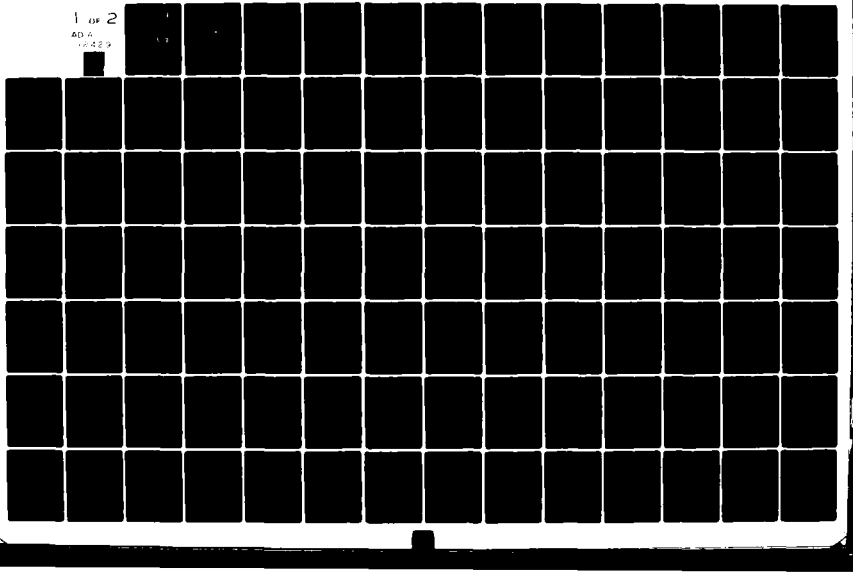
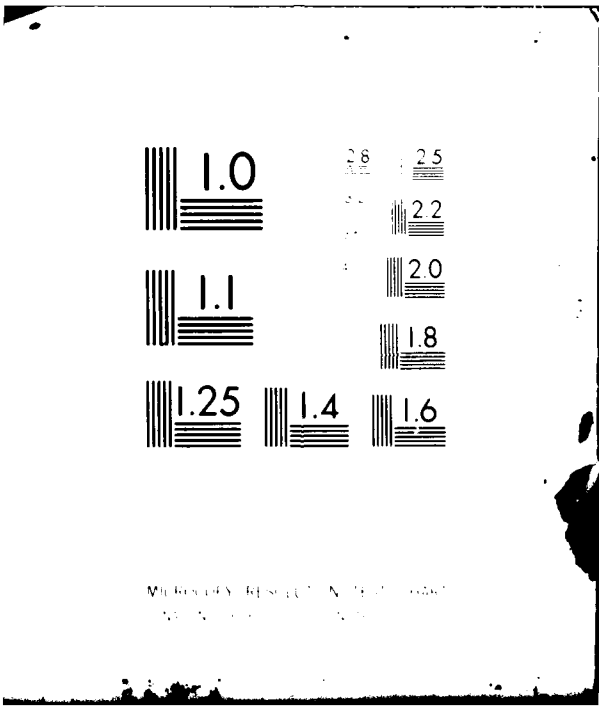
AD-A112 429 NAVAL POSTGRADUATE SCHOOL MONTEREY CA F/G 15/4
A PROTOTYPE PROGRAM FOR TARGET INFORMATION.(U)
JUN 81 R J COULTER
UNCLASSIFIED NPS52-81-007 NL




1.0
2.8
2.5
2.2
2.0
1.1
1.8
1.25
1.4
1.6

PHOTOGRAPH THIS SHEET

AD-A112429

DTIC ACCESSION NUMBER

II

LEVEL

I

INVENTORY

Naval Postgraduate School
Monterey, CA

A Prototype Program for Target Information

DOCUMENT IDENTIFICATION

Jun. 81

Coulter, Ronald J.

Rept. No. NPS52-81-007

DISTRIBUTION STATEMENT A
Approved for public release,
Distribution Unlimited

DISTRIBUTION STATEMENT

ACCESSION FOR	
NTIS GRA&I	☑
DTIC TAB	☐
UNANNOUNCED	☐
JUSTIFICATION	
BY	
DISTRIBUTION /	
AVAILABILITY CODES	
DIST	AVAIL AND/OR SPECIAL
A	

DISTRIBUTION STAMP

DTIC COPY INSPECTED

DTIC
SELECTE
S MAR 23 1982 D
D

DATE ACCESSIONED

8 2

DATE RECEIVED IN DTIC

PHOTOGRAPH THIS SHEET AND RETURN TO DTIC-DDA-2

DTIC FORM OCT 79 70A

DOCUMENT PROCESSING SHEET

NPS52-81-007

NAVAL POSTGRADUATE SCHOOL

Monterey, California

A PROTOTYPE PROGRAM FOR TARGET INFORMATION

Ronald J. Coulter

June 1981

NAVAL POSTGRADUATE SCHOOL
Monterey, California

Rear Admiral J. J. Ekelund
Superintendent

D. A. Schrady
Acting Provost

The work reported herein was supported by the Microcomputer Laboratory, Department of Computer Science, Naval Postgraduate School, Monterey, California.

This report was prepared by:

RONALD J. COULTER
LTCOL, U.S. Marine Corps

Reviewed by:

GORDON H. BRADLEY, Chairman
Department of Computer Science

Released by:

WILLIAM M. TOLLES
Dean of Research

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER NPS52-81-007	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) A Prototype Program for Target Information		5. TYPE OF REPORT & PERIOD COVERED Technical Report 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Ronald J. Coulter LTCOL USMC		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, CA 93940		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Postgraduate School Monterey, CA 93940		12. REPORT DATE June 1981 13. NUMBER OF PAGES 128
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Microcomputer, User interface, Target Information, Data Base, FSCC (Fire Support Coordination Center), Fire Support Coordination, Marine Corps, UCSD Pascal, Amphibious Operations

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This thesis presents the specification, design and implementation of a prototype microcomputer system for the target information section of the Marine Corps fire support coordination center. Currently, the target information section uses a series of index cards, handwritten lists, acetate covered battle maps and grease pencils to perform the target information functions.

The thesis examines and analyzes these functions in detail and proposes

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

20.

a solution in the form of a system, data base and interactive user design. The resultant Microcomputer System for Traget Information (MISTI) employs an ALTOS Z-80 microcomputer, the UCSD Pascal operating system, a user friendly interface and data base technology. It is proposed as an interim system until the Marine Integrated Fire and Air Support System (MIFASS) becomes operational.

TABLE OF CONTENTS

A PROTOTYPE PROGRAM FOR TARGET INFORMATION.....1

Introduction.....1

Background.....3

Statement of the Problem.....7

Nature of the Solution.....8

REFERENCES.....12

APPENDIX--A Program Source Code Listing.....14

APPENDIX--B Text File Listing.....114

DISTRIBUTION.....125

A PROTOTYPE PROGRAM FOR TARGET INFORMATION

Introduction

More and more of the applications of modern amphibious warfare have turned to computerized solutions, from real-time combat systems to the data bases that control the men, materiel and resources needed to wage war. The products of the technological explosion have enabled the Navy-Marine Corps amphibious team to do more, to do it faster and to do it with a degree of efficiency and accuracy previously unobtainable.

This evolution of modern technology has not yet reached the Marine Corps tactical command posts established on the beachhead. The target information section of the landing force fire support coordination center (FSCC) plays a significant role in the conduct of effective coordination of tactical air, artillery and naval gunfire support on targets of high priority. Yet the target information officer and his staff accomplish their important task by the use of index card files, cross-reference files, hand written lists of targets and colored grease pencils on acetate-covered tactical maps. This method is time consuming, slow in response to inquires about target information, tedious and

difficult to maintain in a current status and does not provide information in a sufficiently timely and accurate manner. It is 40 year old technology in the age of computers.

The requirement to automate many of the functions of the tactical command post has been identified and the command post of the future is being planned for and developed now. Until it arrives, there is a need to provide an interim capability to the landing force. An automated solution to the target information function will simplify the task of the target information section considerably, will provide rapid, accurate and timely target information to the members of the FSCC, and can be made operational now, five full years before the planned introduction of the computerized command post.

This report contains a prototype program for target information which will improve the operational capability of the landing force FSCC and show that the implementation of a suitable and effective target information system is possible. This implementation and design of a working prototype will increase operational effectiveness immediately as well as provide a testbed and learning model for the future automated command post. The prototype is designed to perform all the duties and functions of the target information section as currently stated in doctrinal publications. The interim system will hopefully contribute

to the development of the future system and identify areas of concern and improvement before the future Marine Corps system becomes operational.

Background

An important aspect of amphibious fire support coordination (the planning and execution of tactical air, artillery and naval gunfire support so that targets are adequately covered by a suitable weapon or group of weapons) is the function of target information. One of the major duties of the fire support coordinator, that member of the landing force staff responsible for coordination of fire support, is to ensure that the fire support coordination center receives and disseminates available target information to all staff sections and commands requiring the information. He also must work closely with the target information officer and the commander and his staff in the selection of targets and assignment of classification and attack priorities.

Target information is the direct application of combat intelligence to fire support and is a key to the proper employment of supporting arms in conjunction with each of the plans of the amphibious operation. Effective fire support coordination and the planning of amphibious operations generate a continuing requirement for target acquisition, dissemination, evaluation and recommendation

for attack.

To accomplish this important task, the commander of the amphibious task force assigns a target intelligence officer to the supporting arms coordination center (SACC). This officer operates the target information center (TIC) and works closely with the air intelligence officer, the landing force targeting representatives and the supporting arms coordinator. The commander of the landing force has a target information officer (TIO) who operates the target information section (TIS) as an integral part of the landing force fire support coordination center and a target intelligence officer who functions in the landing force intelligence center.

The Navy staff uses a computerized target information system which is part of the shipboard Amphibious Support Information System (ASIS) and maintains the list of targets as part of a data base. Target information operations in the SACC are thus computerized and, while the ASIS target system is not the most modern of data base systems, it is efficient, effective and fast. When the functional responsibility for maintaining targets is passed ashore to the landing force TIO, the computer system is replaced by an index card filing system, which , while effective, is neither fast nor efficient by comparison. Additionally, the index card system lends itself to inaccuracies and omissions in target data, particularly when the information must be

maintained in a timely manner. The tactical requirement for accurate and timely target information is no less critical or important when the landing force is on the beach, yet the system to accomplish this task is antiquated and cumbersome.

The staff of the TIS manually transfers the target information data contained in the ASIS data base to 5 by 8 inch target cards. After duplicating the entire target file, the TIS must construct a cross reference file to list the target by grid location and a cross-index file to keep track of certain types of targets. In addition to the target cards, the TIS also makes up lists of particular categories of targets which may be of interest or value to members of the FSCC.

The TIS obtains intelligence information from landing force and supporting arms agencies, converts this to target information and enters the information into the target card files. The information is made available to the supporting arms representatives in the FSCC and, based on the TIO's recommendations, a decision is made when and how to attack a particular target. Results of attacks on targets, front line reports and intelligence information are used to refine the target list and delete or deprioritize those targets that present a diminished threat to the landing force.

Access to specific information from the target list (for example, more than one category of the cross-index files) requires physically searching through each list and

constructing sub-lists to determine the appropriate information. The constant availability of timely and accurate target information is required for the effective employment of supporting arms and planning of fire support. The TIS plays a key role in providing this information and the constant process of adding to the target list, selecting targets for attack and deleting targets once neutralized is performed by the TIS staff using the target card file.

One of the most complex aspects of modern amphibious warfare is the control and coordination of supporting arms particularly in the transition of responsibility from the Navy in amphibious ships to the Marine Corps combat units ashore. The grease pencils, map boards and field radios that have served Marines so well since the days of Guadalcanal will, in the future, be eclipsed by the automated system called the Marine Integrated Fire and Air Support System (MIFASS).

MIFASS is part of the Marine Corps integrated command and control system called MTACCS (Marine Tactical Command and Control Systems), a collection of eight major systems which will give the Marines a capability of exercising real-time command and control of combat forces in the post-1980 time frame. MIFASS is designed to perform the functions of the fire support coordination center, (FSCC) the direct air support center (DASC) and, to a degree, the artillery fire direction center (FDC) at one central

location called the Fire and Air Support Center (FASC).

It is a distributed processing system in which microcomputers control interactive display devices, manage data bases, perform computational tasks and drive printers to provide hard-copy records of messages and operator decisions. It is currently in full scale engineering development with an initial operational capability planned for the 1986-1987 time frame. MIFASS addresses the requirement for target information by providing the TIO with a digital display device which will have both a graphical representation of the target on a battle map and a video screen for alphanumeric display of target information.

Statement of the Problem

An automated solution to the target information function will not be realized until the introduction of the MIFASS computers into the Fleet Marine Forces. Until such time as the system is delivered, the target information function of the FSCC is tied to the current doctrine and the target card filing system.

In this report, an interim solution to the problem of automating the target information function of the FSCC is presented. It computerizes those basic functions of the TIS in a simple, inexpensive and effective manner. It simplifies the tasks of the TIS, provides a mechanism for rapid and accurate retrieval of target information and could improve

the operational capability of the FSCC.

Nature of the Solution

The design task is broken down into three distinct parts, each of which is influenced by the dual constraints of a microcomputer environment and a friendly user interface. The design is specifically addressed in the thesis which is supported by this program listing report.

The design of the physical and logical data base is influenced by the desire to have a simple yet sufficiently informative data model, a rapid, real-time response and a restricted, single application system. The system design is influenced by the microcomputer environment which restricts the user both in main memory space and the speed of access to secondary storage and the requirement for an effective interactive system for a non-sophisticated user.

The design of the software to implement both the data base and the system is overwhelmingly influenced by the requirement that the system support real-time, interactive processing of a casual, non-programmer. Termed "fool proof" in the vernacular, it requires a sophisticated interface employing user friendly dialogue techniques to ensure that the operation is simple and efficient. For this reason, and to facilitate system portability, a microcomputer compatable high level programming language (UCSD Pascal) is used.

The report is divided into two sections. The first is the source code listing, by module, of the Microcomputer System for Target Information (MISTI) program. The second is a listing of the text files used in the interactive user interface which complements much of the prototype program. The reader is referred to the Naval Postgraduate School masters thesis [1] for additional details on the specifications, design and implementation of the system. This report is issued in conjunction with and as a key element of the thesis.

---

1. Coulter, R.J., A Microcomputer System for Target Information in the Fire Support Coordination Center: A Data Base Approach, Masters Thesis, June 1981.

Senar, A., and Sinombing, T. M., Database Management System for Microcomputers, Masters Thesis, Naval Postgraduate School, 1979.

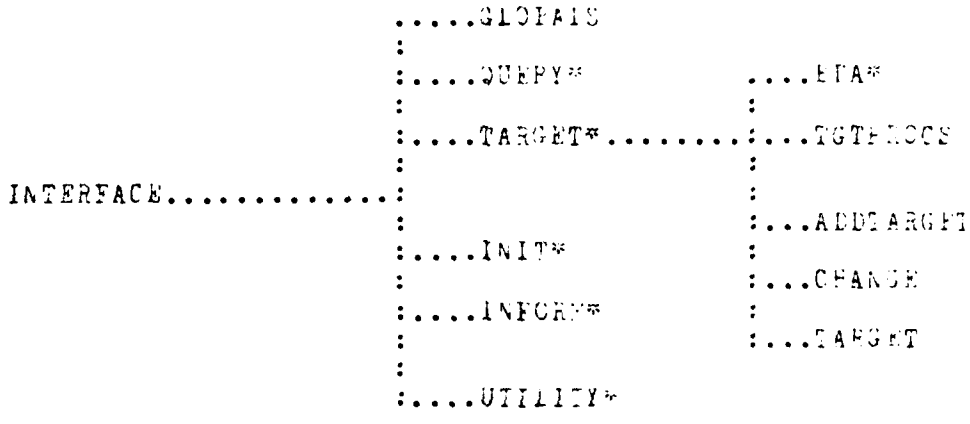
Shneiderman, B., Software Psychology: Human Factors in Computer and Information Systems, Winthrop Publications Inc., 1980.

Smith, L. B., The Use of Interactive Graphics to Solve Numerical Problems, Communications of the ACM, 1978.

Snoderass, R., A Sophisticated Microcomputer User Interface, Procedings of the Third Symposium on Small Systems, 1982.

APPENDIX--A

The source code listing for the MISTI system is organized by functional module. The list is not a compiled listing but is separate by function and by module. In an effort to decrease user confusion, the following outline shows the logical organization of the program. The segment procedures are marked by a *.



The Interface module contains include statements which instruct the compiler to compile the program in the proper logical order. The segment procedures are the first procedures to be compiled. This necessitates the declaring of many of the system routines in the beginning of the listing. The UCSD Pascal include statement allows the user to identify the volume name as well as the file name. "Store" is the name of the volume which contains the source code, thus, it appears in the include commands.

```
(*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*)

(*          MAIN PROGRAM.....INTERFACE          *)

program interface (input,output);


        (*$Istore:globals.text*)


procedure clear; forward;          procedure delay; forward;
procedure prompt; forward;         procedure spacebar; forward;
procedure select; forward;         procedure returnbar; forward;
procedure menuerror; forward;      procedure halt; forward;
procedure error1; forward;         procedure error2; forward;
procedure lines(y : integer); forward;
procedure loadmsg; forward;
procedure getfile( filename : string); forward;



        (*$Istore:query.text*)
        (*$Istore:bda.text*)
        (*$Istore:tgtprocs.text*)
        (*$Istore:addtarget.text*)
        (*$Istore:change.text*)
        (*$Istore:target.text*)
        (*$Istore:init.text*)
        (*$Istore:inform.text*)
        (*$Istore:utility.text*)


           procedure clear;

               begin
                 write(chr( 31 ));
               end;


           procedure prompt;

                begin
                  write(prompter);
                end;

           procedure spacebar;

                begin
                  writeln(spacebr);
                  prompt;
                  repeat
                  read(ch);
                  until ch = ' ';
                end;
```

```
  procedure select;

       begin
         writeln(selectop);
         prompt;
       end;


  procedure lines;

       var   yy : integer;

       begin
         for yy := 1 to y do
         writeln;
       end;


procedure returnbar;

      begin
        writeln(returner);
        prompt;
        repeat
          read(cn);
        until eoln(input);
      end;


procedure loadmsg;

begin
  lines(3);
  write(cnr(14));
  writeln('                PROGRAM BEING READ IN.....PLEASE WAIT');
  write(cnr(24));
  writeln(cnr(29));
end;


   procedure delay;

      var
        x, y, z : integer;

      begin
        z := 0;
        x := 0 ;
        repeat
          for y := 1 to 100 do
            z := z + 1 ;
          x := x + 1 ;
```

```
        until x = 10 ;
      end;


    procedure wait;

      var
        z : integer;

      begin
        lines(2);
        write('      System waiting...');
        for z := 1 to 10 do
        begin
          delay;
          write(dot);
        end;
      end;


  procedure error1;

  begin
    lines(1);
    writeln('          The proper format is a number from the');
    writeln('          options listed above. Please press the');
    writeln('          RETURN key and reenter your choice.');
    lines(1);
    prompt;
  end;


    procedure menuerror;

    begin
      lines(1);
      if eoln (input) then
      begin
        error1;
        readln;
      end
      else error1;
      readln;
    end;


    procedure error2;


begin
  lines(3);
```

```
    writeln('       The target identifier does not currently');
    writeln('       exist in the target file. Please reenter');
    writeln('       the target identifier.');
    lines(1);
    numbout := numbout + 1;
    if numbout = 3 then
    begin
      lines(1);
      writeln('       To leave this procedure, type a C');
      writeln('       followed by pressing the RETURN key.');
      lines(1);
      numbout := 0 ;
    end;
  end;



    procedure getfile;

    (*  filename : string  declared forward   *)

    var   buffer : string;
          usermessage : text;

    begin
      reset(usermessage,filename);
      repeat
        begin
          readln(usermessage,buffer);
          writeln(buffer);
        end;
      until eof(usermessage);
      close(usermessage,lock);
    end;


(*.................................................*)

    procedure buildempty;

    var  j : integer;

  begin
    with emptyIrec do
      begin
        inum := '000000';
        alt := '    ';
        loc := '00000000';
        acc := ' ';
        class := ' ';
        pri := ' ';
        ttype := ' ';
```

```
      sa := ' ';
      stat := ' ';
      desc :=                                                    ';
      rep := desc;
      mapprefer :=                                               ';
      sour := mapprefer;
      photonum :=                                                ';
      DTGact :=                ';
      photocord := loc;
      for i := 1 to numbervar do
        flag[i] := off;
      for i := 1 to 5 do
      begin
        BDA[i].DTGsurv := DTGact;
        BDA[i].fireunit :=                                       ';
        BDA[i].ntrnds                                            ';
        BDA[i].damrep :=  ';
        BDA[i].damass :=  ';
        BDA[i].BDAtext := desc;
      end;
    end;
  with emptyCrec do
    begin
      tnum :=                ';
      alt :=             ';
      loc :=                   ';
      class :=  ';
      pri :=  ';
      acc :=  ';
      ttype :=  ';
      sa :=  ';
      desc :=                                                    ';
    end;
  end;



            procedure buildmap;

var   j : integer;

begin
  i := 0;
  j := 0;
  while not eof(target) do
    begin
      seek(target,i);
      get(target);
      tgtmap[i] := target^.tgtrec.tnum;
      i := i + 1;
      j := j + 1;
      if j = 10 then
```

```
        begin
          write(dot);
          j := 0;
        end;
      end;
  end;


      procedure openfiles;

var   io : integer;

begin
  filecheck := false;
  clear;
  lines(6);
  loadmsg;
  lines 4);
  write('      Opening file...');
  write(dot);
  (*$I-*)
  reset (target,'#5:targetfile.data');
  (*$I+*)
  io := ioresult;
  if io in [4,5,9] then
  begin
    clear;
    writeln(chr(7));
    lines(5);
    writeln(chr(14));
    writeln('            ***   NO DISK IN DRIVE  5  ***');
    write(chr(24));
    writeln(chr(29));
    lines(5);
    writeln('  Insert TARGET diskette in drive 5 and restart system');
    halt;
    getout := true;
    exit(openfiles);
  end;
  if io <> 0 then
  begin
    filecheck := true;
    initialize;
  end;
  write(dot);
  (*$I-*)
  reset (QT,'#5:queryfile.data');
  (*$I+*)
  if not filecheck then
  begin
    io := ioresult;
    if io <> 0 then
     begin
```

```
      filecheck := true;
      initialize;
    end;
  end;
  write(dot);
  buildmap;
  write(dot);
  filecheck := false;
end;


    procedure password;
      var
        n,z : integer;
        user : string;
        valid : boolean;


      begin {1}
        valid := false;
        clear;
        n := 1;
        lines(5);
        while (not valid) and (n <= 5) do
        begin {2}
          writeln('    PLEASE ENTER PASSWORD AND PRESS RETURN KEY');
          prompt;
          readln(keyboard,user);
          z := 1;
          while (not valid) and (z <= 5) do
          if userid[z] = user then
          begin{3}
            passwrd := true;
            valid := true
          end{3}
          else z := z + 1 ;
          if not valid then
            begin{4}
              n:= n + 1 ;
              lines(3);
              writeln('     ** PASSWORD INCORRECT **');
              writeln(chr(7));
              lines(1);
              if n > 5 then
              begin{5}
      writeln('        Please refer to the password instructions');
      writeln('        in the target information system handbook');
      writeln('        for the proper input.');
                lines(4);
                halt;
                lines(4);
      writeln('       **  To restart system type R  **');
                exit(password);
              end;{5}
```

```
            end;{4}
          end;{2}
        end;{1}



      procedure welcome1; forward;

      procedure welcome;

 begin
   clear;
   writeln(dots);
   writeln('        WELCOME TO THE TARGET INFORMATION SYSTEM');
   writeln(dots);
   lines(1);
   writeln('This program is a prototype target information system for');
   writeln('the Fire Support Coordination Center. It is designed to be');
   writeln('used by the personnel of the target information section of');
   writeln('the landing force FSCC in accordance with the principles');
   writeln('outlined in FMFM 7-1 (Fire Support Coordination).');
   lines(2);
   writeln('WARNING:');
   write(chr(14));
   writeln('      *** THIS FILE CONTAINS CONFIDENTIAL MATERIAL ***');
   write(chr(24));
   writeln(chr(29));
   welcome1;
end;


      procedure welcome1;

    begin
writeln(' The diskette file contains targets which are normally classified');
writeln('confidential. The diskettte and all the backup copies should be ');
writeln('handled as normal confidential documents and properly safeguarded.');
writeln('Targets of a higher classification should not be entered on this');
writeln('file. Current emergency destruction procedures for confidential');
writeln('material apply. Re-initializing the system removes all classified');
writeln('information.');
     lines(1);
     spacebar;
   end;



      procedure welcome2;

    begin
      clear;
      lines(5);
```

```
writeln('    If at any time you become confused, in doubt about what');
writeln('to do next or what values to enter when you receive a prompt');
writeln('from the system ( ==> ), you can receive help or information');
writeln('by typing a ?.');
lines(4);
writeln('    If you need more information on how to operate the system,');
writeln('doctrinal guidelines for target information, security requirements');
writeln('or the types of formats used for target information, select option');
writeln('number 1 from the main command menu which follows.');
      lines(2);
      spacebar;
    end;



      procedure mminfo;

    begin
      clear;
      writeln(stars);
      lines(1);
      getfile('mminfo1.text');
      lines(1);
      spacebar;
      clear;
      lines(2);
      getfile('mminfo2.text');
      lines(1);
      spacebar;
      clear;
      lines(5);
      getfile('mminfo3.text');
      lines(5);
      spacebar;
    end;


      procedure mainmenu;

  begin
    clear;
    writeln(stars);
    writeln('     Target Information System Main Command Menu');
    writeln(dots);
    lines(1);
    writeln('    The options are:');
    lines(1);
    writeln('          1.  System Information');
    writeln('          2.  Work on Target File');
    writeln('          3.  Create a Special Target List');
    writeln('          4.  Perform Utility Functions');
    writeln('          5.  Initialize a New System');
    writeln('          6.  Information about this Menu');
    writeln('          7.  Halt Operation');
```

```
  lines(2);
end;



begin (* interface*)

   getout := false;
   restart := false;
   menuloop := false;
   numbout := 0;
   passwrd := false;
   userid[1] := 'COULTER';
   userid[2] := 'E';
   userid[3] := 'e';
   userid[4] := 'MARINE';
   userid[5] := 'marine';
   menuchar := ['1','2','3','4','5','6','7','7'];
   password;
   if not passwrd then exit(interface);
   buildempty;
   openfiles;
   if getout then exit(interface);
   welcome;
   clear;
   welcome2;
   repeat
     mainmenu;
     select;
     read(ch);
     if ch in menuchar then
       begin
       if eoln (input) then readln;
       case ch of
           '1' : begin
                   loadmsg;
                   inform;
                 end;
           '2' : begin
                   loadmsg;
                   targetmod;
                 end;
           '3' : begin
                   loadmsg;
                   query;
                 end;
           '4' : begin
                   loadmsg;
                   utility;
                 end;
           '5' : begin
                   loadmsg;
```

```
                    initialize;
                    if restart then
                      begin
                        write('       Processing....');
                        buildmap;
                        lines(2);
                        writeln('                              FUNCTION  COMPLETE');
                        lines(1);
                        spacebar;
                      end;
                  end;
            '6','?' : minfo;
            '7' : begin
                    getout := true;
                    halt;
                    clear;
                  end;
        end
      end
    else menuerror;
    until getout = true;
end.


(*+++++++++++++++++++ END OF INTERFACE +++++++++++++++++++++*)
```

```
          (*     GLOBALS.TEXT     *)

const

dot = '.';
stars = '**********************************************************************';
dots = '......................................................................';
spacebr = '        PLEASE PRESS SPACEBAR TO CONTINUE';
returner = '      PLEASE PRESS RETURN TO CONTINUE';
prompter = '      ==>';
selectop = '      PLEASE ENTER OPTION NUMBER ';
return = '  Return to Previous Menu';
numbervar = 19;
numbertot = 300;



 type
     mvalue = set of char;
     state = (on,off);

      battledam = packed record
         DTGsurv : string[7];
         fireunit : string[8];
         ntrnds : string[10];
         damrep : char;
         damass : char;
         BDAtext : string[40]
      end;

      terrec = packed record
           flag : packed array [1..numbervar] of state;
           tnum : string[6];
           loc : string[6];
           alt : string[4];
           desc : string[40];
           class : char;
           pri : char;
           stat : char;
           ttype : char;
           sa : char;
           rep : string[40];
           maprefer : string[20];
           sour : string[20];
           photonum : string[15];
           DTGact : string[7];
           photocord : string[5];
           acc : char;
           BDA : packed array [1..3] of battledam
        end;

         querytot = packed record
```

```
            flag : state;
            ttype : char;
            class : char;
            sa : char;
            pri : char;
            acc : char;
            stat : char;
            tnum : string[6];
            loc : string[6];
            alt : string[4];
            desc : string[20]
          end;

     targetmap = packed array [0..numtargt] of string[6];
     gridlocmap = packed array [0..numtargt] of string[6];
     Qtarget = packed array [0..numtargt] of querytgt;


var
     gridmap : gridlocmap;
     tgtmap : targetmap;
     target : file of record tgtrec : targrec end;
     QT : file of record querrec : querytgt end;
     nochar, ch : char;
     restart, menuloop, passwrd, getout : boolean;
     userid : packed array [1 .. 5] of string;
     menuchar, menuchar : myvalue;
     nostring, buffer, str : string;
     recnum, numbout, numbcheck, tlp, range, flag : integer;
     filecheck, current, mandatoryitem, helpme, finished : boolean;
     endEDA, ok, trap, done, quit : boolean;
     x, i, ii, EDAcounter, n : integer;
     emptyrec, currentgt : targrec;
     emptyQrec, currentQT : querytgt;
     database : Qtarget;



(*..................... SYSTEM GLOBALS .........................*)
```

```
                (*      QUERY .TEXT       *)

segment procedure query;

      var     charmenu : mvalue;
              tellused, first : boolean;
              amount, left, searcher, count, reccount : integer;
              index : char;
              Scheck, Pcheck, Acheck, statcheck : string[16];
              Tcheck, Ccheck, emptystring : string[10];
              cat : array [1..6] of string;


      procedure getdatabase;

      var  j : integer;

      begin
        clear;
        lines(4);
        writeln('   Data base being loaded into memory.....Please wait');
        lines(5);
        write('           loading...');
        i := 0;
        j := 0;
        close(QT,lock);
        reset(QT,'#5:queryfile.data');
        while not eof(QT) do
        begin
          seek(QT,i);
          get(QT);
          database[i] := QT^.querrec;
          i := i + 1;
          j := j + 1;
          if j = 10 then
          begin
            write(dot);
            j := 0;
          end;
        end;
        lines(6);
        write('                    LOADING COMPLETE');
        reccount := i;
        for i := 1 to 4 do
          delay;
      end;
```

```
procedure moresearch; forward;

procedure searchdatabase;


begin
  for i := 0 to reccount - 1 do
  begin
  case searcher of
    1 : begin
         if first then
         begin
          if database[i].ttype = index then database[i].flag := on
         end
         else if (database[i].flag = on) and (database[i].ttype <> index)
         then database[i].flag := off;
        end;
    2 : begin
         if first then
         begin
          if database[i].class = index then database[i].flag := on
         end
         else if (database[i].flag = on) and (database[i].class <> index)
         then database[i].flag := off;
        end;
    3 : begin
         if first then
         begin
          if database[i].sa = index then database[i].flag := on
         end
         else if (database[i].flag = on) and  (database[i].sa <> index)
         then database[i].flag := off;
        end;
    4 : begin
         if first then
         begin
          if database[i].pri = index then database[i].flag := on
         end
         else if (database[i].flag = on) and (database[i].pri <> index)
         then database[i].flag := off;
        end;
    5 : begin
         if first then
         begin
          if database[i].acc = index then database[i].flag := on
         end
         else if (database[i].flag = on) and (database[i].acc <> index)
         then database[i].flag := off;
        end;
    6, 7 : moresearch;
    end;
    end;
  amount := 0;
  for i := 0 to reccount - 1 do
```

```
    if database[i].flag = on then amount := amount + 1;
  lines(2);
  writeln('           Number of targets in special list is ',amount);
  lines(2);
  spacebar;
    index := ' ';
    first := false;
  end;


     procedure moresearch;

begin
  case searcher of
    6 : begin {active}
          if first then
          begin
            if (database[i].stat = '1') or (database[i].stat = '2') or
            (database[i].stat = '3') or (database[i].stat = '4') then
            database[i].flag := on
          end
          else if (database[i].flag = on) and ((database[i].stat = '5') or
            (database[i].stat = '6')) then database[i].flag := off;
        end;
    7 : begin {inactive}
          if first then
          begin
            if (database[i].stat = '5') or (database[i].stat = '6') then
            database[i].flag := on
          end
          else if (database[i].flag = on) and ((database[i].stat <> '5') or
            (database[i].stat <> '6')) then database[i].flag := off;
        end;
    end;
  end;


      procedure DBtype;

      begin
        repeat
        clear;
        lines(2);
        getfile('typemenu.text',;
        writeln('            R.',return);
        lines(1);
        select;
        read(cn);
        if cn in charmenu then
        begin
          if eoln(input) then readln;
          if cn in ['Q','q','R','r'] then exit(DBtype)
```

```
    else if ch = '7' then
      begin
        clear;
        getfile('tgttype.text');
        spacebar;
      end
    else begin
           searcher := 1;
           left := left - 1;
           index := ch;
           count := count + 1;
           case index of
             '1' : cat[count] := 'TANK';
             '2' : cat[count] := 'SEAD';
             '3' : cat[count] := 'INST';
             '4' : cat[count] := 'CBAT';
             '5' : cat[count] := ' OP ';
             '6' : cat[count] := 'TERR';
             '7' : cat[count] := 'VEH ';
             '8' : cat[count] := 'FORT';
             '9' : cat[count] := 'MISC';
           end;
           searchdatabase;
           exit(DBtype);
         end
  end
  else menuerror;
  until menuloop = true;
end;


procedure DBclass;

var  temp : string[2];

begin
  temp := '  ';
  repeat
    clear;
    lines(1);
    getfile('classmenu.text');
    writeln('            6.',return);
    lines(2);
    select;
    read(ch);
    if ch in charmenu then
    begin
      if eoln(input) then readln;
      if ch in ['5','0','q','R','r'] then exit(DBclass)
      else if ch = '7' then
        begin
          clear;
          getfile('class.text');
```

```
          spacebar;
        end
      else begin
             searcher := 2;
             left := left - 1;
             count := count + 1;
             case ch of
               '1' : index := 'A';
               '2' : index := 'B';
               '3' : index := 'C';
               '4' : index := 'D';
               '5' : index := 'E';
             end;
             case index of
               'A' : temp := ' A';
               'B' : temp := ' B';
               'C' : temp := ' C';
               'D' : temp := ' D';
               'E' : temp := ' E';
             end;
             cat[count] := concat('Class',temp);
             searchdatabase;
             exit(DBclass);
           end
    end
    else menuerror;
    until menuloop = true;
  end;


procedure DBSAassgn;

begin
  repeat
  clear;
  lines(2);
  getfile('semenu.text');
  writeln('                    R.',return);
  lines(1);
  select;
  read(ch);
  if ch in charmenu then
  begin
    if eoln(input) then readln;
    if ch in ['R','r','e','q'] then exit(DBSAassgn)
    else if ch = '?' then
      begin
        clear;
        getfile('sa.text');
        spacebar;
      end
    else begin
           searcher := 3;
```

```
            left := left - 1;
            index := cn;
            count := count - 1;
            case index of
              '1' : cat[count] := 'ARTY';
              '2' : cat[count] := 'NGF ';
              '3' : cat[count] := 'AIR ';
              '4','5','6','7' : cat[count] := 'COMB';
              '8' : cat[count] := 'OTHR';
              '9' : cat[count] := 'NONE';
            end;
            searchdatabase;
            exit(DPSAassgn);
          end
  end
  else menuerror;
  until menuloop = true;
end;


procedure DEpri;

var temp : string[3];

begin
  repeat
  clear;
  lines(1);
  getfile('tstprimenu.text');
  writeln('                5.',return);
  lines(1);
  select;
  read(cn);
  if cn in charmenu then
  begin
    if eoln(input) then readln;
    if cn in ['5','E','e','C','q'] then exit(DEpri)
    else if cn = '7' then
      begin
      clear;
      getfile('priority.text');
      spacebar;
    end
    else begin
          searcher := 4;
          index := cn;
          left := left - 1;
          count := count + 1;
          case index of
            '1' : temp := 'I  ';
            '2' : temp := 'II ';
            '3' : temp := 'III';
            '4' : temp := 'IV ';
```

```
            end;
            cat[count] := concat('Pri ',temp);
            searchdatabase;
            exit(DBpri);
          end
    end
    else menuerror;
    until menuloop = true;
end;


procedure DBacc;

begin
    repeat
    clear;
    lines(2);
    getfile('tetaccmenu.text');
    writeln('                3.',return);
    lines(1);
    select;
    read(ch);
    if ch in charmenu then
    begin
      if eoln(input) then readln;
      if ch in ['B','R','r','a','&'] then exit(DBacc)
      else if ch = 'T' then
        begin
          clear;
          getfile('tetacc.text');
          spacebar;
        end
      else begin
            searcher := 5;
            index := ch;
            left := left - 1;
            count := count + 1;
            case index of
              '1' : cat[count] := 'CONFIRMED ';
              '2' : cat[count] := 'PROBABLE  ';
              '3' : cat[count] := 'POSSIBLE  ';
              '4' : cat[count] := 'UNKNOWN   ';
            end;
            searchdatabase;
            exit(DBacc);
          end
    end
    else menuerror;
    until menuloop = true;
end;
```

```
  procedure listatus;

  var  act : string[8];
       loop : boolean;

begin
  act := '        ';
  menuchar := ['1','2','?'];
  loop := false;
  repeat
    clear;
    lines(5);
    writeln('      ENTER TARGET STATUS--ACTIVITY');
    lines(1);
    writeln('       The options are:');
    lines(1);
    writeln('          1.  Active');
    writeln('          2.  Inactive');
    lines(1);
    writeln('        PLEASE ENTER OPTION NUMBER AND PRESS RETURN');
    prompt;
    read(ch);
    if ch in menuchar then
    begin
      if not eoln(input) then readln;
      if ch = '1' then
      begin
        searcher := 6;
        searchdatabase;
        loop := true;
        act := ' ACTIVE ';
      end;
      if ch = '2' then
      begin
        searcher := 7;
        searchdatabase;
        loop := true;
        act := 'INACTIVE';
      end;
      if ch = '?' then
      begin
        lines(1);
writeln('    An Active target is one which is found in the target list or' );
writeln('the list of targets. An inactive target is in the inactfile.');
        lines(1);
        spacebar;
      end;
      end
      else if (ch in ['Q','q']) or (eoln(input)) then exit(lrstatus)
      else menuerror;
    until loop = true;
  left := left - 1;
```

```
    count := count + 1;
    cat[count] := act;
  end;



      procedure catmenu;

    begin
      writeln('              Categories for Special listing');
      writeln(lots);
      lines(1);
      writeln('  The listing can contain ',left,' items from the below menu:');
      lines(1);
      writeln('                1.  Target type                    ',Tcheck);
      writeln('                2.  Classification                 ',Ccheck);
      writeln('                3.  Supporting Air Assigned        ',Scheck);
      writeln('                4.  Priority                       ',Pcheck);
      writeln('                5.  Accuracy                       ',Acheck);
      writeln('                6.  Status                         ',Statcheck);
      writeln('              * 7.  Process Information');
      lines(1);
      writeln('        Special list currently contains ',amount,' targets.');
      if amount <= 0 then writeln('                                  Please start a new listing.');
      lines(1);
    end;



      procedure catproc;

      var  taken : string[10];

      begin
        taken := 'Already Selected';
        repeat
          if count >= 6 then
          begin
            clear;
            lines(4);
            writeln('      No more categories available for special list.');
            writeln('                Please print target listing');
            lines(2);
            spacebar;
            exit(catproc);
          end;
          clear;
          case searcher of
            0 : ;
            1 : Tcheck := taken;
            2 : Ccheck := taken;
            3 : Scheck := taken;
            4 : Pcheck := taken;
```

```
      5 : Acheck := taken;
      6, 7 : statcheck := taken;
    end;
    catmenu;
    select;
    read(ch);
    if ch in charmenu then
    begin
      if eoln(input) then readln;
      case ch of
        '1' : begin
                if Icheck = taken then tellused := true
                else DBtype;
              end;
        '2' : begin
                if Scheck = taken then tellused := true
                else DBclass;
              end;
        '3' : begin
                if Scheck = taken then tellused := true
                else DBSmassem;
              end;
        '4' : begin
                if Pcheck = taken then tellused := true
                else DBpri;
              end;
        '5' : begin
                if Acheck = taken then tellused := true
                else DBacc;
              end;
        '6' : begin
                if statcheck = taken then tellused := true
                else DBstatus;
              end;
        'P','p','R','r','Q','q' : exit(catproc);
        '7','8','9' : menuerror;
        '?' : begin
                lines(1);
                writeln('     See prior menu for information');
                lines(1);
                spacebar;
              end;
      end
    end
    else menuerror;
  if tellused then
    begin
      clear;
      lines(5);
      writeln('     Category has already been selected. Please');
      writeln('     choose another category from the unused items');
      writeln('     on the menu listing. To start a new listing,');
      writeln('     choose option 7 to return to the main menu.');
      lines(3);
```

```
            subnbr;
            deleted := false;
          end;
        until readloop = true;
      end;


procedure screenlist;

    var   star : char;
          prtbuff : string[2];
          sabuff : string[4];
          holdcar : char;
          pager : integer;


  procedure header;

      var  listing : string;

      begin
        listing := ' ';
        lines(1);
        writeln('                    SPECIAL TARGET LISTING');
        writeln('                    ----------------------');
        write('Categories:');
        if count > 4 then count := 4;
        for i := 1 to count do
          listing := concat(listing,' ',cat[i]);
        writeln(listing);
        lines(1);
        writeln('TGT NO   CL PRI  LOCATION   ALT  SAAFF  DESCRIPTION');
        writeln('------   -- ---  --------   ---- -----  -----------');
      end;


  begin{screenlist}
    clear;
    header;
    prtbuff := '    ';
    sabuff := '     ';
    pager := 5;
    for i := 1 to reccount - 1 do
    begin
      star := ' ';
      if database[i].flag = on then
      begin
        holdcar := database[i].pri;
        case holdcar of
          '1' : prtbuff := 'I   ';
          '2' : prtbuff := 'II  ';
          '3' : prtbuff := 'III ';
```

```
            '4' : prbuff := 'IV ';
          end;
          holdcar := database[i].sc;
          case holdcar of
            '1' : sabuff := 'ARTY';
            '2' : sabuff := 'NGF ';
            '3' : sabuff := 'AIR ';
            '4','5','6','7' : sabuff := 'COMB';
            '8' : sabuff := 'OTHR';
            '9' : sabuff := 'NOVE';
          end;
if (database[i].stat = '1') or (database[i].stat = '2') then star := '*';
write(database[i].tnum,star,'  ',database[i].class,'  ',prbuff,'  ');
write(database[i].loc,'  ',database[i].alt:4,'  ',sabuff,'  ');
writeln(database[i].desc);
          pager := pager + 1;
          if pager = 20 then
          begin
            lines(1);
            spacebar;
            clear;
            lines(2);
            pager := 4;
          end;
        end;
      end;
      lines(1);
      writeln('     NOTE:  * indicates target list');
      lines(1);
      spacebar;
    end;


        procedure reset;

        begin
          tellused := false;
          amount := 6;
          left := 8;
          count := 2;
          searcher := 2;
          first := true;
          Tcheck := emptystring;
          Ccheck := emptystring;
          Pcheck := emptystring;
          Acheck := emptystring;
          Scheck := emptystring;
          statcheck := emptystring;
          for i := 2 to reccount - 1 do
            database[i].flag := off;
          for i := 1 to 5 do
              cat[i] := emptystring;
        end;
```

```
procedure queryproc;

  begin
    count := 1;
    repeat
    clear;
    lines(2);
    writeln('              SPECIAL TARGET LISTING');
    writeln(dots);
    lines(2);
    writeln('       The options are:');
    lines(1);
    writeln('              1.  Form a new target listing');
    writeln('              2.  Continue to previous');
    writeln('              3.  Write the special list to the screen');
    writeln('              4.  Information about this procedure');
    writeln('              5.',return);
    lines(1);
    select;
    read(ch);
    if ch in ['1','2','3','4','5','R','r'] then
    begin
      if eoln(input) then readln;
      case ch of
        '1' : begin
                reset;
                catproc;
              end;
        '2' : catproc;
        '3' : screenlist;
        '4','?' : begin
                    clear;
                    getfile('queryinfo.txt');
                    spacebar;
                  end;
        '5','R','r' : exit(query);
      end
    end
    else menuerror;
    until menuloop = true;
  end;



      begin {query}
        reccount := 6;
        emptystring :=  '                    ';
        searcher := 0;
charmenu := ['1','2','3','4','5','6','7','8','9','R','r','?','S','s',',','.'];
        getdatabase;
        reset;
        queryproc;
```

```
end;

(*.................END  CURRY.........................*)
```

```
(*      BDA.TEXT     *)

      segment procedure targetmod;

      var
         stat1 : string[2];
         stat2 : string[3];
         ttypebuf : string[4];
         pribuf : string[5];
         sabuf : string[14];
         accbuf : string[9];
         duplicate, fetchback, outprocess, first : boolean;


      procedure fetchtgt(grid : integer); forward;
      procedure readin; forward;
      procedure checkDTG(var strng : string; var check:boolean); forward;
      procedure process; forward;
      procedure cutstring(strngsize : integer); forward;
      procedure putinfile; forward;


segment procedure newBDA;


procedure DTGofBDA;


begin
  currentgt.BDA[BDAcounter].DTGsurv := '      ';
  while not finished do
    begin
      clear;
      lines(6);
      writeln('     ENTER DTG TARGET WAS ATTACKED...6 digits and 1 letter.');
      prompt;
      readin;
      checkDTG(str,ok);
      if quit then exit(DTGofBDA);
      if helpme or not ok then
        begin
          finished := false;
          lines(1);
          getfile('dtgofbda.text');
          lines(1);
          returnbar;
        end;
        if  (flg = 2) and finished  then
        currentgt.BDA[BDAcounter].DTGsurv := str;
      end;
    end;
```

```
     procedure firineunit;

begin
  currentet.BDA[BDAcounter].fireunit := '          ';
  while not finished do
   begin
    lines(2);
    writeln('      ENTER FIRING UNIT....do not exceed 8 characters');
    prompt;
    readin;
    if helpme then
    begin
      finished := false;
      lines(1);
      getfile('unit1.text');
      lines(1);
      returnbar;
    end
    else if length(str) > 8 then
      begin
        lines(1);
        getfile('unit2.text');
        lines(1);
        finished := false;
        currentet.flag[n] := on;
        returnbar;
      end
    else if quit then exit(firineunit)
    else if (flg = 2) and finished then
    currentet.BDA[BDAcounter].fireunit := str;
  end;
end;


     procedure rounds;

begin
  currentet.BDA[BDAcounter].ntrnds := '          ';
  while not finished do
    begin
    lines(2);
    writeln('      ENTER NUMBER AND TYPE OF ROUNDS FIRED');
    prompt;
    readin;
    if helpme then
    begin
      finished := false;
      lines(1);
      getfile('rounds.text');
      lines(1);
      returnbar;
    end
    else if length(str) > 10 then
      begin
```

```
        lines(1);
        getfile('rounds1.text');
        lines(1);
        finished := false;
        currentgt.flag[n] := on;
        returntar;
      end
      else if quit then exit(rounds)
      else if (flg = 2) and finished then
      currentgt.BDA[BDAcounter].intrds := str;
    end;
end;


    procedure damagemenu( paran : integer);

    var  kind : string[8];

    begin
      if paran = 1 then kind := 'REPORTED'
      else kind := 'ASSESSED';
      clear;
      lines(5);
      writeln('      ENTER DAMAGE ',kind);
      lines(1);
      getfile('damagemen.text');
      lines(2);
    end;



    procedure damagrept;

  begin
      currentgt.BDA[BDAcounter].damrep := '9';
      repeat
        damagemenu(1);
        select;
        read(cn);
        if cn in ['1'..'5'] then
        begin
          if eoln(input) then readln;
          finished := true;
          currentgt.BDA[BDAcount].damrep := cn;
        end
        else if eoln(input) then exit(damagrept)
        else if cn in ['C','q'] then
        begin
          quit := true;
          exit(damagrept);
        end
        else if cn = '?' then
        begin
          lines(1);
```

```
            getfile('damrep.text');
            lines (1);
            returnbar;
          end
        else menuerror;
      until finished = true;
    end;


    procedure damagassd;

    begin
      currentgt.BDA[BDAcounter].damass := '9';
      repeat
        damagemenu(2);
        select;
        read(ch);
        if ch in ['1'..'8'] then
        begin
          if eoln(input) then readln;
          finished := true;
          currentgt.BDA[BDAcount].damass := ch;
        end
        else if eoln(input) then exit(damagassd)
        else if ch in ['9','q'] then
        begin
          quit := true;
          exit(damagassd);
        end
        else if ch = '?' then
        begin
          lines(1);
          getfile('damass.text',;
          lines(1);
          returnbar;
      end
     else menuerror;
    until finished = true;
    end;


procedure BDAremarks;
var  x : integer;

begin
  currentgt.BDA[BDAcounter].BDAtext := nostring;
  while not finished do
  begin
    clear;
    lines(6);
    writeln('      ENTER BDA....do not exceed one line',;
    prompt;
    readln;
```

```
    if helpme then
    begin
      finished := false;
      lines(1);
      getfile('bdarem.text');
      lines(1);
      returnbar;
    end;
    if quit then exit(BDAremarks);
    if  (flg = 2) and finished then
     begin
       buffer :=                                                    ';
       cutstring(42);
       currentgt.BDA[BDAcount].BDAtext := str;
     end;
  end;
end;



    procedure BDAinfo;

    begin
      clear;
      lines(2);
      writeln('                BATTLE DAMAGE ASSESSMENT');
      lines(2);
      writeln('     For information on adding a target surveillance');
      writeln('              to the target file, type a ?.');
      lines(3);
      writeln('        **   To continue, press the RETURN key.   **');
      prompt;
      read(cn);
      if cn = '?' then
        begin
          clear;
          lines(2);
          getfile('bdainfo.text');
          lines(3);
          spacebar;
          clear;
        end;
     end;




    begin {newBDA}
      if not current then
      begin
        BDAinfo;
        retcntgt(1);
```

```
    if quit then exit(newBDA);
    i := 1;
    while (currentgt.flag[16 + i] = off) do
      begin
        i := i + 1;
        if i = 4 then
        begin
          i := 1;
          currentgt.flag[16 + i] := on;
        end;
      end;
    BDAcounter := i;
    n := 16 + i;
  end;
  endBDA := false;
  while not endBDA do
  begin
    for ii := 1 to 6 do
    begin
        finished := false;
        case ii of
          1 : BTGorBDA;
          2 : firingunit;
          3 : rounds;
          4 : damagerept;
          5 : damageassd;
          6 : begin
                BDAremarks;
                endBDA := true;
              end;
        end;
    end;
    if current then exit(newBDA)
    else begin
        with currentgt do
          begin
            if stat = '2' then stat := '1'
            else if stat = '4' then stat := '3'
            else if stat = '6' then stat := '5'
          end;
        with currentGT do
          begin
            if stat = '2' then stat := '1'
            else if stat = '4' then stat := '3'
            else if stat = '6' then stat := '5'
          end;
        putinfile;
    end;
end;
```

```
        (*        TGTPROCS.TEXT     *)

(*.....*)


    procedure mandmsg;

    begin
      lines(1);
      getfile('mandmsg.text');
      lines(1);
      spacebar;
    end;


    procedure readin;

    var  len : integer;

    begin
    helpme := false;
    readln(str);
    len := length(str);
    if len = 0 then flg := 0
    else if str[1] = '?' then flg := 1                                  {help}
    else if (len = 1) and (str[1] in ['Q','q']) then flg := 0 {quit}
    else flg := 2;                                                      {continue}
    case flg of
      0 : begin
          if mandatoryitem then mandmsg
          else finished := true;
          end;
      1 : helpme := true;
      2 : begin
            currentwt.flags[n] := off;
            finished := true;
          end;
      3 : quit := true;
        end
    end;



      procedure checknum(var strng : string ; var check : boolean);

      var x,i : integer;

      begin
        check := true;
```

```
        x := length(strng);
        if x = 6 then
          begin
            fetchback := true;
            exit(checknum);
          end;
        if x <> 6 then
        begin
          check := false;
          exit(checknum);
        end;
        for i := 1 to 2 do
        if not (strng[i] in ['A'..'Z']) then
          begin
            check := false;
            writeln('    Use upper case letters for target designator');
            exit(checknum);
          end;
        for i := 3 to 6 do
          if not (strng[i] in ['0'..'9']) then check := false;
        end;



procedure checkdigit (var strng:string; var check : boolean; rng : integer );

var  i, x : integer;

begin
  check := true;
  x := length(strng);
  if x = 0 then
            begin
              fetchback := true;
              exit(checkdigit);
            end;
  if x <> rng then
  begin
    check := false;
    exit(checkdigit);
  end;
  for i := 1 to rng do
    if not (strng[i] in ['0'..'9']) then check := false;
end;



procedure cutstring ;

(*    (strngsize : integer) removed for fwd dec    *)
```

```
var    cutter : integer;

begin
  if length(str) > strmsize then
  begin
    for cutter := 1 to strmsize do
      buffer[cutter] := str[cutter];
    str := buffer;
  end;
end;
```

```
   procedure checkDTG ;

   (*      (var strng : string; var check : boolean) removed for two lec *

   var   i : integer;

      begin

     check := true;
     if length(str) = 7 then exit(checkDTG);
     if length(strng) <> 7 then
     begin
       check := false;
       exit(checkDTG);
     end;
     for i := 1 to 5 do
       if not(strng[i] in ['0'..'9']) then
       begin
         check := false;
         exit(checkDTG);
       end;
     if not (strng[7] in ['A'..'Z']) and not (strng[7] in ['a'..'z']) then
       check := false;
   end;

  (*.....*)
```

```
```

```
```

```
      begin
        finished := false;
        close(1);
        settext(tattrec.text);
        close(1);
        returnstar;
      end;
      if (file = 2) and finished then
      begin
        buffer := '                                        ';
        substring(s,1);
        currentwl.class := st;
        buffer := '                                        ';
        substring(s,1);
        currentwl.class := st;
      end;
    end;
  end;
end;
```

```
procedure tattclass;

begin
  repeat
  clear;
  close(2);
  settext('classmenu.text');
  close(1);
  select;
  readln(ch);
  if ch in ['1'..'3'] then
  begin
    if eoln(input) then readln;
      currentwl.class[1] := ch;
      finished := true;
    case ch of
      '1' : begin
              currentwl.class := 'A';
              currentwl.class := '1';
            end;
      '2' : begin
              currentwl.class := 'B';
              currentwl.class := '2';
            end;
      '3' : begin
              currentwl.class := 'C';
              currentwl.class := 'C';
```

```
        end;
    'a' : begin
            current.class := 'a';
            current.class := 'a';
          end;
    'b' : begin
            current.class := 'b';
            current.class := 'b';
          end;
    'c','f','i' : begin
                    current.lines[n] := on;
                    finished := true;
                  end;
    'h' : begin
            clear;
            lines(1);
            writeln('lines.text');
            lines(1);
            spec(spec);
            current.lines[n] := on;
            finished := true;
          end
      end
  end
  else if on in ['e','s'] then
          begin
            quit := true;
            exit(tgtclass);
          end
  else if (eoln(input)) then readln
  else menuerror;
  until finished = true;
end;



procedure tgtpri;

begin
  repeat
  clear;
  lines(4);
  settitle('tgtprimenu.text');
  lines(1);
  select;
  read(on);
  if on in tenucar then
  begin
    if eoln (input) then readln;
    if on in ['1'..'4'] then
    begin
      current.lines[n] := off;
```

```
        finished := true;
        currentgt.pri := cn;
        currentT.pri := cn;
      end;
    if cn = '?' then
       begin
         clear;
         lines(1);
         getfile('priority.text');
         lines(3);
         spacebar;
       end;
    if cn in ['b','e','R','r'] then menuerror;
    end
    else if cn in ['q','q'] then
           begin
             quit := true;
             exit(tstpri);
           end
    else if (eoln(input)) then readln
    else menuerror;
    until finished = true;
  end;


    procedure tststatus;

    var loop : boolean;
    code : integer;


    procedure active;

    begin
      loop := false;
      repeat
        clear;
        lines(5);
        writeln('      ENTER TARGET STATUS--ACTIVITY');
        lines(1);
        writeln('       The options are:');
        lines(1);
        writeln('          1.  Active');
        writeln('          2.  Inactive');
        lines(1);
        writeln('      PLEASE ENTER OPTION NUMBER AND PRESS RETURN');
        prompt;
        read(cn);
        if cn in menuset then
        begin
          if not eoln(input) then readln;
          if cn = '1' then
          begin
            code := 0;
```

```
            ...  := true;
          end;
          if ch = 'Z' then
          begin
            code := 12;
            loop := true;
          end;
          if ch = '?' then
          begin
            lines(1);
writeln('     ** Define target as the whole is done in the input list **');
writeln('The list of targets, as illustrated, target is in the  sample.');
            lines(1);
            spaces(1);
          end;
          if ch in ['S','s','D','d','F','f'] then
          begin
          else if ch in ['e','E'] then
          begin
            quit := true;
            exit(getlist);
          end
          else if (ord(ch)=27) then
          else
        until loop = true;
    end;

    procedure listit;

    begin
      ...p := false;
      repeat
        lines(1);
        writeln('      ... 3.1415926--- ...');
        lines(1);
        writeln('       The options are');
        lines(1);
        writeln('          1.  Input list');
        writeln('          2.  List of targets');
        lines(1);
        write('Enter choice');
        read(ch);
        if ch in ['1','2'] then
        begin
          if eoln(input) then readln;
          if ch = '1' then
          begin
            code := ord(ch) - 48;
            loop := true;
          end;
          if ch = '2' then
          begin
            code := ord(ch) + 0;
```

```
            loop := true;
          end;
          if ch = '?' then
          begin
            lines(2);
writeln('The target list refers to the current list of the target');
writeln('combatants. If a target is not on the current list then it');
writeln('              is on the list at target.');
            lines(2);
            spacebar;
          end;
          if ch in [' ','-','?',',','.',';',':'] then readln
          else if ch in ['e','E'] then
               begin
                 quit := true;
                 exit(attack);
               end
    else if (eoln(input)) then readln
    else readerror;
  until loop = true;
  end;

  procedure attacker;

  begin
  loop := false;
  repeat
      clear;
      lines(3);
      writeln('      ENTER TARGET STATUS-DESIGNATION FOR ATTACK ');
      lines(1);
      writeln('       AND OPTIONS AS:');
      lines(2);
      writeln('          1. Attack');
      writeln('          2. Not attack');
      lines(1);
      select;
      read(ch);
      if ch in ['1','2'] then
      begin
        if eoln(input) then readln;
        if ch = '1' then
        begin
          code := code + 1;
          loop := true;
        end;
        if ch = '2' then
        begin
          code := code + 2;
          loop := true;
        end;
        if ch = '.' then
        begin
```

```
                    ...;
writeln('Attached terminals are those obtained by appropriate ...');
writeln('for which there is a confirmation of their acquisition.');
                    ...;
                    ...;
                  ...;
                  if ch in ['A','a','O','o',' ','c','C'] then ...
                  ...
                  else if ch in ['q','Q'] then
                      begin
                        quit := true;
                        exit(attached);
                      end
                  else if (ch in [...,ut]) then ...
                  else ...error;
              until ... = true;
          end;


          begin        {main}
            repeat
            ...;
            active;
            if quit then exit(testatus);
            listen;
            if quit then exit(testatus);
            attached;
            if quit then exit(testatus);
            finished := false;
            case code of
              1 : ch := '1';
              2 : ch := '2';
              3 : ch := '3';
              4 : ch := '4';
        5, 6, 7 : begin
                    ...(2);
                    writeln('     Connection not possible....must restart ...');
                    ...;
                    finished := false;
                  end;
              8 : ch := 'b';
              9 : ch := 'b';
            end;
            until finished;
            if ch in ['1','3','5'] then ...
            current[i].fmt[5] := ch;
            current[i].stat := '1';
            current[i].stat := ch;
          end;
```

```
procedure initType;
var   number : integer;

begin
   choices := ['1','2','3','4','5','6','7','8','9'];
   repeat
      clear;
      lines(2);
      writeln('Type of terminal.text');
      lines(1);
      select;
      read(c);
      if c in choices then
      begin
        if c = '1' then terminal;
        finished := true;
      end
      else if c in ['q','Q'] then
      begin
        quit := true;
        exit(initType);
      end
      else if (c in input) then
      begin
        clear;
        lines(3);
        terminal;
      end
      else if c = '?' then
      begin
        lines(1);
        writeln('initType.text');
        lines(1);
        returnter;
      end
      else returnter;
    until finished;
    currentpt.ttype := cu;
    currentat.line[n] := cr;
    currentST.ttype := cu;
end;



procedure setline;

      var x, y : integer;


begin
   currentat.alt := hostline;
   currentST.alt :  hostline;
   while not finished do
   begin
```

```
      clrscr;
      writeln;
      writeln('      Enter the full filename      ');
      writeln;
      readln;
      if quit then exit(bigbit);
      str := inputstr(str);
      for i := 1 to length(str)
        if not (str[i] in ['0'..'9']) then i := length(str);
      if length(str) = 0 then str := '0';
      if newjobs then
      begin
        filename := 'jobs';
        lines(1);
        assign(textfile,'textfile.text');
        lines(1);
        reset(textfile);
      end;
      if (ch = 'A') and filename[1] = 'j'
      then
        rewrite(textfile,1,cnt := str);
        rewrite(textfile,1,cnt := str);
      end;
    end;
  end;
end;



procedure CheckSet;

var   number : integer;

begin
  setnum := ['1','2','3','4','5','6','7','8','9'];
    repeat
      clrscr;
      lines(2);
      assign(f,'setup.text');
      lines(1);
      select;
      read(ch);
      if ch in setnum then
      begin
        if ord(ch)-ord('0') > maxsets then
        writeln('Only ',maxsets,' sets allowed');
      end
      else if ch in ['e','E'] then
      begin
        quit := true;
        exit(checkset);
      end
```

```
      else if ch = '.' then
        begin
          clear;
          lines(5);
          getfile('sa.text');
          lines(1);
          returnter;
        end
        else if eoln(input) then
        begin
          currentgt.sa := '9';
          currentGT.sa := '9';
          exit(SAassem);
        end
        else menuerror;
      until finished;
      currentgt.sa := ch;
      currentgt.flag[n] := on;
      currentGT.sa := ch;
end;


procedure remarks;

begin
  currentgt.rem := nostring;
  while not finished do
  begin
    clear;
    lines(5);
    writeln('      ENTER REMARKS CONCERNING TARGET....DO NOT EXCEED ONE LINE');
    prompt;
    readln;
    if helpme then
    begin
      finished := false;
      lines(1);
      getfile('remarks.text');
      lines(1);
      returnter;
    end;
    if quit then exit(remarks);
    if (ch = 2) and finished then
    begin
      buffer := '                                                    ';
      cutstring(40);
      currentgt.rem := str;
    end;
  end;
end;
```

```
procedure mapref;

begin
  currentgt.mapprefer := nostring;
  while not finished do
  begin
    clear;
    lines(5);
    writeln('      ENTER TARGET MAP REFERENCE....do not exceed 20 characters.');
    prompt;
    readln;
    if helpme then
    begin
      finished := false;
      lines(1);
      getfile('mapref.text');
      lines(1);
      returnter;
    end;
    if quit then exit(mapref);
    if (file = 2) and finished then
    begin
      buffer := '                    ';
      substring(20);
      currentgt.mapprefer := str;
    end;
  end;
end;



procedure source;

begin
  currentgt.sour := nostring;
  while not finished do
  begin
    clear;
    lines(5);
    writeln('      ENTER SOURCE OF TARGET....do not exceed 20 characters.');
    prompt;
    readln;
    if helpme then
    begin
      finished := false;
      lines(1);
      getfile('sour.text');
      lines(1);
      returnter;
    end;
    if quit then exit(source);
    if (file = 2) and finished then
    begin
      buffer := '                    ';
```

```
        outstring(2,);
        currentgt.sour := str;
      end;
    end;
  end;



procedure afotonum;

begin
  currentgt.photonum := nostring;
  while not finished do
  begin
    clear;
    lines(6);
    writeln('      ENTER AERIAL PHOTO NUMBER');
    prompt;
    readin;
    if helpme then
    begin
      finished := false;
      lines(1);
      getfile('afotonum.text');
      lines(1);
      returnter;
    end;
    if quit then exit(afotonum);
    if (fle = 2) and finished then
    begin
      buffer := '                ';
      outstring(15);
      currentgt.photonum := str;
    end;
  end;
end;

        procedure photogrid;

        begin
 range := 5;
 while not finished do
 begin
   clear;
   lines(6);
   getfile('photogrid1.text');
   prompt;
   readin;
   if fle = 0 then
   begin
     currentgt.photocord := nostring;
     exit(photogrid);
   end;
   if (fle = 2) and (length(str) = 1) and (str[1] in ['S','s'] then
```

```
      begin
        currentst.photocord := currentst.loc;
        exit(photoentr);
      end;
    checkvalit (str,ok,range);
    if quit then exit (photoentr);
    if helpme or not ok then
    begin
      finished := false;
      lines(1);
      getfile('photoentr2.text');
      lines(1);
      returnbar;
    end;
    if (ptr = 2) and finished then currentst.photocord := str;
  end;
end;


          procedure LTGactive;

begin
  currentst.LTGact := hostname;
  while not finished do
    begin
      clear;
      lines(6);
      writeln('      ENTER LTG IDENT WAS ACTIVATED...6 digits and 1 letter.');
      prompt;
      readin;
      check TG(str,ok);
      if quit then exit(LTGactive);
      if helpme or not ok then
        begin
          finished := false;
          lines(1);
          getfile('ltgact.text');
          lines(1);
          returnbar;
        end;
        if (ptr = 2) and finished then currentst.LTGact := str;
      end;
    end;



  procedure tgtaccuracy;

  begin
    currentst.acc := '4';
    currentCT.acc := '4';
    repeat
    clear;
```

```
    lines(4);
    getfile('tetaccmenu.text');
    lines(1);
    select;
    read(ch);
    if ch in menuchar then
    begin
      if eoln (input) then readln;
      if ch in ['1'..'4'] then finished := true;
      if ch in ['b','B','R','r'] then menuerror;
      if ch = '?' then
      begin
        lines(1);
        getfile('tetacc.text');
        lines(1);
        returnmenu;
      end;
     end
     else if ch in ['q','Q'] then
          begin
            quit := true;
            exit(tetaccuracy);
          end
    else if eoln(input) then exit(tetaccuracy)
    else menuerror;
    until finished = true;
        currentst.flag[n] := off;
        currentst.acc := ch;
        currentST.acc := ch;
end;
```

```
                (*      CHANGE PROCEDURE        *)

    procedure caseproc; forward;
    procedure displaytet; forward;


  procedure change;

  var  punchout, yes : boolean;


  procedure changeinfo;

  begin
    clear;
    lines(2);
    getfile('changeinfo.text');
    lines(2);
    spacebar;
    clear;
  end;


                procedure yescheck;

    begin
      finished := false;
      yes := false;
      lines(2);
      writeln('      Change Y  Y(es)  N(o)');
      prompt;
      read(ch);
      if (ch = 'Y') or (ch = 'y') then yes := true
      else if ( ch = 'Q') or ( ch = 'q') then punchout := true;
    end;



    procedure chaproc2;

begin
  with currentet do
    begin
      clear;
      lines(5);
      writeln('  Current air photo grid is...',photogrid);
      yescheck;
      if yes then photogrid;
      if punchout then exit(ch.proc2);
      clear;
      lines(5);
```

```
      writeln('  Current ITG target activated is...',active);
      yescheck;
      if yes then ITGactive;
      if punchout then exit(changeproc2);
      clear;
      lines(5);
      writeln('  Current accuracy is...',accuracy);
      yescheck;
      if yes then tgtaccuracy;
      if punchout then exit(changeproc2);
    end;
end;


procedure changeproc;

begin
  quit := false;
  caseproc;
    with currentgt do
      begin
        clear;
        lines(2);
        writeln('        Target Number  ',tnum);
        lines(5);
        writeln('  Current location is...',loc);
        yescheck;
        if yes then tgtloc;
        if punchout then exit(changeproc);
        clear;
        lines(4);
        writeln('  Current description is...',desc);
        yescheck;
        if yes then tgtdesc;
        if punchout then exit(changeproc);
        clear;
        lines(3);
        writeln('  Current class is...',class );
        yescheck;
        if yes then tgtclass;
        if punchout then exit(changeproc);
        clear;
        lines(3);
        writeln('  Current priority is...',pritgt);
        yescheck;
        if yes then tgtpri;
        if punchout then exit(changeproc);
        clear;
        lines(3);
        writeln('  Current status is...',status);
        writeln('     On target list?..',stat2);
        yescheck;
        if yes then tgtstatus;
        if punchout then exit(changeproc);
```

```
        clear;
        lines(5);
        writeln('  Current type is...',ttypeout);
        yescheck;
        if yes then tettype;
        if punchout then exit(changeproc);
        clear;
        lines(5);
        writeln('  Current altitude is...',alt);
        yescheck;
        if yes then tgtalt;
        if punchout then exit(changeproc);
        clear;
        lines(5);
        writeln('  Current supporting arm assigned is...',satel);
        yescheck;
        if yes then SAassign;
        if punchout then exit(changeproc);
        clear;
        lines(5);
        writeln('  Current remarks are...',rem);
        yescheck;
        if yes then remarks;
        if punchout then exit(changeproc);
        clear;
        lines(5);
        writeln('  Current map reference is...',mapreter);
        yescheck;
        if yes then mapref;
        if punchout then exit(changeproc);
        clear;
        lines(5);
        writeln('  Current source is...',sou);
        yescheck;
        if yes then source;
        if punchout then exit(changeproc);
        clear;
        lines(5);
        writeln('  Current air photo number is...',photonum);
        yescheck;
        if yes then arotonum;
        if punchout then exit(changeproc);
        chgproc2;
    end;
  end;



begin{change}
  mandatoryitem := false;
  changeinfo;
  if not current then retcntgt(1);
```

```
if quit then exit(changes);
repeat
  punchout := false;
  clear;
  lines(2);
  writeln('      Target  ',currentgt.tnum,' is loaded into memory');
  lines(2);
  settile('changemenu.text');
  lines(2);
  select;
  read(ch);
  if ch in ['1','2','3','4','R','r','Y'] then
  begin
    if eoln(input) then readln;
    case ch of
     '1' : displaytgt;
     '2' : changeproc;
     '3' : begin
             putinfile;
             outprocess := true;
             exit(change);
           end;
     '4','R','r' : exit(change);
     'Y' : begin
             lines(2);
             settile('changex.text');
             lines(2);
             spacebar;
           end;
    end;
  end;
 until menuloop = true;
end;
```

```
{*************************************************************}

                    {*    TARGET.TEXT    *}

  procedure bullseye;

  var  j : integer;

  begin
    j := x;
    i := y;
    while not eof(target) do
    begin
        seek(target,i);
        get(target);
        writeln(' ',target^.data,'  ');
        i := i - 1;
        j := j + 1;
        if j = 11 then
          begin
            write('*');
            j := 1;
          end;
      end;
  end;


  procedure initbuf;

  var  holder, i : integer;
       first : boolean;

begin
  first := true;
  x := 0;
  range := 5;
  fetchback := false;
  finished := false;
  quit := false;
  mandatoryitem := false;
  n := 1;
  while not finished do
  begin
    lines(5);
    if grid = 2 then writeln('      ENTER GRID LOCATION')
    else writeln('      ENTER TARGET NUMBER');
    prompt;
    readln;
    if grid = 2 then checkgrid (str,ok,range)
    else checknum(str,ok);
    if quit then exit(initbuf);
    if fetchback then goto 1;
    if return or not ok then
```

```
    begin
      finished := false;
      lines(2);
      if grid = 2 then
      begin
      writeln('letter.text');
      end
      else
      begin
      writeln('num.text');
      end;
      lines(2);
      returnbar;
    end;
    if (grid = 2) and finished then
    begin
      clear;
      lines(5);
      if grid = 2 then
      begin
        write('  Searching for grid coordinates ',str);
        quitearlier;
      end
      else write('  Searching for target ',str);
      recnum := 1;
      write(dot);
      if grid = 2 then
        begin
          for recnum := 2 to numbertot - 1 do
          begin
            if writeln[recnum] = str then
            begin
              lines(1);
write('          Target no. ',trunc(recnum),' has coordinates ');
              if first then holder := recnum;
              first := false;
              r := r + 1;
            end;
          end;
          if r = 1 then recnum := normal;
          if r > 1 then
          begin
            finished := true;
            lines(1);
          writeln('     Select the desired target number from the above ');
            lines(1);
            spacebar;
            exit(procedure);
          end
      end
      else write (writeln[recnum] <> str) and not finished then
          recnum := recnum + 1;
       write(dot);
       if recnum = numbertot - 1 then
```

```
        begin
          found := false;
          errors;
        end
        else
        begin
          write(scr);
          seek(target,recnum);
          get(target);
          write(scr);
          current.t := target^.textnum;
          seek(ci,recnum);
          get(ci);
          current.i := ci^.indexnum;
          write(scr);
        end;
      end;
    end;
  end;
```

```
    procedure del;

    begin
      ci^.qwerty := emptystr;
      seek(ci,recnum);
      put(ci);
      target^.textnum := emptystr;
      seek(target,recnum);
      put(target);
      textmap[recnum] := '      ';
    end;


    procedure deletetext;

    begin
      clear;
      lines(2);
      writeln('                    DELETE TEXT');
      lines(2);
      assign('deltext.text');
      retention(1);
      if not quit then del;
    end;




    procedure edit;

    var  holder : char;
```

```
(****)
  with curtmenu1 do
  begin
    case position of
      '1' : status := 'file                ';
      '2' : status := 'var                 ';
      '3' : status := 'bin                 ';
      '4' : status := 'man, mnt            ';
      '5' : status := 'usr, var            ';
      '6' : status := 'mnt, var            ';
      '7' : status := 'usr, mnt, var       ';
      '8' : status := 'other               ';
      '9' : status := 'NONE                ';
    end;
    position := ord;
    case position of
      '1' : boolout := 'CONTIGUOUS';
      '2' : boolout := 'FRAGMENTED';
      '3' : boolout := 'POSSIBLE  ';
      '4' : boolout := 'UNKNOWN   ';
    end;
    position := pid;
    case position of
      '1' : pfilout := '1   ';
      '2' : pfilout := '11  ';
      '3' : pfilout := '111 ';
      '4' : pfilout := '1+  ';
    end;
    position := ttynum;
    case position of
      '1' : ttyp_out := 'ttyp1';
      '2' : ttyp_out := 'ttyp2';
      '3' : ttyp_out := 'ttyp3';
      '4' : ttyp_out := 'ttyp4';
      '5' : ttyp_out := 'ttyp5';
      '6' : ttyp_out := 'ttyp6';
      '7' : ttyp_out := 'ttyp7';
      '8' : ttyp_out := 'ttyp8';
      '9' : ttyp_out := 'ttyp9';
    end;
    position := stat;
    case position of
      '1','2' : begin
                  status := 'active  ';
                  stat2  := 'run     ';
                end;
      'A','a' : begin
                  status := 'Active  ';
                  stat2  := 'wait    ';
                end;
      'S','s' : begin
                  status := 'Inactive';
```

```
                                    stat' :  ') ';
                                  end;
                              end;
                            end;
                          end;


      procedure relocate;


      begin
        clear;
        with currentwt do
          begin
writeln(dots);
writeln('                    : ',wheel,'.',trk,'.',');
writeln('                      ----------.----------');
writeln('Location: ',loc,'      Alt: ',alt,'          Type: ',ttype );
writeln('Description: ',desc);
writeln('Class    : ',class,'                 Status: ',stat);
writeln('Priority : ',prior,'              Int Instr: ',intr,
writeln('SA Assng : ',satur);
writeln('Source of Int: ',sour);
writeln('ITA Activated: ',ita,
writeln('Photo No: ',photono,'       Tape No: ',tapeno);
writeln('Photo Coord: ',photocrd,'   Altitude: ',photalt);
writeln('Remarks: ',rem);
          end;
      end;



procedure AllAcft;

var holder : char;
    tempstr : string[11];
    tempss : string[11];

      begin
        tempss := '           ';
        tempstr := '           ';
writeln('---------------------Unavailable----------------------------');
writeln('            Flight     Volts          Number       Assigned');
writeln('    ID      Unit       Ratio          Operator     Mission ');
writeln('    ---     ------     --------       ----------   --------',
        for i := 1 to 5 do
          if (currentwt.ID[i + 1] = chr(i + 48)) and (currentwt.ID[i]
            begin
              holder := tempstr;
              case holder of
              '1' : tempstr := 'Darwin    ';
              '2' : tempstr := 'Bell      ';
```

```
                'c' : category := [illegible];
                'd' : category := [illegible];
                'e' : category := [illegible];
                'f' : category := [illegible];
                'g' : category := [illegible];
                'h' : category := [illegible];
                'i' : category := [illegible];
                'j' : category := [illegible];
              end;
              writeln(category);
              case colour of
                'b' : category := [illegible];
                'c' : category := [illegible];
                'd' : category := [illegible];
                'e' : category := [illegible];
                'f' : category := [illegible];
                'g' : category := [illegible];
                'h' : category := [illegible];
                'i' : category := [illegible];
                'j' : category := [illegible];
              end;
writeln('  ',colour:2,'    ',[illegible],'    ',[illegible]);
writeln('                ',[illegible],'    ',category);
writeln('Size: ',[illegible]);
            end;
        end;


        procedure [illegible];

          var
            with [illegible] do
              begin
                [illegible];
                [illegible];
                [illegible] := [illegible];
              end;
            writeln('..........',[illegible],'     ..........');
            repeat
              [illegible];
            until eoln([illegible]);
          end;


        procedure [illegible];

        var
          current : [illegible];
          finished : [illegible];
          dummy : [illegible];
          repeat
            [illegible];
            [illegible];
            writeln(                    [illegible]);
```

```
      writeln(f);
      writeln(f,'                    ');
      writeln(f);
      writeln(f,'                    ');
      writeln(f,'                    ');
      writeln(f,'                    ');
      writeln(f,'                    ');
      writeln(f);
      close(f);
      if ch in ['1','2','3','4','5','6','7','8','9','0','-','=',
      repeat
        if eoln(input) then readln;
        case ch of
          '1' : return(1);
          '2' : return(2);
          '3','4' : begin
                      writeln;
                      write('Sub-Menu Not Available ');
                      writeln;
                      ch:='0';
                    end;
          'a','c','d','e','f','g' : write('Not Displayed ');
        end;
      end;
      write('Enter Choice: ');
    until ch in ['1','2','3','4','5','6','7','8','9','0','-','=',
    if ch = '0' then quit := true;
  until quit;
end;




procedure putinfile;


begin
  seek(target,recnum);
  get(target);
  target^.target := oldcontent;
  seek(target,recnum);
  put(target);
  telephone[index] := target^.target.name;
  seek(f ,recnum);
  put(f);
  ch := oldcontent;
  seek(f,recnum);
  put(f);
  currentnet := employee;
  currentno := employee;
end;
```

```
procedure Access;

type
  [illegible] is (a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z);
  [illegible]
    [illegible];
    write(title);
    lines(2);
    write('      [illegible]');
    lines(2);
    write('      [illegible]');
    lines(2);
    write('            1. [illegible]');
    write('            2. [illegible]');
    write('            3. [illegible]');
    write('            4. [illegible]');
    if [illegible] then write('[illegible]');
    write('            5. [illegible]');
    lines(1);
    Status(f);
    [illegible];
    if c in ['1'..'5'] then
      begin
        if eof(input) then halt(1);
        readln(c);
        case c of
          '1' : [illegible];
          '2' : begin
                  [illegible] := true;
                  [illegible];
                end;
          '3' : begin
                  [illegible];
                  change;
                  if [illegible] then [illegible];
                end;
          '4' : halt;
          '5' : if [illegible].stat = '2' then [illegible]
                  begin
                    [illegible] := 1;
                    [illegible];
                    [illegible];
                    [illegible] := false;
                  end
                else [illegible];
        end;
      end
    else if c = '?' then
      begin
        [illegible] := false;
```

```
          clrscr;
          lines(2);
          [illegible]('program.text');
          lines(1);
          space[illegible];
        end
    else [illegible];
  until finish [illegible];
end;


     procedure ch[illegible];

     var  counter : integer;

     begin
       count := 0;
       for [illegible] := 1 to [illegible] do
         if [illegible]
       if count = 0 then
       begin
         clrscr;
         lines(2);
         writeln('  [illegible]');
         lines(2);
         [illegible];
         done := true;
       end
       else begin
              [illegible];
              lines([illegible]);
will [illegible]
              [illegible]
              lines(2);
              writeln('      The options are :');
              lines(1);
              writeln('        1.  Continue working on [illegible]');
              writeln('        2.  Stop working on [illegible]');
              writeln('        3.  [illegible] changes to [illegible]');
              lines(1);
              write[illegible];
              read(ch);
              if ch in ['1','2','3','4'] then
                begin
                  if eoln(input) then readln;
                  case ch of
                    '1' : exit([illegible]);
                    '2' : begin
                            done := true;
                            exit([illegible]);
                          end;
                    '3' : begin
                            done := true;
```

```
                                    current := true;
                                    menu;
                                    edit(current, 'n');
                                    end;
                        'r' : begin
                                clear;
                                lines(5);
                                writeln('addtelinfo.text');
                                lines(2);
                                returnhere;
                              end
                      end
                    else menuerror;
                    clear;
                  until menucop = true;
                end;
            end;


        procedure addtelinfo;

begin
  clear;
  writeln(dots);
  lines(2);
  getfile('addtelinfo.text');
  lines(2);
  spacebar;
  clear;
  lines(2);
  getfile('addtirfl.text');
  lines(2);
end;



        procedure newtelrec;

        var j, i : integer;

        begin
          recnum := 4;
          while telmap[recnum] <> 'nonrec' do
          begin
            recnum := recnum + 1;
            if recnum = numtelrec + 1 then
            begin
              clear;
              lines(11);
              writeln('       File FULL',;
              lines(2);
              writeln('       Records must be deleted in order to continue' ;
```

```
            lines(~);
            spacetar;
            gotz := true;
            exit(nextlevel);
          end;
        end;
        seek(ct,recnum);
        get(ct);
        current1 := ct^.querref;
        seek(target,recnum);
        get(target);
        current := target^.typing;
        for i := 1 to maxcharset do
          current.ilen[i] := 0;
      end;



      procedure writetarget;

      var  i : integer;

begin
  currentrec := 1;
  readonlyflag := true;
  updated := false;
  subtotal := 0;
  titleprot := 'ratio';
  done := false;
  quit := false;
  ok := true;
  clear;
  if first then
    begin
      writeln(curse);
      lines(1);
      getfile('printer.text',
      prompt;
      first := false;
    end
  else
    begin
      lines(5);
writeln('  *** Top Secret Matter, type a 4-digit combination, press return ***');
      prompt;
    end;
  repeat
  read(kb,ch);
  if ch = ' ' then
      begin
        addtoplan1;
        putback1;
```

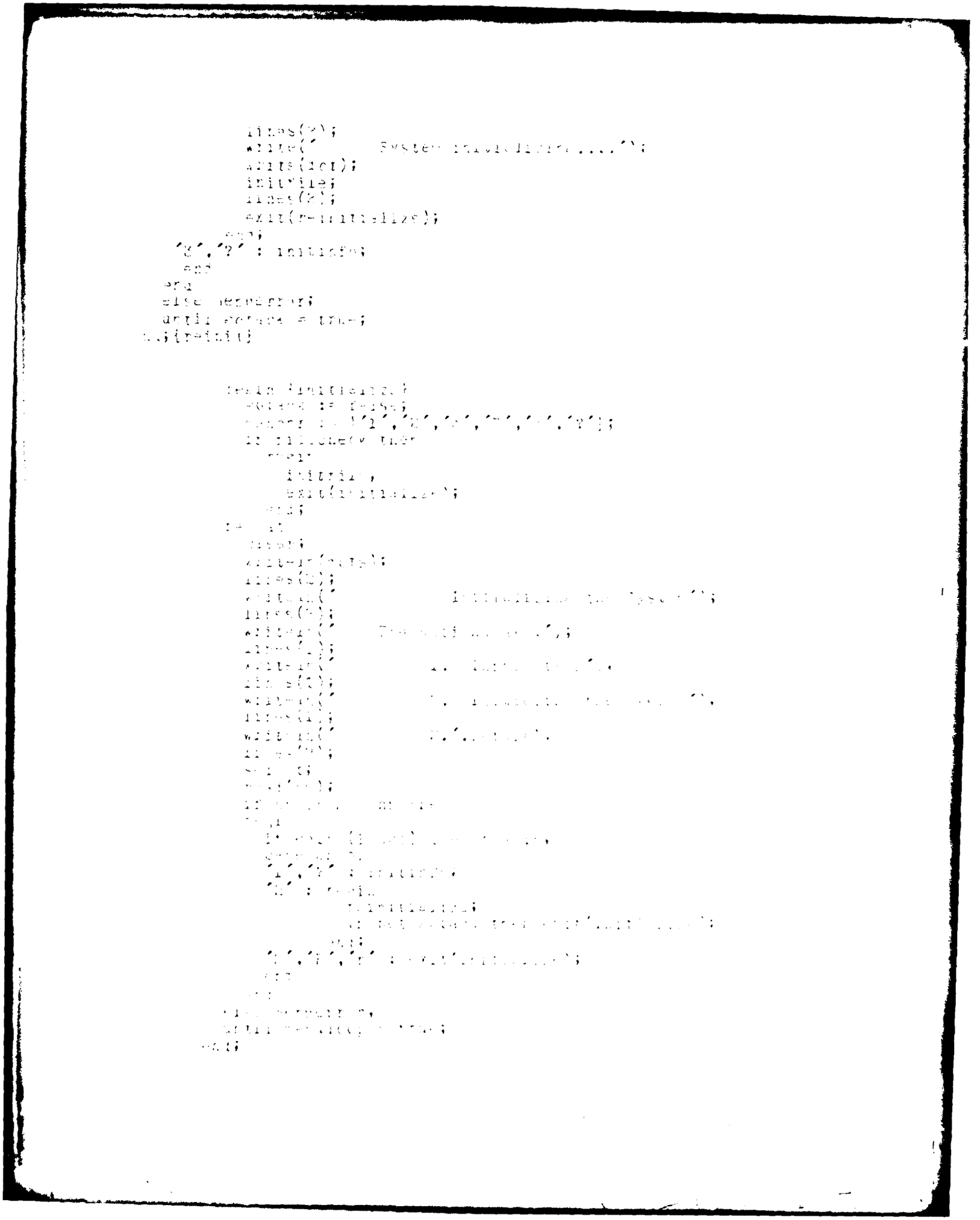
```
        ...;
  until eoln(input);
  ...writeln;
  if ... then ...(...);
  while not ... do
  begin
    for ... := 1 to ... - 2 do
      begin
        if ... then ...(...);
        if ...[...] = ... then
        begin
          case ... of
            ... : ...;
            ... : ...;
            ... : ...;
            ... : ...;
            ... : ...;
            ... : ...;
            ... : ...;
            ... : begin
                    ... := false;
                    ...;
                  end;
          end; {case}
        end;
        ... := false;
      end; {for}
      if ... <> 0 then ...
      else done := true;
    end; {while}
    process;
  end;



      procedure ...;

      begin
        clear;
        writeln(stars);
        lines(2);
        writeln('          ... on the ... file ');
        lines(1);
        writeln('      ... options are:');
        lines(1);
        writeln('            1.  Add a ...');
```

```
        writeln('              2.  change a target');
        writeln('              3.  display a target');
        writeln('              4.  enter a target surveillance');
        writeln('              5.  delete a target');
        writeln('              6.  list targets currently in file');
        writeln('              7.  return');
        lines(2);
      end;


procedure listscreen;

var   j : integer;

    begin
      j := 4;
      clear;
      lines(2);
      writeln('                 list of current targets in file');
      lines(2);
      for i := 1 to numtargets do
        if targrec[i] <> 'zeroed' then
          begin
            write('         ',targrec[i]);
            j := j - 1;
            if j = 0 then
              begin
                writeln;
                j := 4;
              end;
          end;
      lines(2);
      writeln('                    End of target list');
      lines(1);
      spacebar;
    end;



begin {targetmgt};
  hostrine := '      ';
  hostnum := ' ';
  first := true;
  reenteroption := false;
  returnop := false;
  menuset := ['1','2','3','4','5','6','7','8','9','A','B','C','D','E','F','G'];
      repeat
       currmt := false;
       targtemu;
       select;
       read(ch);
       if ch in menuset then
       begin
```

```
    if eof(input) then readln;
    case ch of
      'a','A','+' : addtext;
      'c','C','-' : change;
      'd','D',' ' : display;
      'e' : begin
              canactivate := false;
              rewrdn;
            end;
      'f' : deletetext;
      'g' : listchange;
      'q','Q','x' : exit(tarmethod);
      '?' : begin
              clear;
              lines(5);
              settitle('tarmdx.text');
              lines(5);
              spacebar;
            end
    end
  else beep(1);
until (ch = 'q') or (ch = 'x');
end;
```

```
                (*     INITIALIZE    *)

segment procedure initialize;

    var
          menchr : mvalue;
          goback : boolean;

     procedure initinfo;

     begin
clear;
writeln(stars);
lines(1);
writeln('   If the system has not been initialized when the program is');
writeln('started, it will initialize automatically and create the target');
writeln('files on the diskette. This procedure allows you to re-initialize ');
writeln('the system for a new operation or to restart target information');
writeln('operations from a fresh start.');
lines(2);
writeln('   Option 2 will delete all the current files and allow you to');
writeln('start out from the beginning. The files you delete are not');
writeln('recoverable. Be sure you want to delete all of your files before');
writeln('you select option 2. Use option 3 to return to the main');
writeln('command menu. Pressing the return key will return you to the');
writeln('initialize option menu.');
lines(2);
returnbar;
end;


procedure initfile;

     var  i, j : integer;

begin {initfile}
  restart := true;
  write(dot);
  j := 1;
  write(dot);
  close(target,purge);
  rewrite(target,'#5:targetfile.data');
  for i := 1 to numbertgt do
    begin
      j := j + 1;
      if j = 15 then
      begin
        write(dot);
        j := 1;
      end;
      target^.tgtrec := emptyfrec;
      put(target);
    end;
```

```
    close(target,lock);
    reset(target,'#5:targetfile.data');
    write(dot);
    close(QT,purge);
    rewrite(QT,'#5:queryfile.data');
    write(dot);
    for i := 1 to numbertgt do
      begin
        j := j + 1;
        if j = 15 then
        begin
          write(dot);
          j := 1;
        end;
        QT^.querrec := emptyQrec;
        put(QT);
      end;
    close(QT,lock);
    reset(QT,'#5:queryfile.data');
    write(dot);
end;


  procedure reinitialize;

  begin {reinit}
    goback := false;
    clear;
    lines(4);
writeln('       THIS PROCEDURE WILL DELETE ALL TARGET FILES.');
writeln(dots);
lines(1);
writeln(chr(7));
writeln('          The options are:');
lines(1);
writeln('               1.  Return to Main Command Menu');
lines(1);
writeln('               2.  Reinitialize System');
lines(1);
writeln('               3.  Information');
lines(1);
    select;
    repeat
    read(c);
    if eoln(input) then
    begin
      goback := true;
      exit(reinitialize);
    end;
    if c in ['1'..'3'] then
    begin
      case c of
        '1' : exit(reinitialize);
        '2' : begin
```



```
segment procedure inform;


var
inchar : char;


  procedure informmenu;

  begin
  clear;
  writeln(dots);
  lines(2);
  writeln('  This section provides information about the following:');
  lines(2);
  writeln('          1.  How to operate the system');
  writeln('          2.  Security requirements');
  writeln('          3.  Largest Classifications');
  writeln('          4.  Largest priorities');
  writeln('          5.  Largest Analysis privileges');
  writeln('          6.',return);
  lines(2);
  end;


  procedure userinst;

  begin
    clear;
    writeln('          System operation Instructions');
    writeln(dots);
    lines(14);
    writeln('  TO BE INSERTED');
    lines(5);
    spacebar;
  end;


  procedure format1; forward;
  procedure formatoptions; forward;

  procedure formats;

  begin
    clear;
    lines(1);
    writeln('          Formats used in the displays');
    writeln(dots);
    lines(1);
writeln('    There are two basic formats used in the Largest Information ',
```

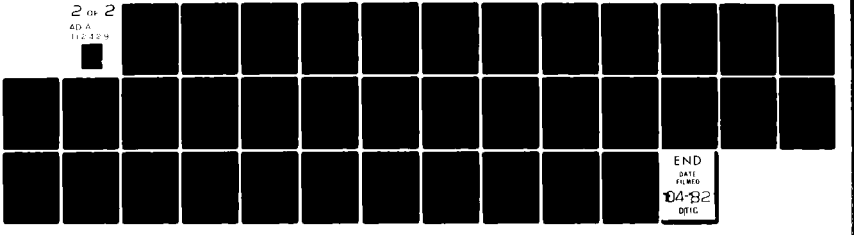
```
writeln('system. The first is a register of the current target assets');
writeln('specified by type (-1 gives current coordination . All of the ',
writeln('information about a particular target, including target');
writeln('surveillance, can be displayed on the screen or printed on the');
writeln('line printer.');
lines(4);
writeln('     The second type of display is the target listing. This');
writeln('listing contains the most important items from the target');
writeln('data set is used primarily by the surveillance aircraft of');
writeln('the fleet. This listing is available in many different types,');
writeln('from the target list to a special listing of particular target');
writeln('characteristics.');
        lines(2);
        statuter;
        formats;
      end;


      procedure formats;

      begin
      clear;
      lines(1);
      writeln('     The procedure in this system allows you to display or');
      writeln('print a list of targets with a particular set of the following ',
      writeln('parameters: status, priority, classification, specific ',
      writeln('air assigned, target type and proximity. For example, you ');
      writeln('could obtain a list of all hostile, active target assigned ',
      writeln('to naval surface ships that were within 100 miles ');
      writeln('in and all reconnaissance targets assigned to aircraft ');
      writeln('carriers either of the 1st or 2nd task force to the ');
      writeln('order.');
      lines(2);
      writeln('     A third list that is displayed or printed is the');
      writeln('target tracking report. The system automatically keeps');
      writeln('track of all transactions for the target list and allows');
      writeln('you to print a formatted report for transmission. ');
      writeln('It uses the standard format of targets added to the list, ');
      writeln('deleted from the list,targets changed and current target');
      writeln('surveillance.');
      lines(1);
      statuter;
      formatoptions;
      end;


      procedure targets; forward;
      procedure lists; forward;
      procedure lists; forward;
      procedure formatoptions; forward;



      procedure formatoptions;
```

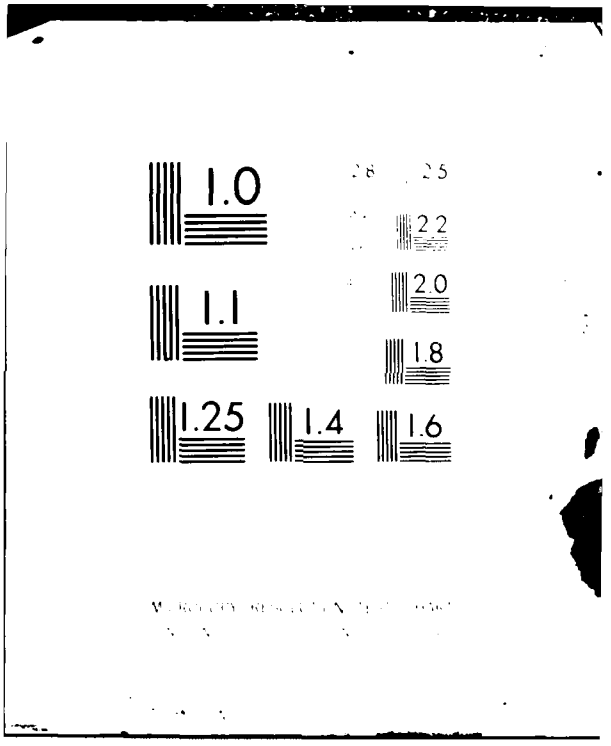
```
      begin
      repeat
        formcurrent;
        select;
        read(kbd,ch);
        if ch in ['1','2','3','4','5','6','?'] then
        begin
          if eoln(input) then readln;
          case ch of
            '1' : targetadd;
            '2' : targetlist;
            '3' : targetdel;
            '4','5','6' : exitoperations;
            '6' : menuerror;
            '?' : begin
                  lines(2);
writeln('  The three options below will display the targets and their');
writeln('formats for the select card, the target list and the select');
writeln('bulletin. Choose one of these three options or type a 'ret' ');
writeln('to continue operations and return to the previous menu.');
                  lines(1);
                  returnhere;
                  end
          end
        end
        else menuerror;
      until terminop = true;
    end;



    procedure formcurrent;

    begin
      clear;
      writeln(accts);
      lines(2);
      writeln('                    The display options are:');
      lines(2);
      writeln('                         1.  Target Card');
      writeln('                         2.  Target Listing');
      writeln('                         3.  Target Bulletin (Format)');
      writeln('                         4.',return:);
      lines(2)
      end;


      procedure Targetadd;

      begin
writeln('Photo No:                    Map Ref   : IRAN 4577-IV');
writeln('Photo Coord:                 Accuracy  : CONFIRMED');
```

```
writeln('Remarks: First tank sighting in sector IV. Attack w/ rockets');
lines(1);
writeln('--------------------STRIKE/BDA--------------------');
writeln('            Firing      NO/type       Damage       Damage');
writeln('   LTG       Unit        Rounds        Reported     Assessed');
writeln('   ---       ------      -------       --------     --------');
writeln(' 1215622    2 F/A-18     2  D-22       DESTROYED    DESTROYED');
writeln('BDA: Both tanks confirmed by AC. No AAA coverage on tgt.');
        lines(1);
        end;


      procedure leardls;

      begin
        clear;
writeln(dots);
writeln('                  : TARGET NO.  AD0612 :');
writeln('                  ---------------------');
writeln('Location: 3465e725          Alt: 8K     Type: TANK');
writeln('Description: 2 T-62 TANKS IN OPEN FIELD');
writeln('Class    : A                      Status : ACTIVE');
writeln('Priority : 1                   Tgt List: YES');
writeln('Assessr  : ALP');
lines(1);
writeln('Source of Tgt: ALP OBSERVER CV-1K');
writeln('LTG Activated: 1216622');
        learn11;
        spacebar;
        end;


      procedure listals;

      begin
        clear;
writeln(stars);
lines(2);
writeln('              TARGET LISTING');
lines(1);
writeln('TGT NO  CL  PRI LOCATION  ALT   SAASO    DESCRIPTION');
writeln('------  --  --- --------  ----  ------   -----------');
writeln('AD0621* A    II 3465b677   8K   ALP      2 T-62 TANKS IN OPEN');
writeln('AD0624* A     I 3452465c   14K  NGP      POSITIVE RUNNER CONFIRM');
writeln('AD0633* B    II 3455b654   4b   ARTY     BTY OF T-14V AT GUNS');
writeln('AL0654  F    IV 3455b566   1K   NGP      BRAGG FORTIFICATIONS');
writeln('AD0655* A    IV 3445b77c   5Z   NONE     SCHOOL BUILDINGS');
writeln('NAK211  C   III 4822b65b    0   NGP      BRAG: FORTIFICATIONS');
writeln('AD0657* A     I 3477b54b   12K  ALP      BTY 2S1-M6-4 IN TREES');
writeln('AD0657* B    IV 3455b7b5   14b  NONE     RAIL/SUPPLY DEPOT');
writeln('AL0656  F   III 3444e9AK   8K   ARTY     BTY 2UG IN TRENCHLINE');
writeln('AD0658* A     I 34225b9c   7K   ALP      12 VEHICLES ALONG ROAD');
        lines(2);
```

```
        writeln('    NOTE: * indicates target list');
        lines(1);
        spacebar;
      end;


      procedure inputlist;

      begin
clear;
lines(2);
writeln('    3.  CANCELLED TARGETS');
writeln('        AD2064, AD2750, AD2050, AD2612, AD2611');
writeln('        AD2646, AD2647, AD2161');
lines(1);
writeln('    4.  DEACTIVATED TARGETS');
writeln('        AD2677, AD2126');
lines(1);
writeln('    5.  CLASSIFICATION/PRIORITY CHANGES');
writeln('         AD2753   A   II');
writeln('         AD2754   C   IV');
writeln('         AD2779   D   III');
writeln('         AD2167   B    IV');
writeln('         AD2121   E     I');
writeln('         AD2221   A    II');
lines(1);
writeln('                                                CLASSIFICATION');
lines(2);
spacebar;
      end;


      procedure inputls;

      begin
      clear;
writeln('                    TARGET EVALUATION');
lines(1);
writeln('CLASSIFICATION');
lines(1);
writeln('    DTG   :  121052Z');
writeln('    FROM  :  CPL  (CTF32.1.1)');
writeln('    TO    :  DISTRIBUTION');
lines(1);
writeln('    SUBJ  :  TARGET NUMBER 12');
lines(1);
writeln('    1.  NEW TARGETS');
writeln('         AD2134  04555044  FORTIFICATIONS     A   I');
writeln('         AD2135  04526577  2 T-34 TANKS       A   I');
writeln('         AD2136  04507062  BUNKER COMPLEX     E   III');
lines(1);
writeln('    2.  BDA');
writeln('         AD2778  BV2 damaged by air strike');
writeln('         AD2135  Destroyed');
```

AD-A112 429 NAVAL POSTGRADUATE SCHOOL MONTEREY CA F/G 15/4
A PROTOTYPE PROGRAM FOR TARGET INFORMATION.(U)
JUN 81 R J COULTER
UNCLASSIFIED NPS52-81-007 NL




1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2

```
writeln('        ADVLIB  Partially damaged by artillery');
lines(1);
spacebar;
Tbuildisl;
end;



      procedure tgtinfo;

      begin
        clear;
        writeln('            Target listing information');
        writeln(dots);
        lines(1);
        getfile('queryinfo.text');
        lines(1);
        spacebar;
      end;

       procedure version1; forward;

       procedure versadd;

       begin
writeln('     CRT CONSOLE: Datamedia Elite 2500');
writeln('        LANGUAGE: Pascal');
writeln(' IMPLEMENTATION: UCSD Pascal (version 1.4)');
writeln('          DESIGN: ItCol R. J. Coulter, USMC');
writeln('     PROGRAMMING: ItCol R. J. Coulter, USMC');
lines(1);
spacebar;
       end;

       procedure version;

       begin
       clear;
       writeln(dots);
writeln('  Microcomputer System for Target Information  (MSTI)');
writeln('  ---------------------------------------------------');
writeln('                  Version 1.0');
lines(1);
writeln(' A prototype microcomputer data base operation system for the');
writeln('     target information section of the Marine Corps fire support');
writeln('     coordination center. It is the result of a masters thesis');
writeln('     submitted at the Naval Postgraduate School.');
lines(1);
writeln('       LOCATION: Department of Computer Science');
writeln('                 Naval Postgraduate School');
writeln('                 Monterey, California');
writeln('            DATE: 1st June 1981');
writeln('SOURCE COMPUTER: Altos ACS 8000-1');
writeln('OBJECT COMPUTER: Altos ACS 8000-1');
```

```
version1;
end;


procedure version1;

begin
clear;
lines(1);
writeln(' System supports upper and lower case. Character delete key ');
writeln('     is <runout> key. Input terminator is <return> key.');
lines(1);
writeln(' FOR INFORMATION:');
lines(1);
writeln(' Professor Lyle A. Cox,Jr.');
writeln('          Naval Postgraduate School');
writeln('          Monterey, California 93943');
writeln('          408-646-2449');
lines(1);
writeln('     LtCol Ronald J. Coulter, USMC');
writeln('          Development Center');
writeln('          MCDEC');
writeln('          Quantico, Virginia 22134');
writeln('          PHONE');
lines(5);
spacebar;
end;

     procedure sysopmenu;

     begin
       clear;
       writeln(dots);
       lines(1);
       writeln('                System Operation');
       lines(1);
       writeln('     The options are:');
       lines(1);
       writeln('          1.  Instructions for the User');
       writeln('          2.  Formats Used in Displays');
       writeln('          3.  Obtaining Information about Targets');
       writeln('          4.  System Technical Information');
       writeln('          5.',return);
       lines(2);
     end;



     procedure systemop;

     begin
       repeat
         sysopmenu;
```

```
        select;
        read(ch);
        if ch in menubar then
        begin
          if eoln (input) then readln;
          case ch of
            '1' : userinst;
            '2' : formats;
            '3' : tgtinfo;
            '4' : version;
            '5','E','e' : exit(systemop);
            '6' : menuerror;
            '7' : begin
                    lines(1);
writeln('    Five options concerning system operation are provided in');
writeln('in the above menu. Select the item you want from these options');
writeln('and type that number on the keyboard. If you do not desire any');
writeln('information on system operations, then use option 5 to return');
writeln('to the previous menu');
                    lines(1);
                    returnbar;
                    end
          end
          end
          else menuerror;
        until menuloop = true;
      end;




    procedure security;

    begin
    clear;
    writeln('           Security Guidelines');
    writeln(dots);
    lines(10);
    writeln('   TO BE IMPLEMENTED');
    lines(7);
    spacebar;
    end;


    procedure tgtclass;

    begin
      clear;
      lines(1);
      getfile('class.text');
      lines(1);
      spacebar;
    end;
```

```
      procedure tetpri;

      begin
        clear;
        lines(1);
        getfile('priority.text');
        lines(3);
        spacebar;
      end;


      procedure anal1; forward;
      procedure anal2; forward;
      procedure anal3; forward;
      procedure anal4; forward;
      procedure anal5; forward;

      procedure tetanal;

      begin
      clear;
      lines(1);
      writeln('                Target Analysis Guidelines');
      writeln(dots);
      lines(1);
writeln('  The following format ensures a logical and orderly examination');
writeln('of all factors to determine the best method of attack of a target.');
lines(1);
writeln('Situation of opposing forces:');
writeln('----------------------------');
writeln('Enemy situation...include information that will aid target analysis.');
lines(1);
writeln('Friendly situation...information that will aid attack of the target.');
lines(1);
writeln('Target characteristics:');
writeln('----------------------');
writeln('Target description....type(personnel, materiel, terrain), number');
writeln('                    of personnel, quantity of materiel and activity.');
lines(1);
writeln('Vulnerability...type and amount of cover, type of materiel, type');
writeln('              of construction, mobility and density of personnel');
writeln('              and material.');
lines(1);
spacebar;
anal1;
end;

procedure anal1;

begin
clear;
lines(1);
writeln('Physical location....grid reference, altitude of target, location');
```

```
writeln('                    of friendly forces and terrain features.');
lines(1);
writeln('Accuracy....of the target location and the agency reporting the');
writeln('              target.');
lines(1);
writeln('Size of area...dimensions and shape of the target area and the');
writeln('              distribution of personnel and material in the area.');
lines(1);
writeln('Terrain and weather...brief analysis of terrain and weather');
writeln('              in the target area. Include any terrain features');
writeln('              which affect the means and method of attack.');
lines(2);
writeln('Target Capabilities:');
writeln('--------------------');
writeln('    The capabilities of the target as they affect the accomplishment');
writeln('  of the mission of the supported unit. Show how a terrain feature');
writeln('  affects enemy capabilities.');
lines(1);
spacebar;
anal2;
end;

procedure anal2;

begin
clear;
lines(2);
writeln('Other Factors:');
writeln('--------------');
writeln('  How do the following affect the  ability of firepower, method');
writeln('of attack and delivery means:');
lines(1);
writeln('Enemy at ta....determined by the type of target (static or');
writeln('              fleeting) and its capabilities.');
lines(1);
writeln('Enemy countermeasures...ability of the army to minimize the');
writeln('              effects of firepower, prevent delivery of supplies');
writeln('              arms and bring countermeasures against military units');
writeln('              after attack.');
lines(1);
writeln('Enemy discipline...factors which will aid in determining the');
writeln('          amount of firepower required to neutralize the target');
writeln('          and discipline of enemy personnel.');
lines(1);
writeln('Obstacles....considerations concerning the possibility of');
writeln('          creating obstacles by attacking the target.');
lines(1);
spacebar;
anal3;
end;


procedure anal3;
```

```
begin
clear;
lines(2);
writeln('Civilian casualties...approximate number of civilians in the');
writeln('       target area at the estimated moment of missile explosion.');
writeln('       Casualties.');
lines(1);
writeln('Surprise....extent to which missile attack is likely to');
writeln('            exploit the element of surprise.');
writeln('            Restrictions on civilians to control.');
lines(1);
writeln('Means of Attack:');
writeln('----------------');
writeln('      All available types of missiles and launching platforms');
writeln('with which it is practical to attack the target area.');
writeln('Cost penalties associated with each means.');
lines(1);
writeln('Analysis of Means of Attack:');
writeln('----------------------------');
writeln('  The effect of each of the factors listed above.');
writeln('Target vulnerability and its relative importance.');
writeln('Ranking the means of attack.');
lines(1);
spacebar;
end;

procedure menu4;

begin
clear;
lines(2);
writeln('1.  Inventory of [illegible] missiles and launching platforms.');
lines(1);
writeln('2.  Extent of available [illegible].');
lines(1);
writeln('3.  Estimate of [illegible].');
lines(1);
writeln('4.  Estimate of civilian casualties.');
lines(1);
writeln('5.  Estimate of probable [illegible].');
lines(1);
writeln('6.  Precautions required for [illegible].');
lines(1);
lines(1);
writeln('Comparison of Means of Attack:');
writeln('------------------------------');
writeln('   The outstanding alternatives are displayed [illegible]');
writeln('of attack and determine which offers the greatest chance of');
writeln('success.');
lines(1);
spacebar;
end;
```

```
procedure xxxx;

begin
ch:=' ';
lines(2);
writeln('                    Press <RETURN>');
writeln('--------------------------------------');
lines(1);
writeln('1.  Type the first letter of the desired option.');
lines(1);
writeln('2.  Press the <RETURN> key.');
lines(1);
writeln('3.  Only certain keys will activate the menu.');
lines(1);
writeln('4.  Type ? after .');
lines(1);
writeln('5.  Select procedure, enter a blank line to return.');
lines(1);
writeln('6.  Refer to introductory documentation.');
lines(2);
writeln;
end;
```

```
        repeat writeln;
           write('Option : [',c,',',d,',',e,',',f,',',g,',',h,',',i,',',j,']');
         repeat
           ch:=upcase(readkey);
           sound(440);
           delay(50);
           nosound;
           if ch in ['A'..'J'] then
             begin
             if ch=chr(13) then ch:=a;
             case ch of
                 'A' : system;
                 'B' : shuttle;
                 'C' : interface;
                 'D' : text_io;
                 'E' : internal;
                 'F','G','H','I','J' : exit(xxxx);
                 '?' : help
        lines(1);
        writeln('  Six options are provided in the menu to the right.');
        writeln('The one that you will most often use is option A. Type the letter A');
        writeln('or the keyboard and the system will respond with the');
        writeln('highlighted letter. If you do not select one of the first');
        writeln('four by the 9th time, the last one will be selected.');
        lines(1);
        return;
```

```
(*            UTILITY.TEXT          *)

segment procedure utility;

    var
          response : myline;
          i : integer;

 procedure change;


procedure passtl;

  begin
  lines(1);
writeln('     The password is changed by creating the  PASSWD  system  subtask  ');
writeln('an input and writing the new password to the PASSWD file.  At the  ');
writeln('end of an operation it is highly advised that all users  logoff  ');
writeln('This will prevent them to their original  user  status.  Initially  ');
writeln('the system will automatically this.');
   lines(1);
   end;


 procedure filist;

    begin
      clear;
      lines(6);
writeln('     There are 3 passwords in the talent initially  (upon  ');
writeln('the system password which consist of numbers. This will  ');
writeln('enable that at least one of the passwords will  ');
writeln(' be the same as the system password.  Then  the  following  ');
writeln('passwords can be input by the  user  to  allow  them  ');
writeln('to have exclusive access to the target file.');
lines(1);
writeln('     Initially, these 3 user passwords are  "null"  and  press  ');
writeln('until changed by this procedure. The password can  be  ');
writeln('to 12 letters or numbers. Examples of passwords  that  be  ');
writeln('personnel last names (SMITH, JONES, SMITH), special  words  ');
writeln('numbers (social security, telephone) or any  combination  ');
      passtl;
      spacebar;
    end;


procedure currentPA;

begin
  clear;
  lines(4);
  writeln('          CURRENT PASSWORDS');
```

```
  writeln('          ----------------------');
  lines(1);
  writeln('          1.  ',userid[1]);
  writeln('          2.  ',userid[2]);
  writeln('          3.  ',userid[3]);
  writeln('          4.  ',userid[4]);
end;



  procedure changepw;

  var  item : integer;
       ok : boolean;
       newpw : string[8];

  begin
    ok := false;
    currentpw;
    writeln('          0.',return);
    lines(2);
    writeln('      ENTER THE NUMBER OF THE PASSWORD YOU WISH TO CHANGE');
    prompt;
    read(item);
    if item in [1..4] then
    begin
      if eoln(input) then readln;
      if item = 0 then exit(changepw);
      repeat
        lines(1);
        writeln('      ENTER NEW PASSWORD.....  [ return ] to cancel');
        prompt;
        readln(newpw);
        if length(newpw) > 1 then
        begin

        {   function  }

          lines(1);
  writeln('      The password can be up to 8 characters in length,');
  writeln('such as a name, SS# or letter-number combination.  Enter it');
  writeln('characters in the space after the system prompt and press the');
  writeln('RETURN key. The prior password will be automatically replaced by');
  writeln('new password substituted for it. If you just press the RETURN key,');
  writeln('the prior password will remain.');
          lines(1);
        end
        else ok := true;
        until ok = true;
        end
        else if length(newPW) = 0 then exit(PWchange);
        lines(2);
        writeln('       Password Changed');
        lines(2);
```

```
    return(r);
  end;

begin {main program}
  repeat
    clear;
    lines(3);
    writeln('                    Graphics Demonstration');
    writeln('Note:');
    lines(1);
    writeln('       This program will');
    lines(1);
    writeln('          1.  Insert disk with CP/M system on it in drive A');
    writeln('          2.  Place a blank disk in drive B');
    writeln('          3.  Strike any key');
    writeln('          4. Return');
    lines(1);
    select;
    read(kbd,
    if eof(kbd) then
    case
      1: writeln('Disk not ready');
      2: case ch of
        '1','2' : initf;
        'A' : begin
                clrscrn;
                lines(3);
                boldon;
              end;
        'B' : pagframe;
        'C','c' : writeln('Hello');
        'D','d','E','e' : repeatloop;
      end;
    end
    else repeatloop;
  until repeatloop = true;
end;


procedure initf;

      var  lines : integer;


procedure displaymenu;

  begin
    clear;
```

```
```

```
```

```
  writeln('    ... ... ...');
  lines(2);
  for i := ... to ... do
    begin
      write;
      ...;
      writeln('    ...');
      lines(1);
    end;
  writeln('                                                    ');
  writeln('                                                    ');
  lines(1);
  writeln('                                                    ');
  writeln('                                                    ');
  lines(2);
  ...;
end;



procedure ...;


begin
  ... := ...;
  ...;
  lines(2);
  writeln('                    ...');
  lines(1);
writeln('    ...                                                    ');
writeln('...                                                       ');
writeln('is ...                                                    ');
writeln('is ...                                                    ');
lines(2);
writeln('      ...                                                 ');
  prompt;
  readln(line);
  if ... = '/' then exit(...);

  { ... }

  lines(4);
  writeln('     ...                                                ');
  lines(1);
  writeln('      ...                                               ');
  lines(1);
  ...;
end;



procedure ...;

begin
```

```
repeat
  reset;
  writeln('       Start - Raw Terminal File');
  writeln;
  line := 1;
  writeln('       Type options here');
  line := 1;
  writeln('         1.  Insert a new Terminal file');
  writeln('         2.  Insert existing');
  writeln('         3.  Return');
  lines(4);
  select;
  read(kb);
  if ch in ['1','2','3','4','5','6'] then
  begin
    if ord(ch) = ord('1') then
      case ch of
        '1': begin
               line := 1;
               writeln('       Insert a new Terminal file');
               line := 2;
               line := line + 1 ;

               {  insert  }

               writeln('     ** Insert file  ');
               line := 3;
               select;
               writeln('Terminal');
             end;
        '2','3' : begin
                    line := 3;
writeln('     Enter the name of the file to be inserted after current line');
writeln('Press return key to display a list of files in the current directory');
writeln('You may enter the name of a file in another directory, but you must');
writeln('a new name for the file that will be created in the current directory');
                    line := 1;
                    delete;
                  end;
        '4','5','6' : writeln('Terminal');
      end
    else
      begin
      else beep(1);
    until ord(ch) = ord('3');
  end;


procedure Lines;

  begin
    lines(4);
writeln('     The translations of the keys are shown in the following table:');
writeln('Each terminal in the Terminal file has its own table.  The table');
writeln('the key, how it is translated, and whether it is a function key.');
```

```
```

```
procedure printlist;


procedure printmenu;

begin
  clear;
  writeln('          Public Table List');
  writeln('List');
  lines(1);
  writeln('     Its options are:');
  lines(1);
  writeln('          L.  Print the Control List');
  writeln('          M.  Print the List of Commands');
  writeln('          P.  Print the Table Layout Only');
  writeln('          Q.  Print a Table Layout (+L)');
  writeln('          S.  Information');
  writeln('          T.  Information on Special Table Lists');
  writeln('          ?.  Information');
  lines(1);
  select;
end;


procedure printinfo;

  begin
    lines(1);
writeln('     The printable list prints all the [illegible]');
writeln('level 1: the table [illegible]');
writeln('to the system it [illegible]');
writeln('from the data base [illegible]');
writeln('on printable lists with special parameters [illegible]');
writeln('[illegible] of the main menu.');
    lines(1);
  end;


procedure printall;

begin
  clear;
writeln('     This procedure prints the complete table [illegible]');
writeln('to the list printer for [illegible]');
writeln('[illegible] personnel. The table [illegible]');
lines(1);
writeln('Col #  Cl [illegible]  Alt  Type    [illegible]');
writeln('------ - --- --------- ---- ------ ----------');
writeln('[illegible]');
writeln('[illegible]');
writeln('[illegible]');
writeln('[illegible]');
```

```
writeln('Please ');
writeln('
lines(1);
writeln('
  printer;
  spooler;
end;



procedure [illegible];

begin
  clear;
lines(3);
writeln('
writeln('
lines(2);
writeln('
writeln('
writeln('
writeln('
writeln('
writeln('
writeln('
lines(1);
writeln('
writeln('
writeln('
writeln('
writeln('
lines(1);
spooler;
end;



procedure print;

begin
  clear;
  lines(4);
  write('
  for t := 1 to 2 do
  begin
    delay;
    write(dot);
  end;
  lines(2);
  writeln('
  lines(1);
```

```
  spacebar;
end;


procedure printlist;

begin
  clear;
  lines(4);
  write('        Printing the list of [illegible]...');
  for t := 1 to 10
  begin
    delay;
    write(dot);
  end;
  lines(3);
  writeln('          *** Function complete ***');
  lines(1);
  spacebar;
end;



procedure printall;

begin
  clear;
  lines(4);
  write('    Printing All Active and Inactive [illegible]...');
  for t := 1 to 10
  begin
    delay;
    write(dot);
  end;
  lines(3);
  writeln('          *** Function Complete ***');
  lines(1);
  spacebar;
end;


procedure writedata;

var  tour : string[10];

begin
  repeat
    option := '12';
    clear;
    lines(2);
    writeln('      Which [illegible] do you [illegible]');
    prompt;
    readln(tour);
    if (tour = '01') or (tour = '02') then
```

```
    else if ... then ...
    else if ... then ...
    begin
      retout := true;
      lines(1);
writeln(' The target number consists of two digits or two digits and a letter.');
writeln(' For example, a response of "7" or "7A" is acceptable.');
lines(1);
writeln(' To print all of the ... type ... or ...');
      lines(1);
      return;
    end;
  until retout = false;
  lines(0);
  write(' Printing the ... for target ''...'' ');
  for t := 1 to ... do
  begin
    delay;
    write(out);
  end;
  lines(0);
  writeln('          ... printing ...');
  lines(1);
  spacout;
end;



begin {printit};
  repeat
    printscr;
    read(c);
    if ... in ... then
    begin
      if ... then ...;
      case ... of
        '1' : print1;
        '2' : print2;
        '3' : print3;
        '4' : print4;
        'P','p' : printrpt;
        'S' : specialrpt;
        'E','e','X' : exit(printit);
        'R' : returnrpt;
      end
    end
    else termination;
  until ... = true;
end;


procedure stats;
```

```
begin
  clear;
  lines(4);
  writeln('  processing Secret Classified Information.....');
  for i := 1 to 10 do
  begin
    delay;
    write('.');
  end;
  lines(1);
  writeln('        of Classified Information Complete  ');
  lines(4);
  return(c);
end;


procedure erase_info;

  begin
    lines(1);
writeln('     This procedure will erase every file in the ');
writeln('all memory the Secret Information. If you have ');
writeln('the erase directory. This is how to erase ');
writeln('the classified information for the disk!! .');
lines(2);
writeln(' The initialization procedure will erase ');
writeln('classifies the diskette and should only be ');
writeln('of an operation and the data is no longer ');
    lines(1);
  end;



procedure erase;

begin
  repeat
    clear;
    lines(1);
    writeln('          Erase the Secret File');
    writeln(dots);
    lines(1);
    writeln('          The options are:');
    lines(1);
    writeln('            1.  Information');
    writeln('            2.  Erase file');
    writeln('            3.',return);
    lines(1);
    select;
    read(ch);
    if ch in ['1','2','3','4','5','6'] then
    begin
      if eoln(input) then readln;
      case ch of
```

```
        '1','?' : begin
                    clear;
                    erase(1);
                    space(1);
                  end;
        '2' : begin
                lines(6);
                write('      Erase All Files.....');
                illcheck := true;
                initialize;
                clear;
                lines(2);
writeln('                    ** Function Complete **');
lines(6);
writeln('      Diskette file erased....Return to main menu.....');
lines(2);
writeln(' **  Part operation by selecting option 2 of the main menu ');
writeln('                  .....then restart system');
                lines(2);
                space(1);
                retout := true;
                exit(erase);
              end;
        '3','e','E' : exit(erase);
      end
    end
    else tenterror;
  until penultcop = true;
end;



procedure copyfiles;

  begin
    lines(2);
writeln('    **  Place the current target diskette in disk drive A **');
writeln('                        (Front Side)');
lines(1);
writeln('    **  Place the back-up diskette in disk drive B **');
writeln('                        (Front Side)');
lines(1);
writeln('    **  Press the RETURN key, the system will automatically');
writeln('        copy the target information from drive A to drive B. **');
lines(1);
writeln('    **  When FUNCTION COMPLETE appears, do the following:');
writeln('          * Remove the back-up diskette from drive B');
writeln('          * Remove the target diskette from drive A');
writeln('          * Place the system diskette back in drive A');
writeln('          * Place the target diskette back in drive B');
writeln('          * Press the RETURN key');
lines(2);
writeln('      PRESS RETURN TO COPY DATA IN:');
    prompt;
```

```
  end;


procedure copyis;

var copyair : string[4];

begin
  clear;
  lines(1);
  writeln('              Copy data base procedure');
  writeln(dots);
  lines(2);
writeln('   This procedure allows you to make a backup copy of');
writeln('   the target diskette. It requires you to switch the');
writeln('   diskettes in the disk drives and use a pre-formatted');
writeln('   back-up diskette. If you do not desire to copy the');
writeln('   data base, then press the RETURN key to return to the');
writeln('   previous menu. The directions in this section must be');
writeln('   followed exactly.');
  lines(3);
  writeln('   ** Type COPY and press the RETURN key');
  prompt;
  readln(copyair);
  if (copyair = 'COPY') or (copyair = 'copy') then
  begin
    clear;
    copyinfo;
    repeat
      read(c);
    until eoln(input);
    clear;
    lines(2);
    write('         COPYING DATA BASE...');
    for t := 1 to 10 do
    begin
      write(dot);
      delay;
    end;
    lines(3);
    writeln('     ** FUNCTION COMPLETE **');
    lines(1);
    returnter;
  end
  else exit(copyis);
end;




  procedure utilmenu;
```

```
begin
  clear;
  lines(1);
  writeln('              THE SYSTEM UTILITY FUNCTIONS');
  writeln(dots);
  lines(1);
  writeln('        The options are:');
  lines(1);
  writeln('            1.  Change the password');
  writeln('            2.  Copy the data base file');
  writeln('            3.  Construct the Journal');
  writeln('            4.  Print the telex out');
  writeln('            5.  Already telex file statistics');
  writeln('            6.  Clear the telex files');
  writeln('            7.  Information on the system times');
  writeln('            8.',return);
  lines(1);
  select;
end;



procedure utilfc;

  begin
writeln('     The fourth option prints the list of telexes, the telex file');
writeln('and telex information in the telex file format. Anything on the');
writeln('by special parameters (like all those #, which is to be used');
writeln('by a different procedure. The statistics display shows a further');
writeln('breakdown of the categories of information in the list of telexes.');
lines(1);
writeln('     Option 6 erases all the information from the diskette. This is');
writeln('done at the end of an operation to facilitate the transmission.');
writeln('The last option returns you to the main command menu.');
  lines(1);
  end;


    procedure utilinfc;

   begin
     clear;
     lines(3);
writeln('     This section provides various housekeeping procedures for');
writeln('the IIO. The first option allows you to change the passwords');
writeln('for the system users. The second permits you to copy the telex');
writeln('files from the target diskette to a second diskette to function');
writeln('as a backup file. The third option constructs a telex edition');
writeln('(JOURNAL) from all the data base transactions since the last JOURNAL');
writeln('was printed. The routine will let you view or print this');
writeln('information and print it in the proper order for it.');
     lines(1);
     utinfo;
```

```
      spacebar;
    end;



begin {utility}
  menunum := ['1','2','3','4','5','6','7','8','9','0'];
  clear;
  getfile('menu5.text',);
  spacebar;
  repeat
    utilmenu;
    read(cn);
    if cn in menunum then
    begin
      if eoln(input) then readln;
      case cn of
        '1' : changedir;
        '2' : copyfile;
        '3' : textout;
        '4' : printtext;
        '5' : stats;
        '6' : begin
                erase;
                if setout then
                begin
                  setout := false;
                  exit(utility);
                end;
              end;
        '7','8' : utilmenu;
        '9','0','r' : exit(utility);
      end
    end
    else menuerror;
  until menuloop = true;
end;
```

::::::::::::::::::::: [illegible].TXT :::::::::::::::::::::

[illegible]

::::::::::::::::::::: [illegible].TXT :::::::::::::::::::::

[illegible]

::::::::::::::::::::: [illegible].TXT :::::::::::::::::::::

[illegible]

1. [illegible]
2. [illegible]

::::::::::::::::::::: PROGRES.TXT :::::::::::::::::::::

[illegible]

:::::::::::::::::::::: [illegible] ::::::::::::::::::::

[illegible]

1. Target Level
2. Unit Location
3. Destination
4. Priority
5. Classification
6. Status
7. Target Type

Since this [illegible]

::::::::::::::::::::: FUNCTION ::::::::::::::::::::::

The function unit is the designation of the primary unit which undertakes the attack of the target. The input is the first letter of each of the function. For example, if you would like, Artillery, Air, Artillery...Artillery, Infantry, Armor, enter the appropriate code or press the return key.

::::::::::::::::::::: TARGET.DAT ::::::::::::::::::::::

After the unit has been [illegible] the target data base. The input is the number of the target. If the target is [illegible], press the return key. An example of a unit would be Artillery...37 / Fire Fly, Artillery...16 - Iron Man.

::::::::::::::::::::: PRINT.DAT ::::::::::::::::::::::

The length of the input [illegible] please [illegible] or less and press the return key

::::::::::::::::::::: [illegible] ::::::::::::::::::::

The options are:

1. [illegible]
2. Destroyed
3. [illegible]
4. [illegible]
5. Neutralized
6. Eliminated
7. [illegible]

:::::::::::::::::::::::: [illegible] ::::::::::::::::::::::::::::

[illegible]

[illegible]:

1. A
2. B
3. C
4. D
5. E

:::::::::::::::::::::::: [illegible] ::::::::::::::::::::::::::::

[illegible]

[illegible]:

1. I
2. II
3. III
4. IV

:::::::::::::::::::::::: [illegible] ::::::::::::::::::::::::::::

[illegible]

[illegible]:

1. Data
2. Cost Center
3. Installation
4. Control Cards
5. Operating Cost
6. Terminal
7. Vendors
8. Certifications
9. Miscellaneous

:::::::::::::::::::: [illegible] ::::::::::::::::::::::

The target type is entered here from the target worksheet. The information will be used to group targets of the same type together. If the target cannot be classified into these categories, use Miscellaneous (option 9).

:::::::::::::::::::: [illegible] ::::::::::::::::::::

The attribute is entered as the total value of the item. It may be four digits from 0001 to 9999. For example, 400. The dollar cost be entered in cents. If the attribute is not known, then press the return key. Please [illegible] data.

:::::::::::::::::::: [illegible] ::::::::::::::::::::

[illegible] will be issued [illegible] artillery [illegible] assistant [illegible] which [illegible] should be chosen. This [illegible] the [illegible] If a subsystem [illegible] this [illegible]

::::::::::::::::::::::: REMARKS.TXT :::::::::::::::::::::::::

The remarks section may be used to [illegible] include both numbers and letters. If the [illegible] characters, they are [illegible] remarks usually include [illegible] of fire, attack restrictions and other [illegible] example...attack with [illegible]

::::::::::::::::::::: [illegible] :::::::::::::::::::::::

Indicate the [illegible] characters. This [illegible] or [illegible] as determined by [illegible]

::::::::::::::::::::: SUBSYS.TXT :::::::::::::::::::::::

Indicate from what [illegible] obtained; artillery survey, air observer, [illegible] aerial photo, [illegible] [illegible] 2% [illegible]

::::::::::::::::::::: REMARKS.TXT :::::::::::::::::::::::

If the target was obtained [illegible] enter the photo number and press the [illegible] example, A211234. If the target [illegible] then just press the return key.

::::::::::::::::::::: PHOTOGRID.TXT :::::::::::::::::::::

** If the photo grid location is the same as the target location, enter an N and press the return key. Otherwise, enter N right [illegible] return key.

ENTER PHOTO COORDINATES

::::::::::::::::::::: PHOTOGRID.TXT :::::::::::::::::::::

The coordinates of the photo target should be entered as a and consist only of numbers. For example, [illegible] A six number coordinate such as 045075 can be shifted [illegible] digit one by adding zeros to give 045075.

If the photo grid location is the same as the target [illegible]

[illegible] enter or 0 to press the return key. Press [illegible]

::::::::::::::::::::::: Co.ACC.DAT :::::::::::::::::::::::::

[illegible] represent the accuracy of the target information. A target which is known to exist is a certain target. A probable and a possible target may [illegible] to exist. If the intelligence evaluator does not feel that any of these areas, then enter option 4 for completion of this [illegible] the system will automatically enter the [illegible]

::::::::::::::::::::: Crewsight.DAT ::::::::::::::::::::::

This [illegible] can only be used to [illegible] that already exist in the list of targets. [illegible] any changes the existing target information with the new [illegible] you will be asked. The system will [illegible] information and ask if you desire to change.

A [illegible] will cause the [illegible] to be changed. [illegible] will cause the system to [illegible] the target that will not be changed [illegible] key. [illegible]

Changes to the [illegible] the [illegible] [illegible] are permitted to obtain [illegible] other [illegible]

::::::::::::::::::::: Create.DAT ::::::::::::::::::::::::

The first option will display the target [illegible] checks each line of target information and allows you to [illegible] those areas desired. The third option writes the [illegible] file and the last option returns you to the [illegible]

::::::::::::::::::::: PRIORI.DAT ::::::::::: :::::::::

Target priorities

PRIORITY I....Targets capable of [illegible] the [illegible] of the risk of action by the [illegible] and its elements.

PRIORITY II...Targets capable of [illegible] with the risk of action by the [illegible] and its elements.

PRIORITY III..Targets capable of [illegible] with the risk of action by the [illegible] and its elements.

PRIORITY IV...Targets capable of [illegible]

[illegible]
and its elements.

[illegible]

Option 1 will return you to the [illegible]
[illegible] so that you may enter the [illegible]
allows you to proceed to the [illegible]
the target in the file, whether [illegible]
to the left of a-1-b-1. [illegible]
you to select the other option [illegible]

[illegible]

You may enter a [illegible]
the other will have you [illegible]
the data by the [illegible]

This procedure [illegible] You will [illegible]
You will then be given the option of writing the [illegible]
file, displaying the file [illegible]
either by descending the [illegible] sort criteria.

To quit this [illegible] type a [illegible]

To print data in [illegible] system prompt says [illegible]
an example of the output [illegible]

Follow all [illegible]

[illegible]

This is the [illegible]

Option 1...provides [illegible] how to operate the [illegible]
additional [illegible] the target file, [illegible]
requirements, inputs used, target [illegible]
and other files.

Option 2...enables you to edit a target [illegible]
the target file, obtain information [illegible]
and display all the [illegible]
target on the screen.

Option 3...enables you to attach a target [illegible]
a parameter or a list of [illegible]
include classification, its file, [illegible]
inactive/attached, and [illegible]
All information for a [illegible]
to displayed.

:::::::::::::::::::::: [illegible] ::::::::::::::::::::::::::::

Option 1...allows you to enter a new target into the data base, print information about a target, change the data base, print information about the data base, display the plane of fire as well as provide a suitable [illegible] if the list is short.

Option 2...initializes the data base to allow the entry of a new operation. All data in the current data base will be destroyed. New information can be entered. The option may also be used to add new data.

Option 3...provides this brief description about the system. No important information can be obtained from this option.

Option 4...exits the operation of the system after writing important information to the disk file. The system will have to be restarted before the data is available.

:::::::::::::::::::::: [illegible] ::::::::::::::::::::::::::::

If you [illegible] pressing [illegible].

:::::::::::::::::::::: [illegible] ::::::::::::::::::::::::::::

These procedures operate on the data base of targets. Option 1 allows you to add a new target to the list of targets. Option 2 allows you to change any information about a select [illegible] the target file. Option 3 displays all the information about a particular target on the select [illegible] target. The target can be found either by target number or grid coordinates.

Option 4 allows you to add a new target [illegible] to the target based on the results of the X-ray shot. Option 5 deletes a target completely from the list of targets. The final option returns you to the main menu. This menu also allows you to use the letter X for exiting targets, D for displaying targets and C for changing targets.

:::::::::::::::::::::: [illegible] ::::::::::::::::::::::::::::

Enter the date-time group that the target was [illegible] supporting arms. The first 2 digits of the [illegible]

(1..31) and the next 2 are the 10 or the year (e.g., 85).
The letter indicates the time zone. For example, also C, E, M, P, or T. The third time zone would be written, 31dec85c. Enter the date and press the return key. If the day of activation is not known, then just press the return key.

::::::::::::::::::::: target.txt ::::::::::::::::::::::::

The Battle Damage Assessment (BDA) can be up to 80 characters long. This is a concise narrative of the surveillance of the target based on the observer report. If there is no report of observation, then this section may be skipped. An example of a BDA is: 4 secondary explosions, 2 vehicles burning.

::::::::::::::::::::: remarks.txt ::::::::::::::::::::::::

Each target has three spaces for recording the target surveillance as a result of attack by supporting arms. This procedure prompts you for each piece of information for the Battle Damage Assessment (BDA). If a return key is entered, it will write over the first BDA space that is the oldest.

Information on multiple surveillances can be retained in the remarks section of the target report by using the update target procedure. Use the display option to view the current BDA information in the target file.

::::::::::::::::::::: bdadtg.txt ::::::::::::::::::::::::

Enter the date-time-group that the target was activated or added to the list of targets. The first 2 digits are the day of the current month (1..31) and the next 4 are the time (0001..2359). The letter indicates the time zone. For example, 1000 on 01 May in the MDT zone would be written, 01may85m. Enter the date and press the return key. If the day of activation is not known, then just press the return key.

::::::::::::::::::::: class.txt ::::::::::::::::::::::::

Target Classification
.....

CLASS A....Targets that threaten ships, aircraft, transshipping and UDT operations.

CLASS B....Targets that threaten assault forces in the ship-to-shore movement and assault of the beach.

CLASS C.....targets that influence of (press ...ance to...
operations either severely or affect the ability
of the enemy to continue resistance.

CLASS D.....targets that will not be fired upon prior to
...ney.

CLASS E.....targets that must not be destroyed to avoid
probable future use of it by friendly troops unless
authorized by the commander.
.....

:::::::::::::::::: [screen].HLP ::::::::::::::::::::::::

ENTER TARGET ACCURACY

The options are:

1. Confirmed
2. Probable
3. Possible
4. Unknown

:::::::::::::::::: [screen].HLP ::::::::::::::::::::::::

ENTER DISPOSITION OF ACCIDENTAL TARGET

The options are:

1. Hit
2. Not
3. Min
4. Hit, Hit
5. Hit, Not
6. Hit, Not
7. Hit, Hit, Not
8. Other
9. None

:::::::::::::::::: [screen].HLP ::::::::::::::::::::::::

The options are:

1. Display the target data
2. Page through the target data
3. Write the changes to the file
4. Return to previous menu

:::::::::::::::::: [screen].HLP ::::::::::::::::::::::::

INSTRUCTIONS FOR ADDING A TARGET
.....

The system will ask for each item of information.

Enter the required information and press the RETURN key.

To leave this procedure at any time, type a ^ while the cursor is at

To skip a section, just press a RETURN key and the system will go
to the next item of information unless the information requested
is mandatory. In that case you must enter information.

To receive more information about this procedure, type a ?

** To continue, press the RETURN key **

::

DISTRIBUTION

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93943	2
3.	Department Chairman, Code 52 Department of Computer Science Naval Postgraduate School Monterey, California 93943	1
4.	Professor Lyle A. Cox, Jr., Code 52Cx Department of Computer Science Naval Postgraduate School Monterey, California 93943	1
5.	Commanding General Attn: Fire Support Coordinator 1st Marine Division, FMF Camp Pendleton, California 92055	1
6.	Commanding General Attn: Fire Support Coordinator 2nd Marine Division, FMF Camp Lejeune, North Carolina 28542	1
7.	Commanding General Attn: Fire Support Coordinator 3rd Marine Division, FMF FPO San Francisco 96602	1
8.	Commanding General Education Center Attn: Supporting Arms Branch MCDEC Quantico, Virginia 22134	1

8. Commanding General 1
Development Center
MCDEC
Quantico, Virginia 22134

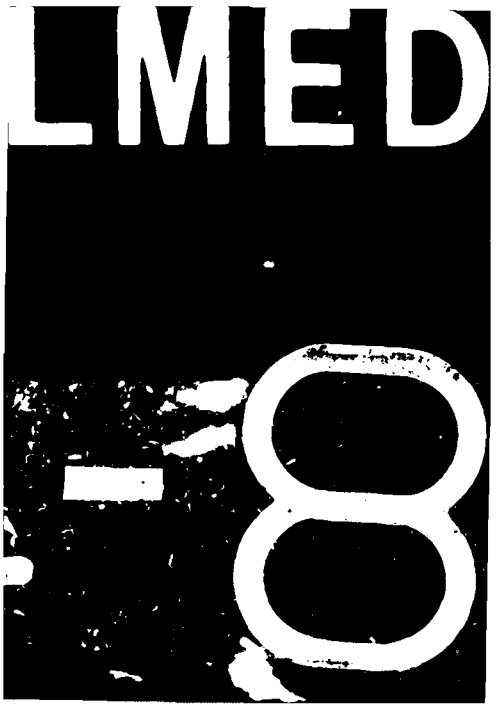
9. Commanding General 1
Attn: Fire Support Section (TEC)
MCAGCTC
Twentynine Palms, California 92227

10. Marine Corps Representative 1
U. S. Army Artillery School
Fort Sill, Oklahoma 73503

11. TRADOC Research Element Monterey, Code [illegible] 1
Naval Postgraduate School
Monterey, California 93940

12. Commanding Officer 1
MCTSSA
Attn: MIFASS Team
Camp Pendleton, California 92055

13. LtCol Ronald J. Coulter, USMC 1
6033 Burnside Landing Drive
Burke, Virginia 22015


LMED
8