AR-006-453


ELECTRONICS RESEARCH LABORATORY

# Electronic Warfare Division

GENERAL DOCUMENT
ERL-0532-GD

## LOGGING PROGRAMMES FOR TRIAL 'THUNDERSTOP'

Robert J. Rossiter

ABSTRACT (U)

This document presents two computer programmes written for Defence Trial 7/423 code-name 'THUNDERSTOP'. The programmes are for the ground station and airborne logging systems.



SEPTEMBER 1990

COPY No.

APPROVED FOR PUBLIC RELEASE

Postal Address: Director, Electronics Research Laboratory,
PO Box 1600, Salisbury, South Australia, 5108.

ERL-0532-GD

# CONTENTS



## LIST OF FIGURES



## LIST OF APPENDICES

THIS IS A BLANK PAGE

# 1 INTRODUCTION

The programmes discussed in this document were written in support of Trial 7/423 code-name 'THUNDERSTOP' (Brunker 1990 and Brunker, Grant, Rossiter and Oermann 1990). The Trial involved the testing of a laser warning receiver (LWR) against three different lasers. These lasers are:

(i) a commercial range finder (SIMRAD)

(ii) a laser target designator (LTD)

(iii) a beam rider simulator (BRIS)

There were two computer programmes written. The first "GSTN" was written for the ground station logging system, and the second "LWREC" was written for the airborne logging system (mounted in a helicopter).

# 2 GROUND STATION LOGGING PROGRAMME - GSTN

## 2.1 Function

This programme was written to monitor and record the response from the Laser Truth Detectors (which detect the emitted radiation when the laser has fired), to output the information to a video screen, and store data for later use. The truth detectors generated a TTL pulse each time the laser fired. These pulses were monitored by a PCL-714 Digital I/O card mounted in each computer. The digital output from these cards was sent to a Matrox Card for insertion of characters on the video signal.

The data was stored on disk at the completion of each run. The ability to change the computer's time was also incorporated in the programme such that it could be checked and rectified if necessary to match an external master clock.

## 2.2 Programme Specifications

| | |
|---|---|
| NAME: | GSTN |
| LANGUAGE: | 'C' |
| COMPILER: | MICROSOFT C VERSION 5.0 |
| COMPILE OPTIONS: | /AL /FPi87 |
| SYSTEM REQUIREMENTS | PCL-714 Data Acquisition Card<br>CGA Monitor |
| PCL-714 channels used: | 4 - Digital Input<br>16 - Digital Output |

Data file storage uses a prefix of up to 6 alpha-numeric characters, an incrementing number in the range 0-99 and the extension ".GSN".

The programme saves the current operating conditions in the options file GSTN.SET, which if present will be reloaded at the next running of the programme.

## 2.3 Operation

Locate the file "GSTN.EXE" in a suitable directory of the computer, and set the 'path' such that it can be accessed. Change to the directory in which data is to be stored, and type "GSTN" at the prompt.

Figure 1 shows the screen layout on entry to this programme. There are five options listed: (1) Setup, (2) Begin Logging, (3) Check Time, (4) Change Time and (5) QUIT. These are described briefly below:

S. Setup

By pressing the 'S' key, you will enter the 'SETUP' section of the programme, in which you will be prompted for a response to the following (as shown in Figure 2):

Q1: Enter the prefix for the storage files.
The prefix referred to is an alpha-numeric string which will be placed at the beginning of the file name. It can be up to 6 characters long

Q2: Enter the file sequencer for the next run.
The file sequencer is a number which comes after the prefix in the file name.

L. Begin Logging

To start monitoring the status of each laser press the 'L' key. The number of pulses received will be displayed on the video monitor during the run. Each time the laser is fired, a black square, will be displayed before the laser's name. The number of pulses fired will be displayed after its name. Also the time, date, and the storage file name for this run will displayed on the screen.

To finish monitoring press any key. The computer will then display the results for this run as shown in Fig 3. Pressing another key will return you to the main menu.

T. Check Time

Pressing the 'T' key will show the current computer time, which can be checked against an external master clock. (See next option if time requires changing.)

C. Change Time

Should the computer time need changing, pressing the 'C' key will run the DOS programme "TIME", to allow you to do so.

Q. Quit

To exit the programme press the 'Q' key. This will save the current setup details on disk so that next time you run the programme you will begin with the same parameters.

## 2.4 Technical Information

The programme was written to monitor the TTL outputs from several truth detectors, and record the total number of pulses emitted by the laser during the run. For BRIS, the total time of illumination is also measured, since this is related to the operational characteristics of the emitter.

Figure 4 is a flow diagram of this programme, and Appendix I is a listing.

### 2.4.1 Digital Input

A TTL level '1' is used to indicate when the laser is ON, and a level '0' indicates the laser is OFF. The TTL outputs from the truth detectors are wired to inputs 1 to 4 of the PCL-714 Digital I/O card.

These inputs are sampled at a rate of 50 Hz by setting up the Intel 8253 timer on the PCL-714 card as a square wave generator of 25 Hz. The programme reads the timer's count continuously until a change in the TTL level occurs.

The digital input is then scanned and all inputs not relating to the truth detectors are set to '0'. The current input status is then compared to the previous scanned status. If a change in one of the inputs has occurred, the video output and related counter are then updated.

### 2.4.2 Video Insertion

Video insertion was performed using a Matrox card which requires a 16-bit digital input defining both the position and character to be inserted on the video screen. This 16-bit digital input was obtained from the digital output of the PCL-714 card. The PCL-714 was modified slightly to additionally provide a strobe to the Matrox card to indicate when data has been downloaded to its input.

The most significant 7 bits, of the 16-bit word, correspond to the character to be inserted The next 9 bits correspond to the video address.

To clear the video screen on start up, a space is inserted at every address on the screen.

```
                LASER WARNING RECEIVER TRIAL
               GROUND STATION LOGGING SYSTEM.
               ------------------------------

SELECT REQUIRED OPTION FROM BELOW
S --> Setup
L --> Begin Logging          Output File Name: delmeNN.GSN
T --> Check Time             Next File No. in Sequence (NN): 1
C --> Change Time
Q --> Quit
```

**Figure 1** Main Menu.

```
                LASER WARNING RECEIVER TRIAL
               GROUND STATION LOGGING SYSTEM.
               ------------------------------

SELECT REQUIRED OPTION FROM BELOW
S --> Setup
L --> Begin Logging          Output File Name: delmeNN.GSN
T --> Check Time             Next File No. in Sequence (NN): 1
C --> Change Time
Q --> Quit
Enter the prefix for the storage files:  delme
Enter the file sequencer for the next run:  1
```

**Figure 2** Setup Menu.

```
                LASER WARNING RECEIVER TRIAL
               GROUND STATION LOGGING SYSTEM.
               ------------------------------

SELECT REQUIRED OPTION FROM BELOW
S --> Setup
L --> Begin Logging          Output File Name: delmeNN.GSN
T --> Check Time             Next File No. in Sequence (NN): 1
C --> Change Time
Q --> Quit
PRESS ANY KEY TO STOP LOGGING !
LTDP   -  10 pulses
LTDB   -   1 pulses
BRIS   -   1 pulses   Time on:  23.12 seconds
SIMRAD -   0 pulses
PRESS ANY KEY TO CONTINUE !
```

**Figure 3** Logging Display.

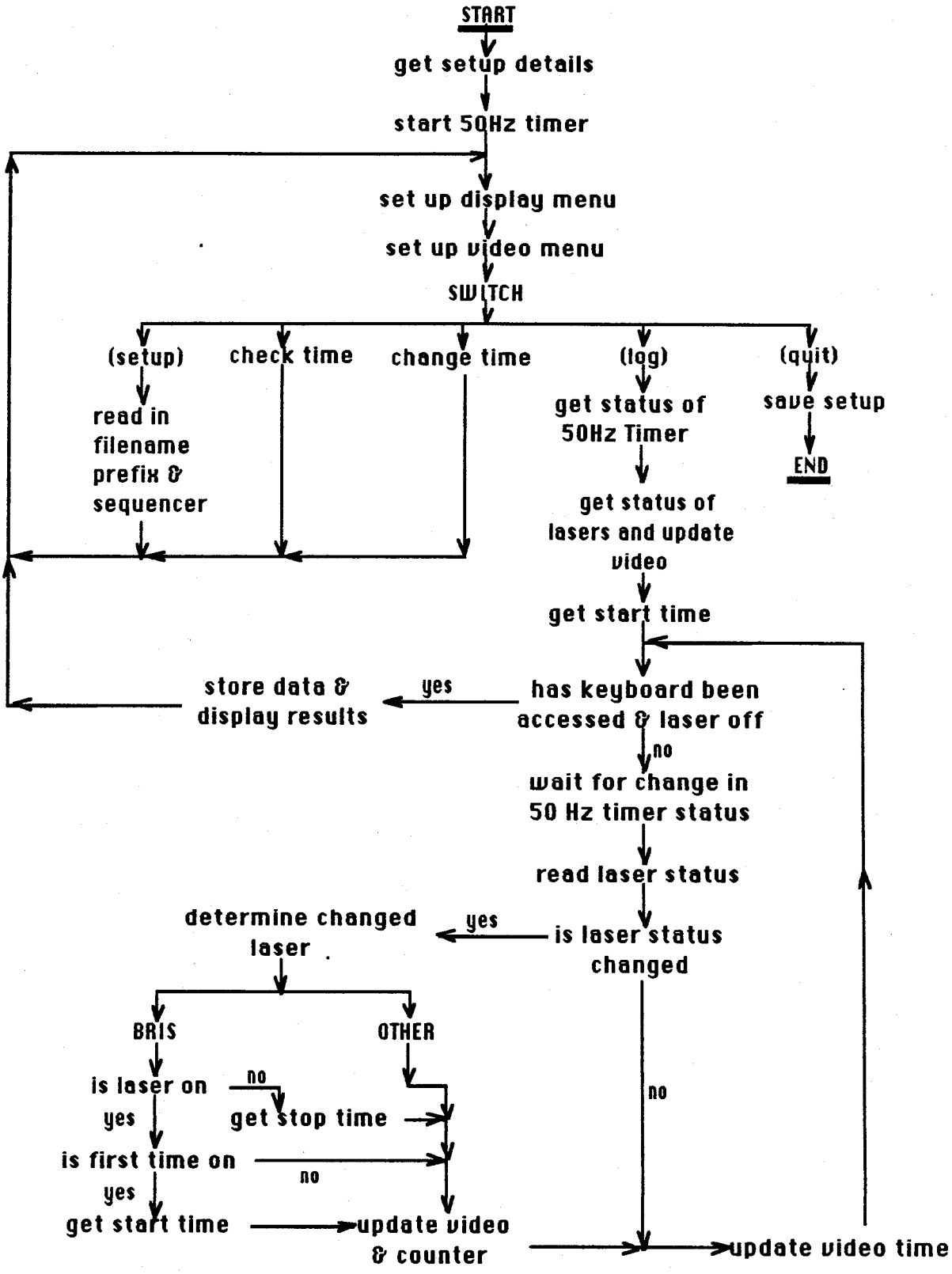


Figure 4 Flow Diagram for Ground Station Logging Programme.

# 3 AIRBORNE LOGGING PROGRAMME - LWREC

## 3.1 Function

This programme has been written to process data from the serial output from the Laser Warning Receiver's (LWR) ECM port via an RS422 Serial Cable. The LWR was mounted on a helicopter, and the computer mounted on a pallet in the helicopter. The data sent from the ECM port relates to the number of laser pulses received by the LWR. The data is stored and processed to determine the number of incoming laser pulses/bursts. These are displayed on the screen while logging and later stored in a file.

The RS422 interface card is a QUA TECH SCB-1040, configured to take in an external receive clock of 400 kHz generated by the LWR.

The data from each run was stored in memory until it was safe to store it on disk. This allows for a maximum of 50 firings to be stored in memory before saving the data.

The ability to change the computer's time was also incorporated in the programme such that it could be checked and rectified if necessary to match an external master clock.

## 3.2 Programme Specifications

| | |
|---|---|
| NAME: | LWREC |
| LANGUAGE: | 'C' |
| COMPILER: | MICROSOFT C VERSION 5.0 |
| COMPILE OPTIONS: | /AL /FPi87 |
| SYSTEM REQUIREMENTS: | SCB-1040 RS422 Card<br>CGA Monitor |

Data file storage uses a prefix of up to 6 alphanumeric characters, an incrementing number in the range 0-99 and the extension "**.HEL**".

The programme saves the current operating conditions in the options file LWR.SET, which if present will be reloaded at the next running of the programme.

## 3.3. Operation

Locate the file "**LWREC.EXE**" in a suitable directory of the computer, and set the 'path' such that it can be accessed. Change to the directory in which the data is to be stored, and type "**LWREC**" at the prompt.

Figure 5 shows the screen layout on entry to this programme. There are six options listed: (1) Setup, (2) Receive, (3) Save Data, (4) Reset, (5) Check/Change Time and (6) QUIT. These are described briefly below:

A. SETUP

By pressing the '**A**' key, you will enter the 'SETUP' section of the programme, in which you will be prompted for a response to the following (as shown in Figure 6):

Q1: Enter the prefix for the storage file.
The prefix referred to is an alpha-numeric string which will be placed at the beginning of the file name. It can be up to 6 characters long.

Q2: Enter the file sequencer for the first run.
The file sequencer is a number which comes after the prefix in the file name. Note that this is the sequencer for the first run, and that if there are any runs stored in memory, this sequencer relates to the first of these runs.

Q3: Enter the maximum data length for the data files.
The maximum data length has been set as variable in order to allow the maximum usage of computer memory, before saving is necessary. Ensure that the length is sufficient for the runs being made. If for any reason the length is not sufficient a warning message will be displayed during logging.

Q4: Select Required Emitter Type.
Emitter selection is made by typing in the number next to it in the list. The 'General' option should only be used if more than 1 emitter type is being simultaneously used.

After completion of the setup section check that your entries have been translated to the upper half of the screen.

B. RECEIVE

By typing the letter 'B' entry will be made to the receive section of the programme, and any data appearing on the RS422 lines will be stored in memory. Figure 7 shows the screen layout for this phase.

'No. of Recorded Hits' relates to the number of bytes received over the line, and 'No. of Recorded Bursts' relates to the number of detected bursts emitted by the laser. The screen will update during the receive section until another key is pressed, after which the results will remain on screen until a further key is pressed.

C. SAVE DATA

To save the data currently in memory press the 'C' key on the computer. You will be asked the question "Is it safe to save data now?" as shown in Figure 8. This question is asked as the hard disk should not be accessed unless the helicopter is in a stable condition. Press 'Y' to continue or 'N' to exit this part of the programme.

D. RESET

By pressing the letter 'D' the programme will:

1. Reset the run counter to 0.
2. Free any memory used for data, and
3. Reset any other variables used in the programme for status.

If you currently have any unsaved data you will be asked whether you wish to continue.

E. Check/Change Time

By pressing the 'E' key you can check that the computers timer is synchronised to an external clock or a wrist watch. If update is not necessary, just press the carriage return key.

To update the time, you must enter the time exactly in the format given, viz. two numbers for each of hour, minute and second, separated by a colon. Any other format will not be accepted, and the time will be unchanged. (See Figure 9).

Q. QUIT

To exit the programme press the 'Q' key. If there is unsaved data you will be asked whether you wish to continue.

You will also be asked whether it is safe to save the 'Setup File' on disk. Saving will allow you to re-enter the programme with the same setup as previously. Do not attempt to save the file if the helicopter is not in a suitable condition, as damage to the hard disk may result.

## 3.4 Technical Information

This programme was written to monitor the data lines of the ECM port of the Laser Warning Receiver. These data lines conform to the RS422 standard. A QUATECH SCB-1040 RS422 communications board is used to receive this data. Figure 10 shows the required jumper selections for the board, to allow for an external receive clock running at 16 times the required Baud rate.

Figure 11 is a flow diagram of this programme, and Appendix II is a listing.

### 3.4.1 Communications Handling

The programme will configure the SCB-1040 board for asynchronous communications, such that an interrupt will be enabled upon receiving a byte at the input. The interrupt handler is determined by the emitter being monitored.

For the LTD and SIMRAD lasers, the interrupt handler will read the byte from the data port and transfer it into the data array. It will then check the computers programme timer and compare this time to the previous accessed time to determine if more than 0.25 seconds has passed. If it has, then a new burst has commenced, and the counters are updated accordingly.

For BRIS, the programme will initiate the Intel 8254 timer for Mode 0 (Interrupt on terminal count), which will set the output low until the count is completed. A count of 3 is used in initialisation such that the counter will be high before the first byte is received.

During the interrupt, the data byte is transferred from the data port to the data array, then the output status of the timer is checked. If the output is high then a new burst has commenced, otherwise it is still part of the previous burst. The counters are updated accordingly. The 8254 timer is then reloaded with a count of 25000, which for a 5 MHz counter corresponds to a time of 5 ms.

### 3.4.2 Memory Management

As it may not always be safe to store data to hard disk while airborne (due to vibrations), the programme was written to save the data in the computers memory until it is deemed safe to store on disk. This is achieved by creating several arrays, which will contain the relevant information relating to each run. One of these arrays is an array of pointer to the data received. Prior to each run the computer will allocate memory for this array, based on the maximum size given in the setup. To ensure that this space is not exceeded the programme checks the number of bytes received and if it exceeds this limit, the communications port is disabled, and the operator is warned.

Up to 50 runs can be kept in memory if required, prior to saving data on disk. If memory is not available, the operator is advised and he should then save his current data before proceeding. After saving the data, memory is freed for further use .

```
                LASER WARNING RECEIVER
              R2422 MONITORING PROGRAM
              ------------------------

SELECT REQUIRED OPTION FROM BELOW

A.  SETUP                 Store Filename:  delNN.HEL

B.  RECEIVE               First File No. in Sequence (NN):  4

C.  SAVE DATA             Maximum Data Length:  100

D.  RESET                 No. of Runs in Memory:  0

E.  CHECK/CHANGE TIME Current Emitter Selection:  LTD

Q.  QUIT
```

**Figure 5** Main Menu.

```
                LASER WARNING RECEIVER
              R2422 MONITORING PROGRAM
              ------------------------

SELECT REQUIRED OPTION FROM BELOW

A.  SETUP                 Store Filename:  delNN.HEL

B.  RECEIVE               First File No. in Sequence (NN):  4

C.  SAVE DATA             Maximum Data Length:  100

D.  RESET                 No. of Runs in Memory:  0

E.  CHECK/CHANGE TIME     Current Emitter Selection:  LTD

Q.  QUIT

Enter the prefix for the storage files:  del
Enter the file sequencer for the first run:  1
Enter the maximum data length for the data files:  1000
Select Required Emitter Type:
   General(0), SIMRAD(1), LTD(2), BRIS(3) :  3
```

**Figure 6** Setup Menu.

```
                    LASER WARNING RECEIVER
                  R2422 MONITORING PROGRAM
                  ------------------------

SELECT REQUIRED OPTION FROM BELOW

A.    SETUP                 Store Filename:  delNN.HEL

B.    RECEIVE               First File No. in Sequence (NN):  4

C.    SAVE DATA             Maximum Data Length:  100

D.    RESET                 No. of Runs in Memory:  0

E.    CHECK/CHANGE TIME     Current Emitter Selection:  LTD

Q.    QUIT

Press Any Key to Stop Monitoring
No. of Recorded Hits:  43
No. of Recorded Bursts:  4
PRESS ANY KEY TO CONTINUE
```

**Figure 7** Receive Display.

```
                    LASER WARNING RECEIVER
                  R2422 MONITORING PROGRAM
                  ------------------------

SELECT REQUIRED OPTION FROM BELOW

A.    SETUP                 Store Filename:  delNN.HEL

B.    RECEIVE               First File No. in Sequence (NN):  4

C.    SAVE DATA             Maximum Data Length:  100

D.    RESET                 No. of Runs in Memory:  0

E.    CHECK/CHANGE TIME     Current Emitter Selection:  LTD

Q.    QUIT

IS IT SAFE TO SAVE DATA NOW (Y/N)?
```

**Figure 8** Save Menu.

```
                LASER WARNING RECEIVER
              R2422 MONITORING PROGRAM
              ------------------------

SELECT REQUIRED OPTION FROM BELOW

A.    SETUP                  Store Filename:  delNN.HEL

B.    RECEIVE                First File No. in Sequence (NN):  4

C.    SAVE DATA              Maximum Data Length:  100

D.    RESET                  No. of Runs in Memory:  0

E.    CHECK/CHANGE TIME      Current Emitter Selection:  LTD

Q.    QUIT

current time is:  15:37:44.46

Enter New Time (HH:MM:SS):  15:39:00
```

**Figure 9** Time Check/Change Menu.

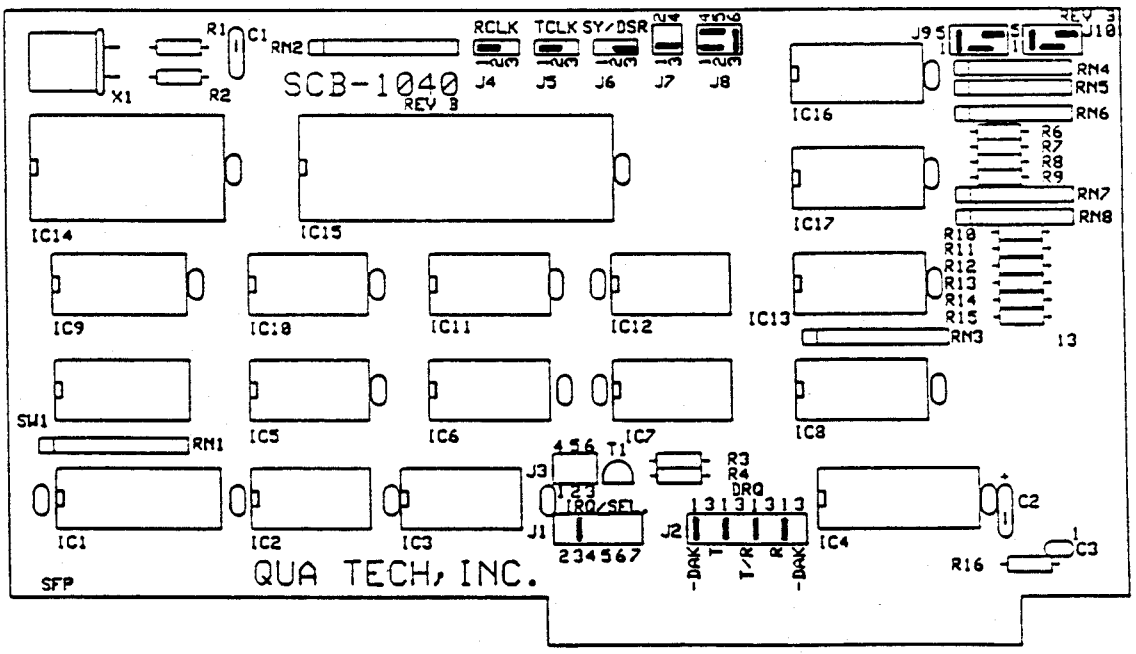


**Figure 10** QUATECH SCB-1040 Board Layout and Jumper Selection.

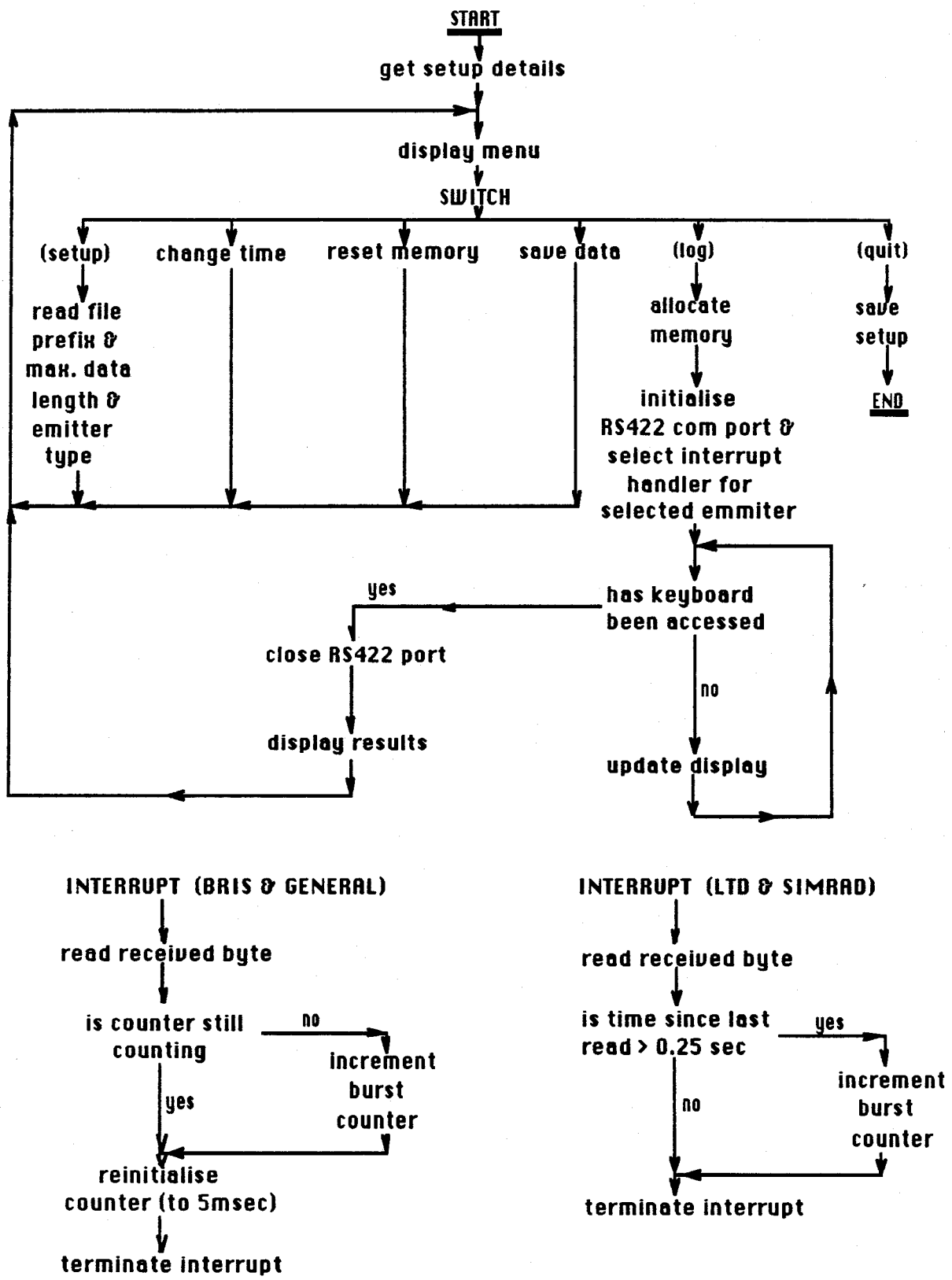


Figure 11 Flow Diagram for Airborne Logging Programme.

## ACKNOWLEDGEMENTS

The author wishes to acknowledge the assistance given by Dr. S.A. Brunker and Mr. A. Buttignol in the preparation of the ancillary equipment and methods used for these logging systems.

## REFERENCES

Brunker, S.A. (Jan 1990) *Evaluation of The AN/AVR-2 Laser Warning Receiver.* Trial Plan for Defence Trial No 7/423.

Brunker, S.A., Grant,K.J., Rossiter, R.J., and Oermann, R.J. (May 1990) *Evaluation of the Laser Warning Receiver AN/AVR-2 For Defence Trial 7/423 Trial Name "THUNDERSTOP".* Report to Directorate of Trials.

# APPENDIX I

## PROGRAMME LISTING OF GSTN.C

```
#include <stdio.h>
#include <graph.h>
#include <dos.h>

char ch,file_pre[7],outfilename[13],curr_time[15];
int i,prt=0x220,fileincr,las_stat,las_stata,lpul[8],file_no;
int br_notstrtd=1;
float br_ontime;
struct dostime_t srt,crt,bris_strt,bris_stp;

main()
{
  int endlog = 0;

  _setvideomode(_HRESBW);
  get_setup();              /* Get setup details */
  set_up_timer();           /* set up 50 Hz timer */

  while(!endlog){
    sprintf(outfilename,"%s%d.gsn",file_pre,file_no);
    set_up_menu();          /* Set up menu area  */
    set_up_video();         /* Set up video output */

    while(!kbhit()) upd_time();
    ch = toupper(getch());
    switch(ch){
      case 'S' :  setup();       /* set up options */
      break;
      case 'L' :  log();         /* log status */
      break;
      case 'T' :  check_time(); /* check time */
      break;
      case 'C' :  change_time();/* change time */
      break;
      case 'Q' :  endlog = 1;    /* exit */
      default  :  ;
    }
  }
  store_setup();                /* save setup */
  _setvideomode(_DEFAULTMODE);  /* reset video mode */
}


get_setup()
{
  FILE *setfile;

  if((setfile=fopen("gstn.set","r"))==NULL){
    sprintf(file_pre,"delme");
    file_no = 1;
  }
```

```
  else {
    fscanf(setfile,"%s %d",file_pre,&file_no);
    close(setfile);
  } }


set_up_menu()
{
/*displays the main menu and selections */

  char buffer[80];

  _clearscreen(_GCLEARSCREEN);

  write_text(1,20, "LASER WARNING RECEIVER TRIAL");
  write_text(2,19,"GROUND STATION LOGGING SYSTEM.");
  write_text(3,19,"------------------------------");

  write_text(5,1,"SELECT REQUIRED OPTION FROM BELOW\n\n");
  _outtext("S  -->  Setup\n\n");
  _outtext("L  -->  Begin Logging\n\n");
  _outtext("T  -->  Check Time\n\n");
  _outtext("C  -->  Change Time\n\n");
  _outtext("Q  -->  QUIT\n\n");

  sprintf(buffer,"Output File Name: %sNN.GSN",file_pre);
  write_text(9,40,buffer);
  sprintf(buffer,"Next File No. in Sequence (NN): %d",file_no);
  write_text(11,40,buffer);
  _settextposition(15,1);
}


set_up_video()
{
  struct dosdate_t cdate;
  char buffer[32];

  clr_vdo();
  vdo_write(3,12,"LTDP");
  vdo_write(3,13,"LTDB");
  vdo_write(3,14,"BRIS");
  vdo_write(3,15,"SIMRAD");
  vdo_write(19,15,outfilename);

  _dos_getdate(&cdate);
  sprintf(buffer,"%2d-%2d-%4d",cdate.day,cdate.month,cdate.year);
  vdo_write(2,1,buffer);
  upd_time();
}


setup()
{
  write_text(17,1,"Enter the prefix for the storage files:  ");
  scanf("%s",file_pre);
  write_text(18,1,"Enter the file sequencer for the next run:  ");
```

```
  scanf("%d",&file_no);
}

store_setup()
{
  FILE *setfp;

  if((setfp = fopen("gstn.set","w")) == NULL) return(1);
  fprintf(setfp,"%s %d",file_pre,file_no);
  fclose(setfp);
}


log()
{
  int stat,stata,i,j;
  char buffer[80];

  outp(prt+3,0x40);
  stat = ((inp(prt+1) + 256*inp(prt+1)) > 100);
  for(i=0;i<4;i++) lpul[i] = 0;
  las_stat = 0;
  br_notstrtd = 1;
  bris_strt = bris_stp;
  las_stata = (inp(prt+6) & 0xF);
  upd_stat();
  las_stat = las_stata;

  _dos_gettime(&srt);
  write_text(17,1,"PRESS ANY KEY TO STOP LOGGING !");

  while(!kbhit() || (las_stat != 0)){
    outp(prt+3,0x40);
    while((stata=((inp(prt+1)+256*inp(prt+1)) >100)) == stat)
      outp(prt+3,0x40);
    stat = stata;
    las_stata = inp(prt+6) & 0xF;
    if(las_stat != las_stata){
      upd_stat();
      las_stat = las_stata;
    }
    upd_time();
  }
  getch();
  store_data();
  write_text(23,1,"PRESS ANY KEY TO CONTINUE !");
  while(!kbhit());
  getch();
}


check_time()
{
  struct dostime_t ctime;

  _dos_gettime(&ctime);
  sprintf(curr_time,"Current time is:  %02d:%02d:%02d.%02d",
```

```
          ctime.hour,ctime.minute,ctime.second,ctime.hsecond);
  write_text(17,1,curr_time);
  write_text(19,1,"Press any key to continue");
  while(!kbhit());
  getch();
}


change_time()
{
  system("time");
}


upd_stat()
{
  char buffer[80];

  if((las_stat & 1) != (las_stata & 1)){
    if((las_stata & 1) == 1){
      lpul[0] += 1;
      vdo_out(482,127);
      itoa(lpul[0],buffer,10);
      vdo_write(10,15,buffer);
    }
    else vdo_out(482,0x20);
  }

  if((las_stat & 2) != (las_stata & 2)){
    if((las_stata & 2) == 2){
      lpul[1] += 1;
      vdo_out(386,127);
      itoa(lpul[1],buffer,10);
      vdo_write(8,12,buffer);
    }
    else vdo_out(386,0x20);
  }

  if((las_stat & 4) != (las_stata & 4)){
    if((las_stata & 4) == 4){
      lpul[2] += 1;
      vdo_out(418,127);
      itoa(lpul[2],buffer,10);
      vdo_write(8,13,buffer);
    }
    else vdo_out(418,0x20);
  }

  if((las_stat & 8) != (las_stata & 8)){
    if((las_stata & 8) == 8){
      if(br_notstrtd) _dos_gettime(&bris_strt);
      br_notstrtd = 0;
      lpul[3] += 1;
      vdo_out(450,127);
      itoa(lpul[3],buffer,10);
      vdo_write(8,14,buffer);
    }
```

```
    else vdo_out(450,0x20);
    _dos_gettime(&bris_stp);
  }
}

upd_time()
{
  _dos_gettime(&crt);
  sprintf(curr_time,"%02d|%02d|%02d.%02d",crt.hour,crt.minute,
    crt.second,crt.hsecond);
  vdo_write(20,1,curr_time);
}


store_data()
{
  FILE *outfp;
  char buffer[80];

  if((outfp = fopen(outfilename,"w")) == NULL){
    write_text(19,1,"UNABLE TO OPEN OUTPUT FILE !!!!!!");
    return(1);
  }

  fprintf(outfp,"Start Time: %02d:%02d:%02d.%02d\n",srt.hour,
         srt.minute,srt.second,srt.hsecond);
  fprintf(outfp,"Stop Time:  %02d:%02d:%02d.%02d\n",crt.hour,
         crt.minute,crt.second,crt.hsecond);

  br_ontime = (float)((long)bris_stp.hour  * 360000 +
          (long)bris_stp.minute      * 6000   +
          (long)bris_stp.second      * 100    +
          (long)bris_stp.hsecond)    / 100    -
       ((float)((long)bris_strt.hour    * 360000 +
          (long)bris_strt.minute     * 6000   +
          (long)bris_strt.second     * 100    +
          (long)bris_strt.hsecond)   / 100);

  sprintf(buffer,"LTDP   - %3d pulses",lpul[1]);
  write_text(18,1,buffer);
  fprintf(outfp,"%s\n",buffer);
  sprintf(buffer,"LTDB   - %3d pulses",lpul[2]);
  write_text(19,1,buffer);
  fprintf(outfp,"%s\n",buffer);
  sprintf(buffer,"BRIS   - %3d pulses   Time on: %5.2f seconds",
    lpul[3],br_ontime);
  write_text(20,1,buffer);
  fprintf(outfp,"%s\n",buffer);
  sprintf(buffer,"SIMRAD - %3d pulses",lpul[0]);
  write_text(21,1,buffer);
  fprintf(outfp,"%s\n",buffer);
  fclose(outfp);
  if(++file_no > 99)  file_no = 0;
}


set_up_timer()
```

```
{
/* sets up counter 1 and 2 as a 50 Hz timer */

  outp(prt+3,0x76);
  outp(prt+1,0xC8);    outp(prt+1,0x00);
  outp(prt+3,0xB6);
  outp(prt+2,0x20);
  outp(prt+2,0x03);
}


clr_vdo()
{
  int i,c;

  for(i=0;i<0x200;i++){
    c = 0x20;
    if (i > 0xFF)   c = ((c<<1)+1);
    else            c = (c<<1);

    outp(prt+13, (char)(i & 0xFF));
    outp(prt+14, (char) c);
  }
}


vdo_out(unsigned add, char c)
{
  if (add > 0x0FF)   c = ((c<<1) + 1);
  else               c = (c<<1);

  outp(prt+13, (char)(add & 0xFF));
  outp(prt+14, c);
}


vdo_write(int x, int y, char *s)
{
  int add, index, lgth,i;

  add = y*32 + x;
  lgth = strlen(s);
  for(index = 0;index<lgth;index++){
    vdo_out(add++,s[index]);
/*  for(i=0;i<10;i++);    */     /* delay for AT's */
  }
  vdo_out(add,0x20);
}


write_text(int a, int b, char *s)
{
  _settextposition(a,b);
  _outtext(s);
}
```

# APPENDIX II

## PROGRAMME LISTING OF LWREC.C

```
#include <stdio.h>
#include <graph.h>
#include <dos.h>
#include <malloc.h>
#include <time.h>

void open_com();
void close_com();
void far *oldfunc;

void interrupt far com_isra();
void interrupt far com_isrb();

int baud_rate=9600,com_base=0x300,com_flag=0,intlev;
char dummy[256],selection,*input[50],file_pre[7],filename[13],
     emitter_name[10];
int rec_length=0,file_no,data_length,endsession=0,no_runs=0,
     inp_lngth[50];
int data_saved = 1,emitter_flag=0,burst_counter[50],em_flag[50];
char *buffer_in;

struct dostime_t str_time[50],stp_time[50];
clock_t t1,t2;


main()
{
  get_setup();
  system("park");
  _setvideomode(_HRESBW);

  t1 = clock();

  while(!endsession){

    display_menu();

    while(!kbhit());

    selection=toupper(getch());
    switch(selection){

      case 'A' : setup();
     break;
      case 'B' : receive();
     break;
      case 'C' : save_data();
     break;
      case 'D' : rst_mem();
     break;
      case 'E' : ch_time();
```

```
      break;
       case 'Q' : if(shutdown()) endsession = 1;
        default  : ;
      }
   }
   _clearscreen(_GCLEARSCREEN);
   _setvideomode(_DEFAULTMODE);
}


get_setup()
{
  FILE *setfile;

  if((setfile=fopen("lwr.set","r"))==NULL){
    sprintf(file_pre,"delme");
    file_no = 1;
    data_length = 8192;
    emitter_flag = 0;
  }
  else {
    fscanf(setfile,"%s %d %d %d",file_pre,&file_no,&data_length,
          &emitter_flag);
    close(setfile);
  }
  set_emitter_type(emitter_flag,emitter_name);
}

set_emitter_type(int flag, char *name)
{
  if(flag == 0)      sprintf(name,"GENERAL");
  else{
    if(flag == 1)    sprintf(name,"SIMRAD");
    else{
      if(flag == 2) sprintf(name,"LTD");
      else{
     flag = 3;    sprintf(name,"BRIS");
      }
    }
  }
}


display_menu()
{
  _clearscreen(_GCLEARSCREEN);

  write_text(1,20,"LASER WARNING RECEIVER");
  write_text(2,19,"RS422 MONITORING PROGRAM");
  write_text(3,19,"-----------------------");

  write_text(5,1,"SELECT REQUIRED OPTION FROM BELOW");

  write_text(7,1,"A.    SETUP");
  write_text(9,1,"B.    RECEIVE");
  write_text(11,1,"C.    SAVE DATA");
  write_text(13,1,"D.    RESET");
```

```
  write_text(15,1,"E.    CHECK/CHANGE TIME");
  write_text(17,1,"Q.    QUIT");

  sprintf(dummy,"Store Filename:  %sNN.HEL",file_pre);
  write_text(7,40,dummy);
  sprintf(dummy,"First File No. in Sequence (NN):  %d",file_no);
  write_text(9,40,dummy);
  sprintf(dummy,"Maximum Data Length:  %d",data_length);
  write_text(11,40,dummy);
  sprintf(dummy,"No. of Runs in Memory:  %d",no_runs);
  write_text(13,40,dummy);
  sprintf(dummy,"Current Emitter Selection:  %s",emitter_name);
  write_text(15,40,dummy);
  _settextposition(19,1);
}

setup()
{
  write_text(19,1,"Enter the prefix for the storage files:  ");
  scanf("%s",file_pre);
  write_text(20,1,"Enter the file sequencer for the first
           run:  ");
  scanf("%d",&file_no);
  write_text(21,1,"Enter the maximum data length for the data
              files:  ");
  scanf("%d",&data_length);
  write_text(22,1,"Select Required Emitter Type: ");
  write_text(23,1,"   General(0), SIMRAD(1), LTD(2),
              BRIS(3) :  ");
  scanf("%d",&emitter_flag);
  set_emitter_type(emitter_flag,emitter_name);
}

receive()
{
  char ch,buffer[80];

  if(no_runs == 50){
    write_text(19,1,"OUT OF MEMORY - PLEASE SAVE DATA BEFORE
                PROCEDING");
    write_text(21,1,"Press any key to continue");
    while(!kbhit());
    ch=getch();
    return(1);
  }
  if((input[no_runs] = (char *)malloc(data_length+200))==NULL){
    printf("MEMORY ALLOCATION FAILED\n\n
              SAVE CURRENT DATA BEFORE PROCEDING");
    printf("\nPress any key to continue");
    while(!kbhit());
    ch=getch();
    return(1);
  }

  buffer_in = input[no_runs];
  rec_length = 0;
  burst_counter[no_runs] = 0;
```

```
  _dos_gettime(&str_time[no_runs]);

  open_com();

  if(com_flag != 1){
    write_text(19,1,"Communications port did not initialise");
    write_text(20,1,"PLEASE CHECK CONFIGURATION BEFORE");
    write_text(21,1,"            PROCEDING");
    write_text(23,1,"Press any key to continue:");
    while(!kbhit());
    ch = getch();
    return(3);
  }

  write_text(19,1,"Press Any Key to Stop Monitoring");
  while((!kbhit()) && (rec_length < data_length)){
    sprintf(buffer,"No. of Recorded Hits:  %d\n",rec_length);
    write_text(20,1,buffer);
    sprintf(buffer,"No. of Recorded Bursts:  %d\n",
              burst_counter[no_runs]);
    write_text(21,1,buffer);
  }
  close_com();

  _dos_gettime(&stp_time[no_runs]);

  if(rec_length >= data_length){
    rec_length = data_length;
    write_text(21,1,"Reached Maximum set data length prior to
               key being hit");
    write_text(22,1,"Press any key to continue");
  }
  ch = getch();
  inp_lngth[no_runs] = rec_length;
  em_flag[no_runs] = emitter_flag;

  data_saved = 0;
  sprintf(buffer,"No. of Recorded Hits:  %d\n",rec_length);
  write_text(20,1,buffer);
  sprintf(buffer,"No. of Recorded Bursts:  %d\n",
              burst_counter[no_runs]);
  write_text(21,1,buffer);
  write_text(22,1,"PRESS ANY KEY TO CONTINUE");
  while(!kbhit());
  getch();
  if(++no_runs >= 50){
    write_text(19,1,"Maximum number of records currently stored");
    write_text(20,1,"in memory.  Please save before proceding:");
    write_text(21,1,"PRESS ANY KEY TO CONTINUE");
    while(!kbhit());
    ch = getch();
  }
}

save_data()
{
```

```
    int i,j;
    FILE *outfp;
    char *outpt,ch,em_name[10];

    write_text(19,1,"IS IT SAFE TO SAVE DATA NOW (Y/N)? ");
    while(!kbhit());
    if((ch=toupper(getch())) != 'Y') return(2);

    for(i=0;i<no_runs;i++){
      j = file_no +i;      sprintf(filename,"%s%d.HEL",file_pre,j);
      if((outfp=fopen(filename,"wb"))==NULL){
        printf("Unable to open output file\n\nPress any key to
               continue");
        while(!kbhit());
        ch=getch();
        return(1);
      }
      set_emitter_type(em_flag[i],em_name);
      fprintf(outfp,"Emitter:  %s\n",em_name);
      fprintf(outfp,"Start Time:  %02d:%02d:%02d.%02d\n",
            str_time[i].hour,str_time[i].minute,str_time[i].second,
            str_time[i].hsecond);
      fprintf(outfp,"Stop Time:   %02d:%02d:%02d.%02d\n",
            stp_time[i].hour,stp_time[i].minute,stp_time[i].second,
            stp_time[i].hsecond);
      fprintf(outfp,"Data Length: %d\n",inp_lngth[i]);
      fprintf(outfp,"No. of Bursts: %d\n",burst_counter[i]);
      outpt = input[i];
      fwrite(outpt,1,inp_lngth[i],outfp);
      fclose(outfp);
      free(input[i]);
      inp_lngth[i] = 0;
    }
    file_no = file_no + no_runs;
    no_runs=0;

    data_saved = 1;
    system("park");
}


rst_mem()
{
    char ch;
    int i;

    if(data_saved != 1){
      write_text(19,1,"There is unsaved data in memory");
      write_text(21,1,"Do you wish to continue (Y/N):  ");
      while(!kbhit());
      if((ch = toupper(getch())) != 'Y') return(1);
    }

    for (i=0;i<no_runs;i++){
      free(input[i]);
      inp_lngth[i] = 0;
      burst_counter[i] = 0;
```

```
    }
    no_runs = 0;
    data_saved = 1;
}


ch_time()
{
    char  input[20];
    int  i;
    struct dostime_t  ot, nt;

    _dos_gettime(&ot);
    printf("current time is:  %02d:%02d:%02d.%02d\n\n",ot.hour,
                    ot.minute,ot.second,ot.hsecond);

    printf("Enter New Time (HH:MM:SS):  ");
    for(i=0;(input[i] = getchar()) != '\n';i++);
    input[i] = '\0';
    if (strlen(input) != 8)  return(-1);

    input[2] = '\0';
    input[5] = '\0';

    nt.hour   = (unsigned char) atoi(input);
    nt.minute = (unsigned char) atoi(&input[3]);
    nt.second = (unsigned char) atoi(&input[6]);
    nt.hsecond = 0;

    _dos_settime(&nt);
}

shutdown()
{
    char ch;
    FILE *setfp;

    if(data_saved != 1){
      write_text(19,1,"There is unsaved data in memory");
      write_text(21,1,"Do yo wish to continue (Y/N):  ");
      while(!kbhit());
      if((ch=toupper(getch())) != 'Y') return(0);
    }
    write_text(21,1,"Is it safe to store setup file to
              disc (Y/N)?");
    while(!kbhit());
    if((ch = toupper(getch())) != 'Y') return(1);

    if((setfp=fopen("lwr.set","w"))==NULL){
      system("park");
      return(1);
    }

    fprintf(setfp,"%s %d %d %d",file_pre,file_no,data_length,
                emitter_flag);
    fclose(setfp);
    system("park");
```

```
  return(1);
}


void open_com()
{
  int cntdiv,cnthi,cntlo;
  int ptemp,i;

  cntdiv = 5000000/16/baud_rate;
  cnthi = cntdiv/256;
  cntlo = cntdiv - (cnthi * 256);

  outp(com_base+3,0x36);     /* set up baud rate timers */
  outp(com_base,cntlo);
  outp(com_base,cnthi);    outp(com_base+3,0x76);
  outp(com_base+1,cntlo);
  outp(com_base+1,cnthi);

  outp(com_base+6,0x18);     /* reset Channels on SCB-1040 */
  outp(com_base+7,0x18);

  for(i=0;i<50;i++);         /* time delay */

  outp(com_base+6,4);        /* select control register 4 */
  outp(com_base+6,0x47);     /* 16*data rate = clock,even parity,
                                        1 stop bit */

  outp(com_base+6,3);        /* select control register 3 */
  outp(com_base+6,0xC1);     /* 8 data bits, enable Rx */

  outp(com_base+6,5);        /* select control register 5 */
  outp(com_base+6,0xE2);     /* 8 data bits, disable Tx */

  intlev = 0xB;              /* select interrupt level for INT 3 */
  oldfunc = _dos_getvect(intlev);
  if((emitter_flag == 1) || (emitter_flag == 2)){
    _dos_setvect(intlev,com_isra);
  }
  else {
    _dos_setvect(intlev,com_isrb);
    outp(com_base+3,0xB0);
    outp(com_base+2,3);
    outp(com_base+2,0);
  }
  _disable();

  ptemp = inp(0x21) & 0xF7;
  outp(0x21,ptemp);          /* unmask interrupt */

  outp(com_base+6,1);        /* select control register 1 */
  outp(com_base+6,0x10);     /* Enable Rx interrupt on SCB-1040 */

  outp(com_base+6,2);        /* select control register 2 */
  outp(com_base+6,0x00);     /* Enable Rx interrupt on SCB-1040 */
```

```
  _enable();
  com_flag = 1;
}


void close_com()
{
  int ptemp;

  if(com_flag = 1){
    _disable();
    ptemp = inp(0x21) | 0x08;
    outp(0x21,ptemp);
    outp(com_base+6,3);
    outp(com_base+6,0xC0);
    outp(com_base+6,1);
    outp(com_base+6,0);
    outp(com_base+6,0x38);      _dos_setvect(intlev,oldfunc);
    _enable();
    com_flag = 0;
  }
}


void interrupt far com_isra()
{
  if(com_flag == 1) buffer_in[rec_length++] = inp(com_base+4);
  if(((t2=clock()) - t1) > 250)  burst_counter[no_runs] += 1;
  t1 = t2;
  outp(com_base+6,0x38);       /* send EOI to 1040 */
  outp(0x20,0x20);             /* send EOI to int contr */
}

void interrupt far com_isrb()
{
  if(com_flag == 1) buffer_in[rec_length++] = inp(com_base+4);
  outp(com_base+3,0xE8);
  if((inp(com_base+2) & 0x80 ) == 0x80)
    burst_counter[no_runs] += 1;
  outp(com_base+3,0xB0);
  outp(com_base+2,168);
  outp(com_base+2,97);
  outp(com_base+6,0x38);       /* send EOI to 1040 */
  outp(0x20,0x20);             /* send EOI to int contr */
}


write_text(int a,int b,char output[256])
{
  _settextposition(a,b);
  _outtext(output);
}
```

THIS IS A BLANK PAGE

# DOCUMENT CONTROL DATA SHEET

Security classification of this page : UNCLASSIFIED

| 1 DOCUMENT NUMBERS | | 2 SECURITY CLASSIFICATION | |
|---|---|---|---|
| AR Number: | AR-006-453 | a. Complete Document: | UNCLASSIFIED |
| Series Number: | ERL-0532-GD | b. Title in Isolation | UNCLASSIFIED |
| Other Numbers: | | c. Summary in Isolation | UNCLASSIFIED |
| | | 3 DOWNGRADING / DELIMITING INSTRUCTIONS | |

**4 TITLE**

LOGGING PROGRAMMES FOR TRIAL 'THUNDERSTOP'

| 5 PERSONAL AUTHOR (S) | 6 DOCUMENT DATE |
|---|---|
| Robert J. Rossiter | SEPTEMBER 1990 |
| | 7 7.1 TOTAL NUMBER OF PAGES: 28 |
| | 7.2 NUMBER OF REFERENCES: 2 |

| 8 8.1 CORPORATE AUTHOR (S) | 9 REFERENCE NUMBERS | |
|---|---|---|
| Electronics Research Laboratory | a. Task: | AIR/89/099 |
| | b. Sponsoring Agency: | |
| 8.2 DOCUMENT SERIES and NUMBER | 10 COST CODE | |
| General Document 0523 | | |

| 11 IMPRINT (Publishing organisation) | 12 COMPUTER PROGRAM (S) (Title (s) and language (s)) |
|---|---|
| Defence Science and Technology Organisation Salisbury | |

**13 RELEASE LIMITATIONS** (of the document)

Approved for Public Release.

Security classification of this page : UNCLASSIFIED

Security classification of this page : UNCLASSIFIED

**14 ANNOUNCEMENT LIMITATIONS** (of the information on these pages)

No limitation

| **15 DESCRIPTORS** | | **16 COSATI CODES** |
|---|---|---|
| a. EJC Thesaurus Terms | C (Programming Language)<br>Computing Programs | 1205<br>0903 |
| b. Non - Thesaurus Terms | Laser warning receivers<br>Ground Station Logging Programme<br>Airborne Logging Programme | |

**17 SUMMARY OR ABSTRACT**
(if this is security classified, the announcement of this report will be similarly classified)

This document presents two computer programmes written for Defence Trial 7/423 code-name 'THUNDERSTOP'. The programmes are for the ground station and airborne logging systems.