehicle	Operational	B	Undefined	31	0	52	0	50	0	32	MOPP 1	2
10-Sep-1992 05:37:44	1250	1011	Undefined	CA23	1253	1253	0	1251	I	Found	ground vehicle	Operational	B	Undefined	31	0	52	0	50	0	32	MOPP 1	2
10-Sep-1992 05:38:21	1268	919	Undefined	SB12	1266	1266	0	1395	I	Found	ground vehicle	Operational	B	Undefined	40	0	52	0	60	0	40	MOPP 1	2
10-Sep-1992 05:38:25	600	962	Undefined	21J	-1	-1	0	571	I	Lost	ground vehicle	Operational	W	Undefined	3	0	3	0	3	0	12	MOPP 1	2
10-Sep-1992 05:38:25	1096	594	Undefined	CS10	727	727	0	0	I	Lost	manpack	Operational	B	Undefined	60	0	3	0	3	0	65	MOPP 1	2
10-Sep-1992 05:38:25	1128	366	Undefined	TB1	1131	1131	0	1126	I	Lost	manpack	Operational	B	Undefined	61	0	3	0	3	0	66	MOPP 1	2
10-Sep-1992 05:40:35	1172	575	Undefined	CD24	1176	1176	0	0	U	Found	ground vehicle	Operational	B	Undefined	31	0	52	0	50	0	32	MOPP 1	2
10-Sep-1992 05:40:36	1088	345	Undefined	TEV1	1149	1149	0	1087	I	Found	ground vehicle	Operational	B	Undefined	3	0	3	0	3	0	133	MOPP 1	2
10-Sep-1992 05:40:46	1175	535	Undefined	CD21	1176	1176	0	1174	U	Found	ground vehicle	Unallocated CGK	B	Undefined	31	0	52	0	50	0	32	MOPP 1	2
10-Sep-1992 05:41:24	1118	701	Undefined	TC2	1121	1121	0	1117	I	Lost	manpack	Operational	B	Undefined	61	0	3	0	3	0	66	MOPP 1	2
10-Sep-1992 05:41:33	1198	819	Undefined	CC1	1202	1202	0	0	U	Found	manpack	Operational	B	Undefined	3	0	3	0	3	0	60	MOPP 1	2
10-Sep-1992 05:41:40	535	731	Undefined	613	948	948	0	863	I	Found	ground vehicle	Operational	O	Undefined	161	0	3	0	3	0	173	MOPP 1	2
10-Sep-1992 05:42:46	1196	563	Undefined	C21C	1197	1197	0	1194	U	Found	ground vehicle	Operational	B	Undefined	40	0	52	0	60	0	40	MOPP 1	2
10-Sep-1992 05:43:24	1252	905	Undefined	CA21	1253	1253	0	1249	U	Found	ground vehicle	Operational	B	Undefined	31	0	52	0	50	0	32	MOPP 1	2
10-Sep-1992 05:43:24	1346	781	Undefined	ST10	1337	1335	0	1337	I	Lost	air defense	Operational	B	Undefined	72	0	3	0	3	0	51	MOPP 1	2
10-Sep-1992 05:43:55	922	772	Undefined	MP11	897	898	0	923	U	Found	manpack	Combat Loss	O	Undefined	3	0	3	0	3	0	190	MOPP 1	2
10-Sep-1992 05:44:01	949	281	Undefined	610	948	948	0	950	U	Found	ground vehicle	Combat Loss	O	Undefined	161	0	3	0	3	0	173	MOPP 1	2
10-Sep-1992 05:44:19	1252	905	Undefined	CA21	1253	1253	0	1249	I	Found	ground vehicle	Operational	B	Undefined	31	0	52	0	50	0	32	MOPP 1	2
10-Sep-1992 05:44:22	1013	266	Undefined	E371	1065	1404	0	1059	I	Found	helicopter	Operational	B	Undefined	82	0	80	0	83	0	110	MOPP 1	2
10-Sep-1992 05:44:28	649	692	Undefined	619	948	948	0	951	U	Found	ground vehicle	Combat Loss	O	Undefined	150	0	161	0	170	0	170	MOPP 1	2
10-Sep-1992 05:44:31	954	500	Undefined	055	948	948	0	953	U	Found	ground vehicle	Combat Loss	O	Undefined	161	0	3	0	3	0	173	MOPP 1	2
10-Sep-1992 05:44:33	1328	920	Undefined	CEN3	1325	1325	0	0	I	Found	ground vehicle	Operational	B	Undefined	50	0	3	0	3	0	42	MOPP 1	2
10-Sep-1992 05:44:38	1175	535	Undefined	CD21	1176	1176	0	1174	U	Found	ground vehicle	Operational	B	Undefined	31	0	52	0	50	0	32	MOPP 1	2
10-Sep-1992 05:44:43	1354	272	Undefined	ST2	1347	1347	0	1355	I	Found	air defense	Operational	B	Undefined	72	0	3	0	3	0	51	MOPP 1	2
10-Sep-1992 05:44:44	1290	659	Undefined	CFD2	728	728	0	0	U	Found	ground vehicle	Operational	B	Undefined	12	0	3	0	3	0	91	MOPP 1	2
10-Sep-1992 05:45:04	1353	735	Undefined	ST1	1347	1347	0	1354	I	Found	air defense	Operational	B	Undefined	72	0	3	0	3	0	51	MOPP 1	2
10-Sep-1992 05:45:08	1355	929	Undefined	ST3	1347	1347	0	1356	I	Found	air defense	Operational	B	Undefined	72	0	3	0	3	0	51	MOPP 1	2
10-Sep-1992 05:45:12	1356	1023	Undefined	ST4	1347	1347	0	0	I	Found	air defense	Operational	B	Undefined	72	0	3	0	3	0	51	MOPP 1	2
10-Sep-1992 05:45:12	1056	604	Undefined	E459	1065	1404	0	1123	I	Found	helicopter	Operational	B	Undefined	82	0	80	0	83	0	110	MOPP 1	2
10-Sep-1992 05:45:24	1086	949	Undefined	TEV3	1149	1149	0	1063	I	Found	ground vehicle	Operational	B	Undefined	3	0	3	0	3	0	133	MOPP 1	2
10-Sep-1992 05:45:24	1167	705	Undefined	CD34	1171	1171	0	0	U	Found	ground vehicle	Operational	B	Undefined	31	0	52	0	50	0	32	MOPP 1	2
10-Sep-1992 05:45:24	1283	923	Undefined	CA55	1285	1285	0	1282	I	Found	ground vehicle	Operational	B	Undefined	31	0	52	0	50	0	32	MOPP 1	2
10-Sep-1992 05:45:50	1283	923	Undefined	CA55	1285	1285	0	1282	I	Found	ground vehicle	Operational	B	Undefined	31	0	52	0	50	0	32	MOPP 1	2
10-Sep-1992 05:45:59	1367	940	Undefined	ET1	1366	1366	0	1368	I	Found	manpack	Operational	B	Undefined	3	0	3	0	3	0	60	MOPP 1	2
10-Sep-1992 05:46:24	1063	341	Undefined	TEV4	1149	1149	0	0	U	Found	ground vehicle	Operational	B	Undefined	3	0	3	0	3	0	133	MOPP 1	2
10-Sep-1992 05:46:24	1087	809	Undefined	TEV2	1149	1149	0	1086	I	Found	ground vehicle	Operational	B	Undefined	3	0	3	0	3	0	133	MOPP 1	2
10-Sep-1992 05:46:24	1169	942	Undefined	CD32	1171	1171	0	1168	U	Found	ground vehicle	Operational	B	Undefined	31	0	52	0	50	0	32	MOPP 1	2
10-Sep-1992 05:46:24	1397	863	Undefined	TFAS	1149	1149	0	1398	I	Found	ground vehicle	Operational	B	Undefined	3	0	3	0	3	0	133	MOPP 1	2
10-Sep-1992 05:46:24	1398	968	Undefined	TMAS	1149	1149	0	1088	I	Found	ground vehicle	Operational	B	Undefined	3	0	3	0	3	0	133	MOPP 1	2
10-Sep-1992 05:47:10	1126	982	Undefined	TB2	1131	1131	0	1125	U	Found	manpack	Combat Loss	B	Undefined	61	0	3	0	3	0	66	MOPP 1	2
10-Sep-1992 05:47:24	873	797	Undefined	B15	1021	795	0	940	I	Lost	ground vehicle	Operational	O	Undefined	140	0	161	0	160	0	160	MOPP 1	2
10-Sep-1992 05:47:24	1238	893	Undefined	CFTB	1241	1241	0	0	I	Found	ground vehicle	Operational	B	Undefined	50	0	3	0	3	0	42	MOPP 1	2
10-Sep-1992 05:47:24	1284	283	Undefined	CA66	1285	1285	0	1283	I	Lost	ground vehicle	Operational	B	Undefined	31	0	52	0	50	0	32	MOPP 1	2
10-Sep-1992 05:47:53	732	842	Undefined	CTOC	1291	1291	0	730	U	Found	ground vehicle	Operational	B	Undefined	50	0	3	0	3	0	42	MOPP 1	2
10-Sep-1992 05:48:01	711	679	Undefined	CUMP	705	705	0	0	I	Found	ground vehicle	Operational	B	Undefined	50	0	3	0	3	0	42	MOPP 1	2
10-Sep-1992 05:48:09	1051	906	Undefined	E311	1065	1404	0	1053	I	Found	helicopter	Operational	B	Undefined	82	0	80	0	83	0	110	MOPP 1	2
10-Sep-1992 05:48:10	1411	857	Undefined	CEV1	1289	1289	0	1210	U	Found	ground vehicle	Operational	B	Undefined	50	0	3	0	3	0	42	MOPP 1	2
10-Sep-1992 05:48:12	1287	752	Undefined	CMAS	1289	1289	0	1288	I	Found	ground vehicle	Operational	B	Undefined	50	0	3	0	3	0	42	MOPP 1	2
10-Sep-1992 05:48:23	183	923	Undefined	CA55	1285	1285	0	1282	I	Found	ground vehicle	Operational	B	Undefined	31	0	52	0	50	0	32	MOPP 1	2

Table A-11

Indirect Fire Target Table

Name	Description	Format
tgt_idx	target sequence number	Integer
side	B(lue) or O(pfor)	1 Char
starting	date-time of target assignment	20 Char
ending	date-time of target de-	20 Char
target	assignment	6 Char
origin	target name	20 Char
definition	element which assigned target	12 Char
x	mission type	Integer
y	UTM x coordinate (meters)	Integer
	UTM y coordinate (meters)	

tgt_id	side	starting	ending	target	origin	definition	x	y
238	O	08-Oct-1991 23:53:21	09-Oct-1991 15:03:28		OPFOR 1-63	ARTY MDM	51000	100900
239	O	08-Oct-1991 23:53:21	09-Oct-1991 15:03:28	1002	OPFOR 1-63	ARTY MDM	49590	99550
240	O	08-Oct-1991 23:53:21	09-Oct-1991 15:03:28	1003	OPFOR 1-63	ARTY MDM	46000	98200
241	O	08-Oct-1991 23:53:21	09-Oct-1991 15:03:28	1004	OPFOR 1-63	ARTY MDM	46300	98350
242	O	08-Oct-1991 23:53:21	09-Oct-1991 15:03:28	1005	OPFOR 1-63	ARTY MDM	46550	98600
243	O	08-Oct-1991 23:53:21	09-Oct-1991 15:03:28	1006	OPFOR 1-63	ARTY MDM	44650	95500
266	O	08-Oct-1991 23:53:21	09-Oct-1991 15:03:28	1029	OPFOR 1-63	ARTY MDM	34550	98400
267	O	08-Oct-1991 23:53:21	09-Oct-1991 15:03:28	1030	OPFOR 1-63	ARTY MDM	34600	98000
268	O	08-Oct-1991 23:53:21	09-Oct-1991 15:03:28	1031	OPFOR 1-63	ARTY MDM	34600	97300
269	O	08-Oct-1991 23:53:21	09-Oct-1991 15:03:28	1032	OPFOR 1-63	ARTY MDM	33650	97250
272	O	08-Oct-1991 23:53:21	09-Oct-1991 15:03:28	1035	OPFOR 1-63	ARTY MDM	30150	96000
273	O	08-Oct-1991 23:53:21	09-Oct-1991 15:03:28	1036	OPFOR 1-63	ARTY MDM	30500	95750
274	O	08-Oct-1991 23:53:21	09-Oct-1991 15:03:28	1037	OPFOR 1-63	ARTY MDM	30850	95550
275	O	08-Oct-1991 23:53:21	09-Oct-1991 15:03:28	1038	OPFOR 1-63	ARTY MDM	30550	94550
276	O	08-Oct-1991 23:53:21	09-Oct-1991 15:03:28	1039	OPFOR 1-63	ARTY MDM	30600	94150
277	O	08-Oct-1991 23:53:21	09-Oct-1991 15:03:28	1040	OPFOR 1-63	ARTY MDM	30650	93750

Table A-12

Indirect Fire Group Table (IFGT)

Name	Description	Format
plan_idx	plan sequence number	Integer
side	B(lue) or O(pfor)	1 Char
starting	date-time of group target assignment	20 Char
ending	date-time of group target de-assignment	20 Char
designator		7 Char
target1	target # 1 name (in IFTT)	10 Char
target2	target # 2 name (in IFTT)	10 Char
target3	.	.
target4	.	.
target5	.	.
target6	.	.
target7	.	.
target8	.	.
target9	.	.
target10	target # 10 name (in IFTT)	10 Char

plan_i	side	starting	ending	designa	target1	target2	target3	target4	target5	target6	target7	target8	target9	target10
77	O	08-Oct-1991 23:58:53	09-Oct-1991 15:03:28	y19	2421005	2421005	2421005	0	0	0	0	0	0	0
78	O	08-Oct-1991 23:58:53	09-Oct-1991 15:03:28	y16	2491012	2491012	2491012	0	0	0	0	0	0	0
79	O	08-Oct-1991 23:58:53	09-Oct-1991 15:03:28	y14	2541017	2541017	2541017	218309	0	0	0	0	0	0
80	O	08-Oct-1991 23:58:53	09-Oct-1991 15:03:28	y13	2571020	2571020	2571020	0	0	0	0	0	0	0
81	O	08-Oct-1991 23:58:53	09-Oct-1991 15:03:28	y12	2611024	2611024	2611024	0	0	0	0	0	0	0
82	O	08-Oct-1991 23:58:53	09-Oct-1991 15:03:28	y11	2641027	2641027	2641027	0	0	0	0	0	0	0
83	O	08-Oct-1991 23:58:53	09-Oct-1991 15:03:28	y9	2671030	2671030	2671030	0	0	0	0	0	0	0
84	O	08-Oct-1991 23:58:53	09-Oct-1991 15:03:28	y8	2711034	2711034	2711034	0	0	0	0	0	0	0
85	O	08-Oct-1991 23:58:53	09-Oct-1991 15:03:28	y6	2741037	2741037	2741037	209605	0	0	0	0	0	0
86	O	08-Oct-1991 23:58:53	09-Oct-1991 15:03:28	y5	2771040	2771040	2771040	0	0	0	0	0	0	0
87	O	08-Oct-1991 23:58:53	09-Oct-1991 15:03:28	y4	2801043	2801043	2801043	0	0	0	0	0	0	0
88	O	08-Oct-1991 23:58:53	09-Oct-1991 15:03:28	y3	2831046	2831046	2831046	0	0	0	0	0	0	0
89	O	08-Oct-1991 23:58:53	09-Oct-1991 15:03:28	y2	2861049	2861049	2861049	0	0	0	0	0	0	0
90	O	08-Oct-1991 23:58:53	09-Oct-1991 15:03:28	y1	2891052	2891052	2891052	0605	0	0	0	0	0	0
91	O	09-Oct-1991 00:12:00	09-Oct-1991 15:03:28	y7	2921055	2921055	2921055	234325	0	0	0	0	0	0
92	O	09-Oct-1991 00:14:49	09-Oct-1991 15:03:28	non1	29633	29633	29633	0	0	0	0	0	0	0

Table A-13

Task Organization Table (TASK_ORG)

Name	Description	Format
element_desc	description of the element	20 Char
element_id	logical player number of element	Integer
side	B(lue) or O(pfor)	1 Char

element_desc	elemen	side
Engineer Co	543	O
1st Tank Co/3rd MRB	560	O
3rd Plt/AT Co	615	O
2nd Plt/AT Co	616	O
Sct Plt/4-17 IN	697	B
TOC/3-41 IN	703	B
Cbt Trains/3-67 AR	705	B
Chem Recon Plt	707	O
Sect 1 3-67 AR	709	B
Sct Plt/3-67 AR	727	B
Sect 2 3-67 AR	728	B
1st Tank Co/4th MRB	746	O
A/1-227th AVN	748	B
2nd Mech Co/4th MRB	755	O
Cbt Trns/3-41 IN	766	B
3rd Mech Co/4th MRB	771	O
HQ/4th MRB	774	O
HQ/1st MRB	785	O
1st Tank Co/1st MRB	792	O
1st Tank Co/2nd MRB	795	O
Sct Plt/3-41 IN	815	B
HQ/3rd MRB	846	O
2/B/1-67 AR	851	B

Table A-14

Minefield Casualty Table (MCT)

Name	Description	Format
time	date-time of casualty	20 Char
lpn	logical player number	Integer
pid	player bumper number	25 Char
side	B(lue) or O(pfor)	1 Char
x	UTM x coordinate (meters)	Integer
y	UTM y coordinate (meters)	Integer

time	lpn	pid	side	x	y

Table A-15

Control Measure Master Table (CM_MASTER)

Name	Description	Format
cm_index	index number of control number	Integer
starting	date-time of control measure activation	20 Char
ending	date-time of control measure deactivation	20 Char
side	B(lue) or O(pfor)	1 Char
echelon	echelon control measure pertains to	9 Char
bos	battle field operating system	3 Char
status	Current or Proposed	8 Char
arc	# of arc's used to draw control measure graphic	Integer
circle	# of circle's used to draw control measure graphic	Integer
ellipse	# of ellipse's used to draw control measure graphic	Integer
line	# of line segments used to draw control measure graphic	Integer
point	# of point's used to draw control measure graphic	Integer
polyline	# of polyline's used to draw control measure graphic	Integer
polygon	# of polygon's used to draw control measure graphic	Integer
rectangle	# of rectangles used to draw control measure graphic	Integer
text	# of text character's used to draw control measure graphic	Integer

Control Measure Master Table

cm_ind	starting	ending	side	echelon	bos	status	arc	circle	ellips	line	point	polyli	polygo	rectan	text
66	07-Sep-1992 00:01:00	10-Sep-1992 23:59:00	B	Brigade	ADA	Current	0	0	3	16	35	57	0	0	47
67	07-Sep-1992 00:01:00	10-Sep-1992 23:59:00	B	Brigade	ADA	Current	0	0	0	0	0	0	0	0	0
69	07-Sep-1992 00:01:00	10-Sep-1992 23:59:00	B	Brigade	Int	Current	0	0	0	0	0	9	0	0	13
70	07-Sep-1992 00:01:00	10-Sep-1992 23:59:00	B	Brigade	Int	Current	0	0	0	0	2	93	0	0	13
81	09-Sep-1992 00:01:00	10-Sep-1992 23:59:00	B	Brigade	Int	Current	0	8	0	0	0	10	0	0	0
82	09-Sep-1992 00:01:00	10-Sep-1992 23:59:00	B	Brigade	Int	Current	0	1	0	0	0	2	0	0	0
83	09-Sep-1992 00:01:00	10-Sep-1992 23:59:00	B	Brigade	Int	Current	0	0	0	0	0	1	0	0	0
84	09-Sep-1992 00:01:00	10-Sep-1992 23:59:00	B	Brigade	Int	Current	0	0	0	0	0	0	0	0	15
85	09-Sep-1992 00:01:00	12-Sep-1992 23:59:00	B	Brigade	Int	Current	0	0	0	8	2	118	0	0	0
87	09-Sep-1992 00:01:00	10-Sep-1992 23:59:00	B	Brigade	Int	Current	0	0	31	6	0	21	12	0	0
133	09-Sep-1992 00:01:00	10-Sep-1992 23:59:00	O	Division	FS	Current	0	2	0	0	0	0	0	0	0
134	10-Sep-1992 00:01:00	11-Sep-1992 23:59:00	B	Platoon	Air	Current	0	0	0	0	0	2	0	0	0
135	09-Sep-1992 00:01:00	10-Sep-1992 23:59:00	B	Section	Int	Current	0	0	0	0	0	3	4	0	0
159	08-Sep-1992 00:01:00	11-Sep-1992 23:59:00	O	Section	Air	Current	0	0	2	0	0	2	0	0	1
256	09-Sep-1992 00:01:00	11-Sep-1992 23:59:00	O	Section	ADA	Current	0	0	0	3	8	12	0	0	9
336	10-Sep-1992 05:40:00	10-Sep-1992 05:41:00	O	Section	CSS	Current	0	2	9	2	0	1	0	0	12
337	10-Sep-1992 06:00:00	10-Sep-1992 06:01:00	O	Section	CSS	Current	0	2	4	0	0	2	0	0	10
338	10-Sep-1992 06:11:00	10-Sep-1992 06:12:00	O	Section	CSS	Current	0	1	7	1	0	2	0	0	9
339	10-Sep-1992 06:25:00	10-Sep-1992 06:26:00	O	Section	CSS	Current	0	2	4	0	0	2	0	0	7
340	10-Sep-1992 06:36:00	10-Sep-1992 06:37:00	O	Section	CSS	Current	0	0	4	0	0	2	0	0	5
341	10-Sep-1992 06:46:00	10-Sep-1992 06:47:00	O	Section	CSS	Current	0	2	4	1	0	3	0	0	7
342	10-Sep-1992 07:04:00	10-Sep-1992 07:05:00	O	Section	CSS	Current	0	2	1	1	0	3	0	0	6
343	10-Sep-1992 07:19:00	10-Sep-1992 07:20:00	O	Section	CSS	Current	0	2	1	1	0	3	0	0	6
344	10-Sep-1992 07:33:00	10-Sep-1992 07:34:00	O	Section	CSS	Current	0	5	1	1	0	3	0	3	8
345	10-Sep-1992 07:52:00	10-Sep-1992 07:53:00	O	Section	CSS	Current	0	1	3	1	0	2	0	0	5
346	10-Sep-1992 08:08:00	10-Sep-1992 08:09:00	O	Section	CSS	Current	0	2	1	0	0	3	0	0	5
347	10-Sep-1992 08:25:00	10-Sep-1992 08:26:00	O	Section	CSS	Current	0	0	1	0	0	1	0	0	3
348	10-Sep-1992 00:01:00	11-Sep-1992 23:59:00	O	Company	FS	Current	0	0	0	0	0	0	2	0	0
349	10-Sep-1992 00:01:00	11-Sep-1992 23:59:00	O	Platoon	CSS	Current	0	0	4	3	0	5	0	0	21
350	10-Sep-1992 06:06:00	10-Sep-1992 06:07:00	O	Section	CSS	Current	0	0	6	0	0	1	0	0	6
354	10-Sep-1992 06:58:00	10-Sep-1992 06:59:00	O	Section	CSS	Current	0	0	0	1	0	0	0	0	4
445	07-Sep-1992 00:01:00	10-Sep-1992 23:59:00	B	Brigade	ADA	Current	0	0	0	0	0	0	0	0	0
447	07-Sep-1992 00:01:00	10-Sep-1992 23:59:00	B	Brigade	Int	Current	0	0	0	0	0	9	0	0	13
479	09-Sep-1992 00:01:00	10-Sep-1992 23:59:00	B	Battalion	Mnv	Current	0	4	16	12	24	6	6	4	3
495	09-Sep-1992 00:01:00	10-Sep-1992 23:59:00	B	Battalion	NU	Current	0	0	0	0	0	0	0	0	0
690	09-Sep-1992 00:01:00	10-Sep-1992 23:59:00	B	Battalion	Mnv	Current	0	0	3	0	0	2	0	0	0
691	09-Sep-1992 00:01:00	10-Sep-1992 23:59:00	B	Division	Mnv	Current	0	0	0	0	0	0	0	0	7
692	09-Sep-1992 00:01:00	10-Sep-1992 23:59:00	B	Section	Mnv	Current	0	0	0	20	0	21	0	0	0
693	09-Sep-1992 00:01:00	10-Sep-1992 23:59:00	B	Company	Mnv	Current	0	0	5	5	0	3	0	0	0
694	09-Sep-1992 00:01:00	10-Sep-1992 23:59:00	B	Division	Mnv	Current	0	0	0	0	0	0	0	0	26
695	09-Sep-1992 00:01:00	10-Sep-1992 23:59:00	B	Battalion	Mnv	Current	0	0	0	0	0	0	0	0	1
709	09-Sep-1992 00:01:00	12-Sep-1992 13:59:00	B	Brigade	Mnv	Current	0	0	0	1	0	37	0	0	23
727	09-Sep-1992 00:01:00	12-Sep-1992 13:59:00	B	Brigade	Mnv	Current	0	0	0	4	0	0	0	0	14
752	09-Sep-1992 00:01:00	10-Sep-1992 23:59:00	O	Section	ADA	Proposed	0	0	39	8	0	26	4	0	0
753	09-Sep-1992 00:01:00	10-Sep-1992 23:59:00	B	Section	ADA	Proposed	0	5	2	0	0	15	8	0	0
848	09-Sep-1992 00:01:00	10-Sep-1992 23:59:00	B	Battalion	CSS	Current	2	3	2	16	9	8	0	0	32
849	09-Sep-1992 00:01:00	10-Sep-1992 23:59:00	B	Platoon	FS	Proposed	0	0	0	8	9	0	0	0	11
850	09-Sep-1992 00:01:00	10-Sep-1992 23:59:00	O	Section	Int	Current	0	2	0	0	0	18	0	0	14

Table A-16

Control Measure Arc Table (ARC)

Name	Description	Format
cm_index	control measure index	Integer
object	sequence number of arc	Integer
x1	UTM x coordinate (meters)	Integer
y1	UTM y coordinate (meters)	Integer
x2	UTM x coordinate (meters)	Integer
y2	UTM y coordinate (meters)	Integer
line_type	graphic representation	20 Char
color	arc color	15 Char

cm_ind	object	x1	y1	x2	y2	line_type	color
848	1	52070	114310	52070	114055	BASIC_LINE	red
848	2	51905	114040	51860	114340	BASIC_LINE	red
851	1	54305	119350	55265	119710	BASIC_LINE	white
851	2	56735	118960	55145	119725	BASIC_LINE	white
851	3	54200	118150	54200	118150	BASIC_LINE	white
851	4	54215	118180	55280	118435	BASIC_LINE	white
851	5	56480	118180	55205	118420	BASIC_LINE	white
851	6	54155	117565	55265	118270	BASIC_LINE	white
851	7	56525	117820	55205	118285	BASIC_LINE	white

852	1	59060	111775	59270	111595	BASIC_LINE	yellow	
852	2	59435	111835	59255	111610	BASIC_LINE	yellow	
852	3	59555	111940	59750	111895	BASIC_LINE	yellow	
852	4	58790	111310	58970	111295	BASIC_LINE	yellow	
852	5	60035	113410	60140	113425	BASIC_LINE	yellow	

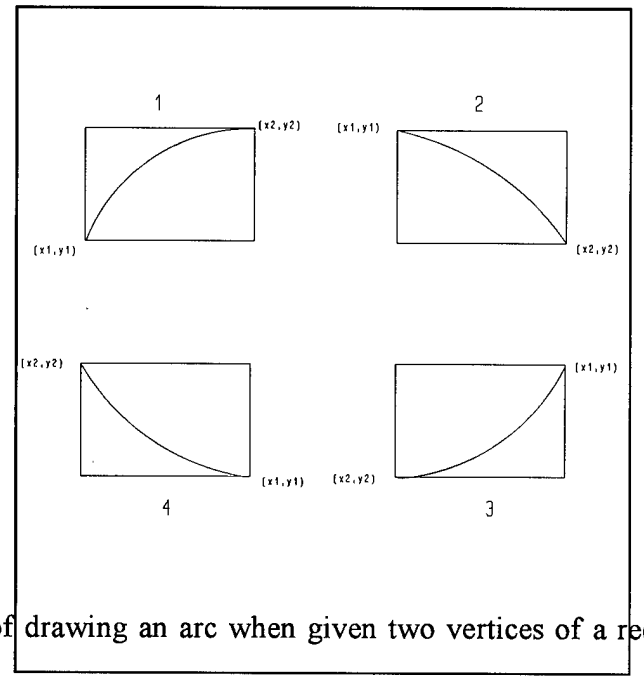


Figure A-1. Method of drawing an arc when given two vertices of a rectangle which inscribes the arc.

Table A-17

Control Measure Circle Table (CIRCLE)

Name	Description	Format
cm_index	control measure index	Integer
object	sequence of circle	Integer
x1	UTM x coordinate (meters) of circle center	Integer
y1	UTM y coordinate (meters) of circle center	Integer
x2	UTM x coordinate (meters) of a point on edge of circle	Integer
y2	UTM y coordinate (meters) of a point on edge of circle	Integer
line_type	outline graphic used	20 Char
fill_type	shading graphic used	20 Char
color	color of outline	20 Char
fill_color	color of shading	20 Char

cm_ind	object	x1	y1	x2	y2	line_type	fill_type	color	fill_color
81	1	55430	118570	55595	118570	BASIC_LINE	NONE	yellow	black
81	2	54680	117235	54890	117235	BASIC_LINE	NONE	yellow	black
81	3	55730	117670	55970	117670	BASIC_LINE	NONE	yellow	black
81	4	56435	117160	56660	117160	BASIC_LINE	NONE	yellow	black
81	5	57515	116425	57785	116425	BASIC_LINE	NONE	yellow	black
81	6	57875	116995	58055	116995	BASIC_LINE	NONE	yellow	black
81	7	61760	117355	61985	117355	BASIC_LINE	NONE	yellow	black
81	8	60335	118735	60545	118735	BASIC_LINE	NONE	yellow	black
82	1	57215	121435	57410	121435	BASIC_LINE	NONE	yellow	black
133	1	25145	117370	25145	117370	BASIC_LINE	THIN_DIAG_LINES	red	yellow
133	2	26030	117055	26915	117055	BASIC_LINE	THIN_DIAG_LINES	red	yellow
336	1	44285	112600	44435	112600	BASIC_LINE	SHADED	black	yellow
336	2	49310	111775	49505	111775	BASIC_LINE	SHADED	black	yellow
337	1	49280	111820	49490	111820	BASIC_LINE	SHADED	black	yellow
337	2	48020	112165	48275	112165	BASIC_LINE	SHADED	black	yellow
338	1	44830	112763	45055	112763	BASIC_LINE	SHADED	blue	white
339	1	54980	113338	55130	113338	BASIC_LINE	SHADED	black	yellow
339	2	52918	111525	53068	111525	BASIC_LINE	SHADED	black	yellow
341	1	61805	115800	62105	115800	BASIC_LINE	SHADED	blue	white
341	2	57980	117800	58230	117800	BASIC_LINE	SHADED	blue	white
342	1	59368	113875	59443	113875	BASIC_LINE	SHADED	black	yellow
342	2	59218	113338	59318	113338	BASIC_LINE	SHADED	black	yellow
343	1	59880	114275	59980	114275	BASIC_LINE	SHADED	black	yellow
343	2	59605	113525	59668	113525	BASIC_LINE	SHADED	black	yellow
344	1	59605	114463	59705	114463	BASIC_LINE	NONE	yellow	black
344	2	60155	115013	60243	115013	BASIC_LINE	NONE	yellow	black
344	3	59843	113800	59918	113800	BASIC_LINE	NONE	yellow	black
344	4	60118	114600	60205	114600	BASIC_LINE	SHADED	black	yellow
344	5	60018	112450	60105	112450	BASIC_LINE	SHADED	black	yellow
345	1	59343	113488	59430	113488	BASIC_LINE	SHADED	blue	white
346	1	61043	115413	61155	115413	BASIC_LINE	SHADED	black	yellow
346	2	60993	116075	61080	116075	BASIC_LINE	SHADED	black	yellow
479	1	29450	114350	29600	114350	THICK_LINE	SHADED	black	white
479	2	31450	114300	31600	114300	THICK_LINE	SHADED	black	white
479	3	31100	114450	31250	114450	THICK_LINE	SHADED	black	white
479	4	29400	114500	29550	114500	THICK_LINE	SHADED	black	white
753	1	41155	129325	41155	129325	BASIC_LINE	SHADED	yellow	green

Table A-18

Control Measure Ellipse Table (ELLIPSE)

Name	Description	Format
cm_index	control measure index	Integer
object	sequence of arc in graphic	Integer
x1	UTM x coordinate (meters) of rectangle vertices inscribing ellipse	Integer
y1	UTM y coordinate (meters) of rectangle vertices inscribing ellipse	Integer
x2	UTM x coordinate (meters) of rectangle vertices inscribing ellipse	Integer
y2	UTM y coordinate (meters) of rectangle vertices inscribing ellipse	Integer
line_type	outline used for graphic	20 Char
fill_type	shading used for graphic	20 Char
rotation	from 0^0 to 360^0	Integer
color	outline color	20 Char
fill_color	shading color	20 Char

cm_ind	object	x1	y1	x2	y2	line_type	fill_type	rotati	color	fill_color
66	1	46775	107365	47360	107665	DASHED_LINE	NONE	118	blue	black
66	2	46400	106075	47120	106420	DASHED_LINE	NONE	95	blue	black
66	3	48320	112990	49205	113410	DASHED_LINE	NONE	150	blue	black
159	1	49155	110100	50930	111100	BASIC_LINE	SHADED	107	green	white
159	2	47355	113200	49630	114800	BASIC_LINE	SHADED	62	green	white
336	1	39185	109675	41990	110425	THICK_LINE	NONE	3	white	black
336	2	40040	111625	40955	112300	THICK_LINE	NONE	104	white	black
336	3	48815	110620	51425	112465	THICK_LINE	NONE	143	white	black
336	4	46175	114070	46805	114580	THICK_LINE	NONE	5	white	black
336	5	49670	115030	50345	115825	THICK_LINE	NONE	50	white	black
336	6	52685	117010	54830	118315	THICK_LINE	NONE	157	white	black
336	7	44405	113305	46385	113890	THICK_LINE	NONE	173	white	black
336	8	44045	112615	44930	113185	THICK_LINE	NONE	157	white	black
336	9	45275	112450	47015	113050	THICK_LINE	NONE	158	yellow	black
337	1	49640	110710	53240	112450	THICK_LINE	NONE	159	white	black
337	2	52670	117040	54785	118315	THICK_LINE	NONE	158	white	black
337	3	50045	115555	50840	116065	THICK_LINE	NONE	143	white	black
337	4	40160	111895	40790	112420	THICK_LINE	NONE	177	white	black
338	1	54643	110013	57005	111738	THICK_LINE	NONE	17	white	black
338	2	51168	111963	53268	112650	THICK_LINE	NONE	10	white	black
338	3	49768	111750	50768	112488	THICK_LINE	NONE	7	white	black
338	4	49618	110163	50168	110663	THICK_LINE	NONE	3	white	black
338	5	46955	110025	48468	110788	THICK_LINE	NONE	95	white	black
338	6	46530	114438	46930	114813	THICK_LINE	NONE	166	white	black

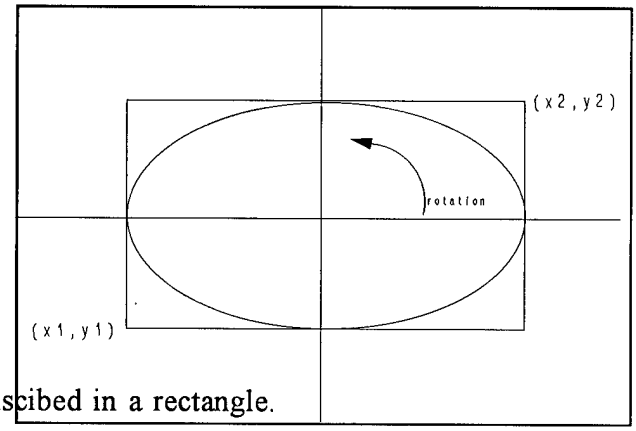


Figure A-2. Ellipse inscribed in a rectangle.

Table A-19

Control Measure Line Table (LINE)

Name	Description	Format
cm_index	control measure index	Integer
object	sequence of line in graphic	Integer
x1	UTM x coordinate (meters) of line start point	Integer
y1	UTM y coordinate (meters) of line start point	Integer
x2	UTM x coordinate (meters) of line end point	Integer
y2	UTM y coordinate (meters) of line end point	Integer
line_type	outline used in graphic	20 Char
color	color of line	20 Char

cm_ind	object	x1	y1	x2	y2	line_type	color
85	1	59690	119110	59690	119110	BASIC_LINE	white
85	2	59225	119110	59225	119065	BASIC_LINE	white
85	3	51950	113395	52040	113065	UNSPEC_WIRE_C	white
85	4	51710	112750	51770	112420	UNSPEC_WIRE_C	white
85	5	51815	112135	51920	111835	UNSPEC_WIRE_C	white
85	6	52145	111610	52280	111385	UNSPEC_WIRE_C	white
85	7	60125	117325	60365	117130	BASIC_LINE	white
85	8	60230	117250	60395	117475	BASIC_LINE	white
256	1	58760	115555	58760	115570	BASIC_LINE	blue
256	2	61680	114750	61355	115013	BASIC_LINE	white
256	3	61343	115025	61730	115663	BASIC_LINE	white
354	1	57193	113975	58193	113738	BASIC_ARROW_C	white
479	1	27700	104700	32800	104800	THICK_LINE	white
479	2	31500	104700	32850	104750	THICK_LINE	white
479	3	27700	108400	27500	104900	THICK_LINE	black
479	4	32600	108350	33050	104600	THICK_LINE	green
479	5	32750	108400	33000	104500	THICK_LINE	black
479	6	32700	108350	33900	108850	THICK_LINE	white
479	7	32750	108250	33800	108700	THICK_LINE	white
479	8	26650	108950	27700	108450	THICK_LINE	white
479	9	26700	108850	27300	108400	THICK_LINE	white
479	10	29200	109950	30000	108850	THICK_LINE	green

Table A-20

Control Measure Point Table (POINT)

Name	Description	Format
cm_index	control measure index	Integer
object	sequence of point	Integer
x	UTM x coordinate (meters)	Integer
y	UTM y coordinate (meters)	Integer
point_type	point description	20 Char
color	color of graphic	20 Char

cm_ind	object	x	y	point_type	color
70	1	647195	110620	SQUAD	white
70	2	647255	110710	SQUAD	white
85	1	655865	116230	SQUAD	white
85	2	657155	113950	SQUAD	white
256	1	661243	119363	MECH_INFANTRY	yellow
256	2	660968	119600	MECH_INFANTRY	yellow
256	3	660543	119550	ARMOR	yellow
256	4	660555	119738	SQUAD	yellow
256	5	660993	119750	SQUAD	yellow
256	6	661280	119500	SQUAD	yellow
256	7	659080	116775	MECH_INFANTRY	yellow
256	8	659068	116925	PLATOON	yellow
479	24	641500	124050	START_RELEASE_PT	black
848	1	654620	119590	START_RELEASE_PT	red
848	2	654395	118435	START_RELEASE_PT	red
848	3	652835	117280	START_RELEASE_PT	red
848	4	652865	118750	START_RELEASE_PT	red
848	5	650975	118210	START_RELEASE_PT	red
848	6	650555	117070	START_RELEASE_PT	red
848	7	655850	118780	START_RELEASE_PT	red
848	8	657905	117800	START_RELEASE_PT	red
848	9	656493	115900	START_RELEASE_PT	red

Table A-21

Control Measure Polyline Table (POLYLINE)

Name	Description	Format
cm_index	control measure index	Integer
object	sequence of polyline in graphic	Integer
seq	sequence of point in polyline	Integer
x	UTM x coordinate (meters)	Integer
y	UTM y coordinate (meters)	Integer
line_type	outline used in graphic	20 Char
color	color of line	20 Char

cm_ind	object	seq	x	y	line_type	color
66	1	1	25220	117235	BASIC_LINE	blue
66	1	2	24200	118345	BASIC_LINE	blue
66	1	3	23705	118540	BASIC_LINE	blue
66	1	4	23120	118450	BASIC_LINE	blue
66	1	5	22430	118060	BASIC_LINE	blue
66	1	6	20885	116725	BASIC_LINE	blue
66	1	7	20480	116035	BASIC_LINE	blue
66	1	8	22715	114970	BASIC_LINE	blue
66	1	9	25205	117235	BASIC_LINE	blue
66	2	1	28280	119695	BASIC_LINE	blue
66	2	2	30680	115795	BASIC_LINE	blue
66	2	3	35540	118705	BASIC_LINE	blue
66	2	4	34730	119905	BASIC_LINE	blue
66	2	5	28325	119665	BASIC_LINE	blue
66	3	1	33140	118990	BASIC_LINE	blue
66	3	2	33350	118750	BASIC_LINE	blue
66	3	3	34475	119515	BASIC_LINE	blue
66	3	4	34490	119860	BASIC_LINE	blue
66	4	1	31115	110485	BASIC_LINE	blue
66	4	2	31985	113140	BASIC_LINE	blue
66	4	3	29720	112915	BASIC_LINE	blue
66	4	4	26675	110530	BASIC_LINE	blue
66	5	1	31700	111445	DASHED_LINE	blue
66	5	2	32060	111730	DASHED_LINE	blue
66	5	3	32180	112585	DASHED_LINE	blue
66	5	4	31940	112780	DASHED_LINE	blue

Table A-22

Control Measure Polygon Table (POLYGON)

Name	Description	Format
cm_index	control measure index	Integer
object	sequence of polygon in graphic	Integer
sequence	sequence of point in polygon	Integer
x	UTM x coordinate (meters)	Integer
y	UTM y coordinate (meters)	Integer
line_type	outline of graphic	20 Char
fill_type	shading of graphic	20 Char
color	color of polygon	20 Char

cm_ind	object	seq	x	y	line_type	fill_type	color	fill_color
87	1	1	56945	114190	BASIC_LINE	ANT_TNK_MINE	yellow	black
87	1	2	56885	114520	BASIC_LINE	ANT_TNK_MINE	yellow	black
87	1	3	58025	113755	BASIC_LINE	ANT_TNK_MINE	yellow	black
87	1	4	57770	113380	BASIC_LINE	ANT_TNK_MINE	yellow	black
87	1	5	57875	113215	BASIC_LINE	ANT_TNK_MINE	yellow	black
87	1	6	58190	113620	BASIC_LINE	ANT_TNK_MINE	yellow	black
135	1	1	58955	115088	CONCERT_WIRE_C	ANT_PER_UNS_MINE	yellow	yellow
135	1	2	59743	115038	CONCERT_WIRE_C	ANT_PER_UNS_MINE	yellow	yellow
135	1	3	59768	115150	CONCERT_WIRE_C	ANT_PER_UNS_MINE	yellow	yellow
135	1	4	58968	115175	CONCERT_WIRE_C	ANT_PER_UNS_MINE	yellow	yellow
135	1	5	58905	115175	CONCERT_WIRE_C	ANT_PER_UNS_MINE	yellow	yellow
135	1	6	58955	115088	CONCERT_WIRE_C	ANT_PER_UNS_MINE	yellow	yellow
348	1	1	59530	117313	THICK_LINE	SHADED	red	white
348	1	2	59055	115050	THICK_LINE	SHADED	red	white
348	1	3	60080	114575	THICK_LINE	SHADED	red	white
348	1	4	61318	114513	THICK_LINE	SHADED	red	white
348	1	5	61430	117213	THICK_LINE	SHADED	red	white
348	1	6	61430	117213	THICK_LINE	SHADED	red	white
348	1	7	59530	117313	THICK_LINE	SHADED	red	white

Table A-23

Control Measure Rectangle Table (RECTANGLE)

Name	Description	Format
cm_index	control measure index	Integer
object	sequence of rectangle in graphic	Integer
x1	UTM x coordinate (meters) of rectangle vertex 1	Integer
y1	UTM y coordinate (meters) of rectangle vertex 1	Integer
x2	UTM x coordinate (meters) of rectangle vertex 2	Integer
y2	UTM y coordinate (meters) of rectangle vertex 2	Integer
line_type	outline of graphic	20 Char
fill_type	shading pattern of graphic	20 Char
color	outline color	20 Char
fill_color	shading color	20 Char

Table A-24

Control Measure Rectangle Table (RECTANGLE)

cm_ind	object	x1	y1	x2	y2	line_type	fill_type	color	fill_color
344	1	60118	115175	60218	115275	THICK_LINE	SHADED	black	black
344	2	59843	114838	59930	114925	THICK_LINE	SHADED	black	black
344	3	60080	114975	60180	115075	THICK_LINE	SHADED	black	black
479	1	3900	161600	101800	159500	THICK_LINE	SHADED	white	white
479	2	29350	109850	31350	110950	THICK_LINE	SHADED	black	white
479	3	27850	107450	29750	107950	THICK_LINE	SHADED	green	white
479	4	30250	107400	32350	107900	THICK_LINE	SHADED	green	white

Table A-25

Control Measure Text Table (TEXT)

cm_ind	object	x	y	font_s	num_ch	font_type	fill_type	text_data	color	fill_color
70	1	46970	112150	18	2	times bold	NONE	MB	white	black
70	2	37970	121645	18	5	times bold	NONE	HH.50	white	black
70	3	40640	121360	18	5	times bold	NONE	HH.40	white	black
70	4	41840	121420	18	5	times bold	NONE	HH.30	white	black
70	5	44825	121360	18	5	times bold	NONE	HH.10	white	black
70	6	48245	121015	18	5	times bold	NONE	HH.50	white	black
70	7	53555	121180	18	5	times bold	NONE	HH.30	white	black
70	8	56360	120910	18	4	times bold	NONE	HH.0	white	black
70	9	59975	121030	18	1	times bold	NONE	H	white	black
70	10	40895	114475	18	3	times bold	NONE	BNS	white	black
70	11	43145	112360	18	17	times bold	NONE	ADVAN. GUARD	white	black
70	12	47240	110440	24	3	times bold	NONE	CRP	white	black
70	13	47465	109990	24	3	times bold	NONE	FSE	white	black
256	1	56518	116500	18	2	times bold	NONE	OP	white	yellow
256	2	60368	119163	12	3	times bold	NONE	T72	yellow	yellow
256	3	61143	119738	9	3	times bold	NONE	BMP	yellow	yellow
256	4	61405	119300	9	3	times bold	NONE	BMP	yellow	yellow
256	5	58968	116450	12	5	times bold	NONE	3XBMP	yellow	yellow
256	6	57468	118038	12	6	times bold	NONE	A SECT	yellow	yellow
256	7	57493	117775	12	6	times bold	NONE	SET OP	yellow	yellow
256	8	57180	117363	12	7	times bold	NONE	PL SECT	yellow	yellow
256	9	57180	117125	12	6	times bold	NONE	SET OP	yellow	yellow
336	1	49820	117820	24	6	times bold	SHADED	SCOUTS	blue	white
336	2	49085	116110	24	1	times bold	SHADED	B	blue	white
336	3	46295	114790	24	1	times bold	SHADED	B	blue	white
336	4	47195	110140	24	1	times bold	SHADED	B	blue	white
336	5	40415	108940	24	1	times bold	SHADED	B	blue	white
336	6	43205	111385	24	1	times bold	SHADED	C	blue	white
336	7	38660	113800	24	7	times bold	SHADED	MORTARS	blue	white
336	8	38690	113320	24	12	times bold	SHADED	+ CBT TRAINS	blue	white
336	9	45470	112270	24	1	times bold	SHADED	D	blue	yellow
336	10	44150	111070	24	3	times bold	SHADED	CDR	black	yellow
336	11	48815	110305	24	2	times bold	SHADED	S3	black	yellow
336	12	43280	114070	24	1	times bold	SHADED	A	blue	white

Table A-26

Control Measure Text Table (TEXT)

Name	Description	Format
cm_index	control measure index	Integer
object	sequence of text string in graphic	Integer
x	UTM x coordinate (meters)	Integer
y	UTM y coordinate (meters)	Integer
font_size	# points in font	Integer
num_chars	# characters in string	Integer
font_type	type of font	20 char
fill_type	outline of graphic	20 Char
text_data	the string	25 Char
color	outline color of text	20 Char
fill_color	fill color of text	20 Char

Table A-27

Weapon Table (WEAPON)

Name	Description	Format
weapon_desc	description of weapon	20 Char
weapon_type	weapon type code	Integer

weapon_desc	weapon
EVP NULL WEAPON	0
EVP 105MM BALLISTIC	1
EVP 120MM BALLISTIC	2
WEAPON UNDEFINED	3
CONTROLLER GUN	4
60MM	10
81MM	11
107MM	12
105MM	20
155MM	21
175MM	22
8INCH	23
227MM ROCKET	24
105MM TANK MAIN GUN	30
120MM TANK MAIN GUN	31
152MM TANK MAIN GUN	32
25MM IFV MAIN GUN	40
50CAL MACHINE GUN	50
M60 MACHINE GUN	51
COAX	52
M249 MACHINE GUN	53

M16 RIFLE	54
40MM GRENADE	55
MARK 19 GRENADE	56
TOW	60
DRAGON	61
VIPER LAW	62
SHILLEGH MISSILE	63
20MM VULCAN	70
CHAPARRAL	71
STINGER	72
30MM AIRBORNE	80
20MM AIRBORNE	81
HELLFIRE	82
2 75INCH ROCKET	83
MAVERICK	84
ROCKEYE	85
120MM	110
160MM	111
240MM	112
122MM HOWITZER	120
152MM HOWITZER	121
152MM GUN HOWITZER	122
203MM GUN HOWITZER	123
122MM ROCKET	124
220MM ROCKET	125
FROG	126
125MM TANK MAIN GUN	140
73MM BMP MAIN GUN	150
30MM BMP MAIN GUN	151
12 7MM MACHINE GUN	160
PKT	161
AKM	162
RPG	163
AT3	170
AT4	171
AT5	172
AT6	173
23MM	180
SA9	181
SA13	182
SA14	183
12 7MM AIRBORNE MG	190
30MM GUN	191
57MM ROCKET	192

Tables A-28

Symbol Table (SYMBOL)

Name	Description	Format
symbol_desc	description of graphic symbol	20 Char
symbol_type	symbol type code	Integer

symbol_desc	symbol
UNDEFINED	0
UNIT MECH INFANTRY	2
UNIT ARMOR	3
UNIT INFANTRY	4
UNIT AIR DEFENSE	5
UNIT ARTILLERY	6
UNIT SP ARTILLERY	7
UNIT ANTI TANK	8
UNIT ELECTRONIC WARFARE	9
UNIT ARMY AVIATION FIXED	10
UNIT ARMY AVIATION ROTARY	11
UNIT AIRBORNE INFANTRY	12
UNIT AIR CAVALRY	13
UNIT CHEMICAL DEFENSE	14
UNIT ARMORED CAVALRY	15
UNIT ENGINEER	16
UNIT MEDICAL	17
UNIT SIGNAL	18
UNIT TRANSPORTATION	19
UNIT SUPPLY AND SERVICE	20
UNIT MAINTENANCE	21
UNIT COMBAT TRAINS MECH	22
UNIT COMBAT TRAINS ARMOR	23
UNIT FIELD TRAINS MECH	24
UNIT FIELD TRAINS ARMOR	25
UNIT LIGHT MORTAR	26
UNIT HEAVY MORTAR	27
A STATION	46
FIELD VIDEO	47
FIELD CONTROLLER	48
FIRE MARKER	49
TANK	60
BRADLEY IFV	70
CFV	71
APC	72
AT APC	73
SP VULCAN	80
MANPACK STINGER	81
SP CHAPARRAL	82
MANPACK M16 M203	90
MANPACK M60	91
MANPACK M249	92

MANPACK TOW	93
MANPACK DRAGON	94
MANPACK VIPER LAW	95
MORTAR	100
SP MORTAR	105
M108 SP HOWITZER	110
M109 SP HOWITZER	111
M107 SP GUN HOWITZER	112
M110 SP HOWITZER	113
MRLS	114
ATTACK HELO	120
OBSERVATION HELO	121
UTILITY HELO	122
FIGHTER ATTACK AC	130
BOMBER	131
FIGHTER BOMBER	132
RECON	133
RADAR	140
JAMMER	141
COLLECTOR	142
TRUCK	143
TANK	160
BMP	170
BRDM	171
BRDM AT	172
ZSU	180
BRDM AD	181
MANPACK SA14	182
MANPACK AKM	190
MANPACK PKT	191
MANPACK AT MISSILE	192
MANPACK RPG	193
M1943 MORTAR	200
M160 MORTAR	201
M240 MORTAR	202
SP HOWITZER	210
SP GUN HOWITZER	211
MRL	212
FROG	213
ATTACK HELO	220
NONCOMBAT HELO	225
FIGHTER	230
BOMBER	231
FIGHTER BOMBER	232
RECON	233
JAMMER	240
COLLECTOR	241
TRUCK	242

Table A-29

Platform Table (PLATFORM)

Name	Description	Format
platform_desc	description of vehicle platform	20 Char
platform_type	platform type code	Integer

platform_desc	platfo
EVP_NULL_PLATFORM	0
UNDEFINED_PLATFORM	1
FIELD_CONTROLLER	10
FIELD_VIDEO	11
FIELD_MARKER	12
M60_TANK	30
M1_TANK	31
M1A1_TANK	32
M551_TANK	33
M2_IFV	40
M3_CFV	41
M113_APC	42
M113_WITH_TOW	43
M901_AT_APC	44
SP_VULCAN	50
MANPACK_STINGER	51
SP_CHAPARRAL	52
MANPACK	60
MANPACK_M16	61
MANPACK_M60	62
MANPACK_M249	63
MANPACK_M203	64
MANPACK_TOW	65
MANPACK_DRAGON	66
MANPACK_VIPER_LAW	67
MANPACK_MARK_19	68
FIST_V	80
M224_MORTAR	85
M125_SP_MORTAR	90
M106_SP_MORTAR	91
M108_SP_HOWITZER	100
M109_SP_HOWITZER	101
M107_SP_GUN_HOWITZER	102
M110_SP_HOWITZER	103
MRLS	104
AH_64	110
AH_1S	111
OH_58	112
UH_1	113
UH_60	114
A10	120
FIGHTER	121

BOMBER	122
FIGHTER_BOMBER	123
RECON	124
RADAR	130
JAMMER	131
COLLECTOR	132
TRUCK	133
TRUCK_HMMWV_MARK_19	135
T72_TANK	160
BMP1	170
BMP2	171
BMP	172
BRDM	173
BRDM2_AT5	174
BRDM_AT3	175
ZSU	180
BRDM2_AD	181
MTLB	182
MANPACK_SA14	183
MANPACK	190
MANPACK_AKM	191
MANPACK_PKT	192
MANPACK_AT3	193
MANPACK_AT4	194
MANPACK_RPG	195
M1943_MORTAR	200
M160_MORTAR	201
M240_MORTAR	202
122_SP_HOWITZER	210
152_SP_HOWITZER	211
152_SP_GUN_HOWITZER	212
203_SP_GUN_HOWITZER	213
BM21_MRL	214
BM27_MRL	215
FROG	216
HIND_D	220
HIND_E	221
UH_HIP	222
OH_HOPLITE	223
FIGHTER	230
BOMBER	231
FIGHTER_BOMBER	232
RECON	233
JAMMER	240
COLLECTOR	241
TRUCK	242

Table A-30

FRAT (Fratricide table)

Name	Description	Format
time	date-time of fratricide event	20 Char
tlpn	target logical player number	Integer
tpid	target player bumper number	8 Char
tside	target side	1 Char
tx	target UTM x coordinate (meters)	Integer
ty	target UTM y coordinate (meters)	Integer
tz	target z coordinate (feet)	Integer
tplatform	target vehicle platform	Integer
pair_type	type of pairing	Integer
weapon	weapon type of firer	Integer
fplatform	firer vehicle platform	Integer
flpn	firer logical player number	Integer
fpid	firer player bumper number	8 Char
fside	firer side	1 Char
fx	firer UTM x coordinate (meters)	Integer
fy	firer UTM y coordinate (meters)	Integer
fz	firer z coordinate (feet)	Integer
result	pairing result	10 Char
distance	distance in meters of engagement	Integer

ANNEX B

SUBDIRECTORY PATH LISTING FOR THE ARI-NTC MISSION DATABASE

Rotations 92_01 through 93_12 have been re-created to conform with the current NTC mission database structure. The current NTC mission databases have a consolidated set of control measures (one set of tables) and indirect fire target tables. This more closely matches the manner in which these data are collected and used by the analysts at the NTC. Also, an additional table has been added, the Fratricide Table, (FRAT.dbf) because all fratricides (matched pair events in the FET) were omitted from paired event messages by the NTC software and placed in their own message. The following subdirectory path listing for the ARI-NTC Mission Database is provided as an index for the compact discs that were delivered to CALL in June of 1995:

CD 1: CTC ARCHIVE DATA
FY 92 MISSION DATABASE

N921: N921_V12,N921_V14,N921C_06,N921C_08,N921C_09,N921C_10,N921CV16,N921CV19

N922: N922A_03,N922AN12,N922ZA05,N922ZA14,N922ZA15

N923: N923_M07,N923_M09,N923AL14,N923AM01,N923AM04,N923AZ03,N923AZ06

N924: N924_M12,N924_M14,N924_M16,N924A_17,N924A_23,N924B118

N925: N925_M09,N925_M11,N925_M12,N925A_16,N925B114,N925B120,N925B121

N926: N926_M17,N926_M19,N926_M21,N926A_11,N926AL13,N926AL15,N926AZ08

N927: N927_M11,N927_M13,N927_M15,N927A_10,N927AL07,N927AL09

N928: N928_M03,N928_M07,N928A_10,N928A_12,N928AM05,N928AM14,N928AM16

N929: N929_M04,N929_M31,N929AL06,N929AL08,N929AZ11,N929AZ13

N92A: N92A_M24,N92AA_12,N92AA_16,N92AA_17,N92AAM25

N92B: N92B_M15,N92B_M19,N92BA_09,N92BA_11,N92BA_13,N92BAM21,N92BAM22

N92C: N92CAL06,N92CAL08,N92CAL10,N92CAL18,N92CAL19

CD 2: CTC ARCHIVE DATA
FY 93 MISSION DATABASE

N931: N931CV16,N931C_09,N931C_11,N931C_13,N931_V06,N931_V08,N931_V04

N932: N932_M09,N932A_03,N932_M07

N933: N933A_05,N933A_07,N933LM01,N933LM29,N933ZA11,N933AM12

N934: N934AL10,N934AL12,N934AL14,N934_M16

N935: N935AL18,N935AL20,N935RL06,N935ZA08,N935ZA09,N935_M15

N936: N936AM19,N936A_13,N936A_15,N936A_17,N936_M07,N936_M09,N936_M11

N937: N937AL04,N937AL06,N937AL08,N937AM17,N937ZA16,N937_M10,N937_M14

N938: N938A_08,N938A_10,N938A_12,N938ML02,N938ML04,N938ML06,N938ZA14

N939: N939AM11,N939MX01,N939AX09,N939AX07,N939AX05,N939AM12

N93A: N93A_C15,N93A_C13,N93A_C11,N93AA_21,N93AA_19,N93AA_17

N93B: N93B_M18,N93B_M16,N93BZA21,N93BA_20,N93BAL17,N93BAL14,N93BAL12,N93BAL10

N93C: N93CVX15,N93CVX13,N93CVX11,N93CCX09,N93CCX07,N93CCV18,N93CCV17

CD 3: CTC ARCHIVE DATA
FY 94 MISSION DATABASE (PARTIAL)

N941: N941A111,N941A113,N941B115,N941B116,N941M104

N942: N942A102,N942A103,N942B112,N942B113,N942C104,N942C109,N942C110,N942M107,N942M108

N943: N943A102,N943B104,N943B107,N943B109,N943B111,N943M205

N944: N944A109,N944A111,N944A113,N944B121,N944B122,N944M115,N944M117,N944M119

N945: N945A106,N945A109,N945B118,N945M112,N945M114

N946: N946B106,N946B107,N946B109,N946B118,N946B119,N946C113,N946C115,N946C117

N947: N947A114,N947B111,N947B113,N947B116,N947B118,N947B120,N947B122,N947B123

N948: N948A108,N948A110,N948A112,N948B120,N948M114,N948M116,N948M118

N949: N949B105,N949B107,N949B116,N949B118

The following NTC rotations have been constructed from October 1994 through March 31, 1995: (note that no rotation was executed for 94-10)

CD 4: CTC ARCHIVE DATA
FY 94/95 MISSION DATABASE (PARTIAL)

N94B: N94BA114,N94BA115,N94BB117,N94BB120,N94BM121,N94BB127,N94BM125

N94C: N94CB111,N94CC114,N94CB118,N94CC120,N94CB121,N94CC122,N94CB124

N951: N951C109,N951C111,N951C113,N951C215,N951C216,N951C217,N951B121,N951B122

N952: N952B106,N952B108,N952B110,N952B118

N953: N953A104,N953A106,N953A108,N953AC10,N953AC14,N953B116,N953B117

N954: N954B115, N954B117,N954B119,N954B127,N954B128

These data are distributed on CD-ROM for your benefit. The data has been segregated by year (except for the last CD-ROM, which contains NTC rotations 94-10 through 95-04). On these CD-ROMs the Archive Finders Guide data, the Battle Damage Assessment data, the Graphics

data and the Take Home Package data for the appropriate year has also been placed on the CD-ROM. The software for using these data is also included on the CD-ROM. This was done so as copies of the CD-ROMs could be made and then distributed by CALL to users who wish to use the archive data and do not wish to access these data remotely.

The first group of NTC mission databases (rotations 92-01 through 93-12) should be used to replace those mission databases you currently have loaded on the CALL network. This can be easily done by deleting those directories N921 through N93C with the 'File Manager' tool in Windows 3.1. Use this same tool to copy the contents of the CD-ROM to your network, replacing the previously deleted files. Do not copy other data from the CD-ROM to the network, as it has been cut by year and will only contain a subset of the data. For the new NTC mission databases, just use file manager to copy the contents of the last CD-ROM, 94-10 through 95-04, to your network. A new 'aridms.dbf' table has been provided on floppy disk to replace the one contained in:

'h:\archive\mission\aridms.dbf'

Do not copy this table from one of the CD-ROMs because, there it only contains the mission information for the NTC mission databases on the CD-ROM it is located on.

ANNEX C

SOURCE CODE AND DOCUMENTATION FOR CREATING THE CONTROL MEASURE TABLES

BUILDCM

The program, 'buildcm', reads the control_measure_area, gets a list of those directories where control measures plans are located, and loads those control measures (by plan) into the nine control measure tables. These tables are:

CM_MASTER.DBF	Control Measure Master Table
ARC.DBF	Arc Object Table
CIRCLE.DBF	Circle Object Table
ELLIPSE.DBF	Ellipse Object Table
POINT.DBF	Point Object Table
LINE.DBF	Line Object Table
POLYLINE.DBF	Polyline Object Table
POLYGON.DBF	Polygon Object Table
RECTANGL.DBF	Rectangle Object Table
TEXT.DBF	Text Object Table
IFGT.DBF	Indirect Fire Group Table
IFTT.DBF	Indirect Fire Target Table

These tables are to be placed in the Rotation directory, i.e. for rotation 95_04, these files should be placed in:

h:\archive\mission\95_04

To compile this program, use the makefile gcmp.mk, as 'make -f gcmp.mk'.

The contents of the makefile list those software modules which compose the program. The file looks like:

```
cm_master:
     cc -o buildcm get_cm_dir.c cm_index.c planned_targets.c group_targets.c
wcx2utm.c wcy2utm.c ntc_time.c -lm
```

GET_CM_DIR.C

```
/*********************************************************/
/*  get_cm_dir.c                                         */
/*********************************************************/

#include    <stdio.h>
#include    <str.h>
main    argc,argv;
```

```
char    *argc;
int     *argv[];
char  *rotation;
int   start_seconds;
int   end_seconds;
{
int   i,    j;
int   stat;
int   buf1[200];
int   buf2[200];
char  *cm_path_1 = {"/usr2/cs_archive_files/\0"};
char  *cm_path_2 = {"/control_measures_area/\0"};
char    *path_array;
char    *cmd1,  *cmd2,  *cmd,   *cmd3,      *cmd4;
char    *subdir1;
char    *subdir2;
FILE  *oc_stream,  *oc_stream2;

      path_array = (char *) malloc(100);

      strcpy(path_array,cm_path_1);
      strcat(path_array,rotation);
      strcat(path_array,cm_path_2);

      cmd1  =     (char *) malloc(100);

      strcpy(cmd1,"ls -F1 \0");
      strcat(cmd1,path_array);
      cmd2  =     (char *) malloc(100);
      strcpy(cmd2,cmd1);
      strcat(cmd1, " >oc.lis \0");

      stat  =     system(cmd1);

      oc_stream   =     fopen("oc.lis","r");
      if (oc_stream == (FILE *) NULL)
      {
            printf ("\n no OC file found...");
            exit();
      }
      subdir1  =   (char *) calloc(25,sizeof(char));

      for (i=0; fscanf(oc_stream,"%s\n",subdir1) != EOF; i++)
      {
            cmd      =   (char *) calloc(75,sizeof(char));
            strcpy(cmd,cmd2);
            strcat(cmd,subdir1);
            strcat(cmd," >oc2.lis");
            stat    =   system(cmd);
            oc_stream2  =     fopen("oc2.lis","r");
            subdir2  =   (char *) calloc(50,sizeof(char));
            for (j=0; fscanf(oc_stream2,"%s\n",subdir2) != EOF; j++)
            {
                  cmd3    =   (char *) calloc(100,sizeof(char));
                  cmd4  =   (char *) calloc(100,sizeof(char));
                        strcpy(cmd3,path_array);
                        strcat(cmd3,subdir1);
                        strcat(cmd3,subdir2);
                  strcpy(cmd4,cmd3);
                        strcat(cmd3,"cm.index");
                  strcat(cmd4,"cm.db");
                  cm_index(cmd3,cmd4,start_seconds,end_seconds);
                        free   (subdir2);
                        subdir2 =   (char *) calloc(50,sizeof(char));
                        free   (cmd3);
```

```
                free   (cmd4);
            }
            free   (subdir1);
            subdir1 =   (char *) calloc(25,sizeof(char));
            free   (cmd);
        }
        fclose(oc_stream);
        fclose(oc_stream2);
}


/************************************************/
/*  cm_index.c                                  */
/************************************************/

#include        <stdio.h>
#include        <stdlib.h>
#include        "cm_index.h"
#include        "cm_header.h"
#include        "ntc_time.h"
#define         PMODE 0644

void    cm_index(ci_path,cdb_path,start_seconds,end_seconds)
char   *ci_path;
char   *cdb_path;
int     start_seconds;
int     end_seconds;
{
    int     ci_buf[15000];
    int    *pci_buf =   ci_buf;
    char    cdb_buf[65528];
    char   *pcdb_buf=   cdb_buf;
    int     a_buf[4096];
    int    *pa_buf  =   a_buf;
    int     c_buf[4096];
    int    *pc_buf  =   c_buf;
    int     e_buf[4096];
    int    *pe_buf  =   e_buf;
    int     l_buf[65528];
    int    *pl_buf  =   l_buf;
    int     p_buf[4096];
    int    *pp_buf  =   p_buf;
    int         o_buf[4096];
    int    *po_buf  =   o_buf;
    int         g_buf[4096];
    int        *pg_buf  =      g_buf;
    int         r_buf[4096];
    int    *pr_buf  =    r_buf;
    int     t_buf[4096];
    int        *pt_buf  =      t_buf;
    int     bytes, dbbytes,    i,  j,   k,  l,   m,   n,   p,   q;
    int         last_pos,      point_idx;
    int         point_array[1024];
    long    rec_offset;

    struct  cm_index   *pcmi;
    struct  cm_header  *pcdb;
    struct  cm_arc     *pa;
    struct  cm_circle  *pc;
    struct  cm_ellipse *pe;
    struct  cm_line    *pl;
    struct  cm_point   *pp;
    struct  cm_polyline *po;
    struct  cm_polygon *pg;
    struct  cm_rect    *pr;
```

```
struct  cm_text     *pt;
struct  date        *ntc_time();
struct  date        *cmi_start[4000];
struct  date        *cmi_end[4000];

int      maxtext;

union  u_tag
{
    int ival;
    short   sval[2];
    char    cval[4];
} uval;

FILE    *ci_stream;
FILE    *cm_db_file;
FILE    *cm_index_file;
static FILE    *master;     /* control measure master file      */
static FILE    *arc;        /* control measure arc objects      */
static FILE    *circle;     /* control measure circle objects   */
static FILE    *ellipse;    /* control measure ellipse objects  */
static FILE    *line;       /* control measure line objects     */
static FILE    *point;      /* control measure point object     */
static FILE    *polyline;   /* control measure polyline object  */
static FILE    *polygon;    /* control measure polygon object   */
static FILE    *rectangle;  /* control measure rectangle object */
static FILE    *text; /* control measure text object     */
static int         knt;
static int             fd;
static int     first_time = 0;
static int  new_index;



cm_index_file    =    fopen(ci_path,"r");
if  (cm_index_file == (FILE *) NULL)
{
    printf("\n Open failure on %s",ci_path);
  goto finished;
}

fd = open(cdb_path,PMODE);
if (fd < 0)
{
  printf("\n open failure on %s",cdb_path);
  goto finished;
}

if (first_time == 0)
{
  master  =    fopen("master.dat","w");
  if  (master ==  (FILE *) NULL )
  {
        printf("\n open failure on control measure master file");
        exit();
  }
  arc      =    fopen("arc.dat","w");
  if  (arc == (FILE *) NULL)
  {
        printf("\n open failure on control measure arc file");
              exit();
  }
  circle  =    fopen("circle.dat","w");
  if  (circle ==  (FILE *) NULL)
  {
```

```
            printf("\n open failure on control measure circle file");
            exit();
        }
        ellipse  =   fopen("ellipse.dat","w");
        if  (ellipse ==  (FILE *) NULL )
        {
                    printf("\n open failure on control measure ellipse file");
            exit();
        }
        line     =   fopen("line.dat","w");
        if  (line == (FILE *) NULL)
        {
            printf("\n open failure on control measure line file");
            exit();
        }
        point  =   fopen("point.dat","w");
        if  (point ==  (FILE *) NULL)
        {
            printf("\n open failure on control measure point file");
            exit();
        }
        polyline  =   fopen("polyline.dat","w");
        if  (polyline ==  (FILE *) NULL)
        {
            printf("\n open failure on control measure polyline file");
            exit();
        }
        polygon    =   fopen("polygon.dat","w");
        if  (polygon == (FILE *) NULL)
        {
            printf("\n open failure on control measure polygon file");
                  exit();
        }
        rectangle  =   fopen("rectangle.dat","w");
        if  (rectangle ==  (FILE *) NULL)
        {
            printf("\n open failure on control measure rectangle file");
            exit();
        }
        text       =   fopen("text.dat","w");
        if  (text ==  (FILE *) NULL)
        {
            printf("\n open failure on control measure text file");
            exit();
        }
        first_time = 1;
        new_index = 0;
        knt = 0;
    }

/*    printf("\n Processing control measure data...\n");       */

    bytes  =   fread(ci_buf, sizeof *ci_buf, 15000, cm_index_file);
 /*   fclose(cm_index_file);    */

    pcmi    =   (struct cm_index *) pci_buf;
    pcdb    =   (struct cm_header *) pcdb_buf;
/*    knt = 0;      */
    for (i=0;   i<bytes/18;    i++)
    {
        if  ((start_seconds  <=  pcmi->end_seconds) &&
             (end_seconds >= pcmi->start_seconds) &&
          ((pcmi->end_seconds - pcmi->start_seconds) < 500000))
        {
/*        printf("\n %s",cdb_path);   */
```

```
fprintf(master,"%5d,",i+1+new_index);
knt++;
cmi_start[knt]  =   (struct date *) malloc(sizeof(struct date));
cmi_end[knt]    =   (struct date *) malloc(sizeof(struct date));
cmi_start[knt]  =   ntc_time(pcmi->start_seconds);
cmi_end[knt]    =   ntc_time(pcmi->end_seconds);
rec_offset = pcmi->file_index;
lseek(fd,rec_offset,0);
dbbytes =   read (fd,cdb_buf,65528);
fprintf(master,"%s,%s",
cmi_start[knt]->date_time,
cmi_end[knt]->date_time);

switch (pcmi->force)
{
    case 1:
    fprintf(master,",B");
    break;

       case 2:
    fprintf(master,",O");
       break;

    case 4:
    fprintf(master,",W");
    break;

    default:
    fprintf(master,",U");
    break;
}
switch (pcmi->echelon)
{
    case 1:
    fprintf(master,",Division ");
    break;

    case 2:
    fprintf(master,",Brigade  ");
    break;

       case 4:
       fprintf(master,",Battalion");
       break;

       case 8:
       fprintf(master,",Company  ");
    break;

       case 16:
       fprintf(master,",Platoon  ");
    break;

    case 32:
    fprintf(master,",Section  ");
    break;

    default:
    fprintf(master,",Mult. Ech");
    break;
}
switch (pcmi->category)
{
       case 1:
       fprintf(master,",Air");
```

```
        break;

        case 2:
        fprintf(master,",ADA");
        break;

    case 4:
    fprintf(master,",CSS");
    break;

    case 8:
    fprintf(master,",FS ");
    break;

    case 16:
    fprintf(master,",Int");
    break;

    case 32:
    fprintf(master,",Mnv");
    break;

    case 64:
        fprintf(master,",Mob");
        break;

    default:
    fprintf(master,",NU ");
    break;
}
switch (pcmi->status)
{
        case 1:
        fprintf(master,",Current ");
        break;

        case 2:
        fprintf(master,",Proposed");
        break;

        default:
    fprintf(master,",Undefine");
    break;
}

fprintf(master,",%3d,%3d,%3d,%3d,%3d,%3d,%3d,%3d,%3d\n",
                pcdb->arc,
                pcdb->circle,
                pcdb->ellipse,
                pcdb->line,
                pcdb->point,
                pcdb->polyline,
                pcdb->polygon,
                pcdb->rect,
                pcdb->text);


last_pos = 26;          /* look past cm header stuff... */
/*******************************/
/*  Do an arc....              */
/*******************************/
if (pcdb->arc > 0)
{

    for (m = 0; m < 16*(pcdb->arc); m++)
```

```
    {
        for (n=0; n<4; n++)
        {
            uval.cval[n] = cdb_buf[last_pos + m*4 + n];
        }
        a_buf[m] = uval.ival;
    }
    last_pos += 16*(pcdb->arc)*4;   /*  get past the arcs */
    pa = (struct cm_arc *) pa_buf;
    for (p = 0; p < pcdb->arc; p++)
    {
        fprintf(arc,"%5d,%5d",i+1+new_index,p+1);
        fprintf(arc,",%6d,%6d,%6d,%6d,",
                        wcx2utm(pa->x1),wcy2utm(pa->y1),
                        wcx2utm(pa->x2),wcy2utm(pa->y2));

        j=20;
        for (k = 0; k < pa->line_type_len; k++)
        {
            if (pa->line_type[j+k] > 31)
            {
                fprintf(arc,"%c",pa->line_type[j+k]);
            }
        }
        fprintf(arc,",%10s\n",*(color_table + pa->color));
        pa++;
    }
}
/*******************************/
/*  Do a circle.....           */
/*******************************/
if (pcdb->circle > 0)
{
    for (m = 0; m < 27*(pcdb->circle); m++)
    {
        for (n=0; n<4; n++)
        {
            uval.cval[n] = cdb_buf[last_pos + m*4 + n];
        }
        c_buf[m] = uval.ival;
    }
    last_pos += 27*(pcdb->circle)*4;  /* get past the circles */
    pc =  (struct cm_circle *) pc_buf;
    for (p = 0; p < pcdb->circle; p++)
    {
        fprintf(circle,"%5d,%5d",i+1+new_index,p+1);
        fprintf(circle,",%6d,%6d,%6d,%6d,",
                wcx2utm(pc->x1),wcy2utm(pc->y1),
                wcx2utm(pc->x2),wcy2utm(pc->y2));

        j=20;
        for (k = 0; k < pc->line_type_len; k++)
        {
            if (pc->line_type[j+k] > 31)
            {
                fprintf(circle,"%c",pc->line_type[j+k]);
            }
        }
        fprintf(circle,",");
        for (k = 0; k < pc->fill_type_len; k++)
        {
            if (pc->fill_type[j+k] > 31)
            {
                fprintf(circle,"%c",pc->fill_type[j+k]);
            }
```

```
        }
        fprintf(circle,",%10s,%10s\n",
                *(color_table + pc->color),
                *(color_table + pc->fill_color));
        pc++;
    }
}
/*******************************/
/*  Do an ellipse.....         */
/*******************************/
if (pcdb->ellipse > 0)
{
    for (m = 0; m < 28*(pcdb->ellipse); m++)
    {
        for (n=0; n<4; n++)
        {
            uval.cval[n] = cdb_buf[last_pos + m*4 + n];
        }
        e_buf[m] = uval.ival;
    }
    last_pos += 28*(pcdb->ellipse)*4;  /* get past the ellipses */
    pe =  (struct cm_ellipse *) pe_buf;
    for (p = 0; p < pcdb->ellipse; p++)
    {
        fprintf(ellipse,"%5d,%5d",i+1+new_index,p+1);
        fprintf(ellipse,",%6d,%6d,%6d,%6d,",
                wcx2utm(pe->x1),wcy2utm(pe->y1),
                wcx2utm(pe->x2),wcy2utm(pe->y2));

        j=20;
        for (k = 0; k < pe->line_type_len; k++)
        {
            if (pe->line_type[j+k] > 31)
            {
                fprintf(ellipse,"%c",pe->line_type[j+k]);
            }
        }
        fprintf(ellipse,",");
        for (k = 0; k < pe->fill_type_len; k++)
        {
            if (pe->fill_type[j+k] > 31)
            {
                fprintf(ellipse,"%c",pe->fill_type[j+k]);
            }
        }
        fprintf(ellipse,",%3d",pe->rotation);
        fprintf(ellipse,",%10s,%10s\n",
                *(color_table + pe->color),
                *(color_table + pe->fill_color));
        pe++;
    }
}
/*******************************/
/*  Do a line.....             */
/*******************************/
if (pcdb->line > 0)
{
    for (m = 0; m < 16*(pcdb->line); m++)
    {
        for (n=0; n<4; n++)
        {
            uval.cval[n] = cdb_buf[last_pos + m*4 + n];
        }
        l_buf[m] = uval.ival;
    }
```

```
        last_pos += 16*(pcdb->line)*4;  /* get past the lines */
        pl =  (struct cm_line *) pl_buf;
        for (p = 0; p < pcdb->line; p++)
        {
            fprintf(line,"%5d,%5d",i+1+new_index,p+1);
            fprintf(line,",%6d,%6d,%6d,%6d,",
                    wcx2utm(pl->x1),wcy2utm(pl->y1),
                    wcx2utm(pl->x2),wcy2utm(pl->y2));

            j=20;
            for (k = 0; k < pl->line_type_len; k++)
            {
                if (pl->line_type[j+k] > 31)
                {
                    fprintf(line,"%c",pl->line_type[j+k]);
                }
            }
            fprintf(line,",%10s\n",*(color_table + pl->color));
            pl++;
        }
    }
    /*********************************/
    /*  Do a point....               */
    /*********************************/
    if (pcdb->point > 0)
    {
        for (m = 0; m < 14*(pcdb->point); m++)
        {
            for (n=0; n<4; n++)
            {
                uval.cval[n] = cdb_buf[last_pos + m*4 + n];
            }
            p_buf[m] = uval.ival;
        }
        last_pos += 14*(pcdb->point)*4;   /*  get past the points */
        pp = (struct cm_point *) pp_buf;

        for (p = 0; p < pcdb->point; p++)
        {
            fprintf(point,"%5d,%5d",i+1+new_index,p+1);
            fprintf(point,",6%d,%6d,",
                            wcx2utm(pp->x),wcy2utm(pp->y));

            j=20;
            for (k = 0; k < pp->symbol_length; k++)
            {
                if (pp->symbol[j+k] > 31)
                {
                    fprintf(point,"%c",pp->symbol[j+k]);
                }
            }
            fprintf(point,",%10s\n",*(color_table + pp->color));
            pp++;
        }
    }
    /*********************************/
    /*  Do a polyline                */
    /*********************************/
  point_idx =    last_pos +
          ((pcdb->polyline)*4)*13 +
          ((pcdb->polygon)*4)*24 +
          ((pcdb->rect)*4)*27 +
          ((pcdb->text)*4)*16;
    if (pcdb->polyline > 0)
    {
```

```
    for (m = 0; m < 13*(pcdb->polyline); m++)
    {
        for (n=0; n<4; n++)
        {
            uval.cval[n] = cdb_buf[last_pos + m*4 + n];
        }
        o_buf[m] = uval.ival;
    }
    last_pos += 13*(pcdb->polyline)*4;  /* get past the polylines */
    po = (struct cm_polyline *) po_buf;
    for (p = 0; p < pcdb->polyline; p++)
    {

     for (m = 0; m < po->num_points*8; m++)
     {
       for (n = 0; n < 4; n++)
       {
           uval.cval[n] = cdb_buf[point_idx + m*4 + n];
       }
       point_array[m] = uval.ival;
     }
     point_idx += po->num_points*8;
     for (q = 0; q < po->num_points; q++)
     {
            fprintf(polyline,"%5d,%3d,%3d",i+1+new_index,p+1,q+1);
       fprintf(polyline,",%6d,%6d,",
             wcx2utm(point_array[q*2]),
             wcy2utm(point_array[q*2 + 1]));
             j=20;
             for (k = 0; k < po->line_type_len; k++)
            {
                if  (po->line_type[j+k] > 31)
                {
                    fprintf(polyline,"%c",po->line_type[j+k]);
                 }
             }
             fprintf(polyline,",%10s\n",*(color_table + po->color));
     }
         po++;
    }
}
/*******************************/
/*  Do a polygon               */
/*******************************/
if (pcdb->polygon > 0)
{
    for (m = 0; m < 24*(pcdb->polygon); m++)
    {
        for (n=0; n<4; n++)
        {
            uval.cval[n] = cdb_buf[last_pos + m*4 + n];
        }
        g_buf[m] = uval.ival;
    }
    last_pos += 24*(pcdb->polygon)*4;  /* get past the polygons */
    pg = (struct cm_polygon *) pg_buf;
    for (p = 0; p < pcdb->polygon; p++)
    {
     for (m = 0; m < pg->num_points*8; m++)
     {
       for (n = 0; n < 4; n++)
       {
           uval.cval[n] = cdb_buf[point_idx + m*4 + n];
       }
       point_array[m] = uval.ival;
```

```
        }
        point_idx += pg->num_points*8;
        for (q = 0; q < pg->num_points; q++)
        {
                fprintf(polygon,"%5d,%3d,%3d",i+1+new_index,p+1,q+1);
          fprintf(polygon,",%6d,%6d,",
                 wcx2utm(point_array[q*2]),
                 wcy2utm(point_array[q*2 + 1]));

                j=20;
                for (k = 0; k < pg->line_type_len; k++)
                {
                    if  (pg->line_type[j+k] > 31)
                    {
                        fprintf(polygon,"%c",pg->line_type[j+k]);
                    }
                }
             fprintf(polygon,",");
                for (k = 0; k < pg->fill_type_len; k++)
                {
                    if (pg->fill_type[j+k] > 31)
                    {
                        fprintf(polygon,"%c",pg->fill_type[j+k]);
                    }
                }

                fprintf(polygon,",%10s,%10s\n",
                        *(color_table + pg->color),
                 *(color_table + pg->fill_color));
        }
            pg++;
        }
    }
    /*******************************/
    /*  Do a rectangle.....        */
    /*******************************/
    if (pcdb->rect > 0)
    {
        for (m = 0; m < 27*(pcdb->rect); m++)
        {
            for (n=0; n<4; n++)
            {
                uval.cval[n] = cdb_buf[last_pos + m*4 + n];
            }
            r_buf[m] = uval.ival;
        }
        last_pos += 27*(pcdb->rect)*4;     /* get past the rects */
        pr =  (struct cm_rect *) pr_buf;

        for (p = 0; p < pcdb->rect; p++)
        {
            fprintf(rectangle,"%5d,%5d,",i+1+new_index,p + 1);
            fprintf(rectangle,"%6d,%6d,%6d,%6d,",
                    wcx2utm(pr->x1),wcy2utm(pr->y1),
                    wcx2utm(pr->x2),wcy2utm(pr->y2));
            j=20;
            for (k = 0; k < pr->line_type_len; k++)
            {
                if (pr->line_type[j+k] > 31)
                {
                    fprintf(rectangle,"%c",pr->line_type[j+k]);
                }
            }
         fprintf(rectangle,",");
            for (l = 0; l < pr->fill_type_len; l++)
```

```
                {
                    if (pr->fill_type[j+l] > 31)
                    {
                        fprintf(rectangle,"%c",pr->fill_type[j+l]);
                    }
                }
                fprintf(rectangle,",%10s,%10s\n",
                            *(color_table + pr->color),
                     *(color_table + pr->fill_color));
            pr++;
            }
        }
        /*******************************/
        /*  Do text...                 */
        /*******************************/
        if (pcdb->text> 0)
        {
            for (m = 0; m < 16*(pcdb->text); m++)
            {
                for (n=0; n<4; n++)
                {
                    uval.cval[n] = cdb_buf[last_pos + m*4 + n];
                }
                t_buf[m] = uval.ival;
            }
            last_pos += 16*(pcdb->text)*4;   /*  get past the points */
            pt = (struct cm_text*) pt_buf;

            for (p = 0; p < pcdb->text; p++)
            {
                fprintf(text,"%5d,%5d",i+1+new_index,p+1);
                fprintf(text,",%6d,%6d,",
                                wcx2utm(pt->x),wcy2utm(pt->y));
             fprintf(text,"%2d,%3d,",pt->font_size,pt->num_chars);
             fprintf(text,"%15s,",*(font_table + pt->font));
                j=20;
                for (k = 0; k < pt->fill_type_len; k++)
                {
                    if (pt->fill_type[j+k] > 31)
                    {
                        fprintf(text,"%c",pt->fill_type[j+k]);
                    }
                }
             fprintf(text,",");
             maxtext = pt->num_chars;
             if (maxtext > 30) maxtext = 20;
             for (l=0; l<maxtext; l++)
             {
               fprintf(text,"%c",cdb_buf[point_idx + l]);
             }
             point_idx += pt->num_chars;
                fprintf(text,",%10s,%10s\n",
                            *(color_table + pt->text_color),
                     *(color_table + pt->fill_color));
                pt++;
            }
        }
    }
      pcmi++;
}
close(fd);
fclose(cm_index_file);
fflush(master);
fflush(arc);
fflush(circle);
```

```
    fflush(ellipse);
    fflush(line);
    fflush(point);
    fflush(rectangle);
    fflush(polyline);
    fflush(polygon);
    fflush(text);
    new_index += bytes/18;
    goto get_out;
    finished:
    fclose(master);
    fclose(arc);
    fclose(circle);
    fclose(ellipse);
    fclose(line);
    fclose(point);
    fclose(rectangle);
    fclose(polyline);
    fclose(polygon);
    fclose(text);
    get_out:;

}
```

CM_INDEX.H

```
/*********************************************************/
/*  Control Measure Index file definition                */
/*********************************************************/
struct  cm_index
{
    int         cm_id;
    int         filler1;
    int         msg_usr_len;
    char        msg_usr_id[40];
    int         start_seconds;
    int         end_seconds;
    short       force;
    short       echelon;
    short       category;
    short       status;
    int         file_index;
};
```

CM_HEADER.H

```
struct  cm_header
{
    int      length;
    short arc;
    short circle;
    short ellipse;
    short line;
    short point;
    short polyline;
    short polygon;
    short rect;
    short text;
};
struct  cm_arc
{
    int         x1;
    int         y1;
    int         x2;
```

```
    int         y2;
    int         filler1;
    int         line_type_len;
    char        line_type[36];
    unsigned    color   :8;
    unsigned    draw_flags  :8;
    unsigned    :0;
};
struct  cm_circle
{
    int         x1;
    int         y1;
    int         x2;
    int         y2;
    int         filler1;
    int         line_type_len;
    char        line_type[36];
    int         filler2;
    int         fill_type_len;
    char        fill_type[36];
    unsigned       color :8;
    unsigned       fill_color :8;
    unsigned       draw_flags :8;
    unsigned       :0;
};
struct  cm_ellipse
{
    int         x1;
    int         y1;
    int         x2;
    int         y2;
    int         filler1;
    int         line_type_len;
    char        line_type[36];
    int         filler2;
    int         fill_type_len;
    char        fill_type[36];
    int         rotation;
    unsigned       color :8;
    unsigned       fill_color :8;
    unsigned       draw_flags :8;
    unsigned       :0;
};
struct  cm_line
{
    int         x1;
    int         y1;
    int         x2;
    int         y2;
    int         filler1;
    int         line_type_len;
    char        line_type[36];
    unsigned    color   :8;
    unsigned    draw_flags  :8;
    unsigned    :0;
};
struct  cm_point
{
    int         filler1;
    int         symbol_length;
    char        symbol[36];
    int         x;
    int         y;
    unsigned    color :8;
    unsigned    mark :8;
```

```
    unsigned    draw_flags :8;
    unsigned       :0;
};
struct  cm_polyline
{
    short       num_points;
    short       point_index;
    int         filler1;
    int         line_type_len;
    char        line_type[36];
    unsigned    color   :8;
    unsigned    draw_flags  :8;
    unsigned    :0;
};
struct  cm_polygon
{
    short       num_points;
    short       point_index;
    int         filler1;
    int         line_type_len;
    char        line_type[36];
    int            filler2;
    int            fill_type_len;
    char     fill_type[36];
    unsigned    color        :8;
    unsigned       fill_color  :8;
    unsigned    draw_flags  :8;
    unsigned    :0;
};
struct  cm_rect
{
    int         x1;
    int         y1;
    int         x2;
    int         y2;
    int         filler1;
    int         line_type_len;
    char        line_type[36];
    int            filler2;
    int            fill_type_len;
    char     fill_type[36];
    unsigned    color        :8;
    unsigned       fill_color  :8;
    unsigned    draw_flags  :8;
    unsigned    :0;
};
struct      cm_text
{
    int            x;
    int            y;
    int            font_size;
    short    num_chars;
    short    char_index;
    int            filler1;
    int            fill_type_len;
    char     fill_type[36];
    unsigned       font       :8;
    unsigned       text_color  :8;
    unsigned       fill_color  :8;
    unsigned       draw_flags  :8;
};

char    *d_f_table[]        =   {"draw normal       ",
                                 "draw alt color    ",
                                 "draw dingers      ",
```

```
                                "no display         ",
                                "deleted            "};

char    *color_table[]      =   {"none              ",
                                 "no color          ",
                                 "aquamarine        ",
                                 "medium aquamarine ",
                                 "black             ",
                                 "blue              ",
                                 "catet blue        ",
                                 "corn flower blue  ",
                                 "dark slate blue   ",
                                 "light blue        ",
                                 "light steel blue  ",
                                 "medium blue       ",
                                 "medium slate blue ",
                                 "midnight blue     ",
                                 "navy blue         ",
                                 "navy              ",
                                 "sky blue          ",
                                 "slate blue        ",
                                 "steel blue        ",
                                 "coral             ",
                                 "cyan              ",
                                 "fire brick        ",
                                 "gold              ",
                                 "goldenrod         ",
                                 "medium goldenrod  ",
                                 "green             ",
                                 "dark green        ",
                                 "dark olive green  ",
                                 "forest green      ",
                                 "lime green        ",
                                 "med. forest green ",
                                 "medium sea green  ",
                                 "med spring green  ",
                                 "pale green        ",
                                 "sea green         ",
                                 "spring green      ",
                                 "yellow green      ",
                                 "dark slate green  ",
                                 "dark slate gray   ",
                                 "dim grey          ",
                                 "dim gray          ",
                                 "light grey        ",
                                 "light gray        ",
                                 "khaki             ",
                                 "magenta           ",
                                 "maroon            ",
                                 "orange            ",
                                 "orchid            ",
                                 "dark orchid       ",
                                 "medium orchid     ",
                                 "pink              ",
                                 "plum              ",
                                 "red               ",
                                 "indian red        ",
                                 "medium violet red ",
                                 "orange red        ",
                                 "violet red        ",
                                 "salmon            ",
                                 "sienna            ",
                                 "tan               ",
                                 "thistle           ",
                                 "turquoise         ",
```

```
                                "dark turquoise    ",
                                "medium turquoise  ",
                                "violet            ",
                                "blue violet       ",
                                "wheat             ",
                                "white             ",
                                "yellow            ",
                                "green yellow      "};
char   *font_table[]      =     {"courier           ",
                                "courier bold      ",
                                "courier oblique   ",
                                "courier bold obli.",
                                "helvetica         ",
                                "helvetica bold    ",
                                "helvetica oblique ",
                                "helvetica bold obl",
                                "times roman       ",
                                "times bold        ",
                                "times italic      ",
                                "times bold italic ",
                                "boston            ",
                                "no change         "};
```

PLANNED_TARGETS.C

```
#include        <stdio.h>
#include        <stdlib.h>
#include        "planned_target.h"
#include    "ntc_time.h"

void    planned_targets(pt_path)
char   *pt_path;
{
    int         target_buf[125000];
    int        *p_tbuf  =   target_buf;
    int         byte_knt,   i,  j,  k;

    struct      ppt     *p_pt;
    struct      date    *ntc_time();
    struct      date    *pt_start[7500];
    struct      date    *pt_end[7500];

    FILE                *planned_target_file;
    FILE                *pt_stream;

    planned_target_file =   fopen(pt_path,"r");
    if  (planned_target_file == (FILE *) NULL)
    {
        printf("\n open failure on: %s",pt_path);
        exit();
    }
    pt_stream   =   fopen("iftt.dat","w");
    if  (pt_stream == (FILE *) NULL)
    {
        printf("\n open failure on iftt.dat ");
        exit();
    }

    byte_knt    =   fread(target_buf,
                          sizeof *target_buf,
                          125000,
                          planned_target_file);
```

```
    p_pt    =   (struct ppt *) p_tbuf;

    for (i = 0; i < byte_knt/28; i++)
    {
        p_pt->start_seconds += 3*31536000 + 86400;
        p_pt->end_seconds   += 3*31536000 + 86400;
            fprintf(pt_stream,"%d,",p_pt->index);
            fprintf(pt_stream,"%c,",force_code[p_pt->force]);
            pt_start[i] = (struct date *) malloc(sizeof(struct date));
            pt_start[i] = ntc_time(p_pt->start_seconds);
            fprintf(pt_stream,"%s,",pt_start[i]->date_time);
            pt_end[i]   = (struct date *) malloc(sizeof(struct date));
            pt_end[i]   = ntc_time(p_pt->end_seconds);
            fprintf(pt_stream,"%s,",pt_end[i]->date_time);

            for (j = 0; j < 6; j++)
            {
                if  (p_pt->target_name[j] <= 32)
                {
                    p_pt->target_name[j] = 32;
                }
                fprintf(pt_stream,"%c",p_pt->target_name[j]);
            }
            fprintf(pt_stream,",");

            for (k = 0; k < 20; k++)
            {
                if  (p_pt->origionator[k] < 31)
                {
                    p_pt->origionator[k] = 32;
                }
                fprintf(pt_stream,"%c",p_pt->origionator[k]);
            }
            fprintf(pt_stream,",%s",*(definition_table + p_pt->definition));
            fprintf(pt_stream,",%6d",wcx2utm(p_pt->x_coordinate));
            fprintf(pt_stream,",%6d\n",wcy2utm(p_pt->y_coordinate));

        ++p_pt;
    }
    fclose(pt_stream);
}
```

PLANNED_TARGETS.H

```
/*********************************************************/
/*  Pre Planned Target file definition                   */
/*********************************************************/
struct ppt
{
    int         index;
    int         start_seconds;
    int         end_seconds;
    int         filler1[7];
    char        target_name[8];
    unsigned    force    :   8;
    unsigned             :   0;
    int         filler2[7];
    char        origionator[20];
    unsigned    definition  :   8;
    unsigned             :   0;
    int         x_coordinate;
    int         y_coordinate;
};
```

```
char    force_code[5]   =   {'U','B','O','W','L'};

char    *definition_table[] =   {"Undefined   ",
                                 "ADA Unknown ",
                                 "ADD LF      ",
                                 "ADA MDM     ",
                                 "ADA HV      ",
                                 "ADA MSL     ",
                                 "ADA POS     ",
                                 "Undefined   ",
                                 "Undefined   ",
                                 "Undefined   ",
                                 "ARMOR Unk   ",
                                 "ARMOR LT    ",
                                 "ARMOR MDM   ",
                                 "ARMOR HV    ",
                                 "ARMOR APC   ",
                                 "ARMOR POS   ",
                                 "Undefined   ",
                                 "Undefined   ",
                                 "Undefined   ",
                                 "Undefined   ",
                                 "ARTY Unk    ",
                                 "ARTY LT     ",
                                 "ARTY MDM    ",
                                 "ARTY HV     ",
                                 "ARTY POS    ",
                                 "Undefined   ",
                                 "Undefined   ",
                                 "Undefined   ",
                                 "Undefined   ",
                                 "Undefined   ",
                                 "ASSY Unk    ",
                                 "ASSY TRP    ",
                                 "ASSY TRPVEH ",
                                 "ASSY TRPMEC ",
                                 "ASSY TYPARM ",
                                 "Undefined   ",
                                 "Undefined   ",
                                 "Undefined   ",
                                 "Undefined   ",
                                 "Undefined   ",
                                 "BUILDING    ",
                                 "BRIDGE      ",
                                 "CEN         ",
                                 "EQUIPMENT   ",
                                 "MORTAR      ",
                                 "Undefined   ",
                                 "Undefined   ",
                                 "Undefined   ",
                                 "Undefined   ",
                                 "Undefined   ",
                                 "PERS Unk    ",
                                 "PERS INF    ",
                                 "PERS DP     ",
                                 "PERS PTL    ",
                                 "PERS WKPARTY",
                                 "PERS POS    ",
                                 "Undefined   ",
                                 "Undefined   ",
                                 "Undefined   ",
                                 "Undefined   ",
                                 "RKTMSL      ",
                                 "SPEC        ",
                                 "SUPPLY      ",
```

```
                                "TERRAIN     ",
                                "VEHICLE     ",
                                "PN          "};
```

GROUP_TARGETS.C

```
/*********************************************************/
/*  group_target.c                                       */
/*********************************************************/

#include        <stdio.h>
#include        <stdlib.h>
#include        "group_target.h"
#include        "ntc_time.h"

void    group_target(gt_path,start_seconds,end_seconds)

char   *gt_path;
int     start_seconds;
int     end_seconds;

{
    int     target_buf[100000];
    int    *p_tbuf = target_buf;
    int     byte_knt,   i,  j,  k,  l;

    struct  gtt     *p_gt;
    struct  date    *ntc_time();
    struct  date    *gt_start[500];
    struct  date    *gt_end[500];

    FILE   *group_target_file;
    FILE   *gt_stream;

    group_target_file   =   fopen(gt_path,"r");  /* open input and out files */
    if  (group_target_file == (FILE *) NULL)
    {
        printf("\n open failure on: ",gt_path);
        exit();
    }
    gt_stream   =   fopen("ifgt.dat","w");
    if  (gt_stream == (FILE *) NULL)
    {
        printf("\n open failure on ifgt.dat.");
        exit();
    }

    byte_knt = fread(target_buf,
                     sizeof *target_buf,
                     100000,
                     group_target_file);
    p_gt    =   (struct gtt *) p_tbuf;

    for (i = 0; i < byte_knt/113; i++)
    {
        p_gt->start_seconds += 3*31536000 + 86400;
        p_gt->end_seconds   += 3*31536000 + 86400;

        if ((p_gt->start_seconds <= end_seconds) &&
            (p_gt->end_seconds >= start_seconds))
        {
            fprintf(gt_stream,"%d",p_gt->index);
            fprintf(gt_stream,",%c",force_cd[p_gt->force]);
```

```
            gt_start[i] = (struct date *) malloc(sizeof(struct date));
            gt_start[i] = ntc_time(p_gt->start_seconds);
            fprintf(gt_stream,",%s",gt_start[i]->date_time);
            gt_end[i]   = (struct date *) malloc(sizeof(struct date));
            gt_end[i]   = ntc_time(p_gt->end_seconds);
            fprintf(gt_stream,",%s,",gt_end[i]->date_time);

            for (j = 28; j < 35; j++)
            {
                if  (p_gt->designator[j] < 32)
                {
                    p_gt->designator[j] =  32;
                }
                fprintf(gt_stream,"%c",p_gt->designator[j]);
            }
            for (k = 0; k < 10; k++)
            {
                fprintf(gt_stream,",%d",p_gt->target[k].index);
                for (l = 28; l < 35; l++)
                {
                    if  (p_gt->target[k].designator[l] < 32)
                    {
                        p_gt->target[k].designator[l] = 32;
                    }
                    fprintf(gt_stream,"%c",p_gt->target[k].designator[l]);
                }
            }
          fprintf(gt_stream,"\n");
        }
      p_gt = ++p_gt;
    }
      fclose(gt_stream);
}
```

GROUP_TARGET.H

```
/*********************************************************/
/*  Group Target file definition                         */
/*********************************************************/
struct  target_array
{
    int         index;
    char        designator[36];
};
struct  gtt
{
    int         index;
    int         start_seconds;
    int         end_seconds;
    char        designator[36];
    unsigned    force   :   8;
    unsigned            :   0;
    struct      target_array    target[10];
};
char    force_cd[]  =   {'U','B','O','W','L'};
```

WCX2UTM.C

```
#include        <stdio.h>

int     wcx2utm(wc_x)

int     wc_x;
```

```
{
    int     utm_x;
    utm_x   =   (wc_x - 6993000) % 1000000;
    return(utm_x);
}
```

WCY2UTM.C

```
#include        <stdio.h>

int     wcy2utm(wc_y)
int     wc_y;
{
    int     utm_y;
    utm_y   =    wc_y % 100000;
    if  (utm_y < 70000) utm_y   +=  100000;
    return(utm_y);
}
```

NTC_TIME.C

```
/*********************************************/
/*  ntc_time.c                                */
/*********************************************/

#include        <stdio.h>
#include        <stdlib.h>
#include        "ntc_time.h"

struct  date    *ntc_time(seconds)
int     seconds;
{
    struct      date        *t;
    char                    *monthString();
    char                    *strmmm;

    int                      seconds_in_minute   =       60;
    int                      seconds_in_hour     =     3600;
    int                      seconds_in_day      =    86400;
    int                      seconds_in_year     = 31536000;
    int                      m1, d1;

    char                     dash    =   '-';
    char                     colon   =   ':';

    t   =   (struct date *) malloc(sizeof(struct date));
    t->num_seconds  =    seconds;
    t->yy           =   (seconds/seconds_in_year);
    t->dd           =   (seconds-(t->yy * seconds_in_year)) / seconds_in_day;

    month(t->dd,t->yy,&m1,&d1);

    t->HH   =   (seconds - ((t->yy * seconds_in_year) +
                            (t->dd * seconds_in_day)))/seconds_in_hour;
    t->MM   =   (seconds - ((t->yy * seconds_in_year) +
                            (t->dd * seconds_in_day)  +
                            (t->HH * seconds_in_hour)))/seconds_in_minute;
    t->SS   =    seconds - ((t->yy * seconds_in_year) +
                            (t->dd * seconds_in_day)  +
                            (t->HH * seconds_in_hour) +
                            (t->MM * seconds_in_minute));
    t->yy = t->yy + 1988;
```

```
    t->mmm = m1;
    t->dd  = d1;

    strmmm  =   monthString(t->mmm);
    t->date_time    =   malloc(25);
    sprintf(t->date_time,"%02d%c%.3s%c%4d %02d%c%02d%c%02d\0",
                        t->dd,dash,strmmm,dash,t->yy,
                        t->HH,colon,t->MM,colon,t->SS);
    return(t);
}
month(days_in_year,years,months_in_year,days_in_month)
int     days_in_year;
int     years;
int     *months_in_year;
int     *days_in_month;
{
    static  int     day_table[2][13]    =
    {
        {0,31,28,31,30,31,30,31,31,30,31,30,31},
        {0,31,29,31,30,31,30,31,31,30,31,30,31}
    };
    int     i,  leap    =   0;
    if  ((years + 88)%4 == 0)   leap    =   1;
    for (i=1; days_in_year > day_table[leap][i]; i++)
    {
        days_in_year    -=  day_table[leap][i];
    }
    *months_in_year =   i;
    *days_in_month  =   days_in_year;
}
char    *monthString(immm)
int      immm;
{
    static  char    *cmmm[] =   {"unk",
                                 "Jan",
                                 "Feb",
                                 "Mar",
                                 "Apr",
                                 "May",
                                 "Jun",
                                 "Jul",
                                 "Aug",
                                 "Sep",
                                 "Oct",
                                 "Nov",
                                 "Dec"};

    return ((immm < 1 || immm > 12) ? cmmm[0] : cmmm[immm]);
}
```

NTC_TIME.H

```
struct  date
{
    int     num_seconds;
    int     dd;
    int     mmm;
    int     yy;
    int     HH;
    int     MM;
    int     SS;
    char   *date_time;
};
```

ANNEX D

SOURCE CODE AND DOCUMENTATION FOR CREATING THE MISSION DATABASE TABLES

BUILDDB

The program, 'builddb', creates the data files for the mission databases. The major source of data is the stream data files and the snapshot files for each phase in the NTC raw data. The data files created by 'builddb' are:

ESIT.DAT	Element State Initialization Table
ESUT.DAT	Element State Update Table
APLT.DAT	Air-Player Position/Location Table
GPLT.DAT	Ground-Player Position/Loction Table
FET.DAT	Fire Event Table
PET.DAT	Pairing Event Table
CT.DAT	Communications Table
MCT.DAT	Minefield Casuality Table
IFMF.DAT	Indirect Fire Missions Fired Table
MID.DAT	Mission Identification Table
FRAT.DAT	Fratricide Log Table
IFCT.DAT	Indirect Fire Casuality Table
TASK_ORG.DAT	Task Organization Table

To build an executable program to create these comma deliminated files, use the 'Makefile' found in the 'usr/builddb' directory of the SUN workstation named 'ARI2'. To use the makefile, enter the following command:

'make -f Makefile'

The contents of the makefile are:

```
wsd:
        cc -o builddb phase_set.c ntc_time.c wsd.c wcx2utm.c wcy2utm.c get_sdi.c
        get_sni.c get_sd.c get_sn.c mvbits.c -lm
```

These tables will be placed in the approiate directory named for the mission identification (if you use the FoxPro scripts provided). For example, mission N952A105 should be placed in:

h:\archive\mission\N952\N952A105

You need to make sure that this directory structure exists before executing the FoxPro programs to create the tables and fill them.

PHASE_SET.C

```
/*********************************************************/
/*  program to read the Phase Set file and construct     */
/*  path strings to open all other files needed to build */
/*  a data set for a mission database.   This is the top */
/*  program for reading the SAIC realtime data.          */
/*********************************************************/
#include        <stdio.h>
#include        <stdlib.h>
#include        "phase_set.h"
#include        "phase_info.h"
#include        "ntc_time.h"

struct          phase_info      *phase_definition();
void            planned_target();

main(argc,argv)
int             argc;
char           *argv[];
{
    int         i,  j,  k,  l,  n,    byte_knt;
    int         phase_buf[2500];
    int        *p_tbuf = phase_buf;
    int         phase_no;

    struct      phase_set       *p_pt;
    struct      date            *ntc_time();
    struct      date            *phase_start[100];
    struct      date            *phase_end[100];
    struct      phase_info      *p_phase_info;

    char       *phase_def[100];
    char       *task_org[100];
    char       *sys_weather[100];
    char       *wsd_path;
    char       *ps_file;

    FILE       *phasefile;
    FILE       *mid_stream;
    FILE       *task_stream;

    char       *p_path_1   =   {"/od/cs_archive_files/\0"};
    char       *p_path_2   =   {"/phase_files/\0"};
    char       *p_path_3   =   {"/od/ds_archive_files/\0"};
    char       *p_path_4   =   {"/display_support/\0"};
    char       *p_path_5   =   {"/control_measures/\0"};

    char       *rotation;
    char        clown[70],      dummy[70];

    short       dummy_array[100];

        if  (argc   !=  4)
        {
            printf("\n Usage: Rotation ground-logging-rate dbname");
```

```
            printf("\n where Rotation is of the form 'yy_rr'");
            printf("\n [yy is year of rotation, rr is the  ");
            printf("\n  rotation sequence number 1 through 14]");
            printf("\n and ground-logging-rate is an integer");
            printf("\n value denoting the seconds between ");
            printf("\n position / location records for ground");
            printf("\n players.  dbname is the ARI-POM research");
          printf("\n database name assigned to the phase.");
            exit();
        }
        rotation    =   (char *) malloc(8);
        rotation    =   argv[1];

        ps_file     =   (char *) malloc(100);
        strcpy(ps_file,p_path_1);
        strcat(ps_file,rotation);
        strcat(ps_file,"/phase_files/phase_set.dat\0");
        phasefile   =   fopen(ps_file,"r");
        if  (phasefile == (FILE *) NULL)
        {
            printf("\n open failure on: %s",ps_file);
            exit();
        }

        mid_stream  =   fopen("mid.dat","w");
        if  (mid_stream == (FILE *) NULL)
        {
            printf("\n open failure on mid.dat");
            exit();
        }

        task_stream =   fopen("task_org.dat","w");
        if  (task_stream == (FILE *) NULL)
        {
            printf("\n open failure on task_org.dat");
            exit();
        }
/*************************/
/*  read phase set file    */
/*************************/
        byte_knt    =   fread(phase_buf,sizeof *phase_buf,2048,phasefile);
      printf("\n bytes read in from phase file...: %d",byte_knt);
        p_pt        =   (struct phase_set *) p_tbuf;

        for (i = 0; i <= byte_knt/67; i++)
        {
            printf("\n  %4d phase: ",i*4 + 1);

            for (j = 0; j < 56; j++)
            {
                if ((p_pt->phase_name[j]   == 25) &&
                    (p_pt->phase_name[j+4] == 25))
                {
                    k = 5;
                    while   (p_pt->phase_name[j+k] != NULL)
                    {
                        clown[k-5] = p_pt->phase_name[j+k];
                        printf("\%c",p_pt->phase_name[j+k]);
                        k++;
                    }
                    clown[k-4] = '\0';
                    phase_def[i*4+1] = (char *) malloc(100);
                    task_org[i*4+1]  = (char *) malloc(100);
                    sys_weather[i*4+1] = (char *) malloc(100);
```

```
                strcpy(phase_def[i*4+1],p_path_1);
                strcat(phase_def[i*4+1],rotation);
                strcat(phase_def[i*4+1],p_path_2);
                strcat(phase_def[i*4+1],clown);
                strcpy(task_org[i*4+1],phase_def[i*4+1]);
                strcpy(sys_weather[i*4+1],phase_def[i*4+1]);
                strcat(phase_def[i*4+1],".phase_def\0");
                strcat(task_org[i*4+1],".task_org\0");
                strcat(sys_weather[i*4+1],".sys_weather\0");
                for (l = 0; l < k; l++) clown[l] = '\0';
            }
        }
        phase_start[i*4+1] = (struct date *) malloc(sizeof(struct date));
      phase_start[i*4+1] = ntc_time(p_pt->start_seconds);
        phase_end[i*4+1]   = (struct date *) malloc(sizeof(struct date));
        phase_end[i*4+1]   = ntc_time(p_pt->end_seconds);
        printf(" start time: %s",phase_start[i*4+1]->date_time);
        n = 1;
        for (j = 0; j < 200; j++)
        {
            if  ((p_pt->dontknow[j] == 25) &&
                 (p_pt->dontknow[j+4] == 25))
            {
                n += 1;
                phase_def[i*4+n] = (char *) malloc(100);
                task_org[i*4+n]  = (char *) malloc(100);
                sys_weather[i*4+n] = (char *) malloc(100);
                strcpy(phase_def[i*4+n],p_path_1);
                strcat(phase_def[i*4+n],rotation);
                strcat(phase_def[i*4+n],p_path_2);
                printf("\n  %4d sub    ",i*4+n);
                k = 5;
                while   (p_pt->dontknow[j+k] != NULL)
                {
                    printf("%c",p_pt->dontknow[j+k]);
                    dummy[k-5] = p_pt->dontknow[j+k];
                    k++;
                }
                dummy[k-4] = '\0';
                strcat(phase_def[i*4+n],dummy);
                strcpy(task_org[i*4+n],phase_def[i*4+n]);
                strcpy(sys_weather[i*4+n],phase_def[i*4+n]);
                strcat(phase_def[i*4+n],".phase_def\0");
                strcat(task_org[i*4+n],".task_org\0");
                strcat(sys_weather[i*4+n],".sys_weather\0");
                for (l = 0; l < k; l++) dummy[l] = '\0';
                j += k;
            }
        }
        p_pt = ++p_pt;
        if  (p_pt->start_seconds == 0) i = 9999;
}

printf("\n Phase or Subphase number: ");
scanf("%d",&phase_no);
printf("\n %s \n",phase_def[phase_no]);
p_phase_info = (struct phase_info *) malloc(sizeof(struct phase_info));
p_phase_info = phase_definition(phase_def[phase_no],task_stream);
wsd_path = (char *) malloc(100);
strcpy(wsd_path,p_path_3);
strcat(wsd_path,rotation);
strcat(wsd_path,"/\0");

phase_start[98] = (struct date *) malloc(sizeof(struct date));
```

```
        phase_start[98] = ntc_time(p_phase_info->start_seconds);
        phase_end[98]   = (struct date *) malloc(sizeof(struct date));
        phase_end[98]   = ntc_time(p_phase_info->end_seconds);

        fprintf(mid_stream,"%s,%s,%s,%s,%d,%s\n",
                p_phase_info->phase_name,
                p_phase_info->phase_type,
                phase_start[98]->date_time,
                phase_end[98]->date_time,
                atoi(argv[2]),
            argv[3]);

      fclose(mid_stream);

        for (i = 0; i < 100; i++)
        {
            dummy_array[i] = p_phase_info->unit_list[i];
        }
        wsd(wsd_path,
            p_phase_info->phase_file_extention,
            dummy_array,
            atoi(argv[2]));

        exit();
}

#include        <stdio.h>
#include        <stdlib.h>
#include        "phase_def.h"

struct          phase_info      *phase_definition(phase_def_file,task_stream)

char           *phase_def_file;
FILE           *task_stream;
{
    int         phase_buf[2500];
    int        *p_tbuf = phase_buf;
    int         byte_knt,   i,  j,  k,  l,  m,  n,  unit_knt;

    struct      phase_def      *p_pt;
    struct      date           *ntc_time();
    struct      date           *phase_start;
    struct      date           *phase_end;
    struct      phase_info     *p_pi;

    FILE       *phasefile1;

    phasefile1  =   fopen(phase_def_file,"r");
    if  (phasefile1 == 0)
    {
        printf("\n open failure on: %s",phase_def_file);
        exit();
    }

    byte_knt    =   fread(phase_buf,sizeof *phase_buf,2048,phasefile1);
    p_pt        =   (struct phase_def *) p_tbuf;
    p_pi        =   (struct phase_info *) malloc(sizeof(struct phase_info));

    for (j = 0; j < 48; j++)
    {
        if ((p_pt->phase_name[j] == 25) &&
            (p_pt->phase_name[j+4] == 25))
        {
            l = 5;
```

```
            while   (p_pt->phase_name[j+l] != NULL)
            {
                p_pi->phase_name[l-5] = p_pt->phase_name[j+l];
                l++;
            }
        }
    }
    m = 0;
    for (l = 36; l < 104; l++)
    {
        if  (p_pt->mission_type[l] > 31)
        {
            p_pi->phase_type[m++] = p_pt->mission_type[l];
        }
    }
    p_pi->phase_type[m++] = '\0';
    phase_start =   (struct date *) malloc(sizeof(struct date));
      /**   Dale, here is where you modify the time the program looks
            for when it creates the file name....
            on occasion, you may need to add or subtract a second to
            get the correct file name.  [jb]      */
    phase_start =   ntc_time(p_pt->start_seconds - 1);
    phase_end   =   (struct date *) malloc(sizeof(struct date));
    phase_end   =   ntc_time(p_pt->end_seconds);
    p_pi->start_seconds = p_pt->start_seconds;
    p_pi->end_seconds   = p_pt->end_seconds;
    unit_knt = 0;
    for (i = 0; i < 50; i++)
    {
        if (p_pt->unit[i].force != 0)
        {
          for (j = 0; j<60; j++)
          {

            if ((p_pt->unit[i].name[j] == 25) &&
                (p_pt->unit[i].name[j + 4] == 25))
            {
                k = 5;
                while   (p_pt->unit[i].name[j+k] != '\0')
                {
                    fprintf(task_stream,"%c",p_pt->unit[i].name[j+k]);
                    k++;
                }
                j = 99;
            }
        }
        fprintf(task_stream,",%d,%c\n",p_pt->unit[i].element_id,
                                       ft[p_pt->unit[i].force]);
        p_pi->unit_list[unit_knt] = p_pt->unit[i].element_id;
        unit_knt++;
      }
    }
      fclose(task_stream);
        sprintf(p_pi->phase_file_extention,"%02d%02d%02d_%02d:%02d\0",
            phase_start->mmm,
            phase_start->dd,
            phase_start->yy - 1900,
            phase_start->HH,
            phase_start->MM);
    return(p_pi);
}
```

PHASE_SET.H

```
struct  phase_set
{
    char        phase_name[56];
    int         start_seconds;
    int         end_seconds;
    unsigned    lf_phase    :   8;
    unsigned    submitted   :   8;
    unsigned    archived    :   8;
    unsigned                :   0;
    char        dontknow[200];
};
```

PHASE_INFO.H

```
struct  phase_info
{
    int     start_seconds;
    int     end_seconds;
    char    phase_file_extention[16];
    char    phase_name[32];
    char    phase_type[32];
    short   unit_list[200];
};
```

NTC_TIME.C

```
#include        <stdio.h>
#include        <stdlib.h>
#include        "ntc_time.h"

struct  date    *ntc_time(seconds)
int     seconds;
{
    struct      date        *t;
    char                    *monthString();
    char                    *strmmm;

    int                      seconds_in_minute   =       60;
    int                      seconds_in_hour     =     3600;
    int                      seconds_in_day      =    86400;
    int                      seconds_in_year     = 31536000;
    int                      m1, d1;

    char                     dash    =   '-';
    char                     colon   =   ':';

    t   =   (struct date *) malloc(sizeof(struct date));
    t->num_seconds  =    seconds;
    t->yy           =   (seconds/seconds_in_year);
    if (t->yy > 4)  seconds -= 86400;
    t->dd           =   (seconds-(t->yy * seconds_in_year)) / seconds_in_day;

    month(t->dd,t->yy,&m1,&d1);

    t->HH   =   (seconds - ((t->yy * seconds_in_year) +
                            (t->dd * seconds_in_day)))/seconds_in_hour;
    t->MM   =   (seconds - ((t->yy * seconds_in_year) +
                            (t->dd * seconds_in_day)  +
                            (t->HH * seconds_in_hour)))/seconds_in_minute;
    t->SS   =    seconds - ((t->yy * seconds_in_year) +
                            (t->dd * seconds_in_day)  +
                            (t->HH * seconds_in_hour) +
                            (t->MM * seconds_in_minute));
    t->yy = t->yy + 1988;
    t->mmm = m1;
    t->dd  = d1;

    strmmm  =   monthString(t->mmm);
    t->date_time    =   malloc(25);
    sprintf(t->date_time,"%02d%c%.3s%c%4d %02d%c02d%c%02d\0",
                        t->dd,dash,strmmm,dash,t->yy,
                        t->HH,colon,t->MM,colon,t->SS);
    return(t);
}
```

```
month(days_in_year,years,months_in_year,days_in_month)
int     days_in_year;
int     years;
int     *months_in_year;
int     *days_in_month;
{
    static  int     day_table[2][13]    =
    {
        {0,31,28,31,30,31,30,31,31,30,31,30,31},
        {0,31,29,31,30,31,30,31,31,30,31,30,31}
    };
    int     i,  leap    =   0;
    if  ((years + 88)%4 == 0)   leap    =   1;
    for (i=1; days_in_year > day_table[leap][i]; i++)
    {
        days_in_year    -=  day_table[leap][i];
    }
    *months_in_year =   i;
    *days_in_month  =   days_in_year;
}
char    *monthString(immm)
int      immm;
{
    static  char    *cmmm[] =   {"unk",
                                 "Jan",
                                 "Feb",
                                 "Mar",
                                 "Apr",
                                 "May",
                                 "Jun",
                                 "Jul",
                                 "Aug",
                                 "Sep",
                                 "Oct",
                                 "Nov",
                                 "Dec"};

    return ((immm < 1 || immm > 12) ? cmmm[0] : cmmm[immm]);
}
```

NTC_TIME.H

```
struct  date
{
    int     num_seconds;
    int     dd;
    int     mmm;
    int     yy;
    int     HH;
    int     MM;
    int     SS;
    char   *date_time;
};
```

WSD.C

```
#include        <stdio.h>
#include        <stdlib.h>
#include        <math.h>
#include        "esut.h"
#include        "esut_update.h"
#include        "ntc_time.h"
```

```
#include        "sni.h"
#include        "sdi.h"

int             sn_buf[96000];
int             sd_buf[40000];
int             sni_buf[1024];
int             sdi_buf[1024];

void            esit();
void            esut();
void            mvbits();
void            miles_fire();
void            new_miles_fire();
void            miles_pair();
void            miles_commo();
void            gplt();
void            aplt();
void            mct();
void            ifmf();
void               fratricide();

void            wsd(path,file_ext,units,log_rate)

char            path[100];
char            file_ext[16];
short           units[100];
int             log_rate;

{
    struct      sdi     *get_sdi();
    int                 *get_sd();
    struct      sni     *get_sni();
    int                 *get_sn();
    struct      date    *ntc_time();

    struct      sdi     *sd[200];
    struct      sni     *sn[200];

    int        *psd;
    int        *psn;
    int         i,  j,  k,  l,  m,  p,      itime;
    int         sd_element_id;

    unsigned    sd_msg_code,
                sd_msg_length;

    char     sni_path[100],
             sdi_path[100],
             sd_path[100],
             sn_path[100];

    FILE        *mfe_stream,
                *mpe_stream,
                *mce_stream,
                *gplt_stream,
                *aplt_stream,
                *esut_stream,
                *mct_stream,
                *ifmf_stream,
                *ifct_stream,
                 *frat_stream;

    union   u_tag
    {
```

```
    unsigned    short   sval[2];
    int                 ival;
    char                cval[4];
}   uval;

mfe_stream  =   fopen("fet.dat","w");
mpe_stream  =   fopen("pet.dat","w");
mce_stream  =   fopen("ct.dat","w");
gplt_stream =   fopen("gplt.dat","w");
aplt_stream =   fopen("aplt.dat","w");
esut_stream =   fopen("esut.dat","w");
mct_stream  =   fopen("mct.dat","w");
ifmf_stream =   fopen("ifmf.dat","w");
ifct_stream =   fopen("ifct.dat","w");
frat_stream =   fopen("frat.dat","w");

strcpy(sni_path,path);
strcpy(sdi_path,path);
strcpy(sn_path,path);
strcpy(sd_path,path);

for (i=0;   i<200;  i++)
{
    sn[i]   =   (struct sni *) malloc(sizeof(struct sni));
    sn[i]   =   get_sni(sni_path,file_ext);
    if  (i == 0)    itime = (sn[i]->pst)->num_seconds;

    sd[i]   =   (struct sdi *) malloc(sizeof(struct sdi));
    sd[i]   =   get_sdi(sdi_path,file_ext);

    if  (sd[i]->sd_period_offset == 0)  exit();

    psn     =   get_sn(sn_path,
                       file_ext,
                       sn[i]->sn_period_offset,
                       sn[i]->sn_period_length);

    psd     =   get_sd(sd_path,
                       file_ext,
                       sd[i]->sd_period_offset,
                       sd[i]->sd_period_length);

    if  (i == 1)    esit(psn,sn[i]->sn_period_length);

    for (j = 0; j < sd[i]->sd_period_length/4; j++)
    {
        uval.ival       =   *(psd + j);
        sd_msg_code     =   uval.sval[0];
        sd_msg_length   =   uval.sval[1];

        if  (sd_msg_code ==    750)
      {
            aplt(psd+j+1,psn,itime,aplt_stream);
        }
        for (k=0;   k<12;   k++)
        {
            if  (sd_msg_code    ==  esut_update[k][0])
            {
                uval.ival       =   *(psd+j+1);
                sd_element_id   =   uval.sval[0];
                mvbits(psd+j+1,
                       esut_update[k][1],
                       esut_update[k][2],
                       psn + (sd_element_id -1)*38,
```

```
                        esut_update[k][3],
                        esut_update[k][4],
                        esut_update[k][5]);
                }
                if  (sd_msg_code < esut_update[k][0])    k = 12;
            }
            if ((sd_msg_code    ==    75)   ||
                (sd_msg_code    ==    85)   ||
                (sd_msg_code    ==   550)   ||
                (sd_msg_code    ==   780)   ||
                (sd_msg_code    ==   800)   ||
                (sd_msg_code    ==   803)   ||
                (sd_msg_code    ==  1320))
            {
                esut(sd_element_id,psn,itime,esut_stream);
            }

            if  (sd_msg_code ==    130)
              {
ifmf(psd+j+1,psn,itime,ifmf_stream,ifct_stream);
            }
            if  (sd_msg_code ==    590)
              {
                ++itime;
                if  (itime % log_rate ==0)
                {
                    gplt(psn,sn[i]->sn_period_length/152,
                         itime,gplt_stream);
                }
            }
            if  (sd_msg_code ==    980)
            {
                    mct(psd+j+1,psn,itime,mct_stream);
            }
            if  (sd_msg_code == 1080)
            {
                    fratricide(psd+j+1,psn,itime,frat_stream);
            }
            if  (sd_msg_code ==   1085)
            {
                    miles_pair(psd+j+1,psn,itime,mpe_stream);
            }
            if  (sd_msg_code ==   1090)
            {
                    miles_commo(psd+j+1,psn,itime,mce_stream);
            }
            if  (sd_msg_code ==   1095)
            {
                    miles_fire(psd+j+1,psn,itime,mfe_stream);
            }
            if  (sd_msg_code ==   1096)
            {
                    new_miles_fire(psd+j+1,psn,itime,mfe_stream);
            }
            j += (sd_msg_length/4);
        }
    }
    fclose("fet.dat");
    fclose("pet.dat");
    fclose("aplt.dat");
    fclose("gplt.dat");
    fclose("mct.dat");
    fclose("ct.dat");
    fclose("ifct.dat");
```

```
    fclose("ifmf.dat");
    fclose("esut.dat");
    fclose("frat.dat");
}

#include        "miles_fire.h"

void    miles_fire(p_sd_msg,p_sn_buf,ftime,mfe_stream)

int    *p_sd_msg;
int    *p_sn_buf;
int     ftime;
FILE   *mfe_stream;

{
    int     i;

    struct  date        *ntc_time();
    struct  date        *fire_time;
    struct  snapshot    *pfid;    /*  pointer to firer id         */
    struct  miles_fire  *pmfe;    /*  pointer to miles fire event */

    pmfe         =  (struct miles_fire  *) p_sd_msg;
    p_sn_buf    +=  (pmfe->element_id - 1)*38;
    pfid         =  (struct snapshot *) p_sn_buf;

    fire_time   =   (struct date *) malloc(sizeof(struct date));
    fire_time   =   ntc_time(ftime);

    fprintf(mfe_stream,"%s,%4d,",fire_time->date_time,pmfe->element_id);
    for (i=28; i<35; i++)
    {
        fprintf(mfe_stream,"%c",pfid->element_description[i]);
    }
    fprintf(mfe_stream,",%c,%6d,%6d,%3d, 0, 0, 0\n",
                    force_table[pfid->force],
                         wcx2utm(pfid->x_coordinate),
                         wcy2utm(pfid->y_coordinate),
                         pmfe->weapon_type);

    free(fire_time);
}

#include        "new_miles_fire.h"

void    new_miles_fire(p_sd_msg,p_sn_buf,ftime,mfe_stream)

int    *p_sd_msg;
int    *p_sn_buf;
int     ftime;
FILE   *mfe_stream;

{
    int     i;

    struct  date            *ntc_time();
    struct  date            *fire_time;
    struct  snapshot        *pfid;    /*  pointer to firer id         */
    struct  new_miles_fire  *pmfe;    /*  pointer to miles fire event */

    pmfe         =  (struct new_miles_fire  *) p_sd_msg;
    p_sn_buf    +=  (pmfe->element_id - 1)*38;
    pfid         =  (struct snapshot *) p_sn_buf;
```

```
    fire_time    =    (struct date *) malloc(sizeof(struct date));
    fire_time    =    ntc_time(ftime);

    fprintf(mfe_stream,"%s,%4d,",fire_time->date_time,pmfe->element_id);
    for (i=28; i<35; i++)
    {
        fprintf(mfe_stream,"%c",pfid->element_description[i]);
    }
    fprintf(mfe_stream,",%c,%6d,%6d,%3d",force_table[pfid->force],
                                         wcx2utm(pfid->x_coordinate),
                                         wcy2utm(pfid->y_coordinate),
                                         pmfe->weapon_type);
    fprintf(mfe_stream,",%3d",pmfe->rounds);
    fprintf(mfe_stream,",%3d",pmfe->tads_heading);
    fprintf(mfe_stream,",%3d\n",pmfe->laser_range);

    free(fire_time);
}

#include        "miles_pair.h"

void    miles_pair(p_sd_msg,p_sn_buf,ptime,pair_stream)

int    *p_sd_msg;
int    *p_sn_buf;
int     ptime;
FILE   *pair_stream;

{
    int     i;
    int    *pfiree_sn_buf;
    int    *pfirer_sn_buf;

    double  dist,   x1, x2, y1, y2, radical;

    struct  date       *ntc_time();
    struct  date       *pair_time;
    struct  snapshot   *pfiree;
    struct  snapshot   *pfirer;
    struct  miles_pair *pmpe;

    pmpe            =   (struct miles_pair *) p_sd_msg;
    pfiree_sn_buf   =   p_sn_buf + (pmpe->firee_id -1)*38;
    pfiree          =   (struct snapshot *) pfiree_sn_buf;

    pair_time       =   (struct date *) malloc(sizeof(struct date));
    pair_time       =   ntc_time(ptime);
    fprintf(pair_stream,"%s,%4d,",pair_time->date_time,pmpe->firee_id);

    for (i = 28; i < 35; i++)
    {
        fprintf(pair_stream,"%c",pfiree->element_description[i]);
    }
    fprintf(pair_stream,",%c,%6d,%6d,%3d,%3d",force_table[pfiree->force],
                                              wcx2utm(pfiree->x_coordinate),
                                              wcy2utm(pfiree->y_coordinate),
                                              pmpe->pairing_result,
                                              pmpe->firer_weapon);
    if ((pmpe->pairing_result > 1) && (pmpe->pairing_result < 8))
    {
        pfirer_sn_buf   =   p_sn_buf + (pmpe->firer_id -1)*38;
        pfirer          =   (struct snapshot *) pfirer_sn_buf;
        fprintf(pair_stream,",%4d,",pmpe->firer_id);
```

```
        for (i=28; i<35; i++)
        {
            fprintf(pair_stream,"%c",pfirer->element_description[i]);
        }
        x1  =   (double) wcx2utm(pfirer->x_coordinate);
        x2  =   (double) wcx2utm(pfiree->x_coordinate);
        y1  =   (double) wcy2utm(pfirer->y_coordinate);
        y2  =   (double) wcy2utm(pfiree->y_coordinate);
        radical =   pow((x1 - x2), (double) 2) +
                    pow((y1 - y2), (double) 2);
        dist    =   sqrt(radical);
        fprintf(pair_stream,",%c,%6d,%6d,%s,%6.0f\n",
                force_table[pfirer->force],
                wcx2utm(pfirer->x_coordinate),
                wcy2utm(pfirer->y_coordinate),
                *(pairing_type_table + pmpe->pairing_type),
                dist);
    }
    if  ((pmpe->pairing_result < 2) || (pmpe->pairing_result > 7))
    {
        fprintf(pair_stream,", , , , , ,%s, 0\n",
              *(pairing_type_table + pmpe->pairing_type));
    }
    free    (pair_time);
}

#include        "fratricide.h"

void    fratricide(p_sd_msg,p_sn_buf,ptime,frat_stream)

int    *p_sd_msg;
int    *p_sn_buf;
int     ptime;
FILE   *frat_stream;

{
    int     i;
    int    *pfiree_sn_buf;
    int    *pfirer_sn_buf;

    double  dist,   x1, x2, y1, y2, radical;

    struct  date       *ntc_time();
    struct  date       *frat_time;
    struct  snapshot   *pfiree;
    struct  snapshot   *pfirer;
    struct  fratricide *pfrat;

    pfrat           =   (struct fratricide *) p_sd_msg;
    pfiree_sn_buf   =   p_sn_buf + (pfrat->firee -1)*38;
    pfiree          =   (struct snapshot *) pfiree_sn_buf;

    frat_time       =   (struct date *) malloc(sizeof(struct date));
    frat_time       =   ntc_time(ptime);
    fprintf(frat_stream,"%s,%4d,",frat_time->date_time,pfrat->firee);

    for (i = 28; i < 35; i++)
    {
        fprintf(frat_stream,"%c",pfiree->element_description[i]);
    }
    fprintf(frat_stream,",%c,%6d,%6d,%6d,%3d,%3d,%3d,%3d",
                                            force_table[pfrat->force],
                                            wcx2utm(pfrat->firee_x),
                                            wcy2utm(pfrat->firee_y),
```

```
                                                pfrat->firee_z,
                                                pfrat->firee_platform,
                                                pfrat->pairing_result,
                                                pfrat->firer_weapon,
                                                pfrat->firer_platform);
    if ((pfrat->pairing_result > 1) && (pfrat->pairing_result < 8))
    {
        pfirer_sn_buf   =   p_sn_buf + (pfrat->firer -1)*38;
        pfirer          =   (struct snapshot *) pfirer_sn_buf;
        fprintf(frat_stream,",%4d,",pfrat->firer);

        for (i=28; i<35; i++)
        {
            fprintf(frat_stream,"%c",pfirer->element_description[i]);
        }
        fprintf(frat_stream,",%6d,%6d,%6d,%s,%6d\n",
                wcx2utm(pfrat->firer_x),
                wcy2utm(pfrat->firer_y),
            pfrat->firer_z,
                *(pairing_type_table + pfrat->pairing_type),
                pfrat->range);
    }
    free    (frat_time);
}

void    esit(sn_e_offset,num_players)

int    *sn_e_offset;
int     num_players;

{
    int     i,  k;

    struct  snapshot   *pe;

    FILE   *esit_stream;

    esit_stream =   fopen("esit.dat","w");
    for (i=0; i<num_players/152; i++)
    {
        pe  =   (struct snapshot *) (sn_e_offset + i*38);
        fprintf(esit_stream,"%4d,%4d,%10s,",
                pe->element_id,
                pe->bunit_number,
               *(element_type_table + pe->element_type));
        for (k=0; k<56; k++)
        {
            if  (pe->element_description[k] > 31)
            {
                fprintf(esit_stream,"%c",pe->element_description[k]);
            }
        }
        fprintf(esit_stream,",%4d,%4d,%4d,%4d",pe->next_higher_line_unit,
                                                pe->next_higher_element,
                                                pe->next_lower_element,
                                                pe->sibling_element);
        fprintf(esit_stream,",%s,%s,%14s,%14s,%c,%10s,",
              *(instrumentation_table + pe->instrumentation_status),
              *(pl_loss_table + pe->pl_loss_status),
              *(rdms_player_table + pe->rdms_player_type),
              *(battle_table + pe->battle_status),
               force_table[pe->force],
              *(echelon_table + pe->echelon));
        for (k=0; k<3; k++)
```

```
        {
            fprintf(esit_stream,"%4d,%4d,",pe->weapon_system[k].w2b1,
                                           pe->weapon_system[k].w2b2);
        }
        fprintf(esit_stream,"%4d,%6s,%3d\n",
                pe->platform,
               *(mopp_level_table + pe->mopp_level),
                pe->symbol_type);
    }
    fclose(esit_stream);
}

#include        "miles_commo.h"

void        miles_commo(p_sd_msg,p_sn_buf,event_time,commo_stream)

int        *p_sd_msg;
int        *p_sn_buf;
int         event_time;
FILE       *commo_stream;

{
    int     i;

    struct  date            *ntc_time();
    struct  date            *commo_time;
    struct  snapshot        *pcid;
    struct  miles_commo     *pmce;

    pmce        =   (struct miles_commo *) p_sd_msg;
    p_sn_buf   +=   (pmce->element_id -1)*38;
    pcid        =   (struct snapshot *) p_sn_buf;

    commo_time  =   (struct date *) malloc(sizeof(struct date));
    commo_time  =   ntc_time(event_time);
    fprintf(commo_stream,"%s,%d,",commo_time->date_time,
                                  pmce->element_id);
    for (i=28; i<35; i++)
    {
        fprintf(commo_stream,"%c",pcid->element_description[i]);
    }
    fprintf(commo_stream,",%c,%6d,%6d,%5d,%1d,%13s\n",
            force_table[pcid->force],
            wcx2utm(pcid->x_coordinate),
            wcy2utm(pcid->y_coordinate),
            pmce->elasped_time,
            pmce->radio_type,
           *(transmission_table + pmce->transmission_type));

    free(commo_time);
}

void        gplt(p_sn_buf,num_players,gtime,gplt_stream)

int    *p_sn_buf;
int     num_players;
int     gtime;
FILE   *gplt_stream;

{
    int         i,  j;

    struct      snapshot    *pgplt;
    struct      date        *ntc_time();
```

```
    struct      date       *gplt_time;

    gplt_time   =   (struct date *) malloc(sizeof(struct date));
    gplt_time   =   ntc_time(gtime);

    for (i = 0; i < num_players; i++)
    {
        pgplt   =   (struct snapshot *) (p_sn_buf + i*38);
        if  ((pgplt->instrumentation_status == 1) &&
          ((pgplt->rdms_player_type > 1) &&
           (pgplt->rdms_player_type < 5)))
        {

          if  ((wcx2utm(pgplt->x_coordinate) == 4600) &&
              (wcy2utm(pgplt->y_coordinate) == 87000))
          {
          }
          else
          {
              fprintf(gplt_stream,"%s,%4d,",gplt_time->date_time,
                                              pgplt->element_id);
                for (j=28; j<35; j++)
                {
                        fprintf(gplt_stream,"%c",
                              pgplt->element_description[j]);
                }
                fprintf(gplt_stream,",%6d,%6d\n",
                        wcx2utm(pgplt->x_coordinate),
                   wcy2utm(pgplt->y_coordinate));
                   }
            }
    }
    free(gplt_time);
}

#include        "aplt.h"

void        aplt(p_sd_buf,p_sn_buf,atime,aplt_stream)

int     *p_sd_buf;
int     *p_sn_buf;
int      atime;
FILE    *aplt_stream;

{
    int         i;

    struct  snapshot    *pe;
    struct  aplt        *paplt;
    struct  date        *ntc_time();
    struct  date        *aplt_time;

    paplt       =  (struct aplt *) p_sd_buf;
    if (paplt->z_coordinate > 0)
    {
    p_sn_buf    +=  (paplt->element_id -1)*38;
    pe          =  (struct snapshot *) p_sn_buf;

        if ((abs(paplt->x_coordinate - pe->x_coordinate) +
           abs(paplt->y_coordinate - pe->y_coordinate)) > 100)
      {
          aplt_time    =    (struct date *) malloc(sizeof(struct date));
          aplt_time    =    ntc_time(atime);
          fprintf(aplt_stream,"%s,%d,",aplt_time->date_time,
```

```
                                    paplt->element_id);
            for (i=28; i<35; i++)
            {
               fprintf(aplt_stream,"%c",pe->element_description[i]);
            }
            fprintf(aplt_stream,",%6d,%6d,%d\n",
                    wcx2utm(paplt->x_coordinate),
                    wcy2utm(paplt->y_coordinate),
                    paplt->z_coordinate);
        }
        free(aplt_time);
    }
}

void    esut(element_id,p_sn_buf,utime,esut_stream)

int     element_id;
int    *p_sn_buf;
int     utime;
FILE   *esut_stream;

{
    int     k;

    struct  date        *ntc_time();
    struct  date        *esut_time;
    struct  snapshot    *pe;

    p_sn_buf   +=   (element_id -1)*38;
    pe          =   (struct snapshot *) p_sn_buf;

    esut_time   =   (struct date *) malloc(sizeof(struct date));
    esut_time   =   ntc_time(utime);

    fprintf(esut_stream,"%s,%4d,%4d,%10s,",
                esut_time->date_time,
                pe->element_id,
                pe->bunit_number,
               *(element_type_table + pe->element_type));
    for (k=0; k<56; k++)
    {
        if  (pe->element_description[k] > 31)
        {
            fprintf(esut_stream,"%c",pe->element_description[k]);
        }
    }
      fprintf(esut_stream,",%4d,%4d,%4d,%4d",
            pe->next_higher_line_unit,
          pe->next_higher_element,
          pe->next_lower_element,
          pe->sibling_element);
    fprintf(esut_stream,",%s,%s,%14s,%14s,%c,%10s,",
          *(instrumentation_table + pe->instrumentation_status),
          *(pl_loss_table + pe->pl_loss_status),
          *(rdms_player_table + pe->rdms_player_type),
          *(battle_table + pe->battle_status),
           force_table[pe->force],
          *(echelon_table + pe->echelon));
    for (k=0; k<3; k++)
    {
        fprintf(esut_stream,"%4d,%4d,",pe->weapon_system[k].w2b1,
                                       pe->weapon_system[k].w2b2);
    }
    fprintf(esut_stream,"%4d,%6s,%3d\n",
```

```
            pe->platform,
           *(mopp_level_table + pe->mopp_level),
            pe->symbol_type);
}

#include        "minefield_cas.h"

void        mct(p_sd_msg,p_sn_buf,event_time,mct_stream)

int     *p_sd_msg;
int     *p_sn_buf;
int      event_time;
FILE    *mct_stream;

{
    int         i;

    struct      date                    *ntc_time();
    struct      date                    *mct_time;
    struct      snapshot                *pcid;
    struct      minefield_casuality     *pmct;

    pmct        =   (struct minefield_casuality *) p_sd_msg;
    p_sn_buf   +=   (pmct->element_id - 1)*38;
    pcid        =   (struct snapshot *) p_sn_buf;

    mct_time    =   (struct date *) malloc(sizeof(struct date));
    mct_time    =   ntc_time(event_time);
    fprintf(mct_stream,"%s,%4d,",mct_time->date_time,
                                 pmct->element_id);
    for (i=28; i<35; i++)
    {
        fprintf(mct_stream,"%c",pcid->element_description);
    }
    fprintf(mct_stream,",%c,%6d,%6d\n",force_table[pcid->force],
                                       wcx2utm(pcid->x_coordinate),
                                       wcy2utm(pcid->y_coordinate));
    free (mct_time);
}

#include            "ifmf.h"

void    ifmf(p_sd_msg,p_sn_buf,ptime,ifmf_stream,ifct_stream)

int     *p_sd_msg;
int     *p_sn_buf;
int      ptime;
FILE    *ifmf_stream;
FILE    *ifct_stream;

{
    int                     i,  j;
    int                    *pbattery_sn_buf;
    int                    *pimpact_sn_buf;

    struct      date       *ntc_time();
    struct      date       *impact_time;
    struct      snapshot   *pbattery;
    struct      snapshot   *pimpact;
    struct      ifmf       *pifmf;

    pifmf           =   (struct ifmf *) p_sd_msg;
    pbattery_sn_buf =   p_sn_buf + (pifmf->event_fire.element_id - 1)*38;
    pbattery        =   (struct snapshot *) pbattery_sn_buf;
```

```
impact_time     =   (struct date *) malloc(sizeof(struct date));
impact_time     =   ntc_time(ptime);

fprintf(ifmf_stream,"%s,%4d,",impact_time->date_time,
                              pifmf->event_fire.element_id);
for (i=28; i<50; i++)
{
    if  (pbattery->element_description[i] > 31)
    {
        fprintf(ifmf_stream,"%c",pbattery->element_description[i]);
    }
}
fprintf(ifmf_stream,",");
for (i=28; i<36; i++)
{
    if  (pifmf->event_fire.target_number[i] > 31)
    {
        fprintf(ifmf_stream,"%c",pifmf->event_fire.target_number[i]);
    }
}
fprintf(ifmf_stream,",");
for (i=0; i< 25; i++)
{
    if  (pifmf->plan_id[i] >= 32)
    {
        fprintf(ifmf_stream,"%c",pifmf->plan_id[i]);
    }
}
fprintf(ifmf_stream,",%c,%6d,%6d,%3d,%s,%s,%6d,%6d\n",
        force_table[pifmf->force],
        wcx2utm(pifmf->event_fire.battery_x),
        wcy2utm(pifmf->event_fire.battery_y),
        pifmf->weapon,
       *(shell_table + pifmf->shell),
       *(fuse_table + pifmf->fuse),
        wcx2utm(pifmf->event_fire.impact_x),
        wcy2utm(pifmf->event_fire.impact_y));

if  (pifmf->num_casualties > 0)
{
     for (j = 0; j < pifmf->num_casualties; j++)
    {
        pimpact_sn_buf  =   p_sn_buf + (pifmf->casualties[j] - 1)*38;
        pimpact         =   (struct snapshot *) pimpact_sn_buf;
      fprintf(ifct_stream,"%s,",impact_time->date_time);
        fprintf(ifct_stream,"%4d,",pifmf->casualties[j]);
        for (i=28; i<35; i++)
        {
            fprintf(ifct_stream,"%c",pimpact->element_description[i]);
        }
        fprintf(ifct_stream,",");
        for (i=28; i<36; i++)
        {
            if  (pifmf->event_fire.target_number[i] > 31)
            {
                fprintf(ifct_stream,"%c",
                        pifmf->event_fire.target_number[i]);
            }
        }
        fprintf(ifct_stream,",");
        for (i=0; i<25; i++)
        {
            if  (pifmf->plan_id[i] > 31)
            {
```

```
                    fprintf(ifct_stream,"%c",pifmf->plan_id[i]);
                }
            }
            fprintf(ifct_stream,",%c,%6d,%6d\n",
                    force_table[pimpact->force],
                    wcx2utm(pimpact->x_coordinate),
                    wcy2utm(pimpact->y_coordinate));
        }
    }
    free(impact_time);
}
```

ESUT.H

```
#ifndef AU
/*********************************************************/
/*  Weapon system array definition in snapshot file      */
/*********************************************************/
struct  weapon_array
{
    int         w1;
    unsigned    w2b1    :   8;
    unsigned    w2b2    :   24;
};
/*********************************************************/
/*  Element State Update Table structure.  Matches the   */
/*  SAIC raw data layout in the snapshot file.           */
/*********************************************************/
struct  snapshot
{
    short           element_id;
    short           bunit_number;
    unsigned        element_type    :   8;
    unsigned        alq_136         :   1;
    unsigned        alq_144         :   1;
    unsigned        filler1         :   6;
    unsigned        rounds_quanity  :   8;
    unsigned                        :   0;
    char            element_description[56];
    int             x_coordinate;
    int             y_coordinate;
    int             z_coordinate;
    short           next_higher_line_unit;
    short           next_higher_element;
    short           next_lower_element;
    short           sibling_element;
    unsigned        mixed_unit_status:  8;
    unsigned        instrumentation_status: 8;
    unsigned        pl_loss_status  :   8;
    unsigned        air_player      :   8;
    unsigned        air_defense     :   8;
    unsigned        rdms_player_type:   8;
    unsigned        unit_flag_player:   16;
    unsigned        contamination_status:   8;
    unsigned                        :   0;
    int             chemical_contamination_level;
    unsigned        battalion_under_training    :   8;
    unsigned        battle_status   :   8;
    unsigned        nuclear_kill_status :   8;
    unsigned        force           :   8;
    unsigned        echelon         :   8;
    unsigned                        :   0;
```

```
    struct          weapon_array    weapon_system[3];
    int             filler2;
    unsigned        platform        :   8;
    unsigned        area_occupied   :   8;
    unsigned        nuclear_posture :   8;
    unsigned        mopp_level      :   8;
    unsigned        chem_alarm      :   8;
    unsigned        radiac_meter    :   8;
    unsigned        unit_radiation_status   :   8;
    unsigned        symbol_type     :   8;
    unsigned        leaf_unit       :   8;
    unsigned                        :   0;
    int             nuclear_radiation_level;
    unsigned                        :   0;
};
/*********************************************************/
/*  following are various tables that define the values  */
/*  of data fields in the Element State Update Table     */
/*********************************************************/
char    *element_type_table[]       =   {"Undefined ",
                                         "Unit      ",
                                         "Unit CP   ",
                                         "Undefined ",
                                         "Undefined "};

char    *air_player_table[]         =   {"Undefined ",
                                         "air player",
                                         "gnd player"};

char    *pl_loss_table[]            =   {"Undefined ",
                                         "Lost P/L  ",
                                         "Found P/L "};

char    *instrumentation_table[]    =   {"Undefined ",
                                         "Instrument",
                                         "Uninstrum."};

char    *mixed_unit_table[]         =   {"Undefined ",
                                         "Mixed     ",
                                         "Not Mixed "};

char    *rdms_player_table[]        =   {"Undefined     ",
                                         "don't track   ",
                                         "manpack       ",
                                         "ground vehicle",
                                         "air defense   ",
                                         "helicopter    ",
                                         "air controller",
                                         "fast flyer    "};

char    *air_defense_table[]        =   {"Undefined   ",
                                         "air defense ",
                                         "not air def."};

char    force_table[5]              =   {'U','B','O','W','L'};

char    *nuclear_kill_table[]       =   {"Undefined ",
                                         "not killed",
                                         "killed    "};

char    *battle_table[]             =   {"Undefined      ",
                                         "Operational    ",
                                         "Combat Loss    ",
                                         "Undefined      ",
```

```
                                        "Undefined      ",
                                        "Accidental kill",
                                        "Allocated CGK  ",
                                        "Unallocated CGK",
                                        "Admin. kill    ",
                                        "Mechanical down",
                                        "Mobility killed"};

char    *battalion_ut[]             =   {"Undefined         ",
                                         "Under Training    ",
                                         "Not Under Training"};

char    *echelon_table[]            =   {"Undefined ",
                                         "Platoon   ",
                                         "Company   ",
                                         "Battalion ",
                                         "Reg / Bde ",
                                         "Division  ",
                                         "Section   "};

char    *chemical_posture_table[]   =   {"Undefined ",
                                         "Open      ",
                                         "Covered   ",
                                         "Protected "};

char    *nuclear_posture_table[]    =   {"Undefined      ",
                                         "Open           ",
                                         "APC            ",
                                         "Tank           ",
                                         "Wheeled Vehicle",
                                         "Fox Hole       ",
                                         "Earth Shelter  "};

char    *area_occupied_table[]      =   {"OA Include ",
                                         "OA Exclude "};

char    *mopp_level_table[]         =   {"MOPP 0",
                                         "MOPP 1",
                                         "MOPP 2",
                                         "MOPP 3",
                                         "MOPP 4"};

char    *contamination_table[]      =   {"None    ",
                                         "Nuclear ",
                                         "Chemical",
                                         "Nuc/Chem"};

char    *leaf_unit_table[]          =   {"Undefined ",
                                         "Leaf Unit ",
                                         "Non Leaf  "};

char    *radiac_meter_table[]       =   {"Undefined        ",
                                         "with Radiac Meter",
                                         "no Radiac Meter  "};

char    *unit_radiation_table[]     =   {"Undefined    ",
                                         "Above level 2",
                                         "Below level 2"};

#endif
```

ESUT_UPDATE.H

```
/*********************************************************/
/*  The following initialization i designed to parameter-*/
/*  ize the snapshot file updates generated by the stream*/
/*  data file.  The update array is 3 dimensional, with  */
/*  the first varing on the number of different updates  */
/*  to be preformed, the second index varies o the number*/
/*  fields to move (from the stream data buffer to the   */
/*  snapshot buffer), and the last index are the data    */
/*  necessary to accomplish the move(s).  The last six   */
/*  items stored in the third dimension are:             */
/*  1)  stream data message number                       */
/*  2)  offset in words from the message header beginning*/
/*      at zero.                                         */
/*  3)  the beginning bit position in the stream data    */
/*      offset word.                                     */
/*  4)  the word offset in the snapshot vector from the  */
/*      element_id, beginning at zero.                   */
/*  5)  the beginning byte position in the snapshot      */
/*      offset word.                                     */
/*  6)  the length of the bit pattern to move from the   */
/*      stream data file to the snapshot vector buffer.  */
/*********************************************************/
int     esut_update[][6]    =   {{  75,  0,  7, 23, 31,  8},
                                 {  85,  1, 31, 24, 31, 32},
                                 { 550,  1, 31, 37, 31, 32},
                                 { 750,  1, 31, 16, 31, 32},
                                 { 750,  2, 31, 17, 31, 32},
                                 { 750,  3, 31, 18, 31, 32},
                                 { 780,  0,  7, 25, 23,  8},
                                 { 790,  1, 31, 16, 31, 32},
                                 { 790,  2, 31, 17, 31, 32},
                                 { 800,  0,  7, 25, 15,  8},
                                 { 803,  1, 31, 37, 31, 32},
                                 {1320,  0,  7, 21, 15,  8}};
```

MILES_FIRE.H

```
/*********************************************************/
/*  Miles firing Event in stream data                    */
/*********************************************************/
struct  miles_fire
{
    short           element_id;
    unsigned        air_defense  :   8;
    unsigned        weapon_type  :   8;
};
```

NEW_MILES_FIRE.H

```
/*********************************************************/
/*  New Miles Firing Event as found in the stream data   */
/*********************************************************/
struct  new_miles_fire
{
    short           element_id;
    unsigned        air_defense :   8;
    unsigned        weapon_type :   8;
    char            rounds;
    unsigned                    :   0;
    int             tads_heading;
    int             laser_range;
};
```

MILES_PAIR.H

```
/*********************************************************/
/*  Miles Pairing Event in stream data                   */
/*********************************************************/
struct  miles_pair
{
    unsigned        pairing_result  :   8;
    unsigned        pairing_class   :   8;
    short           firee_id;
    unsigned        report_type     :   8;
    unsigned        air_player      :   8;
    unsigned                        :   0;
    int             target_hole[8];
    unsigned        firer_platform  :   8;
    unsigned        filler1         :   8;
    short           firer_id;
    unsigned        firer_element   :   8;
    unsigned                        :   8;
    unsigned        firer_weapon    :   8;
    unsigned                        :   0;
    int             firer_range;
    unsigned        fic_status      :   8;
    unsigned                        :   0;
    int             miles_fic;
    unsigned        pairing_type    :   8;
    unsigned                        :   0;
};
char    *pairing_class_table[]  =   {"RDMS pairing",
                                     "LF pairing  "};

char    *report_type_table[]    =   {"Unused       ",
                                     "target only  ",
                                     "tgt & element",
                                     "element only "};

char    *fic_status_table[]     =   {"None          ",
                                     "not applicable",
                                     "partial       ",
                                     "full          "};

char    *pairing_type_table[]    =  {"Undefined     ",
                                     "Hit           ",
                                     "Kill          ",
                                     "Near Miss     ",
                                     "Mobility Kill ",
                                     "Already Killed",
                                     "Down Tgt. Hit "};
```

MILES_COMMO.H

```
/*********************************************************/
/*  Miles Communication message in stream data file      */
/*********************************************************/
struct  miles_commo
{
    short       element_id;
    unsigned        :   0;
    int         elasped_time;
    int         radio_type;
    unsigned    transmission_type   :   8;
    unsigned        :   0;
```

```
};
char    *transmission_table[]   =   {"Undefined     ",
                                     " < 55 seconds",
                                     "=> 55 seconds"};
```

APLT.H

```
/*********************************************************/
/*      Air Player Location Table update (stream data)   */
/*********************************************************/
struct  aplt
{
    short       element_id;
    unsigned    :0;
    int         x_coordinate;
    int         y_coordinate;
    int         z_coordinate;
};
```

MINEFIELD_CAS.H

```
struct  minefield_casuality
{
    short           element_id;
    unsigned        :   0;
    int             casuality_time;
};
```

IFMF.H

```
/*********************************************************/
/*  Indirect Fire Event Fire structure                   */
/*********************************************************/
struct  ef
{
    short       element_id;
    unsigned            :    0;
    int         battery_x;
    int         battery_y;
    char        target_number[36];
    int         impact_x;
    int         impact_y;
};
/*********************************************************/
/*  Indirect Fire Missions Fired structure..             */
/*********************************************************/
struct  ifmf
{
    char        event_id[48];
    char        plan_id[32];
    unsigned    force   :   8;
    unsigned    weapon  :   8;
    unsigned    shell   :   8;
    unsigned    fuse    :   8;
    int         fire_time;
    struct      ef      event_fire;
    short       num_casualties;
    short       casualties[51];
    int         percent_standing_killed;
    int         percent_prone_killed;
```

```
    int         percent_in_hole_killed;
    int         percent_tanks_killed;
    int         percent_apcs_killed;
    int         percent_wheeled_vehicles_killed;
};
char    *shell_table[]  =   {"Undefined      \0",
                             "High Explosive \0",
                             "Undefined      \0",
                             "HERAP          \0",
                             "Undefined      \0",
                             "HC             \0",
                             "Undefined      \0",
                             "Illumination   \0",
                             "Undefined      \0",
                             "White Phosp.   \0",
                             "Undefined      \0",
                             "ICM            \0",
                             "Undefined      \0",
                             "DPICM          \0",
                             "Undefined      \0",
                             "FASCAM         \0",
                             "Undefined      \0",
                             "CLGP           \0"};

char    *fuse_table[]   =   {"None           \0",
                             "Point Detonator \0",
                             "Delay Fuse     \0",
                             "Variable time  \0"};
```

FRATRICIDE.H

```
struct fratricide
{
     unsigned  force:8;
     unsigned      pairing_result:8;
      short firee;
      int           firee_x;
      int           firee_y;
      int           firee_z;
      unsigned      firee_air_type:8;
      unsigned      firee_air_def:8;
      unsigned      firee_platform:8;
     unsigned         :0;
      short         firer;
      unsigned      :0;
      int           firer_x;
      int           firer_y;
      int           firer_z;
      unsigned      firer_platform:8;
      unsigned      firer_element_type:8;
      unsigned      firer_weapon:8;
      unsigned      :0;
      int           range;
      unsigned      fic_status:8;
      unsigned      :0;
      int           miles_fic;
      unsigned      pairing_type:8;
      unsigned      :0;
};
```

WCX2UTM.C

```
#include        <stdio.h>

int     wcx2utm(wc_x)

int     wc_x;
{
    int     utm_x;
    utm_x   =   (wc_x - 6993000) % 1000000;
    return(utm_x);
}
```

WCY2UTM.C

```
#include        <stdio.h>

int     wcy2utm(wc_y)
int     wc_y;
{
    int     utm_y;
    utm_y   =    wc_y % 100000;
    if  (utm_y < 70000) utm_y   +=  100000;
    return(utm_y);
}
```

GET_SDI.C

```
#include        <stdio.h>
#include        <stdlib.h>
#include        "ntc_time.h"
#include        "sdi.h"

struct  sdi    *get_sdi(sdi_path,file_ext)

char    sdi_path[100];
char   *file_ext;

{
    struct      sdi     *s;
    struct      date    *ntc_time();
    struct      date    *st;
    struct      date    *et;

    int         c;
    static      int     buf_idx;
    static      int     open_flag   =   0;

    extern      int     sdi_buf[];

    FILE       *sdi_stream;

    if  (!open_flag)
    {
        strcat(sdi_path,"SDI.\0");
        strcat(sdi_path,file_ext);
        sdi_stream  =   fopen(sdi_path,"rb");
        if  (sdi_stream == (FILE *) NULL)
        {
            printf("\n Open failed on %s ",sdi_path);
            return(NULL);
        }
        open_flag++;
        buf_idx = 0;
```

```
        c   =   fread(sdi_buf,sizeof *sdi_buf,2048,sdi_stream);
        st  =   (struct date *) malloc(sizeof(struct date));
        st  =   ntc_time(sdi_buf[0]);
        et  =   (struct date *) malloc(sizeof(struct date));
        et  =   ntc_time(sdi_buf[1]);
        s   =   (struct sdi *) malloc(sizeof(struct sdi));
      printf("\n SDI buffer start time: %s",st->date_time);
      printf("\n SDI buffer end time:   %s",et->date_time);
      printf("\n SDI bytes read: %d",c);
    }
    else
    {
        free(s);
        s   =   (struct sdi *) malloc(sizeof(struct sdi));
    }
    buf_idx += 2;
    s->sd_period_offset =   sdi_buf[buf_idx];
    s->sd_period_length =   sdi_buf[buf_idx + 1];
    return(s);
}
```

SDI.H

```
/*********************************************************/
/*  Stream Data Index                                    */
/*********************************************************/
struct  sdi
{
    int     sd_period_offset;
    int     sd_period_length;
};
```

GET_SNI.H

```
#include        <stdio.h>
#include        <stdlib.h>
#include        "ntc_time.h"
#include        "sni.h"

struct  sni    *get_sni(sni_path,file_ext)

char    sni_path[100];
char   *file_ext;

{
    struct      sni     *s;
    struct      date    *ntc_time();
    struct      date    *st;
    struct      date    *et;

    int         c;
    static      int     buf_idx;
    static      int     open_flag   =   0;

    extern      int     sni_buf[];

    FILE       *sni_stream;

    if  (!open_flag)
    {
        strcat(sni_path,"SNI.\0");
        strcat(sni_path,file_ext);
```

```
      printf("\n %s",sni_path);
        sni_stream  =
      fopen(sni_path,"rb");
        if  (sni_stream == (FILE *) NULL)
        {
            printf("\n Open failed on %s ",sni_path);
            return(NULL);
        }
        open_flag++;
        buf_idx = 2;

        c = fread(sni_buf, sizeof *sni_buf, 1024, sni_stream);

        st  =   (struct date *) malloc(sizeof(struct date));
        st  =   ntc_time(sni_buf[0]);
        et  =   (struct date *) malloc(sizeof(struct date));
        et  =   ntc_time(sni_buf[1]);
        s   =   (struct sni *) malloc(sizeof(struct sni));
      printf("\n SNI buffer start time: %s",st->date_time);
      printf("\n SNI buffer end time:   %s",et->date_time);
      printf("\n SNI bytes read: %d",c);
    }
    else
    {
        free(s);
        s   =   (struct sni *) malloc(sizeof(struct sni));
    }
    s->pst  =   ntc_time(sni_buf[buf_idx + 3]);
    s->pet  =   ntc_time(sni_buf[buf_idx + 4]);
    s->sn_period_offset =   sni_buf[buf_idx];
    s->sn_period_length =   sni_buf[buf_idx + 1];
    s->sn_num_periods_in_offset =   sni_buf[buf_idx + 2];
    buf_idx += 5;
    return(s);
}
```

SNI.H

```
/*********************************************************/
/*  SnapShot Index                                       */
/*********************************************************/
struct  sni
{
    struct      date        *pst;
    struct      date        *pet;
    int         sn_period_offset;
    int         sn_period_length;
    int         sn_num_periods_in_offset;
};
```

GET_SD.C

```
#include        <stdio.h>
#include        <stdlib.h>
#include        "ntc_time.h"
#define         word_length 4

int            *get_sd(sd_path,file_ext,sd_offset,sd_length)

char    sd_path[100];
char   *file_ext;
int     sd_offset;
```

```
int     sd_length;

{
    struct      sd      *s;
    struct      date    *ntc_time();
    struct      date    *st;
    struct      date    *et;

    int         c;
    static      int     buf_idx;
    static      int     open_flag   =   0;

    extern      int     sd_buf[];
    int         *psd_buf = sd_buf;

    static  FILE        *sd_stream;

    if  (!open_flag)
    {
        strcat(sd_path,"SD.\0");
        strcat(sd_path,file_ext);
        sd_stream  =   fopen(sd_path,"rb");
        if  (sd_stream == (FILE *) NULL)
        {
            printf("\n Open failed on %s ",sd_path);
            return(NULL);
        }
        open_flag++;
        c   =   fread(sd_buf,sizeof *psd_buf,sd_offset/word_length,sd_stream);
        st  =   (struct date *) malloc(sizeof(struct date));
        st  =   ntc_time(sd_buf[0]);
        et  =   (struct date *) malloc(sizeof(struct date));
        et  =   ntc_time(sd_buf[1]);
      printf("\n SD start time: %s",st->date_time);
      printf("\n SD end time:   %s",et->date_time);
      printf("\n SD bytes read: %d",c);
        return(psd_buf);
    }
    lseek(sd_stream,sd_offset,0);
    c   =   fread(sd_buf,sizeof *psd_buf,sd_length/word_length,sd_stream);
    psd_buf =   &sd_buf[0];
/*    printf("\n SD bytes read: %d",c);     */
    return(psd_buf);
}
```

GET_SN.C

```
#include        <stdio.h>
#include        <stdlib.h>
#include        "ntc_time.h"
#define         word_length 4

int            *get_sn(sn_path,file_ext,sn_offset,sn_length)

char    sn_path[100];
char   *file_ext;
int     sn_offset;
int     sn_length;

{
    struct      date    *ntc_time();
    struct      date    *st;
    struct      date    *et;
```

```
    int         c;
    static      int     open_flag   =   0;

    extern      int     sn_buf[];
    int         *psn_buf = sn_buf;

    static  FILE        *sn_stream;

    if  (!open_flag)
    {
        strcat(sn_path,"SN.\0");
        strcat(sn_path,file_ext);
        sn_stream   =   fopen(sn_path,"rb");
        if  (sn_stream == (FILE *) NULL)
        {
            printf("\n Open failed on %s ",sn_path);
            return(NULL);
        }
        open_flag++;
        c   =   fread(sn_buf,sizeof *psn_buf,sn_offset/word_length,sn_stream);
        st  =   (struct date *) malloc(sizeof(struct date));
        st  =   ntc_time(sn_buf[0]);
        et  =   (struct date *) malloc(sizeof(struct date));
        et  =   ntc_time(sn_buf[1]);
      printf("\n SN start time: %s",st->date_time);
      printf("\n SN end time:   %s",et->date_time);
    }
    lseek(sn_stream,sn_offset,0);
    c   =   fread(sn_buf,sizeof *psn_buf,sn_length/word_length,sn_stream);
    psn_buf =   &sn_buf[0];
/*    printf("\n SN bytes read: %d",c);     */
    return(psn_buf);
}
```

MVBITS.C

```
mvbits(fb,fb_word,fb_bit,tb,tb_word,tb_bit,length)

unsigned        *fb;
unsigned         fb_word;
unsigned         fb_bit;
unsigned        *tb;
unsigned         tb_word;
unsigned         tb_bit;
unsigned         length;

{
    unsigned    dummy1;
    unsigned    dummy2;

    dummy1  =   (fb[fb_word] >> (fb_bit+1-length) & ~(~0 << length));
    tb[tb_word] |= dummy1 << tb_bit+1-length;
}
```

ANNEX E

FOXPRO SOFTWARD DOCUMENTATION

For the first phase of a NTC mission database, execute the following command from the command window within FoxPro for Windows:

do make_msn with "**dbrot**","**dbmission**"

where "**dbrot**" is the rotation from which the mission is being created and "**dbmission**" is the mission id (the same as the directory where the .dbf files will be created). Note, you will have to create the directory where the mission database will reside.

Jake_msn.prg

```
PARAMETER dbrot,;
          dbname

STORE 'h:\archive\mission\' + dbrot + '\' + dbname + '\' TO dbpath
STORE dbpath + 'mid.dbf' to mid
create table &mid;
        (phase_name C(20),;
         phase_type C(30),;
         starting   C(20),;
         ending     C(20),;
         log_rate   N(4),;
         dbname     C(10))
close all
STORE dbpath + 'ct.dbf' to ct
create table &ct;
        (time           C(20),;
         lpn            N(4),;
         pid            C(8),;
         side           C(1),;
         x              N(6),;
         y              N(6),;
         duration       N(3),;
         net            N(2),;
         transmission   C(15))
close all
STORE dbpath + 'aplt.dbf' to aplt
create table &aplt;
        (time           C(20),;
         lpn            N(4),;
         pid            C(8),;
         x              N(6),;
         y              N(6),;
         z              N(5))

close all
STORE dbpath + 'gplt.dbf' to gplt
create table &gplt;
        (time           C(20),;
         lpn            N(4),;
         pid            C(8),;
         x              N(6),;
         y              N(6))

close all
```

```
STORE dbpath + 'fet.dbf' to fet
create table &fet;
        (time           C(20),;
         lpn            N(4),;
         pid            C(8),;
         side           C(1),;
         x              N(6),;
         y              N(6),;
         weapon         N(3))

close all
STORE dbpath + 'pet.dbf' to pet
create table &pet;
        (time           C(20),;
         tlpn           N(4),;
         tpid           C(8),;
         tside          C(1),;
         tx             N(6),;
         ty             N(6),;
         pair_type      N(3),;
         weapon         N(3),;
         flpn           N(4),;
         fpid           C(8),;
         fside          C(1),;
         fx             N(6),;
         fy             N(6),;
         result         C(10),;
         distance       N(5))

close all
STORE dbpath + 'frat.dbf' to frat
create table &frat;
        (time           C(20),;
         tlpn           N(4),;
         tpid           C(8),;
         side           C(1),;
         tx             N(6),;
         ty             N(6),;
         tz             N(6),;
         tplatform      N(4),;
         pair_type      N(3),;
         weapon         N(3),;
         fplatform      N(4),;
         flpn           N(4),;
         fpid           C(8),;
         fx             N(6),;
         fy             N(6),;
         fz             N(6),;
         result         C(10),;
         distance       N(5))
close all
STORE dbpath + 'ifmf.dbf' to ifmf
create table &ifmf;
        (time           C(20),;
         lpn            N(4),;
         pid            C(30),;
         target         C(8),;
         plan_id        C(8),;
         side           C(1),;
         battery_x      N(6),;
         battery_y      N(6),;
         weapon         N(3),;
         shell          C(15),;
         fuse           C(15),;
         impact_x       N(6),;
         impact_y       N(6))

close all
```

```
STORE dbpath + 'ifct.dbf' to ifct
create table &ifct;
        (time           C(20),;
         lpn            N(4),;
         pid            C(30),;
         target         C(8),;
         plan_id        C(8),;
         side           C(1),;
         x              N(6),;
         y              N(6))

close all
STORE dbpath + 'esit.dbf' to esit
create table &esit;
        (lpn            N(4),;
         bunit          N(5),;
         player_type    C(10),;
         pid            C(30),;
         nhlu           N(4),;
         nhe            N(4),;
         nle            N(4),;
         sibling        N(4),;
         instrument     C(1),;
         pl_status      C(5),;
         rdms           C(15),;
         battle_status  C(15),;
         side           C(1),;
         echelon        C(10),;
         weapon_1       N(3),;
         fic_1          N(5),;
         weapon_2       N(3),;
         fic_2          N(5),;
         weapon_3       N(3),;
         fic_3          N(5),;
         platform       N(3),;
         mopp_level     C(6),;
         symbol         N(3))

close all
STORE dbpath + 'esut.dbf' to esut
create table &esut;
        (time           C(20),;
         lpn            N(4),;
         bunit          N(5),;
         player_type    C(10),;
         pid            C(30),;
         nhlu           N(4),;
         nhe            N(4),;
         nle            N(4),;
         sibling        N(4),;
         instrument     C(1),;
         pl_status      C(5),;
         rdms           C(15),;
         battle_status  C(15),;
         side           C(1),;
         echelon        C(10),;
         weapon_1       N(3),;
         fic_1          N(5),;
         weapon_2       N(3),;
         fic_2          N(5),;
         weapon_3       N(3),;
         fic_3          N(5),;
         platform       N(3),;
         mopp_level     C(6),;
         symbol         N(3))

close all
STORE dbpath + 'task_org.dbf' to task_org
```

```
create table &task_org;
        (element_desc   C(20),;
         element_id     N(4),;
         side           C(1))

close all
STORE dbpath + 'mct.dbf' to mct
create table &mct;
        (time           C(20),;
         lpn            N(4),;
         pid            C(25),;
         side           C(1),;
         x              N(6),;
         y              N(6))

close all
STORE dbpath + 'weapon.dbf' to weapon
create table &weapon;
        (weapon_desc    C(20),;
         weapon_type    N(3))

close all
STORE dbpath + 'symbol.dbf' to symbol
create table &symbol;
        (symbol_desc    C(25),;
         symbol_type    N(3))

close all
STORE dbpath + 'platform.dbf' to platform
create table &platform;
        (platform_desc    C(25),;
         platform_type    N(3))

close all
return
```

The .dbf file structures are now ready to load with the first phase data. To accomplish this, issue the following command from the FoxPro for Windows command window:

do load_msn with "dbrot","dbmission"

Where "dbrot" is the rotation from which the data are being processed and "dbmission" is the mission id (the same as the directory you create before the previous process).

load_msn.prg

```
PARAMETER       dbrot,;
                dbmsn
STORE "h:\archive\mission\" + dbrot + "\" + dbmsn TO dbpath USE
"h:\archive\mission\aridms.dbf"
STORE LEFT(dbmsn,4) + "??" + RIGHT(dbmsn,2) TO msn_id
DELETE ALL FOR LIKE(msn_id,TRIM(dbname))
PACK
append from g:\ntc\mid.dat type delimited
close all

set default to &dbpath

use MID
append from g:\ntc\mid.dat type delimited
close all
use APLT
append from g:\ntc\aplt.dat type delimited
```

```
close all
use GPLT
append from g:\ntc\gplt.dat type delimited
close all
use FET
append from g:\ntc\fet.dat type delimited
close all
use PET
append from g:\ntc\pet.dat type delimited
close all
use FRAT
append from g:\ntc\frat.dat type delimited
close all
use IFMF
append from g:\ntc\ifmf.dat type delimited
close all
use IFCT
append from g:\ntc\ifct.dat type delimited
close all
use ESIT
append from g:\ntc\esit.dat type delimited
close all
use ESUT
append from g:\ntc\esut.dat type delimited
close all
use TASK_ORG
append from g:\ntc\task_org.dat type delimited
close all
use MCT
append from g:\ntc\mct.dat type delimited
close all
use WEAPON
append from g:\ntc\weapon.sav type delimited
close all
use SYMBOL
append from g:\ntc\symbol.sav type delimited
close all
use PLATFORM
append from g:\ntc\platform.sav type delimited
close all
```

If more than one phase is to be combined to create a mission database, all subsequent phases should be processed by using the FoxPro script, append.prg. This script will only add to those tables which have event records. To execute this program, issue the following command from the FoxPro for Windows command window:

```
do addend with "dbrot","dbmission"
```

Where "dbrot" is the rotation and "dbmission" is the database you are appending data to.

append.prg

```
PARAMETER       dbrot,;
                    dbmsn
STORE "h:\archive\mission\" + dbrot + "\" + dbmsn TO dbpath USE
"h:\archive\mission\aridms.dbf"
append from g:\ntc\mid.dat type delimited
close all

set default to &dbpath
```

```
use MID
append from g:\ntc\mid.dat type delimited
close all
use APLT
append from g:\ntc\aplt.dat type delimited
close all
use GPLT
append from g:\ntc\gplt.dat type delimited
close all
use FET
append from g:\ntc\fet.dat type delimited
close all
use PET
append from g:\ntc\pet.dat type delimited
close all
use IFMF
append from g:\ntc\ifmf.dat type delimited
close all
use IFCT
append from g:\ntc\ifct.dat type delimited
close all
use ESUT
append from g:\ntc\esut.dat type delimited
close all
use MCT
append from g:\ntc\mct.dat type delimited
close all
```

The following FoxPro script is designed to create the table structure of the control measures and indirect fire. The tables are placed in the directory 'h:\archive\mission\dbrot' where dbrot is passed to the program as a parameter. To execute this program, run FoxPro for Windows, and from the command window issue the following command:

do make_rot with 'dbrot'

where 'dbrot' is the rotation of the NTC rotation being processed. i.e. N949, N94A, etc.

```
PARAMETER dbrot

STORE 'h:\archive\mission\' + dbrot + '\'  TO dbpath
STORE dbpath + 'iftt.dbf' to iftt
create table &iftt;
        (tgt_idx        N(4),;
         side           C(1),;
         starting       C(20),;
         ending         C(20),;
         target         C(6),;
         orgin          C(20),;
         definition     C(12),;
         x              N(6),;
         y              N(6))

close all
STORE dbpath + 'ifgt.dbf' to ifgt
create table &ifgt;
        (plan_idx       N(4),;
         side           C(1),;
```

```
        starting        C(20),;
        ending          C(20),;
        designator      C(7),;
        index_1         N(4),;
        target1         C(10),;
        index_2         N(4),;
        target2         C(10),;
        index_3         N(4),;
        target3         C(10),;
        index_4         N(4),;
        target4         C(10),;
        index_5         N(4),;
        target5         C(10),;
        index_6         N(4),;
        target6         C(10),;
        index_7         N(4),;
        target7         C(10),;
        index_8         N(4),;
        target8         C(10),;
        index_9         N(4),;
        target9         C(10),;
        index_10        N(4),;
        target10        C(10))

close all
STORE dbpath + 'cm_master.dbf' to cm_master
create table &cm_master;
        (cm_index       N(5),;
         oc_team        C(20),;
         oc_group       C(20),;
         starting       C(20),;
         ending         C(20),;
         side           C(1),;
         echelon        C(9),;
         bos            C(3),;
         status         C(8),;
         arc            N(4),;
         circle         N(4),;
         ellipse        N(4),;
         line           N(4),;
         point          N(4),;
         polyline       N(4),;
         polygon        N(4),;
         rectangle      N(4),;
         text           N(4))

close all
STORE dbpath + 'arc.dbf' to arc
create table &arc;
        (cm_index       N(5),;
         object         N(3),;
         x1             N(6),;
         y1             N(6),;
         x2             N(6),;
```

```
        y2              N(6),;
        line_type       C(20),;
        color           C(15))

close all
STORE dbpath + 'circle.dbf' to circle
create table &circle;
        (cm_index       N(5),;
        object          N(3),;
        x1              N(6),;
        y1              N(6),;
        x2              N(6),;
        y2              N(6),;
        line_type       C(20),;
        fill_type       C(20),;
        color           C(20),;
        fill_color      C(20))

close all
STORE dbpath + 'ellipse.dbf' to ellipse
create table &ellipse;
        (cm_index       N(5),;
        object          N(3),;
        x1              N(6),;
        y1              N(6),;
        x2              N(6),;
        y2              N(6),;
        line_type       C(20),;
        fill_type       C(20),;
        rotation        N(3),;
        color           C(20),;
        fill_color      C(20))

close all
STORE dbpath + 'line.dbf' to line
create table &line;
        (cm_index       N(5),;
        object          N(3),;
        x1              N(6),;
        y1              N(6),;
        x2              N(6),;
        y2              N(6),;
        line_type       C(20),;
        color           C(20))

close all
STORE dbpath + 'point.dbf' to point
create table &point;
        (cm_index       N(5),;
        object          N(3),;
        x               N(6),;
        y               N(6),;
        point_type      C(20),;
        color           C(20))
```

```
close all
STORE dbpath + 'polyline.dbf' to polyline
create table &polyline;
        (cm_index       N(5),;
         object         N(3),;
         seq            N(3),;
         x              N(6),;
         y              N(6),;
         line_type      C(20),;
         color          C(20))

close all
STORE dbpath + 'polygon.dbf' to polygon
create table &polygon;
        (cm_index       N(5),;
         object         N(3),;
         seq            N(3),;
         x              N(6),;
         y              N(6),;
         line_type      C(20),;
         fill_type      C(20),;
         color          C(20),;
         fill_color     C(20))

close all
STORE dbpath + 'rectangle.dbf' to rectangle
create table &rectangle;
        (cm_index       N(5),;
         object         N(3),;
         x1             N(6),;
         y1             N(6),;
         x2             N(6),;
         y2             N(6),;
         line_type      C(20),;
         fill_type      C(20),;
         color          C(20),;
         fill_color     C(20))

close all
STORE dbpath + 'text.dbf' to text
create table &text;
        (cm_index       N(5),;
         object         N(3),;
         x              N(6),;
         y              N(6),;
         font_size      N(4),;
         num_chars      N(4),;
         font_type      C(20),;
         fill_type      C(20),;
         text_data      C(40),;
         color          C(20),;
         fill_color     C(20))
close all
return
```

When the table structures have been established in the proper directory, they can then be filled with the data extracted from the NTC raw data stream. To execute this FoxPro script, you must be in FoxPro for Windows, and, from the command window issue the following command:

do load_rot with 'dbrot'

where 'dbrot' is the current NTC rotation being processed. i.e. N949m N94A, etc.

```
PARAMETER dbrot

STORE 'h:\archive\mission\' + dbrot + '\'  TO dbpath

set default to &dbpath
use IFTT
append from g:\ntc\iftt.dat type delimited
close all
use IFGT
append from g:\ntc\ifgt.dat type delimited
close all
use CM_MASTE
append from g:\ntc\master.dat type delimited
close all
use ARC
append from g:\ntc\arc.dat type delimited
close all
use CIRCLE
append from g:\ntc\circle.dat type delimited
close all
use ELLIPSE
append from g:\ntc\ellipse.dat type delimited
close all
use LINE
append from g:\ntc\line.dat type delimited
close all
use POINT
append from g:\ntc\point.dat type delimited
close all
use POLYLINE
append from g:\ntc\polyline.dat type delimited
close all
use POLYGON
append from g:\ntc\polygon.dat type delimited
close all
use RECTANGL
append from g:\ntc\rectangl.dat type delimited
close all
use TEXT
append from g:\ntc\text.dat type delimited
close all
```