U. S. DEPARTMENT OF THE INTERIOR

U. S. GEOLOGICAL SURVEY

DESCRIPTIONS OF SEISMIC ARRAY COMPONENTS: PART 2. SOFTWARE MODULES FOR DATA ACQUISITION/PROCESSING

Compiled by

W. H. K. Lee
MS 977, 345 Middlefield Road
Menlo Park, CA 94025

Open-File Report 92-597
August, 1992

INTRODUCTION

In the summer of 1990, funding was available to design and implement two portable seismic arrays for the volcano program. The approach was based on Lee et al. (1989). Several contracts were awarded to commercial companies to design and implement various components needed to build the portable arrays. The purpose of this report to present the software modules for data acquisition and processing – SUDSPICK, SUDSPLOT, SUDSPROC/SUDSMAN, SUDSSQZ, PC-QMAP, AND XDETECT – in detail as submitted by the contractors. Source code on PC-DOS/MS-DOS diskette for this report is presented in U. S. Geological Survey Open-File Report 92-597-B.

SUDSPICK (p. 4 - 41)

SUDSPICK is a computer program for automatic picking of the first P-arrival from waveform files created by the XDETECT realtime data acquisition program.

SUDSPLOT (p. 42 - 77)

SUDSPLOT is a computer program for plotting seismic waveform files created by the XDETECT realtime data acquisition program.

SUDSPROC/SUDSMAN (p. 78 - 119)

SUDSPROC/SUDSMAN are two computer programs for managing the data processing tasks of the waveform files created by the XDETECT realtime data acquisition program.

SUDSSQZ (p. 120 -131)

SUDSSQZ is a computer program to "squeeze" out insignificant data in a waveform file created by the XDETECT realtime data acquisition program.

PC-QMAP (p. 132 - 188)

PC-QMAP consists of two computer programs to plot earthquake hypocenter data on a map. QMAPHP is for using the HP Laserjet printers, and QMAPPS is for using a Postscript laser printer.

XDETECT (p. 189 - 313)

XDETECT (version 2.04) is a computer program for realtime data acquisition and processing. It is an updated version of the XDETECT program released previously in Tottingham and Lee (1989).

REFERENCES

Lee, W. H. K., D. M. Tottingham, and J. O. Ellis (1989). Design and implementation of a PC-based seismic data acquisition, processing, and analysis system, *IASPEI Software Library, 1*, 21-46.

Tottingham, D. M., and W. H. K. Lee (1989). XDETECT: A fast seismic data acquisition and processing program, *U.S. Geol. Surv. Open-File Report 89-205.*

SUDSPICK

A Program to pick P Phase Arrivals in
Seismic Data in SUDS Format

Version 1.23
May 1992

Robert Banfill
Small Systems Support
2 Boston Harbor Place
Big Water, UT 84741-0205

SUDSPICK Version 1.22

Introduction

This program is used to pick phase arrivals from SUDS data files. The program offers two modes of operation; Batch and Interactive. Batch mode operation is initiated from the DOS command line so that SUDSPICK may be called from within DOS batch files to automate data processing. Interactive mode allows the user to view traces on screen and pick phases using a mouse or the keyboard.

Command line syntax

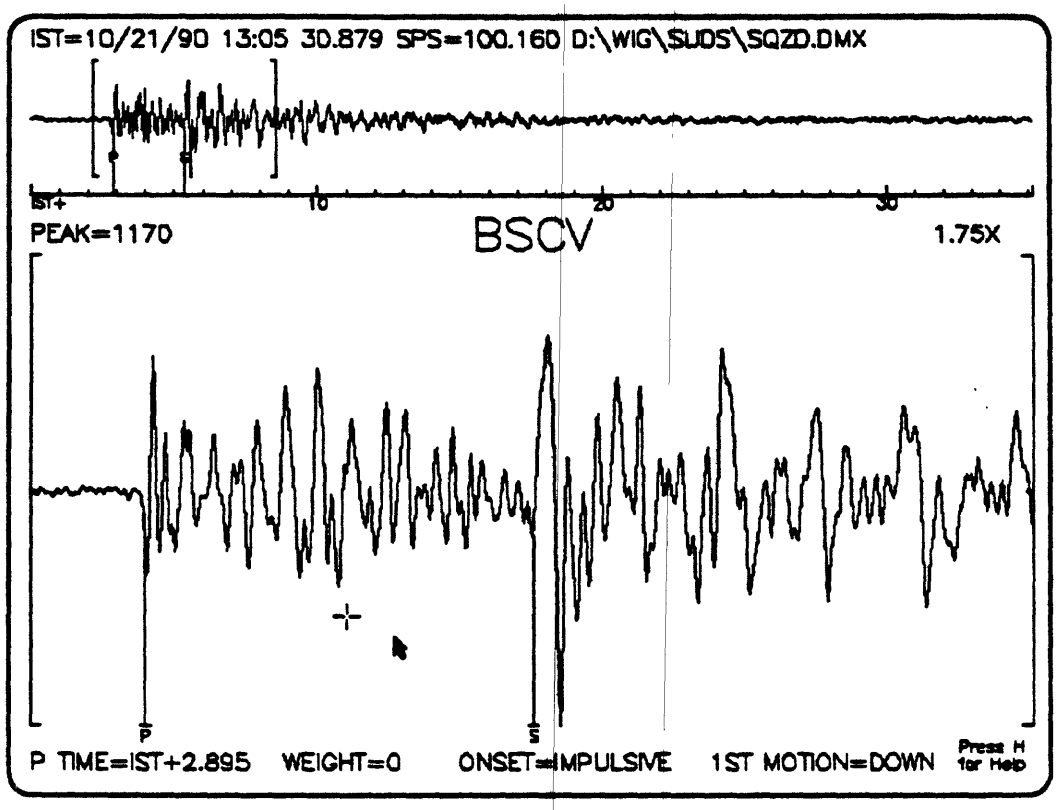
SUDSPICK is executed using the following DOS command line syntax:

```
SUDSPICK [switches] inputfile <return>

switches:
    /Ddriver = Driver filename. (SCREEN.DRV$)
          /B = Batch mode.

   inputfile = SUDS format data file. (SUDSPICK.DMX)

() indicates the default value, arguments may be placed in any order, and
are not case sensitive.
```

The /D option is used to specify the Geograf device driver to be used to drive the display. By default, SUDSPICK will look in its home directory for a driver named SCREEN.DRV. The file SCREEN.DRV that is provided on the distribution diskette will drive a VGA display adapter and is equivalent to VGA.DRV. If you have a different display adapter on your system, you should copy the appropriate driver to SCREEN.DRV so that the program will use that driver by default (*e.g.*, if you have a Hercules Graphics Card, you should issue the following command at the DOS command line: COPY HGC.DRV SCREEN.DRV <return>, assuming that the current working directory is where you installed SUDSPICK).

The /B option initiates batch mode.

Inputfile is the SUDS data file specification. The input file must be demultiplexed and is assumed to have the extension .DMX if an explicit extension is not provided on the command line. It is worthwhile to note that SUDSPICK expands all input file specs to fully qualified file specifications, *i.e*, if the current working directory is C:\DATA and you enter SUDSPICK 90122105 <return>, SUDSPICK expands the input file spec to C:\DATA\90122105.DMX. Any DOS legal partial file spec may be used as well, the input file spec ..\OLDDATA\TEST, would be expanded to C:\OLDDATA\TEST.DMX.



While the trace is displayed on screen, the following keys are used:

Up Arrow, Down Arrow, Left Arrow, Right Arrow or Mouse click moves the crosshair. Hold the main (left) mouse button to drag the crosshair.

Spacebar	P pick if the crosshair is in the lower portion of the screen. Moves the window if the crosshair is above the time axis.
S key	S pick.
U key	First motion is UP.
D key	First motion is DOWN.
I key	Onset is IMPULSIVE.
E key	Onset is EMERGENT.
W key	followed by 0,1,2,3 or 4. Pick WEIGHT.
C key	Clear picks.
+ or - key	Increase or decrease amplitude magnification.

Enter or Return	Save the picks, move to next trace.
Page Up key	Move to previous trace, current picks are not saved.
Page Down key	Move to next trace, current picks are not saved.
Home key	Move to first trace, current picks are not saved.
End key	Move to last trace, current picks are not saved.
Escape key	Exit the program, picks saved in inputbasename.PHA.

SUDSPICK tries to pick the P phase for you, if it was successful, the P Phase arrival will be displayed on the screen. Please note that picks displayed on screen are not saved to the phase file until the Enter or Return key is pressed.

There are six parameters that may be adjusted by the user. These parameters are set in an ASCII text file named SUDSPICK.INI and may be changed using your favorite text editor. SUDSPICK looks for this file in its home directory, so this file should be kept in the same directory as the executable files.

```
# Sample initialization file SUDSPICK.EXE version 1.11 or later
# Lines beginning with #, ! or a space are ignored
# Statements in this file are not case sensitive

skipsamples=50
c1=0.4
c2=1.0
c3=0.3                    (NOTE: c1 to c5 are parameters to be set
c4=0.01                          by users; they have been extended
c5=10.0                          later and became part of SUDSUTIL.INI
                                 file -- see page 10.)
```

The skipsamples entry directly corresponds to the nskip variable, and the c1 through c5 entries directly correspond to the variables with the same name in the source code. If the SUDSPICK.INI file does not exist, the values shown in the sample .INI file are used as default values.

Batch mode operation

When SUDSPICK is executed with the /B option, the program will open the input file and analyze each trace. If a P phase pick was made, an entry will be made in the phase file. The phase file is a HYPO71PC phase file given the same base name as the input file and the extension .PHA (*e.g.*, input file: D:\DATA\90122103.DMX, phase file: D:\DATA\90122103.PHA), and stored in the same directory as the input file, regardless of the current working directory. When SUDSPICK is ran in batch mode, only P phase arrival times are written to the phase file.

Interactive operation

When SUDSPICK is executed without the /B option, the program will pause momentarily, displaying the picking parameters to be used, while it opens files and builds internal data structures and then it will display the first trace.

The SUDSPICK display is made up of the upper portion, which displays the entire trace, and the lower portion that displays a window on the trace with a one sample per pixel resolution on the time axis. The window is marked on the upper trace with brackets and any picks that are not within the window are displayed on the upper trace.

P phase picking algorithm

SUDSPLOT implements a P phase picking algorithm that was originally written by Rex Allen of the USGS. Below is a fragment of FORTRAN source code that contains the algorithm.

```
      INTEGER*2 idata, nskip, ni, nil, idif
      INTEGER*4 nsamp, pick, j
      REAL*4 ail1, bei1, ri, ali, bei, dri, ei, gai, ril
      REAL*8 c1, c2, c3, c4, c5
      DIMENSION idata (*)

!     nsamp  = Number of samples
!     pick   = returns 0 if no pick, otherwise, index of pick
!     idata  = Data buffer
!     nskip  = Number of samples to skip before picking
!     ali    = Short-term average of ei
!     ail1   = Previous value of short-term average of ei
!     bei    = Long-term average of ei
!     bei1   = Previous value of long-term average of ei
!     dri    = Weighted derivative of integer data
!     ei     = Characteristic function
!     gai    = Weighted value of long-term average of E
!     idif   = 1st difference
!     j      = Do loop counter over samples
!     ni     = Value of current data sample
!     nil    = Value of previous digital sample
!     ri     = Present value of real data
!     ril    = Previous value of real data

!     Pick P Phase
      DO j=1, nsamp
!        Prepare one sample
         ni = idata(j)
!        Store present value of real data
         ril = ri
!        Compute first difference
         idif = ni-nil
!        Filter to produce updated value of real data
         ri = c1*ril+REAL(ni)
!        Compute weighted derivative
         dri = c2*REAL(idif)
!        Store present value of digital data
         nil = ni
!        Compute characteristic function
         ei = ri**2+dri**2
!        Compute short-term average of ei
         ali = ail1+c3*(ei-ail1)
         ail1 = ali
!        Compute long-term average of ei
         bei = bei1+c4*(ei-bei1)
         bei1 = bei
         gai = c5*bei
!        Do we have a pick?
         IF (j .GT. nskip .AND. ali .GT. gai) THEN
            pick = j
            EXIT
         ENDIF
      ENDDO
```

Sun 16-Aug-1992 15:43, RB

>>> Notes concerning source code for SUDS utilities <<<

Most of the programs included search for SUDSUTIL.INI in their "home" directory (i.e., the directory where the executable file is located). This file contains initialization information for the various programs. A sample SUDSUTIL.INI is included in the root directory of the diskette as well as in the printed documentation.

Please see the README.TXT file in each subdirectory for specific requirements of each individual porgram.

```
# SUDSUTILS.INI

# Last edit: Fri 10-Jul-1992 16:33, RB

# This file contains initialization data for the various SUDS utilities.

# This file is arranged in sections.  Each section is marked with a
# "Section Header" inside square brakets (e.g., [SUDSPICK]). Each program
# will look for its section and then read the entries following its
# section header.  None of these entries are case-sensitive.  Comments
# are delimited by a pound-sign (#) or a semi-colon (;).  Blank lines
# are ignored.

#-------------------------------------------------------------------------
[PLOTX]
# This section contains entries for the PlotX graphics library.
# All of the programs that use the graphics library will
# use these as default settings.

# Default display mode:
#    1 =   2 color  720x348 Hercules Graphics Card
#    2 =  16 color  640x350 EGA
#    3 =  16 color  640x480 VGA
#    4 =  16 color  800x600 SVGA
#    5 =  16 color 1024x768 SVGA <- ChipSet 1 or 2 only
#    7 = 256 color  640x350 SVGA <- ChipSet 1 or 2 only
#    8 = 256 color  640x480 SVGA <- ChipSet 1 or 2 only
#    9 = 256 color  800x600 SVGA <- ChipSet 1 or 2 only
#   10 = 256 color 1024x768 SVGA <- ChipSet 2 only
DisplayMode = 5

# Default chip set:
#    1 = Tseng Labs ET_3000 (older 512kb cards, Prodesigner, 2 the MAX ...)
#    2 = Tseng Labs ET_4000 (newer 1Mb cards, ProDesigner II, SpeedSTAR ...)
ChipSet = 2

# Prompt before saving the current plot to the queue
PromptBeforeSaving = Y

# Command used to produce hardcopy
# HardCopy = PSPLOT /Pd:\plots\plot.eps
# HardCopy = PSPLOT /Plpt1
HardCopy = LJPLOT /Plpt2 /M3
# Hardcopy = SAVEPLOT c:\plots

# Font vectors filespec
Font = C:\SUDS\SIMPLEX.VEC

# Plot queue directory
PlotQueue = D:\PLOTS

# Directory for temporary files. Should be on a RAM disk if possible
TempDir = D:\

#-------------------------------------------------------------------------
[SUDSMAN]
# This section contains entries for SUDSMAN 1.01 or later.

# Run in verbose mode
Verbose

# Source directory mask
Mask=*.PRE

# Source and destination directories
Source=D:\CODE\SHELL\TEST
Destination=D:\CODE\SHELL\DEST

# Time in seconds after which a file is to be consided inactive
Inactive=0

# Time in seconds to wait between checks for new files
```

```
Wait=15

#-------------------------------------------------------------------------
[SUDSPICK]
# This section contains entries for SUDSPICK 1.10 or later.

# Lines beginning with Pnn= represent params 1 - 25, all other line are
# ingnored.  SUDSPICK uses the comment at the end the label to display
# the value at startup.  This comment must start with a # and must follow
# the value on the same line separated by a least one space.

P10=10.0 #ABS Threshold (sample amplitude)

P3=2.5   #Zero window size (seconds)

P12=6.0  #Additional # of windows to search

P11=3.5  #Signal/noise ratio

# Earthquake wave period criteria
P4=3.0   #Low period limit (seconds)
P13=0.05 #High period limit (seconds)

P1=100.  #'Before' window size (samples)

P2=250.  #'After' window size (samples)

# Pick classification criteria (Rmax normalized by BW/AW)
P5=2.0   #Weight 5 rmax
P6=4.0   #Weight 4 rmax
P7=6.0   #Weight 3 rmax
P8=8.0   #Weight 2 rmax
P9=10.0  #Weight 1 rmax

#-------------------------------------------------------------------------
[SUDSPROC]
# This section contains entries for SUDSPROC 1.01 or later.

# Path to data files
SourceDir = D:\Data

# Path to working directory
WorkingDir = D:\Temp

# Filename extension of raw data files
RawExtension = WVM

# Preferred text editor
Editor = QEDIT.EXE
```

Sun 16-Aug-1992 15:48, RB

>>> Notes on SUDSPICK 1.23 <<<

SUDSPICK was written using Microsoft C/C++ 7.00 and Microsoft FORTRAN 5.00.

Libraries required:

HSUDS.LIB - SUDS data file library version 1.31.
Available from:
Small Systems Support
2 Boston Harbor Place
Big Water, UT 84741-0205
(801) 675-5827 Voice
(801) 675-3730 FAX

L_PLOTX.LIB - PlotX graphics library version 1.12.
Available from:
Small Systems Support
2 Boston Harbor Place
Big Water, UT 84741-0205
(801) 675-5827 Voice
(801) 675-3730 FAX

The following files are included:

SUDSPICK	C	13743	04-22-92	11:27p
INIFILE	FOR	2670	04-28-92	3:04p
PICKLIST	FOR	7141	12-19-91	5:05p
PLTPICK	FOR	17197	01-20-92	11:38p
P_PICK	FOR	16177	04-22-92	11:23p
SUDSPICK	MAK	728	04-28-92	3:19p

```
/*
 * SUDSPICK - P Phase picker
 *
 * Filespec: c:\code\sudspick\sudspick.c
 *
 * Last edit:  Wed 22-Apr-1992 23:25, RB
 *
   Version 0.00 - 0.03 May 1991
      Started with SUDSPICK 1.30.  Rebuilt P picking algorithm.
   Version 0.04 29-Jun-1991, RB
      Major changes to P picker, added Jump.
   Version 1.00 04-Jul-1991, RB
      Fixed problem with SUDS error on last trace
   Version 1.00.01 05-Jul-1991, RB
      weight 5 added
   Version 1.00.02 05-Jul-1991, RB
      Added coda duration and first motion
   Version 1.01.00 14-Jul-1991, RB
      Added 16bit support
   Version 1.10.00 27-Jul-1991, RB
      Major structural changes: pass FI struct to FORTRAN modules.
      Rebuilt picklist & inifile modules, added all motions, and S weight.
      Changed annotation, ini file supports up to 25 params with labels
      taken from comments.
   Version 1.20.00 10-Aug-1991
      Changed to PlotX graphics library. Fixed small bug in jump.
   Version 1.21.00 15-Sep-1991
      When interactive, no weight 5.  Add /N switch.
   Version 1.22.00 09-Nov-1991 23:37, RB
      Huge addressing (HSUDS.LIB).
   Version 1.23.00 12-Dec-1991 11:08, RB
      Updated Plotx library 1.10, added S phase parameters
   Version 1.23.01 Mon 20-Jan-1992 23:30, RB
      Added component to plot annotation
 */

char version[] = "1.23.01";

/* Standard include files */
#include <malloc.h>
#include <stdio.h>
#include <process.h>
#include <string.h>
#include <stdlib.h>

/* SUDS include files */
#include <suds.h>

// Graphics
#include <plotx.h>

/* Prototypes */
void sudspick( char[128], int, int );
int builddb( char[128] );
void upper( char[128] );

extern int _fortran pplot( long *, int _huge *, char *, char *, float *,
                           MS_TIME *, int *, void * );
extern void _fortran picklist( int *, void *, char *, int *);
extern int _fortran done( void );
extern void _fortran p_pick( long *, long *, int _huge *, float *, int *,
                             float *, int *, int *, int *, int * );
extern int _fortran inifile( char *, float * );

extern char *progname;
struct FI { /* --- SUDS file index --- */
   int type;      /* Structure type */
   int pos;       /* Position index */
   char stn[5];   /* Station name */
   MS_TIME ptime;
   MS_TIME stime; /* Pick time */
   int pweight;
```

```
    int sweight;    /* Pick weight */
    int ponset;
    int sonset;     /* Onset -1 = impulsive, 0 = none, 1 = emergent,  */
    int pmotion;
    int smotion;    /* First motion, -3 = Noisy, -2 = - poor D,
                       -1 = Down or dilatation, 0 = none,
                       1 = Up or compression, 2 = + poor U orC */
    int fmp;        /* Coda duration in seconds */
} fi[128];
int dmode, chip;
char font[128];

/* Main Routine */
main( int argc, char *argv[] ) {
    int i, batch, autop;
    char *j, *k;
    char buffer[128], infile[128];

    progname = argv[0];

    /* Setup defaults */
    strcpy( infile, "SUDSPICK.DMX" );
    batch = 0;
    autop = 1;
    dmode = 0;
    font[0] = '\0';
    chip = 0;

    printf( "SUDSPICK Version %.4s - W. Lee & R. Banfill\n\n", version );

    /* Deal with command-line arguments */
    if( argv[1][0] == '?' || argv[1][1] == '?' || argc < 2 )
    {
        printf( "Usage: SUDSPICK [switches] inputfile <- \n" );
        printf( "switches:\n" );
        printf( "           /B = Batch mode. (OFF)\n" );
        printf( "           /N = No auto pick when interactive. (AUTO)\n" );
        printf( "       /Mmode = Display mode:\n" );
        printf( "                 1 =  2 color  720x348 Hercules Graphics Card\n" );
        printf( "                 2 = 16 color  640x350 EGA\n" );
        printf( "                (3)= 16 color  640x480 VGA\n" );
        printf( "                 4 = 16 color  800x600 SVGA\n" );
        printf( "                 5 = 16 color 1024x768 SVGA < Tseng only\n" );
        printf( "       /Cchip = VGA Chipset:\n" );
        printf( "                (1)= Tseng Labs ET3000\n" );
        printf( "                 2 = Tseng Labs ET4000\n" );
        printf( "       /Ffont = font filename. (%s)\n", font );
        printf( "   inputfile = SUDS format waveform file. (%s)\n\n", infile );
        printf( "() indicates the default value.\n");
        printf( "Arguments may be placed in any order, and are not case sensitive.\n" );
        exit( 1 );
    }

    for( i=1; i<argc; i++ ) {
        if( argv[i][0] == '-' || argv[i][0] == '/' ) {
            switch( argv[i][1] ) {
                case 'f':
                case 'F':
                    strcpy( buffer, &argv[i][2] );
                    j = buffer+strlen( buffer )-4;
                    if ( *j != '.' )
                        strcat( buffer, ".vec" );
                    strcpy( font, buffer );
                    break;
                case 'b':
                case 'B':
                    batch = 1;
                    break;
                case 'n':
                case 'N':
                    autop = 0;
                    batch = 0;
```

```
            break;
         case 'm':
         case 'M':
            if( ( dmode = atoi( &argv[i][2] ) ) == 0 )
               fprintf( stderr, "\nERROR: Invalid switch: %s\n\n", &argv[i][0] );
            if( dmode < 1 || dmode > 10 ) {
               fprintf( stderr, "\nWARNING: Mode value out of range: %s\n", &argv[i][0] );
               fprintf( stderr, "Defaulting to Mode 3 (VGA, 640x480)\n\n" );
               dmode = 3;
            }
            break;
         case 'c':
         case 'C':
            if( ( chip = atoi( &argv[i][2] ) ) == 0 )
               fprintf( stderr, "\nERROR: Invalid switch: %s\n\n", &argv[i][0] );
            if( chip < 1 || chip > 2 ) {
               fprintf( stderr, "\nWARNING: Chipset out of range: %s\n", &argv[i][0] );
               fprintf( stderr, "Defaulting to ET3000\n\n" );
               chip = 1;
            }
            break;
         default:
            fprintf( stderr, "\nWARNING: Unrecognized switch: %s\n", &argv[i][0] );
            break;
      }
   }
   else
      strcpy( infile, argv[i] );
      j = infile+strlen( infile )-4;
      if( *j != '.' )
         strcat( infile, ".dmx" );
}

// If batch, auto pick
if( batch == 1 )
   autop = 1;

/* Get the full pathname */
if( _fullpath( buffer, infile, 128 ) != NULL )
   memcpy( infile, buffer, 128 );
else {
   fprintf( stderr, "Invalid input filespec!" );
   exit( 1 );
}
/* Convert everything to upper case */
upper( infile );
upper( font );

sudspick( infile, batch, autop );
exit( 0 );
}

/* SUDSPICK ------------------------------------------------------------*/
void sudspick( char infile[128], int batch, int autop ) {
   FILE *infil;
   register i, j;
   char samtime[34], stime[34], ptime[34], stn[5], buffer[32],
      inif[_MAX_PATH], *p, comp;
   int typ, ponset, pmotion, weight, flag, fii, isk,
      ikey, maxsam, motval, fmp, avenoi, sweight;
   long inp, len, leng, ppick, plen, spick;
   float rate, params[25], s2n;
   MS_TIME ist;

   char _huge *ptr;
   SUDS_DESCRIPTRACE _huge *dt;

   printf( "Waveform file: %s\n\n", infile );
   if( batch == 1 )
      printf( "Batch Mode!\n" );
   else
      printf( "Interactive Mode!\n" );
```

```
if( autop == 0 )
   printf( "Auto picking is DISABLED!\n" );
printf( "\n" );

// Get picking parameters
strcpy( inif, progname );
p = strrchr( inif, '\\' );
p++;
strcpy( p, "SUDSUTIL.INI" );
if( inifile( inif, params ) != 0 ){
   fprintf( stderr, "\nERROR: Reading initialization file: %s\n", inif );
   exit( 1 );
}

/* Build index */
fii = builddb( infile );
switch( fii ) {
   case -1:
      fprintf( stderr, "\nERROR: File contains no data: %s\n", infile );
      exit( 1 );
      break;
   case -2:
      fprintf( stderr, "\nERROR: Unable to open: %s\n", infile );
      exit( 1 );
      break;
   case -3:
      fprintf( stderr, "\nERROR: Data in multiplexed form: %s\n", infile );
      fprintf( stderr, "         Run DEMUX and try again.\n" );
      exit( 1 );
      break;
}

/* Open the WaveForm file */
infil = st_open( infile, "r+b" );

/* Initialize plot software */
if( batch == 0 ) {
   if( iniplt( inif, dmode, chip, font, 0 ) != 0 )
      exit( 1 );
   if( plots( 1 ) != 0 )
      exit( 1 );
}

i = 0;
while( i<=fii ) {
   isk=st_seek( infil, (long)fi[i].pos-1, 0 );
   if( ( inp = st_get( &ptr, &typ, &len, infil ) ) == EOF )
      break;

   dt = (SUDS_DESCRIPTRACE _huge *)ptr;
   // leng = dt->length - (len/sizeof(int));
   leng = dt->length;
   rate = dt->rate+dt->rate_correct;
   ist = dt->begintime+dt->time_correct;
   maxsam = dt->maxdata-dt->avenoise;
   avenoi = dt->avenoise;

   sprintf( samtime, "%s", list_mstime( ist, 4 ) );
   strcpy( stn, dt->dt_name.st_name );
   comp = (char)toupper( dt->dt_name.component );
   ppick = 0;
   spick = 0;

   /* Let the computer try to pick the P phase */
   plen = leng;
   p_pick( &plen, &ppick, (int _huge *)ptr+(len/sizeof(int)), params,
          &weight, &rate, &motval, &fmp, &avenoi, &autop );
   if( ppick == -1 ) {
      if( batch == 0 )
         done( );
      fprintf( stderr, "Not enough memory!\n" );
      st_free( ptr, (int)inp );
```

```
            st_close( infil );
            exit( 1 );
         }

         if( ppick != 0 && fi[i].ptime == 0.) {
            fi[i].ptime = ist+((float)ppick/rate);
            fi[i].pweight = weight;
            fi[i].pmotion = motval;
            fi[i].fmp = fmp;
            sprintf( ptime, "%s", list_mstime( fi[i].ptime, 4 ) );
         }

         /* if in batch mode... */
         if( batch != 0  )    {
            if( batch == 1 )
               printf( "Initial Sample Time = %s\n\n", samtime );
            batch = 2;
            if( ppick != 0 )
               printf( "Station: %s, Component: %c, P arrival: %s, Weight: %d\n", stn, comp, ptir
            i += 1;
         }
         /* Let the user try it */
         else {
            ikey = pplot( &leng, (int _huge *)ptr+(len/sizeof(int)),
                          infile, &comp, &rate, &ist, &maxsam, &fi[i] );
            st_free( ptr, (int)inp );

            switch( ikey ) {
               /* Navigation keys */
               case 0x011B: /* ESCAPE */
                  ppick = -1;
                  break;
               case 0x4900: /* PGUP */
                  if( i > 0 )
                     i--;
                  else
                     i = 0;
                  break;
               case 0x5100: /* PGDN */
                  if( i < fii )
                     i++;
                  else
                     i = fii;
                  break;
               case 0x4f00: /* END */
                  i = fii;
                  break;
               case 0x4700: /* HOME */
                  i = 0;
                  break;
               case 0x246A:
               case 0x244A:
                  plot( 0., 0., 999 );
                  while( 1 ) {
                     printf("\nEnter station to jump to -> " );
                     gets( buffer );
                     upper( buffer );
                     for( j=0; j<=fii; j++ ) {
                        if( strcmp( fi[j].stn, buffer ) == 0 ) {
                           i = j;
                           break;
                        }
                     }
                     if( i == j )
                        break;
                     else
                        printf( "\nStation %s not found!\n", buffer );
                  }
                  plots( 1 );
                  break;
               /* Save the pick */
               case 0x1C0D:    /* RETURN */
```

```
                i++;
                break;
             default:    /* Any other key */
                break;
          }
       }
       st_free( ptr, (int)inp );
       if( ppick == -1 )
          break;
    }
    st_close( infil );
    if( batch == 0 )
       done(  );

    /* Save the picklist */
    picklist( &fii, &fi, infile, &batch );
    return;
}

/* BuildDB ------------------------------------------------------------*/
int builddb( char infile[128] ) {
    FILE *infil;
    register i;
    char _huge *ptr;
    int typ, fii;
    long inp, len;
    static SUDS_DESCRIPTRACE _huge *dt;

    /* Build index of input file */
    if( ( infil = st_open( infile, "r+b" ) ) == NULL )
       return( -2 );
    fii = -1;
    while( ( inp = st_get( &ptr, &typ, &len, infil ) ) != EOF ) {
       switch( typ ) {
          case MUXDATA:
             return( -3 );
             break;
          case DESCRIPTRACE:
             dt = (SUDS_DESCRIPTRACE _huge *)ptr;
             if( strncmp( dt->dt_name.st_name, "IRIG", 4 ) != 0 ) {
                fii += 1;
                fi[fii].type = typ;
                fi[fii].pos = st_tell( infil );
                strcpy( fi[fii].stn, dt->dt_name.st_name );
             }
             break;
       }
       st_free( ptr, (int)inp );
    }
    st_close( infil );
    for( i=0; i<=127; i++ ) {
       fi[i].ptime = 0.;
       fi[i].stime = 0.;
       fi[i].pweight = 5;
       fi[i].sweight = 5;
       fi[i].ponset = 0;
       fi[i].ponset = 0;
       fi[i].pmotion = 0;
       fi[i].smotion = 0;
       fi[i].fmp = 0;
    }
    return( fii );
}

/* Die ----------------------------------------------------------------*/
long die( int in ) {
    exit( 1 );
}

/* Upper --------------------------------------------------------------*/
void upper( char buffer[128] ) {
    char *p;
```

```
    for( p = buffer; *p; p++ )
        *p = toupper( *p );
}
```

```
$TITLE: 'Filespec: c:\code\sudspick\inifile.for'
!---    Last edit: 28-Apr-1992 15:04, RB
!---
!---  Get picking parameters from INI file
!---
!-------------------------------------------------------------------------

      INTEGER*2 FUNCTION INIFILE (inifil, params)

      IMPLICIT NONE

      CHARACTER inifil*128
      INTEGER*2 line, i, j, k
      REAL*4 params(25)

      CHARACTER buffer*128, cbuf*128

      INTEGER*2 LTRIM

      line = 0
      inifile = -1

      OPEN (UNIT=10,FILE=inifil,STATUS='old',ERR=99999)

      WRITE (*,'(A,A,/)') " Initialization file: ",
     +   inifil(:INDEX(inifil, CHAR(0))-1)

      buffer = ' '

!     Look for our section
      DO WHILE (buffer(1:10) .NE. '[SUDSPICK]' )
         READ (UNIT=10,FMT='(A)',ERR=99999,END=90000) buffer
         CALL UPPER(buffer)
         line = line+1
      ENDDO

10000 READ (UNIT=10,FMT='(A)',ERR=99999,END=90000) buffer
!      CALL UPPER(buffer)
      line = line+1

!     Ignore everything that doesnt start with P
      IF (buffer(1:1) .EQ. 'P') THEN
         j = INDEX(buffer,'=')
         READ (buffer(2:j-1), FMT='(I2)',ERR=10010) i
         IF (i .GT. 25) THEN
            WRITE (*,'(A)') ' ERROR: To many parameters: Max=25'
            GOTO 10010
         ELSE
            GOTO 10020
         ENDIF

!        Error
10010    WRITE (*,'(A,I2,A,A,A,A,/)')
     +                  ' ERROR: Unrecognized entry at line ',line,
     +                  ' in ', inifil(:LEN_TRIM(inifil)),': ',
     +                  buffer(:LEN_TRIM(buffer))
         RETURN

10020    k = INDEX(buffer(j+1:),'#')+j
         IF (k .LE. j+2) k = LEN_TRIM(buffer)+1
         READ (buffer(j+1:k-1), FMT='(F16.8)', ERR=10030 ) params(i)
         IF (params(i) - INT(params(i)) .EQ. 0.) THEN
            WRITE (cbuf,FMT='(I10)') INT(params(i))
         ELSE
            WRITE (cbuf,FMT='(F10.1)') params(i)
         ENDIF
         IF (INDEX(buffer,'#') .EQ. 0 ) THEN
            WRITE (*,'(A,I2,A,A)') ' Param ',i,' = ',
     +         cbuf(LTRIM(cbuf):LEN_TRIM(cbuf))
         ELSE
            WRITE (*,'(A,A,A,A)') ' ',
     +         buffer(INDEX(buffer,'#')+1:LEN_TRIM(buffer)),
```

```
     +          ' = ', cbuf(LTRIM(cbuf):LEN_TRIM(cbuf))
          ENDIF
          GOTO 10000

!         Error
10030     WRITE (*,'(A,I2,A,A,A,A,/)')
     +                    ' ERROR: Invalid entry at line ',line,
     +                    ' in ', inifil(:LEN_TRIM(inifil)),': ',
     +                    buffer(:LEN_TRIM(buffer))
          RETURN

        ENDIF

        GOTO 10000

90000   inifile = 0

99999   WRITE (*,'(A)')
        RETURN
        END
```

```
$TITLE: 'Filespec: d:\code\sudspick\picklist.for'
!---    Last edit: 19-Dec-1991 16:59, RB
!---
!---  Write out the phase file
!---
!---  27-Jul-1991, RB.
!---  I completely rebuilt this routine because of the problem of the
!---  hour needing to be the same for all picks (see VOL 1, pp.218).
!---
!-------------------------------------------------------------------------

      INTERFACE TO CHARACTER*128 FUNCTION list_mstime [C]
     +                                                ( mstime, form )
         REAL*8 mstime
         INTEGER*2 form
      END

!-------------------------------------------------------------------------

      INTEGER*2 FUNCTION picklist ( n, pha, wvm, batch )

      IMPLICIT NONE

      STRUCTURE /phaselist/
         INTEGER*2 type
         INTEGER*2 position
         CHARACTER*5 stn
         REAL*8 ptime
         REAL*8 stime
         INTEGER*2 pweight
         INTEGER*2 sweight
         INTEGER*2 ponset
         INTEGER*2 sonset
         INTEGER*2 pmotion
         INTEGER*2 smotion
         INTEGER*2 fmp
      END STRUCTURE

      RECORD /phaselist/ pha(*)
      INTEGER*2 n, batch
      CHARACTER wvm*128

      CHARACTER*128 list_mstime

      INTEGER*2 hr, mn, mhr, i, mnd, mns, picks
      REAL*4 fsec
      CHARACTER time*32, buf*128, phafil*128

      picklist = -1
      n = n+1
      picks = 0
      mhr = 99

!     Open the phase file
      phafil = wvm(:INDEX(wvm,'.'))//'PHA'//'  '
      OPEN (UNIT=10, FILE=phafil, STATUS='unknown')

!---  time  MM/DD/YY HH:MM SS.HHH
!---        123456789a123456789b1  (form 4)

!---  Find the earliest hour
      DO i=1, n
         IF (pha(i).ptime .NE. 0.) THEN
            time = list_mstime (pha(i).ptime, 4)
            READ (time(10:11), '(I2)') hr
            IF (hr .LE. mhr) THEN
               mhr = hr
            ENDIF
         ENDIF
      ENDDO

!---  Phase list output line format
```

```
!---  STN OPM0 YYMMDDHHMMSS.HH       SS.HH S 0                                    00000
!---  123456789a123456789b123456789c123456789d123456789e123456789f123456789g123456789
!---  STN=Station, O=Onset, M=Motion.
!---  note: see IASPEI Vol 1, Chp 9, Sec 5.9 (HYPO71PC phase list) pp. 218

      DO i=1, n

!        If we have a P-pick
         IF (pha(i).ptime .NE. 0.) THEN

!           Cycle if in interactive mode with weight 5
            IF (batch .EQ. 0 .AND. pha(i).pweight .EQ. 5) CYCLE

!           Station
            buf(1:4) = pha(i).stn(1:4)

!           P Onset
            SELECT CASE (pha(i).ponset)
               CASE (-1)
                  buf(5:5) = 'I'
               CASE (1)
                  buf(5:5) = 'E'
               CASE DEFAULT
                  buf(5:5) = ' '
            END SELECT

!           P mark
            buf(6:) = 'P'

!           P Motion
            SELECT CASE (pha(i).pmotion)
               CASE (-3) !Noise
                  buf(7:7) = 'N'
               CASE (-2) !Poor Down
                  buf(7:7) = '-'
               CASE (-1) !Down
                  buf(7:7) = 'D'
               CASE (1) !Up
                  buf(7:7) = 'U'
               CASE (2) !Poor Up
                  buf(7:7) = '+'
               CASE DEFAULT !Blank
                  buf(7:7) = ' '
            END SELECT

!           P weight
            WRITE (buf(8:8), '(I1)') pha(i).pweight

!           time yymmddhhmmss.hhh
!                123456789a123456 (form 0)

!           Year, month, day
            mnd = 0
            time = LIST_MSTIME (pha(i).ptime, 0)
            buf(10:15) = time(1:6)

!           Hour
            READ (time(7:8), '(I2)') hr
            IF (hr .EQ. mhr) THEN
               buf(16:17) = time(7:8)
            ELSEIF (hr .EQ. mhr+1) THEN
               mnd = 60
               WRITE (buf(16:17), '(I2.2)') mhr
            ELSE
               WRITE (*,'(A,A)') ' Phase time out of range: Phour: ',
     +                           pha(i).stn(:4)
               CYCLE
            ENDIF

!           Minute
            READ (time(9:10), '(I2)') mn
            IF (mnd .EQ. 0) THEN
```

```
            buf(18:19) = time(9:10)
         ELSE
            mn = mn+mnd
            IF (mn .GE. 100) THEN
               WRITE (*,'(A,A)') ' Phase time out of range: Pmin: ',
     +                           pha(i).stn(:4)
               CYCLE
            ENDIF
            WRITE (buf(18:19), '(I2.2)') mn
         ENDIF

!        Second
         buf(20:24) = time(11:15)

!        S pick
         IF (pha(i).stime .NE. 0.) THEN

!           yymmddhhmmss.hhh
!           123456789a123456
            time = LIST_MSTIME (pha(i).stime, 0)

            READ (time(9:10), '(I2)') mns
            IF (mns .EQ. mn) THEN
               buf(32:36) = time(11:15)
            ELSEIF (mns .EQ. mn+1) THEN
               READ (time(11:15), '(F5.2)') fsec
               IF (fsec+60. .GE. 100.) THEN
                  WRITE (*,'(A,A)')
     +                 ' Phase time out of range: Ssec: ',
     +                 pha(i).stn(:4)
                  CYCLE
               ENDIF
               WRITE (buf(32:36), '(F5.2)') fsec+60.
            ELSE
               WRITE (*,'(A,A)')
     +              ' Phase time out of range: Ssec: ',
     +              pha(i).stn(:4)
               CYCLE
            ENDIF

!           S Onset
            SELECT CASE (pha(i).sonset)
               CASE (-1)
                  buf(37:37) = 'I'
               CASE (1)
                  buf(37:37) = 'E'
               CASE DEFAULT
                  buf(37:37) = ' '
            END SELECT

!           S mark
            buf(38:38) = 'S'

!           S Motion
            SELECT CASE (pha(i).smotion)
               CASE (-3) !Noise
                  buf(39:39) = 'N'
               CASE (-2) !Poor Down
                  buf(39:39) = '-'
               CASE (-1) !Down
                  buf(39:39) = 'D'
               CASE (1) !Up
                  buf(39:39) = 'U'
               CASE (2) !Poor Up
                  buf(39:39) = '+'
               CASE DEFAULT !Blank
                  buf(39:39) = ' '
            END SELECT

!           S weight
            WRITE (buf(40:40), '(I1)') pha(i).sweight
```

```
            ENDIF

!           Coda duration
            IF (pha(i).fmp .NE. 0) THEN
               WRITE (buf(71:75), '(I5.5)') pha(i).fmp
            ENDIF

!           Write out the line
            WRITE (10, '(A)') buf(:LEN_TRIM(buf))
            picks = picks+1

         ENDIF

      ENDDO

      IF (picks .NE. 0) THEN
!                          123456789012345678
         WRITE (10,'(A)') '                  1'
         CLOSE (10)
         WRITE (*,'(A,I3,A,A)') ' ',picks,' pick(s) written to: ',
     +                           phafil(:LEN_TRIM(phafil))
      ENDIF

      picklist = 0

      RETURN
      END
```

```
$TITLE: 'Filespec: d:\code\sudspick\pltpick.for'
!---    Last edit: 19-Dec-1991 16:29, RB
!---
!---  Plot routines for SUDSPICK
!---
!---  Heavily modified 28-Jul-1991, RB
!-------------------------------------------------------------------------
$STORAGE:2

      INTERFACE TO CHARACTER*128 FUNCTION list_mstime [C]
     +                                              ( mstime, form )
         REAL*8 mstime
         INTEGER*2 form
      END

      INCLUDE 'plotx.fi'

!-------------------------------------------------------------------------

!---  Close up shop

      INTEGER*2 FUNCTION DONE ()

      IMPLICIT INTEGER*2 (a-z)

      CALL PLOT (0.,0.,999)

      done = 0
      RETURN

      END

!-------------------------------------------------------------------------

!---  Plot wiggles for to pick them

      INTEGER*2 FUNCTION PPLOT (len, sam [HUGE], wvm, comp, rate,
     +                          istime, maxsam, pha)

      IMPLICIT NONE

!---  Param's and local variables
      STRUCTURE /phaselist/
         INTEGER*2 type
         INTEGER*2 position
         CHARACTER*5 stn
         REAL*8 ptime
         REAL*8 stime
         INTEGER*2 pweight
         INTEGER*2 sweight
         INTEGER*2 ponset
         INTEGER*2 sonset
         INTEGER*2 pmotion
         INTEGER*2 smotion
         INTEGER*2 fmp
      END STRUCTURE

      RECORD /phaselist/ pha

      CHARACTER buffer*128, wvm*128, maxtxt*5,
     +          srate*8, ist*26, sec*6, comp*1
      INTEGER*2 sam, peaksam, newsam, ikey, maxsam, ierr,
     +          dec, pen, xmax, ymax, xmin, ymin, tic, secs
      INTEGER*4 len, endsam, begsam, ibegsam, iendsam, ipick, ppick,
     +          spick, i
      REAL*4 rate, x, y, ystep, xstep, ybase, sfac, slfac, r,
     +       rymin, rymax, rxmax, rxmin, z, mx, my
      REAL*8 istime
      DIMENSION sam (*)

!---  Functions
      INTEGER*2 LTRIM, KCHARIN, GCURSOR, XPIXELS, YPIXELS
```

```
      CHARACTER*128 LIST_MSTIME

      DATA mx, my /5., 4./

      pplot = 1

      ist = LIST_MSTIME (istime, 4)

!---  Get maximum sample value
      peaksam = 1
      DO i=1,len
         newsam = IABS(sam(i))
         IF (peaksam .LT. newsam) peaksam = newsam
      ENDDO
      WRITE (maxtxt,'(I5.5)') peaksam
      CLOSE (20)

9000  IF (pha.ptime .NE. 0) ppick = INT4((pha.ptime-istime)*rate)
      IF (ppick .LT. 0) ppick = 0
      IF (pha.stime .NE. 0) spick = INT4((pha.stime-istime)*rate)
      IF (spick .LT. 0) spick = 0

      IF (pha.pweight .EQ. 5) THEN
         ipick = 0
      ELSE
         ipick = ppick
      ENDIF

!---  Default scale for lower portion
      s1fac = REAL(maxsam)/REAL(peaksam)
      IF (s1fac .LT. 1.) s1fac = 1.
      IF (s1fac .GT. 100.) s1fac = 100.

!---  Establish plotting constants
10000 sfac = 25.
      rymin = 0.
      rxmin = 0.
      rymax = 8.
      rxmax = 10.
      begsam = 1
      endsam = len
      xmax = XPIXELS( )-1
      ymax = YPIXELS( )-1
      xmin = 0
      ymin = 0

      IF (ipick .LE. (xmax-xmin)/2+1) ipick = (xmax-xmin)/2+1
      IF (ipick .GE. endsam-((xmax-xmin)/2+1))
     +    ipick = endsam-((xmax-xmin)/2+1)
      ibegsam = ipick-((xmax-xmin)/2)
      iendsam = ipick+((xmax-xmin)/2)+1

      dec = REAL(endsam-begsam)/REAL(xmax-xmin+1)
      IF (dec .LT. 1) dec = 1

      ybase = 6.975
      xstep = rxmax/REAL(endsam-begsam)

      IF (sfac .LT. maxsam/REAL(peaksam)) THEN
         ystep = 1.25/((REAL(maxsam)*2.)/sfac)
      ELSE
         ystep = 1.25/(REAL(peaksam)*2.)
      ENDIF

!---  Clear screen
      CALL GFILL (0.,0.,10.,8.,0)

!---  Upper portion ----------------------------------------------------

!---  Plot upper title
      CALL NEWPEN (7)
      buffer = 'FILE='//wvm(:INDEX(wvm,'.')+3)
```

```
      CALL SYMBOL (rxmin,7.75,.175,buffer,0.,LEN_TRIM(buffer))
!      CALL SYMBOL (9.,.27,.125,'PRESS F1 ',0.,8)
!      CALL SYMBOL (9.,.03,.125,'FOR HELP',0.,8)

!---  Plot upper time axis
      secs = INT2(REAL(endsam)/rate)
      IF (secs .GE. 1500) THEN
         tic = 250
      ELSEIF (secs .GE. 750 ) THEN
         tic = 100
      ELSEIF (secs .GE. 300 ) THEN
         tic = 50
      ELSEIF (secs .GE. 150 ) THEN
         tic = 20
      ELSEIF (secs .GE. 50 ) THEN
         tic = 10
      ELSE
         tic = 5
      ENDIF
      CALL NEWPEN (7)
      CALL SYMBOL (rxmin,6.005,.12,'0',0.,1)
      CALL NEWPEN (3)
      CALL PLOT (rxmin,6.25,3)
      CALL PLOT (rxmin,6.15,2)
      DO i=0, endsam/rate
         z = REAL(i)*(xstep*rate)
         CALL PLOT (z,6.15,2)
         IF (MOD(i,tic) .EQ. 0 .AND. i .NE. 0) THEN
            CALL PLOT (z,6.25,2)
            CALL NEWPEN (7)
            WRITE (sec,'(I6)') i
            CALL SYMBOL (z, 6.065, .10, sec(LTRIM(sec):LEN_TRIM(sec))
     +                  ,0., -(LEN_TRIM(sec)-(LTRIM(sec)-1)))
            CALL NEWPEN (3)
         ELSE
            CALL PLOT (z,6.20,2)
         ENDIF
         CALL PLOT (z,6.15,3)
      ENDDO
      CALL PLOT (rxmax,6.15,2)
      CALL PLOT (rxmax,6.25,2)

!---  Plot upper trace
      CALL NEWPEN (7) ! White

      CALL TRACE (rxmin, 6.4, rxmax, 7.55, 0, len, sam)

!---  indicate picks on upper trace
      IF (pha.pweight .NE. 5) THEN
         CALL NEWPEN (12)
         CALL PLOT (ppick*xstep,ybase+(ystep*sam(ppick)),3)
         CALL PLOT (ppick*xstep,6.625,2)
         CALL SYMBOL (ppick*xstep,6.5625,.1,'P',0.,-1)
         CALL PLOT (ppick*xstep,6.50,3)
         CALL PLOT (ppick*xstep,6.15,2)
      ENDIF
      IF (spick .NE. 0) THEN
         CALL NEWPEN (12)
         CALL PLOT (spick*xstep,ybase+(ystep*sam(spick)),3)
         CALL PLOT (spick*xstep,6.625,2)
         CALL SYMBOL (spick*xstep,6.5625,.1,'S',0.,-1)
         CALL PLOT (spick*xstep,6.50,3)
         CALL PLOT (spick*xstep,6.15,2)
      ENDIF
      IF (pha.fmp .NE. 0) THEN
         CALL NEWPEN (12)
         z = xstep*(REAL(ppick)+(REAL(pha.fmp)*rate))
         pen = 3
         DO r=6.35, 7.6, 0.1
            CALL PLOT (z, r, pen)
            IF(pen .EQ. 3) THEN
               pen = 2
```

```
            ELSE
               pen = 3
            ENDIF
         ENDDO
      ENDIF

!---  indicate window on upper trace
      CALL NEWPEN (14)
      CALL PLOT (ibegsam*xstep+.05,6.35,3)
      CALL PLOT (ibegsam*xstep,6.35,2)
      CALL PLOT (ibegsam*xstep,7.6,2)
      CALL PLOT (ibegsam*xstep+.05,7.6,2)

      CALL PLOT (iendsam*xstep-.05,6.35,3)
      CALL PLOT (iendsam*xstep,6.35,2)
      CALL PLOT (iendsam*xstep,7.6,2)
      CALL PLOT (iendsam*xstep-.05,7.6,2)

!---  Lower portion ------------------------------------------------

!---  Draw lower amplitude axes
      CALL NEWPEN (14)
      CALL PLOT (rxmin+.1,.5,3)
      CALL PLOT (rxmin,.5,2)
      CALL PLOT (rxmin,3.,2)
      CALL PLOT (rxmin+.05,3.,2)
      CALL PLOT (rxmin,3.,3)
      CALL PLOT (rxmin,5.5,2)
      CALL PLOT (rxmin+.1,5.5,2)

      CALL PLOT (rxmax-.1,.5,3)
      CALL PLOT (rxmax,.5,2)
      CALL PLOT (rxmax,3.,2)
      CALL PLOT (rxmax-.05,3.,2)
      CALL PLOT (rxmax,3.,3)
      CALL PLOT (rxmax,5.5,2)
      CALL PLOT (rxmax-.1,5.5,2)

      xstep = rxmax/REAL((xmax-xmin)+1)
      ybase = 3.
      ystep = 5./((REAL(maxsam)*2.)/s1fac)

!---  Annotate Peak Magnification and Station name
      CALL NEWPEN (15)
      buffer = pha.stn(:LEN_TRIM(pha.stn(:4)))
      CALL SYMBOL (5.,5.725,.4,buffer,0.,-LEN_TRIM(buffer))

      CALL NEWPEN (7)
      buffer = 'PEAK='//maxtxt//' COMPONENT='//comp
      CALL SYMBOL (0., 5.78, .15, buffer, 0., LEN_TRIM(buffer))
      WRITE (buffer,'(F6.2)') s1fac
      CALL SYMBOL (0., 5.57, .15,
     +    'MAG='//buffer(LTRIM(buffer):6)//'X', 0.,
     +    5+(7-LTRIM(buffer)))

      CALL SYMBOL (6.25, 5.78, .15, 'IST='//ist(:21), 0., 25)
      WRITE (srate,'(F8.3)') rate
      buffer = 'SPS='//srate(LTRIM(srate):8)
      CALL SYMBOL (6.25, 5.57, .15, buffer, 0., LEN_TRIM(buffer))

      CALL ANNOT (pha, istime, rate, ppick, spick)

!---  Plot the trace
      CALL PLOT (rxmin,ybase+(ystep*sam(ibegsam)),3)
      DO i=ibegsam+1, iendsam
         x = (i-ibegsam)*xstep
         y = ybase+(ystep*sam(i))
         IF( y .GT. 5.5) y = 5.5
         IF( y .LT. .5) y = .5
         CALL PLOT (x,y,2)
      ENDDO
```

```
!---  Plot the pick
      IF (pha.pweight .NE. 5 .AND.
     +   (ppick .GE. ibegsam .AND. ppick .LE. iendsam)) THEN
          CALL NEWPEN (12)
          y = ybase+(ystep*sam(ppick))
          IF( y .GT. 5.5) y = 5.5
          IF( y .LT. .5) y = .5
          CALL PLOT (REAL(ppick-ibegsam+1)*xstep,y,3)
          CALL PLOT (REAL(ppick-ibegsam+1)*xstep,.75,2)
          CALL SYMBOL (REAL(ppick-ibegsam+1)*xstep,.63,.12,'P',0.,-1)
          CALL PLOT (REAL(ppick-ibegsam+1)*xstep,.54,3)
          CALL PLOT (REAL(ppick-ibegsam+1)*xstep,.5,2)
          CALL PLOT (REAL(ppick-ibegsam+1)*xstep-.05,.5,3)
          CALL PLOT (REAL(ppick-ibegsam+1)*xstep+.05,.5,2)
      ENDIF

      IF (spick .NE. 0 .AND.
     +   (spick .GE. ibegsam .AND. spick .LE. iendsam)) THEN
          CALL NEWPEN (12)
          y = ybase+(ystep*sam(spick))
          IF( y .GT. 5.5) y = 5.5
          IF( y .LT. .5) y = .5
          CALL PLOT (REAL(spick-ibegsam+1)*xstep,y,3)
          CALL PLOT (REAL(spick-ibegsam+1)*xstep,.75,2)
          CALL SYMBOL (REAL(spick-ibegsam+1)*xstep,.63,.12,'S',0.,-1)
          CALL PLOT (REAL(spick-ibegsam+1)*xstep,.54,3)
          CALL PLOT (REAL(spick-ibegsam+1)*xstep,.5,2)
          CALL PLOT (REAL(spick-ibegsam+1)*xstep-.05,.5,3)
          CALL PLOT (REAL(spick-ibegsam+1)*xstep+.05,.5,2)
      ENDIF

80010 ikey = GCURSOR (mx, my)
      SELECT CASE (ikey)
         CASE (16#1970, 16#01) ! P - p phase pick w/pweight
            IF (my .GE. 6.0) THEN
               ipick = INT4(mx/(rxmax/REAL(endsam)))
               GOTO 10000
            ELSE
               ipick = ibegsam+INT4(mx/xstep)
               ppick = ipick
               CALL GFILL (0., 0., 9., .48, 0)
               CALL NEWPEN (15)
               CALL SYMBOL (5.,.25,.15,'Enter P Weight (0-4)',0., -20)
80012          ikey = KCHARIN ()
               IF (ikey .LT. 48 .OR. ikey .GT. 52) GOTO 80012
               pha.pweight = ikey-48
               CALL GFILL (0., 0., 9., .48, 0)
               CALL SYMBOL (5.,.25,.15,
     +                      'Enter P First Motion (U,D,+,-,N)',
     +                       0., -32)
               ikey = KCHARIN ()
               SELECT CASE (CHAR(ikey))
                  CASE ('U', 'u')
                     pha.pmotion = 1
                  CASE ('D', 'd')
                     pha.pmotion = -1
                  CASE ('+')
                     pha.pmotion = 2
                  CASE ('-')
                     pha.pmotion = -2
                  CASE ('N', 'n')
                     pha.pmotion = -3
                  CASE DEFAULT
                     pha.pmotion = 0
               END SELECT
               CALL GFILL (0., 0., 9., .48, 0)
               CALL SYMBOL (5.,.25,.15,'Enter P Onset (E,I)',0., -19)
               ikey = KCHARIN ()
               SELECT CASE (CHAR(ikey))
                  CASE ('E', 'e')
                     pha.ponset = 1
                  CASE ('I', 'i')
```

```
                  pha.ponset = -1
               CASE DEFAULT
                  pha.ponset = 0
            END SELECT
            CALL ANNOT (pha, istime, rate, ppick, spick)
            GOTO 10000
         ENDIF
      CASE (16#1F73, 16#02) ! S - S phase pick
         IF (my .GE. 6.0) THEN
            ipick = INT4(mx/(rxmax/REAL(endsam)))
            GOTO 10000
         ELSE
            ipick = ibegsam+INT4(mx/xstep)
            spick = ipick
            CALL GFILL (0., 0., 9., .48, 0)
            CALL NEWPEN (15)
            CALL SYMBOL (5.,.25,.15,'Enter S Weight (0-4)',0., -20)
80011       ikey = KCHARIN ()
            IF (ikey .LT. 48 .OR. ikey .GT. 52) GOTO 80011
            pha.sweight = ikey-48
            CALL GFILL (0., 0., 9., .48, 0)
            CALL SYMBOL (5.,.25,.15,
     +                  'Enter S First Motion (U,D,+,-,N)',
     +                   0., -32)
            ikey = KCHARIN ()
            SELECT CASE (CHAR(ikey))
               CASE ('U', 'u')
                  pha.smotion = 1
               CASE ('D', 'd')
                  pha.smotion = -1
               CASE ('+')
                  pha.smotion = 2
               CASE ('-')
                  pha.smotion = -2
               CASE ('N', 'n')
                  pha.smotion = -3
               CASE DEFAULT
                  pha.smotion = 0
            END SELECT
            CALL GFILL (0., 0., 9., .48, 0)
            CALL SYMBOL (5.,.25,.15,'Enter S Onset (E,I)',0., -19)
            ikey = KCHARIN ()
            SELECT CASE (CHAR(ikey))
               CASE ('E', 'e')
                  pha.sonset = 1
               CASE ('I', 'i')
                  pha.sonset = -1
               CASE DEFAULT
                  pha.sonset = 0
            END SELECT
            CALL ANNOT (pha, istime, rate, ppick, spick)
            GOTO 10000
         ENDIF
      CASE (16#2E63) ! C - Coda
         IF (my .GE. 6.0) THEN
            ipick = INT4(mx/(rxmax/REAL(endsam)))
            IF (ipick .LE. ppick) GOTO 80010
            pha.fmp = INT(REAL(ipick-ppick)/rate)
            GOTO 10000
         ELSE
            ipick = ibegsam+INT4(mx/xstep)
            IF (ipick .LE. ppick) GOTO 80010
            pha.fmp = INT(REAL(ipick-ppick)/rate)
            GOTO 10000
         ENDIF
      CASE DEFAULT
         pplot = ikey
         IF (ppick .NE. 0)
     +      pha.ptime = istime+(DFLOAT(ppick)/DFLOAT(rate))
         IF (spick .NE. 0)
     +      pha.stime = istime+(DFLOAT(spick)/DFLOAT(rate))
         ppick = 0
```

```
            spick = 0
            GOTO 90000
      END SELECT

90000 CONTINUE

99999 RETURN

      END

!---------------------------------------------------------------------

      SUBROUTINE ANNOT (pha, istime, rate, ppick, spick)

      IMPLICIT NONE

      STRUCTURE /phaselist/
         INTEGER*2 type
         INTEGER*2 position
         CHARACTER*5 stn
         REAL*8 ptime
         REAL*8 stime
         INTEGER*2 pweight
         INTEGER*2 sweight
         INTEGER*2 ponset
         INTEGER*2 sonset
         INTEGER*2 pmotion
         INTEGER*2 smotion
         INTEGER*2 fmp
      END STRUCTURE

      RECORD /phaselist/ pha

      REAL*8 istime
      REAL*4 rate
      INTEGER*4 ppick, spick

      INTEGER*2 LTRIM
      CHARACTER*128 LIST_MSTIME

      CHARACTER*128 buf, buf1, buf2

      buf1 = ' '
      CALL GFILL (0., 0., 9., .48, 0)
      CALL NEWPEN (7)

      IF (ppick .NE. 0) THEN
         buf1 = 'P: '
         buf = LIST_MSTIME (istime+(DFLOAT(ppick)/DFLOAT(rate)), 4)
         buf1(4:23) = buf(1:20)
         buf1(24:) = ' '
         SELECT CASE (pha.ponset)
            CASE (-1)
               buf1(LEN_TRIM(buf1)+1:) = ', IMPULSIVE'
            CASE (1)
               buf1(LEN_TRIM(buf1)+1:) = ', EMERGENT'
         END SELECT
         SELECT CASE (pha.pmotion)
            CASE (-3)
               buf1(LEN_TRIM(buf1)+1:) = ', NOISE'
            CASE (-2)
               buf1(LEN_TRIM(buf1)+1:) = ', POOR DOWN'
            CASE (-1)
               buf1(LEN_TRIM(buf1)+1:) = ', DOWN'
            CASE (1)
               buf1(LEN_TRIM(buf1)+1:) = ', UP'
            CASE (2)
               buf1(LEN_TRIM(buf1)+1:) = ', POOR UP'
         END SELECT
         WRITE (buf1(LEN_TRIM(buf1)+1:), '(A,I1)') ', WHT=',pha.pweight
         IF (pha.fmp .NE. 0) THEN
            WRITE (buf, '(I5)') pha.fmp
```

```
      buf1(LEN_TRIM(buf1)+1:) = ', FMP='//
+           buf(LTRIM(buf):LEN_TRIM(buf))
    ENDIF
    CALL SYMBOL (0., .27, .15, buf1, 0., LEN_TRIM(buf1))
 ENDIF

 IF (spick .NE. 0) THEN
    buf2 = 'S: '
    buf = LIST_MSTIME (istime+(DFLOAT(spick)/DFLOAT(rate)), 4)
    buf2(4:23) = buf(1:20)
    buf2(24:) = ' '
    SELECT CASE (pha.sonset)
       CASE (-1)
          buf2(LEN_TRIM(buf2)+1:) = ', IMPULSIVE'
       CASE (1)
          buf2(LEN_TRIM(buf2)+1:) = ', EMERGENT'
    END SELECT
    SELECT CASE (pha.smotion)
       CASE (-3)
          buf2(LEN_TRIM(buf2)+1:) = ', NOISE'
       CASE (-2)
          buf2(LEN_TRIM(buf2)+1:) = ', POOR DOWN'
       CASE (-1)
          buf2(LEN_TRIM(buf2)+1:) = ', DOWN'
       CASE (1)
          buf2(LEN_TRIM(buf2)+1:) = ', UP'
       CASE (2)
          buf2(LEN_TRIM(buf2)+1:) = ', POOR UP'
    END SELECT
    WRITE (buf2(LEN_TRIM(buf2)+1:), '(A,I1)') ', WHT=',pha.sweight
    CALL SYMBOL (0., .03, .15, buf2, 0., LEN_TRIM(buf2))
 ENDIF

 RETURN
 END
```

```
$TITLE: 'Filespec: d:\code\pick1\p_pick.for'
!---    Last edit: 02-Aug-1991 19:39, RB
!---    take out first-motion  5/3/91 WL
!---    Major work 29-Jun-1991, RB, WL
!---    Added coda duration & first motion 7/5/91, RB & WL
!---    Fooling around with Leif           7/11/91, WL
!---    Fix debug output & pick point      7/14/91, WL
!---    Fix zero crossing bug              7/16/91, WL
!---    Fine-tune pick                     7/17/91, WL
!---    Fine-tune pick by fitting lines    7/18/91, WL
!---    Try different points for lines     7/21/91, WL
!---    Fix centerning of lines            7/22/91, WL
!---    Added up to 25 picking parameters 26-Jul-1991 RB

!   ------------------------------------------------------------------
!
!            A P phase picker designed by W. H. K. Lee and
!                    Robert Banfill -  April 1991.
!
!   ------------------------------------------------------------------

!---  Params array contains the following parameters, see PICK.INI for
!     actual values.  Up to 25 parameter ma be specified in the .INI
!     file using this form: Pnn=pppp.pppp, where nn is the array
!     subscript and pppp.pppp is the parameter value.  All params are
!     read from the .INI file as reals.


!     Params(1) - Before window (samples)
!     Params(2) - After window (samples)
!     Params(3) - Windows size (seconds)
!     Params(4) - Wave period lower limit (seconds)
!     Params(5) - Rmax < W5, pick is discarded
!     Params(6) - Rmax < W4, pick is weighted 4
!     Params(7) - Rmax < W3, pick is weighted 3
!     Params(8) - Rmax < W2, pick is weighted 2
!     Params(9) - Rmax < W1, pick is weighted 1
!                 Rmax >= W1, pick is weighted 0
!     Params(10) - ABS thershold (sample amplitude)
!     Params(11) - Signal/noise ratio
!     Params(12) - Number of additional windows to be searched
!     Params(13) - Wave period upper limit (seconds)

!   ------------------------------------------------------------------

      INCLUDE 'plotx.fi'

      SUBROUTINE P_PICK( leng, pick, data [HUGE], params,
     +                   weight, rate, motion, fmp, avenoi, autop)

      IMPLICIT INTEGER*2 (a-z)
c     IMPLICIT NONE

!---  This INCLUDE contains the paramater structure definition
      INCLUDE 'plotx.inc'

      INTEGER*2 weight, motion, fmp, avenoi, autop
      INTEGER*4 leng, pick
      REAL*4    params, rate
      INTEGER*2 data(*)
      DIMENSION params(*)

!---  Local vars
      INTEGER*2 jmax, smax, weight1, weight2, xdata(5000)
      INTEGER*2 ierr, bwsize, awsize, countsw(100),
     +          zwsamp, zwcut, count, ilimit
      INTEGER*4 i, j, sum, sum1, sum2, average, aveabs, aw, bw,
     +          irmax, istart, iend, imax, imin, jstart, itemp,
     +          bwzr, awzr, idmax, istep, kmax,
     +          istart1, istart2, npoint1, npoint2
      REAL*4    rmax, loperiod, hiperiod, absw(100),
     +          abswmax, periodsw(100), abswmin, abswcut, rw,
```

```
     +          signal, noise, picksn, sumdr, summin, y1, y2, ry,
     +          b1, b2, sb1, sb2, bmax, bdiff, b2save, sumysq1, sumysq2

!---  Dynamically allocated arrays
      REAL*4    r [ALLOCATABLE, HUGE] (:)
      REAL*4    dr [ALLOCATABLE, HUGE] (:)
      INTEGER*1 zerocr [ALLOCATABLE, HUGE] (:)


!---  This declares the actual parameter structure
!      RECORD /XYPARAMS/ xy

!---  Generate hardcopy when in plot mode 1 or 2
!      CALL HARDCOPY( 1 )

!>>>> Plot in open coming in <<<<<<<<<<
!---  Set graphics, 0 = prompt for plotting mode
!      ierr = PLOTS( 0 )

!---  We have no pick going in
      pick = 0
      motion = 0
      weight = 5
      fmp = 0

!---  Normalize the data & and compute the average  ------------------
      sum = 0
      DO i=1, leng
         data(i) = data(i) - avenoi
         sum = sum + data(i)
      ENDDO
      average = NINT(REAL(sum)/REAL(leng))

!---  Subtract the average -------------------------------------------
      ilimit = HUGE(ilimit)
      DO i=1, leng
         itemp = INT4(data(i))-average
         IF( itemp .GT. ilimit ) THEN
            data(i) = ilimit
         ELSEIF( itemp .LT. -ilimit ) THEN
            data(i) = -ilimit
         ELSE
            data(i) = INT2(itemp)
         ENDIF
      ENDDO

!---  Simple plot ----------------------------------------------------

!---  Make sure we dont run off the end of xdata
      IF( leng .GT. 5000 ) THEN
         j = 5000
      ELSE
         j = leng
      ENDIF

      DO i=1, j
         xdata(i) = i - 1
      ENDDO

!      CALL GFILL( 0., 0., 10., 8., 0 )

!      xy.xaxis = 1       ! 1=plot X axis, 0=no X axis
!      xy.yaxis = 1       ! 1=plot Y axis, 0=no Y axis
!      CALL XYPLOT( xdata, data, j, LOC(xy) )

!---  Wait for a keypress
!      ierr = KCHARIN()


!>>>> Rather than close, clear screen to return to called state
!---  Close the plot
!      CALL PLOT( 0.,0.,999 )
```

```
!      CALL GFILL( 0., 0., 10., 8., 0 )

!---  Bail out if autopick is off ----------------------------------
      IF (autop .EQ. 0) GOTO 99999

      ALLOCATE (r(leng), STAT=ierr)
      IF (ierr .NE. 0) GOTO 50000

      ALLOCATE (dr(leng), STAT=ierr)
      IF (ierr .NE. 0) GOTO 50000

      ALLOCATE (zerocr(leng), STAT=ierr)
      IF (ierr .NE. 0) GOTO 50000

!---  Check Absolute value of entire trace ------------------------
      sum = 0
      DO i=1, leng
         sum = sum + IABS(data(i))
      ENDDO
      aveabs = NINT(REAL(sum)/REAL(leng))

      IF (aveabs .LT. INT(params(10))) goto 80000

!---  Locate zero-crossings ---------------------------------------
      DO i=1, leng-1
         IF (data(i) .GE. 0) THEN
            IF (data(i+1) .GE. 0) THEN
               zerocr(i) = 0
            ELSE
               zerocr(i) = 1
            ENDIF
         ELSE
            IF (data(i+1) .GE. 0) THEN
               zerocr(i) = 1
            ELSE
               zerocr(i) = 0
            ENDIF
         ENDIF
      ENDDO

!---  Determine where to start ------------------------------------
      zwsamp = params(3) * rate
      loperiod = params(4)
      hiperiod = params(13)
      zwcut    = INT(2. * params(3) / loperiod)

!---  Compute period and absolute amplitude for each window
c**    OPEN( 20, FILE='debug.dat' )

      abswmin = 999999.
      jmax = leng/zwsamp
      DO j=1, jmax
         imin = (j-1)*zwsamp + 1
         imax = imin + zwsamp - 1

         sum1 = 0
         sum2 = 0
         DO i=imin,imax
            sum1 = sum1 + zerocr(i)
            sum2 = sum2 + IABS(data(i))
         ENDDO
         periodsw(j) = 2. * (REAL(zwsamp)/(REAL(sum1)+0.000001)) / rate
         absw(j) = REAL(sum2)/REAL(zwsamp)
         IF (abswmin .GT. absw(j)) abswmin = absw(j)
c**       WRITE( 20, FMT='(3I5,2F12.4)') j, sum1, zwsamp,
c**  +                                   periodsw(j), absw(j)

      ENDDO

!---  Count number of windows that meet Freq. and ABS requirments
      jstart = 0
```

```
      count = 0
      abswmax = 0.
      abswmin = params(11)*abswmin
      DO j=1, leng/zwsamp
         countsw(j) = 0
         IF (((periodsw(j) .LT. loperiod) .AND.
     +        (periodsw(j) .GE. hiperiod)) .AND.
     +        (absw(j) .GE. abswmin)) THEN
            countsw(j) = 1
            count = count + 1
            IF (abswmax .LT. absw(j)) THEN
               jstart = j
               abswmax = absw(j)
            ENDIF
         ENDIF
      ENDDO

c**    WRITE( 20, FMT='(I5,F12.4)') jstart, abswmax

!---  No windows pass
      IF (count .eq. 0) GOTO 80000

!---  Look backward from the strongest window for starting window
!---  Require 2 failed windows
      DO j=jstart-1, 2, -1
c**    WRITE( 20, FMT='(3I5)') j, countsw(j), countsw(j-1)
         IF ((countsw(j) .EQ. 0) .AND. (countsw(j-1) .EQ. 0)) EXIT
      ENDDO

!---  Look forward from the starting window
      sum = countsw(j+1) + countsw(j+2) + countsw(j+3) +
     +      countsw(j+4) + countsw(j+5) + countsw(j+6)

!---  4 out of 6 windows must meet Freq. and ABS requirments
      IF (sum .LT. 4) GOTO 80000

!---  Start picking in previous 2 windows
      jstart = j-2
      IF (jstart .LT. 1) jstart = 1

c**    WRITE( 20, FMT='(I5)') jstart

!---  Sliding windows  ------------------------------------------------

!---  Set up window parameters
40000 bwsize = params(1)
      awsize = params(2)
      bw = 0
      aw = 0
      bwzr = 0
      awzr = 0
      rw = REAL(bwsize)/REAL(awsize)

      imin = (jstart-1)*zwsamp + 1
      istart = imin - zwsamp
      IF (istart .LT. 1) istart = 1
      iend = imin + INT(params(12))*zwsamp
      IF (iend .GT. leng) iend = leng

!---  Compute ABS and Zerocrossings for the bw
      DO i=istart, istart+bwsize
         bw = bw + IABS(data(i))
         bwzr = bwzr + zerocr(i)
      ENDDO

!---  Compute ABS and Zerocrossings for the aw
      DO i=istart+bwsize,istart+bwsize+awsize-1
         aw = aw + IABS(data(i))
         awzr = awzr + zerocr(i)
      ENDDO

!---  Establish other initial values
```

```
      IF (bw .LT. 1) bw = aw
      r(istart+bwsize) = REAL(aw)/REAL(bw)
      rmax = r(istart+bwsize)
      irmax = istart + bwsize
      i = istart + bwsize

!---  Slide to the right
      DO i=istart+bwsize+1, iend-awsize

!---     Adjust the bw and aw ABS
         bw = bw + IABS(data(i)) - IABS(data(i-bwsize))
         bwzr = bwzr + zerocr(i) - zerocr(i-bwsize)
         aw = aw + IABS(data(i+awsize-1)) - IABS(data(i-1))
         awzr = awzr + zerocr(i+awsize-1) - zerocr(i-1)
         IF (bw .LT. 1) bw = aw
         r(i) = REAL(aw)/REAL(bw)
         dr(i) = r(i) - r(i-1)
c**      WRITE( 20, FMT='(3I6,2I8,2F12.4,I3)') i,data(i),zerocr(i),
c**  +                                          bw,aw, r(i), dr(i)
!---     Keep track of rmax and index
         IF (r(i) .GT. rmax) THEN
            rmax = r(i)
            irmax = i
         ENDIF

      ENDDO

C---- Search for the biggest drop in r(i) ************************
      istep = 8

      sumdr = 0.
      DO i=irmax,irmax+istep-1
       sumdr = sumdr + dr(i)
      ENDDO

      imin = irmax
      imax = imin + awsize/2
      summin = sumdr
      idmax = imin

      DO i=imin,imax
         sumdr = sumdr - dr(i) + dr(i+istep)
         IF (sumdr .LT. summin) THEN
            summin = sumdr
            idmax = i
         ENDIF

c**      WRITE( 20, FMT='(2I6,4F12.4,I5)') i, data(i), r(i), dr(i),
c**  +                                     sumdr, summin, idmax
      ENDDO

!--- Normalize rmax
      rmax = rmax*(params(1)/params(2))

!--- Pick based on rmax & classify
      pick = irmax
      IF (rmax .LT. params(5)) THEN
         pick = 0
      ELSEIF (rmax .LT. params(6)) THEN
         weight = 4
      ELSEIF (rmax .LT. params(7)) THEN
         weight = 3
      ELSEIF (rmax .LT. params(8)) THEN
         weight = 2
      ELSEIF (rmax .LT. params(9)) THEN
         weight = 1
      ELSE
         weight = 0
      ENDIF

!--- Case for weight <= 2
      IF (weight .LE. 2) THEN
```

```
          weight1 = weight
          IF (pick .LT. idmax) pick = idmax

c**       WRITE( 20, FMT='(3I6)') irmax, idmax, pick

!--- Fine-tune pick by fitting straight lines

          bmax = 0.
          b2save = 0.

          imin = irmax - 5
          imax = idmax + 5

          DO i=imin, imax
             npoint1 = 21
             istart1 = i - npoint1 + 1
             CALL LINE(data, istart1, npoint1, b1, sb1, sumysq1)
             npoint2 = 5
             istart2 = i - (npoint2/2)
             CALL LINE(data, istart2, npoint2, b2, sb2, sumysq2)

             IF (ABS(b1) .LT. 1.) b1 = 1.
             bdiff = ABS(b2/b1)
             IF (bmax .LT. bdiff) THEN
                bmax = bdiff
                kmax = istart2
                b2save = b2
             ENDIF

             y1 = SQRT(sumysq1/REAL(npoint1))
             y2 = SQRT(sumysq2/REAL(npoint2))
             ry = y2/y1

c**          WRITE( 20, FMT='(3I6,I3,F7.1,I6,I3,5F7.1)') i,data(i),istart1,
c**  +        npoint1, b1, istart2, npoint2, b2, bdiff, y1, y2, ry

             IF (bdiff .GE. 30.) GOTO 10
             IF ( ry .GE. params(11)) GOTO 10

          ENDDO

          pick = kmax
          GOTO 20

    10    pick = istart2
          b2save = b2

    20    smax = 0
          j = 0
          DO i = pick+1, pick+awsize
             j = j + 1
             IF (smax .LT. IABS(data(i))) smax = IABS(data(i))
             IF ((zerocr(i) .EQ. 1) .and. (j .GT. 3))  EXIT
          ENDDO
          signal = REAL(smax)

          sum = 0
          DO i = pick-1, pick-bwsize, -1
          sum = sum + IABS(data(i))
          ENDDO
          noise = REAL(sum) / REAL(bwsize)

          picksn = signal/noise

          IF (picksn .LT. (params(5)*0.5)) THEN
             pick = 0
             weight2 = 5
          ELSEIF (picksn .LT. (params(6)*0.5)) THEN
             weight2 = 4
          ELSEIF (picksn .LT. (params(7)*0.5)) THEN
             weight2 = 3
          ELSEIF (picksn .LT. (params(8)*0.5)) THEN
```

```
            weight2 = 2
         ELSEIF (picksn .LT. (params(9)*0.5)) THEN
            weight2 = 1
         ELSE
            weight2 = 0
         ENDIF

         weight = (weight1+weight2+1)/2

c**      WRITE( 20, FMT='(2I6,3F12.4,3I3)') j, pick, signal,noise,
c**  +                              picksn, weight1, weight2, weight

!---        Do first motion
         IF (weight .LE. 2) THEN
            IF (b2save .GT. 1.) motion = 1
            IF (b2save .LT. -1.) motion = -1
         ENDIF
      ENDIF

!---  Do coda duration for weight <= 1 trace only
!     Reject any fmp < 5 sec as unreliable
      IF (weight .LE. 1) THEN
         jstart = pick / zwsamp
         IF (jstart .LE. 1) GOTO 80000
         abswcut = 2.* absw(jstart-1)

         DO j = jstart, jmax-2
            IF (absw(j)   .GE. abswcut) CYCLE
            IF (absw(j+1) .GE. abswcut) CYCLE
            IF (absw(j+2) .GE. abswcut) CYCLE
!            IF (absw(j+3) .GE. abswcut) CYCLE
            fmp = INT( REAL(j - jstart + 1) * params(3))
            IF (fmp .LT. 5) fmp = 0
            EXIT
         ENDDO
      ENDIF

80000 DEALLOCATE (r)
      DEALLOCATE (dr)
      DEALLOCATE (zerocr)

      GOTO 99999

!---  Memory allocation error
50000 pick = -1

99999 IF (pick .EQ. 0) THEN
         pick = leng/2
         weight = 5
      ENDIF

c**    CLOSE( 20 )

      RETURN
      END

!   ---------------------------------------------------------------------
!
!            A simple straight line fit subroutine by W. H. K. Lee
!                              July, 1991.
!
!   ---------------------------------------------------------------------

      SUBROUTINE LINE(data, istart, npoint, b, sb, sumysq)

      IMPLICIT NONE

      INTEGER*2 data
      INTEGER*4 istart, npoint, iend
      DIMENSION data (*)
      REAL*4 b, sb
```

```
!---  Local vars
      INTEGER*4 i, j
      REAL*4 sumx,sumy,sumxy,sumxsq,sumysq,sxsq,sysq,syxsq,n

      iend = istart + npoint - 1
      n = REAL(npoint)
      sumx = 0.
      sumy = 0.
      sumxy = 0.
      sumxsq = 0.
      sumysq = 0.
      j = 0
      DO i=istart, iend
         j = j + 1
         sumx = sumx + REAL(j)
         sumy = sumy + REAL(data(i))
         sumxy = sumxy + (REAL(j * data(i)))
         sumxsq = sumxsq + (REAL(j))**2
         sumysq = sumysq + (REAL(data(i)))**2
      ENDDO
      b = (n*sumxy - sumx*sumy)/(n*sumxsq - sumx**2)
      sxsq = (n*sumxsq - sumx**2)/(n*(n-1)) + 0.000001
      sysq = (n*sumysq - sumy**2)/(n*(n-1)) + 0.000001
      syxsq = ((n-1)/(n-2)) * (sysq - (b**2)*(sxsq))
      sb = SQRT(ABS(syxsq))/(SQRT(ABS(sxsq))*SQRT(ABS(n-1.)))

C**   WRITE( 20, FMT='(/,8F10.2)') sumx,sumy,sumxy,sumxsq,sumysq,sxsq,
C**  +                             sysq,syxsq
C**   WRITE( 20, FMT='(7I6,2F10.2)') j,i,data(istart),
C**  +data(istart+1), data(istart+2),data(istart+3),data(istart+4),b,sb

      RETURN
      END
```

SUDSPLOT

A program to plot pseudo Record-Section Plots of
Seismic Data Stored in SUDS Format

Version 2.0
May 1992

Robert Banfill
Small Systems Support
2 Boston Harbor Place
Big Water, Utah 84741-0205

Contents

Getting Started

1.0 Overview

SUDSPLOT is a component of the IASPEI software libraries. It is one of many programs in the library that are designed to process seismic data stored in SUDS (Seismic Unified Data System) 1.x format. SUDSPLOT generates pseudo record-section plots of digital timeseries data on a variety of graphic displays and hardcopy devices. When used in conjunction with HYPO71PC, it can also mark phase arrival information and coda duration's on the timeseries' and include hypocenter information in the plot annotation.

This program expects input data files to conform to SUDS format as defined in SUDS Version 1.31, R. Banfill, 8 March 1992 and SUDS: Seismic Unified Data System, Peter L. Ward, U.S.G.S. Open-file report 89-188, 29 March 1989.

SUDSPLOT is controlled from the DOS command-line through various "switches". This method of control was chosen because it lends itself to batch processing. Below I will outline a generic data acquisition and processing system:

1) A program such as RTP (real-time processor) or XDETECT is used to record seismic waveforms in SUDS 1.3x format. The file naming convention used for seismic network data is: YYMMDDNN.WVM, where, YY is the year, MM is the month, DD is the day, NN is the event number for this day, WV identifies the file as a SUDS waveform file in multiplexed form and M is the agency code (M = Menlo Park, A = Alaska, etc.). For data recorded on portable autonomous digital seismographs (PADS) such as the RefTek IRIS/PASSCAL instrument, we generally need more accurate time information as well as a station name in the filename. We have established the following naming convention for this type of data: TTTTTTTT.SSN, where, TTTTTTTT is the initial sample time (number of seconds since 1-1-70 00:00:00) of the earliest waveform in the file represented as a 32 bit integer in hexadecimal notation, SS is a two alpha-numeric character station identifier and N is the data stream number. A utility program named STTIME is provided to convert the hex time filename to/from year, month, day, hour, minute, second and day of year.

2) One of these data files would usually be moved to a working directory on a different machine and possibly archived to tape or optical disk. For network data with IRIG time code recorded on one channel, the file would be processed with FIXTIME to correct the initial sample time and sampling rate. The file would then be demultiplexed with DEMUX. Once the file is demultiplexed, the filename extension is usually changed to .DMX.

3) Phase arrivals and coda duration's would then be picked using SUDSPICK. This information is written to disk in a HYPO71PC compatible phase file with the same name as the data file and a .PHA extension.

4) HYPO71PC would then be used to locate the hypocenter. This program generates a "printer output file" named HYPO71PC.PRT. This file should be renamed to the same name as the data file with a .PRT extension. This file contains the hypocenter solution, various statistics and station information.

5) Finally, SUDSPLOT is used to create plots of the waveforms and optionally mark phase arrivals and coda duration's on them. These plots may be printed on most popular laser printers at 300 dot per inch (dpi) and may be displayed on most display adapter / monitor combinations.

Several example batch files will be given in chapter 3 that illustrate this type of processing.

1.1 System Requirements

SUDSPLOT requires an IBM compatible personal computer running PC-DOS or MS-DOS version 3.2 or later. A 80x87 co-processor is not required, but is strongly recommended. Although SUDSPLOT does not directly use extended memory, it does greatly benefit from disk caches and the use of virtual disks (RAM disks) and so several megabytes (Mb) of extended (or less beneficial, expanded) memory is recommended.

SUDSPLOT uses the PlotX graphics library for graphics display and hardcopy.

The PlotX library currently supports the following display adapter / monitor combinations of graphics display:

- Hercules Graphics Card (HGC) / TTL Monochrome Monitor.
- IBM Enhanced Graphics Adapter (EGA) / Enhanced Color Monitor
- IBM Video Graphics Array (VGA) / Analog Color Monitor
- Orchid Prodesigner and other "Super VGA" display adapter using the Tseng Labs ET3000 chipset at resolutions of 800 × 600 in 16 or 256 colors and 1024 × 768 in 16 colors with the appropriate monitor.
- Orchid Prodesigner II, SpeedStar VGA and other "Super VGA" display adapter using the Tseng Labs ET4000 chipset at resolutions of 800 × 600 in 16 or 256 colors and 1024 × 768 in 16 or 256 colors with the appropriate monitor.

The PlotX library produces device independent binary plot files (.PLX files) which are processed by post processing software to generate high resolution hardcopy. SUDSPLOT calls these programs automatically to produce hardcopy. Currently, the following post-processors are provided:

LJPLOT - This program generates 300 dpi plots on the Hewlett Packard Laserjet family of laser printers. This program generates plots only at 300 dpi, therefore, your printer must contain at least 1.5 megabytes on memory. This program incorporates several data compression schemes that are implemented in the Laserjet family. Below is a summary of the various models that are supported by LJPLOT:

- Laserjet, Laserjet Plus and Laserjet series II - These printers, with additional memory may be used, but they do not support any form of data compression. On a typical 386 based machine with the printer connected with the parallel interface, plots will take three to four minutes per page to print.
- Laserjet IIp - This printer implements "PackBits" data compression in the printer. LJPLOT reduces that amount of data that must be transmitted to the printer by a factor of two to five depending on the complexity of the image and print times will be reduced accordingly.
- Laserjet III, IIIp and IIIsi - These printers implement PackBits and a new compression scheme called "Delta" compression. LJPLOT uses both methods of compression to reduce the amount of data transmitted to the printer by a factor of 20 or more. Print times will be dramatically reduced.

PSPLOT - This program generates PostScript page descriptions for use with PostScript devices. The plot vectors are stored in the .PLX file with a granularity of 300 dpi and therefore accuracy is limited to 300 dpi even if the PostScript device is capable of higher resolution.

Other post-processors are under development at this time and will be available in the near future. If you have a specific output device that is not supported and need a post-processor, contact me at the address on the cover and we can either develop the post-processor or provide you with the plot file specification.

1.2 Installation

All files on the distribution disk should be placed in a single directory on your hard disk. This directory should be added the your PATH. SUDSPLOT looks in its "home" directory (*i.e.*, the directory where SUDSPLOT.EXE is located) for various support files such as the SUDSUTIL.INI, all of these files should be kept in the same directory. SUDSUTIL.INI contains initialization information that is needed by SUDSPLOT at start-up.

Below is a general installation procedure:

Insert the distribution disk into drive A: and close the door.

Issue the following commands at the DOS prompt:

```
C:
MD \SUDS
CD \SUDS
XCOPY A:\*.*
```

When copying is complete, you should edit your AUTOEXEC.BAT file to include C:\SUDS in your PATH statement. Your PATH statement should look something like this:

```
PATH C:\;C:\DOS;C:\UTILS;C:\SUDS
```

If you placed the distribution diskette into a drive other than A:, substitute that drive letter for A: in the above commands. If you want to install SUDSPLOT on a drive other than C:, substitute that drive letter for C: in the above commands. Finally, if you want to install SUDSPLOT in a directory other than \SUDS, substitute that directory name for \SUDS in the above commands.

Before using SUDSPLOT, you should reset your computer (press CTRL-ALT-DEL) so that changes to your configuration will take effect.

As mentioned above, at start-up SUDSPLOT looks for a file named SUDSUTIL.INI. This file contains initialization data for several different programs. The file is separated into "sections". Each section starts with a "section header" delimited by square brackets (*e.g.*, [PLOTX] marks the PLOTX section). This is a simple ASCII text file and you may edit it with your favorite text editor.

Because SUDSPLOT uses the PlotX graphics library, it looks at the PLOTX section to find out about your display adapter, where to put plot files, where to find the font file and which post-processor to use. Below is an example of the PLOTX section of SUDSUTIL.INI:

```
[PLOTX]
# This section contains entries for the PlotX graphics library.
# All of the programs that use the graphics library will
# use these as default settings.

# Default display mode:
#    1 =   2 color  720x348 Hercules Graphics Card
#    2 =  16 color  640x350 EGA
#    3 =  16 color  640x480 VGA
#    4 =  16 color  800x600 SVGA
#    5 =  16 color 1024x768 SVGA <- ChipSet 1 or 2 only
#    7 = 256 color  640x350 SVGA <- ChipSet 1 or 2 only
#    8 = 256 color  640x480 SVGA <- ChipSet 1 or 2 only
#    9 = 256 color  800x600 SVGA <- ChipSet 1 or 2 only
#   10 = 256 color 1024x768 SVGA <- ChipSet 2 only
DisplayMode = 3

# Default chip set:
#    1 = Tseng Labs ET_3000 (older 512kb cards, ProDesigner)
#    2 = Tseng Labs ET_4000 (newer 1Mb cards, ProDesigner II)
ChipSet = 2

# Prompt before saving the current plot to the queue
PromptBeforeSaving = Y

# Command used to produce hardcopy
# HardCopy = PSPLOT /Plpt1
HardCopy = LJPLOT /Plpt1 /M3
# Hardcopy = SAVEPLOT C:\PLOTS

# Font vectors filespec
Font = C:\SUDS\SIMPLEX.VEC

# Plot queue directory
PlotQueue = C:\PLOTS

# Directory for temporary files.
TempDir = D:\
```

Most of these settings are pretty straight forward, but a few need further explanation. The Hardcopy entry specifies the command line used to invoke the post-processor. This particular entry specifies that there is a Laserjet III connected to LPT1. If you need to change these settings, type LJPLOT at the DOS prompt to see help and adjust this entry as needed. If you are going to use PSPLOT, again type PSPLOT at the DOS prompt for help. The commented line Hardcopy = SAVEPLOT C:\PLOTS calls a batch file names SAVEPLOT.BAT which simply copies the current plot file to C:\PLOTS and gives it the name PLOT.PLX. You may then process this plot file manually with LJ or PSPLOT. This is handy if you wish to create a Encapsulated PostScript (.EPS) file to include in a publication.

The PlotQueue entry simply points to the directory where plot files will be stored. The TempDir entry points to the directory where temporary files will be created. This should be a Virtual disk or RAM disk if possible, but be sure that there is enough room on this drive for the files. A good rule is that there should be enough space on the temporary drive to hold the SUDS file that you are processing, although sometimes even more is needed.

Take a minute and read though the top portion of SUDSUTIL.INI and carefully adjust the appropriate entries to match your system. I recommend that you use comments to annotate changes that you have made so that you will understand them in the future. After you have done this, you should be ready to run.

1.3 Performance Considerations

As mentioned above, SUDSPLOT will make use of a virtual disk for holding temporary files. If you have at several Mb of extended memory, and will be processing small to medium sized SUDS files, the following line can be added to your CONFIG.SYS file:

```
DEVICE=C:\DOS\VDISK.SYS 2048 /E
```

This will create a two Mb virtual disk in extended Memory and assign the next available drive letter on your system. If you have one hard disk named C:, the virtual disk will be D:.

You may have other software that would like to access the extended memory in your computer such as Microsoft Windows and/or any of the Microsoft Language products. You may need to use HIMEM.SYS or some other Extended Memory Specification (XMS) memory manager and then load RAMDRIVE.SYS instead of the VDISK.SYS driver. An XMS memory manager lets several pieces of software (e.g., a virtual disk and a disk cache, see below) use extended memory simultaneously without conflict, while VDISK.SYS assumes it is the only one using extended memory and will step on any other software that gets in its way. The entries in your CONFIG.SYS file might look like this.

```
DEVICE=C:\DOS\HIMEM.SYS
DEVICE=C:\DOS\RAMDRIVE.SYS 2048
```

The pathnames in the above commands should be modified to match your system, for example, if the HIMEM.SYS device driver is in a directory other than C:\DOS, that directory name should be substituted for C:\DOS in the above commands.

Note that starting with DOS version 5.00, HIMEM.SYS is installed by default.

If you have more extended memory available, you may wish to install a disk cache. A disk cache increases hard disk performance by retaining or "caching" a large amount of data in memory after it has been read from the disk. The next time data is needed from the disk, the software checks the cache first, if it is there it can be accessed very quickly, if it is not, it reads it from the disk, again caching this new data while discarding the oldest data in the cache. Some caching software tries to "look ahead" and anticipate the data that will be requested next. Caching software can bring significant performance increases but this depends on the way that the application program will access the data. Caches generally show the largest performance increases with applications that perform random access searches for small records in moderate to large data files. SUDSPLOT accesses SUDS data files in this way when stations will be ordered by epicentral distance and a large disk cache can speed things up quite a bit. The only drawback to using a disk cache with SUDSPLOT is that SUDS files can grow to be rather large (5 to 8 megabytes) and as such require a very large cache to see significant perform increases. If you will be generating very large SUDS files, I would suggest a minimum of 4 megabytes for a disk cache.

The use of a virtual disk for temporary files will yield the biggest performance boost, so don't sacrifice it for a disk cache. If you want both, I would recommend that you use an XMS memory manager such as HIMEM.SYS from Microsoft or QEMM386.SYS from Quarterdeck Office Systems to manage your extended memory and then install a well behaved virtual disk driver such as RAMDRIVE.SYS and a disk cache such as SMARTDRV.SYS or SMARTDRV.EXE.

One last note about memory. XMS memory is extended memory that is managed by a XMS memory manager. This should not be confused with EMS (expanded memory specification) memory, known simply as "expanded memory". Extended memory is available only on machines that use a 80286 or later processor and is just linear addressable memory beyond 1 megabyte. The processor addresses this memory with 32 bit addresses and the XMS memory manager basically deals with avoiding conflicts between software that access this memory. Expanded memory uses a paging scheme to map 64 kb chunks of extended memory to a page frame below 1 megabyte allowing the processor access to it with 20 bit addresses, but EMS memory is significantly slower to access than extended memory because in addition to managing conflicts, the EMS memory manager must manage this paging process. Although most virtual disk drivers and disk caches can use expanded memory, I strongly recommend that you use extended memory wherever possible.

Using SUDSPLOT

2.0 Command line syntax

SUDSPLOT is controlled from the DOS command line using the following syntax:

```
Usage: SUDSPLOT [switches] SUDSfile [hypofile] [switches]

Switches:
/An - Plot phase arrivals with weight <= n. (5)
/Bn - Baseline, subtract n sample average from trace. (OFF)
/Jn - Jump, start plotting n seconds into trace. (0)
/Ln - Length, plot n seconds of trace. (ALL)
/Pn - Plot mode; 1=screen only, (2)=screen&hardcopy, 3=hardcopy only.
/S  - Prompt before saving plots. (OFF)
/Tn - Plot n traces per page. (AUTO)
/Mn - Maximum amplitude magnification. (1X)
/X  - Windowing decimation. (OFF)
/Xn - Simple decimation, plot every nth sample. (OFF)
/Ifilespec - Alternate .INI file (SUDSUTIL.INI)

SUDSfile - SUDS 1.x data file.
hypofile - HYPO71x output file. (same name as SUDSfile w/.PRT ext.)

See also: [PLOTX] and [SUDSPLOT] sections in SUDSUTIL.INI.
() indicates default value, arguments are not case sensitive.
```

SUDSPLOT requires at least a SUDS file specification on the command line. It will look for a .PRT file with the same path and base name as the SUDS input file by default. It this file is found, the program will extract event and station information from it and plot the traces in order by epicentral distance with phase arrivals and coda duration's marked on the plot. If the .PRT file is not found, the traces will be plotting in the order that they appear in the file.

2.1 Exclude phase arrivals by pick weight option

Form: /A*n*

> *n* = Arrivals with pick weights greater than *n* will be excluded from the plot.

This option is used to filter out traces with pick weights greater than *n*. n should be an integer between 0 and 5. Traditionally, pick weights range from 0 to 4. A weight of 4 meaning that you have no confidence in the pick. A weight of 5 is used by SUDSPICK to mark the trace as unpicked but keep it listed in the phase file so that HYPO71PC will calculate the P phase arrival and epicentral distance.

2.2 Baseline option

Form: /B*n*

n = Subtract the mean of the first *n* samples from each sample in each trace.

This option is used to remove a DC offset from the trace. *n* should be a positive integer. A simple average of the first *n* samples in the trace (starting from the initial sample time, not the first sample plotted when using the /J option) is subtracted from all samples in the trace before plotting.

2.3 Jump and length options

Form: /J*n* and /L*n*

n = Number of seconds to jump or plot, a positive real number.

The /J option allows you to specify in seconds from the initial sample time (IST) where the plot will begin. If no jump value is specified, plotting begins at the IST.

The /L option allows you to specify the number of seconds to plot. If no length value is specified, the entire trace will be plotted.

2.4 Plot mode option

Form: /P*n*

n = Plot mode.

The /P option allows you to specify which method SUDSPLOT will use to generate plots. *n* = 1 means the plots will be generated on the display only. This is the fastest mode, but no hardcopy is generated. *n* = 2, which is the default mode, means that plots will be generated on the display and hardcopy will be generated. *n* = 3 is "batch" mode, hardcopy only will be generated.

2.5 Prompt before saving plot option

Form: /S

/S = Prompt before save or hardcopy of plot.

This simply causes SUDSPLOT to pause once the plot is generated and prompt you as to whether or not you wish to save the plot.

2.6 Traces per page option

Form: /T*n*

n = Number of trace to be plotted per page.

This option specifies the number of traces that will be plotted per page. By default, SUDSPLOT will look at the data file and if available, the .PRT file and automatically determine the optimum number for you. SUDSPLOT can plot between 1 and 32 traces per page.

2.7 Amplitude axis scaling magnification option

Form: /M*n*

n = Maximum amplitude scale factor

By default, SUDSPLOT will plot each trace with absolute amplitude, *i.e.*, full scale is equal to the maximum possible sample value. This option allows you to specify a maximum magnification of amplitude to be performed as each trace is plotted. Suppose you wish plot traces where full scale equals the peak sample value in the trace, but you do not want dead traces (traces with very low peak values) to be magnified to full scale. If you specify /M10 on the command line, SUDSPLOT will magnify the amplitude of each trace so that full scale equals peak sample value so long as the magnification does not exceed 10×, no trace will be magnified more than 10×.

2.8 Decimation option

Form: /X1 or /X*n*

/X1 = No decimation.

/X*n* = Simple decimation, plot every *n*th sample.

By default, SUDSPLOT will automatically determine the best decimation factor and perform windowing decimation (minimum and maximum preserved) to increase plotting performance. The program determines the factor based on the resolution of the output device and the number of samples to plot. This option can bring dramatic performance increases, particularly on large data files.

If /X*n* is specified, *n* must be a positive integer. SUDSPLOT will perform simple decimation by a factor on *n* as it plots the traces. For example, if you specify /X3, the program will simply plot every third sample.

2.9 Alternate .INI file option

Form: */I*filespec*

filespec = The file specification of the .INI that SUDSPLOT should use instead of SUDSUTIL.INI.

By default, SUDSPLOT will look in its "home" directory (*i.e.*,the directory where SUDSPLOT.EXE is located) for a file name SUDSUTIL.INI. This option allows you to specify a different .INI. See also, section 1.2 for more about SUDSUTIL.INI.

2.10 Input file specification

Inputfile is the SUDS data file specification. The input file must be demultiplexed and is assumed to have the extension .DMX if an explicit extension is not provided on the command line. It is worthwhile to note that SUDSPLOT expands all input file specs to fully qualified file specifications, for example, if the current working directory is C:\DATA and you enter SUDSPLOT 90122105 <return>, SUDSPLOT expands the input filespec to C:\DATA\90122105.DMX. Any DOS legal partial filespec may be used as well, the input filespec ..\OLDDATA\TEST, would be expanded to C:\OLDDATA\TEST.DMX.

2.11 .PRT file specification

Form: inputfile *hypofile*

hypofile = The station list file specification.

By default, SUDSPLOT will look for a file with the same path and base name as the input SUDS file with an extension of .PRT. This file is created by HYPO71PC and should contain hypocenter and event information and a list of stations ordered by distance from the hypocenter. SUDSPLOT uses the information in this file to plot only these stations in this order and plot phase arrivals and coda duration's. If this file does not exist, waveforms for all stations contained in the data file will be plotted in the order that they appear in the file. This option allows the user to specify a different station list file.

Examples

3.0 Plot annotation

SUDSPLOT generates plots in "portrait" orientation, i.e., plots are viewed with the page upright. The top line of the plot consists of the time of the first sample on the plot followed by the fully qualified file specification of the input data file.

If SUDSPLOT found the .PRT file for the data, a second line will be added just below the top line that contains "event information". This line starts with the origin time, then latitude, longitude and depth of the hypocenter, magnitude, number of stations used in the hypocenter solution, followed by various statistics concerning the solution.

Figure 1 shows the marking of phase arrivals and coda duration on the traces. It also explains the notation used for station related information.

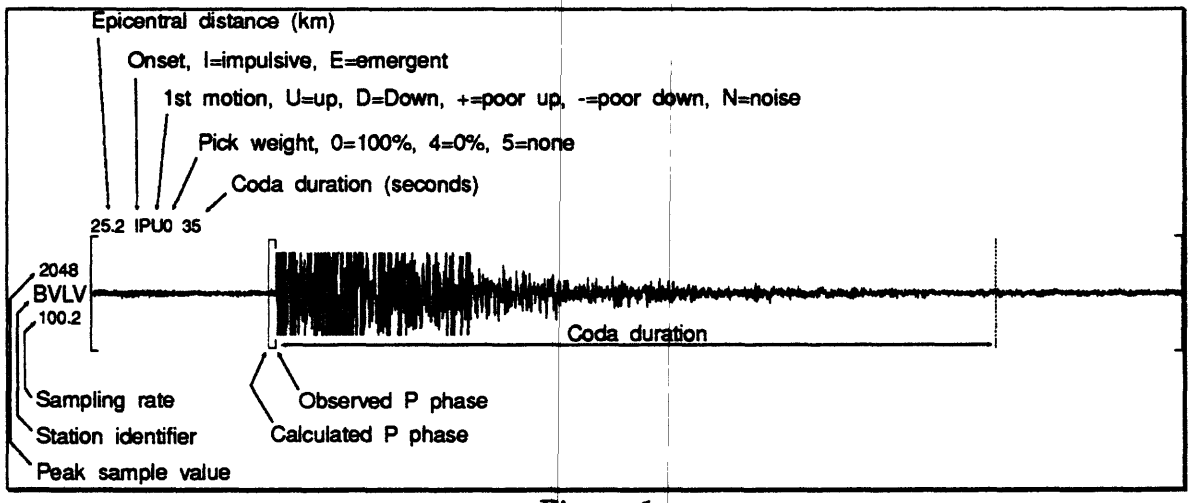


Figure 1

3.1 Sample plots

The plots on the following pages were created using many different plotting options. Figure 2 was generated from a data file that contains a 128 station subset of CalNet. This event was a fairly small local earthquake. The data file was picked automatically using SUDSPICK and then located using HYPO71X. See figure 2 for the actual command line options used to create the plot.

Figure 3 was generated from a data file that was built up from three component records of a small aftershock of the Loma Prieta earthquake. This data was recorded on portable instruments. This plot is simply the raw data from this file. See figure 3 for the actual command line options used to create the plot.

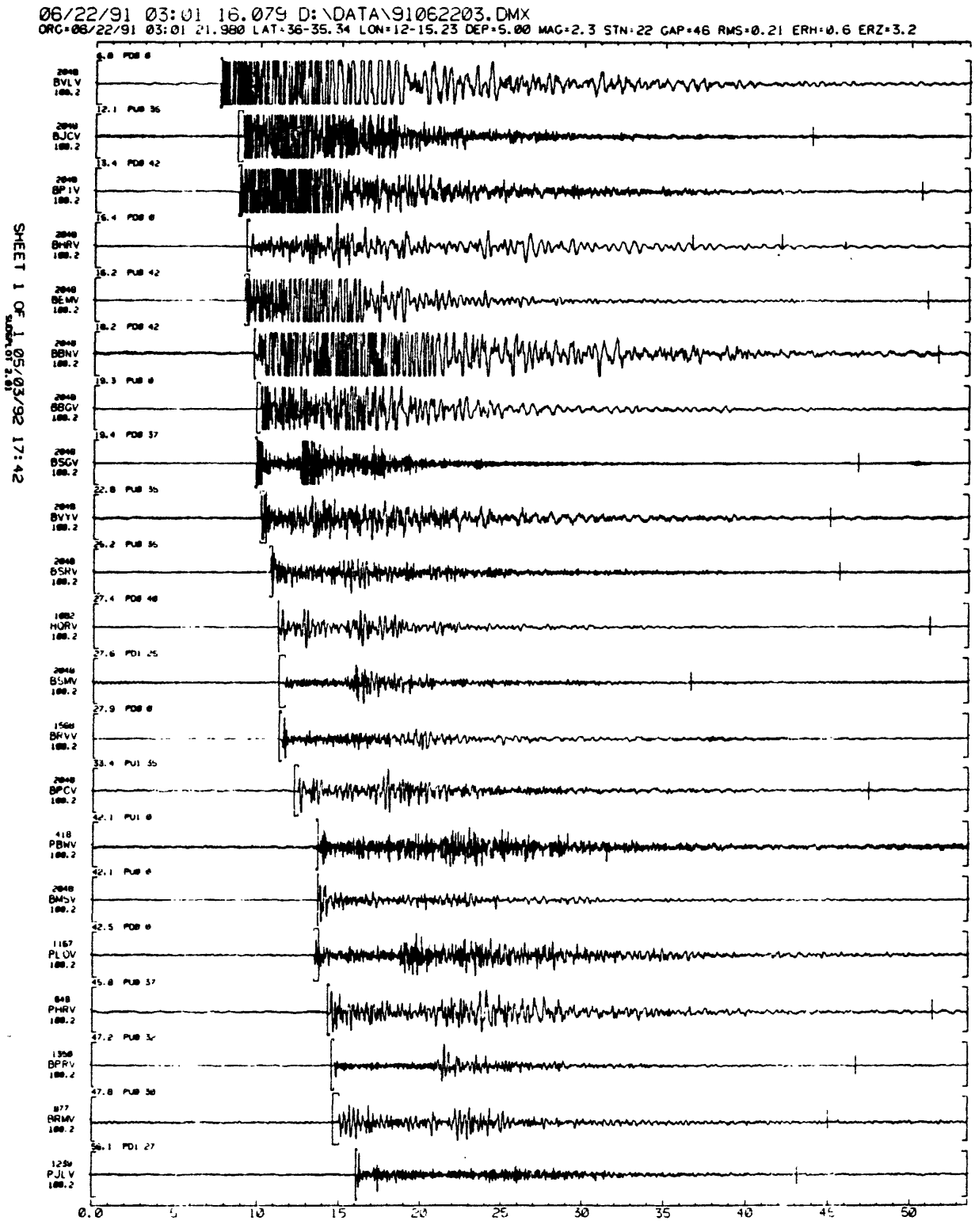


Figure 2 - The command used to create this plot was:

```
SUDSPLOT /M10 /T16 /S 91062203 91062203.PRT
```

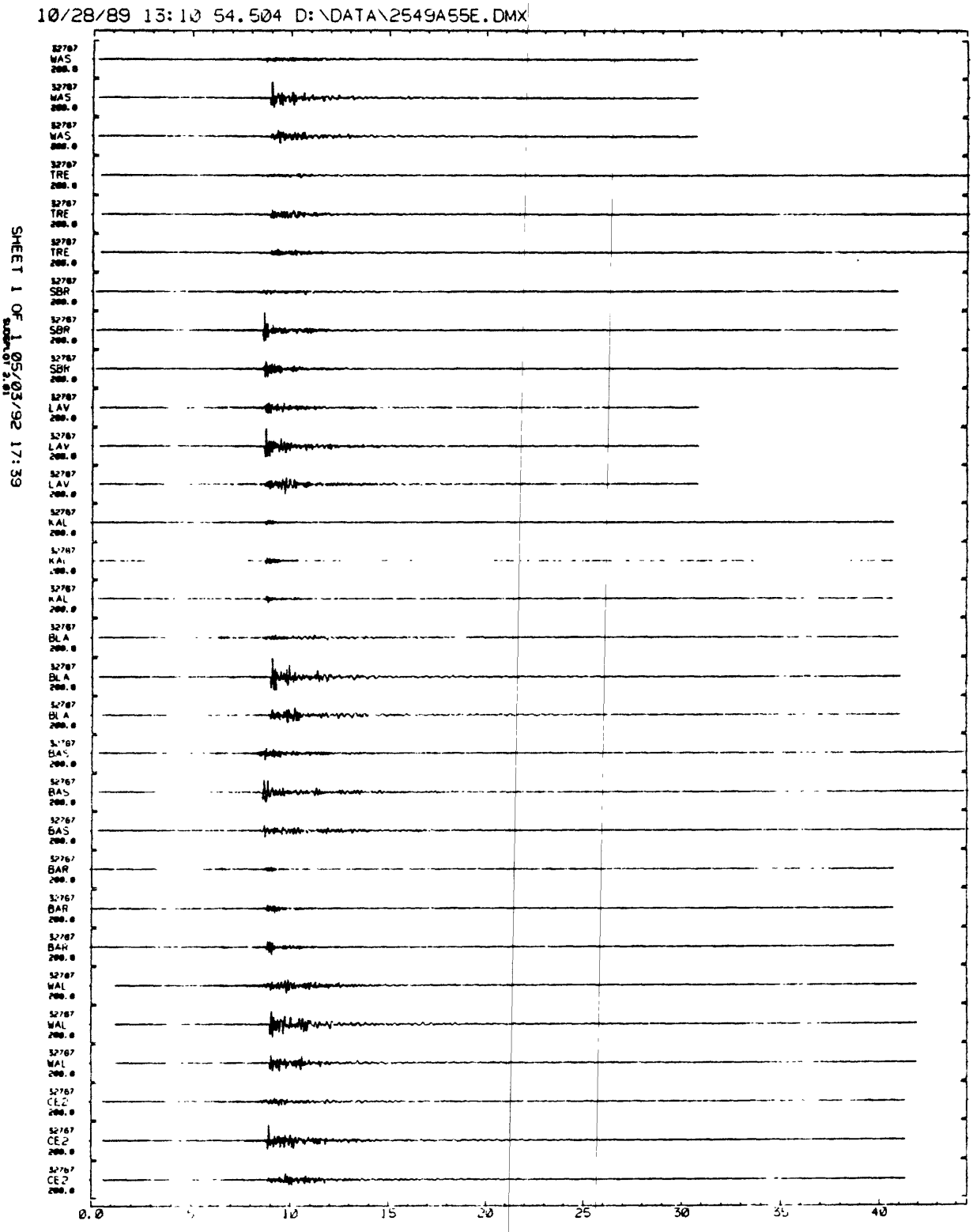


Figure 3 - The command used to create this plot was:

```
SUDSPLOT 2549A55E
```

3.3 Batch processing

SUDSPLOT returns exit codes to the calling program. From the DOS command line, the calling program will be COMMAND.COM, but if SUDSPLOT is called from within a batch file, these exit codes can be checked by using the IF ERRORLEVEL construct. This lets the batch file exit gracefully if an error occurred while processing. An exit code of 0 indicates successful execution, an exit code of 1 indicates an error has occurred (SUDSPLOT will display a message describing the error before it exits). Below are a few simple batch file that illustrate this technique:

```
@ECHO OFF
REM - PLOT.BAT, R.Banfill, 3 May 92

REM - Make sure we have a file to process
:START
IF "%1" == "" GOTO END

REM - Call SUDSPLOT to process the file
SUDSPLOT  /M10 %1

REM - If we had an error, bail out
IF ERRORLEVEL 1 GOTO ERROR

REM - Get the next filename from the command line and do it again
SHIFT
GOTO START

:ERROR
ECHO An error has OCCURRED!

:END
ECHO Done!
```

This batch file accepts multiple .DMX files as command line arguments. It checks that the replaceable parameter %1 is not NULL and then proceeds to process the files one after another. If an error occurs while processing a file, it stops and reports that an error has occurred, otherwise it SHIFTs to make the next argument become %1 and starts again, and repeats this until there are no more files.

Another advantage of calling SUDSPLOT from a batch file is that this provides a convenient way to define default behavior. In the above example the SUDSPLOT command line specifies scaling to 10X max.

Another example of these techniques is a batch file that will create a plot from every file that matches a wildcard file specification:

```
@ECHO OFF
REM - PLOT1.BAT, R.Banfill, 3 May 92

REM - Make sure we have something to do
IF "%1" == "" GOTO END

REM - Let's get to it
FOR %%f IN (%1) DO CALL XPLOT %%f %2 %3 %4 %5 %6 %7 %8 %9
```

```
:END
```

The line containing CALL XPLOT passes the name of the current file and any additional options you may have specified to the following little batch file which actually calls SUDSPLOT and traps any errors the may occur.

```
@ECHO OFF
REM - XPLOT.BAT, R.Banfill, 3 May 92

IF "%1" == "" GOTO END

REM - Call SUDSPLOT
SUDSPLOT /P3 /M10 %1 %2 %3 %4 %5 %6 %7 %8 %9
REM - If no error occurred, return
IF NOT ERRORLEVEL 1 GOTO END

ECHO An ERROR has occurred while processing %1
ECHO Press CTRL-BREAK to abort processing!
PAUSE

:END
```

These two batch files work together to process the files. The command, PLOT1 *.DMX /B200 <return> will plot every file in the current working directory that has a filename extension of .DMX and the /B200 option will be passed on to SUDSPLOT with each file.

One of the primary purposes of SUDSPLOT is to provide fully automatic processing of data as it is acquired. The following batch file is used in conjunction with SUDSMAN (SUDS File Manager) and other programs from the IASPEI library. SUDSMAN is a small program that looks across a network and retrieves data files from the machine that recorded it. The data file is placed in a working directory and a batch file named PROC.BAT is called to process the file. Once the file has been processed, control is returned to SUDSMAN and it continues looking for files to process. This batch file lets you define any type of processing that you wish, the sample below will simply locate the event and create a hardcopy record using SUDSPLOT.

```
@ECHO OFF
REM - PROC.BAT, R.Banfill, 3 May 92
REM - This batch file is called by SUDSMAN
REM - Just the filename (no path or extension) is passed

REM - Make sure we have a file!
IF "%1" == "" GOTO END

REM - Fix the time
FIXTIME 0 %1.WVM

REM - Archive the file
COPY %1.WVM C:\ARCDATA

REM - Demultiplex the file
DEMUX %1.WVM %1.DMX
```

```
REM - Plot raw data
SUDSPLOT  %1
IF ERRORLEVEL 1 GOTO ploterr

REM - Pick phases
SUDSPICK %1 /B
IF ERRORLEVEL 1 GOTO pickerr

REM - Preliminary location
CALL GOHYPO %1

REM - Plot phases and coda
SUDSPLOT /M10 /B200 /A2 %1
IF ERRORLEVEL 1 GOTO ploterr

REM - We're done, cleanup the working directory and return
ECHO y | DEL *.* > NUL
GOTO END

REM - Handle errors
:ploterr
ECHO ERROR: Plotting %1
PAUSE
GOTO END

:pickerr
ECHO ERROR: Picking %1
PAUSE

:END
```

This batch file creates 2 plots, one is just the raw data similar to the plot in figure 3, the other contains phase data, coda duration, and additional annotation similar to the plot in figure 2.

Sun 16-Aug-1992 15:48, RB

>>> Notes on SUDSPLOT 2.0 <<<

SUDSPLOT was written using Microsoft C/C++ 7.00.
The makefile provided is for use with the PWB.

Libraries required:

HSUDS.LIB - SUDS data file library version 1.31.
Available from:
Small Systems Support
2 Boston Harbor Place
Big Water, UT 84741-0205
(801) 675-5827 Voice
(801) 675-3730 FAX

L_PLOTX.LIB - PlotX graphics library version 1.12.
Available from:
Small Systems Support
2 Boston Harbor Place
Big Water, UT 84741-0205
(801) 675-5827 Voice
(801) 675-3730 FAX

The following files are included:

SUDSPLOT C	22325	04-28-92	1:37p
SUDS_DB C	10545	04-27-92	1:56p
SUDSPLOT H	2168	04-21-92	2:42p
SUDSPLOT MAK	3044	04-27-92	3:38p

```
// SUDSPLOT.H
// Copyright (c) Robert Banfill 1992. All rights reserved.

#include <suds.h>

// Data base record definition
typedef struct {
   CHAR    stn[5];   // Station name
   MS_TIME ist;      // Initial sample time
   FLOAT   rate;     // Sampling rate (SPS)
   LG_INT  leng;     // Samples in trace
   LG_INT  offset;   // Struct offset in file
   LG_INT  peak;     // Peak ABS sample value
   FLOAT   delta;    // Epicentral distance (km)
   SH_INT  fmp;      // Coda duration
   MS_TIME p_obs;    // Observed P phase arrival time
   MS_TIME s_obs;    // Observed S phase arrival time
   MS_TIME p_cal;    // Calculated P phase arrival time
   MS_TIME s_cal;    // Calculated S phase arrival time
   SH_INT  p_wht;    // P pick weight
   SH_INT  s_wht;    // S pick weight
   CHAR    p_mot;    // P phase first motion
   CHAR    s_mot;    // S phase first motion
   CHAR    p_ons;    // P phase onset
   CHAR    s_ons;    // P phase onset
} REC;

// Control params
typedef struct {
   SH_INT mode;      // Plotting mode
   SH_INT range;     // Range switch
   SH_INT arr;       // Phase arrivals with weight < arr
   SH_INT dec;       // Decimation switch
   SH_INT tpp;       // Traces per page
   FLOAT  jump;      // Jump seconds
   FLOAT  len;       // Length seconds
   FLOAT  mag;       // Maximum magnifaction factor
   SH_INT samp_max;  // Maximum sample value
   SH_INT samp_bias; // Sample bias (add to sample)
   LG_INT baseline;  // # of samples to a average of baseline
   SH_INT dump;      // Dump DB flag
} PRMS;

// Hypocenter solution
typedef struct {
   SH_INT  located;  // Flag
   MS_TIME origin;   // Origin time
   SH_INT  lat_d;    // Latitude degrees
   FLOAT   lat_m;    // Latitude minutes
   SH_INT  lon_d;    // Longitude degrees
   FLOAT   lon_m;    // Longitude minutes
   FLOAT   depth;    // Focal depth
   FLOAT   mag;      // Magnitude
   SH_INT  stns;     // # of stations used in solution
   SH_INT  gap;      // Gap
   FLOAT   rms;      // Residual
   FLOAT   err_h;    // Horizontal error
   FLOAT   err_z;    // Vertical error
} SOLUTION;

#define MAX_TPP 32
```

```
// SUDSPLOT 2
// Copyright (C) Robert Banfill 1992. All rights reserved.

// True record-sections from SUDS data

// Revision history:
// Versions 0.00 - 1.38
//    See old source code
// Version 2.00  Fri 03-Apr-1992 18:19, RB
//    Major rebuild.  Started over.

#define VERSION "2.01"

// --- Standard includes ---
#include <malloc.h>
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>

#include <plotx.h>

#include "sudsplot.h"

// --- Prototypes, Local ---
SH_INT sudsplot( SH_INT num_recs );
SH_INT plot_trace( SH_INT rec_num, FLOAT x1, FLOAT y1, FLOAT x2, FLOAT y2 );
void   time_axes( FLOAT x1, FLOAT x2, FLOAT y1, FLOAT y2 );
void   compute_times( SH_INT num_recs );
SH_INT snumf( CHAR *buf, FLOAT val );
SH_INT snumi( CHAR *buf, SH_INT val );
void   display_status( void );

// --- Prototypes, External ---
extern SH_INT build_db( void );
extern SH_INT add_rec( SH_INT num_recs );
extern void   init_rec( void );
extern void   print_recs( SH_INT num_recs );
extern void   s_upper( CHAR *buffer );

// --- Defines ---
#define X_MIN 0.0
#define Y_MIN -0.3
#define X_MAX 10.0
#define Y_MAX 8.0
#define SPACING 0.015
#define LEFT_MARGIN 0.5
#define TOP_MARGIN 0.25
#define BOT_MARGIN 0.17

#define TRACE_COLOR 7
#define PICK_COLOR 10
#define AMP_AXIS_COLOR 4
#define TIME_AXIS_COLOR 7
#define TIME_ANNOT_COLOR 14
#define ANNOT_COLOR 7

// --- Global data ---
extern REC *db, *rec;
extern SOLUTION event;

extern CHAR *progname;
extern FILE *sudsfil;
extern FILE *inifil;
extern FILE *hypofil;

extern SH_INT _pbs;

PRMS prms;

CHAR sudsfspec[_MAX_PATH];
```

```
CHAR inifspec[_MAX_PATH];
CHAR hypofspec[_MAX_PATH];

FLOAT xbase, xwid, insec;

MS_TIME early, late, start, end;

//---------------------------------------------------------------------
main( int argc, char *argv[] ) {
   register i;
   CHAR *p;
   SH_INT num_recs = 0, pbs = 0;

   progname = argv[0];

   // Defaults
   prms.mode = 2;        // Interactive
   prms.arr = 5;         // Plot phase arrivals if available
   prms.range = 0;       // Pseudo Record-Sections
   prms.dec = 0;         // Windowing decimation
   prms.tpp = 0;         // Auto traces per page
   prms.jump = 0.0;      // Start at beginning
   prms.len = 99999.0;   // Plot all
   prms.mag = 1.0;       // Absolute amplitude
   prms.baseline = 0L;   // No baseline
   prms.dump = 0;

   sudsfspec[0] = '\0';
   inifspec[0] = '\0';
   hypofspec[0] = '\0';

   pbs = 0;

   // Sign on flag
   fprintf( stderr, "SUDSPLOT - Version %.4s\n", VERSION );
   fprintf( stderr, "Copyright (c) Robert Banfill 1992. All rights reserved.\n\n" );

   // Help
   if( argv[1][0] == '?' || argv[1][1] == '?' || argc == 1 ) {
      printf( "Usage: SUDSPLOT [switches] SUDSfile [hypofile] [switches]\n" );
      printf( "\n" );
      printf( "Switches:\n" );
      printf( "   /An - Plot phase arrivals with weight <= n. (5)\n" );
      printf( "   /Bn - Baseline, subtract n sample average from trace. (OFF)\n" );
      printf( "   /Jn - Jump, start plotting n seconds into trace. (0)\n" );
      printf( "   /Ln - Length, plot n seconds of trace. (ALL)\n" );
      printf( "   /Pn - Plot mode; 1=screen only, (2)=screen & hardcopy, 3=hardcopy only.\n" );
      printf( "   /S  - Prompt before saving plots. (OFF)\n" );
      printf( "   /Tn - Plot n traces per page. (AUTO)\n" );
      printf( "   /Mn - Maximum amplitude magnifacation. (1X)\n" );
      printf( "   /X  - Windowing decimation. (OFF)\n" );
      printf( "   /Xn - Simple decimation, plot every nth sample. (OFF)\n" );
      printf( "   /Ifilespec - Alternate .INI file (SUDSUTIL.INI)\n" );
      printf( "\n" );
      printf( "   SUDSfile - SUDS 1.x data file.\n" );
      printf( "   hypofile - HYPO71x output file. (same name as SUDSfile w/.PRT ext.) \n" );
      printf( "\n" );
      printf( "See also: [PLOTX] and [SUDSPLOT] sections in SUDSUTIL.INI.\n" );
      printf( "() indicates default value, arguments are not case sensitive." );
      exit( 1 );
   }

   // Parse the command-line
   for( i=1; i<argc; i++ ) {
      if( argv[i][0] == '/' || argv[i][0] == '-' ) {
         switch( argv[i][1] ) {
            case 'A':   // Arrivals
            case 'a':
               prms.arr = atoi( &argv[i][2] );
               if( prms.arr < 0 || prms.mode > 5 ) {
                  fprintf( stderr, "ERROR: /A switch: value out of range: 0-5\n" );
                  prms.arr = 5;
```

```
            }
            break;
         case 'P':   // Plot mode
         case 'p':
            prms.mode = atoi( &argv[i][2] );
            if( prms.mode < 1 || prms.mode > 3 ) {
               fprintf( stderr, "ERROR: /P switch: value out of range: 1-3\n" );
               prms.mode = 2;
            }
            break;
         case 'J':   // Jump
         case 'j':
            prms.jump = (FLOAT)atof( &argv[i][2] );
            if( prms.jump < 0.0 ) {
               fprintf( stderr, "ERROR: /J switch: value must be positive\n" );
               prms.jump = 0.0;
            }
            break;
         case 'L':   // Length
         case 'l':
            prms.len = (FLOAT)atof( &argv[i][2] );
            if( prms.len < 0.0 ) {
               fprintf( stderr, "ERROR: /L switch: value must be positive\n" );
               prms.len = 0.0;
            }
            break;
         case 'S':   // Prompt before saving
         case 's':
            pbs = 1;
            break;
         case 'T':   // Traces per page
         case 't':
            prms.tpp = atoi( &argv[i][2] );
            if( prms.tpp < 1 || prms.tpp > 32 ) {
               fprintf( stderr, "ERROR: /T switch: value out of range: 1-32\n" );
               prms.tpp = 0;
            }
            break;
         case 'M':   // Magnification
         case 'm':
            prms.mag = (FLOAT)atof( &argv[i][2] );
            if( prms.mag < 1.0 || prms.mag > 100.0 ) {
               fprintf( stderr, "ERROR: /M switch: value out of range: 1.0-100.0\n" );
               prms.mag = 1.0;
            }
            break;
         case 'X':   // Decimation
         case 'x':
            prms.dec = atoi( &argv[i][2] );
            break;
         case 'R':   // Range
         case 'r':
            prms.range = 1;
            break;
         case 'D':   // Dump
         case 'd':
            prms.dump = 1;
            break;
         case 'B':   // Baseline
         case 'b':
            prms.baseline = atoi( &argv[i][2] );
            if( prms.baseline < 1 ) {
               fprintf( stderr, "ERROR: /B switch: value must be positive\n" );
               prms.baseline = 0;
            }
            break;
         case 'I':
         case 'i':
            _fullpath( inifspec, &argv[i][2], _MAX_PATH );
            break;
      }
   }
```

```
      else {
         if( sudsfspec[0] == '\0' ) {
            _fullpath( sudsfspec, argv[i], _MAX_PATH );
            if( strrchr( sudsfspec, '.' ) == NULL )
               strcat( sudsfspec, ".DMX" );
            s_upper( sudsfspec );
         }
         else if( hypofspec[0] == '\0' ) {
            _fullpath( hypofspec, argv[i], _MAX_PATH );
            s_upper( hypofspec );
         }
      }
   }

   if( sudsfspec[0] == '\0' ) {
      fprintf( stderr, "ERROR: no SUDS data input file specified\n" );
      exit( 1 );
   }

   // If no hypo file specified, use sudsfile with .PRT extension
   if( hypofspec[0] == '\0' ) {
      strcpy( hypofspec, sudsfspec );
      p = strrchr( hypofspec, '.' ) + 1;
      strcpy( p, "PRT" );
   }

   if( inifspec[0] == '\0' ) {
      strcpy( inifspec, progname );
      p = strrchr( inifspec, '\\' ) + 1;
      strcpy( p, "SUDSUTIL.INI" );
   }
   s_upper( inifspec );

   // Init the graphics library
   if( iniplt( inifspec, 0, 0, "\0", 0 ) != 0 )
      exit( 1 );
   _pbs = pbs;
   hardcopy( 1 );
   window( X_MIN, Y_MIN, X_MAX, Y_MAX );

   // Build the database
   if( ( num_recs = build_db( ) ) < 0 )
      exit( 1 );

   display_status( );

   // Let's plot
   if( ! sudsplot( num_recs ) )
      exit( 1 );

   exit( 0 );
}

//---------------------------------------------------------------------
SH_INT sudsplot( SH_INT num_recs ) {
   register i;
   CHAR buf[128];
   SH_INT nchar, pages, page;
   static FLOAT x1, y1, x2, y2, space, tmarg;
   MS_TIME mtime;

   // Compute earliest and latest times
   compute_times( num_recs );

   if( prms.tpp == 0 ) {
      prms.tpp = num_recs;
      for( pages=1; pages<999; pages++ ) {
         if( num_recs/pages <= MAX_TPP ) {
            prms.tpp = num_recs / pages;
            if( prms.tpp*pages < num_recs )
               prms.tpp++;
            break;
```

```
      }
   }
}
else
   pages = ((num_recs - 1) / prms.tpp) + 1;

// Jump and length
start = early;
end = late;
if( prms.jump > 0.0 ) {
   if( start + (MS_TIME)prms.jump < end )
      start += (MS_TIME)prms.jump;
}
if( prms.len != 99999.0 ) {
   if( start + (MS_TIME)prms.len < end )
      end = start + (MS_TIME)prms.len;
}

mtime = get_mstime( );

// Plot each page
for( page=1; page<=pages; page++ ) {
   if( plots( prms.mode ) != 0 )
      exit( 1 );

   space = SPACING;
   tmarg = TOP_MARGIN;

   if( event.located ) {
      tmarg += .1;
      space += .07;
   }

   // x2 = tmarg + (space / 2.0);
   x2 = tmarg + space;
   y1 = LEFT_MARGIN;
   y2 = Y_MAX - 0.03;
   xwid = ((X_MAX - X_MIN - tmarg - BOT_MARGIN ) / prms.tpp ) - space;
   x1 = x2 + xwid;

   newpen( ANNOT_COLOR );
   sprintf( buf, "%s %s", list_mstime( early, 4 ), sudsfspec );
   symbol( X_MIN+.101, 0.0, .1, buf, 90.0, strlen( buf ) );
   if( event.located ) {
      sprintf( buf, "ORG=%s LAT=%02d-%05.2f LON=%02d-%05.2f DEP=%.2f MAG=%.1f STN=%d GAP=%(
         list_mstime( event.origin, 4 ), event.lat_d, event.lat_m,
         event.lon_d, event.lon_m, event.depth, event.mag, event.stns,
         event.gap, event.rms, event.err_h, event.err_z );
      symbol( X_MIN+.21, 0.0, .07, buf, 90.0, strlen( buf ) );
   }

   sprintf( buf, "SHEET %d OF %d %.14s", page, pages, list_mstime( mtime, 4 ) );
   symbol( 2.875, -0.15, .08, buf, 0.0, -strlen( buf ) );
   sprintf( buf, "SUDSPLOT %.4s", VERSION );
   symbol( 2.875, -0.24, .05, buf, 0.0, -strlen( buf ) );

   time_axes( X_MAX, tmarg-0.07, y1, y2 );

   for( i=0; i<prms.tpp; i++ ) {
      if( i+((page-1)*prms.tpp) >= num_recs )
         break;

      if( prms.mode == 3 )
         printf( "Processing sheet %d ", page );

      if( ! plot_trace( i+((page-1)*prms.tpp), x1, y1, x2, y2 ) )
         return( 0 );

      newpen( ANNOT_COLOR );
      nchar = snumi( buf, (SH_INT)rec->peak );
      symbol( xbase-0.085, LEFT_MARGIN/2.0, .045, buf, 90.0, -nchar );
      symbol( xbase, LEFT_MARGIN/2.0, .06, rec->stn, 90.0, -4 );
```

```
            nchar = snumf( buf, rec->rate );
            symbol( xbase+0.085, LEFT_MARGIN/2.0, .045, buf, 90.0, -nchar );

            if( event.located ) {
               sprintf( buf, "%.1f %cP%c%d %d", rec->delta, rec->p_ons,
                  rec->p_mot, rec->p_wht, rec->fmp );
               symbol( x2-.02, LEFT_MARGIN, .045, buf, 90.0, strlen(buf) );
            }

            x1 += xwid + space;
            x2 += xwid + space;
         }
         if( prms.mode == 1 || (prms.mode == 2 && _pbs) )
            getch( );

         plot( 0.0, 0.0, 999 );
      }

      return( 1 );
   }

   //---------------------------------------------------------------------
   SH_INT plot_trace( SH_INT rec_num, FLOAT x1, FLOAT y1, FLOAT x2, FLOAT y2 ) {
      register i;
      FLOAT xstep, ystep, ybase, secs, ywid, ipp, spp, t, x, y, o_in, c_in, f_in;
      SH_INT type, pixels, maxs, mins, offset, clip;
      LG_INT l, n, length, t_length, t_sam, sum, sam, s_sam, e_sam, stop;

      CHAR _huge *ptr;
      SH_INT _huge *data;
      SUDS_DESCRIPTRACE _huge *dt;

      // Get the current record from database
      rec = db + rec_num;

      // Find the trace data in SUDS file
      st_seek( sudsfil, rec->offset, 0 );
      t_length = st_get( &ptr, &type, &length, sudsfil );

      if( type != DESCRIPTRACE ) {
         plot( 0.0, 0.0, 999 );
         fprintf( stderr, "\nERROR: input file out of sync: %s\n", sudsfspec );
         return( 0 );
      }
      dt = (SUDS_DESCRIPTRACE _huge *)ptr;
      if( (rec->ist != dt->begintime+dt->time_correct) ||
          (strncmp( rec->stn, dt->dt_name.st_name, 4 ) != 0) ||
          (rec->leng != dt->length) ) {
         plot( 0.0, 0.0, 999 );
         fprintf( stderr, "\nERROR: database out of sync with input file: %s\n", sudsfspec );
         return( 0 );
      }
      data = (SH_INT _huge *)(dt+1);

      if( prms.mode == 3 )
         printf( "station %s    \r", rec->stn );

      // Baseline
      if( prms.baseline != 0 ) {
         sum = 0L;
         for( l=0; l<prms.baseline; l++ )
            sum += *(data+l)+prms.samp_bias;
         offset = prms.samp_bias - (SH_INT)(sum / prms.baseline);
      }
      else
         offset = prms.samp_bias;

      // Remove offset and recalc peak value
      if( offset != 0 ){
         rec->peak = 0;
         for( l=0; l<rec->leng; l++ ){
            sam = (LG_INT)*(data+l) + (LG_INT)offset;
```

```
      if( sam > 32767L )
         *(data+l) = 32767;
      else if( sam < -32767L )
         *(data+l) = -32767;
      else
         *(data+l) = (SH_INT)sam;
      if( abs(*(data+l)) > rec->peak )
         rec->peak = abs(*(data+l));

   }
}

// Magnifacation
if( (FLOAT)prms.samp_max / (FLOAT)(rec->peak > 0 ? rec->peak : 1) > prms.mag )
   rec->peak = prms.samp_max / (SH_INT)prms.mag;

// Plotting constants
ywid  = y2 - y1;
if( prms.mode > 1 )
   pixels = (SH_INT)(ywid * 300.0);
else
   pixels = video_vpixels;
t_sam = (LG_INT)((end - start) * rec->rate);
xbase = x2 + (xwid / 2.0);
xstep = (xwid / 2.0) / ( rec->peak == 0 ? 1.0 : (FLOAT)rec->peak );
ystep = (y2 - y1) / (FLOAT)t_sam;
s_sam = (LG_INT)(((start-rec->ist) < 0.0 ? 0.0 : (start-rec->ist)) * rec->rate);
e_sam = (LG_INT)(((end-rec->ist) > late ? late : (end-rec->ist)) * rec->rate);
e_sam = (e_sam > rec->leng ? rec->leng : e_sam );
ybase = y1 + ( ((rec->ist-start) < 0.0 ? 0.0 : ((rec->ist-start) * rec->rate) * ystep) );
spp   = (FLOAT)t_sam / pixels;
if( prms.dec == 0 && spp < 2.0 )
   prms.dec = 1;

// Plot the trace
newpen( TRACE_COLOR );
plot( xbase-(xstep*(FLOAT)*data), ybase, 3);
if( prms.dec > 0 ) {
   n = 0;
   for( l=s_sam; l<e_sam; l+=prms.dec ) {
      plot( xbase-(xstep*(FLOAT)*(data+l)), ybase+(ystep*(FLOAT)(n++)), 2 );
   }
}
else {
   ipp = ywid / (FLOAT)pixels;
   t = (FLOAT)(s_sam-1);
   n = s_sam;
   stop = e_sam-(LG_INT)spp-1;
   for( y=0.0; y<=ywid; y+=ipp ) {
      if( n >= stop )
         break;
      mins = 32767;
      maxs = -32767;
      t += spp;
      while( n < (LG_INT)t ) {
         if( mins > *(data+n) )
            mins = *(data+n);
         if( maxs < *(data+n) )
            maxs = *(data+n);
         n++;
      }
      n--;
      plot( xbase-(xstep*(FLOAT)mins), ybase+y, 3 );
      plot( xbase-(xstep*(FLOAT)maxs), ybase+y, 2 );
   }
}

// Free the trace buffer
st_free( ptr, t_length );

// Plot amplitude axes
if( ! prms.range ) {
```

```
      newpen( AMP_AXIS_COLOR );

      // Left amplitude axis
      plot( x1, y1+0.05, 3 );
      plot( x1, y1, 2 );
      plot( xbase, y1, 2 );
      plot( xbase, y1-0.02, 2 );
      plot( xbase, y1, 3 );
      plot( x2, y1, 2 );
      plot( x2, y1+0.05, 2 );

      // Right amplitude axis
      plot( x2, y2-0.05, 3 );
      plot( x2, y2, 2  );
      plot( xbase, y2, 2 );
      plot( xbase, y2+0.02, 2 );
      plot( xbase, y2, 3 );
      plot( x1, y2, 2 );
      plot( x1, y2-0.05, 2 );
   }

   // Plot picks
   if( event.located ) {
      newpen( PICK_COLOR );

      c_in = y1 + ((FLOAT)(rec->p_cal-start) * insec);
      if( rec->p_obs == 0.0 ) {
         if( c_in > y1 && c_in < y2 ) {
            i = 0;
            for( x=x2-0.03; x<=x1+0.03; x+=((x1-x2)/11) ) {
               i++;
               plot( x, c_in, (i%2 == 0 ? 2 : 3) );
            }
         }
      }
      else {
         o_in = y1 + ((FLOAT)(rec->p_obs-start) * insec);
         if( o_in > y1 && o_in < y2 ) {
            plot( x2-.03, o_in, 3 );
            plot( x2-.005, o_in, 2 );
            plot( x1+.03, o_in, 3 );
            plot( x1+.005, o_in, 2 );
         }
         if( c_in > y1 && c_in < y2 ) {
            plot( x2-.03, c_in, 3 );
            plot( x1+.03, c_in, 2 );
         }
         plot( x2-0.03, (c_in < y1 ? y1 : (c_in > y2 ? y2 : c_in)), 3 );
         plot( x2-0.03, (o_in < y1 ? y1 : (o_in > y2 ? y2 : o_in)), 2 );
         plot( x1+0.03, (c_in < y1 ? y1 : (c_in > y2 ? y2 : c_in)), 3 );
         plot( x1+0.03, (o_in < y1 ? y1 : (o_in > y2 ? y2 : o_in)), 2 );
      }

      if( rec->fmp != 0 ) {
         f_in = y1 + ((FLOAT)((rec->p_obs+(DOUBLE)rec->fmp)-start) * insec);
         if( f_in > y1 && f_in < y2 ) {
            plot( x2+.1, f_in, 3 );
            plot( x1-.1, f_in, 2 );
         }
      }
   }
   return( 1 );
}

//---------------------------------------------------------------------
void time_axes( FLOAT x1, FLOAT x2, FLOAT y1, FLOAT y2 ) {
   register i;
   FLOAT x, z, taxbeg, taxend, taxlen;
   SH_INT taxtic, nchar;
   CHAR taxstr[16];

   x = x1;
```

```
   taxlen = (FLOAT)(end - start);
   taxbeg = (FLOAT)(start - early);
   taxend = taxbeg + taxlen;
   insec = (y2 - y1) / (taxlen < 1.0 ? 1.0 : taxlen);

   if( taxlen > 180.0 )
      taxtic = 25;
   else if( taxlen > 60.0 )
      taxtic = 10;
   else if( taxlen > 12.0 )
      taxtic = 5;
   else
      taxtic = 1;

   newpen( TIME_ANNOT_COLOR );
   nchar = snumf( taxstr, taxbeg );
   symbol( x-0.04, y1, 0.07, taxstr, 90.0, -nchar );

   newpen( TIME_AXIS_COLOR );
   plot( x-0.15, y1, 3 );
   plot( x-0.1, y1, 2 );
   for( i=(SH_INT)taxbeg+1; i<=(SH_INT)taxend+1; i++ ) {
      z = (((FLOAT)i - taxbeg) * insec) + y1;
      if( z >= y2 ) {
         plot( x-0.1, y2, 2 );
         plot( x-0.15, y2, 2 );
         break;
      }
      plot( x-0.1, z, 2 );
      if( i % taxtic == 0 ) {
         plot( x-0.15, z, 2 );
         newpen( TIME_ANNOT_COLOR );
         nchar = snumi( taxstr, i );
         symbol( x-0.04, z, 0.07, taxstr, 90.0, -nchar );
         newpen( TIME_AXIS_COLOR );
      }
      else {
         plot( x-0.125, z, 2 );
      }
      plot( x-0.1, z, 3 );
   }
   newpen( TIME_AXIS_COLOR );
   x = x2;
   plot( x+0.05, y1, 3 );
   plot( x, y1, 2 );
   for( i=(SH_INT)taxbeg+1; i<=(SH_INT)taxend+1; i++ ) {
      z = (((FLOAT)i - taxbeg) * insec) + y1;
      if( z >= y2 ) {
         plot( x, y2, 2 );
         plot( x+0.05, y2, 2 );
         break;
      }
      plot( x, z, 2 );
      if( i % taxtic == 0 )
         plot( x+0.05, z, 2 );
      else {
         plot( x+0.025, z, 2 );
      }
      plot( x, z, 3 );
   }

   return;
}

//-----------------------------------------------------------------------
SH_INT snumf( CHAR *buf, FLOAT val ) {
   register i;
   CHAR *p;

   sprintf( buf, "%-6.1f", val );

   i = 0;
```

```
   for( p = buf; *p; p++ ) {
      if( *p == ' ' )
         break;
      i++;
   }

   return( i );
}

//-------------------------------------------------------------------------
SH_INT snumi( CHAR *buf, SH_INT val ) {
   register i;
   CHAR *p;

   sprintf( buf, "%d", val );

   i = 0;
   for( p = buf; *p; p++ )
      i++;

   return( i );
}


//-------------------------------------------------------------------------
void compute_times( SH_INT num_recs ) {
   register i;
   MS_TIME time;

   early = 2147472000.0;
   late = -2147472000.0;

   for( i=0; i<num_recs; i++ ) {
      rec = db+i;
      if( rec->ist < early )
         early = rec->ist;
      time = rec->ist + ((DOUBLE)rec->leng / (DOUBLE)rec->rate);
      if( time > late )
         late = time;
   }

   return;
}

//-------------------------------------------------------------------------
void display_status( void ) {
   printf( "\nDecimation:    " );
   switch( prms.dec ) {
      case 0:
         printf( "Windowing, min & max preserved\n" );
         break;
      case 1:
         printf( "None\n" );
         break;
      default:
         printf( "Simple decimation, factor = %d.\n", prms.dec );
         break;
   }
   printf( "Jump seconds: %.1f\n", prms.jump );
   printf( "Plot seconds: " );
   if( prms.len == 99999.0 )
      printf( "All\n" );
   else
      printf( "%.1f\n", prms.len );
   printf( "Scaling:      " );
   if( prms.mag == 1.0 )
      printf( "Absolute amplitude\n" );
   else
      printf( "Up to %.1fX magnifacation on amplitude axis\n", prms.mag );
   printf( "Baseline:     " );
   if( prms.baseline == 0L )
      printf( "None\n" );
```

```
   else
      printf( "%ld samples\n", prms.baseline );
   printf( "Plot mode:     " );
   switch( prms.mode ) {
      case 1:
         printf( "Display only\n" );
         break;
      case 2:
         printf( "Display and hardcopy\n" );
         break;
      case 3:
         printf( "Hardcopy only\n" );
         break;
   }
   printf( "Traces/page:  " );
   if( prms.tpp == 0 )
      printf( "Best fit\n" );
   else
      printf( "%d\n", prms.tpp );
   if( event.located )
      printf( "Phase arrivals with weight <= %d will be plotted\n", prms.arr );

   printf( "\n" );

   return;
}

//-----------------------------------------------------------------------
LG_INT die( SH_INT in ) {

   exit( in );
}
```

```
// SUDS_DB.C
// Copyright (C) Robert Banfill 1992. All rights reserved.

// SUDS Database routines for SUDSPLOT 2.00

#include <malloc.h>
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>

#include "sudsplot.h"

// --- Prototypes, Local ---
SH_INT build_db( void );
SH_INT hypo_extract( SH_INT num_recs );
SH_INT add_rec( SH_INT num_recs );
void   init_rec( void );
void   dump_db( SH_INT num_recs );
void   s_upper( CHAR *buffer );
void   decode( CHAR *string, SH_INT pos, CHAR *fmt, void *val );
SH_INT compare( REC *rec1, REC *rec2 );
SH_INT pack_db( SH_INT num_recs );

// --- Global data ---
extern CHAR sudsfspec[_MAX_PATH];
extern CHAR inifspec[_MAX_PATH];
extern CHAR hypofspec[_MAX_PATH];

extern PRMS prms;
extern MS_TIME evn_time;

FILE *sudsfil;
FILE *inifil;
FILE *hypofil;

REC *db, *rec;
SOLUTION event;

//---------------------------------------------------------------------------
SH_INT build_db( void ) {
   SH_INT typ, num_recs, sample;
   LG_INT l, inp, len;

   CHAR _huge *ptr;
   SH_INT _huge *data;
   SUDS_DESCRIPTRACE _huge *dt;

   sudsfil = st_open( sudsfspec, "r+b" );

   printf( "Building database ->  %s", sudsfspec );

   num_recs = 0;

   while( ( inp = st_get( &ptr, &typ, &len, sudsfil ) ) != EOF ) {
      switch( typ ) {
         case MUXDATA:
            fprintf( stderr, "\nERROR: %s contains multiplexed data!\n", sudsfspec );
            return( -1 );
            break;
         case DESCRIPTRACE:
            dt = (SUDS_DESCRIPTRACE _huge *)ptr;
            num_recs = add_rec( num_recs );

            strncpy( rec->stn, dt->dt_name.st_name, 4 );
            rec->stn[4] = '\0';
            rec->offset = st_tell( sudsfil )-1;
            rec->ist = dt->begintime+dt->time_correct;
            rec->rate = dt->rate+dt->rate_correct;
            rec->leng = dt->length;
            switch( dt->datatype ) {
```

```
                case 's':
                   prms.samp_bias = -2048;
                   prms.samp_max = 2048;
                   break;
                case 'u':
                   prms.samp_bias = -32767;
                   prms.samp_max = 32767;
                   break;
                case 'q':
                   prms.samp_max = 2048;
                   break;
                case 'i':
                   prms.samp_max = 32767;
                   break;
                default:
                   fprintf( stderr, "\nERROR: unsupported data type: %c in %s\n",
                      dt->datatype, sudsfspec );
                   exit( 1 );
                   break;
             }
             data = (SH_INT _huge *)(dt+1);
             for( l=0; l<dt->length; l++ ) {
                sample = abs( *(data+l) + prms.samp_bias );
                if( sample > rec->peak )
                   rec->peak = sample;
             }
             break;
          }
          st_free( ptr, inp );
       }
       printf( "\rInput SUDS data file: %s\n", sudsfspec );

       if( hypofspec[0] != '\0' ) {
          if( ( hypofil = fopen( hypofspec, "r" ) ) == NULL ) {
             printf( "No hypocenter data found.\n" );
          }
          else {
             printf( "Extracting hypocenter data -> %s", hypofspec );

             num_recs = hypo_extract( num_recs );

             printf( "\rHypocenter data file: %s        \n", hypofspec );
             fclose( hypofil );
          }
       }

       if( prms.dump )
          dump_db( num_recs );

       return( num_recs );
    }

    // ---------------------------------------------------------------------
    SH_INT hypo_extract( SH_INT num_recs ) {
       register i;
       SH_INT d_flag = 0, s_flag = 0;
       SH_INT yr, mn, dy, hr, mi;
       FLOAT sc;
       LG_INT line;
       CHAR buf[256];

       line = 1;
       event.located = 0;

       while( fgets( buf, 255, hypofil ) != NULL ) {
          if( ! d_flag ) {
             if( strncmp( buf, "  DATE", 6 ) == 0 ) {
                d_flag = 1;
                if( fgets( buf, 255, hypofil ) != NULL ) {
                   decode( buf, 1, "%2d", &yr );
                   decode( buf, 3, "%2d", &mn );
                   decode( buf, 5, "%2d", &dy );
```

```
            decode( buf, 8, "%2d", &hr );
            decode( buf, 10, "%2d", &mi );
            decode( buf, 13, "%5f", &sc );
            event.origin = make_mstime( yr+1900, mn, dy, hr, mi, (DOUBLE)sc );
            decode( buf, 19, "%2d", &event.lat_d );
            decode( buf, 22, "%5f", &event.lat_m );
            decode( buf, 28, "%2d", &event.lon_d );
            decode( buf, 32, "%5f", &event.lon_m );
            decode( buf, 38, "%6f", &event.depth );
            decode( buf, 46, "%5f", &event.mag );
            decode( buf, 52, "%2d", &event.stns );
            decode( buf, 58, "%3d", &event.gap );
            decode( buf, 63, "%5f", &event.rms );
            decode( buf, 69, "%4f", &event.err_h );
            decode( buf, 74, "%4f", &event.err_z );
            event.located = 1;
         }
         else
            return( 0 );
      }
      else
         continue;
   }
   else if( ! s_flag && strncmp( buf, "  STN", 5 ) == 0 ) {
      s_flag = 1;
      continue;
   }
   else if( s_flag ) {
      for( i=0; i<num_recs; i++ ) {
         rec = db+i;
         if( strncmp( &buf[1], rec->stn, 4 ) == 0 ) {
            decode( buf, 6, "%5f", &rec->delta );
            decode( buf, 23, "%1d", &rec->p_wht );
            decode( buf, 25, "%2d", &hr );
            decode( buf, 27, "%2d", &mi );
            if( rec->p_wht < 5 ) {
               rec->p_ons = buf[20];
               rec->p_mot = buf[22];
               decode( buf, 36, "%5f", &sc );
               rec->p_obs = event.origin + (DOUBLE)sc;
            }
            decode( buf, 42, "%5f", &sc );
            rec->p_cal = event.origin + (DOUBLE)sc;
            decode( buf, 71, "%3d", &rec->fmp );

            break;
         }
      }
   }
}

// Pack the database
if( ( num_recs = pack_db( num_recs ) ) == 0 ) {
   exit( 1 );
}

// Sort database by epicentral distance
qsort( (void *)db, (size_t)num_recs, sizeof(REC), compare );

return( num_recs );
}

// ---------------------------------------------------------------------
SH_INT pack_db( SH_INT num_recs ) {
   register i;
   REC *pack, *prec;
   SH_INT pack_recs;

   // Squeeze the database based on p weight
   pack_recs = 0;
   pack = NULL;
```

```
   for( i=0; i<num_recs; i++ ) {
      rec = db+i;
      if( rec->p_wht <= prms.arr ) {
         if( ( pack = (REC *)realloc( pack, (pack_recs+1)*sizeof(REC) ) ) != NULL ) {
            prec = pack+pack_recs;
            memcpy( prec, rec, sizeof(REC) );
            pack_recs += 1;
         }
         else {
            fprintf( stderr, "\nERROR: Unable to allocate memory in pack_db( )!\n" );
            return( 0 );
         }
      }
   }

   memcpy( db, pack, pack_recs*sizeof(REC) );
   free( pack );

   return( pack_recs );
}

// ---------------------------------------------------------------------
SH_INT compare( REC *rec1, REC *rec2 ) {
   if( rec1->delta > rec2->delta )
      return( 1 );
   else if( rec1->delta < rec2->delta )
      return( -1 );
   else
      return( 0 );
}

// ---------------------------------------------------------------------
void decode( CHAR *string, SH_INT pos, CHAR *fmt, void *val ) {
   CHAR buf[256];
   SH_INT wid;

   sscanf( fmt, "%%%1d", &wid );
   memset( buf, '\0', 255 );

   strncpy( buf, string+pos, wid );
   sscanf( buf, fmt, val );

   return;
}

// ---------------------------------------------------------------------
SH_INT add_rec( SH_INT num_recs ) {

   // Add a new record to the end of the database
   // return number of records in db if successful, -1 if not

   if( ( db = (REC *)realloc( db, (num_recs+1)*sizeof(REC) ) ) != NULL ) {
      rec = db+num_recs;
      init_rec( );
      num_recs += 1;
   }
   else {
      fprintf( stderr, "\nERROR: Unable to allocate memory in add_rec( )!\n" );
      num_recs = -1;
   }
   return( num_recs );
}

// ---------------------------------------------------------------------
void init_rec( void ) {

   rec->stn[0] = '\0';
   rec->ist    = 0.0;
   rec->rate   = 0.0;
   rec->leng   = 0;
   rec->offset = -1;
   rec->peak   = 0;
```

```
   rec->delta  = 999.0;
   rec->fmp    = 0;
   rec->p_obs  = 0.0;
   rec->s_obs  = 0.0;
   rec->p_cal  = 0.0;
   rec->s_cal  = 0.0;
   rec->p_wht  = 5;
   rec->s_wht  = 5;
   rec->p_mot  = ' ';
   rec->s_mot  = ' ';
   rec->p_ons  = ' ';
   rec->s_ons  = ' ';

   return;
}

// ---------------------------------------------------------------------
void dump_db( SH_INT num_recs ) {
   register i;
   FILE *dumpfil;

   dumpfil = fopen( "SUDSPLOT.DB", "w" );

   printf( "Database dumped tc: SUDSPLOT.DB\n" );

   for( i=0; i<num_recs; i++ ) {
      rec = db+i;
      fprintf( dumpfil, "Record #%d\n", i );
      fprintf( dumpfil, "   Station:      %s\n", rec->stn );
      fprintf( dumpfil, "   IST:          %s\n", list_mstime( rec->ist, 4 ) );
      fprintf( dumpfil, "   Length:       %ld\n", rec->leng );
      fprintf( dumpfil, "   Offset:       %ld\n", rec->offset );
      fprintf( dumpfil, "   Peak:         %ld\n", rec->peak );
      fprintf( dumpfil, "   Delta:        %.1f\n", rec->delta );
      fprintf( dumpfil, "   FMP:          %d\n", rec->fmp );
      fprintf( dumpfil, "   P observed:   %s\n", list_mstime( rec->p_obs, 4 ) );
      fprintf( dumpfil, "   S observed:   %s\n", list_mstime( rec->s_obs, 4 ) );
      fprintf( dumpfil, "   P calculated: %s\n", list_mstime( rec->p_cal, 4 ) );
      fprintf( dumpfil, "   S calculated: %s\n", list_mstime( rec->s_cal, 4 ) );
      fprintf( dumpfil, "   P weight:     %d\n", rec->p_wht );
      fprintf( dumpfil, "   S weight:     %d\n", rec->s_wht );
      fprintf( dumpfil, "   P 1st motion: %c\n", rec->p_mot );
      fprintf( dumpfil, "   S 1st motion: %c\n", rec->s_mot );
      fprintf( dumpfil, "   P Onset:      %c\n", rec->p_ons );
      fprintf( dumpfil, "   S Onset:      %c\n", rec->s_ons );
      fprintf( dumpfil, "\n" );
   }

   fclose( dumpfil );

   return;
}

//---------------------------------------------------------------------
void s_upper( CHAR *buffer ) {
   CHAR *p;

   for( p = buffer; *p; p++ )
      *p = toupper( *p );
}
```

SUDSPROC & SUDSMAN

Programs to Manage the Processing of
Seismic Data Stored in SUDS Format

Version 1.0
May 1992

Robert Banfill
Small Systems Support
2 Boston Harbor Place
Big Water, Utah 84741-0205

Contents

Getting Started

1.0 Overview

SUDSPROC and SUDSMAN are components of the IASPEI software libraries. These programs are designed to automate the processing of data stored in SUDS format as defined in SUDS Version 1.31, R. Banfill, 8 March 1992 and SUDS: Seismic Unified Data System, Peter L. Ward, U.S.G.S. Open-file report 89-188, 29 March 1989.

SUDSPROC or the SUDS processing manager is an interactive tool that displays SUDS and related files on screen and allows the user to process them using a mouse. This "point and click" method is very intuitive and allows you work closely with the data.

SUDSMAN or the SUDS data file manager is designed to manage data acquisition and automated processing. This program is generally used in a networking environment and manages the movement of files from one machine to another and spawns batch files to perform processing.

Below I will outline a generic data acquisition and processing system. Note that several other programs from the IASPEI library are called by this software to perform the actual processing.

1) A program such as RTP (real-time processor) or XDETECT is used to record seismic waveforms in SUDS 1.3x format. The file naming convention used for seismic network data is: YYMMDDNN.WVM, where, YY is the year, MM is the month, DD is the day, NN is the event number for this day, WV identifies the file as a SUDS waveform file in multiplexed form and M is the agency code (M = Menlo Park, A = Alaska, etc.). For data recorded on portable autonomous digital seismographs (PADS) such as the RefTek IRIS/PASSCAL instrument, we generally need more accurate time information as well as a station name in the filename. We have established the following naming convention for this type of data: TTTTTTTT.SSN, where, TTTTTTTT is the initial sample time (number of seconds since 1-1-70 00:00:00) of the earliest waveform in the file represented as a 32 bit integer in hexadecimal notation, SS is a two alpha-numeric character station identifier and N is the data stream number. A utility program named STTIME is provided to convert the hex time filename to/from year, month, day, hour, minute, second and day of year.

   Other naming conventions may be used but these programs expect that when a list of file specification are sorted in ASCII order, they are in chronological order as well.

2) SUDSMAN would manage the data files and oversee first order processing once they are created. Typically, the program would move the file off of the machine that actually created it (on-line machine) to a working directory on

a different machine (off-line machine) for processing. At this point, SUDSMAN would also copy the original file to tape or optical disk for archive purposes. SUDSMAN would then call a user supplied batch file called PROC.BAT to perform automated processing. Below is a outline of a system that would give a preliminary hypocenter solution and generate hardcopy records of the data:

a) Typically network data has IRIG time code recorded on one channel, the file would be processed with FIXTIME to correct the initial sample time and sampling rate. The file would then be demultiplexed with DEMUX. Once the file is demultiplexed, the filename extension is usually changed to .DMX.

b) Phase arrivals and coda duration's would then be picked using SUDSPICK. This information is written to disk in a HYPO71PC compatible phase file with the same name as the data file and a .PHA extension.

c) HYPO71PC would then be used to locate the hypocenter. This program generates a "printer output file" named HYPO71PC.PRT. This file would be renamed to the same name as the data file with a .PRT extension. This file contains the hypocenter solution, various statistics and station information.

d) Finally, SUDSPLOT would be used to create plots of the waveforms and optionally mark phase arrivals and coda duration's on them. These plots may be printed on most popular laser printers at 300 dot per inch (dpi) and may be displayed on most display adapter / monitor combinations.

3) SUDSMAN would check for more data files on the on-line machine. If another file exists, the process would repeat, otherwise the program will sleep for a user specified time and then check again. SUDSMAN keeps an extensive log of all activities will working.

4) SUDSPROC can then be ran and you can peruse the dataset. You might want to manually re-pick phase, refine the hypocenter solution and re-plot the data with the updated phase arrivals and solution

Several example batch files will be given to illustrate this type of processing.

1.1 System Requirements

SUDSPROC and SUDSMAN require an IBM compatible personal computer running PC-DOS or MS-DOS version 3.2 or later. A 80x87 co-processor is not required. Although these programs do not directly use extended memory, they does greatly benefit from disk caches and the use of virtual disks (RAM disks) and so several megabytes (Mb) of extended (or less beneficial, expanded) memory is recommended.

SUDSPROC requires a Microsoft compatible mouse and driver.

SUDSMAN assumes that your network drives appear as regular DOS drives

1.2 Installation

All files on the distribution disk should be placed in a single directory on your hard disk. This directory should be added the your PATH. These programs look in their "home" directory (*i.e.*, the directory where SUDSPROC.EXE or SUDSMAN.EXE are located) for various support files such as the SUDSUTIL.INI, all of these files should be kept in the same directory. SUDSUTIL.INI contains initialization information that is needed by both of these programs at start-up.

Below is a general installation procedure:

Insert the distribution disk into drive A: and close the door.

Issue the following commands at the DOS prompt:

```
C:
MD \SUDS
CD \SUDS
XCOPY A:\*.*
```

When copying is complete, you should edit your AUTOEXEC.BAT file to include C:\SUDS in your PATH statement. Your PATH statement should look something like this:

```
PATH C:\;C:\DOS;C:\UTILS;C:\SUDS
```

If you placed the distribution diskette into a drive other than A:, substitute that drive letter for A: in the above commands. If you want to install SUDSPLOT on a drive other than C:, substitute that drive letter for C: in the above commands. Finally, if you want to install SUDSPLOT in a directory other than \SUDS, substitute that directory name for \SUDS in the above commands.

Before using these programs, you should reset your computer (press CTRL-ALT-DEL) so that changes to your configuration will take effect.

SUDSPROC

2.0 Overview

SUDSPROC looks at data files in a specified directory and displays the files on the screen. You may perform various processing to these files by simply clicking the mouse pointer. In order to use SUDSPROC effectively, you need to edit DOS batch files to suit your needs. It you are not familiar with DOS batch files, please refer to your DOS user reference.

During processing, several intermediate data files are created. For example, lets say that RTP generated a SUDS data file named 92040100.WVM (see 1.0 for a brief discussion of naming conventions). First we would demultiplex the file which creates a new data file named 92040100.DMX. We then pick phases and generate and phase list for input to HYPO71PC and name it 92040100.PHA. We then locate the event and create a "printer output file" with HYPO71PC and name it 92040100.PRT. SUDSPROC displays these files in columns on the screen. On the left is the original data file (92040100.WVM). In the next column to the right, the demultiplexed data file (92040100.DMX). In the next, the phase list (92040100.PHA). The next column is for the .PRT file. The last two columns are simply buttons that allow two produce two different types of plots.

If you have just the .WVM files, they appear in the left column. When you click in the next column, SUDSPROC will invoke a batch file (DMX.BAT for this column) to perform the processing necessary to create the demultiplexed data file. This works the same way for each of the other columns.

The menu bar across the top of the screen has two menus on it, the FILE menu and the EDIT menu. The FILE menu has options for changing the source directory, the working directory, the filename extension for raw data files and the text editor to use for editing the batch files which brings us to the EDIT menu. The EDIT menu lists the five batch files and allows you to modify them on the fly to suit your processing needs and also allows you to edit SUDSUTIL.INI.

It is recommended that you study these batch files closely. The default batch files provided on the disk are profusely commented and you should be careful when modifying them for your particular needs. Each batch file has a comment - rem user specified processing follows - all of the commands above that point should not be modified.

2.1 SUDSPROC & SUDSUTIL.INI

SUDSPROC looks in its "home" directory for SUDSUTIL.INI. This file contains initialization information for the various SUDS utilities. Each program has a "section" in SUDSUTIL.INI. SUDSPROC looks for the [SUDSPROC] section and read the information following the section header. You should read the top portion of this file for information about structure and comments and should comment changes thoroughly for future reference. Below is a sample SUDSPROC section from SUDSUTIL.INI:

```
[SUDSPROC]
# This section contains entries for SUDSPROC 1.01 or later.

# Path to data files
SourceDir = D:\Data

# Path to working directory
WorkingDir = E:\

# Filename extension of raw data files
RawExtension = WVM

# Preferred text editor
Editor = QEDIT.EXE
```

These settings specify the defaults for SUDSPROC. You can override these settings on the command-line when you invoke SUDSPROC or you may change them with the options on the FILE menu.

SourceDir specifies the directory that contains the data files that you wish to process. WorkingDir specifies a temporary working directory. This is so that you may have SUDSPROC copy the files to a large RAM disk to speed processing. RawExtension specifies the filename extension used for raw data files. Editor specifies the name of the editor that you wish to use when editing batch files or SUDSUTIL.INI from inside SUDSPROC. QEDIT.EXE is simple text editor that is provided on the distribution diskette.

2.2 Command-line syntax

SUDSPROC is controlled for the DOS command-line using the following syntax:

```
SUDSPROC - SUDS - R. Banfill

Usage: SUDSPROC [switches] [source_dir] [switches]

Switches:
 /Eext  - ext = Raw SUDS data filename extension. (WVM)

 /Wdir  - dir = Working directory. (current dir)
   source_dir = Directory containing files to be processed. (cur dir)

[] = optional, () = default value.

Returns exit code 1 if an error occurred, otherwise 0.
Arguments are not case-sensitive and may appear in any order.
```

These options are equivalent to the settings in SUDSUTIL.INI. Any settings passed on the command-line will override settings in SUDSUTIL.INI.

2.3 Process batch files

When SUDSPROC spawns a batch file to perform processing, it passes each component part of the data file specification separately. This is so that the batch file call have access to each piece individually or may put the pieces back together to have a fully qualified file specification. If you are not familiar with using replaceable parameters in batch files, you should refer to the batch programming section of your DOS user reference manual.

The following batch file is the default PHA.BAT file provided on the diskette. Each of the other batch files are very similar to this one.

```
@Echo off
rem - PHA.BAT

rem - This batch file should accept a demux'ed (.DMX) SUDS file
rem   and produce a phase file (.PHA).

rem - SUDSPROC passes each component part of the filespec for the
rem   data file separatly so that you have access to each part.

rem   %1 = Path to source directory with trailing backslash
rem   %2 = Name of data file without extension
rem   %3 = Filename extension of the raw data file
rem   %4 = Path to working directory with trailing backslash

rem   Additional parameters will be passed if another batch file
rem   neededthis batch file to execute before it could do its work,
rem   if %5 contains anything, this batch file will not re-invoke
rem   SUDSPROC, thus controlwill be returned to the calling batch
rem   file.
```

```
rem - Make sure that we have a .DMX file
if not exist %1%2.DMX call DMX.BAT %1 %2 %3 %4 *
rem - If DMX.BAT did not succesfully execute, bailout
if not exist %1%2.DMX goto error

rem - If we dont have a copy of the DMX file in working dir, get one
echo.
if not exist %4%2.DMX echo Copying %1%2.DMX to %4%2.DMX
if not exist %4%2.DMX copy %1%2.DMX %4%2.DMX > nul

rem *** Start of user specified processing ***

rem - Pick it
echo.
SUDSPICK %4%2 /B
if errorlevel 1 goto error

rem *** End of user specified processing ***

rem - Copy the phase file back to the source dir
echo.
if not %1 == %4 echo Copying %4%2.PHA to %1%2.PHA
if not %1 == %4 copy %4%2.PHA %1%2.PHA > nul

rem - Delete the phase file from the working directory
if not %1 == %4 echo Deleting %4%2.PHA
if not %1 == %4 del %4%2.PHA

goto end

:error
echo An ERROR occured while processing %4%2.DMX
pause

rem - Re-invoke SUDSPROC if it was the calling process
:end
if "%5" == "" SUDSPROC %1 /E%3 /W%4
```

The batch files provided on the diskette perform chaining to do catch-up processing in a situation, for example, where you click in the .PRT column to locate an event but the file has not been picked yet. In this situation, PRT.BAT would see that the .PHA does not exist and so it would call PHA.BAT to perform this processing before it continues. When one batch file calls another it passes a fifth argument on the command-line, an asterisk (*). This tells the called batch file that it was called from another batch file and so it should return control to the calling batch file rather than re-invoking SUDSPROC.

SUDSMAN

3.0 Overview

SUDSMAN watches for data files to appear in a specified directory (source directory). When a file appears, it moves the file to a working directory and spawns a user supplied batch file named PROC.BAT to process it. This program sleeps when no files are available in the source directory and checks at a specified interval for new files to appear. If several files exist in the source directory, they are processed from oldest to newest.

3.1 SUDSMAN & SUDSUTIL.INI

Like SUDSPROC, SUDSMAN looks in its home directory for SUDSUTIL.INI. If found, the program read the entries following the [SUDSMAN] section header. Below is a sample [SUDSMAN] section from the default .INI file on the diskette:

```
[SUDSMAN]
# This section contains entries for SUDSMAN 1.01 or later.

# Run in verbose mode
Verbose

# Source directory mask
Mask=*.WVM

# Source and destination directories
Source=D:\CODE\SHELL\TEST
Destination=D:\CODE\SHELL\DEST

# Time in seconds after which a file is to be considered inactive
Inactive=120

# Time in seconds to wait between checks for new files
Wait=15
```

The Verbose entry tells SUDSMAN to display status information on the screen while it is running. The Mask entry specifies the wild card mask used on the source directory. This particular entry specifies that all files with a .WVM extension should be processed. The Source entry specifies the directory that contains the file to be processed. This is usually on the on-line machine. The Destination entry specifies the directory (usually on the off-line machine) where the files should be moved to for processing.

The Inactive entry specifies the time that the program should wait after the appearance of a file before trying to move it. The purpose of this is to prevent sharing violations and so forth. This should be set longer that the longest possible data file. The wait entry specifies how often SUDSMAN should check for new data files once it is caught up and sleeping.

3.2 Command-line syntax

SUDSMAN is controlled from the DOS command-line using the following syntax:

```
SUDSMAN - SUDS file manager, Version 1.01, R.Banfill

Usage: SUDSMAN [switches] source_dir dest_dir

Switches:
   /M=mask File name mask for source directory (*.WVM).
   /I=n    Seconds to wait before a file is inactive.
   /W=n    Seconds between checks for new files.
   /V      Verbose mode.
   /S      Safety, Copy and process first file only.
              Source file will not be deleted.

Arguments are not case sensitive.
All of these settings can be made in the .INI file.
Command-line setting override settings in the .INI file.
```

These settings, with the exception of the /S switch are equivalent to the settings in SUDSUTIL.INI and if specified on the command-line will override the settings in the .INI file.

The /S switch is provided for debugging purposes. After changing PROC.BAT, you can use this switch while testing so that you do not lose any data.

3.3 PROC.BAT

The following batch file is just an example of what PROC.BAT might look like. This particular file will archive the original file and then demultiplex, pick, locate and plot the event before returning to SUDSMAN.

```
@ECHO OFF
rem This file is shelled to by SUDSMAN every time it copies a file
rem from it's source directory.  The filename without the
rem extension is passed as %1.
rem

rem Make sure we have somthing to do.
IF "%1" == "" GOTO END

rem *** All user processing follows ***.

rem - Archive the original file to F:
COPY %1.WVM F:\DATA

rem - Corrent the time
FIXTIME 0 %1.WVM

rem - Copy the file to the RAM disk
```

```
COPY %1.wvm D:\ > NUL

rem - Log into the RAM disk
D:

rem - Demultiplex
DEMUX %1.WVM %1.DMX

rem - Pick phases and locate the event
SUDSPICK %1 /b
IF ERRORLEVEL 1 GOTO ERROR
CALL GOHYPO %1

rem - Make a plot with everything on it
SUDSPLOT /m10 %1
IF ERRORLEVEL 1 GOTO ERROR

rem - Archive the phase list and .PRT file
COPY %1.PHA F:\DATA > NUL
COPY %1.PRT F:\DATA > NUL

rem - Clean up the RAM disk for next file
ECHO Y | DEL D:\*.* > NUL

rem - Log back into C:
C:

GOTO END

:ERROR
ECHO.
ECHO An Error has occured while processing %1
PAUSE

:END
```

Sun 16-Aug-1992 15:48, RB

>>> Notes on SUDSPROC 1.00 <<<

SUDSPROC was written using Microsoft C 6.00AX.
The makefile provided is for use with the PWB.

Libraries required:
   UWINMS.LIB   - UltraWin library.
                  Available from:
                     Small Systems Support
                     2 Boston Harbor Place
                     Big Water, UT 84741-0205
                     (801) 675-5827 Voice
                     (801) 675-3730 FAX

The following files are included:

SUDSPROC	C	24638	12-30-91	5:18p
SUDSPROC	MAK	2593	04-28-92	2:06p
DMX	BAT	1809	12-31-91	5:16p
GOHYPO	BAT	671	12-30-91	4:46p
PHA	BAT	1827	01-17-92	12:15a
PLOT1	BAT	1547	04-28-92	2:16p
PLOT2	BAT	1689	04-28-92	2:15p
PRT	BAT	1345	12-31-91	5:17p

```
@Echo off
rem - DMX.BAT

rem - This batch file should take a raw SUDS file, correct the timing,
rem   and demultiplex the data creating a .DMX file.

rem - SUDSPROC passes each component part of the filespec for the data file
rem   separatly so that you have access to each part.

rem   %1 = Path to source directory with trailing backslash, e.g., C:\DATA\
rem   %2 = Name of data file without extension, e.g., 91122800
rem   %3 = Filename extension of the raw data file, e.g., WVM
rem   %4 = Path to working directory with trailing backslash, e.g., D:\

rem   Additional parameters will be passed if another batch file needed
rem   this batch file to execute before it could do its work, if %5 contains
rem   anything, this batch file will not re-invoke SUDSPROC, thus control
rem   will be returned to the calling batch file.

rem - Make sure we have the data file
if not exist %1%2.%3 goto end

rem *** Start of user specified processing ***

rem - Fix the time
echo.
echo Fixing time %1%2.%3
FIXTIME 0 %1%2.%3

rem - Copy the time corrected raw data file to the working directory
echo.
if not %1 == %4 echo Copying %1%2.%3 to %4%2.%3
if not %1 == %4 copy %1%2.%3 %4%2.%3 > nul

rem - Demux the file
echo.
echo Demuxing %4%2.%3
DEMUX %4%2.%3 %4%2.DMX
rem DEMUX16 %1%2.%3 %4%2.DMX

rem - Copy the DMX file back to the source directory, but leave a copy
rem   in the working directory
echo.
if not %1 == %4 echo Copying %4%2.DMX to %1%2.DMX
if not %1 == %4 copy %4%2.DMX %1%2.DMX > nul

rem - Delete the raw data file from the working directory
if not %1 == %4 echo Deleting %4%2.%3
if not %1 == %4 del %4%2.%3

rem *** End of user specified processing ***

rem - Re-invoke SUDSPROC if it was the calling process
:end
if "%5" == "" SUDSPROC %1 /E%3 /W%4
```

```
@echo off
REM - GOHYPO.BAT 28-Dec-1991 01:59, RB

REM - This is the directory were hypo71pc.hdr is located
REM - This must have the trailing backslash
set hdr_dir=C:\SUDS\

if not exist %1.pha goto error1
if not exist %hdr_dir%hypo71pc.hdr goto error2

copy %hdr_dir%hypo71pc.hdr+%1.pha hypo71x.inp > nul

hypo71x

if not exist hypo71x.prt goto error3
copy hypo71x.prt %1.prt > nul
del hypo71x.inp
del hypo71x.prt

goto end

:error1
echo ERROR: %1.PHA does not exist!
pause
goto end

:error2
echo ERROR: %hdr_dir%HYPO71PC.HDR does not exist!
pause
goto end

:error3
echo ERROR: Hypo71 did not process %1.PHA!
pause

:end
set hdr_dir=
```

```
@Echo off
rem - PHA.BAT

rem - This batch file should accept a demux'ed (.DMX) SUDS file and produce
rem   a phase file (.PHA).

rem - SUDSPROC passes each component part of the filespec for the data file
rem   separatly so that you have access to each part.

rem   %1 = Path to source directory with trailing backslash, e.g., C:\DATA\
rem   %2 = Name of data file without extension, e.g., 91122800
rem   %3 = Filename extension of the raw data file, e.g., WVM
rem   %4 = Path to working directory with trailing backslash, e.g., D:\

rem   Additional parameters will be passed if another batch file needed
rem   this batch file to execute before it could do its work, if %5 contains
rem   anything, this batch file will not re-invoke SUDSPROC, thus control
rem   will be returned to the calling batch file.

rem - Make sure that we have a .DMX file
if not exist %1%2.DMX call DMX.BAT %1 %2 %3 %4 *
rem - If DMX.BAT did not succesfully execute, bailout
if not exist %1%2.DMX goto error

rem *** Start of user specified processing ***

rem - If we dont have a copy of the DMX file in the working dir, get one
echo.
if not exist %4%2.DMX echo Copying %1%2.DMX to %4%2.DMX
if not exist %4%2.DMX copy %1%2.DMX %4%2.DMX > nul

rem - Pick it
echo.
SUDSPICK %4%2
if errorlevel 1 goto error

rem - Copy the phase file back to the source dir
echo.
if not %1 == %4 echo Copying %4%2.PHA to %1%2.PHA
if not %1 == %4 copy %4%2.PHA %1%2.PHA > nul

rem - Delete the phase file from the working directory
if not %1 == %4 echo Deleting %4%2.PHA
if not %1 == %4 del %4%2.PHA

rem *** End of user specified processing ***

goto end

:error
echo An ERROR occured while processing %4%2.DMX
pause

rem - Re-invoke SUDSPROC if it was the calling process
:end
if "%5" == "" SUDSPROC %1 /E%3 /W%4
```

```
@Echo off
rem - PLOT1.BAT

rem - This batch file should produce the first type of plot, e.g. just
rem   the raw data.

rem - SUDSPROC passes each component part of the filespec for the data file
rem   separatly so that you have access to each part.

rem   %1 = Path to source directory with trailing backslash, e.g., C:\DATA\
rem   %2 = Name of data file without extension, e.g., 91122800
rem   %3 = Filename extension of the raw data file, e.g., WVM
rem   %4 = Path to working directory with trailing backslash, e.g., D:\

rem   Additional parameters will be passed if another batch file needed
rem   this batch file to execute before it could do its work, if %5 contains
rem   anything, this batch file will not re-invoke SUDSPROC, thus control
rem   will be returned to the calling batch file.

rem - Make sure that we have a .DMX file
if not exist %1%2.DMX call DMX.BAT %1 %2 %3 %4 *
rem - If DMX.BAT did not succesfully execute, bailout
if not exist %1%2.DMX goto error

rem *** Start of user specified processing ***

rem - If we dont have a copy of the DMX file in the working dir, get one
echo.
echo Checking for %2.DMX in %4, copying if required
if not exist %4%2.DMX copy %1%2.DMX %4%2.DMX > nul

rem SUDSPLOT /X /M10 /T* %4%2
SUDSPLOT /M10 %4%2 /P1
if errorlevel 1 goto error

rem *** End of user specified processing ***

goto end

:error
echo An ERROR occured while processing %4%2.DMX
pause

rem - Re-invoke SUDSPROC if it was the calling process
:end
if "%5" == "" SUDSPROC %1 /E%3 /W%4
```

```
@Echo off
rem - PLOT2.BAT

rem - This batch file should produce a second type of plot, e.g. by
rem   epicentral distance with phase arrivals marked, and so on.

rem - SUDSPROC passes each component part of the filespec for the data file
rem   separatly so that you have access to each part.

rem   %1 = Path to source directory with trailing backslash, e.g., C:\DATA\
rem   %2 = Name of data file without extension, e.g., 91122800
rem   %3 = Filename extension of the raw data file, e.g., WVM
rem   %4 = Path to working directory with trailing backslash, e.g., D:\

rem   Additional parameters will be passed if another batch file needed
rem   this batch file to execute before it could do its work, if %5 contains
rem   anything, this batch file will not re-invoke SUDSPROC, thus control
rem   will be returned to the calling batch file.

rem - Make sure that we have a .PRT file
if not exist %1%2.PRT call PRT.BAT %1 %2 %3 %4 *
rem - If PRT.BAT did not succesfully execute, bailout
if not exist %1%2.PRT goto error

rem *** Start of user specified processing ***

rem - If we dont have a copy of the DMX file in the working dir, get one
echo.
echo Checking for %2.DMX in %4, copying if required
if not exist %4%2.DMX copy %1%2.DMX %4%2.DMX > nul

rem - Plot it
echo.
rem SUDSPLOT /X /T* /M10 /A /W3 /S%1%2.prt %1
SUDSPLOT /M10 /A2 %4%2
if errorlevel 1 error

rem Cleanup the working directory
if not %1 == %4 del %4%2.DMX

rem *** End of user specified processing ***

goto end

:error
echo An ERROR occured while processing %4%2.DMX
pause

rem - Re-invoke SUDSPROC if it was the calling process
:end
if "%5" == "" SUDSPROC %1 /E%3 /W%4
```

```
@Echo off
rem - PRT.BAT

rem - This batch file should take a phase file (.PHA) and locate the
rem   earthquake, producing a .PRT file.

rem - SUDSPROC passes each component part of the filespec for the data file
rem   separatly so that you have access to each part.

rem   %1 = Path to source directory with trailing backslash, e.g., C:\DATA\
rem   %2 = Name of data file without extension, e.g., 91122800
rem   %3 = Filename extension of the raw data file, e.g., WVM
rem   %4 = Path to working directory with trailing backslash, e.g., D:\

rem   Additional parameters will be passed if another batch file needed
rem   this batch file to execute before it could do its work, if %5 contains
rem   anything, this batch file will not re-invoke SUDSPROC, thus control
rem   will be returned to the calling batch file.

rem - Make sure that we have a .PHA file
if not exist %1%2.PHA call PHA.BAT %1 %2 %3 %4 *
rem - If PHA.BAT did not succesfully execute, bailout
if not exist %1%2.PHA goto error

rem *** Start of user specified processing ***

call GOHYPO %1%2
if not exist %1%2.PRT goto error

rem *** End of user specified processing ***

goto end

:error
echo An ERROR occured while processing %1%2.PHA
pause

rem - Re-invoke SUDSPROC if it was the calling process
:end
if "%5" == "" SUDSPROC %1 /E%3 /W%4
```

```
/*
 * SUDSPROC.C - SUDS data file processing manager
 *
 * D:\CODE\SUDSPROC\sudsproc.c
 *
 * 20-Dec-1991 15:57, RB
 *
 */

char version[] = "SUDSPROC Version 1.01";

#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <string.h>
#include <malloc.h>
#include <dos.h>
#include <io.h>
#include <errno.h>
#include <direct.h>
#include <stdarg.h>
#include <process.h>

#include <uw.h>
#include <uw_globx.h>
#include <uw_keys.h>

void build_menus( void );
void build_window( void );
int build_dir( void );
int display_dir( void );
void write_line( int, int );
int handle_event( void );
void handle_menu( int );
void dialog_box( char *, char *, int );
void first_screen( void );
void error( char *, ... );
void s_upper( char * );
int comp( char *, char * );
void invoke( char *, int );
void edit( char * );
void ini( void );

typedef struct {
   char name[9];
   char raw[4];
   char dmx[4];
   char pha[4];
   char prt[4];
} INDEX;

struct {
   char raw[4];
   char dmx[4];
   char pha[4];
   char prt[4];
} ext;

char s_dir[_MAX_DIR];
char w_dir[_MAX_DIR];
char editor[_MAX_PATH];
char prog_path[_MAX_PATH];
char ini_file[_MAX_PATH];

INDEX *files;
int entries = 0;

WINDOW main_wn, *wnp = &main_wn;
WINDOW dialog_wn, *dnp = &dialog_wn;
int t_row, b_row, t_rec;
int wins = 0;
```

```
MENU top_menu, *top_mnp = &top_menu;
MENU file_menu, edit_menu;
MENU *drop_mnps[2];

// Pull-down menu item ID's
#define SOURCE_DIR 10
#define RAW_EXT    11
#define EDITOR     12
#define ABOUT      13
#define EXIT       14
#define WORK_DIR   15
#define DMX        20
#define PHA        21
#define PRT        22
#define PLOT1      23
#define PLOT2      24
#define INI        25

//*************************************************************************
main( int argc, char *argv[] ) {
   register i;
   char *j, buf[_MAX_PATH];

   strcpy( prog_path, argv[0] );
   j = strrchr( prog_path, '\\' )+1;
   *j = '\0';

   // Defaults
   strcpy( ext.raw, "WVM" );
   strcpy( ext.dmx, "DMX" );
   strcpy( ext.pha, "PHA" );
   strcpy( ext.prt, "PRT" );
   s_dir[0] = '\0';
   w_dir[0] = '\0';
   strcpy( editor, "QEDIT.EXE" );

   strcpy( ini_file, prog_path );
   strcat( ini_file, "SUDSUTIL.INI" );
   ini( );

   if( argv[1][0] == '?' || argv[1][1] == '?' ) {
      printf( "SUDSPROC - %.4s - R. Banfill\n\n", version );
      printf( "Usage: SUDSPROC [switches] [source_dir] [switches]\n\n" );
      printf( "Switches:\n" );
      printf( "   /Eext  - ext = Raw SUDS data filename extension. (%s)\n\n", ext.raw );
      printf( "   /Wdir  - dir = Working directory. (current dir)\n" );
      printf( "     source_dir = Directory containing files to be processed. (current dir)\n\
      printf( "[] = optional, () = default value.\n\n" );
      printf( "Returns exit code 1 if an error occurred, otherwise 0.\n" );
      printf( "Arguments are not case-sensitive and may appear in any order.\n" );
      exit( 1 );
   }
   for( i=1; i<argc; i++ ) {
      if( argv[i][0] == '/' || argv[i][0] == '-' ) {
         switch( argv[i][1] ) {
            // Raw extension
            case 'e':
            case 'E':
               strncpy( ext.raw, &argv[i][2], 3 );
               s_upper( ext.raw );
               break;
            // Working directory
            case 'w':
            case 'W':
               strncpy( w_dir, &argv[i][2], 3 );
               break;
         }
      }
      else
         // Source directory
         strcpy( s_dir, argv[i] );
   }
```

```
   if( s_dir[0] == '\0' )
      getcwd( s_dir, _MAX_PATH );
   if( w_dir[0] == '\0' )
      getcwd( w_dir, _MAX_PATH );

   _fullpath( buf, s_dir, _MAX_PATH );
   strcpy( s_dir, buf);
   s_upper( s_dir );
   _fullpath( buf, w_dir, _MAX_PATH );
   strcpy( w_dir, buf);
   s_upper( w_dir );

   j = s_dir+strlen( s_dir )-1;
   if( *j != '\\' )
      strcat( s_dir, "\\" );
   j = w_dir+strlen( w_dir )-1;
   if( *j != '\\' )
      strcat( w_dir, "\\" );

   if( build_dir( ) != 0 )
      exit( 1 );

   if( display_dir( ) != 0 )
      exit( 1 );

   exit( 0 );
}

//*************************************************************************
int display_dir( void ) {
   register i, j;
   int key;

   build_window( );
   build_menus( );

   i = 1;
   j = 0;
   key = 0;
   t_row = 0;
   b_row = 21;

   first_screen( );

   // Navigation loop
   while( key != KEY_ESC ) {
      key = handle_event( );

      switch( key ) {
         case( KEY_DN ):   // Scroll down
            if( t_rec < entries-1 ) {
               t_rec++;
               j = t_row;
               for( i=t_rec; i<entries; i++ ) {
                  write_line( j, i );
                  if( j < b_row  )
                     j++;
                  else
                     break;
               }
               if( j < b_row ) {
                  mv_cs( 1, j, wnp );
                  wn_cleos( wnp );
               }
            }
            break;
         case( KEY_UP ):   // Scroll up
            if( t_rec >= 1 ) {
               t_rec--;
               j = t_row;
               for( i=t_rec; i<entries; i++ ) {
```

```
            write_line( j, i );
            if( j < b_row  )
               j++;
            else
               break;
         }
         if( j < b_row ) {
            mv_cs( 1, j+1, wnp );
            wn_cleos( wnp );
         }
      }
      break;
   case( KEY_PGDN ):   // Page down
      if( t_rec < entries-1 ) {
         t_rec += 20;
         if( t_rec >= entries-1 )
            t_rec = entries-1;
         j = t_row;
         for( i=t_rec; i<entries; i++ ) {
            write_line( j, i );
            if( j < b_row  )
               j++;
            else
               break;
         }
         if( j < b_row ) {
            mv_cs( 1, j, wnp );
            wn_cleos( wnp );
         }
      }
      break;
   case( KEY_PGUP ):   // Page up
      if( t_rec >= 1 ) {
         t_rec -= 20;
         if( t_rec < 0 )
            t_rec = 0;
         j = t_row;
         for( i=t_rec; i<entries; i++ ) {
            write_line( j, i );
            if( j < b_row  )
               j++;
            else
               break;
         }
         if( j < b_row ) {
            mv_cs( 1, j+1, wnp );
            wn_cleos( wnp );
         }
      }
      break;
   case( KEY_HOME ):   // Home
      if( t_rec != 0 ) {
         t_rec = 0;
         j = t_row;
         for( i=t_rec; i<entries; i++ ) {
            write_line( j, i );
            if( j < b_row  )
               j++;
            else
               break;
         }
         if( j < b_row ) {
            mv_cs( 1, j+1, wnp );
            wn_cleos( wnp );
         }
      }
      break;
   case( KEY_END ):   // End
      if( t_rec != entries-22 ) {
         t_rec = entries-22;
         if( t_rec < 0 )
            t_rec = 0;
```

```
                j = t_row;
                for( i=t_rec; i<entries; i++ ) {
                   write_line( j, i );
                   if( j < b_row  )
                      j++;
                   else
                      break;
                }
                if( j < b_row ) {
                   mv_cs( 1, j+1, wnp );
                   wn_cleos( wnp );
                }
             }
             break;
       }
    }

    remove_window( wnp );
    wn_destroy( wnp );
    end_mouse( );
    end_video( );

    return( 0 );
}

//*************************************************************************
int handle_event( void ) {
    register i;
    int elevator, file, id;
    char str[80];

    wn_vline( 14, SGL_BDR, wnp );
    wn_vline( 29, SGL_BDR, wnp );
    wn_vline( 44, SGL_BDR, wnp );
    wn_vline( 59, SGL_BDR, wnp );
    wn_vline( 68, SGL_BDR, wnp );
    sprintf( str, " %s %d Events ", s_dir, entries );
    wn_name( str, wnp );
    wn_border( wnp );

    mv_cs( 78, 0, wnp );
    wn_ch( '\x18', wnp );
    mv_cs( 78, 1, wnp );
    wn_co( 20, '\xB1', wnp );
    mv_cs( 78, 21, wnp );
    wn_ch( '\x19', wnp );

    elevator = (int)((20.0/(float)entries)*(float)t_rec)+1;
    if( elevator > 20 )
       elevator = 20;
    mv_cs( 78, elevator, wnp );
    wn_ch( '\xDB', wnp );

    // Event loop
    while( 1 ) {
       Event.key = 0;
       menu_set( top_mnp );

       m_show( );
       id = menu_system( top_mnp, drop_mnps, 0 );
       m_hide( );

       if( id != 0 ) {
          handle_menu( id );
          Event.key = 0;
          break;
       }
       else {
          if( Event.is_mouse ) {
             if( Event.m_x == 79 ) {
                if( Event.m_y == 23 )
                   return( KEY_DN );
```

```
            else if( Event.m_y == 2 )
               return( KEY_UP );
            else if( Event.m_y > elevator+2 && Event.m_y < 23 )
               return( KEY_PGDN );
            else if( Event.m_y < elevator+2 && Event.m_y > 2 )
               return( KEY_PGUP );
         }
         else if( Event.m_x > 15 && Event.m_x < 30 ) {
            file = t_rec+(Event.m_y-2);
            if( file < entries )
               invoke( "DMX.BAT", file );
         }
         else if( Event.m_x > 30 && Event.m_x < 45 ) {
            file = t_rec+(Event.m_y-2);
            if( file < entries )
               invoke( "PHA.BAT", file );
         }
         else if( Event.m_x > 45 && Event.m_x < 60 ) {
            file = t_rec+(Event.m_y-2);
            if( file < entries )
               invoke( "PRT.BAT", file );
         }
         else if( Event.m_x > 60 && Event.m_x < 69 ) {
            file = t_rec+(Event.m_y-2);
            if( file < entries )
               invoke( "PLOT1.BAT", file );
         }
         else if( Event.m_x > 69 && Event.m_x < 78 ) {
            file = t_rec+(Event.m_y-2);
            if( file < entries )
               invoke( "PLOT2.BAT", file );
         }

      }
      else
         break;
   }
}

   return( Event.key );
}

//*************************************************************************
void handle_menu( int i ) {
   char *j, buf[_MAX_PATH];

   switch( i ) {
      case EXIT:
         wn_destroy( wnp );
         end_mouse( );
         end_video( );
         exit( 0 );
      case ABOUT:
         wn_create( 20, 5, 60, 16, DBL_BDR, WN_NORMAL, dnp );
         wn_color( BLACK, LIGHTGRAY, dnp );
         wn_bdr_color( BLACK, LIGHTGRAY, dnp );
         add_window( dnp );
         wn_plst( CENTERED, 1, version, dnp );
         wn_plst( CENTERED, 3, "SUDS Data Processing Manager", dnp );
         wn_plst( CENTERED, 5, "Robert Banfill", dnp );
         wn_plst( CENTERED, 6, "Small Systems Support", dnp );
         wn_plst( CENTERED, 7, "Big Water, UT 84741", dnp );
         wn_plst( CENTERED, 8, "(801) 675-5827", dnp );
         wait_event( );
         Event.key = 0;
         remove_window( dnp );
         wn_destroy( dnp );
         break;
      case SOURCE_DIR:
         dialog_box( " Enter path to new source directory ", s_dir, 64 );
         _fullpath( buf, s_dir, _MAX_PATH );
         strcpy( s_dir, buf);
```

```
      s_upper( s_dir );
      if( s_dir[0] == '\0' ) {
         getcwd( s_dir, _MAX_PATH );
         strcat( s_dir, "\\" );
      }
      else {
         j = s_dir+strlen( s_dir )-1;
         if( *j != '\\' )
            strcat( s_dir, "\\" );
      }
      build_dir( );
      first_screen( );
      break;
   case WORK_DIR:
      dialog_box( " Enter path to new working directory ", w_dir, 64 );
      _fullpath( buf, w_dir, _MAX_PATH );
      strcpy( w_dir, buf);
      if( w_dir[0] == '\0' ) {
         getcwd( w_dir, _MAX_PATH );
         strcat( w_dir, "\\" );
      }
      else {
         j = w_dir+strlen( w_dir )-1;
         if( *j != '\\' )
            strcat( w_dir, "\\" );
      }
      s_upper( w_dir );
      build_dir( );
      first_screen( );
      break;
   case EDITOR:
      dialog_box( " Enter name of text editor ", editor, 64 );
      break;
   case RAW_EXT:
      dialog_box( " Enter filename extension of raw data files ", ext.raw, 3 );
      s_upper( ext.raw );
      build_dir( );
      first_screen( );
      break;
   case DMX:
      edit( "DMX.BAT" );
      break;
   case PHA:
      edit( "PHA.BAT" );
      break;
   case PRT:
      edit( "PRT.BAT" );
      break;
   case PLOT1:
      edit( "PLOT1.BAT" );
      break;
   case PLOT2:
      edit( "PLOT2.BAT" );
      break;
   case INI:
      edit( ini_file );
      wn_create( 10, 5, 70, 11, DBL_BDR, WN_NORMAL, dnp );
      wn_color( BLACK, LIGHTGRAY, dnp );
      wn_bdr_color( BLACK, LIGHTGRAY, dnp );
      add_window( dnp );
      wn_plst( CENTERED, 1, "Do you wish for changes to take effect?",dnp );
      wn_plst( CENTERED, 3, "[Yes]  [No]",dnp );
      m_show( );
      wait_event( );
      m_hide( );
      Event.key = 0;
      remove_window( dnp );
      wn_destroy( dnp );
      if( (Event.m_y == 9) && ((Event.m_x > 34) && (Event.m_x < 40)) ) {
         ini( );
         build_dir( );
         first_screen( );
```

```
            }
            break;
      }

      return;
   }

//*************************************************************************
void dialog_box( char *title, char *str, int len ) {
   char mask[65], temp[65], new[65];

   memset( mask, '_', 64 );
   mask[64] = '\0';
   memset( temp, '*', 64 );
   temp[64] = '\0';
   memset( new, '\0', 64 );
   new[64] = '\0';

   wn_create( 6, 5, 73, 8, DBL_BDR, WN_NORMAL, dnp );
   wn_color( BLACK, LIGHTGRAY, dnp );
   wn_bdr_color( BLACK, LIGHTGRAY, dnp );
   wn_name( title, dnp );
   add_window( dnp );

   mv_cs( 1, 0, dnp );
   wn_printf( dnp, "Current: %s", str );
   mv_cs( 1, 1, dnp );
   if( wn_gets( new, mask, temp, ((CYAN<<4)|BLACK),
                STRIP_ON, dnp ) == KEY_ENTER )
      strncpy( str, new, len );

   remove_window( dnp );
   wn_destroy( dnp );

   Event.key = 0;

   return;
}

//*************************************************************************
void first_screen( void ) {
   register i;

   wn_clear( wnp );

   t_rec = 0;

   // Write out entries
   for( i=0; i<entries; i++ ) {
      if( i <= b_row )
         write_line( t_row+i, i );
      else
         break;
   }
   return;
}

//*************************************************************************
void write_line( int line, int rec ) {
   mv_cs( 1, line, wnp );
   wn_printf( wnp, "%-8s%c%3s   %-8s%c%3s   %-8s%c%3s   %-8s%c%3s   Plot 1   Plot 2",
      (files[rec].raw[0] ? files[rec].name : " "),
      (files[rec].raw[0] ? '.' : ' '), files[rec].raw,
      (files[rec].dmx[0] ? files[rec].name : "Demux"),
      (files[rec].dmx[0] ? '.' : ' '), files[rec].dmx,
      (files[rec].pha[0] ? files[rec].name : "Pick"),
      (files[rec].pha[0] ? '.' : ' '), files[rec].pha,
      (files[rec].prt[0] ? files[rec].name : "Locate"),
      (files[rec].prt[0] ? '.' : ' '), files[rec].prt );
   return;
}
```

```
//*************************************************************************
void invoke( char *batch, int i ) {

   wn_destroy( wnp );
   end_mouse( );
   end_video( );

   if( spawnlp( P_OVERLAY, batch, batch, s_dir, files[i].name, ext.raw,
                w_dir, NULL ) == -1 )
      error( "Unable to spawn process: %s", batch );

   return;
}

//*************************************************************************
void edit( char *file ) {
   char buf[_MAX_PATH];

   if( file[1] != ':' ) {
      strcpy( buf, prog_path );
      strcat( buf, file );
   }
   else
      strcpy( buf, file );

   if( spawnlp( P_WAIT, editor, editor, buf, NULL ) == -1 )
      error( "Unable to spawn task: %s", editor );

   refresh_desktop( );

   return;
}

//*************************************************************************
void build_window( void ) {

   wins = 1;
   init_video( 80, 25 );
   init_mouse( );
   if( !Mouse_exists )
      error( "This program requires a Microsoft or compatible mouse!" );
   wn_create( 0, 1, 79, 24, DBL_BDR, WN_NORMAL, wnp );
   wn_color( LIGHTGRAY, BLUE, wnp );
   wn_bdr_color( WHITE, BLUE, wnp );
   add_window( wnp );
   return;
}

//*************************************************************************
void build_menus( void ) {
   int id, i,
       back_att = (LIGHTGRAY << 4) | BLACK,
       bdr_att = (LIGHTGRAY << 4) | BLACK,
       csr_att = (CYAN << 4) | WHITE,
       first_att = (LIGHTGRAY << 4) | DARKGRAY;

   // Menu bar
   menu_create( 0, 0, 79, 0, M_HORIZONTAL, back_att, bdr_att, csr_att,
                first_att, NO_BDR, WN_NORMAL, top_mnp );
   item_add( " File ", 1, 1, &top_menu );
   item_add( " Edit ", 2, 1, &top_menu );

   drop_mnps[0] = &file_menu;
   drop_mnps[1] = &edit_menu;

   // File menu
   menu_create( 0, 1, 20, 10, M_VERTICAL, back_att, bdr_att, csr_att,
                first_att, DBL_BDR, WN_NORMAL, drop_mnps[0] );
   item_add( " Source Dir...     ", 10, 1, &file_menu );
   item_add( " Working Dir...    ", 15, 1, &file_menu );
   item_add( " Raw extension... ", 11, 1, &file_menu );
   item_add( " Text editor...    ", 12, 1, &file_menu );
```

```
   item_add( " ------------------", 98, 0, &file_menu );
   item_add( " About SUDSPROC...", 13, 1, &file_menu );
   item_add( " ------------------", 99, 0, &file_menu );
   item_add( " Exit              ", 14, 1, &file_menu );

   // Edit menu
   menu_create( 6, 1, 21, 9, M_VERTICAL, back_att, bdr_att, csr_att,
                first_att, DBL_BDR, WN_NORMAL, drop_mnps[1] );
   item_add( " DMX.BAT      ", 20, 1, &edit_menu );
   item_add( " PHA.BAT      ", 21, 2, &edit_menu );
   item_add( " PRT.BAT      ", 22, 2, &edit_menu );
   item_add( " PLOT1.BAT    ", 23, 5, &edit_menu );
   item_add( " PLOT2.BAT    ", 24, 5, &edit_menu );
   item_add( " ------------", 97, 0, &edit_menu );
   item_add( " SUDSUTIL.INI", 25, 1, &edit_menu );

   return;
}

//*************************************************************************
int build_dir( void ) {
   register j;
   int miss;
   struct find_t fi;
   char *ex, *p, name[9], buf[_MAX_PATH];

   entries = 0;

   strcpy( buf, s_dir );
   strcat( buf, "*.*" );


   // Get RAW and .DMX files
   if( _dos_findfirst( buf, _A_NORMAL, &fi ) != 0 )
      error( "No files in source directory: %s", s_dir );

   do {
      // Separate name and extension
      ex = &fi.name[0];
      p = &name[0];
      while( *ex != '.' )
         *p++ = *ex++ ;
      *p = '\0';
      ex++;

      // Get RAW and .DMX files
      if( ( strncmp( ex, ext.raw, 3 ) == 0 ) ||
          ( strncmp( ex, ext.dmx, 3 ) == 0 ) ) {

         // Look up this name
         j = 0;
         while( strcmp( name, files[j].name ) != 0 ) {
            if( j >= entries ) {
               entries++;
               if( ( files = (INDEX *)realloc( files, entries*sizeof(INDEX) ) ) == NULL )
                  error( "Not enough memory!" );
               memset( &files[j], 0, sizeof(INDEX) );
               strcpy( files[j].name, name );
               break;
            }
            j++;
         }
         if( strncmp( ex, ext.raw, 3 ) == 0 )
            strcpy( files[j].raw, ex );
         else if( strncmp( ex, ext.dmx, 3 ) == 0 )
            strcpy( files[j].dmx, ex );
      }
   } while( _dos_findnext( &fi ) == 0 );

   // Get other files
   if( _dos_findfirst( buf, _A_NORMAL, &fi ) != 0 )
      error( "No files in source directory: %s", s_dir );
```

```
   do {
      // Separate name and extension
      ex = &fi.name[0];
      p = &name[0];
      while( *ex != '.' )
         *p++ = *ex++ ;
      *p = '\0';
      ex++;

      // Get PHA and PRT files
      if( ( strncmp( ex, ext.pha, 3 ) == 0 ) ||
          ( strncmp( ex, ext.prt, 3 ) == 0 ) ) {

         // Look up this name
         j = 0;
         while( strcmp( name, files[j].name ) != 0 ) {
            if( j >= entries )
               break;
            else
               j++;
         }
         if( strncmp( ex, ext.pha, 3 ) == 0 )
            strcpy( files[j].pha, ex );
         else if( strncmp( ex, ext.prt, 3 ) == 0 )
            strcpy( files[j].prt, ex );
      }
   } while( _dos_findnext( &fi ) == 0 );

   if( entries == 0 )
      error( "No data files in source directory: %s", s_dir );

   // Sort
   qsort( (void *)files, (size_t)entries, sizeof(INDEX), comp );

   return( 0 );
}

int comp( char *str1, char *str2 ) {
   return( strcmp( str1, str2 ) );
}

//*************************************************************************
// Report error and shut down
void error( char *fmt, ... ) {
   char err_label[128] = "\nERROR: ";

   va_list marker;

   va_start( marker, fmt );

   if( wins ) {
      remove_window( wnp );
      wn_destroy( wnp );
      end_mouse( );
      end_video( );
   }

   strcat( err_label, fmt );
   strcat( err_label, "\n" );

   // Report the error
   vfprintf( stderr, err_label, marker );

   va_end( marker );

   // Bail out
   exit( 1 );
}

//*************************************************************************
void s_upper( char *buffer ) {
```

```
   char *p;
   for( p = buffer; *p; p++ )
      *p = toupper( *p );
}

//*********************************************************************
void ini( void ) {
   register i;
   FILE *inifil;
   char buf[256], *p, *j;

   if( ( inifil = fopen( ini_file, "r" ) ) == NULL ) {
      fprintf( stderr, "\nWARNING: Unable to open initialization file: %s\n", ini_file );
      return;
   }

   buf[0] = '\0';

   // Find the [SUDSPROC] section
   do {
      if( feof( inifil ) )
         break;
      fgets( buf, 255, inifil );
      s_upper( buf );
   } while( strncmp( buf, "[SUDSPROC]", 7 ) != 0 );

   // Read entries
   do {
      if( feof( inifil ) )
         break;
      fgets( buf, 255, inifil );
      s_upper( buf );

      // Strip trailing line-feeds
      if( ( p = strchr( buf, 10 ) ) != NULL )
         *p = '\0';

      if( strncmp( buf, "SOURCEDIR", 9 ) == 0 ) {
         if( ( p = strchr( buf, '=' ) ) != NULL ) {
            p++;
            while( *p == ' ' )
               p++;
            strcpy( s_dir, p );
            if( s_dir[0] == '\0' )
               getcwd( s_dir, _MAX_PATH );
            _fullpath( buf, s_dir, _MAX_PATH );
            strcpy( s_dir, buf);
            s_upper( s_dir );
            j = s_dir+strlen( s_dir )-1;
            if( *j != '\\' )
               strcat( s_dir, "\\" );
         }
         continue;
      }
      if( strncmp( buf, "WORKINGDIR", 10 ) == 0 ) {
         if( ( p = strchr( buf, '=' ) ) != NULL ) {
            p++;
            while( *p == ' ' )
               p++;
            strcpy( w_dir, p );
            if( w_dir[0] == '\0' )
               getcwd( w_dir, _MAX_PATH );
            _fullpath( buf, w_dir, _MAX_PATH );
            strcpy( w_dir, buf);
            s_upper( w_dir );
            j = w_dir+strlen( w_dir )-1;
            if( *j != '\\' )
               strcat( w_dir, "\\" );
         }
         continue;
      }
      if( strncmp( buf, "RAWEXTENSION", 12 ) == 0 ) {
```

```
            if( ( p = strchr( buf, '=' ) ) != NULL ) {
               p++;
               while( *p == ' ' )
                  p++;
               strncpy( ext.raw, p, 3 );
               s_upper( ext.raw );
            }
            continue;
         }
         if( strncmp( buf, "EDITOR", 4 ) == 0 ) {
            if( ( p = strchr( buf, '=' ) ) != NULL ) {
               p++;
               while( *p == ' ' )
                  p++;
               strcpy( editor, p );
               s_upper( editor );
            }
            continue;
         }
      } while( buf[0] != '[' );

      fclose( inifil );
      return;
   }
```

Sun 16-Aug-1992 15:48, RB

>>> Notes on SUDSMAN 1.00 <<<

SUDSMAN was written using Microsoft C 6.00AX.
The makefile provided is for use with the PWB.

Libraries required:
   Standard libraries only.

The following files are included:

SHELL	C	733	06-24-91	2:03p
SUDSMAN	C	11746	11-24-91	3:11p
SHELL	MAK	1761	06-27-91	12:11p
SUDSMAN	MAK	1955	11-24-91	2:59p
SUDSMAN	INI	355	11-24-91	3:04p
PROC	BAT	356	11-24-91	3:08p
PROC1	BAT	981	07-04-91	7:00p

```
@ECHO OFF
REM This file is Shelled to by SUDSMAN every time it copies a file
REM from it's source directory.  The filename without the extension
REM is passed as %1.
REM

REM Make sure we have somthing to do.
IF "%1" == "" GOTO END

REM All user processing follows.


echo This is PROC.BAT...
dir test

echo Returning to SUDSMAN...

:END
```

```
REM @ECHO OFF
REM This file is Shelled to by SUDSMAN every time it copies a file
REM from it's source directory.  The filename without the extension
REM is passed as %1.
REM

REM Make sure we have somthing to do.
IF "%1" == "" GOTO END

REM All user processing follows.

FIXTIME 0 %1.WVM

COPY %1.wvm D:\ > NUL

D:

DEMUX %1.WVM %1.DMX

:TRYAGAIN
PICK1 %1 /b
IF NOT EXIST %1.PHA GOTO TRYAGAIN

CALL GOHYPO %1

PICK2 %1 /b
IF ERRORLEVEL 1 GOTO ERROR

CALL GOHYPO %1

copy hypo71pc.prt prn

echo   > prn

SUDSPLOT /x /t64 /m10 %1 /A

IF ERRORLEVEL 1 GOTO ERROR

rem SUDSSQZ %1
rem IF ERRORLEVEL 1 GOTO ERROR

COPY %1.pha C:\test > NUL

ECHO Y | DEL D:\*.* > NUL

C:
GOTO END

:ERROR
ECHO.
ECHO ********************************************************************************
ECHO An Error has occured while processing %1
ECHO ********************************************************************************
ECHO.
PAUSE

:END
```

```
/*
 *
 * Shell - Spawned by SUDSMAN, in turn Shell spawns PROC.BAT,
 * when control is returned, SUDSMAN is spawned again.  SUDSMANS
 * command line args are passed through so as to preserve them.
 * 24-Jun-1991, RB
 *
 */

#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <process.h>
#include <errno.h>

main( int argc, char *argv[] ) {
   int i;
   char cmd[64];

   strcpy( cmd, getenv( "SUDSFILE" ) );

   i = spawnlp( P_WAIT, "proc.bat", "proc.bat", cmd, NULL );
   if( i < 0 ) {
      perror( "\nERROR: Could not spawn PROC.BAT" );
      exit( i );
   }

   i = spawnvp( P_OVERLAY, "sudsman.exe", argv );

   perror( "\nERROR: Could not spawn: SUDSMAN.EXE" );
   exit( i );
}
```

```
/*
 * SUDSMAN - Copies files from on-line machine to a local working
 * directory and the Spawns Shell.exe which in turn spawns PROC.BAT
 * which process's the file, returns control to Shell.exe which then
 * Spawns SUDSMAN again.  And around and around we go...
 *
 * 24-Nov-1991 14:57, RB
 *
 */

// Version number
char version[] = "1.01.00";

#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <malloc.h>
#include <process.h>
#include <dos.h>
#include <errno.h>
#include <conio.h>
#include <direct.h>
#include <search.h>
#include <time.h>

// Prototypes
void ini_file( char * );
int dir_list( char[] );
char *move_file( int );
int sleep( time_t );
int compare( char *, char * );
unsigned long disk_space( char * );
void upper( char * );

// Global data
char sudsfile[_MAX_PATH], srcdir[_MAX_PATH], desdir[_MAX_PATH],
     buffer[_MAX_PATH];
char *base, mask[13];
int verbose, safe, nfiles;
struct find_t find;
time_t cur_t, chkwait, inact, stop_t, old_t;
struct tm *ltime;
unsigned long srcdrv, desdrv;

//--------------------------------------------------------------------
main( int argc, char *argv[] ) {
   register i;
   int err, flag;
   unsigned long o_srcdrv, o_desdrv;
   char envstr[_MAX_PATH], curdir[_MAX_PATH], *curfile, *ptr;

   verbose = 0;
   safe = 0;
   flag = 0;
   strcpy( srcdir, "?" );
   strcpy( desdir, "?" );
   strcpy( mask, "*.WVM" );



   // Help
   if( argv[1][0] == '?' || argv[1][1] == '?' ) {
      printf( "SUDSMAN - SUDS file manager, Version %.4s, R.Banfill\n\n", version );
      printf( "Usage: SUDSMAN [switches] source_dir dest_dir\n\n" );
      printf( "Switches:\n\n" );
      printf( "   /M=mask File name mask for source directory (*.WVM).\n" );
      printf( "   /I=n    Seconds to wait before a file is inactive.\n" );
      printf( "   /W=n    Seconds between checks for new files.\n" );
      printf( "   /V      Verbose mode.\n" );
      printf( "   /S      Safety, Copy and process first file only.\n" );
      printf( "              Source file will not be deleted.\n\n" );
      printf( "Arguments are not case sensitive.\n" );
```

```
   printf( "All of these settings can be made in the .INI file.\n" );
   printf( "Command-line setting override settings in the .INI file.\n\n" );
   exit( 1 );
}

// Read the .INI file
ini_file( argv[0] );

// Parse the command line
i = 1;
while( argv[i] != NULL ) {
   if( argv[i][0] == '-' || argv[i][0] == '/' ) {
      switch( argv[i][1] ) {
         case 'v':
         case 'V':
            verbose = 1;
            break;
         case 'w':
         case 'W':
            if( ( chkwait = (time_t)atol( &argv[i][3] ) ) == 0 ) {
               printf( "Invalid switch: %s\n\n", argv[i] );
               break;
            }
            if( chkwait < 0 ) {
               printf( "Wait time must be > 0 - %s\n\n", argv[i] );
               break;
            }
            break;
         case 'i':
         case 'I':
            if( ( inact = (time_t)atol( &argv[i][3] ) ) == 0 ) {
               printf( "Invalid switch: %s\n\n", argv[i] );
               break;
            }
            if( chkwait < 0 ) {
               printf( "Inactive time must be > 0 - %s\n\n", argv[i] );
               break;
            }
            break;
         case 'm':
         case 'M':
            strcpy( mask, &argv[i][3] );
            break;
         case 's':
         case 'S':
            printf( "Safety is ON, source files will not be removed.\n\n" );
            safe = 1;
            break;
         default:
            printf( "Unrecognized switch: %s\n\n", argv[i] );
            break;
      }
      i++;
      continue;
   }
   else {
      if( srcdir[0] == '?' ) {
         strcpy( srcdir, argv[i] );
         i++;
         continue;
      }
      if( desdir[0] == '?' ) {
         strcpy( desdir, argv[i] );
         i++;
         continue;
      }
      printf( "Unrecognized switch: %s\n\n", argv[i] );
   }
}

getcwd( curdir, _MAX_PATH );
```

```
// Check out the source directory
if( srcdir[0] == '?' ) {
   printf( "ERROR: No source directory specified.\n\n" );
   exit( 1 );
}
_fullpath( buffer, srcdir, _MAX_PATH );
strcpy( srcdir, buffer );
if( chdir( srcdir ) != 0 ) {
   printf( "ERROR: Source directory does not exist: %s\n\n", srcdir );
   exit( 1 );
}
chdir( curdir );

// Check out the destination directory
if( desdir[0] == '?' )
   strcpy( desdir, curdir );
_fullpath( buffer, desdir, _MAX_PATH );
strcpy( desdir, buffer );
//strcat( buffer, "\\." );
if( chdir( desdir ) != 0 ) {
   printf( "ERROR: Destination directory does not exist: %s\n\n", desdir );
   exit( 1 );
}
chdir( curdir );

upper( srcdir );
upper( desdir );
upper( mask );

srcdrv = disk_space( srcdir );
desdrv = disk_space( desdir );

if( verbose ) {
   printf( "SUDSMAN - SUDS file manager, Version %.4s, R.Banfill\n\n", version );
   printf( " Source file mask: %s\n", mask );
   printf( " Source directory: %s\n", srcdir );
   printf( "Working directory: %s\n\n", desdir );
}

printf( "     Source drive: %lu bytes available.\n", srcdrv );
printf( "Destination drive: %lu bytes available.\n\n", desdrv );


// Lets get to it
while( 1 ) {

   // Get a list of .WVM files
   nfiles = dir_list( srcdir );

   // If theres more than one...
   if( nfiles > 1 || flag ) {
      flag = 0;

      if( kbhit( ) != 0 )
         if( getch( ) == 27 )
            break;

      // Move the file
      curfile = move_file( nfiles );
      ptr = strchr( curfile, '.' );
      *ptr = '\0';

      // Set SUDSFILE
      sprintf( envstr, "SUDSFILE=%s", curfile );
      if( putenv( envstr ) != 0 ) {
         printf(" ERROR: Out of environment.\n\n" );
         exit( 1 );
      }

      // Surrender the machine to SHELL
      i = spawnvp( P_OVERLAY, "shell.exe", argv );
```

```
            printf( "\nERROR: Could not spawn SHELL.EXE %d\n", i );
            exit( i );
         }
         else if( nfiles == 1 ) {

            // Move log files
         /* nfiles = dir_list( srcdir, "*.LOG" );
            while( nfiles > 1 ) {
               if( kbhit( ) != 0 )
                  if( getch( ) == 27 )
                     break;
               curfile = move_file( nfiles );
               nfiles = dir_list( srcdir, "*.LOG" );
            }
         */
            // Go to sleep
            if( sleep( inact ) != 0 )
               break;

            // Set flag and continue
            flag = 1;
            continue;
         }

         // Quit if safety is on
         if( safe )
            break;

         // Go to sleep
         if( sleep( chkwait ) != 0 )
            break;
      }
      if( verbose )
         printf( "\nOff-line\n\n" );
      exit( 1 );
   }

   //--------------------------------------------------------------------
   char *move_file( int nfiles ) {
      char sys[_MAX_PATH];

      // Sort the filenames (ascending)
      qsort( (void *)base, (size_t)nfiles, sizeof(char)*13, compare );

      if( verbose )
         printf( "\r                                                                " );
         printf( "\rMoving: %s\\%s", srcdir, base );

      // Copy the file
      sprintf( sys, "COPY %s\\%s %s > NUL", srcdir, base, desdir );
      if( system( sys ) == -1 ) {
         printf( "\nERROR: Unable to invoke Command.com.\n\n" );
         exit( 1 );
      }

      // Delete the source file
      if( ! safe ) {
         sprintf( sys, "DEL %s\\%s", srcdir, base );
         if( system( sys ) == -1 ) {
            printf( "\nERROR: Unable to invoke Command.com.\n\n" );
            exit( 1 );
         }
      }

      if( verbose )
         printf( "\rMoved: %s\\%s \n", srcdir, base );

      return( base );
   }

   //--------------------------------------------------------------------
   int dir_list( char srcdir[_MAX_PATH] ) {
```

```
   register i;

   if( ( base = (char *)malloc( 13*sizeof(char) ) ) == NULL ) {
      printf( "ERROR: Out of memory.\n\n" );
      exit( 10 );
   }

   strcpy( buffer, srcdir );
   strcat( buffer, "\\" );
   strcat( buffer, mask );

   if( _dos_findfirst( buffer, _A_NORMAL, &find ) != 0 )
      return( 0 );

   strcpy( base, find.name );
   i = 1;

   while( _dos_findnext( &find ) == 0 ) {
      if( ( base = (char *)realloc( base, ((i*13)+13)*sizeof(char) ) ) == NULL ) {
         printf( "ERROR: Out of memory.\n\n" );
         exit( 10 );
      }
      strcpy( base+(i*13), find.name );
      i++;
   }
   return( i );
}

//---------------------------------------------------------------------
int sleep( time_t sec ) {

   time( &cur_t );
   stop_t = cur_t+sec;


   while( cur_t < stop_t ) {
      if( kbhit( ) != 0 )
         if( getch( ) == 27 )
            return( 1 );
      time( &cur_t );
      if( cur_t == old_t )
         continue;
      ltime = localtime( &cur_t );
      strftime( buffer, _MAX_PATH, "Sleeping: %a %m-%d-%y %H:%M:%S", ltime );
      printf( "\r%s", buffer );
      old_t = cur_t;
   }
   return( 0 );
}

//---------------------------------------------------------------------
void ini_file( char *progname ) {
   FILE *handle;
   register i;
   char *ptr, filename[_MAX_PATH];

   strcpy( filename, progname );
   ptr = strchr( filename, '.' );
   strcpy( ptr, ".INI" );

   handle = fopen( filename, "r" );

   while( fgets( buffer, _MAX_PATH, handle ) != NULL ) {
      upper( buffer );
      if( strncmp( buffer, "MASK", 4 ) == 0 ) {
         ptr = strchr( buffer, '=' )+1;
         strcpy( mask, ptr );
         if( ( ptr = strchr( mask, '\n' ) ) != NULL )
            *ptr = '\0';
         continue;
      }
      if( strncmp( buffer, "SOURCE", 6 ) == 0 ) {
```

```
            ptr = strchr( buffer, '=' )+1;
            strcpy( srcdir, ptr );
            if( ( ptr = strchr( srcdir, '\n' ) ) != NULL )
               *ptr = '\0';
            continue;
         }
         if( strncmp( buffer, "DESTINATION", 11 ) == 0 ) {
            ptr = strchr( buffer, '=' )+1;
            strcpy( desdir, ptr );
            if( ( ptr = strchr( desdir, '\n' ) ) != NULL )
               *ptr = '\0';
            continue;
         }
         if( strncmp( buffer, "WAIT", 4 ) == 0 ) {
            ptr = strchr( buffer, '=' )+1;
            chkwait = (time_t)atol( ptr );
            continue;
         }
         if( strncmp( buffer, "INACTIVE", 8 ) == 0 ) {
            ptr = strchr( buffer, '=' )+1;
            inact = (time_t)atol( ptr );
            continue;
         }
         if( strncmp( buffer, "VERBOSE", 7 ) == 0 )
            verbose = 1;
      }
      fclose( handle );
      return;
}

//---------------------------------------------------------------------------
unsigned long disk_space( char *path ) {
      struct diskfree_t drvinfo;
      unsigned drive;

      strcpy( buffer, path );
      drive = (unsigned)buffer[0]-'A'+1;
      _dos_getdiskfree( drive, &drvinfo );
      return( (long)drvinfo.avail_clusters *
              drvinfo.sectors_per_cluster *
              drvinfo.bytes_per_sector );
}

//---------------------------------------------------------------------------
int compare( char *file1, char *file2 ) {
      return( strcmp( file1, file2 ) );
}

//---------------------------------------------------------------------------
void upper( char buffer[_MAX_PATH] )
{
      char *p;

      for( p = buffer; *p; p++ )
         *p = toupper( *p );
      return( p );
}
```

SUDSSQZ

A Program to Remove Insignificant Data from
SUDS Format Data Files

Version 1.12
May 1992

Robert Banfill
Small Systems Support
2 Boston Harbor Place
Big Water, UT 84741-0205

SUDSSQZ Version 1.12

4.1 Introduction

This program is used to clean up and reduce the size of SUDS files. If a phase list is available, the program marks all traces that have phases picked as good. If the HYPO71PC.PRT file is available, the program marks all traces used in the epicenter location as good and then extracts the event location and station locations from the HYPO71PC.PRT file, calculates the epicentral distance of each bad trace (based on a 5.5 km/sec velocity model), and then places phase arrivals on the bad trace. The program then finds the averages of the pre-event, P phase, S phase portions and the average of the whole trace and based on these averages decides whether or not to mark the trace as good. Once this processing is complete, the program writes out the good traces in order of epicentral distance to the output file.

4.2 Command line syntax

SUDSSQZ is controlled entirely from the DOS command line using the following syntax:

```
SUDSSQZ [switches] inputfile outputfile <return>

switches:
      /Slist  Station list filespec. (HYPO71PC.PRT)

() indicates default values and arguments may be placed in any order, and
are not case sensitive.
```

If a phase file with the same basename as the inputfile and an extension of .PHA is available (*i.e.*, if the inputfile spec is C:\DATA\90102101.DMX, then SUDSSQZ will look for a phase file named C:\DATA\90102101.PHA), all stations in this list will be marked as good.

By default, SUDSSQZ will look for a station list named HYPO71PC.PRT in the current working directory, the /S option allows you to give a different file spec for the station list.

The inputfile and outputfile must not have the same name. I use the convention that the inputfile have the extension .DMX, and the outputfile have the extension .SQZ.

Sun 16-Aug-1992 15:48, RB

>>> Notes on SUDSSQZ <<<

SUDSSQZ was written using Microsoft C 6.00AX.
The makefile provided is for use with the PWB.

Libraries required:
   HSUDS.LIB   - SUDS data file library version 1.31.
                  Available from:
                     Small Systems Support
                     2 Boston Harbor Place
                     Big Water, UT 84741-0205
                     (801) 675-5827 Voice
                     (801) 675-3730 FAX

The following files are included:

SUDSSQZ C   17995 07-15-91   1:14p

SUDSSQZ MAK   1959 07-15-91   1:11p

```
/*
 * SUDSSQZ - Squeeze SUDS files - Remove bad traces, sort by epicentral distance
 *
 * Filespec: d:\code\sudssqz\sudssqz.c
 * Last edit: 12-14-90, RB
 *
 * Written by R. Banfill, Small Systems Support, Big Water, Utah.
 *
 * Written in Microsoft C 6.00.
 * Medium Memory Model, Emulator Math Package.
 *
 * Uses the Seismic Unified Data System (SUDS) File Library.
 *
 * Revision history:
 *    Version 1.00 - 12/03/90
 *       First working version.
 *    Version 1.10 - 12/14/90
 *  Added; Analyze unpicked traces, general cleanup.
 *    Version 1.11 - 12/26/90
 *       Added /S to specify station list file on the command line.
 *    Version 1.12 - 1/18/91
 *       Fixed time correction problem.
 */

/* Standard include files */
#include <malloc.h>
#include <stdio.h>
#include <process.h>
#include <string.h>
#include <stdlib.h>
#include <math.h>

/* SUDS include files */
#include <suds.h>
#include <suds_gbl.h>
#include <suds_pro.h>
#include <st_error.h>

#define PI (4.*atan(1.))

/* Prototypes */
void sudssqz( char[128], char[128], char[128] );
int builddb( char[128], char[128] );
void epidis( int );
void analyze( int, char[128] );
void sort( int );
void cleanup( int, char[128], char[128] );
void upper( char[128] );


/* Global data */
struct DB { /* --- Station database --- */
   char stn[5];    /* Station name */
   char stat[5];   /* Trace status */
   float lat;      /* Station latitude */
   float lng;      /* Station longitude */
   float epdis;    /* Epicentral distance */
} db[128];

struct FI { /* --- SUDS file index --- */
   int type;       /* Structure type */
   int pos;        /* Position index */
   char stn[5];    /* Station name */
} fi[512];
int fii;

/* Event origin */
float evnlat, evnlng, evntim;

extern char *progname;

/* Main Routine */
```

```
main( int argc, char **argv )
{
   FILE *hyfil;
   register i;
   char *j, *k;
   char buffer[128], infile[128], outfile[128], path[128], stnfile[128];

   progname = argv[0];

   strcpy( stnfile, "HYPO71PC.PRT" );
   strcpy( infile, "UNDEFINED" );
   strcpy( outfile, "UNDEFINED" );

   printf( "SUDSSQZ Version 1.12 - R.Banfill\n\n" );

   /* Deal with command-line arguments */
   if( argv[1][0] == '?' || argv[1][1] == '?' || argc < 3 )
   {
      printf ( "\nUsage: SUDSSQZ [switches] input_file output_file <~ \n\n" );
      printf ( "switches:\n" );
      printf ( "      /Slist  Station list filespec. (HYPO71PC.PRT)\n\n" );
      printf ( "              () indicates default values.\n\n");
      printf ( "Arguments may be placed in any order, and are not case sensitive.\n" );
      exit( 1 );
   }

   for( i=1; i<=argc; i++ )
   {
      if( argv[i][0] == '-' || argv[i][0] == '/' )
      {
         switch( argv[i][1] )
         {
            case 's':
            case 'S':
               strcpy( stnfile, &argv[i][2] );
               break;
         }
      }
      else
      {
         if( strncmp( infile, "UNDEFINED", 9 ) == 0 )
         {
            strcpy( infile, argv[i] );
            j = strstr( infile, "." );
            if ( j == NULL )
               strcat( infile, ".dmx" );
         }
         else if( strncmp( outfile, "UNDEFINED", 9 ) == 0 )
         {
            strcpy( outfile, argv[i] );
            j = strstr( outfile, "." );
            if ( j == NULL )
               strcat( outfile, ".dmx" );
         }
      }
   }

   /* Expand and valify */
   if( _fullpath( buffer, infile, 128 ) != NULL )
      strcpy( infile, buffer );
   else
   {
      printf( "Invalid input filespec: %s\n", infile );
      exit( 1 );
   }
   if( _fullpath( buffer, outfile, 128 ) != NULL )
      strcpy( outfile, buffer );
   else
   {
      printf( "Invalid output filespec: %s\n", outfile );
      exit( 1 );
   }
```

```
   if( _fullpath( buffer, stnfile, 128 ) != NULL )
      strcpy( stnfile, buffer );
   else
   {
      printf( "Invalid station list filespec: %s\n", stnfile );
      exit( 1 );
   }

   /* Convert everything to upper case */
   upper( infile );
   upper( outfile );
   upper( stnfile );

   printf( "Input filespec:           %s\n", infile );
   printf( "Output filespec:          %s\n", outfile );
   if( ( hyfil = fopen( stnfile, "r" ) ) != NULL )
   {
      fclose( hyfil );
      printf( "Station list filespec:    %s\n", stnfile );
   }
   else
      printf( "\nCannot find Station list: %s\n", stnfile );

   sudssqz( infile, outfile, stnfile );
   exit( 0 );
}

/* SUDSSQZ ------------------------------------------------------------*/
void sudssqz( char infile[128], char outfile[128], char stnfile[128] )
{
   int dbi;

   /* Build the station database */
   dbi = builddb( infile, stnfile );
   if( dbi <= 0 )
      return;

   /* Calculate epicentral distances */
   epidis( dbi );

   /* Analyze bad traces */
   analyze( dbi, infile );

   /* Sort stations */
   sort( dbi );

   /* cleanup file */
   cleanup( dbi, infile, outfile );

   return;
}

/* BuildDB ------------------------------------------------------------*/
int builddb( char infile[128], char stnfile[128] )
{
   register i;
   FILE *infil, *phfil, *hyfil;
   char *ptr, buffer[136], phfile[128], *j;
   int typ, len, dbi, flag, ideg, picked, hypo, irig;
   long inp;
   float fmin;
   static SUDS_STATIDENT *st;

   picked = 0;
   hypo = 0;
   irig = 0;

   /* Get station names from input file */
   infil = st_open( infile, "r+b" );
   fii = -1;
   dbi = -1;
   while( ( inp = st_get( &ptr, &typ, &len, infil ) ) != EOF )
```

```
   {
      fii++;
      fi[fii].type = typ;
      fi[fii].pos = st_tell( infil );

      if( typ == STATIONCOMP ||
          typ == TRIGGERS ||
          typ == DESCRIPTRACE )
      {
         st = (SUDS_STATIDENT *)ptr;
         strcpy( fi[fii].stn, st->st_name );
         if( typ == STATIONCOMP )
         {
            dbi++;
            strcpy( db[dbi].stn, st->st_name );
         }
      }
      st_free( ptr, inp );
   }
   st_close( infil );

   /* Mark time as good, all others as bad */
   for( i=0; i<=127; i++ )
   {
      if( strncmp( db[i].stn, "IRIG", 4 ) == 0 )
      {
         strcpy( db[i].stat, "GOOD" );
         irig = 1;
      }
      else
         strcpy( db[i].stat, "BAD " );
   }

   /* Check for stations in the phase file */
   strcpy( buffer, infile );
   j = strrchr( buffer, '.' );
   strcpy( j, ".PHA" );
   strcpy( phfile, buffer );
   if( ( phfil = fopen( phfile, "r" ) ) != NULL )
   {
      printf( "Phase file:                %s\n", phfile );
      while( fgets( buffer, 128, phfil ) != NULL )
      {
         for( i=0; i<dbi; i++ )
         {
            if( strncmp( db[i].stn, buffer, 4 ) == 0 )
            {
               strcpy( db[i].stat, "GOOD" );
               picked += 1;
               continue;
            }
         }
      }
      fclose( phfil );
   }
   else
      printf( "Cannot find Phase file:   %s\n", phfile );

   /* Check stations against HYPO output file */
   if( ( hyfil = fopen( stnfile, "r" ) ) != NULL )
   {
      flag = 0;
      while( fgets( buffer, 136, hyfil ) != NULL )
      {
/*
0123456789a123456789b123456789c123456789
  DATE    ORIGIN    LAT N    LONG W    DEPTH    MAG NO DM GAP M  RMS  ERH  ERZ Q SQD  ADJ IN I
SDFM I
 901021 13 4 59.61 36-37.42 121- 8.44  12.12        19  6  37 1 0.20  0.7  1.4 B B|A 3.11  0 2
 0.0 3
*/
         if( strncmp( buffer, "  DATE", 6 ) == 0 )
```

```
         {
            fgets( buffer, 136, hyfil );
            sscanf( &buffer[19], "%2i", &ideg );
            sscanf( &buffer[22], "%f", &fmin );
            evnlat = (float)ideg+(fmin/60.0);
            sscanf( &buffer[28], "%3i", &ideg );
            sscanf( &buffer[32], "%f", &fmin );
            evnlng = (float)ideg+(fmin/60.0);
            sscanf( &buffer[10], "%2i", &ideg );
            sscanf( &buffer[13], "%f", &fmin );
            evntim = ((float)ideg*60.0)+fmin;
            continue;
         }
         if( strncmp( buffer, "  STN", 5 ) == 0 )
         {
            flag = 1;
            continue;
         }
         if( flag != 1 )
            continue;
         else
         {
            if( strncmp( buffer, "\n", 1 ) == 0 )
            {
               flag = 0;
               continue;
            }
            for( i=0; i<=dbi; i++ )
            {
               if( strncmp( db[i].stn, &buffer[1], 4 ) == 0 )
               {
                  strcpy( db[i].stat, "GOOD" );
                  sscanf( &buffer[6], "%f", &db[i].epdis );
                  hypo += 1;
                  continue;
               }
            }
         }
      }
      fclose( hyfil );
   }

   /* Get latitudes and longitudes */
   if( ( hyfil = fopen( stnfile, "r" ) ) != NULL )
   {
      flag = 0;
      while( fgets( buffer, 136, hyfil ) != NULL )
      {
/*
0123456789a123456789b123456789c123456789
    L     STN      LAT      LONG     ELV DELAY      FMGC   XMGC KL   PRR   CALR IC       DATE HRMN
    1    NFIV 3741.90N 123 0.00W  107  0.00      0.00    0.00 0 0.00    0.00 0           0    0
    2    NTAV 3755.43N 12235.70W  768  0.00      0.00    0.00 0 0.00    0.00 0           0    0
*/
                           /*  0123456789a12 */
         if( strncmp( buffer, "    L     STN", 13 ) == 0 )
         {
            flag = 1;
            continue;
         }
         if( flag != 1 )
            continue;
         else
         {
            if( strncmp( buffer, "\n", 1 ) == 0 )
               break;
            for( i=0; i<=dbi; i++ )
            {
               if( strncmp( db[i].stn, &buffer[9], 4 ) == 0 )
               {
                  sscanf( &buffer[14], "%2i", &ideg );
                  sscanf( &buffer[16], "%f", &fmin );
```

```
                    db[i].lat = (float)ideg+(fmin/60.0);
                    sscanf( &buffer[23], "%3i", &ideg );
                    sscanf( &buffer[26], "%f", &fmin );
                    db[i].lng = (float)ideg+(fmin/60.0);
                    continue;
                 }
              }
           }
        }
        fclose( hyfil );
     }
     if( hypo == 0 && picked == 0 )
     {
        printf( "\nError: Must provided a Phase file and/or Station list.\n" );
        return( -1 );
     }
     printf("\n%i traces contained in input file.\n", dbi+1 );
     if( irig == 1 )
        printf( "IRIG trace was saved.\n" );
     if( picked > 0 )
        printf("%i traces picked in phase file.\n", picked );
     if( hypo > 0 )
        printf("%i traces used for location.\n", hypo );
     return( dbi );
}

/* Epidis ------------------------------------------------------------*/
void epidis( int dbi )
{
     register i;
     double a, a1, a2, a3, b, b1, b2, b3, phi, sen2, sen4, co, x, y;

     /* Constants for calculation */
     a1 = 1.8553654;
     a2 = 0.0062792;
     a3 = 0.0000319;
     b1 = 1.8428071;
     b2 = 0.0187098;
     b3 = 0.0001583;

     /* Calculate epicentral distances */
     for( i=0; i<=dbi; i++ )
     {
        if( db[i].epdis == 0.0 && strncmp( db[i].stn, "IRIG", 4 ) != 0 )
        {
           phi = (db[i].lat-evnlat)*0.5*PI;
           sen2 = pow( sin( phi ), 2 );
           sen4 = pow( sen2, 2 );
           co = cos( phi );
           a = (a1+a2*sen2+a3*sen4)*co;
           b = b1+b2*sen2+b3*sen4;
           x = 60.0*a*(db[i].lng-evnlng);
           y = 60.0*b*(db[i].lat-evnlat);
           db[i].epdis = sqrt( pow( x, 2 )+pow( y, 2 ) );
        }
     }
     return;
}

/* Analyze -----------------------------------------------------------*/
void analyze( int dbi, char infile[128] )
{
     register k, i, j;
     FILE *infil;
     char *ptr, samtime[34];
     int typ, len, isk, *sam, imin, ana;
     long inp, psam, ssam, noi, pwv, swv, whl, leng;
     float ist, rate, ptime, stime, fsec;
     static SUDS_DESCRIPTRACE *dt;
     MS_TIME istime;

     ana = 0;
```

```
infil = st_open( infile, "r+b" );

/* Look at bad traces */
for( i=0; i<=dbi; i++ )
{
   if( strncmp( db[i].stat, "GOOD", 4 ) != 0 && db[i].epdis != 0. )
   {
      for( j=0; j<=fii; j++ )
      {
         if( strncmp( fi[j].stn, db[i].stn, 4 ) == 0 &&
             fi[j].type == DESCRIPTRACE )
         {
            isk = st_seek( infil, (long)fi[j].pos-1, 0 );
            inp = st_get( &ptr, &typ, &len, infil );

            dt = (SUDS_DESCRIPTRACE *)ptr;
            sam = (int *)ptr+len;
            leng = dt->length-len;
            rate = dt->rate+dt->rate_correct;
            istime = dt->begintime+dt->time_correct;

            /* Get initial sample time */
            if( dt->begintime==atof(NOTIME) )
            {
               sprintf( samtime,"???" );
               ist = 0.0;
            }
            else
            {
               sprintf( samtime, "%s", list_mstime( istime, 4 ) );
            /* 0123456789a123456789b
               MM/DD/YY HH:MM SS.HHH */
               sscanf( &samtime[12], "%2i", &imin );
               sscanf( &samtime[15], "%f", &fsec );
               ist = ((float)imin*60.0)+fsec;
            }

            /* P and S time based on 5.5 km/sec velocity */
            ptime = evntim + (db[i].epdis / 5.5);
            stime = evntim + (db[i].epdis / 3.0);
            psam = (ptime - ist) * rate;
            ssam = (stime - ist) * rate;

            /* Check bounds */
            if( psam <= 100 || ssam >= leng-100 )
            {
               st_free( ptr, inp );
               break;
            }

            /* Establish baseline */
            for( k=0; k<leng; k++ )
            {
               *sam -= 2048;
               sam++;
            }
            sam = (int *)ptr+len;

            /* Calc average noise based on 1st dif */
            noi = 0;
            for( k=0; k<psam; k++ )
            {
               noi += (long)abs( *(sam+1) - *sam );
               sam ++;
            }
            whl = noi;
            noi /= (long)k;

            /* Average P wave */
            pwv = 0;
            for( k=psam; k<ssam; k++ )
```

```
                {
                   pwv += (long)abs( *(sam+1) - *sam );
                   sam ++;
                }
                whl += pwv;
                pwv /= ((long)k - (long)psam);

                /* Average S wave */
                swv = 0;
                for( k=ssam; k<leng; k++ )
                {
                   swv += (long)abs( *(sam+1) - *sam );
                   sam ++;
                }
                whl += swv;
                swv /= ((long)k - (long)ssam);
                whl /= leng;

                /* Decide if the trace is worth keeping */
                if( whl >= noi*2 &&
                    pwv >= noi*3 &&
                    swv >= noi*2 )
                {
                   strcpy( db[i].stat, "GOOD" );
                   ana += 1;
                }

                st_free( ptr, inp );
             }
          }
       }
    }
    st_close( infil );

    printf("%i additional traces analyzed as good.\n", ana );
    return;
}


/* Cleanup ------------------------------------------------------------*/
void cleanup( int dbi, char infile[128], char outfile[128] )
{
    register i, j;
    FILE *infil, *outfil;
    char *ptr;
    int typ, len, outp, isk, count;
    long inp;
    static SUDS_STATIDENT *st;

    count = 0;

    infil = st_open( infile, "r+b" );
    if( ( outfil = st_open( outfile, "w+b" ) ) == NULL )
    {
       printf( "Unable to open output file: %s\n\n", outfile );
       die( 1 );
    }

    /* Write out non station info */
    for( i=0; i<=fii; i++ )
    {
       if( fi[i].type == DETECTOR ||
           fi[i].type == ATODINFO ||
           fi[i].type == TRIGSETTING ||
           fi[i].type == EVENTSETTING ||
           fi[i].type == TIMECORRECTION ||
           fi[i].type == ORIGIN )
       {
          isk = st_seek( infil, (long)fi[i].pos-1, 0 );
          inp = st_get( &ptr, &typ, &len, infil );
          outp = st_put( ptr, typ, (int)inp, outfil );
          strcpy( fi[i].stn, "****" );
```

```
            st_free( ptr, inp );
         }
      }

      /* Write out station oriented data */
      for( i=0; i<=dbi; i++ )
      {
         if( strncmp( db[i].stat, "GOOD", 4 ) == 0 )
         {
            for( j=0; j<=fii; j++ )
            {
               if( strncmp( fi[j].stn, db[i].stn, 4 ) == 0 )
               {
                  isk = st_seek( infil, (long)fi[j].pos-1, 0 );
                  inp = st_get( &ptr, &typ, &len, infil );
                  outp = st_put( ptr, typ, (int)inp, outfil );
                  strcpy( fi[j].stn, "****" );
                  if( typ == DESCRIPTRACE )
                     count += 1;
                  st_free( ptr, inp );
               }
            }
         }
      }
      st_close( outfil );
      st_close( infil );

      printf( "%i traces written to output file.\n", count );
      return;
}


/* Sort ---------------------------------------------------------------*/
void sort( int dbi )
{
      register i, j;
      struct DB temp;

      for( i=0; i<=dbi-1; i++ )
      {
         for( j=1; j<=dbi; j++ )
         {
            if( db[i].epdis < db[j].epdis )
            {
               memcpy( &temp, &db[i].stn, sizeof( temp ) );
               memcpy( &db[i].stn, &db[j].stn, sizeof( temp ) );
               memcpy( &db[j].stn, &temp, sizeof( temp ) );
            }
         }
      }
      return;
}

/* Die ----------------------------------------------------------------*/
long die( int in )
{
      exit(in);
}

/* Upper --------------------------------------------------------------*/
void upper( char buffer[128] )
{
      char *p;

      for( p = buffer; *p; p++ )
         *p = toupper( *p );
      return;
}
```

PC-QMAP

A Program to Create Epicenter Maps

Version 1.10
July 1991

Robert Banfill
Small Systems Support
P.O. Box 410205
Big Water, Utah 84741-0205
(801) 675-5827

(page 134 follows)

Contents

(page 136 follows)

Chapter 1
Getting Started

1.1 Overview

PC-QMAP is a program that generates maps containing earthquake information. This program was originally written by W. H. K. Lee and has been used as the basis for many mapping programs over the past 22 years. This program is a straight port of the original QMAP1 FORTRAN source code with a few small enhancements.

This program takes control, station location, earthquake data and line data input from an unformatted ASCII text file (.INP), renders the map to the display and generates a PostScript® page description, or an HP Graphics Language file, that will in turn generate the map.

1.2 System Requirements

The use of this software requires an IBM-PC compatible computer running IBM PC-DOS or MS-DOS version 3.20 or later. This program uses drivers created by the Geograf Graphics Utilities to render a graphical image to the display and therefore one of the following display adapter/monitor combinations is required:

IBM Color Graphics Adapter @ 640x200 (CGA.DRV)
IBM Enhanced Graphics Adapter @ 640x350 (EGA.DRV)
IBM Video Graphics Array @ 640x480 (VGA.DRV)
Hercules Graphics Card @ 720x348 (HGC.DRV)
Compaq Gas Plasma Display @ 640x400 (GPD.DRV)

This version of QMAP provides two executable programs. QMAPPS.EXE, which generates PostScript output for use with any PostScript device, and QMAPHP.EXE which generates both a HPGL plot file, and a HPGL-2 file for use with HP Laserjet III or III*p* laser printers

To use the PostScript version of QMAP, the following three files must be in a directory specified in the PATH environment variable:

```
QMAPPS.EXE
SCREEN.DRV
QMAP_HDR.PS
```

To use the HPGL version of QMAP the following two files must be in a directory specified in the PATH environment variable:

```
QMAPHP.EXE
SCREEN.DRV
```

QMAP looks for these files in its home directory (*i.e.*, the directory where the executable is located), so all three files must be in the same directory. The appropriate screen driver (see above) should be renamed SCREEN.DRV. For example, if your computer has a VGA display, you would use the following command, assuming the current directory is where you installed QMAP:

```
COPY VGA.DRV SCREEN.DRV
```

You may need to set an infinite retry limit on the port used by QMAPHP if you receive a "Device time-out" or "Device not ready" error message while generating a map. The following command will accomplish this:

```
MODE LPT1:,,P <return>
```

If your printer is connected to a port other than LPT1:, substitute its name for LPT1: in the above command.

If you are using a serial port (COM1: through COM4:), you may need to initialize the port before using the printer. This is also accomplished with the **MODE** command. Please refer to your DOS User Reference for more information on use to initialize a serial port by using the **MODE** command.

Chapter 2
Using PC-QMAP

2.1 Operation

When you execute QMAP you will be prompted for an input filespec (.INP file), the default filespec is QMAP.INP. The INP file is an unformatted ASCII text file that contains control information, station locations, earthquake data and line data. You may enter any DOS legal file specification at this prompt. A sample INP file, LOMA.INP is provided for testing purposes, this file generates a large map (15 sheets) of the Loma Prieta aftershock sequence.

The next prompt will ask for the log filespec (LOG file), the default filespec is the same that you specified for the INP file except with a .LOG extension. Because QMAP will monopolize the display for graphical purposes, all program feedback will be directed to this file for viewing after program execution. This file will contain a summary of all control information, all station locations, all earthquake data and any warnings or errors encountered during execution (*e.g.*, data points outside the map boundaries).

If you are using the PostScript version of QMAP, the final prompt will ask for the PostScript output filespec (.PS file), the default is the same that you as the .INP file except with a .PS extension. This PostScript page description assumes an 8½×11 inch page size and will tile as many pages as necessary to accommodate the map. Each sheet will be labeled with its row:column position in the map and registration marks will be printed at each corner to aid in assembly.

If you are using the HP version of QMAP, the final prompt will ask for the HPGL plot file specification (.PLT file). This file will contain the pen commands that will draw the map on a HP plotter. A second file, given the same filespec as the .PLT file except with an .LJ3 extension, will contain the commands needed to tile the map onto 8½×11 inch pages. Again, each sheet will be labeled with its row:column position and registration marks.

After the prompts, the program will display the map on the screen. The map appears on its side so that it covers the entire display. The gray lines indicate single 8½×11 inch sheets of paper. Stations are displayed as yellow triangles, earthquakes as bright white symbols as specified in the .INP file and line data is drawn in brown. Once all data is displayed, four corners of a small box will be displayed in the lower left hand corner in bright white. This represents a bounding box for the map legend. You may move this box using the arrow keys on the numeric key pad. The bounding box moves in large steps by default, you may, however, move the box a single pixel at a time by holding down a shift key while pressing one of the arrow keys. Once you have positioned the bounding box, press any key to display the legend at that position. Once the legend has been displayed, press any key to exit to the command-line.

If you are using the HPGL version, you will see a status message as the program generates the HPGL-2 file before the program exits to the command-line.

2.2 The Input Data File

The .INP file contains control information and data used to generate the map. This file is broken into several sections, each section begins with a section marker. Each section marker begins with a $ in column 1 followed by at least the first 2 characters of the section name. Section markers are not case-sensitive. There are 6 section markers in all and they must occur in the following order in the .INP file:

1) $Control
2) $Legend
3) $Station Data
4) $Point Data (earthquakes)
5) $Line Data (coast lines, topography, faults, etc...)
6) $End

2.3 $Control Section

The Control section contains the coordinates of the map center, the extent of the map, the scale, and grid interval. The FORTRAN edit descriptor for each numeric field is provided for clarity.

Row 1 $Control

Row 2

Col 1-2	Degree portion of the map center latitude (*e.g.*, 37). Format: I2.
Col 4-8	Minute portion of the map center latitude (*e.g.*, 15.00). Format: F5.2.
Col 10-13	Degree portion of the map center longitude (*e.g.*, 122). Format: I3.
Col 15-19	Minute portion of the map center longitude (*e.g.*, 00.00). Format: F5.2.

Row 3

Col 1-5	EP-Extent of map from center to edge in minutes (*e.g.*, 60.). Format: F5.0. Since one specifies only one number, the map will be a square of twice the extent.

Row 4

Col 1-5	GI-Grid mark interval in minutes (*e.g.*, 15.). Format: F5.0. A grid mark will be plotted every GI minutes from the map center. If no grid is needed, set GI=EP. A maximum of 10 grid marks will be plotted in either direction.

Row 5

Col 1-10	Scale of map (*e.g.*, 250000). Format: F10.0.

Row 6

Col 1-5	Size of station symbol in inches (*e.g.*, 0.5). Format: F5.1.

Row 7 Blank

The following is a sample $Control section:

```
$Control
37 00.00 121 45.00
30.
7.5
250000.
.5
```

2.4 $Legend Section

The Legend section contains the title of the map and class information needed to build the map and legend.

Row 1 $Legend

Row 2 The title of the map, the length depends on the size of the map.

Row 3 The heading for the legend (*i.e.*, Magnitude, Depth, etc...).

Row 4 - *n* Classes, $0 < n \leq 20$.

- Col 1-5 Class interval lower bound. Format: F6.2.
- Col 8 Symbol to be plotted for this class. Format: A1.
- Col 10-15 Size of symbol for this class. Format: F6.3.

Row *n*+1 Blank

The following is an example $Legend section:

```
$Legend
Loma Prieta Aftershock Sequence 10-21-89
Magnitude
  0.   X   .15
  1.5  2   .2
  2.5  3   .25
  3.5  4   .3
  4.5  5   .35
  5.5  6   .4
  6.5  7   .45
  7.5  8   .5
```

In this example, earthquakes of magnitudes 0 to 1.5 will be plotted as an X that is 0.15 inches tall, earthquakes of magnitudes >1.5 to 2.5 will be plotted as a 2 that is 0.2 inches tall, and so on.

2.4 $Stations Section

This section contains station location data. The Station data can be in any format as long as the following information is provided for each station in the following order, one station per line:

A four letter station name.
Degree portion of latitude.
Minute portion of latitude.
Degree portion of longitude.
Minute portion of longitude.

The $Station section is composed as follows:

Row 1	$Stations
Row 2	FORTRAN edit list for reading data, (*i.e.*, (2X,A4,I2,F5.2,1X,I3,F5.2))
Row 3-*n*	Station data, $0 < n \leq 256$.
Row *n*+1	Blank

The following is an example $Stations section:

```
$Stations
(A4,I2,F5.2,1X,I3,F5.2)
NFIV3741.90 12300.00
NTAV3755.43 12235.70
CADV3709.83 12137.55
CAIV3751.68 12225.77
CALV3727.07 12147.95
CBWV3755.45 12206.40
CCYV3733.10 12205.45
```

2.5 $PointData Section

This section contains earthquake data. QMAP was designed specifically to use data from HYPO71PC summary cards but other data sets may be used as long as the following information is provided for each event in the following order, one event per line:

LAT: Degree portion of latitude.
XLAT: Minute portion of latitude.
LON: Degree portion of longitude.
XLON: Minute portion of longitude.
Z: Variable to be plotted.

The $PointData section is composed as follows:

Row 1	$PointData
Row 2	FORTRAN edit list for reading the data.
Row 3-*n*	Point data, $0 < n \leq 2000$.
Row *n*+1	Blank

Since the edit list is also used to read these values as text, the A edit descriptor is required and one must use the T edit descriptor to place the first numeric field (LAT). The edit list in the following example can be used to read data (Z = magnitude) from HYPO71PC summary files.

```
$PointData
(A51, T19, I2, 1X, F5.2, 1X, I3, 1X, F5.2, 9X, F5.2)
891018     4 15.20 37- 2.35 121-52.80  17.78   0.00
891018    12 42.47 37-10.15 121-59.29   6.07   5.00
891018    13  8.76 37-10.72 122- 2.49  11.95   4.30
891018    15 10.86 37- 0.39 121-48.64  15.22   3.65
891018    16 14.33 37-10.14 121-59.50   8.31   3.00
```

The A edit descriptor specifies the length of the line that will transmitted to the LOG file.

2.6 $LineData Section

This section contains coordinate pairs describing lines to be plotted on the map. The Line data can be in any format as long as the following information is provided for each pair in the following order, one pair per line:

Degree portion of latitude.
Minute portion of latitude.
Degree portion of longitude.
Minute portion of longitude.

The $LineData section is composed as follows:

Row 1	$LineData
Row 2	FORTRAN edit list for reading the data.
Row 3-*n*	Line data. The pen will be lifted when a blank line is encountered. Line data should be kept to a minimum do to the increase in the size of the PS

	or PLT file. Some printers may have trouble if more than two or three thousand points are specified.
Row *n*+1	Blank
Row *n*+2	$End

The following is an example $LineData section:

```
$LineData
(I2,F5.2,1X,I3,F5.2)
3722.50 12207.50
3637.50 12122.50

3645.00 12122.50
3722.50 12200.00

3722.50 12152.50
3652.50 12122.50

$End
```

This example will draw three diagonal lines on the map.

2.7 Sample output

The map in figure 1 was generated using the LOMA.INP file provided on the disk but has been reduced by a factor of 5. The original map is tiled onto 15 sheets of paper and measures 28 ½" × 37 ½".

After program execution, the LOG file will contain; the software version and name of the LOG file, the title of the map, all input parameters, station data numbered from 1 to *n*, earthquake data numbered from one to *n* and any warnings about data points that are outside of the map boundaries.

The following is an example of an abbreviated LOG file:

```
PC-QMAP PostScript Version 1.10 Log file: TEST.LOG

*** Control information ***
Center: 37    .00 121 45.00
Extent:    30.
Grid interval:     8.
Scale:     250000.
Station size:    .50
```

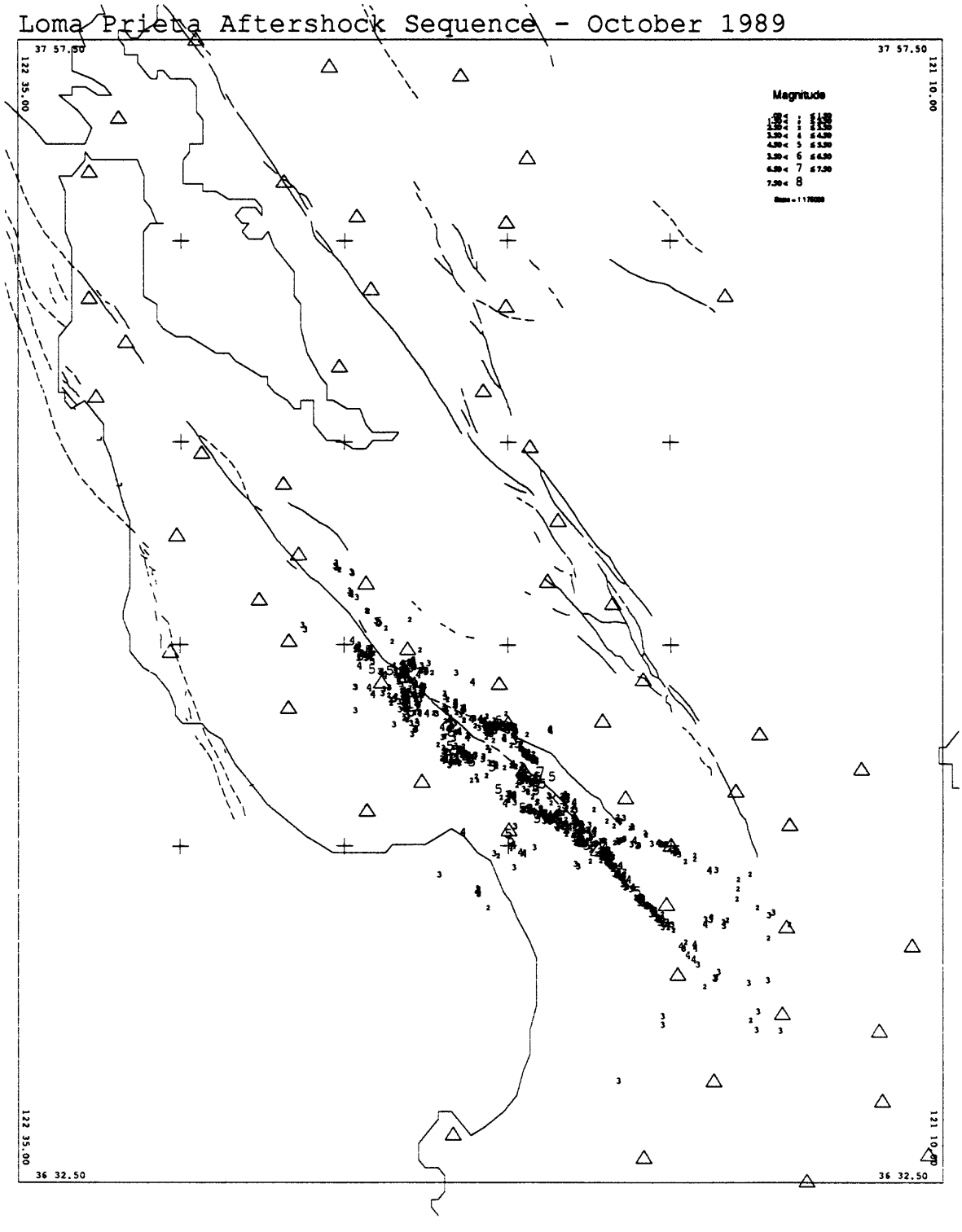

Loma Prieta Aftershock Sequence - October 1989
37 57.50
122 35.00
121 10.00
36 32.50
Magnitude

Figure 1.

```
*** Legend information ***
Title: Loma Prieta Aftershock Sequence 10-21-89
Subtitle: Magnitude
Class 1    .00 X   .150
Class 2   1.50 2   .200
Class 3   2.50 3   .250
Class 4   3.50 4   .300
Class 5   4.50 5   .350
Class 6   5.50 6   .400
Class 7   6.50 7   .450
Class 8   7.50 8   .500

*** Station data ***
  1 NFIV 37-41.90  123-  .00
  2 NTAV 37-55.43  122-35.70
  3 CADV 37- 9.83  121-37.55
  4 CAIV 37-51.68  122-25.77
  5 CALV 37-27.07  121-47.95
  6 CBWV 37-55.45  122- 6.40
  7 CCYV 37-33.10  122- 5.45
 *** DATA NO.    1 IS OUTSIDE MAP BOUNDARIES ***
 *** DATA NO.    2 IS OUTSIDE MAP BOUNDARIES ***
 *** DATA NO.    4 IS OUTSIDE MAP BOUNDARIES ***
 *** DATA NO.    6 IS OUTSIDE MAP BOUNDARIES ***
 *** DATA NO.    7 IS OUTSIDE MAP BOUNDARIES ***

*** Point data ***
    1 - 891018    4 15.20 37- 2.35 121-52.80  17.78   0.00
    2 - 891018   12 42.47 37-10.15 121-59.29   6.07   5.00
    3 - 891018   13  8.76 37-10.72 122- 2.49  11.95   4.30
    4 - 891018   15 10.86 37- 0.39 121-48.64  15.22   3.65
    5 - 891018   16 14.33 37-10.14 121-59.50   8.31   3.00
    6 - 891018   16 34.82 37-10.26 121-59.92   9.02   2.90
    7 - 891018   16 54.39 37- 4.95 121-55.32   9.96   3.75
    8 - 891018   19 17.30 37-10.00 121-59.31  10.36   3.80
    9 - 891018   20 32.52 37-10.15 121-59.19   6.95   3.10
   10 - 891018   22  7.00 37- 5.59 121-53.77   8.26   2.90
   11 - 891018   22 56.16 36-56.63 121-40.95   4.08   2.90
   12 - 891018   23 37.22 37- 2.70 121-45.97   2.04   3.95
```

❑

Sun 16-Aug-1992 15:48, RB

>>> Notes on QMAPHP <<<

QMAPHP was written using Microsoft FORTRAN 5.00.

Libraries required:

UPLOT.LIB - Uplot graphics interface library 1.10.
Available from:
Small Systems Support
2 Boston Harbor Place
Big Water, UT 84741-0205
(801) 675-5827 Voice
(801) 675-3730 FAX

GEOGRAF.LIB - GeoGraf graphics library version 4.00 for MS FORTRAN.
Available from:
GeoComp Corp.
66 Commonwealth Ave.
Concord, MA 01742
(800) 822-2669

The following sources files are included:

QMAP	FOR	11018	09-08-91	2:28p
CURSOR	FOR	5006	06-30-91	10:11p
HP	FOR	5865	09-08-91	4:57p
LABEL	FOR	958	07-01-91	5:31p
LEGEND	FOR	1980	07-01-91	5:30p
SSPLOT	FOR	2892	07-01-91	4:54p
XYCORD	FOR	3083	06-30-91	10:11p
QMAP	MAK	1061	09-08-91	2:17p

```
$TITLE: 'Filespec: d:\code\qmaphp\qmap.for'
!---    Last edit: 08-Sep-1991 14:15, RB
C---
C---  QMAP1 was originally written in July 1969 by W. H. K. Lee
C---
C---     QMAP1 was written as a general purpose program to plot a
C---  maximum of 2500 earthquakes on a map.  Each earthquake was
C---  represented by a symbol of a user specified size.
C---
C---     Original code ported to Microsoft FORTRAN 5.00 in March 1991
C---  Robert Banfill, Small Systems Support, Box 410205, Big Water, UT,
C---  84741-0205, (801) 675-5827.
C---
C---
C-------------------------------------------------------------------------
C---
C---  HP-GL Version 1.00, 30-Jun-1991, RB
C---     Started with PostScript version 1.01.
C---
C---  Version 1.10 08-Sep-1991, RB
C---     Cleaned up, Output raw HPGL (.PLT) file and HPGL-2 file (.LJ3)
C---     formatted for the Laserjet III.
C---
C-------------------------------------------------------------------------

      PROGRAM QMAP

!     Pmax = max events, Smax = max # of stn's,
!     Cmax = max classes, Gmax= max grid points
      PARAMETER (PMAX=2000, SMAX=256, CMAX=20, GMAX=200)

      REAL*4 glat(GMAX), glon(GMAX), gx(GMAX), gy(GMAX)
      REAL*4 clat(2), clon(2), cx(2), cy(2)
      REAL*4 plat(PMAX), plon(PMAX), px(PMAX), py(PMAX)
      REAL*4 slat(SMAX), slon(SMAX), sx(SMAX), sy(SMAX)
      REAL*4 sz(CMAX), c(CMAX), ht(PMAX)
      REAL*4 llat, llon, lx, ly, x, y, ds

      CHARACTER sym(PMAX)*1
      CHARACTER ista(SMAX)*4
      CHARACTER frmt*80, card*80, s(CMAX)*1
      CHARACTER plus*1, tri*1
      CHARACTER driver*64, port*64, buffer*128, ifile*64, lfile*64
      CHARACTER title*80, subtitle*64

      INTEGER*2 rows, cols
      INTEGER*4 pen

      WRITE(*,'(/,A,/)') ' PC-QMAP, HP-GL Version 1.10'

      plus = '+'
      tri = CHAR(10)

      nmax = PMAX

!     Prompt for input filespec
      ifile = 'QMAP.INP'
      WRITE(*,'(A,\)') ' Enter Input filespec (QMAP.INP)->'
      READ(*,'(A)') buffer
      IF (buffer(1:1) .NE. ' ') ifile = buffer

!     Open input file as unit 5
      OPEN (UNIT=5, FILE=ifile, STATUS='old', ERR=50000)

!     Prompt for input filespec
      lfile = ifile(:INDEX(ifile,'.',.TRUE.))//'LOG'
      CALL UPPER (lfile)
      WRITE(*,'(A,A,A,\)') ' Enter Log filespec (',
     +                     lfile(:LEN_TRIM(lfile)),')->'
      READ(*,'(A)') buffer
      IF (buffer(1:1) .NE. ' ') lfile = buffer
```

```
      CALL PLOTS (0)

!     Plot paper lines
      CALL NEWPEN (8)
      DO r=0., rowi, 7.5
         CALL PLOT (r, 0., 3)
         CALL PLOT (r, coli, 2)
      ENDDO
      DO r=0., coli, 10.
         CALL PLOT (0., r, 3)
         CALL PLOT (rowi, r, 2)
      ENDDO

!     Set origin
      CALL PLOT (1., 0.5, -3)

!     Read in legend section
20010 READ(5,'(A)') card
      CALL UPPER(card)
      IF (card(1:3) .EQ. '$LE') THEN
         READ(5,'(A)') title
         READ(5,'(A)') subtitle
         DO i=1, 20
30000       FORMAT (A,T1,F6.2,1X,A1,1X,F6.3)
            READ(5,30000) card, c(i), s(i), sz(i)
            IF (card(1:6) .EQ. '      ') THEN
               nc = i-1
               EXIT
            ENDIF
         ENDDO
      ELSE
         GOTO 20010
      ENDIF
      WRITE(6,'(/,A)') '*** Legend information ***'
      WRITE(6,'(A,A)') 'Title: ',title(:LEN_TRIM(title))
      WRITE(6,'(A,A)') 'Subtitle: ',subtitle(:LEN_TRIM(subtitle))
      DO i=1, nc
30001    FORMAT (A,I2,1X,F6.2,1X,A1,1X,F6.3)
         WRITE(6,30001) 'Class', i, c(i), s(i), sz(i)
      ENDDO

      zsz = .5

!     Plot the title and legend
      CALL NEWPEN (7)
      CALL SYMBOL (0.,0.,0.7,title,90.,LEN_TRIM(title))
      CALL HP_SY (title(:LEN_TRIM(title)), 90, -.25, 0., 0.85)

C     Set origin for subsequent plot
      CALL PLOT (.25, 0., -3)

C     Generate grid points
      ng = 2*ep/gi-1
      IF (ng .LE. 0) GO TO 22
      IF (ng .GT. 10) ng = 10
      ngp = ng*ng
      DO 20 i=1, ng
      DO 20 j=1, ng
      k = (i-1)*ng+j
      glat(k) = olat+ep-i*gi
      glon(k) = olon+ep-j*gi
      sym(k) = plus
      ht(k) = 0.5
   20 CONTINUE
      CALL XYCORD (olat, olon, glat, glon, ngp, gy, gx)

C     Plot grid points and boundaries
      CALL SSPLOT (014, gx, gy, ngp, xleft, xright, ylower, yupper,
     +             xleng, yleng, sym, ht, scale)
      GO TO 225

C     Plot boundaries
```

```
   22 CALL SSPLOT (014, gx, gy, 0, xleft, xright, ylower, yupper,
     +             xleng, yleng, sym, ht, scale)

C     Labeling lat and long of the boundaries
  225 i = clat(2)/60.
      a1 = i
      a2 = clat(2)-i*60.
      j = clon(1)/60.
      b1 = j
      b2 = clon(1)-j*60.
      i = clat(1)/60.
      c1 = i
      c2 = clat(1)-i*60.
      j = clon(2)/60.
      d1 = j
      d2 = clon(2)-j*60.
      CALL NEWPEN (7)
      CALL LABELX (0.5, 0.125, 0.25, b1, b2)
      CALL LABELX (xl-2.25, 0.125, 0.25, b1, b2)
      CALL LABELY (xl-0.125, 0.5, 0.25, a1, a2)
      CALL LABELY (xl-0.125, yl-2.0, 0.25, a1, a2)
      CALL LABELX (xl-2.25, yl-0.375, 0.25, d1, d2)
      CALL LABELX (0.5, yl-0.375, 0.25, d1, d2)
      CALL LABELY (0.375, yl-2.0, 0.25, c1, c2)
      CALL LABELY (0.375, 0.5, 0.25, c1, c2)

C     Read in station data and plot stations
      WRITE(6,'(/,A)') '*** Station data ***'
20020 READ(5,'(A)') card
      CALL UPPER(card)
      IF (card(1:3) .EQ. '$ST') THEN
         READ(5,'(A)') frmt
         DO i=1, SMAX
            READ(5,frmt) ista(i), lat, xlat, lon, xlon
            IF (ista(i)(1:4) .EQ. '    ') EXIT
            WRITE(6,31) i, ista(i), lat, xlat, lon, xlon
   31       FORMAT(I3,1X,A4,1X,I2,'-',F5.2,1X,1X,I3,'-',F5.2)
            slat(i) = 60.*REAL(lat)+xlat
            slon(i) = 60.*REAL(lon)+xlon
            sym(i) = tri
            ht(i) = ssz
         ENDDO
      ELSE
         GOTO 20020
      ENDIF
      nsp = i-1
      IF (nsp .LE. 0) GO TO 345
      CALL XYCORD (olat, olon, slat, slon, nsp, sy, sx)
      CALL NEWPEN (14)
      CALL SSPLOT (104, sx, sy, nsp, xleft, xright, ylower, yupper,
     +             xleng, yleng, sym, ht, scale)
  345 CONTINUE

C     Read in data format and data
      WRITE(6,'(/,A)') '*** Point data ***'
20030 READ(5,'(A)') card
      CALL UPPER(card)
      IF (card(1:3) .NE. '$PO') THEN
         GOTO 20030
      ELSE
         READ(5,'(A)') frmt
         n = 1
   40    READ(5,frmt) card, lat, xlat, lon, xlon, z
         IF (card(1:6) .EQ. '      ') GO TO 60
         plat(n) = 60.*REAL(lat)+xlat
         plon(n) = 60.*REAL(lon)+xlon
C        Find symbol for Z
   41    DO 50 i=1, nc
         IF (z .GT. c(i)) GO TO 45
         sym(n) = s(i)
         ht(n) = sz(i)
         GO TO 55
```

```
   45    CONTINUE
   50    CONTINUE
         sym(n) = s(nc)
         ht(n) = sz(nc)
   55    WRITE(6,56) n, card(:LEN_TRIM(card))
   56    FORMAT(I5,' - ',A)
  565    n = n+1
         IF (n .LE. nmax) GO TO 40
         WRITE(6,'(A)') '*** ERROR: To many data points, ',
     +                  'remaining points ignored ***'
      ENDIF

C     Plot data
   60 npoint = n-1
      IF (npoint .LE. 0) GO TO 80
      CALL XYCORD (olat, olon, plat, plon, npoint, py, px)
      CALL NEWPEN (15)
      CALL SSPLOT (114, px, py, npoint, xleft, xright, ylower, yupper,
     +             xleng, yleng, sym, ht, scale)

!     Read in line data
      CALL NEWPEN (6)
      ds = 1.E05/scale
20040 READ(5,'(A)',END=80) card
      CALL UPPER(card)
      IF (card(1:3) .NE. '$LI') THEN
         GOTO 20040
      ELSE
         READ(5,'(A)',END=80) frmt
         pen = 3
         DO WHILE (.TRUE.)
            READ(5,'(A)',END=80) card
            CALL UPPER(card)
            IF (card(1:6) .EQ. '      ') THEN
               pen = 3
               CYCLE
            ELSEIF (card(1:4) .EQ. '$END') THEN
               EXIT
            ENDIF
            READ(card,frmt) lat, xlat, lon, xlon
            llat = 60.*REAL(lat)+xlat
            llon = 60.*REAL(lon)+xlon
            CALL XYCORD( olat, olon, llat, llon, 1, ly, lx)
            x = (lx-xleft)*ds
            y = (ly-ylower)*ds
            CALL PLOT (x, y, pen)
            IF (pen .EQ. 2) THEN
               CALL HP_LT (x, y)
            ELSE
               CALL HP_MT (x, y)
            ENDIF
            pen = 2
         ENDDO
      ENDIF

   80 CALL NEWPEN (3)
      CALL LEGEND (.75, 2., scale, subtitle, nc, c, s, sz, rowi, coli)

      ikey = KCHARIN ()

  100 CLOSE(5)
      CLOSE(6)
      CALL PLOT (0.,0.,999)
      ierr = GCLOSE ()
      CALL HP_CLOSE (rows, cols)
      GOTO 99999

50000 WRITE(*,'(/,A,A)') ' ERROR: Cannot open ',
     +                   ifile(:LEN_TRIM(ifile))

99999 STOP' '
      END
```

```
$TITLE: 'Filespec: d:\code\qmap\cursor.for'
!---    Last edit: 13-Apr-1991 19:44, RB
!---
!---  Put up a box cursor, return x y of top middle
!---
!-------------------------------------------------------------------------

      SUBROUTINE CURSOR (x, y, xlen, ylen, uxmax, uymax)

      IMPLICIT INTEGER*2 (i-n)

      COMMON /cur/ iux, iuy, ilx, ily, iuxo, iuyo, ilxo, ilyo, sav
      INTEGER*2 iux, iuy, ilx, ily, iuxo, iuyo, ilxo, ilyo
      INTEGER*1 sav(84)

      REAL*4 x, y, xlen, ylen
      INTEGER*1 oldcolor

      INTEGER*2 SFCOL

      INCLUDE 'driver.inc'

      x = x+1.25
      y = y+.5

      xcon = xmax/uxmax
      ycon = ymax/uymax

      iux = INT2((x+xlen)*xcon)
      iuy = INT2((y+(ylen/2.))*ycon)
      ilx = INT2(x*xcon)
      ily = INT2((y-(ylen/2.))*ycon)

      oldcolor = INT1(currfcolor)

10000 CALL CURDRAW ()

!---  Get a keystroke
20000 nkey = 0
      nkey = KCHARIN()

!---  See if shifted nav key, set step size
      IF (nkey .GE. 49 .AND. nkey .LE. 57) THEN
        istep = 1
      ELSE
        istep = 10
      ENDIF

      SELECT CASE (nkey)
         CASE (19712,54) !Right
            IF (iux+istep .GE. xmax-1) GOTO 20000
            iux = iux + istep
            ilx = ilx + istep
         CASE (19200,52) !Left
            IF (ilx-istep .LT. 0) GOTO 20000
            ilx = ilx - istep
            iux = iux - istep
         CASE (18432,56) !Up
            IF (iuy+istep .GE. ymax-1) GOTO 20000
            iuy = iuy + istep
            ily = ily + istep
         CASE (20480,50) !Down
            IF (ily-istep .LT. 0) GOTO 20000
            ily = ily - istep
            iuy = iuy - istep
         CASE DEFAULT
            i2key = nkey
      END SELECT

!---  Restore the screen
30000 CALL CURRESTORE ()
```

```
!---  If not an exit key, go agian
      IF (i2key .NE. nkey) GOTO 10000

!---  Translate coords, restore current foreground color
90000 x = REAL(ilx)/xcon-1.25
      y = REAL(ily+((iuy-ily)/2))/ycon-.5
      ierr = SFCOL(oldcolor)

      RETURN
      END


!-----------------------------------------------------------------------

      SUBROUTINE CURDRAW ()

      IMPLICIT INTEGER*2 (i-n)

      COMMON /cur/ iux, iuy, ilx, ily, iuxo, iuyo, ilxo, ilyo, sav
      INTEGER*2 iux, iuy, ilx, ily, iuxo, iuyo, ilxo, ilyo
      INTEGER*1 sav(84)

      INTEGER*2 RPIXEL, WPIXEL, SFCOL

      INCLUDE 'driver.inc'

      j = 0
      ierr = SFCOL(INT1(nfcolors-1))
      iuxo = iux
      iuyo = iuy
      ilxo = ilx
      ilyo = ily

      DO i = ily, ily+10
         j = j+1
         sav(j) = RPIXEL(ilx, i)
         ierr = WPIXEL(ilx, i)
      ENDDO
      DO i = ilx+1, ilx+10
         j = j+1
         sav(j) = RPIXEL(i, ily)
         ierr = WPIXEL(i, ily)
      ENDDO
      DO i = iuy, iuy-10, -1
         j = j+1
         sav(j) = RPIXEL(iux, i)
         ierr = WPIXEL(iux, i)
      ENDDO
      DO i = iux-1, iux-10, -1
         j = j+1
         sav(j) = RPIXEL(i, iuy)
         ierr = WPIXEL(i, iuy)
      ENDDO

      DO i = ily, ily+10
         j = j+1
         sav(j) = RPIXEL(iux, i)
         ierr = WPIXEL(iux, i)
      ENDDO
      DO i = ilx+1, ilx+10
         j = j+1
         sav(j) = RPIXEL(i, iuy)
         ierr = WPIXEL(i, iuy)
      ENDDO
      DO i = iuy, iuy-10, -1
         j = j+1
         sav(j) = RPIXEL(ilx, i)
         ierr = WPIXEL(ilx, i)
      ENDDO
      DO i = iux-1, iux-10, -1
         j = j+1
         sav(j) = RPIXEL(i, ily)
```

```
      ierr = WPIXEL(i, ily)
   ENDDO

   RETURN
   END

!-------------------------------------------------------------------------

   SUBROUTINE CURRESTORE ()

   IMPLICIT INTEGER*2 (i-n)

   COMMON /cur/ iux, iuy, ilx, ily, iuxo, iuyo, ilxo, ilyo, sav
   INTEGER*2 iux, iuy, ilx, ily, iuxo, iuyo, ilxo, ilyo
   INTEGER*1 sav(84)

   INTEGER*2 WPIXEL, SFCOL

   j = 0

   DO i = ilyo, ilyo+10
      j = j+1
      ierr = SFCOL(sav(j))
      ierr = WPIXEL(ilxo, i)
   ENDDO
   DO i = ilxo+1, ilxo+10
      j = j+1
      ierr = SFCOL(sav(j))
      ierr = WPIXEL(i, ilyo)
   ENDDO
   DO i = iuyo, iuyo-10, -1
      j = j+1
      ierr = SFCOL(sav(j))
      ierr = WPIXEL(iuxo, i)
   ENDDO
   DO i = iuxo-1, iuxo-10, -1
      j = j+1
      ierr = SFCOL(sav(j))
      ierr = WPIXEL(i, iuyo)
   ENDDO

   DO i = ilyo, ilyo+10
      j = j+1
      ierr = SFCOL(sav(j))
      ierr = WPIXEL(iuxo, i)
   ENDDO
   DO i = ilxo+1, ilxo+10
      j = j+1
      ierr = SFCOL(sav(j))
      ierr = WPIXEL(i, iuyo)
   ENDDO
   DO i = iuyo, iuyo-10, -1
      j = j+1
      ierr = SFCOL(sav(j))
      ierr = WPIXEL(ilxo, i)
   ENDDO
   DO i = iuxo-1, iuxo-10, -1
      j = j+1
      ierr = SFCOL(sav(j))
      ierr = WPIXEL(i, ilyo)
   ENDDO

   RETURN
   END
```

```
$TITLE: 'Filespec: d:\code\qmaphp\hp.for'
!---    Last edit: 08-Sep-1991 14:01, RB
!
! Output routines for HP-GL
!
!-----------------------------------------------------------------------

      SUBROUTINE HP_INIT (ifile)

      CHARACTER buffer*128, ofile*64, ifile*64, lj3file*64
      COMMON /hp/ ofile, lj3file

!     Prompt for output filespec
      ofile = ifile(:INDEX(ifile,'.',.TRUE.))//'PLT'
      CALL UPPER (ofile)
      WRITE(*,'(A,A,A,\)') ' Enter HP-GL plot output filespec (',
     +                     ofile(:LEN_TRIM(ofile)),')->'
      READ(*,'(A)') buffer
      IF (buffer(1:1) .NE. ' ') ofile = buffer

!     Build Laserjet III output filespec
      lj3file = ofile(:INDEX(ofile,'.',.TRUE.))//'LJ3'

!     Open HP-GL output file as unit 10
      OPEN (UNIT=10, FILE=ofile)

99999 RETURN
      END

!-----------------------------------------------------------------------

      SUBROUTINE HP_LT (x, y)

      REAL*4 x, y, x1, y1

      x1 = x+1.25
      y1 = y+.5

10001 FORMAT ('PD',F6.2,',',F6.2,';')

      WRITE(10,10001) x1, y1

      RETURN
      END

!-----------------------------------------------------------------------

      SUBROUTINE HP_MT (x, y)

      REAL*4 x, y, x1, y1

      x1 = x+1.25
      y1 = y+.5

10002 FORMAT ('PU',F6.2,',',F6.2,';')

      WRITE(10,10002) x1, y1

      RETURN
      END

!-----------------------------------------------------------------------

      SUBROUTINE HP_SY (st, an, x, y, ht)

      INTEGER*2 an, ln
      REAL*4 ht, x, y, PI, degrad, sina, cosa, x1, y1
      CHARACTER st*(*)
      CHARACTER buf*128

      x1 = x+1.25
      y1 = y+.5
```

```
      PI = 4.*ATAN(1.)
      degrad = PI/180.
      cosa = COS(REAL(an)*degrad)
      sina = SIN(REAL(an)*degrad)

10003 FORMAT ('PU',F6.2,',',F6.2,';',
     +        'SI',F4.2,',',F4.2,';',
     +        'DI',F4.2,',',F4.2,';',
     +        'LB',A,';')

      ln = LEN_TRIM(st)
      buf = st(:ln)//CHAR(3)
      ln = ln+1

      IF (ht .GT. 0) THEN
         WRITE(10,10003) x1, y1, ht*.8, ht, cosa, sina, buf(:ln)
      ELSE
         ht = ABS(ht)
         WRITE(10,10003) x1, y1-((ht*.8)*(REAL(ln)/2.)), ht*.8, ht,
     +                   cosa, sina, buf(:ln)
      ENDIF

      RETURN
      END

!-----------------------------------------------------------------------

      SUBROUTINE HP_TRI (ht, x, y)

      REAL*4 ht, x, y, x1, y1

      x1 = x+1.25
      y1 = y+.5

10004 FORMAT ('PU',F6.2,',',F6.2,';',
     +        'PD',F6.2,',',F6.2,',',
     +             F6.2,',',F6.2,',',
     +             F6.2,',',F6.2,';')

      WRITE(10,10004) x1-(ht/2.), y1,
     +                x1+(ht/2.), y1-(ht/2.),
     +                x1+(ht/2.), y1+(ht/2.),
     +                x1-(ht/2.), y1

      RETURN
      END

!-----------------------------------------------------------------------

      SUBROUTINE HP_CRS (ht, x, y)

      REAL*4 ht, x, y, x1, y1

      x1 = x+1.25
      y1 = y+.5

10005 FORMAT ('PU',F6.2,',',F6.2,';',
     +        'PD',F6.2,',',F6.2,';',
     +        'PU',F6.2,',',F6.2,';',
     +        'PD',F6.2,',',F6.2,';')

      WRITE(10,10005) x1-(ht/2.), y1,
     +                x1+(ht/2.), y1,
     +                x1, y1-(ht/2.),
     +                x1, y1+(ht/2.)

      RETURN
      END

!-----------------------------------------------------------------------
```

```
      SUBROUTINE HP_LL (uval, sym, hgt, lval, ang, x, y)

      REAL*4 hgt, x, y
      INTEGER*2 ang, ln
      CHARACTER uval*(*), sym*(*), lval*(*)
      CHARACTER buf*32


      ln = LEN_TRIM (lval)
      buf = lval(:ln)//" <"
      ln = LEN_TRIM (buf)

      CALL HP_SY (buf(:ln), ang, x, y-(REAL(ln)*.19)-.35, .4)

      CALL HP_SY (sym, ang, x, y-(hgt*.16), hgt)

      IF (uval(1:1) .NE. ' ') THEN
         ln = LEN_TRIM (uval)
         buf = '<= '//uval(:ln)
         ln = ln+3
         CALL HP_SY (buf(:ln), ang, x, y+.35, .4)
      ENDIF

      RETURN
      END
!-------------------------------------------------------------------------

      SUBROUTINE HP_CLOSE (rows, cols)

      INTEGER*2 rows, cols, i, j
      CHARACTER buf*256, ofile*64, lj3file*64
      COMMON /hp/ ofile, lj3file

10000 FORMAT (A1,'%0B',/,
     +         'SP 2;',
     +         'PW 0;',
     +         'PU203,508;',
     +         'DI0,1;',
     +         'SI;',
     +         'SL.15;',/,
     +         'LBSheet ',I1,' :',I1,' of',I2,A,
     +         'SL;')

!---  Registration mark
10001 FORMAT ('PU',I4,',',I4,';',
     +        'CI101;PR;PU-152,0;PD304, 0;PU-152,-152;PD0,304;PA;')

10002 FORMAT('IP 254,254,7874,9906;IW 254,254,7874,9906;')

10003 FORMAT('SC',F6.2,',',F6.2,',',F6.2,',',F6.2,';')

      WRITE (*,'(/,/)')

      OPEN (11,FILE=lj3file)

      DO i=1, rows
         DO j=1, cols
            WRITE (*,'(A,I2,A,I2)')
     +         '+Generating HPGL-2 output: ',
     +         i, ':', j
            WRITE (11, 10000) CHAR(27), i, j, rows*cols, CHAR(3)

            WRITE (11, 10001) 254, 254
            WRITE (11, 10001) 7874, 254
            WRITE (11, 10001) 7874, 9906
            WRITE (11, 10001) 254, 9906

            WRITE (11, 10002)
            WRITE (11, 10003) (i-1)*7.5, i*7.5, (j-1)*9.5, j*9.5

            REWIND (10)
```

```
            DO WHILE (.TRUE.)
               READ (10,FMT='(A)',END=20000) buf
               WRITE(11,'(A)') buf(:LEN_TRIM(buf))
            ENDDO

20000       WRITE(11,'(A,A)') CHAR(27), 'E'
         ENDDO
      ENDDO

      CLOSE(10)
      CLOSE(11)

      WRITE (*,'(A,A)') '+HPGL-2 output for Laserjet III in: ',
     +   lj3file(:LEN_TRIM(lj3file))//'                    '

      WRITE (*,'(/,A,A)') ' Raw HPGL output in: ',
     +   ofile(:LEN_TRIM(ofile))

      RETURN
      END
```

```
$TITLE: 'Filespec: d:\code\qmaphp\label.for'
!---    Last edit: 30-Jun-1991 20:09, RB
C---
C---  Label routines
C---
C-----------------------------------------------------------------------

      SUBROUTINE LABELX (xs, ys, ht, f1, f2)

      REAL*4 xs, ys, ht, f1, f2
      CHARACTER longitude*9

   10 FORMAT (I3,' ',I2.2,'.'I2.2)
      WRITE(longitude,10) INT(f1), INT(f2), INT((f2-INT(f2))*100)
      CALL SYMBOL (xs, ys, ht, longitude, 0., 9)

      CALL HP_SY (longitude, 0, xs, ys, ht*1.5)

      RETURN
      END

C-----------------------------------------------------------------------

      SUBROUTINE LABELY (xs, ys, ht, f1, f2)

      REAL*4 xs, ys, ht, f1, f2
      CHARACTER latitude*8

   10 FORMAT (I2,' ',I2.2,'.'I2.2)
      WRITE(latitude,10) INT(f1), INT(f2), INT((f2-INT(f2))*100)
      CALL SYMBOL (xs, ys, ht, latitude, 90., 8)

      CALL HP_SY (latitude, 90, xs, ys, ht*1.5)

      RETURN
      END
```

```
$TITLE: 'Filespec: d:\code\qmaphp\legend.for'
!---    Last edit: 30-Jun-1991 20:09, RB
!---
!---  Creates the map legend
!---
!-----------------------------------------------------------------------

      SUBROUTINE LEGEND (x,y,scale,subtitle,nc,c,s,sz,rowi,coli)

      IMPLICIT INTEGER*2 (i-n)

      REAL*4 x, y, scale, c(*), sz(*), rowi, coli
      INTEGER*2 nc, ilen
      CHARACTER subtitle*(*), s*1(*)

      CHARACTER buf1*64, buf2*64, uval*6, lval*6

!     Establish location
      xlen = 1.5
      DO i=1, nc
         xlen = xlen+sz(i)
      ENDDO
      ylen = LEN_TRIM(subtitle)*.35
      IF (ylen .LT. 2.) ylen = 2.
      CALL CURSOR (x, y, xlen, ylen, rowi, coli)

!     Place legend title
      ilen = LEN_TRIM(subtitle)
      CALL SYMBOL (x+.5, y, .35, subtitle, 90., -ilen)
      CALL HP_SY (subtitle(:ilen), 90, x+.5,
     +            y-((REAL(ilen)/2.)*.28), .6)

!     Legend lines
      x = x+1.
      DO i=1, nc
         WRITE(buf1,'(F5.2)') c(i)
         lval = buf1(LTRIM(buf1):)
         IF (c(i+1) .NE. 0.) THEN
            WRITE(buf1,'(F5.2)') c(i+1)
            uval = buf1(LTRIM(buf1):)
         ELSE
            uval = ' '
         ENDIF
         IF (sz(i) .GE. .2) THEN
            x = x+sz(i)
         ELSE
            x = x+.2
         ENDIF
         CALL SYMBOL (x, y-.75, .25, lval, 90., -5)
         CALL SYMBOL (x, y-(sz(i)/2.), sz(i), s(i), 90., -1)
         IF (uval(1:1) .NE. ' ')
     +      CALL SYMBOL (x, y+.75, .25, uval, 90., -5)
         CALL HP_LL (uval, s(i), sz(i), lval, 90, x, y)
      ENDDO

!     Place scale
!              123456789a1234
      buf1 = 'Scale = 1:'
      WRITE(buf2,'(F10.0)') scale
      buf1(11:) = buf2(LTRIM(buf2):INDEX(buf2,'.',.TRUE.)-1)
      ilen = LEN_TRIM(buf1)
      CALL SYMBOL (x+.5, y, .2, buf1, 90., -ilen)
      CALL HP_SY (buf1(:LEN_TRIM(buf1)), 90, x+.5,
     +            y-((REAL(ilen)/2.)*.19), .4)

      RETURN
      END
```

```
$TITLE: 'Filespec: d:\code\qmaphp\ssplot.for'
!---    Last edit: 30-Jun-1991 20:10, RB
C---
C---  Subroutine SSPLOT
C---
C---  Special symbol plot using USGS plotter
C---
C---  Arguments:
C---     ICODE
C---        1st digit
C---           =0 - New plot
C---           =1 - Same axis
C---        2nd digit
C---           =0 - Center symbol on tri
C---           =1 - Center symbol from keypunch
C---        3rd digit
C---           =0 - No axis
C---           =1 - X axis
C---           =2 - X axis and Y axis
C---           =3 - All four axes
C---
C---     XARRAY   - Array of X coordinates
C---     YARRAY   - Array of Y coordinates
C---     NPOINT   - Number of points
C---     XLEFT    - X value at left side of X axis
C---     XRIGHT   - X value at right side of X axis
C---     YLOWER   - Y value at lower end of Y axis
C---     YUPPER   - Y value at upper end of Y axis
C---     XLENG    - Length of X axis
C---     YLENG    - Length of Y axis
C---     SYM      - Symbol to be used
C---     HEIGHT   - The height of the symbol to be plotted
C---     SCALE    - Scale of map to be plotted
C---
C-----------------------------------------------------------------------

      SUBROUTINE SSPLOT (icode, xarray, yarray, npoint, xleft, xright,
     +                   ylower, yupper, xleng, yleng, sym, height,
     +                   scale)

      REAL*4 xarray(*), yarray(*), height(*)
      CHARACTER sym(*)*1

C     Initialization
      ic = icode/100
      jc = (icode/10)-(icode/100)*10
      kc = icode-ic*100-jc*10
      xl = xleng
      yl = yleng
      ds = 1.E05/scale

C     Test for ic
      IF (ic .NE. 0) GO TO 15

C     Draw axes
      IF (kc .LE. 0) GO TO 15
      CALL PLOT (xl, 0., 2)
      CALL HP_MT (0., 0.)
      CALL HP_LT  (xl, 0.)
      IF (kc-2) 15, 12, 11
   11 CALL PLOT (xl, yl, 2)
      CALL HP_LT  (xl, yl)
      CALL PLOT (0., yl, 2)
      CALL HP_LT (0., yl)
   12 CALL PLOT (0., 0., 2)
      CALL HP_LT (0., 0.)

   15 CALL PLOT (0., 0., 3)
      IF (npoint .LT. 1) RETURN

!     Check bounds
      DO 30 i=1, npoint
```

```
      k = i
      IF (xarray(i) .LT. xleft) GO TO 25
      IF (xarray(i) .GT. xright) GO TO 25
      IF (yarray(i) .LT. ylower) GO TO 25
      IF (yarray(i) .GT. yupper) GO TO 25

C     Center symbol
   18 x = (xarray(i)-xleft)*ds
      y = (yarray(i)-ylower)*ds
      CALL SYMBOL (x, y, height(k), sym(k), 90., -1)

      SELECT CASE (ICHAR(sym(k)))
      CASE (43)
         CALL HP_CRS (height(k), x, y)
      CASE (10)
         CALL HP_TRI (height(k), x, y)
      CASE DEFAULT
         CALL HP_SY (sym(k), 90, x+(height(k)*.5),
     +               y-(height(k)*.4), height(k))
      END SELECT
      GO TO 30

   25 WRITE(6,26) k
   26 FORMAT(' *** DATA NO.',I5,' IS OUTSIDE MAP BOUNDARIES ***')
   30 CONTINUE

      RETURN
      END
```

```
$TITLE: 'Filespec: d:\code\qmap\xycord.for'
!---    Last edit: 05-Apr-1991 17:23, RB
C---
C---     Subroutine XYCORD to compute the absolute coordinates PX and
C---  PY for N points whose latitude and longitude are PLAT and PLON
C---  with respect to the origin given by OLAT and OLON.
C---
C---     Latitudes and longitude must be given in minutes.
C---
C---     PX and PY are givin in 10**5 inches.
C---
C---     Calculations apply mainly to latitudes between 0ł and 70ł
C---
C-------------------------------------------------------------------------

      SUBROUTINE XYCORD (olat, olon, plat, plon, n, px, py)

      REAL mlat

      DIMENSION plat(*), plon(*), px(*), py(*)
      DIMENSION a(75), b(75)

      DATA a /1.855365, 1.855369, 1.855374, 1.855383, 1.855396,
     +        1.855414, 1.855434, 1.855458, 1.855487, 1.855520,
     1        1.855555, 1.855595, 1.855638, 1.855683, 1.855733,
     +        1.855786, 1.855842, 1.855902, 1.855966, 1.656031,
     2        1.856100, 1.856173, 1.856248, 1.856325, 1.856404,
     +        1.856488, 1.856573, 1.856661, 1.856750, 1.856843,
     3        1.856937, 1.857033, 1.857132, 1.857231, 1.857331,
     +        1.857435, 1.857538, 1.857643, 1.857750, 1.857858,
     4        1.857964, 1.858074, 1.858184, 1.858294, 1.858403,
     +        1.858512, 1.858623, 1.858734, 1.858842, 1.858951,
     5        1.859061, 1.859170, 1.859276, 1.859384, 1.859488,
     +        1.859592, 1.859695, 1.859798, 1.859896, 1.859995,
     6        1.860094, 1.860187, 1.860279, 1.860369, 1.860459,
     +        1.860544, 1.860627, 1.860709, 1.860787, 1.860861,
     7        1.860934, 0., 0., 0., 0./
      DATA b/ 1.842808, 1.842813, 1.842830, 1.842858, 1.842898,
     +        1.842950, 1.843011, 1.843085, 1.843170, 1.843265,
     1        1.843372, 1.843488, 1.843617, 1.843755, 1.843903,
     +        1.844062, 1.844230, 1.844408, 1.844595, 1.844792,
     2        1.844998, 1.845213, 1.845437, 1.845668, 1.845907,
     +        1.846153, 1.846408, 1.846670, 1.846938, 1.847213,
     3        1.847495, 1.847781, 1.848073, 1.848372, 1.848673,
     +        1.848980, 1.849290, 1.849605, 1.849922, 1.850242,
     4        1.850565, 1.850890, 1.851217, 1.851543, 1.851873,
     +        1.852202, 1.852531, 1.852860, 1.853188, 1.853515,
     5        1.853842, 1.854165, 1.854487, 1.854805, 1.855122,
     +        1.855433, 1.855742, 1.856045, 1.856345, 1.856640,
     6        1.856928, 1.857212, 1.857490, 1.857762, 1.858025,
     +        1.858283, 1.858553, 1.858775, 1.859008, 1.859235,
     7        1.859452, 0., 0., 0., 0./

      DO 10 i=1, n
      dlat = plat(i)-olat
      dlon = plon(i)-olon
      mlat = (plat(i)+olat)/120.
      jlat = mlat
      k = jlat+1
      IF (k .LT. 1) k = 1
      IF (k .GT. 71) k = 71
      flat = mlat-jlat
      aa = a(k)+flat*(a(k+1)-a(k))
      bb = b(k)+flat*(b(k+1)-b(k))
      x = aa*COS(mlat*.0174533)*dlon
      y = bb*dlat
      px(i) = -x*0.393696
      py(i) = -y*0.393696
   10 CONTINUE

      RETURN
      END
```

Sun 16-Aug-1992 15:48, RB

>>> Notes on QMAPPS <<<

QMAPPS was written using Microsoft FORTRAN 5.00.

Libraries required:

UPLOT.LIB - Uplot graphics interface library 1.10.
Available from:
Small Systems Support
2 Boston Harbor Place
Big Water, UT 84741-0205
(801) 675-5827 Voice
(801) 675-3730 FAX

GEOGRAF.LIB - GeoGraf graphics library version 4.00 for MS FORTRAN.
Available from:
GeoComp Corp.
66 Commonwealth Ave.
Concord, MA 01742
(800) 822-2669

The following files are included

QMAP	FOR	10781	09-15-91	6:57p
CONVERT	FOR	1235	04-15-91	8:01p
CURSOR	FOR	5006	04-13-91	9:37p
LABEL	FOR	956	04-08-91	10:59p
LEGEND	FOR	1983	09-15-91	7:00p
MAKEQMAP	FOR	1645	09-15-91	4:11p
PS	FOR	6843	09-08-91	4:24p
SSPLOT	FOR	3106	09-15-91	2:03p
XYCORD	FOR	3083	04-05-91	6:38p
PLOTLIB	INC	1739	08-04-91	2:12p
PS_FORMS	INC	914	04-15-91	8:22p
CONVERT	MAK	736	04-12-91	3:30p
MAKEQMAP	MAK	544	09-15-91	3:49p
QMAP	MAK	1083	09-15-91	7:02p
TEST	INP	1255	09-15-91	6:38p
QMAP_HDR	PS	3794	04-13-91	5:56p

```
$TITLE: 'Filespec: d:\code\qmapps\qmap.for'
!---    Last edit: 08-Sep-1991 16:18, RB
C---
C---  QMAP1 was originally written in July 1969 by W. H. K. Lee
C---
C---     QMAP1 was written as a general purpose program to plot a
C---  maximum of 2500 earthquakes on a map.  Each earthquake was
C---  represented by a symbol of a user specified size.
C---
C---     Original code ported to Microsoft FORTRAN 5.00 in March 1991
C---  Robert Banfill, Small Systems Support, Box 410205, Big Water, UT,
C---  84741-0205, (801) 675-5827.
C---
C-------------------------------------------------------------------------

      PROGRAM QMAP

!     Pmax = max events, Smax = max # of stn's,
!     Cmax = max classes, Gmax= max grid points
      PARAMETER (PMAX=2000, SMAX=256, CMAX=20, GMAX=200)

      REAL*4 glat(GMAX), glon(GMAX), gx(GMAX), gy(GMAX)
      REAL*4 clat(2), clon(2), cx(2), cy(2)
      REAL*4 plat(PMAX), plon(PMAX), px(PMAX), py(PMAX)
      REAL*4 slat(SMAX), slon(SMAX), sx(SMAX), sy(SMAX)
      REAL*4 sz(CMAX), c(CMAX), ht(PMAX)
      REAL*4 llat, llon, lx, ly, x, y, ds

      CHARACTER sym(PMAX)*1
      CHARACTER ista(SMAX)*4
      CHARACTER frmt*80, card*80, s(CMAX)*1
      CHARACTER plus*1, tri*1
      CHARACTER driver*64, port*64, buffer*128, ifile*64, lfile*64
      CHARACTER title*80, subtitle*64

      INTEGER*2 rows, cols
      INTEGER*4 pen

      WRITE(*,'(/,A,/)') ' PC-QMAP PostScript Version 1.02'

      plus = '+'
      tri = CHAR(10)

      nmax = PMAX

!     Prompt for input filespec
      ifile = 'QMAP.INP'
      WRITE(*,'(A,\)') ' Enter Input filespec (QMAP.INP)->'
      READ(*,'(A)') buffer
      IF (buffer(1:1) .NE. ' ') ifile = buffer

!     Open input file as unit 5
      OPEN (UNIT=5, FILE=ifile, STATUS='old', ERR=50000)

!     Prompt for input filespec
      lfile = ifile(:INDEX(ifile,'.',.TRUE.))//'LOG'
      CALL UPPER (lfile)
      WRITE(*,'(A,A,A,\)') ' Enter Log filespec (',
     +                     lfile(:LEN_TRIM(lfile)),')->'
      READ(*,'(A)') buffer
      IF (buffer(1:1) .NE. ' ') lfile = buffer

!     Redirect unit 6 (all terminal output) to a log file
      OPEN (UNIT=6, FILE=lfile)

    1 WRITE(6,5) 'PC-QMAP PostScript Version 1.02 Log file: ',
     +           lfile(:LEN_TRIM(lfile))
    5 FORMAT(A,A,/)

      CALL PS_INIT (ifile)

!     Initialize graphics routines
```

```
      CALL GETARG (0, driver, ierr)
      driver(INDEX(driver, '\', .TRUE.)+1:) = 'SCREEN.DRV$'
      ierr = LDDRV (driver)
      IF (ierr .LT. 0) THEN
         WRITE(*,'(/,A,A)') ' ERROR: Unable load driver: ',
     +                      driver(:LEN_TRIM(driver))
         STOP' '
      ENDIF
      port = 'CON:'
      ierr = INIPLT (port, 0)
      IF (ierr .LT. 0) THEN
         WRITE(*,'(/,A,A,I5)') ' ERROR: Unable to initialize: ',
     +                         port(:LEN_TRIM(port)),ierr
         STOP' '
      ENDIF

!     Read in control section
20000 READ(5,'(A)',END=100) card
      CALL UPPER(card)
      IF (card(1:3) .EQ. '$CO') THEN
         READ(5,'(I2,1X,F5.2,1X,I3,1X,F5.2)') lat, xlat, lon, xlon
         READ(5,'(F5.0)') ep
         READ(5,'(F5.0)') gi
         READ(5,'(F10.0)') scale
         READ(5,'(F5.2)') ssz
      ELSE
         GOTO 20000
      ENDIF
      WRITE(6,'(A)') '*** Control information ***'
      WRITE(6,'(A,I2,1X,F5.2,1X,I3,1X,F5.2)') 'Center: ',
     +      lat, xlat, lon, xlon
      WRITE(6,'(A,F5.0)') 'Extent: ', ep
      WRITE(6,'(A,F5.0)') 'Grid interval: ', gi
      WRITE(6,'(A,F10.0)') 'Scale: ', scale
      WRITE(6,'(A,F5.2)') 'Station size: ', ssz

      olat = 60.*REAL(lat)+xlat
      olon = 60.*REAL(lon)+xlon

C     Generate plot bounds
      clat(1) = olat+ep
      clon(1) = olon+ep
      clat(2) = olat-ep
      clon(2) = olon-ep
      CALL XYCORD (olat, olon, clat, clon, 2, cx, cy)
      xright = cy(2)
      xleft = cy(1)
      yupper = cx(2)
      ylower = cx(1)
      xleng = (xright-xleft)*1.E05/scale
      yleng = (yupper-ylower)*1.E05/scale
      xl = xleng
      yl = yleng

!     Figure out how many pages needed for output
      rows = INT(xleng/7.5)+1
      cols = INT(yleng/10.)+1
      rowi = REAL(rows)*7.5
      coli = REAL(cols)*10.

!     Let's plot...
      CALL WINDOW (0., 0., rowi+.1, coli+.1)
      CALL PLOTS (0)

!     Plot paper lines
      CALL NEWPEN (8)
      DO r=0., rowi, 7.5
         CALL PLOT (r, 0., 3)
         CALL PLOT (r, coli, 2)
      ENDDO
      DO r=0., coli, 10.
         CALL PLOT (0., r, 3)
```

```
         CALL PLOT (rowi, r, 2)
      ENDDO

!     Set origin
      CALL PLOT (1., 0.5, -3)
      CALL PS_TR (1., 0.5)

!     Read in legend section
20010 READ(5,'(A)') card
      CALL UPPER(card)
      IF (card(1:3) .EQ. '$LE') THEN
         READ(5,'(A)') title
         READ(5,'(A)') subtitle
         DO i=1, 20
30000       FORMAT (A,T1,F6.2,1X,A1,1X,F6.3)
            READ(5,30000) card, c(i), s(i), sz(i)
            IF (card(1:6) .EQ. '      ') THEN
               nc = i-1
               c(i) = 999.
               EXIT
            ENDIF
         ENDDO
      ELSE
         GOTO 20010
      ENDIF
      WRITE(6,'(/,A)') '*** Legend information ***'
      WRITE(6,'(A,A)') 'Title: ',title(:LEN_TRIM(title))
      WRITE(6,'(A,A)') 'Subtitle: ',subtitle(:LEN_TRIM(subtitle))
      DO i=1, nc
30001    FORMAT (A,I2,1X,F6.2,1X,A1,1X,F6.3)
         WRITE(6,30001) 'Class', i, c(i), s(i), sz(i)
      ENDDO

      zsz = .5

!     Plot the title and legend
      CALL NEWPEN (7)
      CALL SYMBOL (0.,0.,0.7,title,90.,LEN_TRIM(title))
      CALL PS_SY (title(:LEN_TRIM(title)), 90, 0., 0., 0.85)

C     Set origin for subsequent plot
      CALL PLOT (.25, 0., -3)
      CALL PS_TR (.25, 0.)

C     Generate grid points
      ng = 2*ep/gi-1
      IF (ng .LE. 0) GO TO 22
      IF (ng .GT. 10) ng = 10
      ngp = ng*ng
      DO 20 i=1, ng
      DO 20 j=1, ng
      k = (i-1)*ng+j
      glat(k) = olat+ep-i*gi
      glon(k) = olon+ep-j*gi
      sym(k) = plus
      ht(k) = 0.5
   20 CONTINUE
      CALL XYCORD (olat, olon, glat, glon, ngp, gy, gx)

C     Plot grid points and boundaries
      CALL SSPLOT (014, gx, gy, ngp, xleft, xright, ylower, yupper,
     +             xleng, yleng, sym, ht, scale, ista)
      GO TO 225

C     Plot boundaries
   22 CALL SSPLOT (014, gx, gy, 0, xleft, xright, ylower, yupper,
     +             xleng, yleng, sym, ht, scale, ista)

C     Labeling lat and long of the boundaries
  225 i = clat(2)/60.
      a1 = i
      a2 = clat(2)-i*60.
```

```
      j = clon(1)/60.
      b1 = j
      b2 = clon(1)-j*60.
      i = clat(1)/60.
      c1 = i
      c2 = clat(1)-i*60.
      j = clon(2)/60.
      d1 = j
      d2 = clon(2)-j*60.
      CALL NEWPEN (7)
      CALL LABELX (0.5, 0.125, 0.25, b1, b2)
      CALL LABELX (xl-2.25, 0.125, 0.25, b1, b2)
      CALL LABELY (xl-0.125, 0.5, 0.25, a1, a2)
      CALL LABELY (xl-0.125, yl-2.0, 0.25, a1, a2)
      CALL LABELX (xl-2.25, yl-0.375, 0.25, d1, d2)
      CALL LABELX (0.5, yl-0.375, 0.25, d1, d2)
      CALL LABELY (0.375, yl-2.0, 0.25, c1, c2)
      CALL LABELY (0.375, 0.5, 0.25, c1, c2)

C     Read in station data and plot stations
      WRITE(6,'(/,A)') '*** Station data ***'
20020 READ(5,'(A)') card
      CALL UPPER(card)
      IF (card(1:3) .EQ. '$ST') THEN
         READ(5,'(A)') frmt
         DO i=1, SMAX
            READ(5,frmt) ista(i), lat, xlat, lon, xlon
            IF (ista(i)(1:4) .EQ. '    ') EXIT
            WRITE(6,31) i, ista(i), lat, xlat, lon, xlon
   31       FORMAT(I3,1X,A4,1X,I2,'-',F5.2,1X,1X,I3,'-',F5.2)
            slat(i) = 60.*REAL(lat)+xlat
            slon(i) = 60.*REAL(lon)+xlon
            sym(i) = tri
            ht(i) = ssz
         ENDDO
      ELSE
         GOTO 20020
      ENDIF
      nsp = i-1
      IF (nsp .LE. 0) GO TO 345
      CALL XYCORD (olat, olon, slat, slon, nsp, sy, sx)
      CALL NEWPEN (14)
      CALL SSPLOT (104, sx, sy, nsp, xleft, xright, ylower, yupper,
     +             xleng, yleng, sym, ht, scale, ista)
  345 CONTINUE

C     Read in data format and data
      WRITE(6,'(/,A)') '*** Point data ***'
20030 READ(5,'(A)') card
      CALL UPPER(card)
      IF (card(1:3) .NE. '$PO') THEN
         GOTO 20030
      ELSE
         READ(5,'(A)') frmt
         n = 1
   40    READ(5,frmt) card, lat, xlat, lon, xlon, z
         IF (card(1:6) .EQ. '      ') GO TO 60
         plat(n) = 60.*REAL(lat)+xlat
         plon(n) = 60.*REAL(lon)+xlon
C        Find symbol for Z
   41    DO 50 i=1, nc
         IF (z .GT. c(i+1)) GO TO 45
         sym(n) = s(i)
         ht(n) = sz(i)
         GO TO 55
   45    CONTINUE
   50    CONTINUE
         sym(n) = s(nc)
         ht(n) = sz(nc)
   55    WRITE(6,56) n, card(:LEN_TRIM(card))
   56    FORMAT(I5,' - ',A)
  565    n = n+1
```

```
         IF (n .LE. nmax) GO TO 40
         WRITE(6,'(A)') '*** ERROR: To many data points, ',
     +                  'remaining points ignored ***'
      ENDIF

C     Plot data
   60 npoint = n-1
      IF (npoint .LE. 0) GO TO 80
      CALL XYCORD (olat, olon, plat, plon, npoint, py, px)
      CALL NEWPEN (15)
      CALL SSPLOT (114, px, py, npoint, xleft, xright, ylower, yupper,
     +             xleng, yleng, sym, ht, scale, ista)

!     Read in line data
      CALL NEWPEN (6)
      ds = 1.E05/scale
20040 READ(5,'(A)',END=80) card
      CALL UPPER(card)
      IF (card(1:3) .NE. '$LI') THEN
         GOTO 20040
      ELSE
         READ(5,'(A)',END=80) frmt
         pen = 3
         DO WHILE (.TRUE.)
            READ(5,'(A)',END=80) card
            CALL UPPER(card)
            IF (card(1:6) .EQ. '      ') THEN
               pen = 3
               CYCLE
            ELSEIF (card(1:4) .EQ. '$END') THEN
               EXIT
            ENDIF
            READ(card,frmt) lat, xlat, lon, xlon
            llat = 60.*REAL(lat)+xlat
            llon = 60.*REAL(lon)+xlon
            CALL XYCORD( olat, olon, llat, llon, 1, ly, lx)
            x = (lx-xleft)*ds
            y = (ly-ylower)*ds
            CALL PLOT (x, y, pen)
            IF (pen .EQ. 2) THEN
               CALL PS_LT (x, y)
            ELSE
               CALL PS_MT (x, y)
            ENDIF
            pen = 2
         ENDDO
      ENDIF

   80 CALL NEWPEN (3)
      CALL LEGEND (.75, 2., scale, subtitle, nc, c, s, sz, rowi, coli)

      ikey = KCHARIN ()

  100 CLOSE(5)
      CLOSE(6)
      CALL PLOT (0.,0.,999)
      ierr = GCLOSE ()
      CALL PS_CLOSE (rows, cols)
      GOTO 99999

50000 WRITE(*,'(/,A,A)') ' ERROR: Cannot open ',
     +                   ifile(:LEN_TRIM(ifile))

99999 STOP' '
      END
```

```
$TITLE: 'Filespec: d:\code\qmap\convert.for'
!---    Last edit: 15-Apr-1991 18:02, RB

      PROGRAM CONVERT

      CHARACTER buffer*256
      INTEGER*2 lat, lon
      REAL*4 xlat, xlon

      OPEN(10, FILE='sfbay.geo')
      OPEN(11, FILE='sfbay.dat')

      DO WHILE (.TRUE.)
         READ(10, FMT='(A)', END=999) buffer
         DO i=1,6*18, 18
            READ(buffer(i+1:i+8), '(F8.3)') xlat
            READ(buffer(i+10:i+18), '(F8.3)') xlon
            IF (xlat .EQ. 0. .AND. xlon .EQ. 0.) THEN
               WRITE(11,'(A)')
            ELSE
               lat = INT(xlat)
               xlat = (xlat-REAL(lat))*60
               lon = INT(xlon)
               xlon = (xlon-REAL(lon))*60
               WRITE(11,'(I2,1X,F5.2,1X,I3,1X,F5.2)') lat,
     +            xlat, lon, xlon
               WRITE(*,'(1X,I2,1X,F5.2,1X,I3,1X,F5.2)') lat,
     +            xlat, lon, xlon
            ENDIF
         ENDDO
      ENDDO

!  123456789a1234567
!123456789a123456789b123456789c123456789d123456789
!12345678912345678912345678912345678912345678912345
!   0.0000   0.0000  38.9669-123.6638  38.9676-123.6671  38.9745-123.6755  38.9802-123.6807  3

  999 CLOSE(10)
      CLOSE(11)

      END
```

```
$TITLE: 'Filespec: d:\code\qmap\cursor.for'
!---    Last edit: 13-Apr-1991 19:44, RB
!---
!---  Put up a box cursor, return x y of top middle
!---
!-----------------------------------------------------------------------

      SUBROUTINE CURSOR (x, y, xlen, ylen, uxmax, uymax)

      IMPLICIT INTEGER*2 (i-n)

      COMMON /cur/ iux, iuy, ilx, ily, iuxo, iuyo, ilxo, ilyo, sav
      INTEGER*2 iux, iuy, ilx, ily, iuxo, iuyo, ilxo, ilyo
      INTEGER*1 sav(84)

      REAL*4 x, y, xlen, ylen
      INTEGER*1 oldcolor

      INTEGER*2 SFCOL

      INCLUDE 'driver.inc'

      x = x+1.25
      y = y+.5

      xcon = xmax/uxmax
      ycon = ymax/uymax

      iux = INT2((x+xlen)*xcon)
      iuy = INT2((y+(ylen/2.))*ycon)
      ilx = INT2(x*xcon)
      ily = INT2((y-(ylen/2.))*ycon)

      oldcolor = INT1(currfcolor)

10000 CALL CURDRAW ()

!---  Get a keystroke
20000 nkey = 0
      nkey = KCHARIN()

!---  See if shifted nav key, set step size
      IF (nkey .GE. 49 .AND. nkey .LE. 57) THEN
        istep = 1
      ELSE
        istep = 10
      ENDIF

      SELECT CASE (nkey)
         CASE (19712,54) !Right
            IF (iux+istep .GE. xmax-1) GOTO 20000
            iux = iux + istep
            ilx = ilx + istep
         CASE (19200,52) !Left
            IF (ilx-istep .LT. 0) GOTO 20000
            ilx = ilx - istep
            iux = iux - istep
         CASE (18432,56) !Up
            IF (iuy+istep .GE. ymax-1) GOTO 20000
            iuy = iuy + istep
            ily = ily + istep
         CASE (20480,50) !Down
            IF (ily-istep .LT. 0) GOTO 20000
            ily = ily - istep
            iuy = iuy - istep
         CASE DEFAULT
            i2key = nkey
      END SELECT

!---  Restore the screen
30000 CALL CURRESTORE ()
```

```
!---  If not an exit key, go agian
      IF (i2key .NE. nkey) GOTO 10000

!---  Translate coords, restore current foreground color
90000 x = REAL(ilx)/xcon-1.25
      y = REAL(ily+((iuy-ily)/2))/ycon-.5
      ierr = SFCOL(oldcolor)

      RETURN
      END


!-----------------------------------------------------------------------

      SUBROUTINE CURDRAW ()

      IMPLICIT INTEGER*2 (i-n)

      COMMON /cur/ iux, iuy, ilx, ily, iuxo, iuyo, ilxo, ilyo, sav
      INTEGER*2 iux, iuy, ilx, ily, iuxo, iuyo, ilxo, ilyo
      INTEGER*1 sav(84)

      INTEGER*2 RPIXEL, WPIXEL, SFCOL

      INCLUDE 'driver.inc'

      j = 0
      ierr = SFCOL(INT1(nfcolors-1))
      iuxo = iux
      iuyo = iuy
      ilxo = ilx
      ilyo = ily

      DO i = ily, ily+10
         j = j+1
         sav(j) = RPIXEL(ilx, i)
         ierr = WPIXEL(ilx, i)
      ENDDO
      DO i = ilx+1, ilx+10
         j = j+1
         sav(j) = RPIXEL(i, ily)
         ierr = WPIXEL(i, ily)
      ENDDO
      DO i = iuy, iuy-10, -1
         j = j+1
         sav(j) = RPIXEL(iux, i)
         ierr = WPIXEL(iux, i)
      ENDDO
      DO i = iux-1, iux-10, -1
         j = j+1
         sav(j) = RPIXEL(i, iuy)
         ierr = WPIXEL(i, iuy)
      ENDDO

      DO i = ily, ily+10
         j = j+1
         sav(j) = RPIXEL(iux, i)
         ierr = WPIXEL(iux, i)
      ENDDO
      DO i = ilx+1, ilx+10
         j = j+1
         sav(j) = RPIXEL(i, iuy)
         ierr = WPIXEL(i, iuy)
      ENDDO
      DO i = iuy, iuy-10, -1
         j = j+1
         sav(j) = RPIXEL(ilx, i)
         ierr = WPIXEL(ilx, i)
      ENDDO
      DO i = iux-1, iux-10, -1
         j = j+1
         sav(j) = RPIXEL(i, ily)
```

```
      ierr = WPIXEL(i, ily)
   ENDDO

   RETURN
   END

!-------------------------------------------------------------------------

   SUBROUTINE CURRESTORE ()

   IMPLICIT INTEGER*2 (i-n)

   COMMON /cur/ iux, iuy, ilx, ily, iuxo, iuyo, ilxo, ilyo, sav
   INTEGER*2 iux, iuy, ilx, ily, iuxo, iuyo, ilxo, ilyo
   INTEGER*1 sav(84)

   INTEGER*2 WPIXEL, SFCOL

   j = 0

   DO i = ilyo, ilyo+10
      j = j+1
      ierr = SFCOL(sav(j))
      ierr = WPIXEL(ilxo, i)
   ENDDO
   DO i = ilxo+1, ilxo+10
      j = j+1
      ierr = SFCOL(sav(j))
      ierr = WPIXEL(i, ilyo)
   ENDDO
   DO i = iuyo, iuyo-10, -1
      j = j+1
      ierr = SFCOL(sav(j))
      ierr = WPIXEL(iuxo, i)
   ENDDO
   DO i = iuxo-1, iuxo-10, -1
      j = j+1
      ierr = SFCOL(sav(j))
      ierr = WPIXEL(i, iuyo)
   ENDDO

   DO i = ilyo, ilyo+10
      j = j+1
      ierr = SFCOL(sav(j))
      ierr = WPIXEL(iuxo, i)
   ENDDO
   DO i = ilxo+1, ilxo+10
      j = j+1
      ierr = SFCOL(sav(j))
      ierr = WPIXEL(i, iuyo)
   ENDDO
   DO i = iuyo, iuyo-10, -1
      j = j+1
      ierr = SFCOL(sav(j))
      ierr = WPIXEL(ilxo, i)
   ENDDO
   DO i = iuxo-1, iuxo-10, -1
      j = j+1
      ierr = SFCOL(sav(j))
      ierr = WPIXEL(i, ilyo)
   ENDDO

   RETURN
   END
```

```
$TITLE: 'Filespec: d:\code\qmap\label.for'
!---    Last edit: 05-Apr-1991 17:23, RB
C---
C---  Label routines
C---
C-----------------------------------------------------------------------

      SUBROUTINE LABELX (xs, ys, ht, f1, f2)

      REAL*4 xs, ys, ht, f1, f2
      CHARACTER longitude*9

   10 FORMAT (I3,' ',I2.2,'.'I2.2)
      WRITE(longitude,10) INT(f1), INT(f2), INT((f2-INT(f2))*100)
      CALL SYMBOL (xs, ys, ht, longitude, 0., 9)

      CALL PS_SY (longitude, 0, xs, ys, ht*1.3)

      RETURN
      END

C-----------------------------------------------------------------------

      SUBROUTINE LABELY (xs, ys, ht, f1, f2)

      REAL*4 xs, ys, ht, f1, f2
      CHARACTER latitude*8

   10 FORMAT (I2,' ',I2.2,'.'I2.2)
      WRITE(latitude,10) INT(f1), INT(f2), INT((f2-INT(f2))*100)
      CALL SYMBOL (xs, ys, ht, latitude, 90., 8)

      CALL PS_SY (latitude, 90, xs, ys, ht*1.3)

      RETURN
      END
```

```
$TITLE: 'Filespec: d:\code\qmap\legend.for'
!---    Last edit: 13-Apr-1991 18:53, RB
!---
!---  Creates the map legend
!---
!-----------------------------------------------------------------------

      SUBROUTINE LEGEND (x,y,scale,subtitle,nc,c,s,sz,rowi,coli)

      IMPLICIT INTEGER*2 (i-n)

      REAL*4 x, y, scale, c(*), sz(*), rowi, coli
      INTEGER*2 nc
      CHARACTER subtitle*(*), s*1(*)

      CHARACTER buf1*64, buf2*64, uval*6, lval*6

!     Establish location
      xlen = 1.5
      DO i=1, nc
         IF (sz(i)*1.25 .GT. .3) THEN
            xlen = xlen+(sz(i)*1.25)
         ELSE
            xlen = xlen+.3
         ENDIF
      ENDDO
      ylen = LEN_TRIM(subtitle)*.35
      IF (ylen .LT. 2.) ylen = 2.
      CALL CURSOR (x, y, xlen, ylen, rowi, coli)

!     Place legend title
      CALL SYMBOL (x+.5, y, .35, subtitle, 90., -LEN_TRIM(subtitle))
      CALL PS_SY (subtitle(:LEN_TRIM(subtitle)), 90, x+.5, y, -.5)

!     Legend lines
      x = x+1.
      DO i=1, nc
         WRITE(buf1,'(F5.2)') c(i)
         lval = buf1(LTRIM(buf1):)
         IF (c(i+1) .NE. 999.) THEN
            WRITE(buf1,'(F5.2)') c(i+1)
            uval = buf1(LTRIM(buf1):)
         ELSE
            uval = ' '
         ENDIF
         IF (sz(i)*1.25 .GT. .3) THEN
            x = x+(sz(i)*1.25)
         ELSE
            x = x+.3
         ENDIF
         CALL SYMBOL (x, y-.75, .25, lval, 90., -5)
         CALL SYMBOL (x, y, sz(i), s(i), 90., -1)
         IF (uval(1:1) .NE. ' ')
     +      CALL SYMBOL (x, y+.75, .25, uval, 90., -5)
         CALL PS_LL (uval, s(i), sz(i)*1.5, lval, 90, x, y)
      ENDDO

!     Place scale
!              123456789a1234
      buf1 = 'Scale = 1:'
      WRITE(buf2,'(F10.0)') scale
      buf1(11:) = buf2(LTRIM(buf2):INDEX(buf2,'.',.TRUE.)-1)
      CALL SYMBOL (x+.5, y, .2, buf1, 90., -LEN_TRIM(buf1))
      CALL PS_SY (buf1(:LEN_TRIM(buf1)), 90, x+.5, y, -.3)

      RETURN
      END
```

```
$TITLE: 'Filespec: d:\code\qmap\ps.for'
!---    Last edit: 08-Apr-1991 22:59, RB
!
! Output routines for PostScript
!
!-----------------------------------------------------------------------

      SUBROUTINE PS_INIT (ifile)

      CHARACTER buffer*128, ofile*64, hfile*64, ifile*64

      COMMON /output/ ofile

      INCLUDE 'ps_forms.inc'

      CALL GETARG (0, hfile, ierr)
      hfile(INDEX(hfile, '\', .TRUE.)+1:) = 'QMAP_HDR.PS'

!     Open Postscript header file
      OPEN (UNIT=11, FILE=hfile, STATUS='OLD',
     +      MODE='READ', ERR=50001)

!     Prompt for output filespec
      ofile = ifile(:INDEX(ifile,'.',.TRUE.))//'PS'
      CALL UPPER (ofile)
      WRITE(*,'(A,A,A,\)') ' Enter PostScript output filespec (',
     +                     ofile(:LEN_TRIM(ofile)),')->'
      READ(*,'(A)') buffer
      IF (buffer(1:1) .NE. ' ') ofile = buffer

!     Open PostScript output file as unit 10
      OPEN (UNIT=10, FILE=ofile)

!     Write out PostScript header
20000 READ (11,FMT='(A)',END=20001) buffer
      IF (LEN_TRIM(buffer) .NE. 0) THEN
         WRITE(10,'(A)') buffer(:LEN_TRIM(buffer))
      ELSE
         WRITE(10,'(A)')
      ENDIF
      GOTO 20000
20001 CLOSE (11)

      path = .FALSE.
      plen = 0
      lx = 99999.
      SEC_MAX = 50
      sec = 1
      slen = 0

      WRITE(10,10010) sec

      GOTO 99999

50001 WRITE(*,'(/,A,A)') ' ERROR: Cannot open ',
     +                 hfile(:LEN_TRIM(hfile))
      STOP' '

99999 RETURN
      END

!-----------------------------------------------------------------------

      SUBROUTINE PS_TR (x, y)

      REAL*4 x, y

      INCLUDE 'ps_forms.inc'

      IF (path) THEN
         WRITE(10,'(A)') 'stroke'
         lx = 99999.
```

```
         path = .FALSE.
         plen = 0
      ENDIF

      IF (slen .GT. SEC_MAX) THEN
         WRITE(10,10011)
         sec = sec+1
         WRITE(10,10010) sec
         slen = 0
      ENDIF

      WRITE(10,10000) x, y
      slen = slen+1

      RETURN
      END

!-----------------------------------------------------------------------

      SUBROUTINE PS_LT (x, y)

      REAL*4 x, y

      INCLUDE 'ps_forms.inc'

      IF (slen .GE. SEC_MAX) THEN
         IF (path) THEN
            WRITE(10,'(A)') 'stroke'
            path = .FALSE.
         ENDIF
         WRITE(10,10011)
         sec = sec+1
         WRITE(10,10010) sec
         slen = 0
      ENDIF

      IF (.NOT. path) THEN
         WRITE(10,'(A)') 'newpath'
         path = .TRUE.
         plen = 0
         IF (lx .NE. 99999.) THEN
            WRITE(10,10002) lx, ly
         ELSE
            WRITE(10,10002) 0., 0.
         ENDIF
         slen = slen+1
      ENDIF

      WRITE(10,10001) x, y
      lx = x
      ly = y
      plen = plen+1
      slen = slen+1

      RETURN
      END

!-----------------------------------------------------------------------

      SUBROUTINE PS_MT (x, y)

      REAL*4 x, y

      INCLUDE 'ps_forms.inc'

      IF (slen .GE. SEC_MAX) THEN
         IF (path) THEN
            WRITE(10,'(A)') 'stroke'
            path = .FALSE.
         ENDIF
         WRITE(10,10011)
         sec = sec+1
```

```
      WRITE(10,10010) sec
      slen = 0
   ENDIF

   IF (.NOT. path) THEN
      WRITE(10,'(A)') 'newpath'
      path = .TRUE.
      plen = 0
   ENDIF

   WRITE(10,10002) x, y
   lx = x
   ly = y
   slen = slen+1

   RETURN
   END

!-----------------------------------------------------------------------

   SUBROUTINE PS_SY (st, an, x, y, ht)

   INTEGER*2 an
   REAL*4 ht, x, y
   CHARACTER st*(*)

   INCLUDE 'ps_forms.inc'

   IF (path) THEN
      WRITE(10,'(A)') 'stroke'
      lx = 99999.
      path = .FALSE.
      plen = 0
   ENDIF

   IF (slen .GT. SEC_MAX) THEN
      WRITE(10,10011)
      sec = sec+1
      WRITE(10,10010) sec
      slen = 0
   ENDIF

   IF (ht .GT. 0) THEN
      WRITE(10,10003) st, an, x, y, ht
   ELSE
      WRITE(10,10007) st, an, x, y, ABS(ht)
   ENDIF
   slen = slen+1

   RETURN
   END

!-----------------------------------------------------------------------

   SUBROUTINE PS_TRI (ht, x, y)

   REAL*4 ht, x, y

   INCLUDE 'ps_forms.inc'

   IF (path) THEN
      WRITE(10,'(A)') 'stroke'
      lx = 99999.
      path = .FALSE.
      plen = 0
   ENDIF

   IF (slen .GT. SEC_MAX) THEN
      WRITE(10,10011)
      sec = sec+1
      WRITE(10,10010) sec
      slen = 0
```

```
      ENDIF

      WRITE(10,10004) ht, x, y
      slen = slen+1

      RETURN
      END

!-----------------------------------------------------------------------

      SUBROUTINE PS_CRS (ht, x, y)

      REAL*4 ht, x, y

      INCLUDE 'ps_forms.inc'

      IF (path) THEN
         WRITE(10,'(A)') 'stroke'
         lx = 99999.
         path = .FALSE.
         plen = 0
      ENDIF

      IF (slen .GT. SEC_MAX) THEN
         WRITE(10,10011)
         sec = sec+1
         WRITE(10,10010) sec
         slen = 0
      ENDIF

      WRITE(10,10005) ht, x, y
      slen = slen+1

      RETURN
      END

!-----------------------------------------------------------------------

      SUBROUTINE PS_LL (uval, sym, hgt, lval, ang, x, y)

      REAL*4 hgt, x, y
      INTEGER*2 ang
      CHARACTER uval*(*), sym*(*), lval*(*)

      INCLUDE 'ps_forms.inc'

      IF (path) THEN
         WRITE(10,'(A)') 'stroke'
         lx = 99999.
         path = .FALSE.
         plen = 0
      ENDIF

      IF (slen .GT. SEC_MAX) THEN
         WRITE(10,10011)
         sec = sec+1
         WRITE(10,10010) sec
         slen = 0
      ENDIF

      IF (uval(1:1) .NE. ' ') THEN
         WRITE(10,10006) uval(:LEN_TRIM(uval)), sym, hgt,
     +                   lval(:LEN_TRIM(lval)), ang, x, y
      ELSE
         WRITE(10,10006) ' ', sym, hgt,
     +                   lval(:LEN_TRIM(lval)), ang, x, y
      ENDIF
      slen = slen+1

      RETURN
      END
!-----------------------------------------------------------------------
```

```
      SUBROUTINE PS_CLOSE (rows, cols)

      INTEGER*2 rows, cols

      CHARACTER ofile*64
      COMMON /output/ ofile

      INCLUDE 'ps_forms.inc'

      IF (path) THEN
         WRITE(10,'(A)') 'stroke'
         lx = 99999.
         path = .FALSE.
         plen = 0
      ENDIF

      WRITE(10,10011)
      WRITE(10,10020)
      DO i=1, sec
         WRITE(10,'(A,I3.3)') 'section', i
      ENDDO
      WRITE(10,10021) rows, cols

      CLOSE(10)

      WRITE (*,'(/,/,A,A)') ' PostScript output to: ',
     +   ofile(:LEN_TRIM(ofile))

      RETURN
      END
```

```
$TITLE: 'Filespec: d:\code\qmap\ssplot.for'
!---    Last edit: 06-Apr-1991 16:20, RB
C---
C---  Subroutine SSPLOT
C---
C---  Special symbol plot using USGS plotter
C---
C---  Arguments:
C---     ICODE
C---        1st digit
C---           =0 - New plot
C---           =1 - Same axis
C---        2nd digit
C---           =0 - Center symbol on tri
C---           =1 - Center symbol from keypunch
C---        3rd digit
C---           =0 - No axis
C---           =1 - X axis
C---           =2 - X axis and Y axis
C---           =3 - All four axes
C---
C---     XARRAY   - Array of X coordinates
C---     YARRAY   - Array of Y coordinates
C---     NPOINT   - Number of points
C---     XLEFT    - X value at left side of X axis
C---     XRIGHT   - X value at right side of X axis
C---     YLOWER   - Y value at lower end of Y axis
C---     YUPPER   - Y value at upper end of Y axis
C---     XLENG    - Length of X axis
C---     YLENG    - Length of Y axis
C---     SYM      - Symbol to be used
C---     HEIGHT   - The height of the symbol to be plotted
C---     SCALE    - Scale of map to be plotted
C---
C-------------------------------------------------------------------------

      SUBROUTINE SSPLOT (icode, xarray, yarray, npoint, xleft, xright,
     +                   ylower, yupper, xleng, yleng, sym, height,
     +                   scale, ista)

      REAL*4 xarray(*), yarray(*), height(*)
      CHARACTER sym(*)*1, ista(*)*4

C     Initialization
      ic = icode/100
      jc = (icode/10)-(icode/100)*10
      kc = icode-ic*100-jc*10
      xl = xleng
      yl = yleng
      ds = 1.E05/scale

C     Test for ic
      IF (ic .NE. 0) GO TO 15

C     Draw axes
      IF (kc .LE. 0) GO TO 15
      CALL PLOT (xl, 0., 2)
      CALL PS_MT (0., 0.)
      CALL PS_LT  (xl, 0.)
      IF (kc-2) 15, 12, 11
   11 CALL PLOT (xl, yl, 2)
      CALL PS_LT  (xl, yl)
      CALL PLOT (0., yl, 2)
      CALL PS_LT (0., yl)
   12 CALL PLOT (0., 0., 2)
      CALL PS_LT (0., 0.)

   15 CALL PLOT (0., 0., 3)
      IF (npoint .LT. 1) RETURN

!     Check bounds
      DO 30 i=1, npoint
```

```
      k = i
      IF (xarray(i) .LT. xleft) GO TO 25
      IF (xarray(i) .GT. xright) GO TO 25
      IF (yarray(i) .LT. ylower) GO TO 25
      IF (yarray(i) .GT. yupper) GO TO 25

C     Center symbol
   18 x = (xarray(i)-xleft)*ds
      y = (yarray(i)-ylower)*ds
      CALL SYMBOL (x, y, height(k), sym(k), 90., -1)

      SELECT CASE (ICHAR(sym(k)))
      CASE (43)
         CALL PS_CRS (height(k), x, y)
      CASE (10)
         CALL PS_TRI (height(k), x, y)
         CALL PS_SY (ista(k), 90, x+(height(k)*.6),
     +              y, -(height(k)*.25))
         CALL SYMBOL (x+(height(k)*.6), y, height(k)*.25,
     +               ista(k), 90., -4)
      CASE DEFAULT
         CALL PS_SY (sym(k), 90, x+(height(k)*.5),
     +              y-(height(k)*.45), height(k)*1.5)
      END SELECT
      GO TO 30

   25 WRITE(6,26) k
   26 FORMAT(' *** DATA NO.',I5,' IS OUTSIDE MAP BOUNDARIES ***')
   30 CONTINUE

      RETURN
      END
```

```
$TITLE: 'Filespec: d:\code\qmap\xycord.for'
!---    Last edit: 05-Apr-1991 17:23, RB
C---
C---     Subroutine XYCORD to compute the absolute coordinates PX and
C---  PY for N points whose latitude and longitude are PLAT and PLON
C---  with respect to the origin given by OLAT and OLON.
C---
C---     Latitudes and longitude must be given in minutes.
C---
C---     PX and PY are givin in 10**5 inches.
C---
C---     Calculations apply mainly to latitudes between 0ł and 70ł
C---
C-----------------------------------------------------------------------

      SUBROUTINE XYCORD (olat, olon, plat, plon, n, px, py)

      REAL mlat

      DIMENSION plat(*), plon(*), px(*), py(*)
      DIMENSION a(75), b(75)

      DATA a /1.855365, 1.855369, 1.855374, 1.855383, 1.855396,
     +        1.855414, 1.855434, 1.855458, 1.855487, 1.855520,
     1        1.855555, 1.855595, 1.855638, 1.855683, 1.855733,
     +        1.855786, 1.855842, 1.855902, 1.855966, 1.656031,
     2        1.856100, 1.856173, 1.856248, 1.856325, 1.856404,
     +        1.856488, 1.856573, 1.856661, 1.856750, 1.856843,
     3        1.856937, 1.857033, 1.857132, 1.857231, 1.857331,
     +        1.857435, 1.857538, 1.857643, 1.857750, 1.857858,
     4        1.857964, 1.858074, 1.858184, 1.858294, 1.858403,
     +        1.858512, 1.858623, 1.858734, 1.858842, 1.858951,
     5        1.859061, 1.859170, 1.859276, 1.859384, 1.859488,
     +        1.859592, 1.859695, 1.859798, 1.859896, 1.859995,
     6        1.860094, 1.860187, 1.860279, 1.860369, 1.860459,
     +        1.860544, 1.860627, 1.860709, 1.860787, 1.860861,
     7        1.860934, 0., 0., 0., 0./
      DATA b/ 1.842808, 1.842813, 1.842830, 1.842858, 1.842898,
     +        1.842950, 1.843011, 1.843085, 1.843170, 1.843265,
     1        1.843372, 1.843488, 1.843617, 1.843755, 1.843903,
     +        1.844062, 1.844230, 1.844408, 1.844595, 1.844792,
     2        1.844998, 1.845213, 1.845437, 1.845668, 1.845907,
     +        1.846153, 1.846408, 1.846670, 1.846938, 1.847213,
     3        1.847495, 1.847781, 1.848073, 1.848372, 1.848673,
     +        1.848980, 1.849290, 1.849605, 1.849922, 1.850242,
     4        1.850565, 1.850890, 1.851217, 1.851543, 1.851873,
     +        1.852202, 1.852531, 1.852860, 1.853188, 1.853515,
     5        1.853842, 1.854165, 1.854487, 1.854805, 1.855122,
     +        1.855433, 1.855742, 1.856045, 1.856345, 1.856640,
     6        1.856928, 1.857212, 1.857490, 1.857762, 1.858025,
     +        1.858283, 1.858553, 1.858775, 1.859008, 1.859235,
     7        1.859452, 0., 0., 0., 0./

      DO 10 i=1, n
      dlat = plat(i)-olat
      dlon = plon(i)-olon
      mlat = (plat(i)+olat)/120.
      jlat = mlat
      k = jlat+1
      IF (k .LT. 1) k = 1
      IF (k .GT. 71) k = 71
      flat = mlat-jlat
      aa = a(k)+flat*(a(k+1)-a(k))
      bb = b(k)+flat*(b(k+1)-b(k))
      x = aa*COS(mlat*.0174533)*dlon
      y = bb*dlat
      px(i) = -x*0.393696
      py(i) = -y*0.393696
   10 CONTINUE

      RETURN
      END
```

```
!---  Last edit: 04-Aug-1991 14:11, RB
!-------------------------------------------------------------------------

      COMMON /plotlib/
     +   dev_mode,       ! Display mode
     +   dev_chip,       ! Display adapter chip set
     +   dev_colors,     ! Number of available colors
     +   dev_xmin,       ! Minimum X coordinate in device space
     +   dev_ymin,       ! Minimum Y coordinate in device space
     +   dev_xmax,       ! Maximum X coordinate in device space
     +   dev_ymax,       ! Maximum Y coordinate in device space
     +   dev_cur_x,      ! Current X coordinate in device space
     +   dev_cur_y,      ! Current Y coordinate in device space
     +   user_col,       ! Current forground color
     +   user_xmin,      ! Minimum X in user space
     +   user_ymin,      ! Minimum Y in user space
     +   user_xmax,      ! Maximum X in user space
     +   user_ymax,      ! Maximum Y in user space
     +   user_cur_x,     ! Current X in user space
     +   user_cur_y,     ! Current Y in user space
     +   lib_x_con,      ! X conversion constant
     +   lib_y_con,      ! Y conversion constant
     +   plot_mode,      ! Plotting mode
     +   pi,             ! PI
     +   deg2rad,        ! Radians = Degrees * deg2rad
     +   font_spec       ! Font vectors filespec

      INTEGER*2 dev_mode, dev_chip, dev_colors, dev_xmin, dev_ymin,
     +          dev_xmax, dev_ymax,
     +          dev_cur_x, dev_cur_y, user_col
      REAL*4    user_xmin, user_ymin, user_xmax, user_ymax,
     +          user_cur_x, user_cur_y
      REAL*4    lib_x_con, lib_y_con, pi, deg2rad
      INTEGER*2 plot_mode
      CHARACTER font_spec*80

!-------------------------------------------------------------------------
```

```
      COMMON /Postscript/ path, plen, sec, slen, lx, ly, SEC_MAX
      LOGICAL*1 path
      INTEGER*2 plen, slen, sec, SEC_MAX
      REAL*4 lx,ly

!     Formats for PostScript output
10000 FORMAT (F6.2,1X,F6.2,' tr') ! translate
10001 FORMAT (F6.2,1X,F6.2,' lt') ! lineto
10002 FORMAT (F6.2,1X,F6.2,' mt') ! moveto
10003 FORMAT ('(',A,')',I4,1X,F6.2,1X,F6.2,1X,F4.2,' sy') ! symbol
10004 FORMAT (F4.2,1X,F6.2,1X,F6.2,' tri') ! triangle
10005 FORMAT (F4.2,1X,F6.2,1X,F6.2,' crs') ! cross (plus)
10006 FORMAT ('(',A,')',1X,'(',A,')',1X,F6.2,1X,'(',A,')',
     +       1X,I4,1X,F6.2,1X,F6.2,' lline') ! legend line
10007 FORMAT ('(',A,')',I4,1X,F6.2,1X,F6.2,1X,F4.2,' csy') ! center symbol
10010 FORMAT (/,'/section',I3.3,' {') ! section def
10011 FORMAT ('} def')
10020 FORMAT (/,'/map {') ! main def
10021 FORMAT ('} def',/,/,'{map} ',I2,1X,I2,' tile',/,/,
     +       '%%Trailer',/,'restore')
```

```
%!PS-Adobe-1.0
%%Creator: R. Banfill
%%Title: QMAP Procedures
%%For: QMAP version 1.01
%%CreationDate: April 7 1991 Last Edit: 4-13-91
%%EndComments

% All x y coords are given in inches, all angles in degrees

save
0 setlinewidth

% Tiling procedure
% Call: {picture_proc} rows cols tile
/tile
 { /rows exch def
   /columns exch def
   /bigpictureproc exch def
   newpath
    leftmargin botmargin moveto
    0 pageheight rlineto
    pagewidth 0 rlineto
    0 pageheight neg rlineto
   closepath clip
   leftmargin botmargin translate
   0 1 rows 1 sub
    { /rowcount exch def
      0 1 columns 1 sub
       { /colcount exch def
         gsave
          pagewidth colcount mul neg
          pageheight rowcount mul neg
          translate
          gsave
           pagewidth colcount mul
           pageheight rowcount mul
           translate
           initclip
           0 0 reg_mark
           0 pageheight reg_mark
           pagewidth pageheight reg_mark
           pagewidth 0 reg_mark
           /Helvetica-Oblique findfont 10 scalefont setfont
           gsave
            -3 15 translate
            90 rotate 0 0 moveto (Sheet: ) show
            colcount 1 add nstr cvs show (:) show
            rowcount 1 add nstr cvs show
            ( of ) show columns rows mul nstr cvs show
           grestore
          grestore
          bigpictureproc
          gsave showpage grestore
         grestore
       } for
    } for
 } def

% Print a registration mark
% Call: x y reg_mark
/reg_mark {
   gsave
    translate
    newpath
     0 0 moveto
     0 0 .1 inch 0 360 arc
     .15 inch 0 moveto
     -.15 inch 0 lineto
     0 .15 inch moveto
     0 -.15 inch lineto
    stroke
   grestore
```

```
  } def

% Convert points to inches
/inch {72 mul} def

% Variables
/nstr 2 string def
/leftmargin .5 inch def
/botmargin .5 inch def
/pagewidth 7.5 inch def
/pageheight 10 inch def

% Translate, lineto and moveto in inches
% Call: x y tr
/tr { exch inch exch inch translate } def
/lt { exch inch exch inch lineto } def
/mt { exch inch exch inch moveto } def

% Symbol call
% Call: (string) angle x y height sy
/sy { /Courier findfont
       exch inch scalefont setfont
       gsave tr
       rotate 0 0 moveto
       show grestore
    } def

% Print a triangle
% Call: height x y tri
/tri { gsave tr
        newpath
         /sz exch inch 2 div def
         sz neg 0 moveto
         sz .6 mul sz lineto
         sz .6 mul sz neg lineto
         closepath
        stroke
       grestore
     } def

% Print a cross
% Call: x y crs
/crs { gsave tr
        newpath
         /sz exch inch 2 div def
         sz neg 0 moveto
         sz 0 lineto
         0 sz neg moveto
         0 sz lineto
         closepath
        stroke
       grestore
     } def

% Print a legend line: l_val < symbol <= u_val
% Call: upper_val (symbol) height lower_val angle x y lline
/lline {
   gsave
    tr rotate 0 0 mt
    /Symbol findfont .25 inch scalefont setfont
    dup stringwidth pop 72 div
    ( \74) stringwidth pop 72 div
    add .35 add neg 0 mt
    show ( \74) show
    /Courier findfont
    exch inch scalefont setfont
    0 0 mt
    dup stringwidth pop 72 div
    2 div neg 0 mt show
    .35 0 mt
    /Symbol findfont .25 inch scalefont setfont
    dup ( ) ne { (\243 ) show show } if
```

```
   grestore
  } def

% Center string at x y
% Call: (string) angle x y height csy
/csy {
   gsave
    /Helvetica findfont
    exch inch scalefont setfont
    tr rotate 0 0 mt
    dup stringwidth pop 72 div
    2 div neg 0 mt
    show
   grestore
  } def

%%EndProlog

% Map sections follow
```

```
      PROGRAM MAKEQMAP

      IMPLICIT INTEGER*2 (a-z)

      CHARACTER hstn(256)*22, pstn(256)*5, buf*128, evn*128

      OPEN (10,FILE='hypo71pc.hdr')

      i = 0
   10 READ (10,'(A)', END=20) buf
      IF (i .GT. 0 .AND. buf(1:20) .EQ. '                    ') GOTO 20
      IF (buf(1:2) .EQ. '  ' .AND. buf(3:3) .NE. ' ') THEN
         i = i+1
         hstn(i) = buf(:22)
      ENDIF
      GOTO 10

   20 CLOSE (10)
      j = 0
      OPEN (10,FILE='hypo71pc.prt')

   30 READ (10,'(A)') buf
      IF ( buf(1:6) .EQ. '  DATE') THEN
         READ (10,'(A)') evn
      ELSE
         GOTO 30
      ENDIF

      j = -1
   40 READ (10,'(A)', END=50) buf
      IF (j .EQ. -1 .AND. buf(1:5) .EQ. '  STN') THEN
         j = 0
         GOTO 40
      ENDIF
      IF (j .EQ. -1) GOTO 40
      IF (j .GT. 0 .AND. buf(1:5) .EQ. '     ') GOTO 50

      j = j+1
      pstn(j) = buf(1:5)
      GOTO 40

   50 CLOSE (10)

      OPEN (10,FILE='qmap.inp')
      OPEN (11,FILE='makeqmap.inp')

   60 READ (11,'(A)',END=70) buf
      WRITE (10,'(A)') buf(:LEN_TRIM(buf))
      GOTO 60
   70 CLOSE (11)

      DO k=1, j
         DO l=1, i
            IF (hstn(l)(3:6) .EQ. pstn(k)(2:5)) THEN
               WRITE (10,'(A)') hstn(l)
            ENDIF
         ENDDO
      ENDDO

      WRITE (10,'(A)') ' '
      WRITE (10,'(A)') '$PointData'
      WRITE (10,'(A,A)') '(A80, T20, I2, 1X, F5.2, 1X, ',
     +                   'I3, 1X, F5.2, 2X, F5.2)'
      WRITE (10,'(A)') evn(:LEN_TRIM(evn))
      WRITE (10,'(A)') ' '
      WRITE (10,'(A)') '$End'

      CLOSE (10)

      STOP' '
      END
```

XDETECT V2.0.4

Summary of New User Interface and Communication Features

by

Dean Tottingham

June 14, 1992

1. INTRODUCTION

This document summarizes the user interface and communication features that are supported fully tested and debugged in XDETECT V2.0.4. XDETECT V2.0.4 represents the second, release version of XDETECT. Many of the features described in the following sections have been in place for some time and have undergone significant testing in pre-release, beta versions V2.0.1, V2.0.2, and V2.0.3.

2. USER INTERFACE FEATURES

New features:

a. Added support for the PowerSTOR subroutine library. PowerSTOR allows us to break the 640K RAM limitation imposed by DOS and to take advantage of the megabytes of extended memory. We now use extended memory to store the pre-trigger queue.

b. Added a sophisticated lexer and recursive descent parser/lexer. This new lexer/parser allows us to do extensive syntax checking on the input file.

Changes to the input file:

a. Added the FirstDifference parameter. The FirstDifference parameter activates/deactivates the first difference calculation in the XDETECT trigger algorithm. Normally activated, the first difference calculation filters out low frequency signals which reduces the algorithm's sensitivity to long period events. With the first difference calculation deactivated, the XDETECT trigger algorithm can detect long period, teleseism events.

b. Added the LTALowerBound parameter. The LTALowerBound parameter sets the minimum background noise level permitted for LTA. By initializing LTA to the LTALowerBound, we minimize false triggers at startup. LTA values less than the LTALowerBound setting are adjusted upwards to the LTALowerBound value. The default LTALowerBound is 30. Setting the LTALowerBound less than/greater than 30 will have the effect of increasing/decreasing the trigger algorithm's sensitivity to small-signal events.

c. Added the PreEventRecording parameter. The PreEventRecording parameter activates/deactivates the writing of a small "pre-event" file in addition to the normal waveform file when an event is detected. This feature is normally deactivated. The whole point of writing such a file is to give off-line picker/location programs (that are monitoring the disk for new events) a head start in computing picks and locations. This "pre-event" file has a .PRE file type.

d. Added the AutoFreerun parameter. The AutoFreeRun parameter activates/deactivates the XDETECT's free run mode at startup. The XDETECT free run mode is normally deactivated.

e. Added the FreerunBlockTime parameter. The FreerunBlockTime parameter specifies the size of each free run file in seconds. A free run event will then span across multiple waveform files, each FreerunBlockTime seconds long. The default setting of FreerunBlockTime is infinite.

f. Added the STAverageWindow and LTAverageWindow parameters. The STAverageWindow and LTAverageWindow parameters specify the number of samples used to compute the short-term average and long term average respectively. The default LTA window size is 256 data points while the default STA window size is 16 data points.

g. Eliminated the NumberOfExtBuffers parameter. The number of external buffers is computed empirically.

h. Changed the ChannelBlocksize parameter semantics. The ChannelBlocksize defaults to 32768 / # of channels in station list.

Changes to the screen:
---

a. Added support for 16-bit data. In particular, added code to scale data points based on a variable A/D dynamic range.

b. Added the uptime indicator on the screen to help the user gauge XDETECT reliability.

c. Added the multiplexer bank switch indicator on the screen to help the user gauge multiplexer reliability.

d. Added the channel scroll feature. The channel scroll feature allows the user to view all input channels in 16 channel chunks. You enter the scroll display mode by pressing either CNTRL-PGUP or CNTRL-PGDN. Repeated CNTRL-PGUP commands cause XDETECT to scroll upward through the station list sixteen channels at a time. Repeated CNTRL-PGDOWN commands cause XDETECT to scroll downward through the station list sixteen channels at a time. Press CNTRL-HOME to exit the scroll display mode. In the scroll display mode, XDETECT maintains a scroll bar on the left side of the screen indicating which channels are currently being viewed with respect to the entire station list.

e. Added the help screen to show key bindings. Press F1 to display the help screen and any key to return to the trace display screen.

f. Added the time scroll feature. XDETECT automatically scrolls data onto the screen when buffers are less than 512 samples in size.

g. Added the selected channel display feature. The user can now select any 32 channels to be displayed on the screen by simply adding a Display= ON attribute to the corresponding station definitions in the input file.

3. COMMUNICATION FEATURES

XDETECT V2.0.4 supports the remote communication requirements specified in the Work Statement for Requisition# 9930-0450. In particular, running on a MS-DOS PC, XDETECT can write waveform and log files to another MS-DOS PC, to a SCO UNIX PC, and to a SunOS Sun-4 (i.e., a Sparcstation). In fact, XDETECT can write files to any file system as long as the destination file system looks like a logical drive on the PC.

TottCo Consulting Group decided not to implement their own communication protocol in light of the protocols available in both the public and private domains.

TottCo Consulting Group recommends PC/NFS, a Sun Microsystems product that permits transparent file transfer between MSDOS and UNIX machines. In particular, PC/NFS allows you to:

- Share files with other DOS users on your network, without exchanging diskettes.
- Transfer files between systems by using simple DOS and UNIX commands.
- Share files with users of different operating systems in your NFS network including UNIX and VMS.
- Use the file-sharing and file-locking services provided by DOS 3.1.
- Remotely log into non-NFS systems.
- Access all of these facilities either directly or an Ethernet network or over a serial line, such as a modem or an RS-232 line.

4. SOURCE CODE LISTINGS

The following pages contain source code listings of the following modules: mbase36.c, mbase36.h, mboot.asm, mconst.h, mdemux.c, mdemux.h, mdemux.asm, mdosfunc.h, mdsp.c, mdsp.h, mdspstub.c, mdspstub.h, mdt28xx.c, mdt28xx.h, merror.c, merror.h, mevent.c, mevent.h, mfile.c, mfile.h, mfreerun.c, mfreerun.h, mhalfsp.c, mhalfsp.h, mhelp.h, mlatlon.c, mlatlon.h, mlexer.c, mlexer.h, mlocate.c, mlocate.h, mlog.c, mlog.h, mparse.c, mparse.h, mpick.c, mpick.h, mqueue.c, mqueue.h, mreboot.c, mreboot.h, mscreen.c, mscreen.h, mscrnhdr.c, mscrnhdr.h, mstation.c, mstation.h, msudsini.c, msudsini.h, mtrigger.c, mtrigger.h, mutils.c, mutils.h, powrstor.h, suds.h, timer.c, timer.h, xdetect.c, xdetect.h.

```
/* FILE: mbase36.c                                  (D. Tottingham  04/26/92)

This is a collection of C functions that manage base 36 numbers for xdetect.
All functions have been written and compiled medium.  The following functions
are included:

b_convert_base10 ()             convert base 36 number into base 10 number
b_incr_base36 ()                increment a base 36 number
b_init_base36 ()                initialize a base 36 number

EXTERNAL FUNCTIONS CALLED:
   none

HISTORY:
   none

*/


/*************************************************************************
                            INCLUDE FILES

*************************************************************************/
#include "mbase36.h"
#include "mconst.h"


/*************************************************************************
                                GLOBALS

*************************************************************************/
char * base36_num[] = {                 /* base36 conversion array */
       "0", "1", "2", "3", "4", "5", "6", "7",
       "8", "9", "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"
       };


/*-----------------------------------------------------------------------*
 *                        b_convert_base10                               *
 *-----------------------------------------------------------------------*/
/* Convert a base 36 number into a base 10 number.                       */

PUBLIC
unsigned int b_convert_base10 (number)
B_BASE36 number;
{
   return (number.high * 36 + number.low);
}

/*-----------------------------------------------------------------------*
 *                          b_incr_base36                                *
 *-----------------------------------------------------------------------*/
/* Increment a base 36 number.                                           */

PUBLIC
void b_incr_base36 ( number_ptr )
B_BASE36 * number_ptr;
{
   number_ptr->low++;
   if (number_ptr->low == 36) {
      number_ptr->high++;
      number_ptr->low = 0;
   }
}

/*-----------------------------------------------------------------------*
 *                          b_init_base36                                *
 *-----------------------------------------------------------------------*/
/* Initialize a base 36 number.                                          */

PUBLIC
void b_init_base36 ( number_ptr )
B_BASE36 * number_ptr;
{
   (*number_ptr).high = 0;
   (*number_ptr).low  = -1;
}
```

```
/* FILE: mbase36.h                                  (D. Tottingham  04/26/92)

This is an include file of the defines, data structure definitions and
external data declarations for base 36 number manipulation.

*/

#ifndef _MBASE36_
#define _MBASE36_

/************************************************************************
                                INCLUDES

************************************************************************/
#include "mconst.h"


/************************************************************************
                          STRUCTURE DEFINITIONS

The following structure definitions are included in mbase36.h, so that all
modules can have access to them.

************************************************************************/
typedef struct {
        int     high;
        int     low;
} B_BASE36;


/************************************************************************
                              EXTERNAL DATA

The following external data can be accessed by any module that includes
this file.

************************************************************************/
PUBLIC char * base36_num[];              /* base36 conversion array */


/************************************************************************
                          EXTERNAL DECLARATIONS

These functions can be called from all modules that include this file.

************************************************************************/
PUBLIC unsigned int b_convert_base10 (B_BASE36);
PUBLIC void b_init_base36 (B_BASE36 *);
PUBLIC void b_incr_base36 (B_BASE36 *);

#endif
```

```
; FILE: mboot.asm                                (D. Tottingham  07/05/89)
;
; void boot ( temperature)
; int temperature;
;
; This routine reboots the system.  To reboot the system, two things must be
; done:
;
;      1.  Set the BIOS reset_flag at address 0040H:0072H to 7F7FH for a warm
;          boot or to 1234H for a cold boot.
;      2.  Do a hardwired jump to the low-level BIOS at address 0FFFFH:0000H.
;
; The variable passed to this routine is as follows:
;
;       temperature     = warm/cold boot indicator set as follows:
;                                 0  =    COLD boot
;                                 1  =    WARM boot
;
; This is a MEDIUM model subroutine.
;

parmdef         struc
                dw      (?)                     ; old bp
                dd      (?)                     ; return address
temperature     dw      (?)
parmdef         ends


_BIOS           segment at 0FFFFh
                assume  cs:_BIOS
bios_address:
_BIOS           ends

                public  _boot

_TEXT           segment byte public 'CODE'
                assume  cs:_TEXT

_boot           proc    far
                push    bp                      ; save bp
                mov     bp, sp
                push    ds

                mov     ax, 40h                 ; ds:[dx] -> 40h:72h
                mov     ds, ax
                mov     bx, 72h

; Set reset_flag according to value of temperature
                mov     ax, [bp].temperature    ; get temperature
                cmp     ax, 0                   ; is it COLD?
                jnz     warm

cold:           mov     [bx], 07F7Fh            ; it's COLD
                jmp     short reboot

warm:           mov     [bx], 01234h            ; it's WARM


; Reboot the system by jumping to address 0FFFFh:0000h
reboot:
                jmp     FAR PTR bios_address

                pop     ds
                pop     bp
                ret
_boot           endp

_TEXT           ends
                end
```

```
/* FILE: mconst.h                                (D. Tottingham  04/26/92)

This is an include file of defines, data structure definitions and
external declarations that are common in all xdetect modules.

*/

#ifndef _MCONST_
#define _MCONST_

/************************************************************************
                                DEFINES
************************************************************************/
#define PUBLIC                  extern
#define PRIVATE                 static

/* Boolean Constants */
#define TRUE                    1
#define FALSE                   0
#define ON                      TRUE
#define OFF                     FALSE
#define YES                     TRUE
#define NO                      FALSE

/* Miscellaneous Constants */
#define MAX_FILENAME_LENGTH     50
#define MAX_NUMSTR_LENGTH       20
#define MAX_STATION_NAME        10
#define RADIX                   10
#define SEGMENT_SIZE            32768L          /* words */


/************************************************************************
                                TYPEDEFS
************************************************************************/
typedef unsigned int            FLAG;
typedef unsigned char           BYTE;

#endif
```

```
/* FILE: mdemux.c                                   (D. Tottingham  04/26/92)

This is a collection of C functions that manage the demux data queue for
xdetect.  All functions have been written and compiled medium model. The
following functions are included:

dm_get_current_time ()          get current time
dm_get_first_buffer ()          get first new demux buffer from demux queue
dm_get_head_buffer ()           get buffer at head of demux queue
dm_get_new_buffers ()           get buffer(s) from ADC and demultiplex
dm_get_next_buffer ()           get next buffer from demux queue
dm_get_pre_event_time ()        get pre event time
dm_get_tail_buffer ()           get buffer at tail of demux queue
dm_get_uptime ()                get session up time.
dm_initialize ()                initialize the demux queues
dm_initialize_params ()         initialize parameters
dm_initialize_time ()           initialize the time
dm_reset ()                     reset this module and release all storage
dm_set_PreEventTime ()          set PreEventTime constant

EXTERNAL FUNCTIONS CALLED:

dt_get_channel_size ()          get channel block size
dt_get_digitization_rate ()     get digitization rate
dt_get_new_buffers ()           get new mux buffer(s) from DT2821
dt_get_next_buffer ()           get buffer at tail of mux queue
dt_get_number_of_ext_buffers () get number of external buffers
dt_get_scan_count ()            get scan count
er_abort ()                     display an error message then quit
o_write_logfile ()              write string to log file stream
q_dequeue ()                    dequeue a data link from a data queue
q_enqueue ()                    enqueue a data link on a data queue
q_initialize ()                 initialize a data queue
suds_initialize ()              initialize a suds structure
suds_initialize_tag ()          initialize a suds structtag
u_convert_time ()               convert a struct timeb into abs. time

HISTORY:
   none

*/


/*************************************************************************
                            INCLUDE FILES
*************************************************************************/
#include <malloc.h>
#include <stdio.h>
#include <sys\types.h>
#include <sys\timeb.h>
#include <time.h>

#include "mconst.h"
#include "mdemux.h"
#include "mdt28xx.h"
#include "merror.h"
#include "mlog.h"
#include "mqueue.h"
#include "msudsini.h"
#include "mutils.h"
#include "powrstor.h"
#include "xdetect.h"


/*************************************************************************
                                GLOBALS
*************************************************************************/
PRIVATE Q_QUEUE demux_queue;
PRIVATE Q_LINK * head_ptr, * first_ptr;
PRIVATE double current_time, startup_time;
PRIVATE double pre_event_time;
```

```
/*-----------------------------------------------------------------------*
 *                              adjust_queue                             *
 *-----------------------------------------------------------------------*/
/* Adjust the pre-trigger queue and return the used ST_heap_ptr.         */

PRIVATE
struct STORheap far * adjust_queue (buffer_time)
double buffer_time;
{
   struct STORheap far * sheap_ptr;
   STORstatus stat;
   Q_TYPE type;

   sheap_ptr = NULL;
   if ((current_time - startup_time) > (pre_event_time + buffer_time)) {
      if (q_dequeue (&demux_queue, &type)) {
         sheap_ptr = type.buffer->ST_heap_ptr;
         _ffree (type.buffer);
      }
   }
   return (sheap_ptr);
}

/*-----------------------------------------------------------------------*
 *                       create_available_buffers                        *
 *-----------------------------------------------------------------------*/
/* Convert NEW buffers into AVAILABLE buffers.                           */

PRIVATE
void create_available_buffers ()
{
   Q_LINK * hptr;

   for (hptr = demux_queue.head; hptr != NULL; hptr = hptr->next)
      if (hptr->type.buffer->buffer_status == NEW_BUFFER) {
         hptr->type.buffer->buffer_status = AVAILABLE_BUFFER;
      }
}

/*-----------------------------------------------------------------------*
 *                         get_available_buffer                          *
 *-----------------------------------------------------------------------*/
/* Find an available data buffer.                                        */

PRIVATE
int far * get_available_buffer ()
{
   Q_LINK * hptr;

   for (hptr = demux_queue.head;
        hptr != NULL && hptr->type.buffer->buffer_status != NEW_BUFFER;
        hptr = hptr->next)
      if (hptr->type.buffer->buffer_status == AVAILABLE_BUFFER) {
         hptr->type.buffer->buffer_status = ARCHIVED_BUFFER;
         return (hptr->type.buffer->data);
      }
   return (NULL);
}

/*-----------------------------------------------------------------------*
 *                                get_data                               *
 *-----------------------------------------------------------------------*/
/* If data is not in base memory, get it from the Sheap.                 */

PRIVATE
void get_data (buffer)
Q_BUFFER far * buffer;
{
   STORstatus stat;
   unsigned int actual_buffer_size;
```

```
   if (buffer == NULL || buffer->buffer_status != ARCHIVED_BUFFER)
      return;

   actual_buffer_size = buffer->buffer_size * sizeof(unsigned);
   if ((buffer->data = get_available_buffer()) == NULL) {
      buffer->data = (int far *) _fmalloc ( actual_buffer_size);
      if (buffer->data == NULL) er_abort (DM_NO_STORAGE);
   }

   buffer->buffer_status = AVAILABLE_BUFFER;
   stat = getSTOR(*(buffer->ST_heap_ptr), 0L, buffer->data, actual_buffer_size);
   if (stat != Okay) er_abort (DM_POWRSTOR + stat);
}

/*------------------------------------------------------------------------*
 *                          dm_get_current_time                           *
 *------------------------------------------------------------------------*/
/* Get current time.                                                      */

PUBLIC
double dm_get_current_time ()
{
   return (current_time);
}

/*------------------------------------------------------------------------*
 *                          dm_get_first_buffer                           *
 *------------------------------------------------------------------------*/
/* Get first new demux buffer from demux queue.                           */

PUBLIC
Q_BUFFER far * dm_get_first_buffer ()
{
   head_ptr = first_ptr;
   if (head_ptr != NULL) {
      get_data (head_ptr->type.buffer);
      return (head_ptr->type.buffer);
   } else return (NULL);
}

/*------------------------------------------------------------------------*
 *                          dm_get_head_buffer                            *
 *------------------------------------------------------------------------*/
/* Get buffer at head of queue.                                           */

PUBLIC
Q_BUFFER far * dm_get_head_buffer ()
{
   head_ptr = demux_queue.head;
   if (head_ptr != NULL) {
      get_data (head_ptr->type.buffer);
      return (head_ptr->type.buffer);
   } else return (NULL);
}

/*------------------------------------------------------------------------*
 *                          dm_get_new_buffers                            *
 *------------------------------------------------------------------------*/
/* Get buffer from ADC.  Update buffer_start_time.  Demultiplex the raw
   data and save link in data queue.                                      */

PUBLIC
void dm_get_new_buffers ()
{
   Q_BUFFER far * mux, far * demux;
   Q_TYPE type;
   STORstatus stat;
   unsigned int actual_buffer_size;

   /* Wait for a buffer to complete */
   dt_get_new_buffers ();
   first_ptr = NULL;

   /* Free up the new demux data buffers */
   create_available_buffers();

   while ( (mux = dt_get_next_buffer()) != NULL) {
      demux = (Q_BUFFER far *) _fmalloc (sizeof (Q_BUFFER));
      if (demux == NULL) er_abort (DM_NO_STORAGE);

      suds_initialize (MUXDATA, &demux->info);
      suds_initialize_tag (MUXDATA, &demux->structtag);

      demux->bank_switched = mux->bank_switched;
      demux->dynamic_range = mux->dynamic_range;
      demux->buffer_status = NEW_BUFFER;
      demux->buffer_size = mux->buffer_size;
      demux->structtag = mux->structtag;
      demux->info = mux->info;
      demux->info.begintime = current_time;
      demux->info.loctime = 0;

      actual_buffer_size = mux->buffer_size * sizeof(unsigned);
      demux->data = get_available_buffer();
      if (demux->data == NULL) {
         demux->data = (int far *) _fmalloc ( actual_buffer_size);
         if (demux->data == NULL) er_abort (DM_NO_STORAGE);
      }

      demux->ST_heap_ptr = adjust_queue (mux->seconds_in_buffer);
      if (demux->ST_heap_ptr == NULL) {
         demux->ST_heap_ptr = (struct STORheap far *) _fmalloc ( sizeof(struct STORh
         if (demux->ST_heap_ptr == NULL) er_abort (DM_NO_STORAGE);
         stat = newSheap( demux->ST_heap_ptr);
         if (stat != Okay) er_abort (DM_POWRSTOR + stat);
         stat = setSheap( *(demux->ST_heap_ptr),
                          (unsigned long) (actual_buffer_size));
         if (stat != Okay) er_abort (DM_POWRSTOR + stat);
      }

      /* Demultiplex the mux buffer */
      to_demux (mux->data, demux->data, mux->buffer_size *
                sizeof(unsigned), mux->info.numchans);

      /* Copy the demultiplexed data into the Sheap */
      stat = putSTOR( *(demux->ST_heap_ptr), demux->data, 0L, actual_buffer_size);
      if (stat != Okay) er_abort (DM_POWRSTOR + stat);

      /* Put the demuz buffer in the queue */
      type.buffer = demux;
      q_enqueue (&demux_queue, type);

      current_time += mux->seconds_in_buffer;
      if (first_ptr == NULL) first_ptr = demux_queue.tail;
   }
}

/*------------------------------------------------------------------------*
 *                          dm_get_next_buffer                            *
 *------------------------------------------------------------------------*/
/* Get next buffer from queue.                                            */

PUBLIC
Q_BUFFER far * dm_get_next_buffer ()
{
   head_ptr = head_ptr->next;
   if (head_ptr != NULL) {
      get_data (head_ptr->type.buffer);
      return (head_ptr->type.buffer);
   } else return (NULL);
}

/*------------------------------------------------------------------------*
 *                          dm_get_pre_event_time                         *
 *------------------------------------------------------------------------*/
/* Get pre event time.                                                    */
```

```
PUBLIC
double dm_get_pre_event_time ()
{
   return (pre_event_time);
}

/*-------------------------------------------------------------------------*
 *                          dm_get_tail_buffer                             *
 *-------------------------------------------------------------------------*/
/* Get buffer at tail of queue.                                            */

PUBLIC
Q_BUFFER far * dm_get_tail_buffer ()
{
   if (demux_queue.tail != NULL) {
      get_data (demux_queue.tail->type.buffer);
      return (demux_queue.tail->type.buffer);
   } else return (NULL);
}

/*-------------------------------------------------------------------------*
 *                          dm_get_uptime                                  *
 *-------------------------------------------------------------------------*/
/* Get session up time.                                                    */

PUBLIC
double dm_get_uptime ()
{
   return (current_time - startup_time);
}

/*-------------------------------------------------------------------------*
 *                          dm_initialize                                  *
 *-------------------------------------------------------------------------*/
/* Initialize the queues and calculate some key values.                    */

PUBLIC
void dm_initialize ()
{
   static union INITelement init_storage[2];
   STORstatus stat;
   int low_limit, high_limit;
   unsigned int needed_size, chunks_in_block, blocks_in_queue;
   unsigned long samples_in_queue, bytes_in_block, channel_size;

   /* Dedicate an area of extended memory to PowerSTOR. Upper and
      lower limits are specified in 16K chunks from the bottom of
      base memory.  Note that there are 8 16K chunks in each ATLAB
      buffer (128K bytes) and 64 16K chunks to the 1M byte line.
    */
   low_limit = 64;
   high_limit = -(dt_get_number_of_ext_buffers() * 8 + 1);

   /* Calculate the amount of PowerSTOR needed to hold the pre-trigger queue */
   channel_size = dt_get_channel_size();
   bytes_in_block = channel_size * dt_get_acan_count() * sizeof(unsigned);
   chunks_in_block = bytes_in_block / SMALLEST_CHUNK;
   if (bytes_in_block % SMALLEST_CHUNK) chunks_in_block++;
   samples_in_queue = pre_event_time * dt_get_digitization_rate();
   blocks_in_queue = (samples_in_queue / channel_size) + 1;
   if (samples_in_queue % channel_size) blocks_in_queue++;
   needed_size = chunks_in_block * blocks_in_queue + POWRSTOR_USAGE;
   if (needed_size < 4) needed_size = 4;

   /* Fill the init_storage array */
   init_storage[0].EXTmem.memType = ExtMem;
   init_storage[0].EXTmem.LoLimit = low_limit;
   init_storage[0].EXTmem.HiLimit = high_limit;
   init_storage[1].Nulmem.memType = NulMem;

   /* Initialize PowerSTOR */
   stat = InitPwr( needed_size, init_storage);
   if (stat != Okay) er_abort (DM_POWRSTOR + stat);

   q_initialize (&demux_queue);
   first_ptr = head_ptr = NULL;
}

/*-------------------------------------------------------------------------*
 *                          dm_initialize_params                           *
 *-------------------------------------------------------------------------*/
/* Initialize parameters.                                                  */

PUBLIC
void dm_initialize_params ()
{
   pre_event_time = PRE_EVENT_TIME;
}

/*-------------------------------------------------------------------------*
 *                          dm_initialize_time                             *
 *-------------------------------------------------------------------------*/
/* Initialize the time.                                                    */

PUBLIC
void dm_initialize_time ()
{
   struct timeb current_timeb;

   ftime (&(current_timeb));
   startup_time = current_time = u_convert_time (current_timeb);
}

/*-------------------------------------------------------------------------*
 *                          dm_reset                                       *
 *-------------------------------------------------------------------------*/
/* Reset this module and release all storage.                              */

PUBLIC
void dm_reset ()
{
   Q_TYPE type;

   while (q_dequeue (&demux_queue, &type)) {
      if (type.buffer->buffer_status != ARCHIVED_BUFFER)
         _ffree (type.buffer->data);
      _ffree (type.buffer->ST_heap_ptr);
      _ffree (type.buffer);
   }
   QuitPwr();
}

/*-------------------------------------------------------------------------*
 *                          dm_set_PreEventTime                            *
 *-------------------------------------------------------------------------*/
/* Set PreEventTime constant.                                              */

PUBLIC
void dm_set_PreEventTime (PreEventTime)
double PreEventTime;
{
   pre_event_time = PreEventTime;
}
```

```
/* FILE: mdemux.h                                (D. Tottingham  04/26/92)

This is an include file of defines, data structure definitions and
external declarations that are common in the mdemux module.

*/

#ifndef _MDEMUX_
#define _MDEMUX_

/************************************************************************
                                INCLUDES

************************************************************************/
#include "mconst.h"
#include "mqueue.h"


/************************************************************************
                                DEFINES

************************************************************************/
#define PRE_EVENT_TIME                  11      /* 11 seconds */

#define NEW_BUFFER                      3
#define AVAILABLE_BUFFER                2
#define ARCHIVED_BUFFER                 1


/************************************************************************
                        EXTERNAL DECLARATIONS

These functions can be called from all modules that include this file.

************************************************************************/
PUBLIC double dm_get_current_time ();
PUBLIC Q_BUFFER far * dm_get_first_buffer ();
PUBLIC Q_BUFFER far * dm_get_head_buffer ();
PUBLIC void dm_get_new_buffers ();
PUBLIC Q_BUFFER far * dm_get_next_buffer ();
PUBLIC double dm_get_pre_event_time ();
PUBLIC Q_BUFFER far * dm_get_tail_buffer ();
PUBLIC double dm_get_uptime ();
PUBLIC void dm_initialize ();
PUBLIC void dm_initialize_params ();
PUBLIC void dm_initialize_time ();
PUBLIC void dm_reset ();
PUBLIC void dm_set_PreEventTime (double);

#endif
```

```
; FILE: mdmux.asm                                (D. Tottingham  07/21/89)
;
; to_demux ( sbuffer, dbuffer, bufsize, channelnum )
; unsigned int far * sbuffer;
; unsigned int far * dbuffer;
; unsigned int bufsize;
; int channelnum;
;
; This function demultiplexes the source buffer into the destination buffer,
; where:
;
;       sbuffer         -> far pointer to source buffer
;       dbuffer         -> far pointer to destination buffer
;       bufsize         = size of the source buffer in bytes
;                         size must be a power of two
;       channelnum      = number of channels
;
; NOTE: Source and destination buffers must be a power of two in length.
;       In order to make this a high speed demux routine, bit shifting is
;       involved which forces a bufsize of a power of two in length.
;
; This is a MEDIUM model subroutine.
;

;               include mmdefs.inc              ; memory model definitions

                public  _to_demux

parmdef         struc                           ; stack structure
                dw      ?                       ; old bp
                dd      ?                       ; return address
sbuffer         dd      ?
dbuffer         dd      ?                       ; far pointer
bufsize         dw      ?
channelnum      dw      ?
parmdef         ends

allhi           equ     0FFFFh

_TEXT           segment byte public 'CODE'
                assume cs:_TEXT
_to_demux       proc    far
                push    bp                      ; save old bp
                mov     bp,sp

                push    ds                      ; save registers
                push    es
                push    si
                push    di
                push    ax
                push    bx
                push    cx
                push    dx

                lds     si, [bp].sbuffer        ; pnt to buffers
                les     di, [bp].dbuffer

                sub     si, si                  ; reset si

; generate mask in bx for testing end of buffer
; this mask will have all most significant bits set to 1

                mov     ax, [bp].bufsize        ; save bufsize
                mov     bx, allhi
loop0:          shr     ax, 1
                je      done0
                shl     bx, 1
                jmp short loop0
done0:

; generate mask in dx for resetting source pointer to top of source buffer
; this mask is the one's complement of bx

                mov     dx, bx
                not     dx

; initialize channel counter in cx and mux length in [bp].channelnum

                mov     cx, [bp].channelnum
                add     [bp].channelnum, cx

; cycle channelnum times through the source buffer

loop1:          dec     cx
                jl      done

loop2:          push    si
                add     si, WORD PTR [bp].sbuffer
                mov     ax, [si]
                movsw
                pop     si
                add     si, [bp].channelnum
                test    bx, si
                jz      loop2

; wrap around for another channel

                and     si, dx
                inc     si
                inc     si
                jmp     short loop1

done:           pop     dx
                pop     cx
                pop     bx
                pop     ax
                pop     di
                pop     si
                pop     es
                pop     ds
                pop     bp

                ret
_to_demux       endp

_TEXT           ends
                end
```

```
/* FILE: mdosfunc.h                                (D. Tottingham  11/24/89)

This is an include file of the defines, data structure definitions and
external data declarations for using the mdosfunc library.

*/

#ifndef _MDOSFUNC_
#define _MDOSFUNC_

/************************************************************************
                                INCLUDES

************************************************************************/
#include "mconst.h"


/************************************************************************
                                DEFINES

************************************************************************/
/* open_file () Commands */
#define READ                    0
#define WRITE                   1


/************************************************************************
                        EXTERNAL DECLARATIONS

These functions can be called from all modules that include this file.

************************************************************************/
PUBLIC unsigned close_file (int);
PUBLIC unsigned create_file (char *);
PUBLIC unsigned open_file (char *, char);
PUBLIC unsigned read_file (int, int, int far *);
PUBLIC unsigned seek_file (int, long, char);
PUBLIC unsigned write_file (int, int, int far *);

#endif
```

```
/* FILE: mdspstub.c                                (D. Tottingham  04/26/92)

This file contains stubs for functions normally found in mdsp.c.  They are
stubbed for several reasons: (1) this functionality is not supported in the
xdetect distribution, (2) this functionality is subject to change, and
(3) this functionality has not undergone significant testing which could
hamper other debugging efforts.

All functions have been written and compiled medium model.  The following
functions are included:

dsp_continue_band_recording ()  continue to record fft band info into .fft file
dsp_domain_check ()             analyze the current buffer for spectral lines
dsp_get_dsp_status ()           get dsp_enabled flag
dsp_get_spectral_channel ()     return spectral trigger channel
dsp_get_status ()               return char for spectral signal
dsp_initialize ()               initialize DSP package
dsp_initialize_params ()        initialize key parameters
dsp_set_band ()                 set band limits
dsp_set_bufs_to_avg ()          set buffers to average
dsp_set_dsp_status ()           set dsp_enabled flag
dsp_set_fft_file_status ()      set fft_file_enabled flag
dsp_set_MaxCalibrationTime ()   passes max calibration time to mdsp
dsp_set_spectral_status ()      set spectral_enabled flag
dsp_spectral_detection ()       return flag for spectral_detection
dsp_spectral_detection_done ()  return flag for spectral recording done

EXTERNAL FUNCTIONS CALLED:
   none

HISTORY:
   none


/*****************************************************************************
                              INCLUDE FILES
*****************************************************************************/
#include <stdio.h>

#include "mconst.h"


/*****************************************************************************
                                 GLOBALS

*****************************************************************************/


/*---------------------------------------------------------------------------*
 *                          dsp_continue_band_recording                      *
 *---------------------------------------------------------------------------*/
/* Continue to record fft band info into .fft file                           */

PUBLIC
void dsp_continue_band_recording ()
{
}

/*---------------------------------------------------------------------------*
 *                          dsp_domain_check                                 *
 *---------------------------------------------------------------------------*/
/* Analyze the current buffer for characteristic spectral lines and          */
/* return flag to signal main() that an interesting signal is occurring.     */

PUBLIC
FLAG dsp_domain_check ()
{
   return FALSE;
}

/*---------------------------------------------------------------------------*
 *                          dsp_get_dsp_status                               *
 *---------------------------------------------------------------------------*/
/* Get dsp_enabled.                                                          */

PUBLIC
FLAG dsp_get_dsp_status ()
{
   return FALSE;
}

/*---------------------------------------------------------------------------*
 *                          dsp_get_spectral_channel                         *
 *---------------------------------------------------------------------------*/
/* Return spectral trigger channel.                                          */

PUBLIC
unsigned int dsp_get_spectral_channel ()
{
   return 0;
}

/*---------------------------------------------------------------------------*
 *                          dsp_get_status                                   *
 *---------------------------------------------------------------------------*/
/* Return char for spectral signal.                                          */

PUBLIC
char dsp_get_status ()
{
   return FALSE;
}

/*---------------------------------------------------------------------------*
 *                          dsp_initialize_params                            *
 *---------------------------------------------------------------------------*/
/* Initialize key parameters.                                                */

PUBLIC
void dsp_initialize_params ()
{
}

/*---------------------------------------------------------------------------*
 *                          dsp_initialize                                   *
 *---------------------------------------------------------------------------*/
/* Initialize arrays and variables used in DSP package.                      */

PUBLIC
void dsp_initialize()
{
}

/*---------------------------------------------------------------------------*
 *                          dsp_set_band                                     *
 *---------------------------------------------------------------------------*/
/* Set band limits.                                                          */

PUBLIC
void dsp_set_band (low, high)
float low, high;
{
}

/*---------------------------------------------------------------------------*
 *                          dsp_set_bufs_to_avg                              *
 *---------------------------------------------------------------------------*/
/* Set buffers to average.                                                   */

PUBLIC
void dsp_set_bufs_to_avg (number)
int number;
{
}
```

```
/*-------------------------------------------------------------------------*
 *                          dsp_set_dsp_status                             *
 *-------------------------------------------------------------------------*/
/* Set dsp_enabled flag.                                                   */

PUBLIC
void dsp_set_dsp_status (dsp_status)
FLAG dsp_status;
{
}

/*-------------------------------------------------------------------------*
 *                        dsp_set_fft_file_status                          *
 *-------------------------------------------------------------------------*/
/* Set fft_file_enabled flag.                                              */

PUBLIC
void dsp_set_fft_file_status (fft_file_status)
FLAG fft_file_status;
{
}

/*-------------------------------------------------------------------------*
 *                         dsp_set_MaxCalibrationTime                      *
 *-------------------------------------------------------------------------*/
/* Passes max calibration time to mdsp.                                    */

PUBLIC
void dsp_set_MaxCalibrationTime (time)
double time;
{
}

/*-------------------------------------------------------------------------*
 *                       dsp_set_spectral_status                           *
 *-------------------------------------------------------------------------*/
/* Set spectral_enabled flag.                                              */

PUBLIC
void dsp_set_spectral_status (spectral_status)
FLAG spectral_status;
{
}

/*-------------------------------------------------------------------------*
 *                         dsp_spectral_detection                          *
 *-------------------------------------------------------------------------*/
/* Return flag for spectral_detection.                                     */

PUBLIC
FLAG dsp_spectral_detection ()
{
   return FALSE;
}

/*-------------------------------------------------------------------------*
 *                      dsp_spectral_detection_done                        *
 *-------------------------------------------------------------------------*/
/* Return flag for spectral recording done.                                */

PUBLIC
FLAG dsp_spectral_detection_done ()
{
  return FALSE;
}
```

```
/* FILE: mdspstub.h                                    (D. Tottingham  04/26/92)

This is an include file of defines, data structure definitions and
external declarations for the mdspstub module.

*/

#ifndef _MDSP_
#define _MDSP_

/************************************************************************
                                INCLUDES
************************************************************************/
#include "mconst.h"


/************************************************************************
                                DEFINES
************************************************************************/
#define DSP_CALIBRATION         1               /* fft signal classes */


/************************************************************************
                          EXTERNAL DECLARATIONS

These functions can be called from all modules that include this file.

************************************************************************/
PUBLIC void dsp_continue_band_recording ();
PUBLIC FLAG dsp_domain_check ();
PUBLIC FLAG dsp_get_dsp_status ();
PUBLIC unsigned int dsp_get_spectral_channel ();
PUBLIC char dsp_get_status ();
PUBLIC void dsp_initialize_params ();
PUBLIC void dsp_initialize ();
PUBLIC void dsp_set_band (float, float);
PUBLIC void dsp_set_bufs_to_avg (int);
PUBLIC void dsp_set_dsp_status (FLAG);
PUBLIC void dsp_set_fft_file_status (FLAG);
PUBLIC void dsp_set_MaxCalibrationTime (double);
PUBLIC void dsp_set_spectral_status (FLAG);
PUBLIC FLAG dsp_spectral_detection ();
PUBLIC FLAG dsp_spectral_detection_done ();

#endif
```

```
/* FILE: mdspstub.c                                (D. Tottingham  04/26/92)

This file contains stubs for functions normally found in mdsp.c.  They are
stubbed for several reasons: (1) this functionality is not supported in the
xdetect distribution, (2) this functionality is subject to change, and
(3) this functionality has not undergone significant testing which could
hamper other debugging efforts.

All functions have been written and compiled medium model.  The following
functions are included:

dsp_continue_band_recording ()  continue to record fft band info into .fft file
dsp_domain_check ()             analyze the current buffer for spectral lines
dsp_get_dsp_status ()           get dsp_enabled flag
dsp_get_spectral_channel ()     return spectral trigger channel
dsp_get_status ()               return char for spectral signal
dsp_initialize ()               initialize DSP package
dsp_initialize_params ()        initialize key parameters
dsp_set_band ()                 set band limits
dsp_set_bufs_to_avg ()          set buffers to average
dsp_set_dsp_status ()           set dsp_enabled flag
dsp_set_fft_file_status ()      set fft_file_enabled flag
dsp_set_MaxCalibrationTime ()   passes max calibration time to mdsp
dsp_set_spectral_status ()      set spectral_enabled flag
dsp_spectral_detection ()       return flag for spectral_detection
dsp_spectral_detection_done ()  return flag for spectral recording done

EXTERNAL FUNCTIONS CALLED:
   none

HISTORY:
   none


/*************************************************************************
                              INCLUDE FILES
*************************************************************************/
#include <stdio.h>

#include "mconst.h"


/*************************************************************************
                                GLOBALS
*************************************************************************/


/*-----------------------------------------------------------------------*
 *                        dsp_continue_band_recording                    *
 *-----------------------------------------------------------------------*/
/* Continue to record fft band info into .fft file                       */

PUBLIC
void dsp_continue_band_recording ()
{
}

/*-----------------------------------------------------------------------*
 *                          dsp_domain_check                             *
 *-----------------------------------------------------------------------*/
/* Analyze the current buffer for characteristic spectral lines and      */
/* return flag to signal main() that an interesting signal is occurring. */

PUBLIC
FLAG dsp_domain_check ()
{
   return FALSE;
}

/*-----------------------------------------------------------------------*
 *                          dsp_get_dsp_status                           *
 *-----------------------------------------------------------------------*/
/* Get dsp_enabled.                                                      */

PUBLIC
FLAG dsp_get_dsp_status ()
{
   return FALSE;
}

/*-----------------------------------------------------------------------*
 *                        dsp_get_spectral_channel                       *
 *-----------------------------------------------------------------------*/
/* Return spectral trigger channel.                                      */

PUBLIC
unsigned int dsp_get_spectral_channel ()
{
   return 0;
}

/*-----------------------------------------------------------------------*
 *                             dsp_get_status                            *
 *-----------------------------------------------------------------------*/
/* Return char for spectral signal.                                      */

PUBLIC
char dsp_get_status ()
{
   return FALSE;
}

/*-----------------------------------------------------------------------*
 *                        dsp_initialize_params                          *
 *-----------------------------------------------------------------------*/
/* Initialize key parameters.                                            */

PUBLIC
void dsp_initialize_params ()
{
}

/*-----------------------------------------------------------------------*
 *                          dsp_initialize                               *
 *-----------------------------------------------------------------------*/
/* Initialize arrays and variables used in DSP package.                  */

PUBLIC
void dsp_initialize()
{
}

/*-----------------------------------------------------------------------*
 *                             dsp_set_band                              *
 *-----------------------------------------------------------------------*/
/* Set band limits.                                                      */

PUBLIC
void dsp_set_band (low, high)
float low, high;
{
}

/*-----------------------------------------------------------------------*
 *                          dsp_set_bufs_to_avg                          *
 *-----------------------------------------------------------------------*/
/* Set buffers to average.                                               */

PUBLIC
void dsp_set_bufs_to_avg (number)
int number;
{
}
```

```
/*-------------------------------------------------------------------------*
 *                          dsp_set_dsp_status                             *
 *-------------------------------------------------------------------------*/
/* Set dsp_enabled flag.                                                   */

PUBLIC
void dsp_set_dsp_status (dsp_status)
FLAG dsp_status;
{
}

/*-------------------------------------------------------------------------*
 *                        dsp_set_fft_file_status                          *
 *-------------------------------------------------------------------------*/
/* Set fft_file_enabled flag.                                              */

PUBLIC
void dsp_set_fft_file_status (fft_file_status)
FLAG fft_file_status;
{
}

/*-------------------------------------------------------------------------*
 *                        dsp_set_MaxCalibrationTime                       *
 *-------------------------------------------------------------------------*/
/* Passes max calibration time to mdsp.                                    */

PUBLIC
void dsp_set_MaxCalibrationTime (time)
double time;
{
}

/*-------------------------------------------------------------------------*
 *                        dsp_set_spectral_status                          *
 *-------------------------------------------------------------------------*/
/* Set spectral_enabled flag.                                              */

PUBLIC
void dsp_set_spectral_status (spectral_status)
FLAG spectral_status;
{
}

/*-------------------------------------------------------------------------*
 *                        dsp_spectral_detection                           *
 *-------------------------------------------------------------------------*/
/* Return flag for spectral_detection.                                     */

PUBLIC
FLAG dsp_spectral_detection ()
{
   return FALSE;
}

/*-------------------------------------------------------------------------*
 *                      dsp_spectral_detection_done                        *
 *-------------------------------------------------------------------------*/
/* Return flag for spectral recording done.                                */

PUBLIC
FLAG dsp_spectral_detection_done ()
{
  return FALSE;
}
```

```
/* FILE: mdspstub.h                                (D. Tottingham  04/26/92)

This is an include file of defines, data structure definitions and
external declarations for the mdspstub module.

*/

#ifndef _MDSP_
#define _MDSP_

/*************************************************************************
                                INCLUDES

*************************************************************************/
#include "mconst.h"


/*************************************************************************
                            EXTERNAL DECLARATIONS

These functions can be called from all modules that include this file.

*************************************************************************/
PUBLIC void dsp_continue_band_recording ();
PUBLIC FLAG dsp_domain_check ();
PUBLIC FLAG dsp_get_dsp_status ();
PUBLIC unsigned int dsp_get_spectral_channel ();
PUBLIC char dsp_get_status ();
PUBLIC void dsp_initialize_params ();
PUBLIC void dsp_initialize ();
PUBLIC void dsp_set_band (float, float);
PUBLIC void dsp_set_bufs_to_avg (int);
PUBLIC void dsp_set_dsp_status (FLAG);
PUBLIC void dsp_set_fft_file_status (FLAG);
PUBLIC void dsp_set_MaxCalibrationTime (double);
PUBLIC void dsp_set_spectral_status (FLAG);
PUBLIC FLAG dsp_spectral_detection ();
PUBLIC FLAG dsp_spectral_detection_done ();

#endif
```

```
/* FILE: mdt28xx.c                                (D. Tottingham  04/26/92)

This is a collection of C functions that manage the DT28xx board for xdetect.
All functions have been written and compiled medium model.  The following
functions are included:

dt_buffer_full ()                 return buffer status
dt_dump_configuration ()          dump the DT2821 configuration
dt_get_atodinfo ()                return DT_28xx structure
dt_get_bankswitch_count ()        return bank switch count for this session
dt_get_buffer_size ()             get buffer size
dt_get_channel_size ()            get channel block size
dt_get_channel_gain ()            get channel gain
dt_get_clip_value ()              get clip value
dt_get_clock_source ()            get clock source
dt_get_data_type ()               get data type
dt_get_data_units ()              get data units
dt_get_dc_offset ()               get dc offset
dt_get_digitization_rate ()       get digitization rate
dt_get_input_range ()             get input range
dt_get_new_buffers ()             get new buffer(s) from DT2821
dt_get_next_buffer ()             get next buffer from mux queue
dt_get_number_of_ext_buffers ()   get number of external buffers
dt_get_scan_count ()              get scan count
dt_get_trigger_source ()          get trigger source
dt_initialize_ADC ()              initialize the DT2821
dt_initialize_buffers ()          initialize buffers
dt_initialize_mux ()              initialize multiplexer
dt_initialize_params()            initialize parameters
dt_set_channel ()                 put channel and gain in channel and gain lists
dt_set_ChannelBlocksize ()        set channel block size
dt_set_ChannelGain ()             set the channel gain
dt_set_ClockSource ()             set clock source
dt_set_DigitizationRate ()        set the digitization rate
dt_set_mux_type ()                set the multiplexer type
dt_set_TriggerSource ()           set trigger source
dt_start_ADC ()                   start continuous data acquisition
dt_stop_ADC ()                    stop continuous data acquisition

EXTERNAL FUNCTIONS CALLED:

er_abort ()                       display an error message then quit
h_increment_bankswitch_ctr ()     increment bank switch counter
o_write_logfile ()                write string to log file stream
suds_initialize ()                initialize a suds structure
suds_initialize_tag ()            initialize a suds structtag

HISTORY:
   none

*/



/*************************************************************************
                            INCLUDE FILES
*************************************************************************/
#include <conio.h>
#include <malloc.h>
#include <math.h>
#include <stdio.h>

#include <atldefs.h>                    /* Data Translation includes    */
#include "atlerrs.h"

#include "mconst.h"
#include "mdt28xx.h"
#include "merror.h"
#include "mlog.h"
#include "mqueue.h"
#include "mscrnhdr.h"
#include "mstation.h"
#include "msudsini.h"
#include "mutils.h"
#include "xdetect.h"


/*************************************************************************
                            DEFINES
*************************************************************************/
/* Multiplexer types */
#define NO_MUX                  0x00ff
#define SE_128x16_V2X           0x0000
#define SE_128x16_V1            0x0001
#define DI_64x4_V1              0x0008

/* Multiplexer commands */
#define NULL_COMMAND            0x0000
#define WRITE_NBANKS            0x0001
#define READ_MUXID              0x0002
#define READ_CURRENT_BANK       0x0003

/* Gain types */
#define NO_GAIN                 0
#define GAIN_1                  1
#define GAIN_1248               2
#define GAIN_500                4

/* DT28xx Constants */
#define BOARD_CLOCK_FREQUENCY   4e6
#define DIO_CONTROL             0x0006
#define DIO_DATA                0x000a
#define DT28XX_UNIT             1
#define HIGH_BYTE_ENABLE        0x0002
#define LOW_BYTE_ENABLE         0x0001
#define WORD_ENABLE             0x0003
#define WORD_DISABLE            0x0000

/*************************************************************************
                            GLOBALS
*************************************************************************/
PRIVATE struct {
        unsigned int mux_id;
        unsigned int mux_factor;
        char *mux_descriptor;
        int adc_device_id;
        FLAG diff_inputs;
} mux_configuration[] = {
        DI_64x4_V1,     16,     "DI_64x4_V1",           (DT2823         |
                                                         DT2827),       YES,

        SE_128x16_V1,   8,      "SE_128x16_V1",         (DT2821         |
                                                         DT2821_F_SE    |
                                                         DT2821_G_SE    |
                                                         DT2824_PGH     |
                                                         DT2824_PGL     |
                                                         DT2825),       NO,

        SE_128x16_V2X,  8,      "SE_128x16_V2X",        (DT2821         |
                                                         DT2821_F_SE    |
                                                         DT2821_G_SE    |
                                                         DT2824_PGH     |
                                                         DT2824_PGL     |
                                                         DT2825),       NO,

        NO_MUX,         1,      NULL,                   UNUSED,         NO
};

PRIVATE struct {
        int adc_device_id;
        unsigned int gain_type;
        unsigned int n_bits;
```

```
} dt28xx_info[] = {
        DT2821,         (GAIN_1 | GAIN_1248),           12,
        DT2821_F_SE,    (GAIN_1 | GAIN_1248),           12,
        DT2821_F_DI,    (GAIN_1 | GAIN_1248),           12,
        DT2821_G_SE,    (GAIN_1 | GAIN_1248),           12,
        DT2821_G_DI,    (GAIN_1 | GAIN_1248),           12,
        DT2823,         GAIN_1,                         16,
        DT2824_PGH,     (GAIN_1 | GAIN_1248),           12,
        DT2824_PGL,     (GAIN_1 | GAIN_500),            12,
        DT2825,         (GAIN_1 | GAIN_500),            12,
        DT2827,         GAIN_1,                         16,
        DT2828,         GAIN_1,                         12,
        UNUSED,         NO_GAIN,                        0
};

PRIVATE struct {
        unsigned int max_gain;
        unsigned int gain_type;
} max_gain_map[] = {
        1,              GAIN_1,
        8,              GAIN_1248,
        500,            GAIN_500,
        0,              NO_GAIN,
};

PRIVATE struct {
        unsigned int gain;
        unsigned int gain_type;
} gain_map[] = {
        1,              GAIN_1,
        2,              GAIN_1248,
        4,              GAIN_1248,
        8,              GAIN_1248,
        10,             GAIN_500,
        100,            GAIN_500,
        500,            GAIN_500,
        0,              NO_GAIN
};

PRIVATE DT_28XX adc_info, far * a = &adc_info;
PRIVATE AL_CONFIGURATION config, far * c = &config;
PRIVATE int status;
PRIVATE Q_QUEUE mux_queue;
PRIVATE Q_LINK * next_ptr;
PRIVATE double digitization_rate;
PRIVATE unsigned long buffer_size;
PRIVATE unsigned long channel_blocksize;
PRIVATE unsigned int scan_count;
PRIVATE unsigned int channel_gain;
PRIVATE unsigned int max_channel;
PRIVATE unsigned int bankswitch_count;
PRIVATE unsigned int mux_id;
PRIVATE FLAG mux_installed;


/*-------------------------------------------------------------------------*
 *                          get_dynamic_range                              *
 *-------------------------------------------------------------------------*/
/* Get the dynamic range from the dt28xx_info table.                       */

PRIVATE
FLAG get_dynamic_range (dr_ptr)
unsigned int *dr_ptr;
{
   int i;

   for (i = 0; dt28xx_info[i].adc_device_id != UNUSED; i++)
      if (dt28xx_info[i].adc_device_id == c->device_id) {
         *dr_ptr = dt28xx_info[i].n_bits;
         return TRUE;
      }

   return FALSE;
}

/*-------------------------------------------------------------------------*
 *                             get_max_gain                                *
 *-------------------------------------------------------------------------*/
/* Get the maximum gain possible for given device.                         */

PRIVATE
FLAG get_max_gain (mg_ptr)
unsigned int *mg_ptr;
{
   unsigned int max_gain_type;
   int i, j;

   for (i = 0; dt28xx_info[i].adc_device_id != UNUSED &&
               dt28xx_info[i].adc_device_id != c->device_id; i++);

   if (dt28xx_info[i].adc_device_id != UNUSED) {
      max_gain_type = NO_GAIN;
      if (dt28xx_info[i].gain_type & GAIN_500)
         max_gain_type = GAIN_500;
      else if (dt28xx_info[i].gain_type & GAIN_1248)
         max_gain_type = GAIN_1248;
      else if (dt28xx_info[i].gain_type & GAIN_1)
         max_gain_type = GAIN_1;
      for (j = 0; max_gain_map[j].gain_type != max_gain_type; j++);
      *mg_ptr = max_gain_map[j].max_gain;
      return TRUE;
   }

   return FALSE;
}

/*-------------------------------------------------------------------------*
 *                          get_mux_descriptor                             *
 *-------------------------------------------------------------------------*/
/* Get the multiplexer descriptor for the given mux_id.                    */

PRIVATE
FLAG get_mux_descriptor (mux_id, md_ptr)
unsigned int mux_id;
char ** md_ptr;
{
   int i;

   for (i = 0; mux_configuration[i].mux_id != NO_MUX; i++)
      if (mux_configuration[i].mux_id == mux_id) {
         *md_ptr = mux_configuration[i].mux_descriptor;
         return TRUE;
      }
   *md_ptr = NULL;
   return FALSE;
}

/*-------------------------------------------------------------------------*
 *                           get_mux_factor                                *
 *-------------------------------------------------------------------------*/
/* Get the multiplex factor for the given mux_id.                          */

PRIVATE
FLAG get_mux_factor (mux_id, mf_ptr)
unsigned int mux_id;
unsigned int *mf_ptr;
{
   int i;

   for (i = 0; mux_configuration[i].mux_id != NO_MUX; i++)
      if (mux_configuration[i].mux_id == mux_id) {
         *mf_ptr = mux_configuration[i].mux_factor;
         return TRUE;
      }
   return FALSE;
}
```

```
/*-------------------------------------------------------------------------*
 *                          compute_dc_offset                              *
 *-------------------------------------------------------------------------*/
/* Compute the dc offset.                                                  */

PRIVATE
FLAG compute_dc_offset (dc_ptr)
unsigned int *dc_ptr;
{
   unsigned int i, dr;

   /* Mid-point is a function of the encoding scheme. */
   switch (c->device_flags.encoding) {
      case OFFSET_BINARY:
         if (get_dynamic_range (&dr)) {
            for (i = 1, *dc_ptr = 1; i < dr; i++)
               *dc_ptr <<= 1;
            return TRUE;
         }
         break;

      case STRAIGHT_BINARY:
      case TWOS_COMPLEMENT:
         *dc_ptr = 0;
         return TRUE;
         break;
   }

   return FALSE;
}

/*-------------------------------------------------------------------------*
 *                          send_mux_command                               *
 *-------------------------------------------------------------------------*/
/* Send the given command and data to multiplexer.                         */

/* Command word, upper nibble definitions */
#define LATCH_2827      0x0080
#define CLEAR_BIT       0x0040
#define LATCH_2821      0x0020

PRIVATE
void send_mux_command (command, data)
unsigned int command, data;
{
   unsigned int command_word;

   command_word = command | CLEAR_BIT | (data << 8);

   /* Send the command, latch = 0 */
   outpw ((c->base_address + DIO_DATA), command_word);

   /* Send the command, latch = 1 */
   outpw ((c->base_address + DIO_DATA), command_word | LATCH_2827);

   /* Send the command, latch = 0
   outpw ((c->base_address + DIO_DATA), command_word);
   */
}

/*-------------------------------------------------------------------------*
 *                          allocate_int_buffers                           *
 *-------------------------------------------------------------------------*/
/* Allocate "internal" buffer queue for ATLAB.  All internal buffers
   reside in the base 640K of user RAM.                                    */

PRIVATE
void allocate_int_buffers ()
{
   static unsigned int far int_buffer_number;
   unsigned int actual_blocksize, actual_buffersize, temp;
   double seconds_in_buffer;
   unsigned long offset;
   Q_TYPE type;
   Q_BUFFER far * mux;
   char * netwname;

   actual_buffersize = (buffer_size <= MAX_BUFFER_SIZE) ? buffer_size : MAX_BUFFER_S
   actual_blocksize = actual_buffersize / scan_count;
   seconds_in_buffer = actual_blocksize / digitization_rate;

   q_initialize (&mux_queue);
   netwname = st_get_netwname ();
   for (offset = 0; offset < buffer_size; offset += MAX_BUFFER_SIZE) {
      mux = (Q_BUFFER far *) _fmalloc (sizeof (Q_BUFFER));
      if (mux == NULL) er_abort (DT_NO_STORAGE);

      mux->data = (int far *) _fmalloc ( actual_buffersize * sizeof(unsigned));
      if (mux->data == NULL) er_abort (DT_NO_STORAGE);

      AL_DECLARE_BUFFER ( &int_buffer_number, mux->data, actual_buffersize );

      if (get_dynamic_range (&temp))
         mux->dynamic_range = temp;

      mux->buffer_number = int_buffer_number;
      mux->buffer_size = actual_buffersize;
      mux->seconds_in_buffer = seconds_in_buffer;

      suds_initialize (MUXDATA, &mux->info);
      suds_initialize_tag (MUXDATA, &mux->structtag);

      mux->structtag.len_data = actual_buffersize * sizeof(unsigned);
      u_strncpy (mux->info.netname, ((char far *)netwname), 4);

      if (compute_dc_offset (&temp))
         mux->info.dc_offset = temp;

      mux->info.blocksize = actual_blocksize;
      mux->info.numchans = scan_count;
      mux->info.dig_rate = digitization_rate;
      mux->info.typedata = 's';
      type.buffer = mux;
      q_enqueue (&mux_queue, type);
   }
}

/*-------------------------------------------------------------------------*
 *                          allocate_ext_buffers                           *
 *-------------------------------------------------------------------------*/
/* Allocate an "external" buffer for ATLAB.  All external buffers reside
   ABOVE the base 640K of user RAM.                                        */

PRIVATE
void allocate_ext_buffers ()
{
   static unsigned far ext_buffer_number;

   a->info.extended_bufs = 0;
   while (AL_ALLOCATE_XM_BUFFER (&ext_buffer_number, buffer_size) == ALE_NORMAL) {
      AL_LINK_BUFFER ( ext_buffer_number );
      a->info.extended_bufs++;
   }
}

/*-------------------------------------------------------------------------*
 *                          clear_mux                                      *
 *-------------------------------------------------------------------------*/
/* Clear the multiplexer.                                                  */

PRIVATE
void clear_mux ()
{
   unsigned int command_word;
```

```
   outpw ((c->base_address + DIO_CONTROL), LOW_BYTE_ENABLE);

   command_word = (NULL_COMMAND & 0x0f) | LATCH_2821 | LATCH_2827;

   /* Send the NULL_COMMAND, clear = 0 */
   outpw ((c->base_address + DIO_DATA), command_word);

   /* Send the command, clear = 1 */
   outpw ((c->base_address + DIO_DATA), command_word | CLEAR_BIT);
}

/*-------------------------------------------------------------------------*
 *                          devflags_to_bitmap                             *
 *-------------------------------------------------------------------------*/
/* Convert DEV_FLAGS structure to compact 16-bit bit map.                  */

PRIVATE
unsigned short devflags_to_bitmap (devflags)
DEV_FLAGS devflags;
{
   unsigned short bitmap;

   bitmap = ((unsigned short)devflags.DMA_mode) << 14;
   bitmap += ((unsigned short)devflags.SE_DI) << 13;
   bitmap += ((unsigned short)devflags.unipolar) << 12;
   bitmap += ((unsigned short)devflags.encoding) << 10;
   bitmap += ((unsigned short)devflags.sign_extended) << 9;

   return (bitmap);
}

/*-------------------------------------------------------------------------*
 *                          read_current_bank                              *
 *-------------------------------------------------------------------------*/
/* Read current bank from multiplexer.                                     */

PRIVATE
unsigned int read_current_bank ()
{
   unsigned current_bank;

   outpw ((c->base_address + DIO_CONTROL), LOW_BYTE_ENABLE);
   send_mux_command (READ_CURRENT_BANK | LATCH_2821, 0);
   current_bank = inpw (c->base_address + DIO_DATA);
   return ((current_bank >> 8) & 0x00ff);
}

/*-------------------------------------------------------------------------*
 *                             read_muxid                                  *
 *-------------------------------------------------------------------------*/
/* Read multiplexer id from multiplexer.                                   */

PRIVATE
unsigned int read_muxid()
{
   unsigned int muxid;

   outpw ((c->base_address + DIO_CONTROL), LOW_BYTE_ENABLE);
   send_mux_command (READ_MUXID | LATCH_2821, 0);
   muxid = inpw (c->base_address + DIO_DATA);
   return ((muxid >> 8) & 0x00ff);
}

/*-------------------------------------------------------------------------*
 *                            write_nbanks                                 *
 *-------------------------------------------------------------------------*/
/* Send number of banks to multiplexer.                                    */

PRIVATE
void write_nbanks (nbanks)
unsigned int nbanks;
{
   unsigned int command_word;

   outpw ((c->base_address + DIO_CONTROL), WORD_ENABLE);

   send_mux_command (WRITE_NBANKS, nbanks);
   send_mux_command (WRITE_NBANKS | LATCH_2821, nbanks);
   send_mux_command (NULL_COMMAND | LATCH_2821, nbanks);
}

/*-------------------------------------------------------------------------*
 *                              valid_gain                                 *
 *-------------------------------------------------------------------------*/
/* Determines whether gain is valid in this system.                        */

PRIVATE
FLAG valid_gain (gain)
unsigned int gain;
{
   int i, j;
   unsigned int gain_type;

   if (gain == 0)
      return TRUE;

   /* Translate gain into gain type */
   for (i = 0; gain_map[i].gain_type != NO_GAIN &&
               gain_map[i].gain != gain; i++);

   /* Look for device and gain match in dt28xx_info table */
   if (gain_map[i].gain_type != NO_GAIN) {
      for (j = 0; dt28xx_info[j].adc_device_id != UNUSED; j++)
         if (c->device_id == dt28xx_info[j].adc_device_id) {
            if (gain_map[i].gain_type & dt28xx_info[j].gain_type)
               return TRUE;
            else return FALSE;
         }
   } else return FALSE;
}

/*-------------------------------------------------------------------------*
 *                           valid_multiplexer                             *
 *-------------------------------------------------------------------------*/
/* Determines whether multiplexer is valid in this system.                 */

PRIVATE
FLAG valid_multiplexer (mux_id)
unsigned int mux_id;
{
   int i;

   for (i = 0; mux_configuration[i].mux_id != NO_MUX; i++)
      if (mux_id == mux_configuration[i].mux_id &&
          (c->device_id & mux_configuration[i].edc_device_id)) {
         if ((c->device_flags.SE_DI && mux_configuration[i].diff_inputs) ||
             (!c->device_flags.SE_DI && !mux_configuration[i].diff_inputs))
            return TRUE;
         else return FALSE;
      }

   return FALSE;
}

/*-------------------------------------------------------------------------*
 *                             valid_mux_id                                *
 *-------------------------------------------------------------------------*/
/* Determines whether mux id is valid.                                     */

PRIVATE
FLAG valid_mux_id (mux_id)
unsigned int mux_id;
{
   int i;

   for (i = 0; mux_configuration[i].mux_id != NO_MUX; i++)
```

```
      if (mux_id == mux_configuration[i].mux_id)
         return TRUE;

   return FALSE;
}

/*-------------------------------------------------------------------------*
 *                           dt_buffer_full                                *
 *-------------------------------------------------------------------------*/
/* Return buffer status.                                                   */

PUBLIC
FLAG dt_buffer_full ()
{
   static unsigned int far buffer_number, far count_remaining;
   int errnum;

   errnum = AL_TEST_BUFFER (&buffer_number, &count_remaining);
   if (errnum == ALE_NOTINPROG)
      er_abort (DT_IO_NOTINPROG);

   return ((count_remaining) ? FALSE : TRUE);
}

/*-------------------------------------------------------------------------*
 *                         dt_dump_configuration                           *
 *-------------------------------------------------------------------------*/
/* Dump the DT2821 configuration to s stream.                              */

PUBLIC
void dt_dump_configuration (stream)
FILE * stream;
{
   int i;

   if ( status == ALE_NORMAL ) {
      if (c->device_id == UNUSED)
         fprintf(stream, "No device defined.\r\n");
      else {
         fprintf (stream, "Device id is ");
         switch (c->device_id) {
            case DT2828:
                  fprintf(stream, "DT2828");
                  break;
            case DT2827:
                  fprintf(stream, "DT2827");
                  break;
            case DT2821_F_SE:
                  fprintf(stream, "DT2821-F-16SE");
                  break;
            case DT2821_F_DI:
                  fprintf(stream, "DT2821-F-8DI");
                  break;
            case DT2821:
                  fprintf(stream, "DT2821");
                  break;
            case DT2821_G_DI:
                  fprintf(stream, "DT2821_G_DI");
                  break;
            case DT2821_G_SE:
                  fprintf(stream, "DT2821_G_SE");
                  break;
            case DT2823:
                  fprintf(stream, "DT2823");
                  break;
            case DT2825:
                  fprintf(stream, "DT2825");
                  break;
            case DT2824_PGH:
                  fprintf(stream, "DT2824_PGH");
                  break;
            case DT2824_PGL:
                  fprintf(stream, "DT2824_PGL");
                  break;
            case DT2829:
                  fprintf(stream, "DT2829");
                  break;
         }
         fprintf(stream, ", ");

         switch (c->device_flags.DMA_mode) {
            case PROGRAMMED_IO:
                  fprintf(stream, "programmed I/O");
                  break;
            case SINGLE_CHANNEL:
                  fprintf(stream, "single channel DMA");
                  break;
            case DUAL_CHANNEL:
                  fprintf(stream, "dual channel DMA");
                  break;
         }
         fprintf(stream, ", ");

         if (c->device_flags.SE_DI)
            fprintf(stream, "DI, ");
         else
            fprintf(stream, "SE, ");

         if (c->device_flags.unipolar)
            fprintf(stream, "unipolar, ");
         else
            fprintf(stream, "bipolar, ");

         switch (c->device_flags.encoding) {
            case OFFSET_BINARY:
                  fprintf(stream, "offset binary encoding");
                  break;
            case STRAIGHT_BINARY:
                  fprintf(stream, "straight binary encoding");
                  break;
            case TWOS_COMPLEMENT:
                  fprintf(stream, "two's complement encoding");
                  if (c->device_flags.sign_extended)
                     fprintf(stream, ", sign extended");
                  break;
         }
      fprintf(stream, "\r\n");
      fprintf(stream, "Base I/O address is %4x hex.\r\n",c->base_address);
/*
      fprintf(stream, "%d A/D channels.\r\n",c->channel_count);
*/
      }
   }
}

/*-------------------------------------------------------------------------*
 *                           dt_get_atodinfo                               *
 *-------------------------------------------------------------------------*/
/* Return DT_28xx structure.                                               */

PUBLIC
DT_28XX far * dt_get_atodinfo ()
{
   return (a);
}


/*-------------------------------------------------------------------------*
 *                        dt_get_bankswitch_count                          *
 *-------------------------------------------------------------------------*/
/* Return bank switch count for this session.                              */

PUBLIC
unsigned int dt_get_bankswitch_count ()
{
   return (bankswitch_count);
```

```
}

/*-------------------------------------------------------------------------*
 *                          dt_get_buffer_size                             *
 *-------------------------------------------------------------------------*/
/* Get buffer size.                                                        */

PUBLIC
unsigned long dt_get_buffer_size ()
{
   return (buffer_size);
}

/*-------------------------------------------------------------------------*
 *                         dt_get_channel_gain                             *
 *-------------------------------------------------------------------------*/
/* Get channel gain.                                                       */

PUBLIC
unsigned int dt_get_channel_gain (channel)
unsigned int channel;
{
   return ((channel < c->channel_count) ? a->gain_list[channel] : channel_gain);
}

/*-------------------------------------------------------------------------*
 *                         dt_get_channel_size                             *
 *-------------------------------------------------------------------------*/
/* Get channel block size.                                                 */

PUBLIC
unsigned long dt_get_channel_size ()
{
   return (channel_blocksize);
}

/*-------------------------------------------------------------------------*
 *                          dt_get_clip_value                              *
 *-------------------------------------------------------------------------*/
/* Get the clip value.                                                     */

PUBLIC
unsigned long dt_get_clip_value ()
{
   unsigned int i, dr;
   unsigned long cv;

   cv = 1;
   if (get_dynamic_range (&dr))
      for (i = 0; i < dr; i++)
         cv <<= 1;

   return cv;
}

/*-------------------------------------------------------------------------*
 *                         dt_get_clock_source                             *
 *-------------------------------------------------------------------------*/
/* Get clock source.                                                       */

PUBLIC
char dt_get_clock_source ()
{
   return (a->info.timing_source);
}

/*-------------------------------------------------------------------------*
 *                          dt_get_data_type                               *
 *-------------------------------------------------------------------------*/
/* Get the data type.                                                      */

PUBLIC
char dt_get_data_type ()
{
   return 's';
}

/*-------------------------------------------------------------------------*
 *                          dt_get_data_units                              *
 *-------------------------------------------------------------------------*/
/* Get the data units.                                                     */

PUBLIC
char dt_get_data_units ()
{
   return 'd';
}

/*-------------------------------------------------------------------------*
 *                           dt_get_dc_offset                              *
 *-------------------------------------------------------------------------*/
/* Get the dc offset.                                                      */

PUBLIC
unsigned int dt_get_dc_offset ()
{
   unsigned int offset;

   compute_dc_offset (&offset);
   return offset;
}

/*-------------------------------------------------------------------------*
 *                       dt_get_digitization_rate                          *
 *-------------------------------------------------------------------------*/
/* Get digitization rate.                                                  */

PUBLIC
double dt_get_digitization_rate ()
{
   return (digitization_rate);
}

/*-------------------------------------------------------------------------*
 *                          dt_get_input_range                             *
 *-------------------------------------------------------------------------*/
/* Get the input range.                                                    */

PUBLIC
unsigned int dt_get_input_range ()
{
   if (c->device_flags.unipolar)
      return 10;
   else
      return 20;
}

/*-------------------------------------------------------------------------*
 *                          dt_get_new_buffers                             *
 *-------------------------------------------------------------------------*/
/* Wait for buffer to complete.  Copy external buffer into user
   accessible mux queue.                                                   */

PUBLIC
void dt_get_new_buffers ()
{
   static unsigned int far released_buffer, far unit;
   unsigned long offset;
   FLAG bank_switched;
   int errnum;
   Q_LINK * head;

   /* Wait for a buffer to complete */
   offset = 0;
   unit = DT28XX_UNIT;
```

```
   bank_switched = FALSE;
   errnum = AL_RELEASE_BUFFER (&unit, &released_buffer);
   switch (errnum) {
      case ALE_NORMAL:
      case ALE_BANKSWITCH:
         /* Make a scene if we have a bank switch */
         if (mux_installed && mux_id != SE_128x16_V1 &&
             errnum == ALE_BANKSWITCH) {
            o_write_logfile ("WARNING: Bank switch occured on multiplexer\n");
            if (Debug_enabled)
               printf ("WARNING: Bank switch occured on multiplexer\n");
            bankswitch_count++;
            h_increment_bankswitch_ctr();
            bank_switched = TRUE;
         }

         head = next_ptr = mux_queue.head;
         while (head != NULL) {
            AL_COPY_BUFFER (released_buffer, head->type.buffer->buffer_number, offset,
                            head->type.buffer->buffer_size);
            offset += head->type.buffer->buffer_size;
            head->type.buffer->bank_switched = bank_switched;
            head = head->next;
         }
         AL_RETURN_BUFFER (released_buffer);

         break;

      case ALE_INPTMO:
         er_abort (DT_TIMEOUT);
         break;

      case ALE_BUFFER:
      default:
         er_abort (DT_OVERFLOW);
         break;
      }
}

/*-------------------------------------------------------------------------*
 *                          dt_get_next_buffer                             *
 *-------------------------------------------------------------------------*/
/* Get next buffer from mux queue.                                         */

PUBLIC
Q_BUFFER far * dt_get_next_buffer ()
{
   Q_LINK * this_ptr;

   this_ptr = next_ptr;
   if (next_ptr != NULL)
      next_ptr = next_ptr->next;

   return (this_ptr->type.buffer);
}

/*-------------------------------------------------------------------------*
 *                       dt_get_number_of_ext_buffers                      *
 *-------------------------------------------------------------------------*/
/* Get number of external buffers.                                         */

PUBLIC
int dt_get_number_of_ext_buffers ()
{
   return (a->info.extended_bufs);
}

/*-------------------------------------------------------------------------*
 *                            dt_get_scan_count                            *
 *-------------------------------------------------------------------------*/
/* Get scan count.                                                         */

PUBLIC
int dt_get_scan_count ()
{
   return (scan_count);
}

/*-------------------------------------------------------------------------*
 *                          dt_get_trigger_source                          *
 *-------------------------------------------------------------------------*/
/* Get trigger source.                                                     */

PUBLIC
char dt_get_trigger_source ()
{
   return (a->info.trigger_source);
}

/*-------------------------------------------------------------------------*
 *                          dt_initialize_ADC                              *
 *-------------------------------------------------------------------------*/
/* Initialize the DT28xx for ADC.                                          */

PUBLIC
void dt_initialize_ADC ()
{
   static long far number_of_ticks;
   unsigned int actual_scancount;
   char out_str[80], *mx_descript;
   int errnum;

   /* Do some error checking */
   if (!scan_count)
      er_abort (DT_NO_STATIONS);
   else if (! u_ispow2( (unsigned long)scan_count))
      er_abort (DT_INVALID_SCANCOUNT);

   actual_scancount = (scan_count > c->channel_count) ? c->channel_count :
                      scan_count;

   /* Reset the device */
   AL_RESET ();

   /* Give ATLAB the channel and gain lists */
   AL_SETUP_ADC ( ((a->info.trigger_source == 'i') ? INTERNAL_TRIGGER : EXTERNAL_TRI
                  ((a->info.timing_source == 'i') ? INTERNAL_CLOCK : EXTERNAL_CLOCK)
                  actual_scancount, a->channel_list, a->gain_list);

   /* Compute and send number of banks to multiplexer */
   if (mux_installed) {
      switch (mux_id) {
         case SE_128x16_V1:
            /* Set the external mux bits */
            a->info.external_mux = scan_count / c->channel_count;
            outpw ((a->info.base_address + DIO_DATA), a->info.external_mux);
            outpw ((a->info.base_address + DIO_CONTROL), LOW_BYTE_ENABLE);
            break;

         default:
            a->info.external_mux = (scan_count - 1) / c->channel_count;
            write_nbanks (a->info.external_mux);
      }
   }

   /* Send a description of the current mux to the screen and to the log file */
   if (get_mux_descriptor (mux_id, &mx_descript)) {
      sprintf (out_str, "%s multiplexer installed.\n", mx_descript);
      o_write_logfile (out_str);
      printf ("%s", out_str);
   } else {
      sprintf (out_str, "No multiplexer installed.\n");
      o_write_logfile (out_str);
      printf ("%s", out_str);
   }
```

```
   /* Write the number of channels being digitized to screen and to log file */
   sprintf (out_str, "%3d channels defined\n", scan_count);
   o_write_logfile (out_str);
   printf ("%s", out_str);

   /* Set the sampling rate */
   number_of_ticks = (long) (BOARD_CLOCK_FREQUENCY /
                     (digitization_rate * scan_count));
   if ((errnum = AL_SET_CLOCK (number_of_ticks)) != ALE_NORMAL) {
      if (errnum == ALE_SMALLP) er_abort (DT_FREQTOOHIGH);
      else er_abort (DT_FREQTOOLOW);
   }

   if (a->info.timing_source == 'i') {
      AL_RETURN_CLOCK (&number_of_ticks);
      digitization_rate = (double)(BOARD_CLOCK_FREQUENCY) / number_of_ticks /
                          scan_count;
   }

   /* Set the timeout rate */
   AL_SET_TIMEOUT ( TIME_OUT * (unsigned int) (((double)channel_blocksize) /
                    digitization_rate));
}

/*-------------------------------------------------------------------------*
 *                          dt_initialize_buffers                          *
 *-------------------------------------------------------------------------*/
/* Initialize buffers.                                                     */

PUBLIC
void dt_initialize_buffers ()
{
   buffer_size = channel_blocksize * scan_count;
   buffer_size = (! buffer_size) ? SEGMENT_SIZE : buffer_size;
   channel_blocksize = buffer_size / scan_count;

   /* Do some error checking first */
   if (! u_ispow2( channel_blocksize))
      er_abort (DT_INVALID_CHANNELSIZE);
   else if (buffer_size > SEGMENT_SIZE)
      er_abort (DT_INVALID_BUFFERSIZE);

   allocate_int_buffers ();
   allocate_ext_buffers ();
}


/*-------------------------------------------------------------------------*
 *                            dt_initialize_mux                            *
 *-------------------------------------------------------------------------*/
/* Initialize multiplexer.                                                 */

PUBLIC
void dt_initialize_mux()
{
   mux_installed = FALSE;
}

/*-------------------------------------------------------------------------*
 *                          dt_initialize_params                           *
 *-------------------------------------------------------------------------*/
/* Initialize parameters.                                                  */

PUBLIC
void dt_initialize_params()
{
   unsigned int mux_factor, max_gain;

   /* Set error processing */
   AL_SET_ERROR_CONTROL_WORD (Debug_enabled);

   /* Open a channel to the ATLAB device driver */
   if (AL_INITIALIZE () != ALE_NORMAL) er_abort (DT_NO_ATLDRV);

   /* Address board DT28XX_UNIT, the first unit */
   if (AL_SELECT_BOARD ( DT28XX_UNIT ) != ALE_NORMAL) er_abort (DT_NO_BOARD);

   /* Get the current configuration */
   status = AL_GET_CONFIGURATION (c);

   /* Initialize the multiplexer */
   bankswitch_count = 0;
   clear_mux();
   if (! mux_installed) {

      /* Is there a mux installed? Get multiplexer id. */
      if ((mux_id = read_muxid()) != NO_MUX)
         mux_installed = TRUE;
      else mux_installed = FALSE;

      /* Multiplexer is installed. Is it a valid mux type? */
      if (mux_installed && !valid_multiplexer (mux_id))
         er_abort (DT_INVALID_MUX);
   }

   /* Initialize max_channel */
   max_channel = c->channel_count;
   if (mux_installed && get_mux_factor (mux_id, &mux_factor))
      max_channel *= mux_factor;

   /* Set channel gain based on max gain and channel gain default */
   channel_gain = CHANNEL_GAIN;
   if (!valid_gain (channel_gain)) {
      if (get_max_gain (&max_gain))
         channel_gain = max_gain;
      else er_abort (DT_INVALID_GAIN);
   }

   digitization_rate = DIGITIZATION_RATE;
   channel_blocksize = 0;
   scan_count = 0;

   suds_initialize (ATODINFO, &a->info);
   suds_initialize_tag (ATODINFO, &a->structtag);

   if (status == ALE_NORMAL) {
      a->info.base_address = c->base_address;
      a->info.device_id = c->device_id;
      a->info.device_flags = devflags_to_bitmap (c->device_flags);
   }
   a->info.trigger_source = 'i';
   a->info.timing_source = 'i';
   a->info.external_mux = 0;

   a->channel_list = (int far *) _fmalloc (c->channel_count * sizeof(int));
   if (a->channel_list == NULL) er_abort (DT_NO_STORAGE);

   a->gain_list = (int far *) _fmalloc (c->channel_count * sizeof(int));
   if (a->gain_list == NULL) er_abort (DT_NO_STORAGE);
}

/*-------------------------------------------------------------------------*
 *                             dt_set_channel                              *
 *-------------------------------------------------------------------------*/
/* Put channel number in channel list and gain number in gain list.        */

PUBLIC
FLAG dt_set_channel (channel, gain)
unsigned int channel;
unsigned int gain;
{
   char out_str[80];

   if (channel >= max_channel) {
      sprintf (out_str, "WARNING: Invalid channel %3d ignored.\n", channel);
      o_write_logfile (out_str);
```

```
      printf ("%s", out_str);
      return FALSE;
   }

   if (channel != scan_count)
      er_abort (DT_INVALID_CHANNEL);

   if (! mux_installed) {
      if (!valid_gain (gain))
         er_abort (DT_INVALID_GAIN);

      a->channel_list[scan_count] = channel;
      a->gain_list[scan_count] = (gain == 0) ? channel_gain : gain;
   } else if (scan_count < c->channel_count) {
      a->channel_list[scan_count] = channel;
      a->gain_list[scan_count] = channel_gain;
   }
   scan_count++;
   return TRUE;
}

/*-------------------------------------------------------------------------*
 *                        dt_set_ChannelBlocksize                          *
 *-------------------------------------------------------------------------*/
/* Set the channel block size.                                             */

PUBLIC
void dt_set_ChannelBlocksize (blocksize)
unsigned long blocksize;
{
   channel_blocksize = blocksize;
}

/*-------------------------------------------------------------------------*
 *                          dt_set_ChannelGain                             *
 *-------------------------------------------------------------------------*/
/* Set the channel gain.                                                   */

PUBLIC
void dt_set_ChannelGain (gain)
unsigned int gain;
{
   int i;

   if (!valid_gain (gain))
      er_abort (DT_INVALID_GAIN);
   channel_gain = gain;
   for (i = 0; i < scan_count && i < c->channel_count; i++)
      a->gain_list[i] = gain;
}

/*-------------------------------------------------------------------------*
 *                          dt_set_ClockSource                             *
 *-------------------------------------------------------------------------*/
/* Set clock source.                                                       */

PUBLIC
void dt_set_ClockSource (timing_source)
int timing_source;
{
   a->info.timing_source = (timing_source == INTERNAL_CLOCK) ? 'i' : 'e';
}

/*-------------------------------------------------------------------------*
 *                        dt_set_DigitizationRate                          *
 *-------------------------------------------------------------------------*/
/* Set the digitization rate.                                              */

PUBLIC
void dt_set_DigitizationRate (rate)
double rate;
{
   digitization_rate = rate;
}

/*-------------------------------------------------------------------------*
 *                           dt_set_mux_type                               *
 *-------------------------------------------------------------------------*/
/* Set the multiplexer type.                                               */

PUBLIC
void dt_set_mux_type (type)
unsigned int type;
{
   mux_installed = TRUE;
   mux_id = type;
   if (mux_id == NO_MUX)
      mux_id = SE_128x16_V1;

   if (!valid_mux_id (mux_id))
      er_abort (DT_INVALID_MUX);
}

/*-------------------------------------------------------------------------*
 *                          dt_set_TriggerSource                           *
 *-------------------------------------------------------------------------*/
/* Set trigger source.                                                     */

PUBLIC
void dt_set_TriggerSource (trigger_source)
int trigger_source;
{
   a->info.trigger_source = (trigger_source == INTERNAL_TRIGGER) ? 'i' : 'e';
}

/*-------------------------------------------------------------------------*
 *                             dt_start_ADC                                *
 *-------------------------------------------------------------------------*/
/* Start continuous data acquisition.                                      */

PUBLIC
void dt_start_ADC ()
{
   /* Clear the multiplexer */
   if (mux_installed && mux_id != SE_128x16_V1)
      clear_mux();

   AL_CONTINUOUS_ADC ();
}

/*-------------------------------------------------------------------------*
 *                             dt_stop_ADC                                 *
 *-------------------------------------------------------------------------*/
/* Stop continuous data acquisition.  Reset the DT28xx.                    */

PUBLIC
void dt_stop_ADC ()
{
   AL_STOP ();
   AL_RESET ();
   AL_TERMINATE ();
}
```

```
/* FILE: mdt28xx.h                                (D. Tottingham  04/26/92)

This is an include file of the defines, data structure definitions and
external data declarations for using the mdt28xx module.

*/

#ifndef _MDT28XX_
#define _MDT28XX_

/*************************************************************************
                              INCLUDES

*************************************************************************/
#include "mconst.h"
#include "mqueue.h"


/*************************************************************************
                              DEFINES

*************************************************************************/
/* Defaults */
#define CHANNEL_GAIN            4
#define CLOCK_SOURCE            INTERNAL_CLOCK
#define DIGITIZATION_RATE       100.0           /* Samples/sec/channel */
#define TIME_OUT                10
#define TRIGGER_SOURCE          INTERNAL_TRIGGER

/* Buffer Constants */
#define MAX_BUFFER_SIZE         16384L


/*************************************************************************
                      STRUCTURE DEFINITIONS

*************************************************************************/
typedef struct {
        int far * channel_list;
        int far * gain_list;
        SUDS_STRUCTTAG structtag;
        SUDS_ATODINFO  info;
} DT_28XX;


/*************************************************************************
                      EXTERNAL DECLARATIONS

These functions can be called from all modules that include this file.

*************************************************************************/
PUBLIC FLAG dt_buffer_full ();
PUBLIC void dt_dump_configuration (FILE *);
PUBLIC DT_28XX far * dt_get_atodinfo ();
PUBLIC unsigned int dt_get_bankswitch_count ();
PUBLIC unsigned long dt_get_buffer_size ();
PUBLIC unsigned long dt_get_channel_size ();
PUBLIC unsigned int dt_get_channel_gain (unsigned int);
PUBLIC unsigned long dt_get_clip_value ();
PUBLIC char dt_get_clock_source ();
PUBLIC char dt_get_data_type ();
PUBLIC char dt_get_data_units ();
PUBLIC unsigned int dt_get_dc_offset ();
PUBLIC double dt_get_digitization_rate ();
PUBLIC unsigned int dt_get_input_range ();
PUBLIC void dt_get_new_buffers ();
PUBLIC Q_BUFFER far * dt_get_next_buffer ();
PUBLIC int dt_get_number_of_ext_buffers ();
PUBLIC int dt_get_scan_count ();
PUBLIC char dt_get_trigger_source ();
PUBLIC void dt_initialize_buffers ();
PUBLIC void dt_initialize_params();
PUBLIC FLAG dt_set_channel (unsigned int, unsigned int);
PUBLIC void dt_set_ChannelBlocksize (unsigned long);
PUBLIC void dt_set_ChannelGain (unsigned int);
PUBLIC void dt_set_ClockSource (int);
PUBLIC void dt_set_DigitizationRate (double);
PUBLIC void dt_set_mux_type (unsigned int);
PUBLIC void dt_set_TriggerSource (int);
PUBLIC void dt_start_ADC ();
PUBLIC void dt_stop_ADC ();

#endif
```

```
/* FILE: merror.c                                (D. Tottingham  04/26/92)

This is a collection of C functions that manage the error messages for xdetect.
All functions have been written and compiled medium model.  The following
functions are included:

er_abort ()                     display an error message then quit

EXTERNAL FUNCTIONS CALLED:

dm_reset ()                     reset this module and release all storage
dt_stop_ADC ()                  stop continuous data acquisition
o_write_logfile ()              write string to log file stream
s_reset_screen ()               reset the monochrome screen to text
s_screen_on ()                  return screen flag

HISTORY:
   none

*/


/*************************************************************************
                              INCLUDE FILES
*************************************************************************/
#include <stdio.h>
#include <malloc.h>

#include "mconst.h"
#include "mdemux.h"
#include "mdt28xx.h"
#include "merror.h"
#include "mlog.h"
#include "mscreen.h"


/*************************************************************************
                              DEFINES
*************************************************************************/
#define FARHEAP         1
#define NEARHEAP        2


/*************************************************************************
                              GLOBALS
*************************************************************************/
PRIVATE ER_MESSAGE em[] =  {
{ DM_NO_STORAGE          , "Unable to allocate storage in DM_GET_NEW_BUFFER."    , TRUE ,
{ DSP_INVALID_FILE       , "Cannot open .fft file."                              , FALSE,
{ DSP_NO_BOARD           , "Symmetric Research TMS32020/C25 board not found."    , FALSE,
{ DT_FREQTOOHIGH         , "Requested aggregate frequency is too high."          , FALSE,
{ DT_FREQTOOLOW          , "Requested aggregate frequency is too low."           , FALSE,
{ DT_INVALID_BUFFERSIZE  , "Aggregate buffer size exceeds 64K bytes."            , FALSE,
{ DT_INVALID_CHANNEL     , "Invalid channel number."                             , FALSE,
{ DT_INVALID_CHANNELSIZE, "ChannelBlocksize must be a power of 2."               , FALSE,
{ DT_INVALID_GAIN        , "Invalid gain number."                                , FALSE,
{ DT_INVALID_MUX         , "Invalid multiplexer installed."                      , FALSE,
{ DT_INVALID_SCANCOUNT   , "Number of station definitions must be a power of 2.", FALSE,
{ DT_IO_NOTINPROG        , "A/D error -- i/o not in progress."                   , FALSE,
{ DT_NO_ATLDPV           , "ATLAB device driver not found."                      , FALSE,
{ DT_NO_BOARD            , "Illegal DT28xx board number.  Check installation."   , FALSE,
{ DT_NO_STATIONS         , "No station definitions found in input file."         , FALSE,
{ DT_NO_STORAGE          , "Unable to allocate buffer in DT_INITIALIZE_BUFFER."  , TRUE ,
{ DT_OVERFLOW            , "ATLAB buffer overflow."                              , FALSE,
{ DT_TIMEOUT             , "ATLAB user timeout exceeded on input."               , FALSE,
{ DT_UNAVAILABLE_XB      , "Extended memory buffer is not available."            , FALSE,
{ F_DISK_FULL            , "Disk is full."                                       , FALSE,
{ F_INVALID_DRIVE        , "Invalid drive specification in path name."           , FALSE,
{ F_NO_STORAGE           , "Unable to allocate storage in F_INITIALIZE_PARAMS."  , TRUE ,
{ HS_NO_STORAGE          , "Unable to allocate storage in HS_LOCATE_EVENT."      , TRUE ,
{ L_NO_STORAGE           , "Unable to allocate storage in L_LOCATE_EVENT."       , TRU
{ O_DISK_FULL            , "Disk is full."                                       , FAL
{ O_INVALID_DRIVE        , "Invalid drive specification in log path name."       , FAL
{ O_INVALID_LOGFILE      , "Unable to open log file."                            , FAL
{ PK_NO_STORAGE          , "Unable to allocate storage in PK_PICK_EVENT."        , TRU
{ Q_NO_STORAGE           , "Unable to allocate storage in Q_SAVE_LINK."          , TRU
{ S_NO_STORAGE           , "Unable to allocate storage in S_PLOT_BUFFER."        , TRU
{ SP_INVALID_SIZE        , "Illegal spectral size."                              , FAL
{ SP_NO_BOARD            , "Symmetric Research TMS32020/C25 board not found."    , FAL
{ SP_NO_STORAGE          , "Unable to allocate storage in SP_COMPUTE_SPECTRUM."  , TRU
{ ST_INVALID_FILE        , "Invalid station file."                               , FAL
{ ST_NO_STORAGE          , "Unable to allocate storage in ST_ADD_STATION."       , TRU
{ U_NO_STORAGE           , "Unable to allocate storage in U_BUILD_TIMELINE."     , TRU
{ T_INVALID_CHANNEL      , "Invalid channel requested."                          , FAL
{ T_INVALID_LTASIZE      , "LTA window size must be a power of two."             , FAL
{ T_INVALID_STASIZE      , "STA window size must be a power of two."             , FAL
{ T_NO_STORAGE           , "Unable to allocate storage in T_SET_TRIGGER."        , TRU
{ X_INVALID_INPUTFILE    , "Cannot open input file."                             , FAL
{ DM_POWRSTOR            , "Unexpected PowrSTOR error."                          , FAL
{ DM_POWRSTOR_1          , "PowrSTOR initialization is invalid."                 , FAL
{ DM_POWRSTOR_2          , "PowrSTOR initialization data is bad."                , FAL
{ DM_POWRSTOR_3          , "PowrSTOR is not initialized."                        , FAL
{ DM_POWRSTOR_4          , "Not enough memory available to PowrSTOR."            , FAL
{ DM_POWRSTOR_5          , "PowrSTOR file error."                                , FAL
{ DM_POWRSTOR_6          , "PowrSTOR expanded memory error."                     , FAL
{ DM_POWRSTOR_7          , "PowrSTOR extended memory error."                     , FAL
{ DM_POWRSTOR_8          , "PowrSTOR address is invalid."                        , FAL
{ DM_POWRSTOR_9          , "No more memory available to PowrSTOR."               , FAL
{ DM_POWRSTOR_10         , "PowrSTOR heap is invalid."                           , FAL
{ DM_POWRSTOR_11         , "No more heaps available to PowrSTOR"                 , FAL
{ UNEXPECTED_ERROR       , "Unexpected error."                                   , FAL


/*-----------------------------------------------------------------------*
 *                              dump_heap                                *
 *-----------------------------------------------------------------------*/
/* Dump the far heap.                                                    */

PRIVATE
void dump_heap (stream, selector)
FILE *stream;
unsigned int selector;
{
   struct _heapinfo hinfo;
   long freebytes, usedbytes;
   int heapstatus;

   freebytes = usedbytes = 0;
   hinfo._pentry = NULL;
   while ((heapstatus =
          ((selector == FARHEAP) ? _fheapwalk(&hinfo) : _nheapwalk(&hinfo))) ==
          _HEAPOK) {
      fprintf (stream, "%6s block at %p of size %4.4u\n",
             (hinfo._useflag == _USEDENTRY ? "USED" : "FREE"),
             hinfo._pentry, hinfo._size);
      if (hinfo._useflag == _USEDENTRY)
         usedbytes += hinfo._size;
      else freebytes += hinfo._size;
   }

   fprintf (stream, "Used bytes: %ld, free bytes: %ld\n", usedbytes, freebytes);

   switch (heapstatus) {
      case _HEAPEMPTY:
         fprintf (stream, "OK: Empty heap.\n");
         break;
      case _HEAPEND:
         fprintf (stream, "OK: End of heap.\n");
         break;
      case _HEAPBADPTR:
         fprintf (stream, "ERROR: Bad pointer to heap.\n");
         break;
      case _HEAPBADBEGIN:
```

```
            fprintf (stream, "ERROR: Bad start of heap.\n");
            break;
         case _HEAPBADNODE:
            fprintf (stream, "ERROR: Bad node in heap.\n");
            break;
         default:
            fprintf (stream, "ERROR: Unrecognized heapstatus code.\n");
            break;
      }
   }

/*-------------------------------------------------------------------------*
 *                              quit0                                      *
 *-------------------------------------------------------------------------*/
/* Quit without resetting the screen.                                      */

PRIVATE
void quit0 ()
{
   dt_stop_ADC ();
   dm_reset ();
   exit ( 1);
}

/*-------------------------------------------------------------------------*
 *                              er_abort                                   *
 *-------------------------------------------------------------------------*/
/* Display an error message then quit.                                     */

PUBLIC
void er_abort (message_id)
unsigned int message_id;
{
   static char message[80];
   FILE *stream;
   int i;

   if (s_screen_on())
      s_reset_screen();

   for (i = 0; em[i].id != UNEXPECTED_ERROR && em[i].id != message_id; i++);
   sprintf (message, "ERROR %3d: %s\n", message_id, em[i].text);
   printf (message);
   if (em[i].to_logfile)
      o_write_logfile (message);
   if (em[i].memory_dump) {
      printf ("Heaps have been dumped in dumpheap.@@@\n");
      if (stream = fopen ("dumpheap.@@@", "w")) {
         fprintf (stream, "**** Dump of FAR heap ****\n");
         dump_heap (stream, FARHEAP);
         fprintf (stream, "\n\n**** Dump of NEAR heap ****\n");
         dump_heap (stream, NEARHEAP);
         fclose (stream);
      }
   }
   quit0();
}
```

```
/* FILE: merror.h                                (D. Tottingham  04/26/92)

This is an include file of defines, data structure definitions and
external declarations that are common in the merror module.

*/

#ifndef _MERROR_
#define _MERROR_

/*************************************************************************
                                INCLUDES
*************************************************************************/
#include "mconst.h"


/*************************************************************************
                            STRUCTURE DEFINITIONS

*************************************************************************/
typedef struct {
        unsigned int id;
        char * text;
        FLAG memory_dump;
        FLAG to_logfile;
} ER_MESSAGE;


/*************************************************************************
                                DEFINES
*************************************************************************/
#define UNEXPECTED_ERROR        0

#define DM_NO_STORAGE           100
#define DSP_INVALID_FILE        101
#define DSP_NO_BOARD            102
#define DT_FREQTOOHIGH          103
#define DT_FREQTOOLOW           104
#define DT_INVALID_BUFFERSIZE   105
#define DT_INVALID_CHANNEL      106
#define DT_INVALID_CHANNELSIZE  107
#define DT_INVALID_GAIN         108
#define DT_INVALID_MUX          109
#define DT_INVALID_SCANCOUNT    110
#define DT_IO_NOTINPROG         111
#define DT_NO_ATLDRV            112
#define DT_NO_BOARD             113
#define DT_NO_STATIONS          114
#define DT_NO_STORAGE           115
#define DT_OVERFLOW             116
#define DT_TIMEOUT              117
#define DT_UNAVAILABLE_XB       118
#define F_DISK_FULL             119
#define F_INVALID_DRIVE         120
#define F_NO_STORAGE            121
#define HS_NO_STORAGE           122
#define L_NO_STORAGE            123
#define O_DISK_FULL             124
#define O_INVALID_DRIVE         125
#define O_INVALID_LOGFILE       126
#define PK_NO_STORAGE           127
#define Q_NO_STORAGE            128
#define S_NO_STORAGE            129
#define SP_INVALID_SIZE         130
#define SP_NO_BOARD             131
#define SP_NO_STORAGE           132
#define ST_INVALID_FILE         133
#define ST_NO_STORAGE           134
#define U_NO_STORAGE            135
#define T_INVALID_CHANNEL       136
#define T_INVALID_LTASIZE       137
#define T_INVALID_STASIZE       138
#define T_NO_STORAGE            139
#define X_INVALID_INPUTFILE     140
#define DM_POWRSTOR             141
#define DM_POWRSTOR_1           142
#define DM_POWRSTOR_2           143
#define DM_POWRSTOR_3           144
#define DM_POWRSTOR_4           145
#define DM_POWRSTOR_5           146
#define DM_POWRSTOR_6           147
#define DM_POWRSTOR_7           148
#define DM_POWRSTOR_8           149
#define DM_POWRSTOR_9           150
#define DM_POWRSTOR_10          151
#define DM_POWRSTOR_11          152

/*************************************************************************
                            EXTERNAL DECLARATIONS

These functions can be called from all modules that include this file.

*************************************************************************/
PUBLIC void er_abort (unsigned int);


#endif
```

```
/* FILE: mevent.c                                    (D. Tottingham  04/26/92)

This is a collection of C functions that manage events for xdetect.
As of now, this entails managing the event warning bell.  Eventually, this
module will contain a full-featured event manager.  All functions have been
written and compiled medium model.  The following functions are included:

e_initialize ()                 initialize event parameters
e_get_bell_status ()            get bell_enabled
e_set_bell_status ()            set bell_enabled
e_ring_bell ()                  ring event alert bell if enabled
e_toggle_bell_status ()         toggle bell_enabled


EXTERNAL FUNCTIONS CALLED:
   none


HISTORY:
   none

*/


/*************************************************************************
                              INCLUDE FILES
*************************************************************************/
#include <stdio.h>

#include "mconst.h"
#include "mevent.h"


/*************************************************************************
                                GLOBALS
*************************************************************************/
PRIVATE FLAG bell_enabled;


/*-----------------------------------------------------------------------*
 *                            e_initialize                               *
 *-----------------------------------------------------------------------*/
/* Initialize event parameters.                                          */

PUBLIC
void e_initialize ()
{
   bell_enabled = BELL_ENABLED;
}

/*-----------------------------------------------------------------------*
 *                          e_get_bell_status                            *
 *-----------------------------------------------------------------------*/
/* Get bell_enabled.                                                     */

PUBLIC
FLAG e_get_bell_status ()
{
   return bell_enabled;
}

/*-----------------------------------------------------------------------*
 *                          e_set_bell_status                            *
 *-----------------------------------------------------------------------*/
/* Set bell_enabled.                                                     */

PUBLIC
void e_set_bell_status (bell_status)
FLAG bell_status;
{
   bell_enabled = bell_status;
}

/*-----------------------------------------------------------------------*
 *                            e_ring_bell                                *
 *-----------------------------------------------------------------------*/
/* Ring event alert bell if enabled.                                     */

PUBLIC
void e_ring_bell ()
{
   if (bell_enabled)
      printf ("\a");
}

/*-----------------------------------------------------------------------*
 *                         e_toggle_bell_status                          *
 *-----------------------------------------------------------------------*/
/* Toggle bell_enabled.                                                  */

PUBLIC
void e_toggle_bell_status ()
{
   bell_enabled = (bell_enabled) ? FALSE : TRUE;
}
```

```
/* FILE: mevent.h                                  (D. Tottingham  04/26/92)

This is an include file of defines, data structure definitions and
external declarations that are common in the mevent module.

*/

#ifndef _MEVENT_
#define _MEVENT_

/************************************************************************
                                INCLUDES

************************************************************************/
#include "mconst.h"


/************************************************************************
                                DEFINES

************************************************************************/
#define BELL_ENABLED            TRUE


/************************************************************************
                            EXTERNAL DECLARATIONS

These functions can be called from all modules that include this file.

************************************************************************/
PUBLIC void e_initialize ();
PUBLIC FLAG e_get_bell_status ();
PUBLIC void e_set_bell_status (FLAG);
PUBLIC void e_ring_bell ();
PUBLIC void e_toggle_bell_status ();

#endif
```

```
/* FILE: mfile.c                                (D. Tottingham  04/26/92)

This is a collection of C functions that manage the waveform files for xdetect.
All functions have been written and compiled medium model.  The following
functions are included:

f_check_pathname ()             check pathname for validity
f_get_event_number ()           get current event number
f_get_pathname ()               get psthname
f_initialize_index ()           initialize the index file
f_initialize_params ()          initialize key parameters
f_set_NetworkNodeId ()          set network node id
f_set_pathname ()               set pathname
f_set_preevent_status           set preevent_status flag
f_write_buffers ()              write data buffer(s) to disk

EXTERNAL FUNCTIONS CALLED:

b_convert_base10 ()             convert base 36 number into base 10 number
b_incr_base36 ()                increment a base 36 number
b_init_base36 ()                initialize a base 36 number
dm_get_current_time ()          get current time
dm_get_first_buffer ()          get first new demux buffer from demux queue
dm_get_head_buffer ()           get buffer at head of queue
dm_get_next_buffer ()           get next buffer from queue
dm_get_tail_buffer ()           get buffer at tail of demux queue
dsp_get_spectral_channel ()     return spectral trigger channel
er_abort ()                     display an error message then quit
h_increment_event_ctr ()        increment event counter
h_initialize_path ()            initialize path_name
h_set_event_ctr ()              set event counter
h_update ()                     update the header
l_get_location ()               get location if event was located
o_write_logfile ()              write string to log file stream
pk_get_first_arrival ()         get first p-arrival
pk_get_first_magnitude ()       get first magnitude
pk_get_next_arrival ()          get next p-arrival
pk_get_next_magnitude ()        get next magnitude
st_get_head_station ()          get station at head of station queue
st_get_next_station ()          get next station from station queue
suds_initialize ()              initialize a suds structure
suds_initialize_tag ()          initialize a suds structtag
t_get_head_trigger ()           get trigger information at head of channel queue
t_get_last_trigger ()           get last trigger
t_get_next_trigger ()           get next trigger from channel queue
u_build_date ()                 build a date: YRMNDY
u_build_date_fn ()              build a date filename: pathname\YRMNDY
u_get_diskspace ()              get the percentage of free disk space
u_strncpy ()                    copies n characters of far string2 to far string1

HISTORY:
   none

*/


/*************************************************************************
                              INCLUDE FILES
*************************************************************************/
#include <dos.h>
#include <malloc.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>

#include "mbase36.h"
#include "mconst.h"
#include "mdosfunc.h"
#include "mdemux.h"
#include "mdsp.h"
#include "mdt28xx.h"
#include "merror.h"
#include "mfile.h"
#include "mlocate.h"
#include "mlog.h"
#include "mpick.h"
#include "mscrnhdr.h"
#include "mstation.h"
#include "msudsini.h"
#include "mtrigger.h"
#include "mutils.h"
#include "xdetect.h"


/*************************************************************************
                                GLOBALS
*************************************************************************/
PRIVATE F_HEADER far file_header;
PRIVATE char event_filename[MAX_FILENAME_LENGTH * 2];
PRIVATE char temp_filename[MAX_FILENAME_LENGTH * 2];
PRIVATE char index_filename[MAX_FILENAME_LENGTH * 2];
PRIVATE char path_name[MAX_FILENAME_LENGTH];
PRIVATE int far index_buffer[INDEX_FILESIZE];
PRIVATE char far header_buffer[BLOCK_SIZE];
PRIVATE int current_offset;
PRIVATE B_BASE36 index;
PRIVATE long event_number;
PRIVATE FLAG preevent_status;


/*-----------------------------------------------------------------------*
 *                              build_index_fn                           *
 *-----------------------------------------------------------------------*/
/* Build a index filename: pathname\INDEX.@@@                            */

PRIVATE
void build_index_fn ()
{
   sprintf (index_filename, "%s%s", path_name, INDEX_FILENAME);
}

/*-----------------------------------------------------------------------*
 *                              build_calib_fn                           *
 *-----------------------------------------------------------------------*/
/* Build a calibration filename: pathname\YRMNDYxx.CLy
   where xx is the calibration channel in hex, and y is the network code.*/

PRIVATE
int build_calib_fn (abs_time)
double abs_time;
{
   int j;

   j = u_build_date_fn (abs_time, path_name, event_filename);

   j += sprintf (event_filename+j, "%02x.CL%c", dsp_get_spectral_channel(),
                 file_header.info.net_node_id[0]);

   return (j);
}

/*-----------------------------------------------------------------------*
 *                            build_event_number                         *
 *-----------------------------------------------------------------------*/
/* Build a unique event number: YRMNDYxxx, where xxx is a three digit
   base10 number.                                                        */

PRIVATE
long build_event_number (abs_time)
double abs_time;
{
   long date, event_number;
```

```
   date = u_build_date (abs_time);

   event_number = (date * 1000) + b_convert_base10(index);
   return (event_number);
}

/*-------------------------------------------------------------------------*
 *                            build_event_fn                               *
 *-------------------------------------------------------------------------*/
/* Build an event filename: pathname\YRMNDYxx.WVy
   where xx is a two digit base36 number, and y is the network code.      */

PRIVATE
int build_event_fn (abs_time)
double abs_time;
{
   int j;

   j = u_build_date_fn (abs_time, path_name, event_filename);

   j += sprintf (event_filename+j, "%s%s.WV%c", base36_num[index.high],
                 base36_num[index.low], file_header.info.net_node_id[0]);

   return (j);
}

/*-------------------------------------------------------------------------*
 *                           build_preevent_fn                             *
 *-------------------------------------------------------------------------*/
/* Build a preevent filename: YRMNDYxx.PRE
   where xx is a two digit base36 number.                                 */

PRIVATE
int build_preevent_fn (abs_time)
double abs_time;
{
   long date;
   int j;

   date= u_build_date (abs_time);
   j = sprintf (event_filename, "%ld%s%s.PRE", date, base36_num[index.high],
          base36_num[index.low]);

   return (j);
}

/*-------------------------------------------------------------------------*
 *                            build_temp_fn                                *
 *-------------------------------------------------------------------------*/
/* Build a temporary filename: pathname\YRMNDYxx.TMP
   where xx is a two digit base36 number.                                 */

PRIVATE
int build_temp_fn (abs_time)
double abs_time;
{
   int j;

   j = u_build_date_fn (abs_time, path_name, temp_filename);

   j += sprintf (temp_filename+j, "%s%s.TMP", base36_num[index.high],
                 base36_num[index.low]);

   return (j);
}

/*-------------------------------------------------------------------------*
 *                              update_index                               *
 *-------------------------------------------------------------------------*/
/* Increment the index and update the index file.                         */

PRIVATE
void update_index ()
{
   struct tm * local_tm;
   long time;
   int fd;

   time = (long) (dm_get_current_time());
   local_tm = localtime (&time);

   if ( index_buffer[DAY]   == local_tm->tm_mday  &&
        index_buffer[MONTH] == local_tm->tm_mon + 1  &&
        index_buffer[YEAR]  == local_tm->tm_year) {
      b_incr_base36 (&index);
      index_buffer[HIGH] = index.high;
      index_buffer[LOW] = index.low;
   } else {
      b_init_base36 (&index);
      b_incr_base36 (&index);
      index_buffer[DAY] = local_tm->tm_mday;
      index_buffer[MONTH] = local_tm->tm_mon + 1;
      index_buffer[YEAR] = local_tm->tm_year;
      index_buffer[HIGH] = 0;
      index_buffer[LOW] = 0;
   }

   fd = create_file (index_filename);
   write_file (fd, INDEX_FILESIZE * sizeof(int), index_buffer);
   close_file (fd);

   h_increment_event_ctr ();
}

/*-------------------------------------------------------------------------*
 *                             write_to_buffer                             *
 *-------------------------------------------------------------------------*/
/* Buffer file header output.                                             */

PRIVATE
void write_to_buffer (fd, length, source)
int fd;
int length;
char far * source;
{
   char far * hdr;

   if (current_offset+length > BLOCK_SIZE) {
      write_file (fd, current_offset, ((int far *) header_buffer));
      current_offset = 0;
   }
   if (length >= BLOCK_SIZE) {
      write_file (fd, length, ((int far *) source));
   } else {
      hdr = &(header_buffer[current_offset]);
      movedata (FP_SEG(source), FP_OFF(source),
                FP_SEG(hdr), FP_OFF(hdr), length);
      current_offset += length;
   }
}

/*-------------------------------------------------------------------------*
 *                               write_data                                *
 *-------------------------------------------------------------------------*/
/* Write demux data to new waveform file.                                 */

PRIVATE
void write_data (fd, first_buffer)
int fd;
Q_BUFFER far * first_buffer;
{
   Q_BUFFER far * head;
   for (head = first_buffer; head != NULL; head = dm_get_next_buffer()) {
      write_file (fd, head->structtag.len_struct + sizeof(SUDS_STRUCTTAG),
                  ((int far *) &(head->structtag)));
      write_file (fd, (head->buffer_size * sizeof(unsigned)), head->data);
```

```
   }
}

/*-------------------------------------------------------------------------*
 *                            write_header                                 *
 *-------------------------------------------------------------------------*/
/* Construct and write a file header to new waveform file.                 */

PRIVATE
void write_header (fd)
int fd;
{
   DT_28XX far * atodinfo;
   Q_CHANNEL far * triggers;
   Q_STATION far * stationcomp;
   T_EVENT far * eventsetting;
   T_TRIGGER far * trigsetting;

   current_offset = 0;

   /* Write DETECTOR structure */
   write_to_buffer (fd, ((int)(file_header.structtag.len_struct +
                    sizeof(SUDS_STRUCTTAG))),
                    ((char far *) &(file_header.structtag)));

   /* Write ATODINFO structure */
   atodinfo = dt_get_atodinfo();
   write_to_buffer (fd, ((int)(atodinfo->structtag.len_struct +
                    sizeof(SUDS_STRUCTTAG))),
                    ((char far *) &(atodinfo->structtag)));

   /* Write TRIGSETTING structure */
   trigsetting = t_get_trigsetting();
   write_to_buffer (fd, ((int)(trigsetting->structtag.len_struct +
                    sizeof(SUDS_STRUCTTAG))),
                    ((char far *) &(trigsetting->structtag)));

   /* Write EVENTSETTING structure */
   eventsetting = t_get_eventsetting();
   write_to_buffer (fd,((int)(eventsetting->structtag.len_struct +
                    sizeof(SUDS_STRUCTTAG))),
                    ((char far *) &(eventsetting->structtag)));

   /* Write STATIONCOMP structures */
   stationcomp = st_get_head_station ();
   while (stationcomp != NULL) {
      write_to_buffer (fd, ((int)(stationcomp->structtag.len_struct +
                       sizeof(SUDS_STRUCTTAG))),
                       ((char far *) &(stationcomp->structtag)));
      stationcomp = st_get_next_station ();
   }

   /* Write TRIGGERS structures */
   triggers = t_get_head_trigger ();
   while (triggers != NULL) {
      write_to_buffer (fd, ((int)(triggers->structtag.len_struct +
                       sizeof(SUDS_STRUCTTAG))),
                       ((char far *) &(triggers->structtag)));
      triggers = t_get_next_trigger ();
   }

   /* Write the header_buffer to disk */
   if (current_offset) {
      write_file (fd, current_offset, ((int far *) header_buffer));
      current_offset = 0;
   }
}

/*-------------------------------------------------------------------------*
 *                            write_trailer                                *
 *-------------------------------------------------------------------------*/
/* Construct and write a file trailer to new waveform file.                */

PRIVATE
void write_trailer (fd)
int fd;
{
   Q_ARRIVAL far * arrival;
   Q_LOCATION far * location;
   Q_MAGNITUDE far * magnitude;

   current_offset = 0;

   /* Write FEATURES structures */
   arrival = pk_get_first_arrival ();
   while (arrival != NULL) {
      write_to_buffer (fd, ((int)(arrival->structtag.len_struct +
                       sizeof(SUDS_STRUCTTAG))),
                       ((char far *) &(arrival->structtag)));
      arrival = pk_get_next_arrival ();
   }

   magnitude = pk_get_first_magnitude ();
   while (magnitude != NULL) {
      write_to_buffer (fd, ((int)(magnitude->structtag.len_struct +
                       sizeof(SUDS_STRUCTTAG))),
                       ((char far *) &(magnitude->structtag)));
      magnitude = pk_get_next_magnitude ();
   }

   /* Write ORIGIN structure if available */
   if ((location = l_get_location()) != NULL)
      write_to_buffer (fd,((int)(location->structtag.len_struct +
                       sizeof(SUDS_STRUCTTAG))),
                       ((char far *) &(location->structtag)));

   /* Write the header_buffer to disk */
   if (current_offset) {
      write_file (fd, current_offset, ((int far *) header_buffer));
      current_offset = 0;
   }
}

/*-------------------------------------------------------------------------*
 *                          f_check_pathname                               *
 *-------------------------------------------------------------------------*/
/* Check pathname for validity.                                            */

PUBLIC
void f_check_pathname ()
{
   long ds;
   int drive;

   /* Do we have any diskspace? */
   if ((ds = u_get_diskspace (path_name, &drive)) == 0)
      er_abort (F_DISK_FULL);
   else if (ds == INVALID_DRIVE)
      er_abort (F_INVALID_DRIVE);

   h_initialize_path (path_name);
}

/*-------------------------------------------------------------------------*
 *                          f_get_event_number                             *
 *-------------------------------------------------------------------------*/
/* Get current event number.                                               */

PUBLIC
long f_get_event_number ()
{
   return (event_number);
}

/*-------------------------------------------------------------------------*
 *                          f_get_pathname                                 *
```

```
 *-----------------------------------------------------------------------*/
/* Get pathname.                                                         */

PUBLIC
char * f_get_pathname ()
{
   return (path_name);
}

/*-----------------------------------------------------------------------*
 *                        f_initialize_index                             *
 *-----------------------------------------------------------------------*/
/* Open an existing index file or create a new one.  Start index counter
   where it left off in the existing index file or at zero if a new file
   is created.                                                           */

PUBLIC
void f_initialize_index ()
{
   struct tm * local_tm;
   long time;
   int fd;

   h_set_event_ctr ( 0);

   time = (long) (dm_get_current_time());
   local_tm = localtime (&time);
   build_index_fn();

   if ((fd = open_file (index_filename, READ)) != 0) {
      read_file (fd, INDEX_FILESIZE * sizeof(int), index_buffer);
      if ( index_buffer[DAY]   == local_tm->tm_mday  &&
           index_buffer[MONTH] == local_tm->tm_mon + 1   &&
           index_buffer[YEAR]  == local_tm->tm_year) {
         index.high = index_buffer[HIGH];
         index.low  = index_buffer[LOW];
         close_file (fd);
         return;
      }
   }
   b_init_base36 (&index);
   fd = create_file (index_filename);
   index_buffer[DAY] = local_tm->tm_mday;
   index_buffer[MONTH] = local_tm->tm_mon + 1;
   index_buffer[YEAR] = local_tm->tm_year;
   index_buffer[HIGH] = 0;
   index_buffer[LOW] = -1;
   write_file (fd, INDEX_FILESIZE * sizeof(int), index_buffer);
   close_file (fd);
}

/*-----------------------------------------------------------------------*
 *                        f_initialize_params                            *
 *-----------------------------------------------------------------------*/
/* Initialize key parameters.                                            */

PUBLIC
void f_initialize_params ()
{
   preevent_status = PREEVENT_ENABLED;

   strcpy (path_name, PATH_NAME);

   suds_initialize (DETECTOR, &(file_header.info));
   suds_initialize_tag (DETECTOR, &(file_header.structtag));

   file_header.info.dalgorithm = 'x';
   file_header.info.versionnum = VERSIONNUM;
   u_strncpy (file_header.info.net_node_id, ((char far *) NETWORK_NODE_ID), 10);
}

/*-----------------------------------------------------------------------*
 *                        f_set_NetworkNodeId                            *
 *-----------------------------------------------------------------------*/
/* Set network node id.                                                  */

PUBLIC
void f_set_NetworkNodeId (id)
char far * id;
{
   u_strncpy (file_header.info.net_node_id, id, 10);
}

/*-----------------------------------------------------------------------*
 *                          f_set_pathname                               *
 *-----------------------------------------------------------------------*/
/* Set pathname.                                                         */

PUBLIC
void f_set_pathname (pn)
char pn[];
{
   unsigned int length;

   strncpy (path_name, pn, (MAX_FILENAME_LENGTH - 1));
   length = strlen(path_name);
   if (path_name[length-1] != ':' && path_name[length-1] != '\\')
      strcat (path_name, "\\");
}

/*-----------------------------------------------------------------------*
 *                      f_set_preevent_status                            *
 *-----------------------------------------------------------------------*/
/* Set preevent_status pathname.                                         */

PUBLIC
void f_set_preevent_status (status)
FLAG status;
{
   preevent_status = status;
}

/*-----------------------------------------------------------------------*
 *                          f_write_buffers                              *
 *-----------------------------------------------------------------------*/
/* Write data buffer(s) to disk.                                         */

PUBLIC
void f_write_buffers (command, event_type)
unsigned int command;
char event_type;
{
   double eventtime;
   int fd, drive;
   char out_str[80];

   switch (command) {
      case START_FILE:
         eventtime = dm_get_current_time();
         file_header.info.event_type = event_type;
         file_header.info.event_number = event_number =
            build_event_number (eventtime);

         switch (event_type) {
            case F_CALIBRATION:
               build_calib_fn (eventtime);
               fd = create_file (event_filename);
               write_header (fd);
               write_data (fd, dm_get_head_buffer());
               close_file (fd);
               break;

            case F_EARTHQUAKE:
               update_index ();

               if (preevent_status) {
```

```
                build_temp_fn (eventtime);
                fd = create_file (temp_filename);
                write_header (fd);
                write_data (fd, dm_get_head_buffer());
                close_file (fd);
                build_preevent_fn (eventtime);
                sprintf (out_str, "rename %s %s", temp_filename,
                        event_filename);
                system (out_str);
             }

             build_event_fn (eventtime);
             fd = create_file (event_filename);
             write_header (fd);
             write_data (fd, dm_get_head_buffer());
             close_file (fd);
             break;

          case F_PREERUN:
          default:
             update_index ();
             build_event_fn (eventtime);

             fd = create_file (event_filename);
             write_header (fd);
             write_data (fd, dm_get_first_buffer());
             close_file (fd);
       }
       break;

    case CONTINUE_FILE:
       fd = open_file (event_filename, WRITE);
       seek_file (fd, 0L, SEEK_END);
       write_data (fd, dm_get_first_buffer());
       close_file (fd);
       break;

    case END_FILE:
       fd = open_file (event_filename, WRITE);
       seek_file (fd, 0L, SEEK_END);
       write_trailer (fd);
       close_file (fd);

       sprintf (out_str, "\nEvent saved in: %s \n\n", event_filename);
       if (Debug_enabled)
          printf (out_str);
       o_write_logfile (out_str);
       break;
 }

 if (u_get_diskspace(path_name, &drive) == 0)
    er_abort (F_DISK_FULL);
 h_update();
}
```

```
/* FILE: mfile.h                                    (D. Tottingham  04/26/92)

This is an include file of defines, data structure definitions and
external declarations that are common in the mfile module.

*/

#ifndef _MFILE_
#define _MFILE_

/************************************************************************
                                INCLUDES
************************************************************************/
#include "mbase36.h"
#include "mconst.h"
#include "suds.h"


/************************************************************************
                                DEFINES
************************************************************************/
#define INDEX_FILENAME          "INDEX.@@@"
#define INDEX_FILESIZE          10

#define DAY                     0
#define MONTH                   1
#define YEAR                    2
#define HIGH                    3
#define LOW                     4

#define PATH_NAME               "C:"
#define NETWORK_NODE_ID         "M"
#define BLOCK_SIZE              4096

#define PREEVENT_ENABLED        FALSE

/* Event types */
#define F_CALIBRATION           'c'
#define F_EARTHQUAKE            'e'
#define F_FREERUN               'f'

/* Commands */
#define START_FILE              1
#define CONTINUE_FILE           2
#define END_FILE                3


/************************************************************************
                        STRUCTURE DEFINITIONS

The following structure definitions are included in here, so that all
modules can have access to them.

************************************************************************/
typedef struct {
        SUDS_STRUCTTAG structtag;
        SUDS_DETECTOR info;
} F_HEADER;


/************************************************************************
                        EXTERNAL DECLARATIONS

These functions can be called from all modules that include this file.

************************************************************************/
PUBLIC void f_check_pathname ();
PUBLIC long f_get_event_number ();
PUBLIC char * f_get_pathname ();
PUBLIC void f_initialize_index ();
PUBLIC void f_initialize_params ();
PUBLIC void f_set_pathname (char *);
PUBLIC void f_set_preevent_status (FLAG);
PUBLIC void f_write_buffers (unsigned int, char);

#endif
```

```
/* FILE: mfreerun.c                                (D. Tottingham  04/26/92)

This is a collection of C functions that manage the free run events for xdetect.
All functions have been written and compiled medium model.  The following
functions are included:

fr_continue_freerun ()          continue free run event if started
fr_get_block_size ()            get block size in seconds
fr_initialize ()                initialize free run parameters
fr_set_block_size ()            set block size in seconds
fr_start_freerun ()             start free run event
fr_stop_freerun ()              stop free run event
fr_toggle_freerun ()            toggle free run event


EXTERNAL FUNCTIONS CALLED:

dm_get_current_time ()          get current time
f_write_buffers ()              write data buffer(s) to disk
h_toggle_status ()              toggle a status bit

HISTORY:
   none

*/


/*************************************************************************
                              INCLUDE FILES
*************************************************************************/
#include <stdio.h>

#include "mconst.h"
#include "mdemux.h"
#include "mfile.h"
#include "mfreerun.h"
#include "mscrnhdr.h"


/*************************************************************************
                              GLOBALS
*************************************************************************/
PRIVATE unsigned int freerun_state;
PRIVATE double block_size, next_block_time;


/*-----------------------------------------------------------------------*
 *                        fr_continue_freerun                            *
 *-----------------------------------------------------------------------*/
/* Continue free run event if started.                                   */

PUBLIC
FLAG fr_continue_freerun ()
{
   switch (freerun_state) {
      case START_FREERUN:
         f_write_buffers (START_FILE, F_FREERUN);
         next_block_time = dm_get_current_time() + block_size;
         freerun_state = CONTINUE_FREERUN;
         h_toggle_status (H_FREE_RUN);
         break;

      case CONTINUE_FREERUN:
         if (block_size == 0.0 || dm_get_current_time() < next_block_time)
            f_write_buffers (CONTINUE_FILE, F_FREERUN);
         else {
            f_write_buffers (END_FILE, F_FREERUN);
            f_write_buffers (START_FILE, F_FREERUN);
            next_block_time = dm_get_current_time() + block_size;
         }
         break;

      case STOP_FREERUN:
         f_write_buffers (END_FILE, F_FREERUN);
         freerun_state = READY_STATE;
         h_toggle_status (H_FREE_RUN);
         break;

      case READY_STATE:
      default:
         break;
   }
   return (freerun_state == CONTINUE_FREERUN);
}

/*-----------------------------------------------------------------------*
 *                        fr_get_block_size                              *
 *-----------------------------------------------------------------------*/
/* Get block size in seconds.                                            */

PUBLIC
double fr_get_block_size ()
{
   return block_size;
}

/*-----------------------------------------------------------------------*
 *                        fr_initialize                                  *
 *-----------------------------------------------------------------------*/
/* Initialize free run parameters.                                       */

PUBLIC
void fr_initialize ()
{
   freerun_state = READY_STATE;
   block_size = next_block_time = 0.0;
}

/*-----------------------------------------------------------------------*
 *                        fr_set_block_size                              *
 *-----------------------------------------------------------------------*/
/* Set block size in seconds.                                            */

PUBLIC
void fr_set_block_size (size)
double size;
{
   block_size = size;
}

/*-----------------------------------------------------------------------*
 *                        fr_start_freerun                               *
 *-----------------------------------------------------------------------*/
/* Start free run event.                                                 */

PUBLIC
void fr_start_freerun ()
{
   if (freerun_state == READY_STATE) {
      freerun_state = START_FREERUN;
   }
}

/*-----------------------------------------------------------------------*
 *                        fr_stop_freerun                                *
 *-----------------------------------------------------------------------*/
/* Stop free run event.                                                  */

PUBLIC
void fr_stop_freerun ()
{
   if (freerun_state == CONTINUE_FREERUN)
      freerun_state = STOP_FREERUN;
   else if (freerun_state == START_FREERUN)
```

```
    freerun_state = READY_STATE;
}

/*-------------------------------------------------------------------------*
 *                          fr_toggle_freerun                              *
 *-------------------------------------------------------------------------*/
/* Toggle free run event.                                                  */

PUBLIC
void fr_toggle_freerun ()
{
   if (freerun_state == READY_STATE) {
      freerun_state = START_FREERUN;
   } else if (freerun_state == CONTINUE_FREERUN) {
      freerun_state = STOP_FREERUN;
   }
}
```

```
/* FILE: mfreerun.h                                (D. Tottingham  04/26/92)

This is an include file of defines, data structure definitions and
external declarations that are common in the mfreerun module.

*/

#ifndef _MFREERUN_
#define _MFREERUN_

/*************************************************************************
                                INCLUDES

*************************************************************************/
#include "mconst.h"


/*************************************************************************
                                DEFINES

*************************************************************************/
#define READY_STATE             0
#define START_FREERUN           1
#define CONTINUE_FREERUN        2
#define STOP_FREERUN            3


/*************************************************************************
                          EXTERNAL DECLARATIONS

These functions can be called from all modules that include this file.

*************************************************************************/
PUBLIC FLAG fr_continue_freerun ();
PUBLIC double fr_get_block_size ();
PUBLIC void fr_initialize ();
PUBLIC void fr_set_block_size (double);
PUBLIC void fr_start_freerun ();
PUBLIC void fr_stop_freerun ();
PUBLIC void fr_toggle_freerun ();

#endif
```

```
/* FILE: mhalfsp.c                                  (D. Tottingham  04/26/92)

These C functions implement the halfspace location algorithm developed by
John Lahr et. al.  All functions have been written and compiled medium model.
The following functions are included:

hs_initialize ()                    initialize halfspace module
hs_locate_event ()                  locate event using halfspace algorithm
hs_set_CriticalPhi ()               set critical phi
hs_set_HalfspaceVelocity ()         set halfspace velocity

EXTERNAL FUNCTIONS CALLED:

er_abort ()                         display an error message then quit
ll_to_latlon ()                     convert x, y to lat, lon
pk_get_first_arrival ()             get first p-arrival
pk_get_first_magnitude ()           get first magnitude
pk_get_last_arrival ()              get last p-arrival
pk_get_last_magnitude ()            get last magnitude
pk_get_next_arrival ()              get next p-arrival
pk_get_next_magnitude ()            get next magnitude
pk_get_narrivals ()                 get number of p-arrivals
pk_get_nmagnitudes ()               get number of magnitudes
st_get_station ()                   get station from station queue

HISTORY:

V1.95   (09/29/90)  Added rms residual calculation to hs_locate_event().

*/


/*************************************************************************
                              INCLUDE FILES
*************************************************************************/
#include <malloc.h>
#include <math.h>
#include <stdio.h>

#include "mconst.h"
#include "merror.h"
#include "mhalfsp.h"
#include "mlatlon.h"
#include "mpick.h"
#include "mstation.h"
#include "xdetect.h"
#include "suds.h"


/*************************************************************************
                              GLOBAL DATA
*************************************************************************/
PRIVATE unsigned int critical_phi;
PRIVATE double hs_velocity, hs_vsq, hsv_inverted;


/*-----------------------------------------------------------------------*
 *                              deter3x3                                 *
 *-----------------------------------------------------------------------*/
/* Find 3 x 3 determinant of 4 x 4 matrix.  Parameter matrix is a 4 x 4
   matrix.  Parameters col1, col2, col3 are the columns of the 4 x 4
   matrix to be used to compute the determinant.                         */

PRIVATE
double deter3x3 (matrix, col0, col1, col2)
double * * matrix;
unsigned int col0, col1, col2;
{
   double deter;

   deter = ( matrix[col0][0] * ( matrix[col1][1] * matrix[col2][2] -
           matrix[col2][1] * matrix[col1][2] )) -
           ( matrix[col1][0] * ( matrix[col0][1] * matrix[col2][2] -
           matrix[col2][1] * matrix[col0][2] )) +
           ( matrix[col2][0] * ( matrix[col0][1] * matrix[col1][2] -
           matrix[col1][1] * matrix[col0][2] ));
   return (deter);
}

/*-----------------------------------------------------------------------*
 *                              hs_initialize                            *
 *-----------------------------------------------------------------------*/
/* Initialize the halfspace module.                                      */

PUBLIC
void hs_initialize ()
{
   critical_phi = CRITICAL_PHI;
   hs_velocity = HALFSPACE_VELOCITY;
   hs_vsq = hs_velocity * hs_velocity;
   hsv_inverted = 1.0 / hs_velocity;
}

/*-----------------------------------------------------------------------*
 *                              hs_locate_event                          *
 *-----------------------------------------------------------------------*/
/* Locate event using halfspace algorithm.                               */

PUBLIC
FLAG hs_locate_event (origin)
SUDS_ORIGIN far * origin;
{
   Q_ARRIVAL far *pick_ptr, far *first_pick, far *last_pick;
   Q_MAGNITUDE far *mag_ptr;
   Q_STATION far *station_ptr, far *last_station;
   double *sym_tensor[4], *normal_eqn[4], d, x, y, rel_origin;
   double deltat, deltax, deltay, depthsq, coda_magnitude, residual;
   unsigned int n_arrivals, n_magnitudes, i, j, k;
   LL_LATLON latlon;
   FLAG located_flag, first_time;

   /* Do we have enough p-arrivals? */
   if ((n_arrivals = pk_get_narrivals()) < critical_phi) return (FALSE);

   /* Allocate some storage */
   for (i = 0; i < 4; i++) {
      sym_tensor[i] = (double *) malloc (4 * sizeof(double));
      normal_eqn[i] = (double *) malloc (n_arrivals * sizeof(double));
      if (sym_tensor[i] == NULL || normal_eqn[i] == NULL) er_abort (HS_NO_STORAGE);
   }

   /* Setup normal equations */
   first_pick = pk_get_first_arrival ();
   last_pick = pk_get_last_arrival ();
   last_station = st_get_station (last_pick->channel);
   for (i = 0, pick_ptr = first_pick; i < (n_arrivals - 1); i++,
            pick_ptr= pk_get_next_arrival()) {
      station_ptr = st_get_station (pick_ptr->channel);
      normal_eqn[0][i] = last_station->x - station_ptr->x;
      normal_eqn[1][i] = last_station->y - station_ptr->y;
      normal_eqn[2][i] = -(hs_vsq) * (last_pick->rel_arrival - pick_ptr->rel_arrival
      normal_eqn[3][i] = (last_station->xsq + last_station->ysq -
                          (last_pick->rel_arrival * last_pick->rel_arrival * hs_vsq)
                          station_ptr->xsq - station_ptr->ysq +
                          (pick_ptr->rel_arrival * pick_ptr->rel_arrival * hs_vsq))
   }

   /* Compute sum of squares matrix from coefficients */
   for (i = 0; i < 4; i++)
      for (j = i; j < 4; j++) {
         sym_tensor[i][j] = 0.0;
         for (k = 0; k < (n_arrivals - 1); k++)
            sym_tensor[i][j] += normal_eqn[i][k] * normal_eqn[j][k];
```

```
            sym_tensor[j][i] = sym_tensor[i][j];
        }

    /* Solve simultaneous equations using determinants */
    d = deter3x3 (sym_tensor, 0, 1, 2);

    if (d > DETERMINANT_LIMIT) {
        x = deter3x3 (sym_tensor, 3, 1, 2) / d;
        y = deter3x3 (sym_tensor, 0, 3, 2) / d;
        rel_origin = deter3x3 (sym_tensor, 0, 1, 3) / d;

        /* Convert x, y to lat, lon */
        latlon = ll_to_latlon (y, -x, last_station->mean_latlon);
        origin->or_lat = latlon.lat;
        origin->or_long = -latlon.lon;

        /* Calculate origin time */
        origin->orgtime = rel_origin + first_pick->info.time;

        /* Calculate depth */
        deltat = last_pick->rel_arrival - rel_origin;
        deltax = last_station->x - x;
        deltay = last_station->y - y;
        depthsq = hs_vsq*deltat*deltat - deltax*deltax - deltay*deltay;
        if (depthsq > 0.0)
            origin->depth = sqrt (depthsq);
        else origin->depth = 0.0;

        /* Some fields in origin structure need filling in */
        origin->num_stats = origin->rep_p = origin->used_p = n_arrivals;
        origin->rep_s = origin->used_s = 0;

        /* Calculate magnitude */
        first_time = TRUE;
        if ((n_magnitudes = pk_get_nmagnitudes()) != 0) {
            for (mag_ptr = pk_get_first_magnitude(); mag_ptr != NULL; mag_ptr = pk_get_next
                coda_magnitude = -0.87 + (2.0 * log10(mag_ptr->info.time));
                if (first_time) {
                    origin->magnitude = (float) coda_magnitude;
                    first_time = FALSE;
                } else if (coda_magnitude > origin->magnitude)
                    origin->magnitude = (float) coda_magnitude;
            }
            origin->rep_m = origin->used_m = n_magnitudes;
            origin->mag_type = 'c';
        }

        /* Calculate rms of residuals */
        origin->res_rms = 0.0;

        for (i = 0, pick_ptr = pk_get_first_arrival (); pick_ptr != NULL; i++,
                pick_ptr = pk_get_next_arrival()) {
            station_ptr = st_get_station (pick_ptr->channel);
            residual = (station_ptr->x - x) * (station_ptr->x - x);
            residual += (station_ptr->y - y) * (station_ptr->y - y);
            residual += origin->depth * origin->depth;
            residual = -(hsv_inverted * sqrt(residual));
            residual += pick_ptr->rel_arrival - rel_origin;
            origin->res_rms += ((float)(residual * residual));
        }
        origin->res_rms = sqrt(((double)(origin->res_rms)) / n_arrivals);

        located_flag = TRUE;
    } else located_flag = FALSE;

    /* Free storage */
    for (i = 0; i < 4; i++) {
        free (sym_tensor[i]); free (normal_eqn[i]);
    }
    return (located_flag);
}

/*----------------------------------------------------------------------*
 *                          hs_set_CriticalPhi                          *
 *----------------------------------------------------------------------*/
/* Set critical phi.                                                    */

PUBLIC
void hs_set_CriticalPhi (cp)
unsigned int cp;
{
    critical_phi = cp;
}

/*----------------------------------------------------------------------*
 *                        hs_set_HalfspaceVelocity                      *
 *----------------------------------------------------------------------*/
/* Set halfspace velocity.                                              */

PUBLIC
void hs_set_HalfspaceVelocity (velocity)
double velocity;
{
    hs_velocity = velocity;
    hs_vsq = hs_velocity * hs_velocity;
    hsv_inverted = 1.0 / hs_velocity;
}
```

```
/* FILE: mhalfsp.h                                  (D. Tottingham  04/26/92)

This is an include file of defines, data structure definitions and
external declarations that are common in the mhalfsp module.

*/

#ifndef _MHALFSP_
#define _MHALFSP_

/************************************************************************
                                INCLUDES

************************************************************************/
#include "mconst.h"
#include "suds.h"


/************************************************************************
                                DEFINES

************************************************************************/
#define DETERMINANT_LIMIT       0.00001
#define CRITICAL_PHI            4
#define HALFSPACE_VELOCITY      6.5


/************************************************************************
                          EXTERNAL DECLARATIONS

These functions can be called from all modules that include this file.

************************************************************************/
PUBLIC void hs_initialize ();
PUBLIC FLAG hs_locate_event (SUDS_ORIGIN far *);
PUBLIC void hs_set_CriticalPhi (unsigned int);
PUBLIC void hs_set_HalfspaceVelocity (double);

#endif
```

```
/* FILE: mhelp.h                                    (D. Tottingham  04/26/92)

This is an include file of the help information for the mscreen module.

*/

#ifndef _MHELP_
#define _MHELP_

/*************************************************************************
                                INCLUDES

*************************************************************************/
#include "mconst.h"


/*************************************************************************
                              EXTERNAL DATA

The following external data can be accessed by any modules that includes
this file.

*************************************************************************/
char * help_info [] = {
"CTRL B          toggle event alert bell",
"CTRL F          toggle free run",
"CTRL L          toggle real-time pick and location",
"CTRL R          toggle autoreboot",
"CTRL T          toggle autotrigger",
"",
"UP ARROW        amplify waveforms",
"DOWN ARROW      attenuate waveforms",
"",
"CTRL HOME       view selected channels",
"CTRL PGDOWN     view next 16 channels",
"CTRL PGUP       view previous 16 channels",
"",
"CTRL Q          QUIT",
"",
"",
"Hit any key to continue.",
NULL };

#endif
```

```
/* FILE: mlatlon.c                                    (D. Tottingham  04/26/92)

This is a collection of C helper functions that manage latlon coordinate
transformations for xdetect.  All functions have been written and compiled
medium model.  The following functions are included:

ll_fold ()                      convert lat, lon to geocentric lat, lon
ll_to_latlon ()                 convert delta lat, lon to lat, lon
ll_to_xy ()                     convert lat, lon to x, y

EXTERNAL FUNCTIONS CALLED:
   none

HISTORY:
   none

*/


/*************************************************************************
                            INCLUDE FILES

*************************************************************************/
#include <math.h>

#include "mconst.h"
#include "mlatlon.h"


/*-----------------------------------------------------------------------*
 *                              hypot                                    *
 *-----------------------------------------------------------------------*/
/* Calculate euclidian distance accurately without overflow.             */

PRIVATE
double hypot (x, y)
double x, y;
{
   double new_x, new_y, temp;

   new_x = fabs(x);
   new_y = fabs(y);

   /* Swap new_x and new_y if new_x > new_y */
   if (new_x > new_y) {
      temp = new_x; new_x = new_y; new_y = temp;
   }

   if (new_y) {
      new_x /= new_y;
      return ( new_y * sqrt (new_x * new_x + 1.0));
   } else return (0.0);
}

/*-----------------------------------------------------------------------*
 *                            xy_to_polar                                *
 *-----------------------------------------------------------------------*/
/* Convert x, y to r, theta                                              */

PRIVATE
LL_POLAR xy_to_polar (x, y)
double x, y;
{
   LL_POLAR p;

   p.r = hypot (x, y);
   p.theta = 0.0;
   if (y || x) p.theta = atan2 (y, x);

   return (p);
}

/*-----------------------------------------------------------------------*
 *                           latlon_to_polar                             *
 *-----------------------------------------------------------------------*/
/* Convert geocentric lat, lon to polar coordinates.                     */

PRIVATE
LL_POLAR latlon_to_polar (latlon, mean_latlon)
LL_LATLON latlon, mean_latlon;
{
   LL_POLAR p1;
   double st0, st1, ct0, ct1, sdlon, cdlon, cdelta, radius, t1, t2;

   st0 = cos(mean_latlon.lat);
   ct0 = sin(mean_latlon.lat);
   t1 = (st0 * st0) / EQURAD / EQURAD;
   t2 = (ct0 * ct0) / POLRAD / POLRAD;
   radius = 1.0 / sqrt (t1 + t2);
   st1 = cos(latlon.lat);
   ct1 = sin(latlon.lat);
   sdlon = sin(mean_latlon.lon - latlon.lon);
   cdlon = cos(mean_latlon.lon - latlon.lon);
   cdelta = (st0 * st1 * cdlon) + (ct0 * ct1);

   p1 = xy_to_polar (((st0 * ct1) - (st1 * ct0 * cdlon)), st1 * sdlon);
   p1.r = radius * atan2 (p1.r, cdelta);

   if (p1.theta < 0.0) p1.theta += TWOPI;
   else if (p1.theta >= TWOPI) p1.theta -= TWOPI;
   p1.theta *= DEG;

   return (p1);
}

/*-----------------------------------------------------------------------*
 *                              unfold                                   *
 *-----------------------------------------------------------------------*/
/* Convert geocentric lat, lon to lat, lon.                              */

PRIVATE
LL_LATLON unfold (lat, lon)
double lat, lon;
{
   LL_LATLON latlon;

   if ((HALFPI - fabs(lat)) >= 0.02)
      latlon.lat = C1 * (lat + ((lat >= 0) ? C2 : -C2));
   else latlon.lat = atan ( tan(lat) / C1);

   latlon.lat *= DEG;
   latlon.lon = lon * DEG;

   return (latlon);
}

/*-----------------------------------------------------------------------*
 *                              ll_fold                                  *
 *-----------------------------------------------------------------------*/
/* Convert lat, lon to geocentric lat, lon.                              */

PUBLIC
LL_LATLON ll_fold (lat, lon)
double lat, lon;
{
   LL_LATLON geocentric;
   int sign;

   /* Compute the geocentric latitude */
   sign = (lat >= 0) ? 1 : -1;
   geocentric.lat = fabs(lat) * RAD;
   if ((HALFPI - geocentric.lat) >= 0.02)
      geocentric.lat = atan (C1 * tan(geocentric.lat));
   else geocentric.lat /= C1 - C2;
   geocentric.lat *= sign;
```

```
   /* Compute the geocentric longitude */
   geocentric.lon = lon * RAD;

   return (geocentric);
}

/*-----------------------------------------------------------------------*
 *                          ll_to_latlon                                 *
 *-----------------------------------------------------------------------*/
/* Convert delta lat, lon to lat, lon.  The parameter pair mean_lat,
   mean_long is the latitude, longitude reference coordinate where
   x,y = (0, 0).                                                         */

PUBLIC
LL_LATLON ll_to_latlon (delta_lat, delta_lon, mean_latlon)
double delta_lat, delta_lon;
LL_LATLON mean_latlon;
{
   LL_LATLON latlon;
   LL_POLAR  p1, p2;
   double st0, st1, ct0, ct1, cz0, radius, sdelta, cdelta, t1, t2;
   double lat, lon;

   p1 = xy_to_polar (delta_lat, delta_lon);
   if (p1.theta < 0.0) p1.theta += TWOPI;

   st0 = cos (mean_latlon.lat); ct0 = sin (mean_latlon.lat);

   t1 = (st0 * st0) / EQURAD / EQURAD;
   t2 = (ct0 * ct0) / POLRAD / POLRAD;
   radius = 1.0 / sqrt (t1 + t2);
   sdelta = sin (p1.r / radius);
   cdelta = cos (p1.r / radius);
   cz0 = cos (p1.theta);
   ct1 = (st0 * sdelta * cz0) + (ct0 * cdelta);

   p2 = xy_to_polar (((st0 * cdelta) - (ct0 * sdelta * cz0)), sdelta* sin(p1.theta));
   lat = atan2 (ct1, p2.r);
   lon = mean_latlon.lon + p2.theta;

   if (fabs(lon) > PI) lon -= (lon >= 0) ? TWOPI : -TWOPI;
   latlon = unfold (lat, lon);
   return (latlon);
}

/*-----------------------------------------------------------------------*
 *                          ll_to_xy                                     *
 *-----------------------------------------------------------------------*/
/* Convert lat, lon to x, y.  The parameter pair mean_lat, mean_long is
   the latitude, longitude reference coordinate where x,y = (0, 0).      */

PUBLIC
LL_XY ll_to_xy (lat, lon, mean_latlon)
double lat, lon;
LL_LATLON mean_latlon;
{
   LL_POLAR p;
   LL_XY rect;

   p = latlon_to_polar (ll_fold (lat, lon), mean_latlon);
   rect.x = p.r * sin (p.theta * RAD);
   rect.y = p.r * cos (p.theta * RAD);
   return (rect);
}
```

```
/* FILE: mlatlon.h                                (D. Tottingham  04/26/92)

This is an include file of defines, data structure definitions and
external declarations that are common in the mlatlon module.

*/

#ifndef _MLATLON_
#define _MLATLON_


/************************************************************************
                                INCLUDES

************************************************************************/
#include "mconst.h"


/************************************************************************
                                DEFINES

************************************************************************/
#define PI              3.14159265
#define TWOPI           2.0 * (PI)
#define HALFPI          0.5 * (PI)
#define RAD             (PI) / 180.0
#define DEG             1.0 / (RAD)
#define EQURAD          6378.2064
#define EQURADSQ        (EQURAD) * (EQURAD)
#define POLRAD          6356.5838
#define POLRADSQ        (POLRAD) * (POLRAD)
#define FLAT            ((EQURAD) - (POLRAD)) / EQURAD
#define C1              (1.0 - (FLAT)) * (1.0 - (FLAT))
#define C2              (HALFPI) * (1.0 / (C1) - 1.0)


/************************************************************************
                        STRUCTURE DEFINITIONS

The following structure definitions are included here so that all
modules can have access to them.

************************************************************************/
typedef struct {
        double lat;
        double lon;
} LL_LATLON;

typedef struct {
        double r;
        double theta;
} LL_POLAR;

typedef struct {
        double x;
        double y;
} LL_XY;


/************************************************************************
                        EXTERNAL DECLARATIONS

These functions can be called from all modules that include this file.

************************************************************************/
PUBLIC LL_LATLON ll_fold (double, double);
PUBLIC LL_LATLON ll_to_latlon (double, double, LL_LATLON);
PUBLIC LL_XY ll_to_xy (double, double, LL_LATLON);

#endif
```

```
/* FILE: mlexer.c                                (D. Tottingham  04/26/92)

This is a collection of C functions that lexes the control file for xdetect.
All functions have been written and compiled medium model.  The following
functions are included:

lx_initialize ()                initialize the lexer
lx_match ()                     match a token with the lookahead
lx_set_stream ()                set the input stream

EXTERNAL FUNCTIONS CALLED:

o_write_logfile ()              write string to log file stream

HISTORY:
   none

*/


/*************************************************************************
                            INCLUDE FILES

*************************************************************************/
#include <stdio.h>
#include <ctype.h>

#include "mconst.h"
#include "mlexer.h"
#include "mlog.h"


/*************************************************************************
                            GLOBAL DATA

*************************************************************************/
PRIVATE LX_KEYWORD keywords[] = {
   {"AuthorityCode"             ,  C_AUTHORITY             },
   {"AutoFreerun"               ,  C_AUTO_FREERUN          },
   {"AutoLocation"              ,  C_AUTO_LOCATION         },
   {"AutoLog"                   ,  C_AUTO_LOG              },
   {"Autoreboot"                ,  C_AUTO_REBOOT           },
   {"Autotrigger"               ,  C_AUTO_TRIGGER          },
   {"BandLimits"                ,  C_BAND_LIMITS           },
   {"BandRecording"             ,  C_BAND_RECORDING        },
   {"BuffersToAverage"          ,  C_BUFFERS_TO_AVERAGE    },
   {"CalibrationRecording"      ,  C_CALIBRATION_RECORDING },
   {"Ch"                        ,  C_CH                    },
   {"ChannelBlocksize"          ,  C_CHANNEL_BLOCKSIZE     },
   {"ChannelGain"               ,  C_CHANNEL_GAIN          },
   {"ClockSource"               ,  C_CLOCK_SOURCE          },
   {"Component"                 ,  C_COMPONENT             },
   {"CriticalAlpha"             ,  C_CRITICAL_ALPHA        },
   {"CriticalBeta"              ,  C_CRITICAL_BETA         },
   {"CriticalGamma"             ,  C_CRITICAL_GAMMA        },
   {"CriticalMu"                ,  C_CRITICAL_MU           },
   {"CriticalNu"                ,  C_CRITICAL_NU           },
   {"CriticalPhi"               ,  C_CRITICAL_PHI          },
   {"DigitizationRate"          ,  C_DIGITIZATION_RATE     },
   {"Display"                   ,  C_DISPLAY               },
   {"EventAlertBell"            ,  C_EVENT_ALERT_BELL      },
   {"EventContinuationCount"    ,  C_EVENT_CONTINUE_COUNT  },
   {"EXTERNAL"                  ,  C_EXTERNAL              },
   {"FALSE"                     ,  C_FALSE                 },
   {"FirstDifference"           ,  C_FIRSTDIFF             },
   {"FreerunBlockTime"          ,  C_FREERUN_BLOCK_TIME    },
   {"Gain"                      ,  C_GAIN                  },
   {"HalfspaceVelocity"         ,  C_HALFSPACE_VELOCITY    },
   {"INTERNAL"                  ,  C_INTERNAL              },
   {"LogPathName"               ,  C_LOG_PATHNAME          },
   {"LTALowerBound"             ,  C_LTA_LOWER_BOUND       },
   {"LTAverageWindow"           ,  C_LTA_WINDOW            },
   {"MaxCalibrationTime"        ,  C_MAX_CALIBRATION_TIME  },
   {"MaxEventTime"              ,  C_MAX_EVENT_TIME        },
   {"MinEventTime"              ,  C_MIN_EVENT_TIME        },
   {"NetworkName"               ,  C_NETWORK               },
   {"NetworkNodeId"             ,  C_NETWORK_NODE_ID       },
   {"OFF"                       ,  C_OFF                   },
   {"ON"                        ,  C_ON                    },
   {"PathName"                  ,  C_PATHNAME              },
   {"PreEventRecording"         ,  C_PREEVENT_RECORDING    },
   {"PreEventTime"              ,  C_PRE_EVENT_TIME        },
   {"STAverageWindow"           ,  C_STA_WINDOW            },
   {"StName"                    ,  C_STNAME                },
   {"Time"                      ,  C_TIME                  },
   {"Trigger"                   ,  C_TRIGGER               },
   {"TriggerConfirmationCount"  ,  C_TRIGGER_CONFIRM_COUNT },
   {"TriggerSource"             ,  C_TRIGGER_SOURCE        },
   {"TriggerTimeLimit"          ,  C_TRIGGER_TIME_LIMIT    },
   {"TRUE"                      ,  C_TRUE                  },
   { NULL                       ,  -1                      }
};

PRIVATE FILE * stream;
PRIVATE char out_str[85];

LX_INFO lex_info;


/*-----------------------------------------------------------------------*
 *                              lexer                                    *
 *-----------------------------------------------------------------------*/
/* Get a token from the input stream.                                     */

PRIVATE
void lexer()
{
   int ch, index;

   /* Munch white space and comments */
   while (isspace(lex_info.lexeme[0] = ch = fgetc(stream)) ||
          ch == COMMENT_CHAR) {
      if (ch == '\n')
         lex_info.lineno++;
      else if (ch == COMMENT_CHAR) {
         while ((lex_info.lexeme[0] = ch = fgetc(stream)) != '\n');
         lex_info.lineno++;
      }
   }
   lex_info.lexeme[1] = '\0';

   if (ch == EOF)
      lex_info.lookahead = C_DONE;

   else if (isdigit(lex_info.lexeme[0])) {
      lex_info.lookahead = C_CONSTANT;
      fscanf(stream, "%[0123456789.]", lex_info.lexeme+1);

   } else if (isalpha(lex_info.lexeme[0])) {
      lex_info.lookahead = C_ID;
      for (index = 1; index < MAXLEX-1 && isalnum(ch = fgetc(stream)); index++)
         lex_info.lexeme[index] = ch;

      lex_info.lexeme[index] = '\0';
      if (index == MAXLEX-1) {
         sprintf (out_str, "WARNING: Command %s has been truncated to ",
                           "maximum command length.\n", lex_info.lexeme);
         o_write_logfile (out_str);
         printf (out_str);
      }
      ungetc (ch, stream);
      for (index = 0; keywords[index].token != -1; index++)
         if (!strcmp(keywords[index].name, lex_info.lexeme))
            lex_info.lookahead = keywords[index].token;
```

```
   } else if (lex_info.lexeme[0] == '"') {
      lex_info.lookahead = C_STRING;
      for (index = 0; index < MAXLEX-1 && (ch = fgetc(stream)) != '"'; index++)
         lex_info.lexeme[index] = ch;
      lex_info.lexeme[index] = '\0';

   } else if (lex_info.lexeme[0] == '\'') {
      lex_info.lookahead = C_CHARACTER;
      lex_info.lexeme[0] = fgetc(stream);
      lex_info.lexeme[1] = '\0';
      if ((ch = fgetc (stream)) != '\'') {
         sprintf (out_str, "WARNING: Invalid character constant on line %d\n", lex_info.
         o_write_logfile (out_str);
         printf (out_str);
      }
   } else lex_info.lookahead = *lex_info.lexeme;
}

/*-------------------------------------------------------------------------*
 *                          lx_initialize                                  *
 *-------------------------------------------------------------------------*/
/* Initialize the lexer.                                                   */

PUBLIC
void lx_initialize ()
{
   lex_info.lookahead = 0;
   lex_info.lineno = 1;
   lex_info.error = 0;
}

/*-------------------------------------------------------------------------*
 *                            lx_match                                     *
 *-------------------------------------------------------------------------*/
/* Match a token with the lookahead.                                       */

PUBLIC
void lx_match (token)
int token;
{
   if (lex_info.error) {
      if (token == ';') {
         lex_info.error = OFF;
         lexer();
      }

   } else if (lex_info.lookahead != token) {
      sprintf(out_str, "WARNING: Syntax error on line %d of control file.\n", lex_info.l
      o_write_logfile (out_str);
      printf (out_str);
      while (lex_info.lookahead != ';' && lex_info.lookahead != C_DONE)
         lexer();
      lex_info.error = ON;

   } else lexer();
}

/*-------------------------------------------------------------------------*
 *                          lx_set_stream                                  *
 *-------------------------------------------------------------------------*/
/* Set the input stream.                                                   */

PUBLIC
void lx_set_stream (the_stream)
FILE * the_stream;
{
   stream = the_stream;
}
```

```
/* FILE: mlexer.h                                  (D. Tottingham  04/26/92)

This is an include file of defines, data structure definitions and
external declarations that are common in the mlexer module.

*/

#ifndef _MLEXER_
#define _MLEXER_

/************************************************************************
                                INCLUDES

************************************************************************/
#include "mconst.h"


/************************************************************************
                                DEFINES

************************************************************************/
#define MAXLEX                  80
#define COMMENT_CHAR            '#'

#define C_CHARACTER             301
#define C_CONSTANT              302
#define C_DONE                  303
#define C_ID                    304
#define C_STRING                305

#define C_AUTHORITY             306
#define C_AUTO_FREERUN          307
#define C_AUTO_LOCATION         308
#define C_AUTO_LOG              309
#define C_AUTO_REBOOT           310
#define C_AUTO_TRIGGER          311
#define C_BAND_LIMITS           312
#define C_BAND_RECORDING        313
#define C_BUFFERS_TO_AVERAGE    314
#define C_CALIBRATION_RECORDING 315
#define C_CH                    316
#define C_CHANNEL_BLOCKSIZE     317
#define C_CHANNEL_GAIN          318
#define C_CLOCK_SOURCE          319
#define C_COMPONENT             320
#define C_CRITICAL_ALPHA        321
#define C_CRITICAL_BETA         322
#define C_CRITICAL_GAMMA        323
#define C_CRITICAL_MU           324
#define C_CRITICAL_NU           325
#define C_CRITICAL_PHI          326
#define C_DIGITIZATION_RATE     327
#define C_DISPLAY               328
#define C_EVENT_ALERT_BELL      329
#define C_EVENT_CONTINUE_COUNT  330
#define C_EXTERNAL              331
#define C_FALSE                 332
#define C_FIRSTDIFF             333
#define C_FREERUN_BLOCK_TIME    334
#define C_GAIN                  335
#define C_HALFSPACE_VELOCITY    336
#define C_INTERNAL              337
#define C_LOG_PATHNAME          338
#define C_LTA_LOWER_BOUND       339
#define C_LTA_WINDOW            340
#define C_MAX_CALIBRATION_TIME  341
#define C_MAX_EVENT_TIME        342
#define C_MIN_EVENT_TIME        343
#define C_NETWORK               344
#define C_NETWORK_NODE_ID       345
#define C_OFF                   346
#define C_ON                    347
#define C_PATHNAME              348
#define C_PRE_EVENT_TIME        349
#define C_PREEVENT_RECORDING    350
#define C_STA_WINDOW            351
#define C_STNAME                352
#define C_TIME                  353
#define C_TRIGGER               354
#define C_TRIGGER_CONFIRM_COUNT 355
#define C_TRIGGER_SOURCE        356
#define C_TRIGGER_TIME_LIMIT    357
#define C_TRUE                  358


/************************************************************************
                        STRUCTURE DEFINITIONS

************************************************************************/
typedef struct {
        char *name;
        int token;
} LX_KEYWORD;

typedef struct {
        int lookahead;
        int lineno;
        int error;
        char lexeme[MAXLEX];
} LX_INFO;


/************************************************************************
                                GLOBAL DATA

************************************************************************/
PUBLIC LX_INFO lex_info;


/************************************************************************
                        EXTERNAL DECLARATIONS

These functions can be called from all modules that include this file.

************************************************************************/
PUBLIC void lx_initialize ();
PUBLIC void lx_match (int);
PUBLIC void lx_set_stream (FILE *);

#endif
```

```
/* FILE: mlocate.c                                    (D. Tottingham  04/26/92)

This is a collection of C helper functions that manage event location for
xdetect.  All functions have been written and compiled medium model.  The
following functions are included:

l_display_location ()           write location to stream
l_get_first_location ()         get first location
l_get_location ()               get location if event was located
l_get_location_status ()        get location_enabled
l_get_next_location ()          get next location
l_initialize ()                 initialize location module
l_locate_event ()               locate the event
l_reset_location ()             reset location flag
l_set_location_status ()        set location_enabled flag
l_toggle_location_status ()     toggle location_enabled flag

EXTERNAL FUNCTIONS CALLED:

er_abort ()                     display an error message then quit
hs_initialize ()                initialize halfspace module
hs_locate_event ()              locate event using halfspace algorithm
q_dequeue ()                    dequeue a data link from a data queue
q_enqueue ()                    enqueue a data link on a data queue
q_initialize ()                 initialize a data queue
suds_initialize ()              initialize a suds structure
suds_initialize_tag ()          initialize a suds structtag

HISTORY:

V1.95   (09/29/90)  Added rms residual to l_display_location().

*/


/*************************************************************************
                              INCLUDE FILES
*************************************************************************/
#include <malloc.h>
#include <math.h>
#include <stdio.h>

#include "mconst.h"
#include "merror.h"
#include "mfile.h"
#include "mhalfsp.h"
#include "mlocate.h"
#include "mqueue.h"
#include "msudsini.h"
#include "mutils.h"


/*************************************************************************
                               GLOBAL DATA
*************************************************************************/
PRIVATE Q_QUEUE location_queue;
PRIVATE Q_LINK * head_ptr;
PRIVATE unsigned int location_count;
PRIVATE FLAG location_flag;
PRIVATE FLAG location_enabled;


/*-----------------------------------------------------------------------*
 *                            adjust_queue                               *
 *-----------------------------------------------------------------------*/
/* Create and save a data link in a data queue.                          */

PRIVATE
void adjust_queue (location)
Q_LOCATION far * location;
{
   Q_TYPE type;

   type.location = location;
   q_enqueue (&location_queue, type);
   if (location_count == MAX_LOCATIONS) {
      if (q_dequeue (&location_queue, &type))
         _ffree (type.location);
   } else location_count++;
}

/*-----------------------------------------------------------------------*
 *                          l_display_location                           *
 *-----------------------------------------------------------------------*/
/* Write location to stream.                                             */

PUBLIC
void l_display_location (stream)
FILE * stream;
{
   Q_LOCATION far * location;
   char * timeline;

   if (location_flag) {
      location = location_queue.tail->type.location;

      fprintf (stream, "DATE        ORIGIN        LAT       LONG       DEPTH   MAG
      timeline = u_build_timeline (location->info.orgtime, 2);
      fprintf (stream, "%s  %6.3lf %c  %7.3lf %c  %7.2lf ",
               timeline, fabs(location->info.or_lat),
               (location->info.or_lat >= 0.0) ? 'N' : 'S',
               fabs(location->info.or_long),
               (location->info.or_long >= 0.0) ? 'E' : 'W',
               location->info.depth);
      if (location->info.magnitude != atof(NODATA))
         fprintf (stream, "%4.1f", location->info.magnitude);
      else fprintf (stream, "    ");

      fprintf (stream, "%2d %4.1f\n", location->info.num_stats,
               location->info.res_rms);

      free (timeline);
   } else fprintf (stream, "\n*** Unable to locate event ***\n");
}

/*-----------------------------------------------------------------------*
 *                        l_get_first_location                           *
 *-----------------------------------------------------------------------*/
/* Get first location from location queue.                               */

PUBLIC
Q_LOCATION far * l_get_first_location ()
{
   head_ptr = location_queue.head;
   if (head_ptr != NULL)
      return (head_ptr->type.location);
   else return (NULL);
}

/*-----------------------------------------------------------------------*
 *                            l_get_location                             *
 *-----------------------------------------------------------------------*/
/* Get location if event was located.                                    */

PUBLIC
Q_LOCATION far * l_get_location ()
{
   if (location_flag && location_queue.tail != NULL)
      return (location_queue.tail->type.location);
   else return (NULL);
}

/*-----------------------------------------------------------------------*
```

```
 *                          l_get_location_status                          *
 *-------------------------------------------------------------------------*/
/* Get location_enabled.                                                   */

PUBLIC
FLAG l_get_location_status ()
{
   return location_enabled;
}

/*-------------------------------------------------------------------------*
 *                          l_get_next_location                            *
 *-------------------------------------------------------------------------*/
/* Get next location from location queue.                                  */

PUBLIC
Q_LOCATION far * l_get_next_location ()
{
   head_ptr = head_ptr->next;
   if (head_ptr != NULL)
      return (head_ptr->type.location);
   else return (NULL);
}

/*-------------------------------------------------------------------------*
 *                             l_initialize                                *
 *-------------------------------------------------------------------------*/
/* Initialize the location module.                                         */

PUBLIC
void l_initialize ()
{
   location_enabled = LOCATION_ENABLED;
   q_initialize (&location_queue);
   location_count = 0;
   location_flag = FALSE;
   hs_initialize ();
}

/*-------------------------------------------------------------------------*
 *                             l_locate_event                              *
 *-------------------------------------------------------------------------*/
/* Locate event using halfspace algorithm.                                 */

PUBLIC
FLAG l_locate_event ()
{
   Q_LOCATION far *location_ptr;
   Q_TYPE type;

   if (! location_enabled)
      return FALSE;

   /* Initialize a new origin structure */
   location_ptr = (Q_LOCATION far *) _fmalloc (sizeof(Q_LOCATION));
   if (location_ptr == NULL) er_abort (L_NO_STORAGE);
   suds_initialize_tag (ORIGIN, &location_ptr->structtag);
   suds_initialize (ORIGIN, &location_ptr->info);

   /* Do a halfspace location */
   if (location_flag = hs_locate_event (&location_ptr->info)) {
      location_ptr->info.number = f_get_event_number();
      location_ptr->info.authority = st_get_authority();
      location_ptr->info.version = '0';
      location_ptr->info.or_status = 'a';
      location_ptr->info.program = 's';
      location_ptr->info.effective = u_timestamp();
      adjust_queue (location_ptr);
   } else _ffree (location_ptr);

   return (location_flag);
}

/*-------------------------------------------------------------------------*
 *                            l_reset_location                             *
 *-------------------------------------------------------------------------*/
/* Reset the location flag.                                                */

PUBLIC
void l_reset_location ()
{
   location_flag = FALSE;
}

/*-------------------------------------------------------------------------*
 *                          l_set_location_status                          *
 *-------------------------------------------------------------------------*/
/* Set location_enabled flag.                                              */

PUBLIC
void l_set_location_status (location_status)
FLAG location_status;
{
   location_enabled = location_status;
}

/*-------------------------------------------------------------------------*
 *                        l_toggle_location_status                         *
 *-------------------------------------------------------------------------*/
/* Toggle location_enabled flag.                                           */

PUBLIC
void l_toggle_location_status ()
{
   location_enabled = (location_enabled) ? FALSE : TRUE;
}
```

```
/* FILE: mlocate.h                                  (D. Tottingham  04/26/92)

This is an include file of defines, data structure definitions and
external declarations that are common in the mlocate module.

*/

#ifndef _MLOCATE_
#define _MLOCATE_

/************************************************************************
                                INCLUDES
************************************************************************/
#include "mconst.h"
#include "mqueue.h"


/************************************************************************
                                DEFINES
************************************************************************/
#define LOCATION_ENABLED        TRUE
#define MAX_LOCATIONS           10


/************************************************************************
                          EXTERNAL DECLARATIONS

These functions can be called from all modules that include this file.
************************************************************************/
PUBLIC void l_display_location (FILE *);
PUBLIC Q_LOCATION far * l_get_first_location ();
PUBLIC Q_LOCATION far * l_get_location ();
PUBLIC FLAG l_get_location_status ();
PUBLIC Q_LOCATION far * l_get_next_location ();
PUBLIC void l_initialize ();
PUBLIC FLAG l_locate_event ();
PUBLIC void l_reset_location ();
PUBLIC void l_set_location_status ();
PUBLIC void l_toggle_location_status ();

#endif
```

```
/* FILE: mlog.c                                    (D. Tottingham  04/26/92)

This is a collection of C functions that manage the log file for xdetect.
All functions have been written and compiled medium model.  The following
functions are included:

o_check_pathname ()             check log pathname for validity
o_initialize_params ()          initialize log pathname
o_open_logfile ()               open a log file for output
o_set_pathname ()               set log pathname
o_write_logfile ()              write string to log file stream

EXTERNAL FUNCTIONS CALLED:

er_abort ()                     display an error message then quit
u_build_date_fn ()              build a date filename: pathname\YRMNDY
u_build_timeline ()             convert abs. time to an ascii string
u_timestamp ()                  get timestamp

HISTORY:
   none

*/


/*************************************************************************
                            INCLUDE FILES
*************************************************************************/
#include <stdio.h>
#include <string.h>
#include <sys\types.h>
#include <sys\timeb.h>
#include <time.h>

#include "mconst.h"
#include "merror.h"
#include "mlog.h"
#include "mutils.h"


/*************************************************************************
                            GLOBALS
*************************************************************************/
PRIVATE char log_filename[MAX_FILENAME_LENGTH * 2];
PRIVATE char log_pathname [MAX_FILENAME_LENGTH];


/*-----------------------------------------------------------------------*
 *                          build_log_fn                                 *
 *-----------------------------------------------------------------------*/
/* Build a log filename as such: YRMNDY.LOG.                             */

PRIVATE
void build_log_fn ()
{
   struct timeb time;
   double abs_time;
   int j;

   ftime (&time);
   abs_time = u_convert_time (time);
   j = u_build_date_fn ( abs_time, log_pathname, log_filename);

   j += sprintf (log_filename+j, ".LOG");
}

/*-----------------------------------------------------------------------*
 *                      o_initialize_params                              *
 *-----------------------------------------------------------------------*/
/* Initialize log pathname.                                              */

PUBLIC
void o_initialize_params ()
{
   strncpy (log_pathname, LOG_PATHNAME, MAX_FILENAME_LENGTH);
}

/*-----------------------------------------------------------------------*
 *                          o_check_pathname                             *
 *-----------------------------------------------------------------------*/
/* Check pathname for validity.                                          */

PUBLIC
void o_check_pathname ()
{
   long ds;
   int drive;

   /* Do we have any diskspace? */
   if ((ds = u_get_diskspace (log_pathname, &drive)) == 0)
      er_abort (O_DISK_FULL);
   else if (ds == INVALID_DRIVE)
      er_abort (O_INVALID_DRIVE);
}

/*-----------------------------------------------------------------------*
 *                          o_open_logfile                               *
 *-----------------------------------------------------------------------*/
/* Open a log file for output.                                           */

PUBLIC
FILE * o_open_logfile ()
{
   FILE * stream;
   char * timeline;

   if (LOG_FILE) {
      build_log_fn ();
      if ((stream = fopen (log_filename, "a")) == NULL)
         er_abort (O_INVALID_LOGFILE);
      timeline = u_build_timeline (u_timestamp(), 0);
      fprintf (stream, "\n*** %14.12s ***\n", &timeline[11]);
      free (timeline);
      return (stream);
   } else return (NULL);
}

/*-----------------------------------------------------------------------*
 *                          o_set_pathname                               *
 *-----------------------------------------------------------------------*/
/* Set log pathname.                                                     */

PUBLIC
void o_set_pathname (pathname)
char pathname[];
{
   unsigned int length;

   strncpy (log_pathname, pathname, (MAX_FILENAME_LENGTH - 1));
   length = strlen(log_pathname);
   if (log_pathname[length-1] != ':' && log_pathname[length-1] != '\\')
      strcat (log_pathname, "\\");
}

/*-----------------------------------------------------------------------*
 *                          o_write_logfile                              *
 *-----------------------------------------------------------------------*/
/* Write string to log file stream.                                      */

PUBLIC
void o_write_logfile (str_ptr)
char * str_ptr;
{
   FILE * log_stream;
```

```
    if (LOG_FILE) {
        log_stream = o_open_logfile();
        fprintf (log_stream, "%s", str_ptr);
        fclose (log_stream);
    }
}
```

```
/* FILE: mlog.h                                    (D. Tottingham  04/26/92)

This is an include file of defines, data structure definitions and
external declarations that are common in the mlog module.

*/

#ifndef _MLOG_
#define _MLOG_

/************************************************************************
                              INCLUDES

************************************************************************/
#include "mconst.h"


/************************************************************************
                              DEFINES

************************************************************************/
#define LOG_FILE                ON
#define LOG_PATHNAME            ""


/************************************************************************
                        EXTERNAL DECLARATIONS

These functions can be called from all modules that include this file.

************************************************************************/
PUBLIC void o_initialize_params ();
PUBLIC void o_initialize_pathname ();
PUBLIC FILE * o_open_logfile ();
PUBLIC void o_set_pathname (char *);
PUBLIC void o_write_logfile (char *);

/** dtotting: new but not implemented
PUBLIC FLAG o_get_logfile_status ();
PUBLIC void o_set_logfile_status (FLAG);
PUBLIC void o_toggle_logfile_status ();
**/

#endif
```

```
/* FILE: mparse.c                                   (D. Tottingham  04/26/92)

This is a collection of C functions that parse the control file for xdetect.
All functions have been written and compiled medium model.  The following
functions are included:

p_initialize ()                 initialize parser
p_parse_commands ()             parse commands in input stream

EXTERNAL FUNCTIONS CALLED:

dm_set_PreEventTime ()          set PreEventTime constant
dt_set_channel ()               put channel in channel list
dt_set_ChannelBlocksize ()      set the channel block size
dt_set_ChannelGain ()           set the channel gain
dt_set_ClockSource ()           set clock source
dt_set_DigitizationRate ()      set the digitization rate
dt_set_TriggerSource ()         set trigger source
dsp_set_band ()                 set band limits
dsp_set_bufs_to_avg ()          set buffers to average
dsp_set_calibration_status ()   set calibration_enabled flag
dsp_set_dsp_status ()           set dsp_enabled flag
dsp_set_fft_file_status ()      set fft_file_enabled flag
dsp_set_MaxCalibrationTime ()   passes max calibration time to mdsp
e_set_bell_status ()            set bell_enabled flag
f_set_NetworkNodeId ()          set network node id
f_set_pathname ()               set pathname
f_set_preevent_status           set preevent_status flag
fr_set_block_size ()            set block size in seconds
fr_start_freerun ()             start free run event
fr_stop_freerun ()              stop free run event
h_set_trigger_status ()         set trigger status
hs_set_CriticalPhi ()           set critical phi
hs_set_HalfspaceVelocity ()     set halfspace velocity
l_set_location_status ()        set location_enabled flag
lx_initialize ()                initialize the lexer
lx_match ()                     match a token with the lookahead
lx_set_stream ()                set the input stream
o_set_pathname ()               set log pathname
r_set_reboot_status ()          set reboot_enabled flag
r_set_reboot_time ()            set reboot time
s_add_channel ()                add a channel to the home display
st_add_station ()               add station to station queue
st_set_authority ()             set authority code
t_set_CriticalAlpha ()          set CriticalAlpha constant
t_set_CriticalBeta ()           set CriticalBeta constant
t_set_CriticalGamma ()          set CriticalGamma constant
t_set_CriticalMu ()             set CriticalMu constant
t_set_CriticalNu ()             set CriticalNu constant
t_set_EventContinueCount ()     set EventContinuationCount constant
t_set_first_difference ()       turn first difference on/off
t_set_LTA_lower_bound           set LTA lower bound
t_set_LTAWindow ()              set length of long-term average
t_set_MaxEventTime ()           set MaxEventTime constant
t_set_MinEventTime ()           set MinEventTime constant
t_set_STAWindow ()              set length of short-term average
t_set_Trigger ()                set trigger status for given channel
t_set_TriggerConfirmCount ()    set TriggerConfirmationCount constant
t_set_TriggerTimeLimit ()       set TriggerTimeLimit constant
t_set_trigger_status ()         set trigger_enabled flag

HISTORY:
   none

*/


/*************************************************************************
                            INCLUDE FILES

*************************************************************************/
#include <ctype.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <atldefs.h>

#include "mconst.h"
#include "mdemux.h"
#include "mdsp.h"
#include "mdt28xx.h"
#include "mevent.h"
#include "mfile.h"
#include "mfreerun.h"
#include "mhalfsp.h"
#include "mlexer.h"
#include "mlog.h"
#include "mlocate.h"
#include "mparse.h"
#include "mreboot.h"
#include "mscrnhdr.h"
#include "mstation.h"
#include "mtrigger.h"
#include "xdetect.h"


/*-----------------------------------------------------------------------*
 *                          process_bandlimits                           *
 *-----------------------------------------------------------------------*/
/* Process band limits.                                                  */

PRIVATE
void process_bandlimits ()
{
   float high, low;

   lx_match (',');
   lx_match (C_BAND_LIMITS);
   lx_match ('=');
   lx_match ('(');

   while (lex_info.lookahead != ')') {
      lx_match ('(');
      if (lex_info.lookahead == C_CONSTANT)
         low = atof(lex_info.lexeme);
      lx_match (C_CONSTANT);
      lx_match (',');
      if (lex_info.lookahead == C_CONSTANT) {
         high = atof(lex_info.lexeme);
         dsp_set_band (low, high);
      }
      lx_match (C_CONSTANT);
      lx_match (')');
      if (lex_info.lookahead != ')')
         lx_match (',');
   }
   lx_match (')');
}

/*-----------------------------------------------------------------------*
 *                          process_station                              *
 *-----------------------------------------------------------------------*/
/* Process station command.                                              */

PRIVATE
void process_station ()
{
   char network[MAXLEX], st_name[MAXLEX], component;
   FLAG valid_definition, trigger, display;
   unsigned int channel, gain;

   /* Initialize defaults */
   valid_definition = TRUE;
   gain = 0;
   trigger = ON;
```

```
   display = OFF;

   /* Process channel number */
   channel = atoi(lex_info.lexeme);
   lx_match (C_CONSTANT);
   lx_match (',');

   /* Process station id */
   lx_match (C_STNAME);
   lx_match ('=');
   if (lex_info.lookahead == C_STRING ||
       lex_info.lookahead == C_ID)
      strcpy (st_name, lex_info.lexeme);
   else valid_definition = FALSE;
   if (lex_info.lookahead == C_STRING)
      lx_match (C_STRING);
   else lx_match (C_ID);
   lx_match (',');

   /* Process component */
   lx_match (C_COMPONENT);
   lx_match ('=');
   if (lex_info.lookahead == C_CHARACTER ||
       lex_info.lookahead == C_ID)
      component = lex_info.lexeme[0];
   else valid_definition = FALSE;
   if (lex_info.lookahead == C_ID)
      lx_match (C_ID);
   else lx_match (C_CHARACTER);

   /* Process optional parameters */
   while (lex_info.lookahead != ';') {
      lx_match (',');
      switch (lex_info.lookahead) {
         case C_DISPLAY:
            lx_match (C_DISPLAY);
            lx_match ('=');
            if (lex_info.lookahead == C_ON || lex_info.lookahead == C_OFF) {
               display = (lex_info.lookahead == C_ON) ? ON : OFF;
               lx_match (lex_info.lookahead);
            } else lx_match (C_ON);
            break;

         case C_GAIN:
            lx_match (C_GAIN);
            lx_match ('=');
            if (lex_info.lookahead == C_CONSTANT)
               gain = atoi(lex_info.lexeme);
            lx_match (C_CONSTANT);
            break;

         case C_TRIGGER:
            lx_match (C_TRIGGER);
            lx_match ('=');
            if (lex_info.lookahead == C_ON || lex_info.lookahead == C_OFF) {
               trigger = (lex_info.lookahead == C_ON) ? ON : OFF;
               lx_match (lex_info.lookahead);
            } else lx_match (C_ON);
            break;
      }
   }
   if (valid_definition) {
      if (df_set_channel (channel, gain)) {
         st_add_station (st_name, component, channel);
         t_set_trigger (channel, trigger);
         if (display) s_add_channel (channel);
      }
   }
}

/*-------------------------------------------------------------------------*
 *                            p_initialize                                 *
 *-------------------------------------------------------------------------*/
/* Initialize parser.                                                      */

PUBLIC
void p_initialize ()
{
   lx_initialize();
}

/*-------------------------------------------------------------------------*
 *                            p_parse_commands                             *
 *-------------------------------------------------------------------------*/
/* Parse commands in input stream.                                         */

PUBLIC
void p_parse_commands (stream)
FILE * stream;
{
   FLAG spectral_status, fft_file_status;
   int hour, minute, second;
   double f;

   lx_set_stream (stream);
   lx_match (lex_info.lookahead);

   spectral_status = fft_file_status = FALSE;
   while (lex_info.lookahead != C_DONE) {
      switch (lex_info.lookahead) {
         case C_AUTHORITY:
            lx_match (C_AUTHORITY);
            lx_match ('=');
            if (lex_info.lookahead == C_CONSTANT)
               st_set_authority (atoi(lex_info.lexeme));
            lx_match (C_CONSTANT);
            break;

         case C_AUTO_FREERUN:
            lx_match (C_AUTO_FREERUN);
            lx_match ('=');
            switch (lex_info.lookahead) {
               case C_ON:
                  fr_start_freerun();
                  lx_match (C_ON);
                  break;

               case C_OFF:
                  fr_stop_freerun();
                  lx_match (C_OFF);
                  break;

               default:
                  lx_match (C_ON);
            }
            break;

         case C_AUTO_LOCATION:
            lx_match (C_AUTO_LOCATION);
            lx_match ('=');
            if (lex_info.lookahead == C_ON || lex_info.lookahead == C_OFF) {
               l_set_location_status ((lex_info.lookahead == C_ON) ? ON : OFF);
               lx_match (lex_info.lookahead);
            } else lx_match (C_ON);
            break;

         case C_AUTO_REBOOT:
            lx_match (C_AUTO_REBOOT);
            lx_match ('=');
            if (lex_info.lookahead == C_ON || lex_info.lookahead == C_OFF) {
               r_set_reboot_status ((lex_info.lookahead == C_ON) ? ON : OFF);
               lx_match (lex_info.lookahead);
            } else lx_match (C_ON);

            if (lex_info.lookahead != ';') {
               lx_match (',');
```

```
            lx_match (C_TIME);
            lx_match ('=');
            hour = atoi(lex_info.lexeme);
            lx_match (C_CONSTANT);
            lx_match (':');
            minute = atoi(lex_info.lexeme);
            lx_match (C_CONSTANT);
            lx_match (':');
            second = atoi(lex_info.lexeme);
            lx_match (C_CONSTANT);
            r_set_reboot_time (hour, minute, second);
         }
         break;

      case C_AUTO_TRIGGER:
         lx_match (C_AUTO_TRIGGER);
         lx_match ('=');
         if (lex_info.lookahead == C_ON || lex_info.lookahead == C_OFF) {
            h_set_trigger_status ((lex_info.lookahead == C_ON) ? ON : OFF);
            t_set_trigger_status ((lex_info.lookahead == C_ON) ? ON : OFF);
            lx_match (lex_info.lookahead);
         } else lx_match (C_ON);
         break;

      case C_BAND_RECORDING:
         lx_match (C_BAND_RECORDING);
         lx_match ('=');
         if (lex_info.lookahead == C_ON || lex_info.lookahead == C_OFF) {
            fft_file_status = (lex_info.lookahead == C_ON) ? ON : OFF;
            dsp_set_fft_file_status (fft_file_status);
            lx_match (lex_info.lookahead);
            if (lex_info.lookahead == ',')
               process_bandlimits ();
         } else lx_match (C_ON);
         break;

      case C_BUFFERS_TO_AVERAGE:
         lx_match (C_BUFFERS_TO_AVERAGE);
         lx_match ('=');
         if (lex_info.lookahead == C_CONSTANT)
            dsp_set_bufs_to_avg (atoi(lex_info.lexeme));
         lx_match (C_CONSTANT);
         break;

      case C_CALIBRATION_RECORDING:
         lx_match (C_CALIBRATION_RECORDING);
         lx_match ('=');
         if (lex_info.lookahead == C_ON || lex_info.lookahead == C_OFF) {
            spectral_status = (lex_info.lookahead == C_ON) ? ON : OFF;
            dsp_set_spectral_status (spectral_status);
            lx_match (lex_info.lookahead);
         } else lx_match (C_ON);
         break;

      case C_CH:
         lx_match (C_CH);
         lx_match ('=');
         if (lex_info.lookahead == C_CONSTANT) {
            process_station ();
         } else lx_match (C_CONSTANT);
         break;

      case C_CHANNEL_BLOCKSIZE:
         lx_match (C_CHANNEL_BLOCKSIZE);
         lx_match ('=');
         if (lex_info.lookahead == C_CONSTANT)
            dt_set_ChannelBlocksize (atoi(lex_info.lexeme));
         lx_match (C_CONSTANT);
         break;

      case C_CHANNEL_GAIN:
         lx_match (C_CHANNEL_GAIN);
         lx_match ('=');
         if (lex_info.lookahead == C_CONSTANT)
            dt_set_ChannelGain (atoi(lex_info.lexeme));
         lx_match (C_CONSTANT);
         break;

      case C_CLOCK_SOURCE:
         lx_match (C_CLOCK_SOURCE);
         lx_match ('=');
         if (lex_info.lookahead == C_INTERNAL || lex_info.lookahead == C_EXTERNAL
            dt_set_ClockSource ((lex_info.lookahead == C_INTERNAL) ? INTERNAL_CLO
                                EXTERNAL_CLOCK);
            lx_match (lex_info.lookahead);
         } else lx_match (C_INTERNAL);
         break;

      case C_CRITICAL_ALPHA:
         lx_match (C_CRITICAL_ALPHA);
         lx_match ('=');
         if (lex_info.lookahead == C_CONSTANT)
            t_set_CriticalAlpha (atoi(lex_info.lexeme));
         lx_match (C_CONSTANT);
         break;

      case C_CRITICAL_BETA:
         lx_match (C_CRITICAL_BETA);
         lx_match ('=');
         if (lex_info.lookahead == C_CONSTANT)
            t_set_CriticalBeta (atoi(lex_info.lexeme));
         lx_match (C_CONSTANT);
         break;

      case C_CRITICAL_GAMMA:
         lx_match (C_CRITICAL_GAMMA);
         lx_match ('=');
         if (lex_info.lookahead == C_CONSTANT)
            t_set_CriticalGamma (atoi(lex_info.lexeme));
         lx_match (C_CONSTANT);
         break;

      case C_CRITICAL_MU:
         lx_match (C_CRITICAL_MU);
         lx_match ('=');
         if (lex_info.lookahead == C_CONSTANT)
            t_set_CriticalMu (atoi(lex_info.lexeme));
         lx_match (C_CONSTANT);
         break;

      case C_CRITICAL_NU:
         lx_match (C_CRITICAL_NU);
         lx_match ('=');
         if (lex_info.lookahead == C_CONSTANT)
            t_set_CriticalNu (atoi(lex_info.lexeme));
         lx_match (C_CONSTANT);
         break;

      case C_CRITICAL_PHI:
         lx_match (C_CRITICAL_PHI);
         lx_match ('=');
         if (lex_info.lookahead == C_CONSTANT)
            hs_set_CriticalPhi (atoi(lex_info.lexeme));
         lx_match (C_CONSTANT);
         break;

      case C_DIGITIZATION_RATE:
         lx_match (C_DIGITIZATION_RATE);
         lx_match ('=');
         if (lex_info.lookahead == C_CONSTANT)
            dt_set_DigitizationRate (f = atof(lex_info.lexeme));
         lx_match (C_CONSTANT);
         break;

      case C_EVENT_ALERT_BELL:
         lx_match (C_EVENT_ALERT_BELL);
```

```
      lx_match ('=');
      if (lex_info.lookahead == C_ON || lex_info.lookahead == C_OFF) {
         e_set_bell_status ((lex_info.lookahead == C_ON) ? ON : OFF);
         lx_match (lex_info.lookahead);
      } else lx_match (C_ON);
      break;

   case C_EVENT_CONTINUE_COUNT:
      lx_match (C_EVENT_CONTINUE_COUNT);
      lx_match ('=');
      if (lex_info.lookahead == C_CONSTANT)
         t_set_EventContinueCount (atoi(lex_info.lexeme));
      lx_match (C_CONSTANT);
      break;

   case C_FIRSTDIFF:
      lx_match (C_FIRSTDIFF);
      lx_match ('=');
      if (lex_info.lookahead == C_ON || lex_info.lookahead == C_OFF) {
         t_set_first_difference ((lex_info.lookahead == C_ON) ? ON : OFF);
         lx_match (lex_info.lookahead);
      } else lx_match (C_ON);
      break;

   case C_FREERUN_BLOCK_TIME:
      lx_match (C_FREERUN_BLOCK_TIME);
      lx_match ('=');
      if (lex_info.lookahead == C_CONSTANT)
         fr_set_block_size (atof(lex_info.lexeme));
      lx_match (C_CONSTANT);
      break;

   case C_HALFSPACE_VELOCITY:
      lx_match (C_HALFSPACE_VELOCITY);
      lx_match ('=');
      if (lex_info.lookahead == C_CONSTANT)
         hs_set_HalfspaceVelocity (atof(lex_info.lexeme));
      lx_match (C_CONSTANT);
      break;

   case C_LOG_PATHNAME:
      lx_match (C_LOG_PATHNAME);
      lx_match ('=');
      if (lex_info.lookahead == C_STRING) {
         printf ("WARNING: Alternate log file pathname is no longer being supporte
         /* o_set_pathname (lex_info.lexeme); */
      }
      lx_match (C_STRING);
      break;

   case C_LTA_LOWER_BOUND:
      lx_match (C_LTA_LOWER_BOUND);
      lx_match ('=');
      if (lex_info.lookahead == C_CONSTANT)
         t_set_LTA_lower_bound (atoi(lex_info.lexeme));
      lx_match (C_CONSTANT);
      break;

   case C_LTA_WINDOW:
      lx_match (C_LTA_WINDOW);
      lx_match ('=');
      if (lex_info.lookahead == C_CONSTANT)
         t_set_LTAWindow (atoi(lex_info.lexeme));
      lx_match (C_CONSTANT);
      break;

   case C_MAX_CALIBRATION_TIME:
      lx_match (C_MAX_CALIBRATION_TIME);
      lx_match ('=');
      if (lex_info.lookahead == C_CONSTANT)
         dsp_set_MaxCalibrationTime (atof(lex_info.lexeme));
      lx_match (C_CONSTANT);
      break;

   case C_MAX_EVENT_TIME:
      lx_match (C_MAX_EVENT_TIME);
      lx_match ('=');
      if (lex_info.lookahead == C_CONSTANT)
         t_set_MaxEventTime (atof(lex_info.lexeme));
      lx_match (C_CONSTANT);
      break;

   case C_MIN_EVENT_TIME:
      lx_match (C_MIN_EVENT_TIME);
      lx_match ('=');
      if (lex_info.lookahead == C_CONSTANT)
         t_set_MinEventTime (atof(lex_info.lexeme));
      lx_match (C_CONSTANT);
      break;

   case C_NETWORK:
      lx_match (C_NETWORK);
      lx_match ('=');
      if (lex_info.lookahead == C_STRING ||
          lex_info.lookahead == C_ID)
         st_set_netwname (lex_info.lexeme);
      if (lex_info.lookahead == C_STRING)
         lx_match (C_STRING);
      else lx_match (C_ID);
      break;

   case C_NETWORK_NODE_ID:
      lx_match (C_NETWORK_NODE_ID);
      lx_match ('=');
      if (lex_info.lookahead == C_STRING ||
          lex_info.lookahead == C_ID)
         f_set_NetworkNodeId (((char far *)lex_info.lexeme));
      if (lex_info.lookahead == C_STRING)
         lx_match (C_STRING);
      else lx_match (C_ID);
      break;

   case C_PATHNAME:
      lx_match (C_PATHNAME);
      lx_match ('=');
      if (lex_info.lookahead == C_STRING) {
         f_set_pathname (lex_info.lexeme);
         f_get_pathname();
      }
      lx_match (C_STRING);
      break;

   case C_PRE_EVENT_TIME:
      lx_match (C_PRE_EVENT_TIME);
      lx_match ('=');
      if (lex_info.lookahead == C_CONSTANT)
         dm_set_PreEventTime (atof(lex_info.lexeme));
      lx_match (C_CONSTANT);
      break;

   case C_PREEVENT_RECORDING:
      lx_match (C_PREEVENT_RECORDING);
      lx_match ('=');
      if (lex_info.lookahead == C_ON || lex_info.lookahead == C_OFF) {
         f_set_preevent_status ((lex_info.lookahead == C_ON) ? ON : OFF);
         lx_match (lex_info.lookahead);
      } else lx_match (C_ON);
      break;

   case C_STA_WINDOW:
      lx_match (C_STA_WINDOW);
      lx_match ('=');
      if (lex_info.lookahead == C_CONSTANT)
         t_set_STAWindow (atoi(lex_info.lexeme));
      lx_match (C_CONSTANT);
      break;
```

```
      case C_TRIGGER_CONFIRM_COUNT:
         lx_match (C_TRIGGER_CONFIRM_COUNT);
         lx_match ('=');
         if (lex_info.lookahead == C_CONSTANT)
            t_set_TriggerConfirmCount (atoi(lex_info.lexeme));
         lx_match (C_CONSTANT);
         break;

      case C_TRIGGER_SOURCE:
         lx_match (C_TRIGGER_SOURCE);
         lx_match ('=');
         if (lex_info.lookahead == C_INTERNAL || lex_info.lookahead == C_EXTERNAL) {
            dt_set_TriggerSource ((lex_info.lookahead == C_INTERNAL) ? INTERNAL_TRIGG
                                  EXTERNAL_TRIGGER);
            lx_match (lex_info.lookahead);
         } else lx_match (C_INTERNAL);
         break;

      case C_TRIGGER_TIME_LIMIT:
         lx_match (C_TRIGGER_TIME_LIMIT);
         lx_match ('=');
         if (lex_info.lookahead == C_CONSTANT)
            t_set_TriggerTimeLimit (atof(lex_info.lexeme));
         lx_match (C_CONSTANT);
         break;
   }
lx_match (';');
}
dsp_set_dsp_status (spectral_status || fft_file_status);
```

```
/* FILE: mparse.h                                  (D. Tottingham  04/26/92)

This is an include file of defines, data structure definitions and
external declarations that are common in the mparse module.

*/

#ifndef _MPARSE_
#define _MPARSE_

/************************************************************************
                                INCLUDES

************************************************************************/
#include "mconst.h"


/************************************************************************
                          EXTERNAL DECLARATIONS

These functions can be called from all modules that include this file.

************************************************************************/
PUBLIC void p_initialize ();
PUBLIC void p_parse_commands (FILE *);

#endif
```

```
/* FILE: mpick.c                                    (D. Tottingham  04/26/92)

This is a collection of C helper functions that manage a real-time picker for
xdetect.  All functions have been written and compiled medium model.  The
following functions are included:

pk_get_first_arrival ()          get first p-arrival
pk_get_first_magnitude ()        get first magnitude
pk_get_last_arrival ()           get last p-arrival
pk_get_last_magnitude ()         get last magnitude
pk_get_next_arrival ()           get next p-arrival
pk_get_next_magnitude ()         get next magnitude
pk_get_narrivals ()              get number of arrivals
pk_get_nmagnitudes ()            get number of magnitudes
pk_initialize ()                 initialize pick module
pk_pick_arrivals ()              pick arrivals using trigger information
pk_pick_magnitudes ()            pick magnitudes using coda duration
pk_reset_picks ()                reset pick queue

EXTERNAL FUNCTIONS CALLED:

er_abort ()                      display an error message then quit
q_dequeue ()                     dequeue a data link from a data queue
q_enqueue ()                     enqueue a data link on a data queue
q_initialize ()                  initialize a data queue
q_insert_link ()                 insert a data link into a data queue
suds_initialize ()               initialize a suds structure
suds_initialize_tag ()           initialize a suds structtag
st_get_authority ()              get authority
st_get_station ()                get station from station queue
t_get_head_trigger ()            get trigger information at head of channel queue
t_get_next_trigger ()            get next trigger from channel queue
u_timestamp ()                   get timestamp

HISTORY:
   none

*/


/*************************************************************************
                              INCLUDE FILES
*************************************************************************/
#include <malloc.h>
#include <stdio.h>
#include <stdlib.h>

#include "mconst.h"
#include "mdemux.h"
#include "merror.h"
#include "mqueue.h"
#include "msudsini.h"
#include "mstation.h"
#include "mtrigger.h"
#include "mutils.h"
#include "xdetect.h"


/*************************************************************************
                              GLOBAL DATA
*************************************************************************/
PRIVATE Q_QUEUE arrival_queue, magnitude_queue;
PRIVATE Q_LINK * arrival_ptr, * magnitude_ptr;
PRIVATE unsigned int n_arrivals, n_magnitudes;


/*-----------------------------------------------------------------------*
 *                            insert_arrival                             *
 *-----------------------------------------------------------------------*/
/* Insert p-arrival in arrival queue based on arrival time.              */

PRIVATE
void insert_arrival (arrival)
Q_ARRIVAL far * arrival;
{
   Q_LINK *current, *previous;
   Q_TYPE type;
   FLAG done;

   type.arrival = arrival;
   previous = NULL;
   for (current = arrival_queue.head, done = FALSE; !done && current != NULL;
        current = current->next) {
      if (arrival->info.time < current->type.arrival->info.time) {
         q_insert_link (&arrival_queue, previous, type);
         done = TRUE;
      } else previous = current;
   }
   if (!done) q_enqueue (&arrival_queue, type);
}

/*-----------------------------------------------------------------------*
 *                            process_coda                               *
 *-----------------------------------------------------------------------*/
/* Compute signal value for given channel (specified in the arrival) and
   check against noise.  If signal/noise < 2, compute coda duration and
   return it to the caller.                                              */

PRIVATE
double process_coda (arrival_ptr, buffer_ptr)
Q_LINK *arrival_ptr;
Q_BUFFER far * buffer_ptr;
{
   Q_ARRIVAL far * arrival;
   long channel_offset, start_offset, i, max_index;
   double deltat;
   int ns;

   arrival = arrival_ptr->type.arrival;

   start_offset = 0;
   if ((deltat = arrival->info.time - buffer_ptr->info.begintime) > 0.0)
      start_offset = (long) (deltat * buffer_ptr->info.dig_rate);

   max_index = 0;
   arrival->signal = 0;
   channel_offset = buffer_ptr->info.blocksize * arrival->channel;
   for (i = (channel_offset + start_offset); i < (buffer_ptr->info.blocksize + chann
      ns = abs (buffer_ptr->data[i] - buffer_ptr->info.dc_offset);
      if (ns > arrival->signal) {
         arrival->signal = ns;
         max_index = i;
      }
      if ((arrival->signal / arrival->noise) > 2)
         return (0.0);
   }
   deltat = ((double)(max_index - channel_offset)) / buffer_ptr->info.dig_rate;
   return ( deltat + buffer_ptr->info.begintime - arrival->info.time );
}

/*-----------------------------------------------------------------------*
 *                          pk_get_first_arrival                         *
 *-----------------------------------------------------------------------*/
/* Get first p-arrival from arrival queue.                               */

PUBLIC
Q_ARRIVAL far * pk_get_first_arrival ()
{
   arrival_ptr = arrival_queue.head;
   if (arrival_ptr != NULL)
      return (arrival_ptr->type.arrival);
   else return (NULL);
}
```

```
/*----------------------------------------------------------------------*
 *                        pk_get_first_magnitude                        *
 *----------------------------------------------------------------------*/
/* Get first magnitude from magnitude queue.                            */

PUBLIC
Q_MAGNITUDE far * pk_get_first_magnitude ()
{
   magnitude_ptr = magnitude_queue.head;
   if (magnitude_ptr != NULL)
      return (magnitude_ptr->type.magnitude);
   else return (NULL);
}

/*----------------------------------------------------------------------*
 *                         pk_get_last_arrival                          *
 *----------------------------------------------------------------------*/
/* Get last p-arrival from arrival queue.                               */

PUBLIC
Q_ARRIVAL far * pk_get_last_arrival ()
{
   if (arrival_queue.tail != NULL)
      return (arrival_queue.tail->type.arrival);
   else return (NULL);
}

/*----------------------------------------------------------------------*
 *                        pk_get_last_magnitude                         *
 *----------------------------------------------------------------------*/
/* Get last magnitude from magnitude queue.                             */

PUBLIC
Q_MAGNITUDE far * pk_get_last_magnitude ()
{
   if (magnitude_queue.tail != NULL)
      return (magnitude_queue.tail->type.magnitude);
   else return (NULL);
}

/*----------------------------------------------------------------------*
 *                         pk_get_next_arrival                          *
 *----------------------------------------------------------------------*/
/* Get next p-arrival from arrival queue.                               */

PUBLIC
Q_ARRIVAL far * pk_get_next_arrival ()
{
   arrival_ptr = arrival_ptr->next;
   if (arrival_ptr != NULL)
      return (arrival_ptr->type.arrival);
   else return (NULL);
}

/*----------------------------------------------------------------------*
 *                        pk_get_next_magnitude                         *
 *----------------------------------------------------------------------*/
/* Get next magnitude from magnitude queue.                             */

PUBLIC
Q_MAGNITUDE far * pk_get_next_magnitude ()
{
   magnitude_ptr = magnitude_ptr->next;
   if (magnitude_ptr != NULL)
      return (magnitude_ptr->type.magnitude);
   else return (NULL);
}

/*----------------------------------------------------------------------*
 *                         pk_get_narrivals                             *
 *----------------------------------------------------------------------*/
/* Get number of p-arrivals in arrival queue.                           */

PUBLIC
unsigned int pk_get_narrivals ()
{
   return (n_arrivals);
}

/*----------------------------------------------------------------------*
 *                         pk_get_nmagnitudes                           *
 *----------------------------------------------------------------------*/
/* Get number of magnitudes in magnitude queue.                         */

PUBLIC
unsigned int pk_get_nmagnitudes ()
{
   return (n_magnitudes);
}

/*----------------------------------------------------------------------*
 *                          pk_initialize                               *
 *----------------------------------------------------------------------*/
/* Initialize the pick module.                                          */

PUBLIC
void pk_initialize ()
{
   q_initialize (&arrival_queue);
   q_initialize (&magnitude_queue);
   n_arrivals = n_magnitudes = 0;
}

/*----------------------------------------------------------------------*
 *                          pk_pick_arrivals                            *
 *----------------------------------------------------------------------*/
/* Pick arrivals using trigger information.                             */

PUBLIC
void pk_pick_arrivals ()
{
   Q_CHANNEL far *trig_ptr;
   Q_LINK *link_ptr;
   Q_ARRIVAL far *new_arrival;
   Q_TYPE type;
   double first_arrival;

   /* Get new picks */
   for (trig_ptr = t_get_head_trigger(); trig_ptr != NULL; trig_ptr = t_get_next_tri
      if ((st_get_station(trig_ptr->channel))->in_masterlist &&
          trig_ptr->info.trig_value) {

         /* Initialize a new arrival structure */
         new_arrival = (Q_ARRIVAL far *) _fmalloc (sizeof(Q_ARRIVAL));
         if (new_arrival == NULL) er_abort (PK_NO_STORAGE);
         suds_initialize_tag (FEATURE, &new_arrival->structtag);
         suds_initialize (FEATURE, &new_arrival->info);

         /* Fill it in */
         new_arrival->channel = trig_ptr->channel;
         new_arrival->signal = trig_ptr->signal;
         new_arrival->noise = trig_ptr->noise;
         new_arrival->info.obs_phase = 50;       /* p first arrival */
         new_arrival->info.fe_name = trig_ptr->info.tr_name;
         new_arrival->info.data_source = 'r';
         new_arrival->info.tim_qual = '4';
         new_arrival->info.time = trig_ptr->info.trig_time;
         new_arrival->info.pick_authority = st_get_authority ();
         new_arrival->info.time_of_pick = u_timestamp ();

         insert_arrival (new_arrival);
         n_arrivals++;
      }
   }
```

```
   /* Compute relative arrivals based on first arrival */
   first_arrival = arrival_queue.head->type.arrival->info.time;
   for (link_ptr = arrival_queue.head; link_ptr != NULL; link_ptr = link_ptr->next)
      link_ptr->type.arrival->rel_arrival = link_ptr->type.arrival->info.time - first_ar
}

/*-----------------------------------------------------------------------*
 *                          pk_pick_magnitudes                           *
 *-----------------------------------------------------------------------*/
/* Pick magnitudes using coda duration.                                  */

PUBLIC
void pk_pick_magnitudes ()
{
   Q_BUFFER far *b_head;
   Q_LINK *a_head, *m_head;
   Q_MAGNITUDE far *new_magnitude;
   Q_TYPE type;
   double coda_duration;
   FLAG found;

   /* Check the arrival queue */
   if (arrival_queue.head == NULL) return;

   for (b_head = dm_get_first_buffer(); b_head != NULL; b_head = dm_get_next_buffer())
      for (a_head = arrival_queue.head; a_head != NULL; a_head = a_head->next) {
         for (found = FALSE, m_head = magnitude_queue.head;
              m_head != NULL && !found; m_head = m_head->next)
            if (m_head->type.magnitude->channel == a_head->type.arrival->channel)
               found = TRUE;

         if (! found && (coda_duration = process_coda (a_head, b_head)) > 0.0) {
            /* Initialize a new magnitude structure */
            new_magnitude = (Q_MAGNITUDE far *) _fmalloc (sizeof(Q_MAGNITUDE));
            if (new_magnitude == NULL) er_abort (PK_NO_STORAGE);
            suds_initialize_tag (FEATURE, &new_magnitude->structtag);
            suds_initialize (FEATURE, &new_magnitude->info);

            /* Fill it in */
            new_magnitude->channel = a_head->type.arrival->channel;
            new_magnitude->info.obs_phase = 2;      /* f finis */
            new_magnitude->info.fe_name = a_head->type.arrival->info.fe_name;
            new_magnitude->info.data_source = 'r';
            new_magnitude->info.tim_qual = '4';
            new_magnitude->info.time = coda_duration;
            new_magnitude->info.pick_authority = a_head->type.arrival->info.pick_authori
            new_magnitude->info.time_of_pick = u_timestamp ();

            type.magnitude = new_magnitude;
            q_enqueue (&magnitude_queue, type);
            n_magnitudes++;
         }
      }
}

/*-----------------------------------------------------------------------*
 *                          pk_reset_picks                               *
 *-----------------------------------------------------------------------*/
/* Reset pick queues.                                                    */

PUBLIC
void pk_reset_picks ()
{
   Q_TYPE type;

   while (q_dequeue (&arrival_queue, &type))
      _ffree (type.arrival);
   n_arrivals = 0;

   while (q_dequeue (&magnitude_queue, &type))
      _ffree (type.magnitude);
   n_magnitudes = 0;
}
```

```
/* FILE: mpick.h                                    (D. Tottingham  04/26/92)

This is an include file of defines, data structure definitions and
external declarations that are common in the mpick module.

*/

#ifndef _MPICK_
#define _MPICK_

/************************************************************************
                                INCLUDES
************************************************************************/
#include "mconst.h"
#include "mqueue.h"


/************************************************************************
                            EXTERNAL DECLARATIONS

These functions can be called from all modules that include this file.

************************************************************************/
PUBLIC Q_ARRIVAL far * pk_get_first_arrival ();
PUBLIC Q_MAGNITUDE far * pk_get_first_magnitude ();
PUBLIC Q_ARRIVAL far * pk_get_last_arrival ();
PUBLIC Q_MAGNITUDE far * pk_get_last_magnitude ();
PUBLIC Q_APRIVAL far * pk_get_next_arrival ();
PUBLIC Q_MAGNITUDE far * pk_get_next_magnitude ();
PUBLIC unsigned int pk_get_narrivals ();
PUBLIC unsigned int pk_get_nmagnitudes ();
PUBLIC void pk_initialize ();
PUBLIC void pk_pick_arrivals ();
PUBLIC void pk_pick_magnitudes ();
PUBLIC void pk_reset_picks ();

#endif
```

```
/* FILE: mqueue.c                                    (D. Tottingham  04/26/92)

This is a collection of C functions that manage the data queues for xdetect.
All functions have been written and compiled medium model. The following
functions are included:

q_delete_link ()                delete a data link from a data queue
q_dequeue ()                    dequeue a data link from a data queue
q_enqueue ()                    enqueue a data link on a data queue
q_initialize ()                 initialize a data queue
q_insert_link ()                insert a data link into a data queue

EXTERNAL FUNCTIONS CALLED:

er_abort ()                     display an error message then quit

HISTORY:
   none

*/


/*************************************************************************
                            INCLUDE FILES

*************************************************************************/
#include <malloc.h>
#include <stdio.h>

#include "mconst.h"
#include "merror.h"
#include "mqueue.h"


/*-----------------------------------------------------------------------*
 *                              dequeue                                  *
 *-----------------------------------------------------------------------*/
/* Dequeue a data link from the data queue.                              */

PRIVATE
FLAG dequeue ( queue_ptr, link_ptr_ptr )
Q_QUEUE * queue_ptr;
Q_LINK * * link_ptr_ptr;
{
   if (queue_ptr->head != NULL) {
      *link_ptr_ptr = queue_ptr->head;
      queue_ptr->head = queue_ptr->head->next;

      if (queue_ptr->head == NULL)
         queue_ptr->tail = NULL;
      return (TRUE);

   } else return (FALSE);
}

/*-----------------------------------------------------------------------*
 *                              enqueue                                  *
 *-----------------------------------------------------------------------*/
/* Enqueue a data link to the data queue.                                */

PRIVATE
void enqueue ( queue_ptr, link_ptr )
Q_QUEUE * queue_ptr;
Q_LINK * link_ptr;
{
   if (queue_ptr->tail == NULL) {
      /* First link */
      queue_ptr->tail = link_ptr;
      queue_ptr->head = link_ptr;
   } else {
      queue_ptr->tail->next = link_ptr;
      queue_ptr->tail = link_ptr;
   }
}

/*-----------------------------------------------------------------------*
 *                           q_delete_link                               *
 *-----------------------------------------------------------------------*/
/* Delete a data link from a data queue.                                 */

PUBLIC
FLAG q_delete_link (queue_ptr, current_ptr, previous_ptr, type_ptr)
Q_QUEUE * queue_ptr;
Q_LINK * current_ptr;
Q_LINK * previous_ptr;
Q_TYPE * type_ptr;
{
   Q_LINK * link_ptr;

   if (previous_ptr == NULL)
      return ( q_dequeue(queue_ptr, type_ptr));
   else {
      previous_ptr->next = current_ptr->next;
      *type_ptr = current_ptr->type;
      free (current_ptr);
      return (TRUE);
   }
}

/*-----------------------------------------------------------------------*
 *                              q_dequeue                                *
 *-----------------------------------------------------------------------*/
/* Delete a data link from a data queue.                                 */

PUBLIC
FLAG q_dequeue (queue_ptr, type_ptr)
Q_QUEUE * queue_ptr;
Q_TYPE * type_ptr;
{
   Q_LINK * link_ptr;

   if (dequeue (queue_ptr, &link_ptr)) {
      *type_ptr = link_ptr->type;
      free (link_ptr);
      return (TRUE);
   } else return (FALSE);
}

/*-----------------------------------------------------------------------*
 *                              q_enqueue                                *
 *-----------------------------------------------------------------------*/
/* Create and save a data link in a data queue.                          */

PUBLIC
void q_enqueue (queue_ptr, type)
Q_QUEUE * queue_ptr;
Q_TYPE type;
{
   Q_LINK * new_link_ptr;

   new_link_ptr = (Q_LINK *) calloc (1, sizeof(Q_LINK));

   if (new_link_ptr == NULL) er_abort (Q_NO_STORAGE);

   new_link_ptr->type = type;
   new_link_ptr->next = NULL;
   enqueue (queue_ptr, new_link_ptr);
}

/*-----------------------------------------------------------------------*
 *                            q_initialize                               *
 *-----------------------------------------------------------------------*/
/* Initialize a data queue.                                              */

PUBLIC
void q_initialize ( queue_ptr )
```

```
Q_QUEUE * queue_ptr;
{
   queue_ptr->head = NULL;
   queue_ptr->tail = NULL;
}

/*-------------------------------------------------------------------------*
 *                          q_insert_link                                  *
 *-------------------------------------------------------------------------*/
/* Insert a data link into a data queue.                                   */

PUBLIC
void q_insert_link (queue_ptr, previous_ptr, type)
Q_QUEUE * queue_ptr;
Q_LINK * previous_ptr;
Q_TYPE type;
{
   Q_LINK * link_ptr;

   link_ptr = (Q_LINK *) calloc (1, sizeof(Q_LINK));
   if (link_ptr == NULL) er_abort (Q_NO_STORAGE);
   link_ptr->type = type;

   if (previous_ptr == NULL) {
      link_ptr->next = queue_ptr->head;
      queue_ptr->head = link_ptr;
   } else {
      link_ptr->next = previous_ptr->next;
      previous_ptr->next = link_ptr;
   }
}
```

```
/* FILE: mqueue.h                                  (D. Tottingham  04/26/92)

This is an include file of defines, data structure definitions and
external declarations that are common in the mqueue module.

*/

#ifndef _MQUEUE_
#define _MQUEUE_

/*************************************************************************
                                INCLUDES
*************************************************************************/
#include "mconst.h"
#include "mlatlon.h"
#include "powrstor.h"
#include "suds.h"


/*************************************************************************
                          STRUCTURE DEFINITIONS

The following structure definitions are included here so that all
modules can have access to them.

*************************************************************************/
typedef struct {
        unsigned int channel;
        double rel_arrival;
        int signal;
        int noise;
        SUDS_STRUCTTAG structtag;
        SUDS_FEATURE info;
} Q_ARRIVAL;

typedef struct {
        int far * data;
        unsigned int buffer_status;
        unsigned int dynamic_range;
        unsigned int buffer_number;
        unsigned int buffer_size;
        double seconds_in_buffer;
        SUDS_STRUCTTAG structtag;
        SUDS_MUXDATA info;
        struct STORheap far * ST_hesp_ptr;  /* temporarily a far pointer */
        FLAG bank_switched;
} Q_BUFFER;

typedef struct {
        unsigned int channel;
        int signal;
        int noise;
        SUDS_STRUCTTAG structtag;
        SUDS_TRIGGERS info;
} Q_CHANNEL;

typedef struct {
        SUDS_STRUCTTAG structtag;
        SUDS_ORIGIN info;
} Q_LOCATION;

typedef struct {
        unsigned int channel;
        SUDS_STRUCTTAG structtag;
        SUDS_FEATURE info;
} Q_MAGNITUDE;

typedef struct {
        double x, y;
        double xsq, ysq;
        LL_LATLON mean_latlon;
        SUDS_STRUCTTAG structtag;
        SUDS_STATIONCOMP info;
        FLAG in_masterlist;
} Q_STATION;

typedef struct {
        double trigger_time;
        Q_CHANNEL far * qch;
        FLAG trigger_flag;
} Q_TRIGGER;

typedef union {
        Q_ARRIVAL far * arrival;
        Q_BUFFER  far * buffer;
        Q_CHANNEL far * channel;
        Q_LOCATION far * location;
        Q_MAGNITUDE far * magnitude;
        Q_STATION far * station;
        Q_TRIGGER * trigger;
} Q_TYPE;

typedef struct q_link {
        Q_TYPE type;
        struct q_link * next;
} Q_LINK;

typedef struct {
        Q_LINK * head;
        Q_LINK * tail;
} Q_QUEUE;


/*************************************************************************
                          EXTERNAL DECLARATIONS

These functions can be called from all modules that include this file.

*************************************************************************/
PUBLIC FLAG q_delete_link (Q_QUEUE *, Q_LINK *, Q_LINK *, Q_TYPE *);
PUBLIC FLAG q_dequeue (Q_QUEUE *, Q_TYPE *);
PUBLIC void q_enqueue (Q_QUEUE *, Q_TYPE);
PUBLIC void q_initialize (Q_QUEUE *);
PUBLIC void q_insert_link (Q_QUEUE *, Q_LINK *, Q_TYPE);

#endif
```

```
/* FILE: mreboot.c                                    (D. Tottingham  04/26/92)

This is a collection of C functions that provide a reboot facility for xdetect.
All functions have been written and compiled medium model.  The following
functions are included:

r_check_time ()                 compare time with reboot time
r_get_reboot_status ()          get reboot_enabled
r_get_reboot_time ()            get reboot time
r_initialize_reboot ()          initialize reboot structure
r_set_reboot_time ()            set reboot time
r_set_reboot_status ()          set reboot_enabled flag
r_toggle_reboot_status ()       toggle reboot_enabled flag

EXTERNAL FUNCTIONS CALLED:

dm_get_current_time ()          get current time

HISTORY:
   none

*/



/************************************************************************
                              INCLUDE FILES
************************************************************************/
#include <stdio.h>
#include <time.h>

#include "mconst.h"
#include "mdemux.h"
#include "mreboot.h"


/************************************************************************
                              GLOBALS
************************************************************************/
PRIVATE R_INFO reboot_info, *r = &reboot_info;
PRIVATE FLAG reboot_enabled;


/*----------------------------------------------------------------------*
 *                          r_check_time                                *
 *----------------------------------------------------------------------*/
/* Compare time with reboot time.  If reboot time has been reached,
   do a warm boot.                                                      */

PUBLIC
FLAG r_check_time ()
{
   static int first_time = TRUE;
   long time;
   struct tm *current_time;

   if (! reboot_enabled)
      return FALSE;

   if (!first_time)
      r->previous_time = r->current_time;

   /* Convert buffer_start_time into current_time */
   time = (long) (dm_get_current_time());
   current_time = gmtime (&time);
   r->current_time = ((double) current_time->tm_hour) * 3600 +
                     current_time->tm_min * 60 + current_time->tm_sec;

   if (!first_time) {
      /* Is it time to reset the system?  */
      if ((r->current_time >= r->reboot_time) &&
          (r->previous_time <= r->reboot_time))
         boot (WARM);

      /* Special case at midnight */
      } else if ((r->current_time >= r->reboot_time) &&
                 (r->previous_time >= r->reboot_time))
         boot (WARM);
   } else first_time = FALSE;
   return ( FALSE);
}

/*----------------------------------------------------------------------*
 *                        r_get_reboot_status                           *
 *----------------------------------------------------------------------*/
/* Get reboot_enabled.                                                  */

PUBLIC
FLAG r_get_reboot_status ()
{
   return reboot_enabled;
}

/*----------------------------------------------------------------------*
 *                        r_get_reboot_time                             *
 *----------------------------------------------------------------------*/
/* Get reboot time.                                                     */

PUBLIC
char * r_get_reboot_time ()
{
   static char out_str[10];

   sprintf (out_str, "%02d:%02d:%02d", r->hour, r->minute, r->second);
   return (out_str);
}

/*----------------------------------------------------------------------*
 *                        r_initialize_reboot                           *
 *----------------------------------------------------------------------*/
/* Initialize reboot structure.                                         */

PUBLIC
void r_initialize_reboot ()
{
   reboot_enabled = REBOOT_ENABLED;
   r->reboot_time = REBOOT_TIME;
}

/*----------------------------------------------------------------------*
 *                        r_set_reboot_time                             *
 *----------------------------------------------------------------------*/
/* Set reboot time.                                                     */

PUBLIC
void r_set_reboot_time (hour, minute, second)
int hour;
int minute;
int second;
{
   r->hour = hour;
   r->minute = minute;
   r->second = second;
   r->reboot_time = ((double) hour) * 3600 + r->minute * 60 + r->second;
}

/*----------------------------------------------------------------------*
 *                        r_set_reboot_status                           *
 *----------------------------------------------------------------------*/
/* Set reboot_enabled flag.                                             */

PUBLIC
void r_set_reboot_status (reboot_status)
```

```
FLAG reboot_status;
{
   reboot_enabled = reboot_status;
}

/*------------------------------------------------------------------------*
 *                        r_toggle_reboot_status                          *
 *------------------------------------------------------------------------*/
/* Toggle reboot_enabled flag.                                            */

PUBLIC
void r_toggle_reboot_status ()
{
   reboot_enabled = (reboot_enabled) ? FALSE : TRUE;
}
```

```
/* FILE: mreboot.h                                  (D. Tottingham  04/26/92)

This is an include file of the defines, data structure definitions and
external data declarations for using the mreboot module.

*/

#ifndef _MREBOOT_
#define _MREBOOT_

/************************************************************************
                                INCLUDES

************************************************************************/
#include "mconst.h"


/************************************************************************
                                DEFINES

************************************************************************/
#define REBOOT_ENABLED          FALSE

/* Boot Constants */
#define COLD                    0
#define WARM                    1

/* Reboot Constants */
#define REBOOT_TIME             0               /* seconds from midnight */


/************************************************************************
                          STRUCTURE DEFINITIONS

The following structure definitions are included in mreboot.h, so that all
modules can have access to them.

************************************************************************/
typedef struct {
        double reboot_time;     /* reboot time in seconds from midnight */
        double current_time;    /* buffer_start_time in seconds from midnight */
        double previous_time;   /* previous current_time */
        int hour;
        int minute;
        int second;
} R_INFO;


/************************************************************************
                          EXTERNAL DECLARATIONS

These functions can be called from all modules that include this file.

************************************************************************/
PUBLIC FLAG r_check_time ();
PUBLIC FLAG r_get_reboot_status ();
PUBLIC char * r_get_reboot_time ();
PUBLIC void r_initialize_reboot ();
PUBLIC void r_set_reboot_time (int, int, int);
PUBLIC void r_set_reboot_status (FLAG);
PUBLIC void r_toggle_reboot_status ();

#endif
```

```
/* FILE: mscreen.c                                  (D. Tottingham  04/26/92)

This is a collection of C functions that manage the screen for xdetect.
All functions have been written and compiled medium.  The following functions
are included:

s_add_channel ()                add a channel to the home display
s_clear ()                      clear the screen
s_initialize ()                 initialize the screen module
s_initialize_screen ()          initialize the graphics screen
s_display_bar ()                display the bsr
s_display_ids ()                display station id, channel, and trigger status
s_display_traces ()             display normalized traces on the screen
s_display_triggers ()           display triggers on the screen
s_reset_screen ()               reset the grsphics screen to text
s_remove_channel ()             remove a chsnnel from the home display
s_screen_on ()                  return screen flag
s_select_display ()             select the display mode
s_select_screen ()              select a screen
s_set_channelrange ()           set the channel range in the block display
s_view_help ()                  display the help screen

EXTERNAL FUNCTIONS CALLED:

dm_get_first_buffer ()          get first new demux buffer from demux queue
dm_get_next_buffer ()           get next buffer from demux queue
dsp_get_status ()               return char for spectral signal
dsp_get_spectral_channel ()     return spectral trigger channel
dt_get_channel_gain ()          get channel gain
dt_get_channel_size ()          get chsnnel block size
dt_get_scan_count ()            get scan count
er_abort ()                     display an error message then quit
o_write_logfile ()              write string to log file stream
st_build_textline ()            build s textline for the mscreen module
t_get_channel_trigger_state ()  get trigger state for given channel

HISTORY:
   none

*/


/*************************************************************************
                            INCLUDE FILES

*************************************************************************/
#include <dos.h>
#include <stdio.h>
#include <malloc.h>
#include <string.h>

#include "mconst.h"
#include "mdemux.h"
#include "mdsp.h"
#include "mdt28xx.h"
#include "merror.h"
#include "mhelp.h"
#include "mlog.h"
#include "mscreen.h"
#include "mstation.h"
#include "mtrigger.h"
#include "mutils.h"
#include "mvideo.h"


/*************************************************************************
                                GLOBALS

*************************************************************************/
PRIVATE S_DISPLAY display[N_DISPLAYS], *current_display;
PRIVATE S_HEADER header, *h = &header;
PRIVATE unsigned int old_size, current_size;
PRIVATE int far * raw_data;
PRIVATE int far * plot_data;
PRIVATE int far * old_plot_data;
PRIVATE unsigned int screen_id;
PRIVATE int display_id;
PRIVATE char textline[MAX_TEXTLINE];
PRIVATE FLAG reset_display;


/*-------------------------------------------------------------------*
 *                          build_textline                           *
 *-------------------------------------------------------------------*/
/* Construct a text string for the specified channel.  The string
   contains the channel number, station id, channel status, latitude,
   longitude, and channel gain.                                      */

PRIVATE
int build_textline (channel, textline)
int channel;
char textline[];
{
   int j;

   j = st_build_textline (channel, textline);
   j += sprintf (textline+j, (t_get_channel_trigger_state(channel)) ? " ON " : "
   j += sprintf (textline+j, "%ld", dt_get_channel_gain(channel));

   return (j);
}

/*-------------------------------------------------------------------*
 *                             clear_bar                             *
 *-------------------------------------------------------------------*/
/* Clear bar from the screen.                                        */

PRIVATE
void clear_bar ()
{
   fill ( BIG_RECT_I1, BIG_RECT_J1, BIG_RECT_I2, BIG_RECT_J2, BCKGND_COLOR);
}

/*-------------------------------------------------------------------*
 *                           clear_display                           *
 *-------------------------------------------------------------------*/
/* Clear ids, triggers, and traces from the screen.                  */

PRIVATE
void clear_display ()
{
   int i, j, k, i_increment, npoints;
   unsigned int channel_offset;

   if (current_display != NULL) {
      i_increment = (int) ((VERTICAL_PIXS) / current_display->orientation);
      i = (TOP_MARGIN) + (i_increment / 2);

      channel_offset = HORIZONTAL_PIXS - old_size;
      j = LEFT_MARGIN + channel_offset;

      for (k = 0; k < current_display->n_channels; k++, i += i_increment,
                                          channel_offset += HORIZONTAL_PIXS) {
         gputs ( i-4, 15, "                ", 15, TEXT_OFF_COLORS);
         if (old_plot_data)
            plot( i, j, &old_plot_data[channel_offset], old_size, PLOT_OFF_COLOR);
      }
      if (plot_data) {
         _ffree (plot_data); plot_data = NULL;
      }
      if (old_plot_data) {
         _ffree (old_plot_data); old_plot_data = NULL;
      }
      if (raw_data) {
```

```
        _ffree (raw_data); raw_data = NULL;
      }
   }
   old_size = current_size = 0;
   reset_display = TRUE;
}

/*-------------------------------------------------------------------------*
 *                            display_bar                                  *
 *-------------------------------------------------------------------------*/
/* Draw the channel block position indicator on the screen.  The big
   rectangle all channels.  The small rectangle represents the portion
   of and the location of the block that is currently being viewed.  The
   small rectangle is called the block position indicator.                 */

PRIVATE
void display_bar (begin_channel, n_channels, max_channel)
unsigned int begin_channel;
unsigned int n_channels;
unsigned int max_channel;
{
   static unsigned top_limit, bottom_limit, old_limit = FALSE;

   /* Draw the big rectangle */
   rectangle ( BIG_RECT_I1, BIG_RECT_J1, BIG_RECT_I2, BIG_RECT_J2, BIG_RECT_COLOR);

   /* Clear the small rectangle if it exists */
   if (old_limit)
      rectangle ( top_limit, SMALL_RECT_J1, bottom_limit, SMALL_RECT_J2, BCKGND_COLOR);

   /* Compute new small rectangle */
   top_limit= begin_channel * BIG_RECT_WDTH / max_channel + (BIG_RECT_I1 + 1);
   bottom_limit = (begin_channel+n_channels) * BIG_RECT_WDTH / max_channel
                  + (BIG_RECT_I1 + 1);

   /* Draw the little rectangle */
   rectangle (top_limit, SMALL_RECT_J1, bottom_limit, SMALL_RECT_J2, SMALL_RECT_COLOR);
   old_limit = TRUE;
}

/*-------------------------------------------------------------------------*
 *                            update_header                                *
 *-------------------------------------------------------------------------*/
/* Draw the screen header.                                                 */

PRIVATE
void update_header ()
{
   static char hdr_string[90], decimation_str[20], shift_str[5];
   char * left_tline, * right_tline;
   int i, magnification;

   left_tline = u_build_timeline (h->left_time, FALSE);
   right_tline = u_build_timeline (h->right_time, FALSE);

   /* Prepare header string */
   if (h->decimation <= 1)
      sprintf (decimation_str, "Undecimated");
   else
      sprintf (decimation_str, "Decimated %2d", h->decimation);

   /* Prepare shift string */
   for (magnification = 1, i = 0; i < h->magnification;
        magnification <<= 1, i++);
   sprintf (shift_str, "%3dX", magnification);

   sprintf (hdr_string, "%.11s%.4s %-14.12s  %7.2f Sps %-4s  %-14s %14.12s",
                    left_tline, &left_tline[24], &left_tline[11],
                    h->srate, shift_str, decimation_str, &right_tline[11]);

   /* Write header strings to screen */
   gputs ( HEADER_I, 4, hdr_string, 90, HEADER_COLORS);

   free (left_tline);
   free (right_tline);
}

/*-------------------------------------------------------------------------*
 *                            display_traces                               *
 *-------------------------------------------------------------------------*/
/* Display normalized traces on the screen.                                */

PRIVATE
void display_traces (plot_size)
unsigned int plot_size;
{
   unsigned int i, j, old_j, k, i_increment;
   unsigned int channel_offset, old_offset;
   int far * temp;

   channel_offset = HORIZONTAL_PIXS - plot_size;
   old_offset = HORIZONTAL_PIXS - old_size;

   i_increment = (int) ((VERTICAL_PIXS) / current_display->orientation);
   i = (TOP_MARGIN) + (i_increment / 2);
   j = LEFT_MARGIN + channel_offset;
   old_j = LEFT_MARGIN + old_offset;

   update_header ();

   for (k = 0; k < current_display->n_channels; k++, i += i_increment,
                                       channel_offset += HORIZONTAL_PIXS,
                                       old_offset += HORIZONTAL_PIXS) {
      if (old_plot_data != NULL) {
         if (old_size == plot_size)
            plotmod( i, j, &old_plot_data[channel_offset], &plot_data[channel_offset
                     plot_size, PLOTMOD_COLORS);
         else {
            plot( i, old_j, &old_plot_data[old_offset], old_size, PLOT_OFF_COLOR);
            plot( i, j, &plot_data[channel_offset], plot_size, PLOT_ON_COLOR);
         }
      } else
         plot( i, j, &plot_data[channel_offset], plot_size, PLOT_ON_COLOR);
   }

   temp = plot_data;
   plot_data = old_plot_data;
   old_plot_data = temp;
   old_size = plot_size;
}

/*-------------------------------------------------------------------------*
 *                            s_add_channel                                *
 *-------------------------------------------------------------------------*/
/* Add a channel to the home display.                                      */

PUBLIC
FLAG s_add_channel (channel)
unsigned int channel;
{
   if (display[HOME].n_channels < MAX_DISPLAY) {
      display[HOME].channel_list[display[HOME].n_channels] = channel;
      display[HOME].n_channels++;
      display[HOME].orientation++;
      if (screen_id) clear_display();
      return (TRUE);
   } else return (FALSE);
}

/*-------------------------------------------------------------------------*
 *                            s_clear                                      *
 *-------------------------------------------------------------------------*/
/* Clear the screen.                                                       */

PUBLIC
void s_clear ()
```

```
{
   turnon( BCKGND_COLOR);
}

/*-------------------------------------------------------------------------*
 *                              s_display_bar                              *
 *-------------------------------------------------------------------------*/
/* Display the bar.                                                        */

PUBLIC
void s_display_bar ()
{
   display_bar (display[BLOCK].begin_channel, display[BLOCK].n_channels,
                dt_get_scan_count());
}

/*-------------------------------------------------------------------------*
 *                              s_display_ids                              *
 *-------------------------------------------------------------------------*/
/* Display station id, channel, and trigger status.                        */

PUBLIC
void s_display_ids ()
{
   unsigned int i, k, l, i_increment;
   unsigned int channel;

   if (screen_id != SCREEN_1) return;

   i_increment = (int) ((VERTICAL_PIXS) / current_display->orientation);
   i = (TOP_MARGIN) + (i_increment / 2);

   for (k = 0; k < current_display->n_channels; k++) {
      channel = (current_display->type == LIST) ?
                current_display->channel_list[k] :
                current_display->begin_channel + k;
      l = build_textline (channel, textline);
      gputs ( i-4, 15, textline, 1+1, TEXT_COLORS);
      i += i_increment;
   }
}

/*-------------------------------------------------------------------------*
 *                            s_display_traces                             *
 *-------------------------------------------------------------------------*/
/* Display normalized traces on the screen.                                */

PUBLIC
void s_display_traces (rel_decimation, rel_bitshift)
int rel_decimation;
int rel_bitshift;
{
   Q_BUFFER far * b_head;
   double last_time;
   unsigned long channel_blocksize;
   unsigned int i, k, plot_size, channel_offset, index;
   unsigned int initial_offset, copy_size, copy_from, copy_to;
   int far * from_ptr, far * to_ptr, new_bitshift, lsb_mask, adjust;
   FLAG same_buffer;
   unsigned int bit_shift;

   /* Make sure that SCREEN_1 is on */
   if (screen_id != SCREEN_1) return;

   /* Check the demux and display queues */
   if ((b_head = dm_get_first_buffer()) == NULL) return;

   /* Compute decimation and plot_size */
   channel_blocksize = dt_get_channel_size ();
   h->decimation = rel_decimation + ( channel_blocksize <= HORIZONTAL_PIXS) ? 1
                 : channel_blocksize / HORIZONTAL_PIXS;
   if (!h->decimation) h->decimation = 1;
   plot_size = channel_blocksize / h->decimation;

   /* Compute new bit_shift */
   new_bitshift = h->magnificstion + rel_bitshift;
   if (new_bitshift >= 0 && new_bitshift <= MAX_MAGNIFICATION)
      h->magnification = new_bitshift;

   /* Compute time */
   h->srate = b_head->info.dig_rate;
   last_time = h->right_time;
   h->right_time = b_head->info.begintime + (((double) channel_blocksize) / h->srate
   h->left_time = h->right_time - (((double) HORIZONTAL_PIXS) * h->decimation / h->s
   same_buffer = (h->right_time - last_time) ? FALSE : TRUE;

   if (raw_data == NULL ) {
/*
      raw_data = (int far *) _fmalloc (HORIZONTAL_PIXS * current_display->n_channels
*/
      raw_data = (int far *) _fmalloc (HORIZONTAL_PIXS * MAX_DISPLAY * sizeof(int));
      if (raw_data == NULL) er_abort (S_NO_STORAGE);
   }

   if (reset_display || ! same_buffer) {
      /* Adjust old rsw data */
      initial_offset = HORIZONTAL_PIXS - plot_size;
      if (current_size > 0 && plot_size < HORIZONTAL_PIXS) {
         copy_size = (initial_offset > current_size) ? current_size : initial_offset
         copy_from = HORIZONTAL_PIXS - copy_size;
         copy_to = HORIZONTAL_PIXS - channel_blocksize - copy_size;
         from_ptr = &(raw_data[copy_from]);
         to_ptr = &(rsw_data[copy_to]);
         copy_size += HORIZONTAL_PIXS * (current_display->n_channels - 1);
         movedata (FP_SEG(from_ptr), FP_OFF(from_ptr), FP_SEG(to_ptr),
                   FP_OFF(to_ptr), copy_size * sizeof(int));
         current_size += plot_size;
         if (current_size > HORIZONTAL_PIXS) current_size = HORIZONTAL_PIXS;
      } else current_size = plot_size;

      /* Get new raw data */
      index = initial_offset;
      for (i = 0; i < current_display->n_channels; i++) {
         channel_offset = b_head->info.blocksize *
                          ((current_display->type == LIST) ?
                           current_display->channel_list[i] :
                           current_display->begin_channel + i);
         for (; b_head != NULL; b_head = dm_get_next_buffer()) {
            for (k = channel_offset; k < (b_head->info.blocksize + channel_offset);
                 k += h->decimation, index++)
               raw_data[index] = b_head->data[k] - b_head->info.dc_offset;
         }
         index += initial_offset;
         b_head = dm_get_first_buffer();
      }
      reset_display = FALSE;
   }

   if (plot_data == NULL ) {
/*
      plot_data = (int far *) _fmalloc (HORIZONTAL_PIXS *
                                        current_display->n_channels * sizeof(int));
*/
      plot_data = (int far *) _fmalloc (HORIZONTAL_PIXS *
                                        MAX_DISPLAY * sizeof(int));
      if (plot_data == NULL) er_abort (S_NO_STORAGE);
   }

   /* Compute the scale factor (bit_shift).  Assume that the dynamic range
      remains constant. */
   for (bit_shift = 0; (CHANNEL_HEIGHT >> bit_shift); bit_shift++);
   bit_shift = b_head->dynamic_range - bit_shift;
   bit_shift -= h->magnification;

   /* Generate plot data */
   for (k = 0; k < HORIZONTAL_PIXS * current_display->n_channels; k++) {
```

```
      adjust = 0; lsb_mask = 0;
      if (raw_data[k] < 0) {
         adjust = 1;
         lsb_mask = (raw_data[k]+1) & 0x0001;
      }
      plot_data[k] = ((raw_data[k]+lsb_mask) >> bit_shift) + adjust;
   }

   display_traces (current_size);
}

/*-------------------------------------------------------------------------*
 *                          s_display_triggers                             *
 *-------------------------------------------------------------------------*/
/* Display triggers on the screen.                                         */

PUBLIC
void s_display_triggers ()
{
   unsigned int i, j, k, l, i_increment, channel;
   FLAG clear_flag;

   if (screen_id != SCREEN_1) return;

   i_increment = (int) ((VERTICAL_PIXS) / current_display->orientation);
   i = (TOP_MARGIN) + (i_increment / 2);
   j = LEFT_MARGIN;

   for (k = 0; k < current_display->n_channels; k++) {
      channel = (current_display->type == LIST) ?
                current_display->channel_list[k] :
                (current_display->begin_channel + k);
      clear_flag = FALSE;
      switch (t_get_channel_trigger_state(channel)) {
         case CHANNEL_TRIGGERED_EVENT:
            gputs ( i-4, j-8, "E", 1, TRIGGER_ON_COLORS);
            break;
         case CHANNEL_TRIGGERED:
            gputs ( i-4, j-8, "T", 1, TRIGGER_ON_COLORS);
            break;
         case CHANNEL_TRIGGER_ENABLED:
         case CHANNEL_TRIGGER_DISABLED:
         default:
            clear_flag = TRUE;
      }
      switch (dsp_get_status()) {
         case DSP_CALIBRATION:
            if (dsp_get_spectral_channel() ==  channel) {
               gputs ( i-4, j-8, "C", 1, TRIGGER_ON_COLORS);
               clear_flag = FALSE;
            }
            break;
         case OFF:
         default: ;
      }
      if (clear_flag) gputs ( i-4, j-8, " ", 1, TRIGGER_OFF_COLORS);
      i += i_increment;
   }
}

/*-------------------------------------------------------------------------*
 *                            s_initialize                                 *
 *-------------------------------------------------------------------------*/
/* Initialize the screen module.                                           */

PUBLIC
void s_initialize ()
{
   current_display = NULL;
   display_id = -1;
   display[HOME].type = LIST;
   display[HOME].n_channels = 0;
   display[HOME].orientation = 0;
   display[BLOCK].type = RANGE;
   display[BLOCK].n_channels = 0;
   display[BLOCK].orientation = 0;
   display[BLOCK].begin_channel = 0;

   old_size = 0;
   raw_data = plot_data = old_plot_data = NULL;

   /* Header */
   h->magnification = 0;
   h->decimation = 1;
   h->right_time = 0;
}

/*-------------------------------------------------------------------------*
 *                          s_initialize_screen                            *
 *-------------------------------------------------------------------------*/
/* Initialize the graphics screen.                                         */

PUBLIC
void s_initialize_screen ()
{
   set_graphics();
   turnon( 0);
   screen_id = SCREEN_1;
}

/*-------------------------------------------------------------------------*
 *                          s_remove_channel                               *
 *-------------------------------------------------------------------------*/
/* Remove a channel from the home display.                                 */

PUBLIC
FLAG s_remove_channel (channel)
unsigned int channel;
{
   /* If channel is in home display's channel list,
         remove it and decrement n_channels and orientation
         if screen is on, clear display;
         return (TRUE);
      else return (FALSE);
   */
}

/*-------------------------------------------------------------------------*
 *                            s_reset_screen                               *
 *-------------------------------------------------------------------------*/
/* Reset the monochrome screen to text.                                    */

PUBLIC
void s_reset_screen ()
{
   screen_id = SCREEN_0;
   set_video (0x3);
#if defined (EGA) || defined (VGA)
#elif defined (MONO)
   set_text ();
#endif
}

/*-------------------------------------------------------------------------*
 *                            s_screen_on                                  *
 *-------------------------------------------------------------------------*/
/* Return screen flag.                                                     */

PUBLIC
FLAG s_screen_on ()
{
   return ((screen_id == SCREEN_0) ? FALSE : TRUE);
}

/*-------------------------------------------------------------------------*
 *                          s_select_display                               *
```

```
 *-------------------------------------------------------------------------*/
/* Select the display mode.                                                */

PUBLIC
void s_select_display (display_mode)
unsigned int display_mode;
{
   screen_id = SCREEN_1;
   clear_display();
   clear_bar();
   display_id = display_mode;
   current_display = &display[display_mode];
}

/*-------------------------------------------------------------------------*
 *                            s_select_screen                              *
 *-------------------------------------------------------------------------*/
/* Select a screen.                                                        */

PUBLIC
void s_select_screen (screen)
unsigned int screen;
{
   screen_id = screen;
}

/*-------------------------------------------------------------------------*
 *                          s_set_channelrange                             *
 *-------------------------------------------------------------------------*/
/* Set the channel range in the block display.                             */

PUBLIC
FLAG s_set_channelrange (rel_begin, n_channels)
int rel_begin;
unsigned int n_channels;
{
   unsigned int scan_count;
   unsigned int old_count;

   old_count = display[BLOCK].n_channels;

   scan_count = dt_get_scan_count();
   if (display[BLOCK].begin_channel+rel_begin >= scan_count) return (FALSE);
   if (display[BLOCK].begin_channel+rel_begin < 0) return (FALSE);

   display[BLOCK].begin_channel += rel_begin;
   if ((display[BLOCK].begin_channel+n_channels) >= scan_count)
      display[BLOCK].n_channels = scan_count - display[BLOCK].begin_channel;
   else display[BLOCK].n_channels = n_channels;

   if (display[BLOCK].n_channels > MAX_DISPLAY)
      display[BLOCK].n_channels = MAX_DISPLAY;
   display[BLOCK].orientation = display[BLOCK].n_channels;


   if (screen_id == SCREEN_1) {
      if (old_count != display[BLOCK].n_channels)
         clear_display();
      else {
         current_size = 0;
         reset_display = TRUE;
      }
      display_bar (display[BLOCK].begin_channel, display[BLOCK].n_channels, scan_count);
   }
   return (TRUE);
}

/*-------------------------------------------------------------------------*
 *                             s_view_help                                 *
 *-------------------------------------------------------------------------*/
/* Display the help screen.                                                */

PUBLIC
void s_view_help ()
{
   static int start_i = 50, j = 14;
   int line, i;

   /* Clear the screen */
   turnon( BCKGND_COLOR);

   screen_id = SCREEN_2;

   /* Display lines until NULL is reached */
   for (i = start_i, line = 0; help_info[line]; line++, i += 10)
      gputs (i, j, help_info[line], strlen(help_info[line]), HELP_COLORS);
}
```

```
/* FILE: mscreen.h                                (D. Tottingham  04/26/92)

This is an include file of defines, data structure definitions and
external declarations that are common in the mscreen module.

*/

#ifndef _MSCREEN_
#define _MSCREEN_

/*************************************************************************
                                INCLUDES
*************************************************************************/
#include "mconst.h"
#include "mvideo.h"


/*************************************************************************
                                DEFINES
*************************************************************************/
#define MAX_MAGNIFICATION       3
#define MAX_BLOCK               16
#define MAX_DISPLAY             32
#define MAX_TEXTLINE            40

#define N_DISPLAYS              2

/* Display types */
#define LIST                    0
#define RANGE                   1

/* Display ids */
#define HOME                    0
#define BLOCK                   1

/* Screen ids */
#define SCREEN_0                0               /* text */
#define SCREEN_1                1               /* waveforms */
#define SCREEN_2                2               /* help */

/* Screen dimensions */
#define TOP_MARGIN              50
#define BOTTOM_MARGIN           9
#define LEFT_MARGIN             126
#define RIGHT_MARGIN            2
#define CHANNEL_HEIGHT          16
#define HORIZONTAL_PIXS         512
#define VERTICAL_PIXS           ((MAXI) - (TOP_MARGIN) - (BOTTOM_MARGIN))

/* Bar Constants */
#define BIG_RECT_I1             TOP_MARGIN
#define BIG_RECT_J1             3
#define BIG_RECT_I2             ((MAXI) - (BOTTOM_MARGIN) - 1)
#define BIG_RECT_J2             13
#define BIG_RECT_WDTH           ((BIG_RECT_I2) - (BIG_RECT_I1) - 2)
#define BIG_RECT_COLOR          YELLOW

#define SMALL_RECT_J1           ((BIG_RECT_J1) + 1)
#define SMALL_RECT_J2           ((BIG_RECT_J2) - 1)
#define SMALL_RECT_COLOR        WHITE

/* Colors */
#define BCKGND_COLOR            BLACK
#define HEADER_I                32
#define HEADER_COLORS           YELLOW, BCKGND_COLOR, BCKGND_COLOR
#define HELP_COLORS             YELLOW, BCKGND_COLOR, BCKGND_COLOR
#define PLOT_ON_COLOR           WHITE
#define PLOT_OFF_COLOR          BCKGND_COLOR
#define PLOTMOD_COLORS          WHITE, BCKGND_COLOR
#define TEXT_COLORS             YELLOW, BCKGND_COLOR, BCKGND_COLOR
#define TEXT_OFF_COLORS         BCKGND_COLOR, BCKGND_COLOR, BCKGND_COLOR
#define TRIGGER_ON_COLORS       LIGHT_RED, BCKGND_COLOR, BCKGND_COLOR
#define TRIGGER_OFF_COLORS      BCKGND_COLOR, BCKGND_COLOR, BCKGND_COLOR


/*************************************************************************
                        STRUCTURE DEFINITIONS

*************************************************************************/
typedef struct {
        unsigned int type;
        unsigned int n_channels;
        unsigned int orientation;
        int begin_channel;
        unsigned int channel_list[MAX_DISPLAY];
} S_DISPLAY;

typedef struct {
        float       srate;
        unsigned int decimation;
        int         magnification;
        double      left_time;
        double      right_time;
} S_HEADER;


/*************************************************************************
                        EXTERNAL DECLARATIONS

These functions can be called from all modules that include this file.

*************************************************************************/
PUBLIC FLAG s_add_channel (unsigned int);
PUBLIC void s_clear ();
PUBLIC void s_display_bar ();
PUBLIC void s_display_ids ();
PUBLIC void s_display_traces (int, int);
PUBLIC void s_display_triggers ();
PUBLIC void s_initialize ();
PUBLIC void s_initialize_screen ();
PUBLIC FLAG s_remove_channel (unsigned int);
PUBLIC void s_reset_screen ();
PUBLIC FLAG s_screen_on ();
PUBLIC void s_select_display (unsigned int);
PUBLIC void s_select_screen (unsigned int);
PUBLIC FLAG s_set_channelrange (int, unsigned int);
PUBLIC void s_view_help ();

#endif
```

```
/* FILE: mscrnhdr.c                                    (D. Tottingham  04/26/92)

This is a collection of C helper functions that manage the status header
information that is kept on the top of the monochrome display for
xdetect.  All functions have been written and compiled medium model.  The
following functions are included:

h_increment_bankswitch_ctr ()   increment bank switch counter
h_increment_event_ctr ()        increment event counter
h_initialize_params ()          initialize a screen header
h_initialize_path ()            initialize path_name
h_set_event_ctr ()              set event counter
h_set_trigger_status ()         set trigger status
h_set_uptime ()                 update uptime.
h_toggle_status ()              toggle a status bit
h_update ()                     update the header

EXTERNAL FUNCTIONS CALLED:

dm_get_uptime ()                get session up time.
u_get_diskspace ()              get the percentage of free disk space
u_gmtime ()                     convert time value into structure.

HISTORY:
   none


/*****************************************************************************
                              INCLUDE FILES
*****************************************************************************/
#include <stdio.h>
#include <string.h>

#include "mconst.h"
#include "mdemux.h"
#include "mscrnhdr.h"
#include "mutils.h"
#include "mvideo.h"
#include "xdetect.h"


/*****************************************************************************
                              GLOBAL DATA
*****************************************************************************/
PRIVATE H_STATUS header, *h = &header;
PRIVATE char path_name[MAX_FILENAME_LENGTH];


/*-------------------------------------------------------------------------*
 *                          build_storage_stat                             *
 *-------------------------------------------------------------------------*/
/* Build a storage status string.  Percentage of free space is converted
   converted into an ascii string.                                         */

PRIVATE
void build_storage_stat (storage_status)
char * storage_status;
{
   static char * device_id[] = {"A:","B:","C:","D:","E:","F:","G:",NULL};
   long percent_avail;
   int drive;

   percent_avail = u_get_diskspace (path_name, &drive);

   sprintf (storage_status, "  %2s%3ld%%", device_id[drive], percent_avail);
}

/*-------------------------------------------------------------------------*
 *                      h_increment_bankswitch_ctr                         *
 *-------------------------------------------------------------------------*/
/* Increment bank switch counter.                                          */

PUBLIC
void h_increment_bankswitch_ctr ()
{
   h->bankswitch_ctr++;
   h_update();
}

/*-------------------------------------------------------------------------*
 *                        h_increment_event_ctr                            *
 *-------------------------------------------------------------------------*/
/* Increment event counter.                                                */

PUBLIC
void h_increment_event_ctr ()
{
   h->event_ctr++;
   h_update();
}

/*-------------------------------------------------------------------------*
 *                          h_initialize_params                            *
 *-------------------------------------------------------------------------*/
/* Initialize a screen header.                                             */

PUBLIC
void h_initialize_params ()
{
   h->milestone = REVISION_NAME;
   h->event_status = H_ASLEEP;
   h->event_ctr = 0;
   h->uptime.days = 0;
   h->uptime.hours = 0;
   h->uptime.minutes = 0;
   h->uptime.seconds = 0;
   h->uptime.millisec = 0;
}

/*-------------------------------------------------------------------------*
 *                          h_initialize_path                              *
 *-------------------------------------------------------------------------*/
/* Initialize path_name.                                                   */

PUBLIC
void h_initialize_path (pathname)
char pathname[];
{
   strcpy (path_name, pathname);
}

/*-------------------------------------------------------------------------*
 *                            h_set_event_ctr                              *
 *-------------------------------------------------------------------------*/
/* Set event counter.                                                      */

PUBLIC
void h_set_event_ctr (event_ctr)
int event_ctr;
{
   h->event_ctr = event_ctr;
   h_update();
}

/*-------------------------------------------------------------------------*
 *                          h_set_trigger_status                           *
 *-------------------------------------------------------------------------*/
/* Set trigger status.                                                     */

PUBLIC
void h_set_trigger_status (trigger_status)
int trigger_status;
```

```
{
   if (trigger_status)
      h->event_status |= H_AUTOTRIGGER_ENABLED;
   else
      h->event_status &= ~H_AUTOTRIGGER_ENABLED;
}

/*-------------------------------------------------------------------------*
 *                            h_toggle_status                              *
 *-------------------------------------------------------------------------*/
/* Toggle a status bit.                                                    */

PUBLIC
void h_toggle_status (status_bit)
int status_bit;
{
   if (h->event_status & status_bit)
      /* Bit is on so turn it off */
      h->event_status &= ~status_bit;
   else
      h->event_status |= status_bit;
   h_update();
}

/*-------------------------------------------------------------------------*
 *                              h_update                                   *
 *-------------------------------------------------------------------------*/
/* Updates the status header information on monochrome display.            */

PUBLIC
void h_update ()
{
   static char * event_status[] = {             /* status conversion array */
                  "Recording Event",                        /* 0     */
                  "Recording Calibration",                  /* 1     */
                  "Free Run Enabled",                       /* 2     */
                  "Autotriggering",                         /* 3     */
                  "Asleep"                                  /* 4     */
                  };

   static char top_hdr[90], right_hdr[70];
   unsigned short bit_ptr;
   int i, j;

   /* Get the current storage status */
   build_storage_stat (h->storage_status);

   /* Prepare right header string */
   for (bit_ptr = 1, i = 0; !(h->event_status & bit_ptr); bit_ptr <<= 1, i++);

   j = sprintf (right_hdr, "%-7s free ", h->storage_status);
   if (h->event_ctr == 0)
      j += sprintf (right_hdr+j, " No events");
   else if (h->event_ctr == 1)
      j += sprintf (right_hdr+j, "  1 event ");
   else j += sprintf (right_hdr+j, "%3d events", h->event_ctr);

   if (h->bankswitch_ctr == 0)
      j += sprintf (right_hdr+j, "        ");
   else
      j += sprintf (right_hdr+j, "/ %3d bs", h->bankswitch_ctr);

   j += sprintf (right_hdr+j, "%23s", event_status[i]);

   /* Prepare top header string */
   sprintf (top_hdr, "%-10.10s F1=help %60s", h->milestone, right_hdr);

   /* Write header strings on screen */
   gputs ( HDR1_I1, 4, top_hdr, 90, HEADER_COLORS);

   /* Prepare uptime string */
   sprintf (top_hdr, "Up %3d days, %02d:%02d:%02d", h->uptime.days,
            h->uptime.hours, h->uptime.minutes, h->uptime.seconds);

   /* Write uptime on next line */
   gputs ( HDR1_I2, 4, top_hdr, 90, HEADER_COLORS);
}

/*-------------------------------------------------------------------------*
 *                             h_set_uptime                                *
 *-------------------------------------------------------------------------*/

PUBLIC
void h_set_uptime ()
{
   u_gmtime (dm_get_uptime(), &(h->uptime));
   h_update();
}
```

```
/* FILE: mscrnhdr.h                                (D. Tottingham  04/26/92)

This is an include file of the defines, data structure definitions and
external data declarations for the mscrnhdr module.

*/

#ifndef _MSCRNHDR_
#define _MSCRNHDR_

/*************************************************************************
                                INCLUDES
*************************************************************************/
#include "mconst.h"
#include "mutils.h"
#include "mvideo.h"


/*************************************************************************
                                DEFINES
*************************************************************************/
#define MAX_STORAGE_STAT_LENGTH 10

/* Event Status Bits */
#define H_ASLEEP                0x0010
#define H_AUTOTRIGGER_ENABLED   0x0008
#define H_FREE_RUN              0x0004
#define H_RECORD_CALIBRATION    0x0002
#define H_RECORD_EVENT          0x0001

/* Header Constants */
#define HDR1_I1                 5
#define HDR1_I2                 18
#define HDR2_I1                 ((MAXI) - 8)
#define BCKGND_COLOR            BLACK
#define HEADER_COLORS           YELLOW, BCKGND_COLOR, BCKGND_COLOR


/*************************************************************************
                        STRUCTURE DEFINITIONS
*************************************************************************/
typedef unsigned short          BIT_MAP;

typedef struct {
        char            *milestone;
        BIT_MAP         event_status;
        char            storage_status[MAX_STORAGE_STAT_LENGTH];
        int             event_ctr;
        unsigned int    bankswitch_ctr;
        U_ELAPSED_TIME uptime;
} H_STATUS;


/*************************************************************************
                        EXTERNAL DECLARATIONS

These functions can be called from all modules that include this file.

*************************************************************************/
PUBLIC void h_increment_bankswitch_ctr ();
PUBLIC void h_increment_event_ctr ();
PUBLIC void h_initialize_params ();
PUBLIC void h_initialize_path (char *);
PUBLIC void h_set_event_ctr (int);
PUBLIC void h_set_trigger_status (int);
PUBLIC void h_set_uptime ();
PUBLIC void h_toggle_status (int);
PUBLIC void h_update ();

#endif
```

```
/* FILE: mstation.c                                (D. Tottingham  04/26/92)

This is a collection of C functions that manage the station information for
for xdetect.  All functions have been written and compiled medium model.
The following functions are included:

st_add_station ()               add station to station queue.
st_build_textline ()            build a textline for the mscreen module
st_get_authority ()             get organization processing the data.
st_get_head_station ()          get station at head of station queue
st_get_next_station ()          get next station from station queue
st_get_netwname ()              get network name
st_get_station ()               get station from station queue
st_initialize_params ()         initializa key parameters
st_initlalize_stations ()       initialize station list
st_set_authority ()             set authority code
st_set_gain ()                  set gain for particular channel
st_set_netwname ()              set network name

EXTERNAL FUNCTIONS CALLED:

er_abort ()                     display an error message then quit
dt_get_channel_gain ()          get channel gain
dt_get_clip_value ()            get clip value
dt_get_data_type ()             get data type
dt_get_data_units ()            get data units
dt_get_dc_offset ()             get dc offset
dt_get_input_range ()           get input range
ll_fold ()                      convert lat, lon to geocentric lat, lon
ll_to_xy ()                     convert lat, lon to x, y
q_enqueue ()                    enqueue a data link on a data queue
q_initialize ()                 initialize a data quaue
suds_initialize ()              initialize a suds structura
suds_initialize_tag ()          initializa a suds structtag
t_set_netwname ()               set network name
u_strncpy ()                    copies far string2 to far string1
u_strnlen ()                    get length of far string
u_timestamp ()                  get timestamp

HISTORY:
   none

*/


/*************************************************************************
                            INCLUDE FILES
*************************************************************************/
#include <io.h>
#include <malloc.h>
#include <math.h>
#include <stdio.h>
#include <string.h>

#include "mconst.h"
#include "mdosfunc.h"
#include "mdt28xx.h"
#include "merror.h"
#include "mlatlon.h"
#include "mqueue.h"
#include "mstation.h"
#include "msudsini.h"
#include "mtrigger.h"
#include "mutils.h"


/*************************************************************************
                            GLOBALS
*************************************************************************/
PRIVATE Q_QUEUE station_queue;
PRIVATE Q_LINK * head_ptr;
PRIVATE LL_LATLON mean_latlon;
PRIVATE double mean_lat, mean_lon;
PRIVATE char netwname[4];
PRIVATE unsignad int n_stations, station_fd, station_size;
PRIVATE int authority;


/*-----------------------------------------------------------------------*
 *                              b_search                                 *
 *-----------------------------------------------------------------------*/
/* Look in sorted station file for a specific atation using a hash value
   based binary search.                                                  */

PRIVATE
int b_search (fd, size, search_key, stationcomp_ptr)
unsigned int fd;
unsigned int size;
unsigned long aearch_key;
SUDS_STATIONCOMP far * stationcomp_ptr;
{
   static SUDS_STRUCTTAG far structtag;
   unsigned int head, tail, current;
   unsigned long current_key;

   head = 0, tail = size - 1;
   while (1) {
      current = (head + tail) / 2;

      /* Get the current station component from station file */
      seek_file (fd, (long) (current * (sizeof(SUDS_STRUCTTAG) +
                             sizeof(SUDS_STATIONCOMP))), SEEK_SET);
      read_file (fd, sizeof(SUDS_STRUCTTAG), ((int far *)&structtag));
      if (structtag.id_struct != STATIONCOMP) er_abort (ST_INVALID_FILE);
      read_file (fd, sizeof(SUDS_STATIONCOMP), ((int far *)stationcomp_ptr));

      /* Do the aearch */
      current_key = *((unsigned long far *) &(stationcomp_ptr->channel));
/* printf ("search_key, current_key = %lu, %lu\n", search_key, current_key); */
      if (search_key == current_key)
         return (current);
      else if (head == tail)
         return (-1);
      else if (search_key < current_key)
         tail = (tail == current) ? head : current;
      else {
         head = (head == current) ? tail : current;
/* printf ("head, current = %u, %u\n", head, current);  */
      }
   }
}

/*-----------------------------------------------------------------------*
 *                            gat_hashvalue                              *
 *-----------------------------------------------------------------------*/
/* Generate an aggregate hash value for given network, station name, and
   component.                                                            */

PRIVATE
unsigned long get_haahvalue (network, at_name, component)
char * network;
char * st_name;
char component;
{
   static unsigned int shifter1[] = {25, 16, 23, 18};
   atatic unsigned int shifter2[] = {12, 0, 5, 7, 2};
   static unsigned int shifter3 = 10;
   unsigned long hash;
   unsigned int i;

   hash = 0;
   for (i = 0; network[i] && i < 4; i++)
      hash += ((unsigned long) ((network[i] - 48) & 0x7f)) << shifter1[i];
```

```
   for (i = 0; st_name[i] && i < 5; i++)
      hash += ((unsigned long) ((st_name[i] - 48) & 0x7f)) << shifter2[i];

   hash += ((unsigned long) component) << shifter3;
   return (hash);
}

/*-------------------------------------------------------------------*
 *                          in_masterlist                            *
 *-------------------------------------------------------------------*/
/* Determines if station component is defined in STATION.###.  If it is,
   the station information is copied directly into the suds structure.   */

PRIVATE
FLAG in_masterlist (network, st_name, component, ptr)
char network[];
char st_name[];
char component;
SUDS_STATIONCOMP far * ptr;
{
   static SUDS_STRUCTTAG far structtag;
   unsigned int size;
   unsigned long hash_value;
   int current_ptr, i;
   FLAG done, first_time;

   hash_value = get_hashvalue (network, st_name, component);
   if ((current_ptr = b_search (station_fd, station_size, hash_value, ptr)) >= 0) {
/* printf ("Found hash value for ");
for (i = 0; i < 10; i++)
   printf ("%c", ((char far *)ptr->sc_name.network)[i]);
printf ("\n");  */
      /* Is this the right station ? */
      if (!u_strncmp (ptr->sc_name.network, network, 4) &&
          !u_strncmp (ptr->sc_name.st_name, st_name, 5) &&
          ptr->sc_name.component == component) {
/* printf ("Found %s in station file\n", st_name);  */
         return (TRUE);
      }

      /* Wrong station. Check for collisions.
         Stations with unique hash values have their data_type = 0.
         All stations with the same hash value receive incremental data_type
         values (i.e. 0, 1, 2 ...) as they are added to the station file.
         To search a group of stations with the same hash value for a
         specific station, we simply scan between stations with data_type
         values = 0.                                                    */

      if (ptr->data_type) {
         current_ptr -= ptr->data_type;
/* printf ("Adjusted current_ptr by %d\n", ptr->data_type);  */
      } else current_ptr++;

      for (first_time = TRUE, done = FALSE; ! done;) {
         seek_file (station_fd, (long) (current_ptr * (sizeof(SUDS_STRUCTTAG) +
                                    sizeof(SUDS_STATIONCOMP))), SEEK_SET);
         read_file (station_fd, sizeof(SUDS_STRUCTTAG), ((int far *)&structtag));
         if (structtag.id_struct != STATIONCOMP) er_abort (ST_INVALID_FILE);
         read_file (station_fd, sizeof(SUDS_STATIONCOMP), ((int far *)ptr));
         if (ptr->data_type || first_time) {
            if (!u_strncmp (ptr->sc_name.network, network, 4) &&
                !u_strncmp (ptr->sc_name.st_name, st_name, 5) &&
                ptr->sc_name.component == component) {
/* printf ("Found %s in station file\n", st_name);  */
               return (TRUE);
            }
            first_time = FALSE;
         } else done = TRUE;
         current_ptr++;
      }
   }
/* printf ("Could not find %s in station file: hash value = %lu\n", st_name,
       hash_value);   */
   return (FALSE);
}

/*-------------------------------------------------------------------*
 *                          st_add_station                           *
 *-------------------------------------------------------------------*/
/* Add station to station queue.                                     */

PUBLIC
void st_add_station (st_name, component, channel)
char st_name[];
char component;
unsigned int channel;
{
   Q_STATION far * qst;
   Q_TYPE type;
   int atod_gain;

   qst = (Q_STATION far *) _fmalloc (sizeof(Q_STATION));
   if (qst == NULL) er_abort (ST_NO_STORAGE);
   suds_initialize_tag (STATIONCOMP, &qst->structtag);

   atod_gain = dt_get_channel_gain(channel);

   /* Get suds structure from master station list, if possible */
   if (! in_masterlist (netwname, st_name, component, &qst->info)) {
      suds_initialize (STATIONCOMP, &qst->info);
      u_strncpy (qst->info.sc_name.network, ((char far *)netwname), 4);
      u_strncpy (qst->info.sc_name.st_name, ((char far *)st_name), 5);
      qst->info.sc_name.component = component;
      qst->info.sensor_type = 't';
      qst->info.polarity    = 'n';
      qst->info.effective = u_timestamp();
      qst->info.max_gain = atod_gain;
      qst->in_masterlist = FALSE;
   } else {
      qst->info.max_gain *= atod_gain;
      qst->in_masterlist = TRUE;
      mean_lat += qst->info.st_lat;
      mean_lon += qst->info.st_long;
      n_stations++;
   }

   qst->info.channel = channel;
   qst->info.atod_gain = atod_gain;

   qst->info.data_type   = dt_get_data_type ();
   qst->info.data_units  = dt_get_data_units ();
   qst->info.clip_value = ((float)dt_get_clip_value ()) - dt_get_dc_offset ();
   qst->info.con_mvolts = (dt_get_input_range () / qst->info.max_gain *
                           1000.00) / (qst->info.clip_value * 2.00);
   type.station = qst;
   q_enqueue (&station_queue, type);
}


/*-------------------------------------------------------------------*
 *                         st_build_textline                         *
 *-------------------------------------------------------------------*/
/* Build a textline for the mscreen module.                          */

PUBLIC
int st_build_textline (channel, textline)
unsigned int channel;
char * textline;
{
   Q_LINK * head;
   char st_name[5];
   int j;

   head = station_queue.head;
   for (head = station_queue.head; head != NULL; head = head->next)
      if (head->type.station->info.channel == channel) {
```

```
         u_strncpy (((char far *)st_name), head->type.station->info.sc_name.st_name, 4);
         j = sprintf (textline, "%3d %-4.4s", channel, st_name);
         return (j);
      }
   return ( 0);
}

/*-------------------------------------------------------------------------*
 *                          st_get_authority                               *
 *-------------------------------------------------------------------------*/
/* Get organization processing the data.                                   */

PUBLIC
int st_get_authority ()
{
   return (authority);
}

/*-------------------------------------------------------------------------*
 *                        st_get_head_station                              *
 *-------------------------------------------------------------------------*/
/* Get station at head of station queue.                                   */

PUBLIC
Q_STATION far * st_get_head_station ()
{
   head_ptr = station_queue.head;
   if (head_ptr != NULL)
      return (head_ptr->type.station);
   else return (NULL);
}

/*-------------------------------------------------------------------------*
 *                        st_get_next_station                              *
 *-------------------------------------------------------------------------*/
/* Get next station from station queue.                                    */

PUBLIC
Q_STATION far * st_get_next_station ()
{
   head_ptr = head_ptr->next;
   if (head_ptr != NULL)
      return (head_ptr->type.station);
   else return (NULL);
}

/*-------------------------------------------------------------------------*
 *                          st_get_netwname                                *
 *-------------------------------------------------------------------------*/
/* Get network name.                                                       */

PUBLIC
char * st_get_netwname ()
{
   return (netwname);
}

/*-------------------------------------------------------------------------*
 *                          st_get_station                                 *
 *-------------------------------------------------------------------------*/
/* Get station from station queue.                                         */

PUBLIC
Q_STATION far * st_get_station (channel)
unsigned int channel;
{
   Q_LINK * head;

   head = station_queue.head;
   while (head != NULL) {
      if (head->type.station->info.channel == channel)
         return (head->type.station);
      head = head->next;
   }
   return ( NULL);
}

/*-------------------------------------------------------------------------*
 *                        st_initialize_params                             *
 *-------------------------------------------------------------------------*/
/* Initialize key parameters.                                              */

PUBLIC
void st_initialize_params ()
{
   authority = AUTHORITY;
   mean_lat = mean_lon = 0.0;
   n_stations = 0;
   strcpy (netwname, NETWORK_NAME);
   q_initialize (&station_queue);

   if (station_fd = open_file (STATION_FILENAME, READ))
      station_size = (unsigned int) (filelength (station_fd) /
                     (sizeof(SUDS_STRUCTTAG) + sizeof(SUDS_STATIONCOMP)));
   else er_abort (ST_INVALID_FILE);
}

/*-------------------------------------------------------------------------*
 *                        st_initialize_stations                           *
 *-------------------------------------------------------------------------*/
/* Initialize station list.                                                */

PUBLIC
void st_initialize_stations ()
{
   Q_LINK * link_ptr;
   LL_XY xy;

   if (n_stations == 0) return;
   mean_lat /= n_stations;
   mean_lon /= n_stations;

   mean_latlon = ll_fold (mean_lat, -mean_lon);
   for (link_ptr = station_queue.head; link_ptr != NULL; link_ptr = link_ptr->next)
      if (link_ptr->type.station->in_masterlist) {
         link_ptr->type.station->mean_latlon = mean_latlon;
         xy = ll_to_xy (link_ptr->type.station->info.st_lat,
                        -link_ptr->type.station->info.st_long,
                        mean_latlon);
         link_ptr->type.station->x = xy.x;
         link_ptr->type.station->y = xy.y;
         link_ptr->type.station->xsq = xy.x * xy.x;
         link_ptr->type.station->ysq = xy.y * xy.y;
      }
   }
   close_file (station_fd);
}

/*-------------------------------------------------------------------------*
 *                          st_set_authority                               *
 *-------------------------------------------------------------------------*/
/* Set authority code.                                                     */

PUBLIC
void st_set_authority (code)
int code;
{
   authority = code;
}

/*-------------------------------------------------------------------------*
 *                            st_set_gain                                  *
 *-------------------------------------------------------------------------*/
/* Set gain for particular channel.                                        */

PUBLIC
```

```
void st_set_gain (channel, gain)
unsigned int channel;
unsigned int gain;
{
   Q_LINK * head;

   head = station_queue.head;
   while (head != NULL) {
      if (head->type.station->info.channel == channel) {
         head->type.station->info.con_mvolts *= head->type.station->info.max_gain;
         head->type.station->info.max_gain /= head->type.station->info.atod_gain;
         head->type.station->info.max_gain *= gain;
         head->type.station->info.con_mvolts /= head->type.station->info.max_gain;
         head->type.station->info.atod_gain = gain;
         return;
      }
      head = head->next;
   }
}

/*-------------------------------------------------------------------------*
 *                          st_set_netwname                                *
 *-------------------------------------------------------------------------*/
/* Set network name.                                                       */

PUBLIC
void st_set_netwname (nwn)
char nwn[];
{
   strncpy (netwname, nwn, 4);
   t_set_netwname (nwn);
}
```

```
/* FILE: mstation.h                                (D. Tottingham  04/26/92)

This is an include file of defines, data structure definitions and
external declarations that are common in the mstation module.

*/

#ifndef _MSTATION_
#define _MSTATION_

/************************************************************************
                                INCLUDES
************************************************************************/
#include "mconst.h"
#include "mqueue.h"


/************************************************************************
                                DEFINES
************************************************************************/
#define NETWORK_NAME            "USGS"
#define STATION_FILENAME        "STATION.@@@"
#define AUTHORITY               101             /* CALNET */


/************************************************************************
                        EXTERNAL DECLARATIONS

These functions can be called from all modules that include this file.

************************************************************************/
PUBLIC void st_add_station (char *, char, unsigned int);
PUBLIC int st_build_textline (unsigned int, char *);
PUBLIC int st_get_authority ();
PUBLIC Q_STATION far * st_get_head_station ();
PUBLIC Q_STATION far * st_get_next_station ();
PUBLIC Q_STATION far * st_get_station (unsigned int);
PUBLIC char * st_get_netwname ();
PUBLIC void st_initialize_params ();
PUBLIC void st_initialize_stations ();
PUBLIC void st_set_authority (int);
PUBLIC void st_set_gain (unsigned int, unsigned int);
PUBLIC void st_set_netwname (char *);

#endif
```

```
/* FILE: msudsini.c                                (D. Tottingham  04/26/92)

This is a collection of routines that initialize suds atructures.
The program has been written and compiled modium model.  The following public
functions are defined in this file:

suds_initialize ()              initialize a suds structure
suds_initialize_tag ()          initialize a suds structtag

EXTERNAL FUNCTIONS CALLED:

u_strcpy ()                     copies far string2 to far string1

HISTORY:
   none



/*************************************************************************
                            INCLUDE FILES

*************************************************************************/
#include <stdlib.h>
#include <string.h>

#include "mconst.h"
#include "msudsini.h"
#include "mutils.h"


/*************************************************************************
                            GLOBALS

*************************************************************************/
PRIVATE struct {
   LG_INT  fstype;          /* Structure types or identifiers              */
   SH_INT  ftype;           /* Type of variable in field                   */
   LG_INT  flength;         /* Length of variable in the field             */
   LG_INT  offset;          /* Offset of variable pointer from beginning of
                               structure measured in bytes, where first byte
                               of structure=0.                             */
   STRING  *initval;        /* Value to initialize structure               */
   LG_INT  nextfstype;      /* If structure, this is fstype                */
} initializer[] = {
   NO_STRUCT,    STR,    1,      0,      NOSTRG, 0,
/*1*/
   ATODINFO,     SHT,    1,      0,      NODATA, 0,
   ATODINFO,     SHT,    1,      2,      NODATA, 0,
   ATODINFO,     SHT,    1,      4,      NODATA, 0,
   ATODINFO,     SHT,    1,      6,      NODATA, 0,
   ATODINFO,     SHT,    1,      8,      NODATA, 0,
   ATODINFO,     CHR,    1,      12,     NOCHAR, 0,
   ATODINFO,     CHR,    1,      13,     NOCHAR, 0,
/*8*/
   COMMENT,      SHT,    1,      0,      NODATA, 0,
   COMMENT,      SHT,    1,      2,      NODATA, 0,
   COMMENT,      SHT,    1,      4,      NODATA, 0,
   COMMENT,      SHT,    1,      6,      NODATA, 0,
/*12*/
   DESCRIPTRACE,STI,     1,      0,      NOSTRG, STAT_IDENT,
   DESCRIPTRACE,MST,     1,      12,     NOTIME, 0,
   DESCRIPTRACE,SHT,     1,      20,     NODATA, 0,
   DESCRIPTRACE,CHR,     1,      22,     NOCHAR, 0,
   DESCRIPTRACE,CHR,     1,      23,     NOCHAR, 0,
   DESCRIPTRACE,SHT,     1,      24,     NOLIST, 0,
   DESCRIPTRACE,SHT,     1,      26,     NOLIST, 0,
   DESCRIPTRACE,LNG,     1,      28,     NODATA, 0,
   DESCRIPTRACE,FLT,     1,      32,     NODATA, 0,
   DESCRIPTRACE,FLT,     1,      36,     NODATA, 0,
   DESCRIPTRACE,FLT,     1,      40,     NODATA, 0,
   DESCRIPTRACE,FLT,     1,      44,     NODATA, 0,
   DESCRIPTRACE,LNG,     1,      48,     NODATA, 0,
   DESCRIPTRACE,MST,     1,      52,     NOTIME, 0,
   DESCRIPTRACE,FLT,     1,      60,     NODATA, 0,
/*27*/
   DETECTOR,     CHR,    1,      0,      NOCHAR, 0,
   DETECTOR,     CHR,    1,      1,      NOCHAR, 0,
   DETECTOR,     STR,    10,     2,      NOSTRG, 0,
   DETECTOR,     FLT,    1,      12,     NODATA, 0,
   DETECTOR,     LNG,    1,      16,     NODATA, 0,
   DETECTOR,     LNG,    1,      20,     NODATA, 0,
/*33*/
   EVENTSETTING,STR,     4,      0,      NOSTRG, 0,
   EVENTSETTING,MST,     1,      4,      NOTIME, 0,
   EVENTSETTING,SHT,     1,      12,     NODATA, 0,
   EVENTSETTING,SHT,     1,      14,     NODATA, 0,
   EVENTSETTING,SHT,     1,      16,     NODATA, 0,
   EVENTSETTING,SHT,     1,      18,     NODATA, 0,
   EVENTSETTING,FLT,     1,      20,     NODATA, 0,
   EVENTSETTING,FLT,     1,      24,     NODATA, 0,
   EVENTSETTING,CHR,     1,      28,     NOCHAR, 0,
   EVENTSETTING,CHR,     1,      29,     NOCHAR, 0,
   EVENTSETTING,SHT,     1,      30,     NODATA, 0,
/*44*/
   FEATURE,      STI,    1,      0,      NOSTRG, STAT_IDENT,
   FEATURE,      SHT,    1,      12,     NOLIST, 0,
   FEATURE,      CHR,    1,      14,     NOCHAR, 0,
   FEATURE,      CHR,    1,      15,     NOCHAR, 0,
   FEATURE,      SHT,    1,      16,     NODATA, 0,
   FEATURE,      CHR,    1,      18,     NOCHAR, 0,
   FEATURE,      CHR,    1,      19,     NOCHAR, 0,
   FEATURE,      CHR,    1,      20,     NOCHAR, 0,
   FEATURE,      CHR,    1,      21,     NOCHAR, 0,
   FEATURE,      SHT,    1,      22,     NODATA, 0,
   FEATURE,      MST,    1,      24,     NOTIME, 0,
   FEATURE,      FLT,    1,      32,     NODATA, 0,
   FEATURE,      FLT,    1,      36,     NODATA, 0,
   FEATURE,      STT,    1,      40,     NOTIME, 0,
   FEATURE,      SHT,    1,      44,     NODATA, 0,
   FEATURE,      SHT,    1,      46,     NODATA, 0,
/*60*/
   MUXDATA,      STR,    4,      0,      "unk",  0,
   MUXDATA,      MST,    1,      4,      NOTIME, 0,
   MUXDATA,      SHT,    1,      12,     NODATA, 0,
   MUXDATA,      SHT,    1,      14,     NODATA, 0,
   MUXDATA,      FLT,    1,      16,     NODATA, 0,
   MUXDATA,      CHR,    1,      20,     NOCHAR, 0,
   MUXDATA,      CHR,    1,      21,     NOCHAR, 0,
   MUXDATA,      SHT,    1,      22,     NODATA, 0,
   MUXDATA,      LNG,    1,      24,     NODATA, 0,
   MUXDATA,      LNG,    1,      28,     NODATA, 0,
/*70*/
   ORIGIN,       LNG,    1,      0,      NODATA, 0,
   ORIGIN,       SHT,    1,      4,      NOLIST, 0,
   ORIGIN,       CHR,    1,      6,      NOCHAR, 0,
   ORIGIN,       CHR,    1,      7,      NOCHAR, 0,
   ORIGIN,       CHR,    1,      8,      NOCHAR, 0,
   ORIGIN,       CHR,    1,      9,      NOCHAR, 0,
   ORIGIN,       CHR,    1,      10,     NOCHAR, 0,
   ORIGIN,       CHR,    1,      11,     NOCHAR, 0,
   ORIGIN,       LNG,    1,      12,     NODATA, 0,
   ORIGIN,       MST,    1,      16,     NOTIME, 0,
   ORIGIN,       LLT,    1,      24,     NODATA, 0,
   ORIGIN,       LLT,    1,      32,     NODATA, 0,
   ORIGIN,       FLT,    1,      40,     NODATA, 0,
   ORIGIN,       FLT,    1,      44,     NODATA, 0,
   ORIGIN,       FLT,    1,      48,     NODATA, 0,
   ORIGIN,       FLT,    1,      52,     NODATA, 0,
   ORIGIN,       STR,    6,      56,     NOSTRG, 0,
   ORIGIN,       SHT,    1,      62,     NODATA, 0,
   ORIGIN,       FLT,    1,      64,     NODATA, 0,
   ORIGIN,       SHT,    1,      68,     NODATA, 0,
   ORIGIN,       SHT,    1,      70,     NODATA, 0,
   ORIGIN,       SHT,    1,      72,     NODATA, 0,
```

```
   ORIGIN,        SHT,    1,      74,     NODATA, 0,
   ORIGIN,        SHT,    1,      76,     NODATA, 0,
   ORIGIN,        SHT,    1,      78,     NODATA, 0,
   ORIGIN,        SHT,    1,      80,     NODATA, 0,
   ORIGIN,        SHT,    1,      82,     NODATA, 0,
   ORIGIN,        FLT,    1,      84,     NODATA, 0,
   ORIGIN,        FLT,    1,      88,     NODATA, 0,
   ORIGIN,        FLT,    1,      92,     NODATA, 0,
   ORIGIN,        STT,    1,      96,     NOTIME, 0,
/*101*/
   STAT_IDENT,    STR,    4,      0,      "unk",  0,
   STAT_IDENT,    STR,    5,      4,      NOSTRG, 0,
   STAT_IDENT,    CHR,    1,      9,      NOCHAR, 0,
   STAT_IDENT,    SHT,    1,      10,     NOLIST, 0,
/*105*/
   STATIONCOMP,   STI,    1,      0,      NOSTRG, STAT_IDENT,
   STATIONCOMP,   SHT,    1,      12,     NODATA, 0,
   STATIONCOMP,   SHT,    1,      14,     NODATA, 0,
   STATIONCOMP,   LLT,    1,      16,     NODATA, 0,
   STATIONCOMP,   LLT,    1,      24,     NODATA, 0,
   STATIONCOMP,   FLT,    1,      32,     NODATA, 0,
   STATIONCOMP,   CHR,    1,      36,     NOCHAR, 0,
   STATIONCOMP,   MIN,    1,      37,     NOLIST, 0,
   STATIONCOMP,   CHR,    1,      38,     NOCHAR, 0,
   STATIONCOMP,   CHR,    1,      39,     NOCHAR, 0,
   STATIONCOMP,   SHT,    1,      40,     NOLIST, 0,
   STATIONCOMP,   CHR,    1,      42,     NOCHAR, 0,
   STATIONCOMP,   CHR,    1,      43,     NOCHAR, 0,
   STATIONCOMP,   CHR,    1,      44,     NOCHAR, 0,
   STATIONCOMP,   CHR,    1,      45,     NOCHAR, 0,
   STATIONCOMP,   CHR,    1,      46,     NOCHAR, 0,
   STATIONCOMP,   MIN,    1,      47,     NOLIST, 0,
   STATIONCOMP,   FLT,    1,      48,     NODATA, 0,
   STATIONCOMP,   FLT,    1,      52,     NODATA, 0,
   STATIONCOMP,   FLT,    1,      56,     NODATA, 0,
   STATIONCOMP,   SHT,    1,      60,     NODATA, 0,
   STATIONCOMP,   SHT,    1,      62,     NODATA, 0,
   STATIONCOMP,   STT,    1,      64,     NOTIME, 0,
   STATIONCOMP,   FLT,    1,      68,     NODATA, 0,
   STATIONCOMP,   FLT,    1,      72,     NODATA, 0,
/*130*/
   STRUCTTAG,     CHR,    1,      0,        "S", 0,
   STRUCTTAG,     CHR,    1,      1,     MACHINE, 0,
   STRUCTTAG,     SHT,    1,      2,      NODATA, 0,
   STRUCTTAG,     LNG,    1,      4,      NODATA, 0,
/*134*/
   TIMECORRECTION, STI,   1,      0,      NOSTRG, STAT_IDENT,
   TIMECORRECTION, MST,   1,      12,     NOTIME, 0,
   TIMECORRECTION, FLT,   1,      20,     NODATA, 0,
   TIMECORRECTION, CHR,   1,      24,     NOCHAR, 0,
   TIMECORRECTION, CHR,   1,      25,     NOCHAR, 0,
   TIMECORRECTION, STT,   1,      26,     NOTIME, 0,
   TIMECORRECTION, SHT,   1,      30,     NODATA, 0,
/*141*/
   TRIGGERS,      STI,    1,      0,      NOSTRG, STAT_IDENT,
   TRIGGERS,      SHT,    1,      12,     NODATA, 0,
   TRIGGERS,      SHT,    1,      14,     NODATA, 0,
   TRIGGERS,      SHT,    1,      16,     NODATA, 0,
   TRIGGERS,      SHT,    1,      18,     NODATA, 0,
   TRIGGERS,      SHT,    1,      20,     NODATA, 0,
   TRIGGERS,      SHT,    1,      22,     NODATA, 0,
   TRIGGERS,      MST,    1,      24,     NOTIME, 0,
/*149*/
   TRIGSETTING,   STR,    4,      0,       "unk", 0,
   TRIGSETTING,   MST,    1,      4,      NOTIME, 0,
   TRIGSETTING,   SHT,    1,      12,     NODATA, 0,
   TRIGSETTING,   SHT,    1,      14,     NODATA, 0,
   TRIGSETTING,   SHT,    1,      16,     NODATA, 0,
   TRIGSETTING,   SHT,    1,      18,     NODATA, 0,
   TRIGSETTING,   SHT,    1,      20,     NODATA, 0,
   TRIGSETTING,   SHT,    1,      22,     NODATA, 0,
   TRIGSETTING,   FLT,    1,      24,     NODATA, 0,
   TRIGSETTING,   FLT,    1,      28,     NODATA, 0,
   TRIGSETTING,   CHR,    1,      32,     NOCHAR, 0,
   TRIGSETTING,   CHR,    1,      33,     NOCHAR, 0,
   TRIGSETTING,   SHT,    1,      34,     NODATA, 0
/*162*/
};

PRIVATE unsigned int structure_offset[]={
/*  0,  1,  2, 3, 4,  5,  6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21
    0,101, 130, 0, 0,105, 60,12, 0, 0, 44,  0,  0,  0, 70,  0,  0,  0,  0,  0,  8,  0
};

PRIVATE unsigned int structure_size[]={
/*  0 */        0,
/*  1 */        sizeof(SUDS_STATIDENT),
/*  2 */        sizeof(SUDS_STRUCTTAG),
/*  3 */        0,
/*  4 */        0,
/*  5 */        sizeof(SUDS_STATIONCOMP),
/*  6 */        sizeof(SUDS_MUXDATA),
/*  7 */        sizeof(SUDS_DESCRIPTRACE),
/*  8 */        0,
/*  9 */        0,
/* 10 */        sizeof(SUDS_FEATURE),
/* 11 */        0,
/* 12 */        0,
/* 13 */        0,
/* 14 */        sizeof(SUDS_ORIGIN),
/* 15 */        0,
/* 16 */        0,
/* 17 */        0,
/* 18 */        0,
/* 19 */        0,
/* 20 */        sizeof(SUDS_COMMENT),
/* 21 */        0,
/* 22 */        0,
/* 23 */        0,
/* 24 */        0,
/* 25 */        sizeof(SUDS_TRIGGERS),
/* 26 */        sizeof(SUDS_TRIGSETTING),
/* 27 */        sizeof(SUDS_EVENTSETTING),
/* 28 */        sizeof(SUDS_DETECTOR),
/* 29 */        sizeof(SUDS_ATODINFO),
/* 30 */        sizeof(SUDS_TIMECORRECTION),
/* 31 */        0
};


/*-------------------------------------------------------------------------*
 *                          initialize_value                               *
 *-------------------------------------------------------------------------*/
/* Initialize a suds value.                                                */

PRIVATE
void initialize_value (i, ptr)
int i;
void far * ptr;
{
   char far * farstring;

   switch(initializer[i].ftype) {
      case CHR:
         *((char far *)ptr) = initializer[i].initval[0];
         break;
      case BTS:
      case MIN:
         *((char far *)ptr) = atoi(initializer[i].initval);
         break;
      case STR:
         farstring = (char far *) initializer[i].initval;
         u_strcpy (((char far *)ptr), farstring);
         break;
      case SHT:
         *((short far *)ptr) = atoi(initializer[i].initval);
```

```
            break;
         case LNG:
            *((long far *)ptr) = atoi(initializer[i].initval);
            break;
         case FLT:
            *((float far *)ptr) = atof(initializer[i].initval);
            break;
         case LLT:
         case DBL:
            *((double far *)ptr) = atof(initializer[i].initval);
            break;
         case STT:
            *((ST_TIME far *)ptr) = atoi(initializer[i].initval);
            break;
         case MST:
            *((MS_TIME far *)ptr) = atof(initializer[i].initval);
            break;
         case BTW:
            *((unsigned short far *)ptr) = atoi(initializer[i].initval);
            break;
         case CAL:
         case CPX:
         case STI:
         default:
            break;
      }
   }

   /*-----------------------------------------------------------------------*
    *                          suds_initialize                              *
    *-----------------------------------------------------------------------*/
   /* Initialize a suds structure.                                          */

   PUBLIC
   void suds_initialize (type, ptr)
   unsigned int type;
   void far * ptr;
   {
      register unsigned int i;

      for (i = structure_offset[type]; initializer[i].fstype--type; i++) {
         if (initializer[i].nextfstype != 0) {
            suds_initialize (initializer[i].nextfstype, ((char far *)ptr)+initializer[i].of
         } else initialize_value (i, ((char far *)ptr)+initializer[i].offset);
      }
   }

   /*-----------------------------------------------------------------------*
    *                        suds_initialize_tag                            *
    *-----------------------------------------------------------------------*/
   /* Initialize a suds structtag.                                          */

   PUBLIC
   void suds_initialize_tag (type, ptr)
   unsigned int type;
   SUDS_STRUCTTAG far * ptr;
   {
      suds_initialize (STRUCTTAG, ptr);
      ptr->id_struct = type;
      ptr->len_struct = structure_size[type];
      ptr->len_data = 0;
   }
```

```
/* FILE: msudsini.h                                (D. Tottingham  04/26/92)

This is an include file of defines, data structure definitions and
external declarations that are common in the msudsini module.

*/

#ifndef _MSUDSINI_
#define _MSUDSINI_

/*************************************************************************
                                INCLUDES

*************************************************************************/
#include "mconst.h"
#include "suds.h"


/*************************************************************************
                          EXTERNAL DECLARATIONS

These functions can be called from all modules that include this file.

*************************************************************************/
PUBLIC void suds_initialize (unsigned int, void far *);
PUBLIC void suds_initialize_tag (unsigned int, SUDS_STRUCTTAG far *);

#endif
```

```
/* FILE: mtrigger.c                                (D. Tottingham  04/26/92)

This is a collection of routines that manage the trigger for xdetect.  The
low-level routines contained in this module (namely, triggered_on_channel
and energy_valid_on_channel) are based on the algorithm presented in
Lee's "Location and Real-Time Detection of Microearthquakes Along the San Andreas
Fault System in Central California."  The program has been written and compiled
medium model.  The following public functions are defined in this file:

t_display_event_params ()        write event parameters to stream
t_display_triggers ()            write event information to stream
t_display_trigger_params ()      write trigger parameters to stream
t_event_detected ()              check to see whether event is done
t_event_done ()                  return event status
t_get_channel_trigger_state ()   get trigger state for given channel
t_get_eventsetting ()            get T_EVENT structure
t_get_head_trigger ()            get trigger information at head of channel queue
t_get_last_trigger ()            get last trigger
t_get_next_trigger ()            get next trigger from channel queue
t_get_trigger_status ()          get trigger_enabled
t_get_trigger_time ()            get trigger time for a channel
t_get_trigsetting ()             get T_TRIGGER structure
t_initialize ()                  initialize the trigger algorithm
t_initialize_params ()           initialize the trigger parameters
t_reset_event ()                 reset event variables
t_reset_trigger ()               reset trigger variables
t_set_CriticalAlpha ()           set CriticalAlpha constant
t_set_CriticalBeta ()            set CriticalBeta constant
t_set_CriticalGamma ()           set CriticalGamma constant
t_set_CriticalMu ()              set CriticalMu constant
t_set_CriticalNu ()              set CriticalNu constant
t_set_event ()                   set event detected flag to TRUE
t_set_EventContinueCount ()      set EventContinuationCount constant
t_set_first_difference ()        turn first difference on/off
t_set_LTA_lower_bound            set LTA lower bound
t_set_LTAWindow ()               set length of long-term average
t_set_MaxEventTime ()            set MaxEventTime constant
t_set_MinEventTime ()            set MinEventTime constant
t_set_netwname ()                set network name
t_set_STAWindow ()               set length of short-term average
t_set_Trigger ()                 set trigger status for given channel
t_set_TriggerConfirmCount ()     set TriggerConfirmationCount constant
t_set_TriggerTimeLimit ()        set TriggerTimeLimit constant
t_set_trigger_status ()          set trigger_enabled flag
t_toggle_trigger_status ()       toggle trigger_enabled flag
t_trigger_check ()               look for new triggers

EXTERNAL FUNCTIONS CALLED:

dm_get_first_buffer ()           get first new demux buffer from demux queue
dm_get_next_buffer ()            get next buffer from demux queue
h_set_trigger_status ()          set trigger status
q_delete_link ()                 delete a data link from a data queue
q_enqueue ()                     enqueue a data link on a data queue
q_initialize ()                  initialize a data queue
q_insert_link ()                 insert a data link from a data queue
s_display_triggers ()            display triggers on the screen
suds_initialize ()               initialize a suds structure
suds_initialize_tag ()           initialize a suds structtag
u_build_timeline ()              convert abs. time to an ascii string
u_ispow2 ()                      determine whether i_num is a power of two
u_log2 ()                        compute the base-2 logarithm of i_num
u_strncpy ()                     copies n characters of far string2 to far string1
u_timestamp ()                   get timestamp


HISTORY:
   none


/*************************************************************************
                              INCLUDE FILES
*************************************************************************/
#include <ctype.h>
#include <io.h>
#include <malloc.h>
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys\types.h>
#include <sys\timeb.h>
#include <time.h>

#include "mconst.h"
#include "mdemux.h"
#include "merror.h"
#include "mqueue.h"
#include "mscreen.h"
#include "mscrnhdr.h"
#include "mstation.h"
#include "msudsini.h"
#include "mtrigger.h"
#include "mutils.h"


/*************************************************************************
                                  GLOBALS
*************************************************************************/
PRIVATE T_EVENT eventsetting, far * e = &eventsetting;
PRIVATE T_TRIGGER trigsetting, far * t = &trigsetting;
PRIVATE Q_QUEUE channel_queue, trigger_queue;
PRIVATE Q_LINK *head_ptr;
PRIVATE Q_TRIGGER *lasttrig;
PRIVATE unsigned int trigger_location;
PRIVATE double last_trigger;
PRIVATE FLAG event_flag;
PRIVATE FLAG trigger_enabled;


/*-----------------------------------------------------------------------*
 *                        energy_valid_on_channel                        *
 *-----------------------------------------------------------------------*/
/* Check for energy on given channel.                                    */

PRIVATE
FLAG energy_valid_on_channel (tch, channel_buffer, n_pts)
Q_CHANNEL far * tch;
int far * channel_buffer;
unsigned n_pts;
{
   FLAG energy_valid;
   int dx, new_sta, new_lta;
   int gamma;
   unsigned k, inspect;

   /* Initialize sta/lta and flags */
   new_sta = (*tch).info.abs_sta;
   new_lta = (*tch).info.abs_lta;
   energy_valid = FALSE;
   inspect = 0;

   /* Main loop */
   for (k = 1; k < n_pts; k++) {
      /* Compute first difference and new sta */
      dx = abs (channel_buffer[k] - channel_buffer[k-1]);
      new_sta += (dx - new_sta) >> t->sta_shift;

      /* Compute new lta */
      new_lta += ((new_sta - new_lta) >> t->lta_shift);
      if (new_lta < t->lta_lower_bound) new_lta = t->lta_lower_bound;
```

```
      /* Do we still have an event? */
      if (energy_valid == FALSE) {
         if ((k - inspect) <= e->event_continuation_count) {
            gamma = new_sta / (*tch).info.lta;

            /* Do we have enough energy? */
            if (gamma < e->critical_gamma)
               inspect = k;
         }
         else energy_valid = TRUE;
      }
   }

   /* Store channel's sta/lta */
   (*tch).info.abs_sta = new_sta;
   (*tch).info.abs_lta = new_lta;

   return (energy_valid);
}

/*------------------------------------------------------------------------*
 *                          triggered_on_channel                          *
 *------------------------------------------------------------------------*/
/* Look for a trigger on the given channel.                               */

PRIVATE
FLAG triggered_on_channel (qch, channel_buffer, mid_point, n_pts )
Q_CHANNEL far * qch;
int far * channel_buffer;
unsigned int mid_point;
unsigned int n_pts;
{
   int dx, ns, new_sta, new_lta;
   int alpha, beta;
   unsigned k;
   FLAG trigger_flag;

   /* Initialize sta and lta for this channel */
   new_sta = (*qch).info.abs_sta;
   new_lta = (*qch).info.abs_lta;

   /* Initialize trigger_time and flags */
   trigger_location = 0;
   trigger_flag = FALSE;

   /* Main loop */
   for (k = 1; k < n_pts; k++) {
      /* Compute first difference if FirstDifference is ON and new sta */
      if (t->first_difference)
         dx = abs (channel_buffer[k] - channel_buffer[k-1]);
      else dx = channel_buffer[k];

      new_sta += (dx - new_sta) >> t->sta_shift;

      /* Compute new lta */
      new_lta += (new_sta - new_lta) >> t->lta_shift;
      if (new_lta < t->lta_lower_bound) new_lta = t->lta_lower_bound;

      /* If we have not hit a trigger, continue checking */
      if (trigger_flag == FALSE) {
         /* Compute signal and noise */
         ns = abs (channel_buffer[k] - mid_point);
         (*qch).noise += (ns - (*qch).noise) >> 8;
         if ((*qch).noise < NOISE_INIT) (*qch).noise = NOISE_INIT;

         /* Compute alpha */
         alpha = dx / new_lta;

         /* Do we have a trigger? */
         if (alpha > t->critical_alpha) {
            trigger_flag = TRUE;
            trigger_location = k;
            (*qch).info.lta = new_lta;
            (*qch).info.sta = new_sta;
         } else continue;
      }

      /* We have hit a trigger.  Are we still within time window? */
      if ((k - trigger_location) <= t->trigger_confirmation_count) {

         /* We are within the time window so calculate beta */
         beta = new_sta / (*qch).info.lta;

         /* Do we have enough energy? */
         if (beta < t->critical_beta)
            trigger_flag = FALSE;
      }
   }

   /* Store channel's LTA and STA */
   (*qch).info.abs_sta = new_sta;
   (*qch).info.abs_lta = new_lta;

   return (trigger_flag);
}

/*------------------------------------------------------------------------*
 *                          trigger_window_check                          *
 *------------------------------------------------------------------------*/
/* Make sure that current triggers are still within the allowed trigger
   window.                                                                */

PRIVATE
void trigger_window_check (time)
double time;
{
   Q_LINK *current, *previous, *temp;
   Q_TYPE type;

   current = trigger_queue.head;
   previous = NULL;
   while (current != NULL) {
      if (fabs(time - current->type.trigger->trigger_time) >
            t->trigger_time_limit) {
         temp = current;
         current->type.trigger->qch->info.trig_value = OFF;
         current = current->next;
         if (q_delete_link (&trigger_queue, temp, previous, &type))
            free (type.trigger);
      } else {
         previous = current;
         current = current->next;
      }
   }
}

/*------------------------------------------------------------------------*
 *                          t_display_event_params                        *
 *------------------------------------------------------------------------*/
/* Write event parameters to the specified stream.                        */

PUBLIC
void t_display_event_params (stream)
FILE * stream;
{
   fprintf (stream, "min_event_time             = %5f\n", e->min_event_time);
   fprintf (stream, "max_event_time             = %5f\n", e->max_event_time);
   fprintf (stream, "event_continuation_count   = %5d\n", e->event_continuation_coun
   fprintf (stream, "critical_gamma             = %5d\n", e->critical_gamma);
   fprintf (stream, "critical_mu                = %5d\n", e->critical_mu);
}

/*------------------------------------------------------------------------*
 *                          t_display_triggers                            *
 *------------------------------------------------------------------------*/
```

```
/* Write event information to the specified stream.                    */

PUBLIC
void t_display_triggers (stream)
FILE * stream;
{
   Q_LINK * head;
   char * timeline;

   timeline = u_build_timeline (last_trigger, TRUE);

   fprintf (stream, "\n\n\nEVENT DETECTED      %s\n\n",
            timeline );
   fprintf (stream, "CH    AVG                        \n");
   fprintf (stream, "      Short   Long   Time        \n");
   fprintf (stream, "---   -----   -----  -------------------------- \n");

   free (timeline);

   head = channel_queue.head;
   while (head != NULL) {
      if (head->type.channel->info.trig_value) {
         timeline = u_build_timeline (head->type.channel->info.trig_time, TRUE);
         fprintf (stream, "%3d %c %4d    %4d  %s\n",
                  head->type.channel->channel,
                  (head->type.channel->info.trig_value == ON) ? ' ': 'E',
                  head->type.channel->info.sta,
                  head->type.channel->info.lta, timeline);
         free (timeline);
      }
      head = head->next;
   }
   fprintf (stream, "\n");
}

/*---------------------------------------------------------------------*
 *                      t_display_trigger_params                       *
 *---------------------------------------------------------------------*/
/* Write trigger parameters to specified stream.                       */

PUBLIC
void t_display_trigger_params (stream)
FILE * stream;
{
   fprintf (stream, "critical_alpha            = %5d\n", t->critical_alpha);
   fprintf (stream, "trigger_confirmation_count = %5d\n", t->trigger_confirmation_count);
   fprintf (stream, "critical_beta             = %5d\n", t->critical_beta);
   fprintf (stream, "trigger_time_limit        = %5f\n", t->trigger_time_limit);
   fprintf (stream, "critical_nu               = %5d\n", t->critical_nu);
   fprintf (stream, "lta_window                = %5d\n", t->lta_window);
   fprintf (stream, "sta_window                = %5d\n", t->sta_window);
}

/*---------------------------------------------------------------------*
 *                          t_event_done                               *
 *---------------------------------------------------------------------*/
/* Check to see whether event is done.  A minimum post trigger window
   is saved regardless of energy content.  Past this point until the
   maximum post trigger window is reached, the event must pass the
   energy_valid_on_channel test.                                       */

PUBLIC
FLAG t_event_done ()
{
   Q_BUFFER far * b_head;
   Q_LINK * c_head;
   int far * channel_buffer;
   unsigned channel_ctr, i;

   for (b_head = dm_get_first_buffer(); b_head != NULL;
        b_head = dm_get_next_buffer ()) {
      if ((b_head->info.begintime - last_trigger) < e->min_event_time) {
         /* Just update the sta and lta */
         for (c_head = channel_queue.head; c_head != NULL;
              c_head = c_head->next ) {
            channel_buffer = b_head->data + b_head->info.blocksize * c_head->type.ch
            energy_valid_on_channel (c_head->type.channel, channel_buffer, b_head->i
         }
      } else if ((b_head->info.begintime - last_trigger) < e->max_event_time) {
         /* Scan buffer for any activity */
         channel_ctr = 0;

         for (c_head = channel_queue.head; c_head != NULL;
              c_head = c_head->next) {
            channel_buffer = b_head->data + b_head->info.blocksize * c_head->type.ch
            if (energy_valid_on_channel (c_head->type.channel, channel_buffer, b_hea
               channel_ctr++;
         }
         event_flag = (channel_ctr < e->critical_mu) ? FALSE : TRUE;
         if (!event_flag) return ( TRUE);
      } else {
         event_flag = FALSE;
         return ( TRUE);
      }
   }
   return ( FALSE);
}

/*---------------------------------------------------------------------*
 *                          t_event_detected                           *
 *---------------------------------------------------------------------*/
/* Return event status.                                                */

PUBLIC
FLAG t_event_detected ()
{
   return (event_flag);
}

/*---------------------------------------------------------------------*
 *                    t_get_channel_trigger_state                      *
 *---------------------------------------------------------------------*/
/* Get trigger state for given channel.                                */

PUBLIC
unsigned int t_get_channel_trigger_state (channel)
unsigned int channel;
{
   Q_LINK * head;

   for (head = channel_queue.head; head != NULL; head = head->next) {
      if (head->type.channel->channel == channel)
         return (head->type.channel->info.trig_value + 1);
   }
   return CHANNEL_TRIGGER_DISABLED;
}

/*---------------------------------------------------------------------*
 *                        t_get_eventsetting                           *
 *---------------------------------------------------------------------*/
/* Get T_EVENT structure.                                              */

PUBLIC
T_EVENT far * t_get_eventsetting ()
{
   return (e);
}

/*---------------------------------------------------------------------*
 *                        t_get_head_trigger                           *
 *---------------------------------------------------------------------*/
/* Get trigger information at head of channel queue.                   */

PUBLIC
Q_CHANNEL far * t_get_head_trigger ()
{
```

```
   head_ptr = channel_queue.head;
   if (head_ptr != NULL)
      return (head_ptr->type.channel);
   else return (NULL);
}

/*-------------------------------------------------------------------------*
 *                          t_get_last_trigger                             *
 *-------------------------------------------------------------------------*/
/* Get last trigger.                                                       */

PUBLIC
double t_get_last_trigger ()
{
   return (last_trigger);
}

/*-------------------------------------------------------------------------*
 *                          t_get_next_trigger                             *
 *-------------------------------------------------------------------------*/
/* Get next trigger from channel queue.                                    */

PUBLIC
Q_CHANNEL far * t_get_next_trigger ()
{
   head_ptr = head_ptr->next;
   if (head_ptr != NULL)
      return (head_ptr->type.channel);
   else return (NULL);
}

/*-------------------------------------------------------------------------*
 *                          t_get_trigger_status                           *
 *-------------------------------------------------------------------------*/
/* Get trigger_enabled.                                                    */

PUBLIC
FLAG t_get_trigger_status ()
{
   return trigger_enabled;
}

/*-------------------------------------------------------------------------*
 *                          t_get_trigger_time                             *
 *-------------------------------------------------------------------------*/
/* Get trigger time for specified channel.                                 */

PUBLIC
double t_get_trigger_time (channel)
unsigned int channel;
{
   Q_LINK * head;


   for (head = channel_queue.head; head != NULL; head = head->next) {
      if (head->type.channel->channel == channel)
         return (head->type.channel->info.trig_time);
   }
   er_abort (T_INVALID_CHANNEL);
}

/*-------------------------------------------------------------------------*
 *                          t_get_trigsetting                              *
 *-------------------------------------------------------------------------*/
/* Return T_TRIGGER structure.                                             */

PUBLIC
T_TRIGGER far * t_get_trigsetting ()
{
   return (t);
}

/*-------------------------------------------------------------------------*
 *                          t_initialize                                   *
 *-------------------------------------------------------------------------*/
/* Initialize the trigger algorithm.                                       */

PUBLIC
void t_initialize ()
{
   Q_LINK *c_head;
   for (c_head = channel_queue.head; c_head != NULL; c_head = c_head->next) {
      c_head->type.channel->signal = 0;
      c_head->type.channel->noise = NOISE_INIT;
      c_head->type.channel->info.trig_value = FALSE;
      c_head->type.channel->info.abs_sta = 0;
      c_head->type.channel->info.abs_lta = t->lta_lower_bound;
   }
}

/*-------------------------------------------------------------------------*
 *                          t_initialize_params                            *
 *-------------------------------------------------------------------------*/
/* Initialize trigger parameters.                                          */

PUBLIC
void t_initialize_params ()
{
   suds_initialize_tag (EVENTSETTING, &(e->structtag));
   suds_initialize (EVENTSETTING, &(e->eventsetting));

   e->eventsetting.algorithm = 'x';
   u_strncpy (e->eventsetting.netwname, (char far *)(st_get_netwname()), 4);

   e->critical_gamma = CRITICAL_GAMMA;
   e->critical_mu = CRITICAL_MU;
   e->event_continuation_count = EVENT_CONTINUATION_COUNT;
   e->max_event_time = MAX_EVENT_TIME;
   e->min_event_time = MIN_EVENT_TIME;
   e->eventsetting.beginttime = u_timestamp ();

   event_flag = FALSE;

   trigger_enabled = TRIGGER_ENABLED;
   h_set_trigger_status (trigger_enabled);

   suds_initialize_tag (TRIGSETTING, &(t->structtag));
   suds_initialize (TRIGSETTING, &(t->trigsetting));

   t->trigsetting.algorithm = 'x';
   u_strncpy (t->trigsetting.netwname, (char far *)(st_get_netwname()), 4);

   t->critical_alpha = CRITICAL_ALPHA;
   t->critical_beta = CRITICAL_BETA;
   t->critical_nu = CRITICAL_NU;
   t->trigger_confirmation_count = TRIGGER_CONFIRMATION_COUNT;
   t->trigger_time_limit = TRIGGER_TIME_LIMIT;
   t->lta_shift = LTA_SHIFT;
   t->sta_shift = STA_SHIFT;
   t->lta_window = (unsigned int) pow (((double) t->lta_shift), 2.0);
   t->sta_window = (unsigned int) pow (((double) t->sta_shift), 2.0);
   t->trigsetting.beginttime = u_timestamp ();
   t->first_difference = FIRST_DIFFERENCE;
   t->lta_lower_bound = LTA_LOWER_BOUND;

   q_initialize (&channel_queue);
   q_initialize (&trigger_queue);
   last_trigger = 0;
   lasttrig = NULL;
}

/*-------------------------------------------------------------------------*
 *                          t_reset_event                                  *
 *-------------------------------------------------------------------------*/
/* Reset event variables.                                                  */
```

```
PUBLIC
void t_reset_event ()
{
   event_flag = FALSE;
}

/*-------------------------------------------------------------------------*
 *                          t_reset_trigger                                *
 *-------------------------------------------------------------------------*/
/* Reset trigger variables.                                                */

PUBLIC
void t_reset_trigger ()
{
   Q_LINK * head;

   for (head = channel_queue.head; head != NULL; head = head->next)
      head->type.channel->info.abs_lta = head->type.channel->info.abs_sta;
}

/*-------------------------------------------------------------------------*
 *                          t_set_CriticalAlpha                            *
 *-------------------------------------------------------------------------*/
/* Set CriticalAlpha constant.                                             */

PUBLIC
void t_set_CriticalAlpha (c)
int c;
{
   t->critical_alpha = c;
   t->trigsetting.beginttime = u_timestamp ();
}

/*-------------------------------------------------------------------------*
 *                          t_set_CriticalBeta                             *
 *-------------------------------------------------------------------------*/
/* Set CriticalBeta constant.                                              */

PUBLIC
void t_set_CriticalBeta (c)
int c;
{
   t->critical_beta = c;
   t->trigsetting.beginttime = u_timestamp ();
}

/*-------------------------------------------------------------------------*
 *                          t_set_CriticalGamma                            *
 *-------------------------------------------------------------------------*/
/* Set CriticalGamma constant.                                             */

PUBLIC
void t_set_CriticalGamma (c)
int c;
{
   e->critical_gamma = c;
   e->eventsetting.beginttime = u_timestamp ();
}

/*-------------------------------------------------------------------------*
 *                          t_set_CriticalMu                               *
 *-------------------------------------------------------------------------*/
/* Set CriticalMu constant.                                                */

PUBLIC
void t_set_CriticalMu (c)
int c;
{
   e->critical_mu = c;
   e->eventsetting.beginttime = u_timestamp ();
}

/*-------------------------------------------------------------------------*
 *                          t_set_CriticalNu                               *
 *-------------------------------------------------------------------------*/
/* Set CriticalNu constant.                                                */

PUBLIC
void t_set_CriticalNu (c)
int c;
{
   t->critical_nu = c;
   t->trigsetting.beginttime = u_timestamp ();
}

/*-------------------------------------------------------------------------*
 *                          t_set_event                                    *
 *-------------------------------------------------------------------------*/
/* Set event detected flag to TRUE.                                        */

PUBLIC
void t_set_event ()
{
   event_flag = TRUE;
}

/*-------------------------------------------------------------------------*
 *                          t_set_EventContinueCount                       *
 *-------------------------------------------------------------------------*/
/* Set EventContinuationCount constant.                                    */

PUBLIC
void t_set_EventContinueCount (count)
int count;
{
   e->event_continuation_count = count;
   e->eventsetting.beginttime = u_timestamp ();
}

/*-------------------------------------------------------------------------*
 *                          t_set_first_difference                         *
 *-------------------------------------------------------------------------*/
/* Turn first difference on/off.                                           */

PUBLIC
void t_set_first_difference (status)
FLAG status;
{
   t->first_difference = status;
}

/*-------------------------------------------------------------------------*
 *                          t_set_LTA_lower_bound                          *
 *-------------------------------------------------------------------------*/
/* Set LTA lower bound.                                                    */

PUBLIC
void t_set_LTA_lower_bound (lb)
unsigned int lb;
{
   t->lta_lower_bound = lb;
}

/*-------------------------------------------------------------------------*
 *                          t_set_LTAWindow                                *
 *-------------------------------------------------------------------------*/
/* Set length of long-term average.                                        */

PUBLIC
void t_set_LTAWindow (length)
unsigned int length;
{
   if (! u_ispow2 ((unsigned long) length))
      er_abort (T_INVALID_LTASIZE);
   t->lta_window = length;
   t->lta_shift = (unsigned int) u_log2 ((unsigned long) length);
```

```
/* printf ("lta_shift = %u\n", t->lta_shift);   */
}

/*-------------------------------------------------------------------------*
 *                          t_set_MaxEventTime                             *
 *-------------------------------------------------------------------------*/
/* Set MaxEventTime constant.                                              */

PUBLIC
void t_set_MaxEventTime (m)
double m;
{
   e->max_event_time = m;
   e->eventsetting.beginttime = u_timestamp ();
}

/*-------------------------------------------------------------------------*
 *                          t_set_MinEventTime                             *
 *-------------------------------------------------------------------------*/
/* Set MinEventTime constant.                                              */

PUBLIC
void t_set_MinEventTime (m)
double m;
{
   e->min_event_time = m;
   e->eventsetting.beginttime = u_timestamp ();
}

/*-------------------------------------------------------------------------*
 *                          t_set_netwname                                 *
 *-------------------------------------------------------------------------*/
/* Set network name.                                                       */

PUBLIC
void t_set_netwname (nwn)
char nwn[];
{
   u_strncpy (t->trigsetting.netwname, (char far *)nwn, 4);
   u_strncpy (e->eventsetting.netwname, (char far *)nwn, 4);
   t->trigsetting.beginttime = u_timestamp ();
   e->eventsetting.beginttime = u_timestamp ();
}

/*-------------------------------------------------------------------------*
 *                          t_set_STAWindow                                *
 *-------------------------------------------------------------------------*/
/* Set length of short-term average.                                       */

PUBLIC
void t_set_STAWindow (length)
unsigned int length;
{
   if (! u_ispow2 ((unsigned long) length))
      er_abort (T_INVALID_STASIZE);
   t->sta_window = length;
   t->sta_shift = (unsigned int) u_log2 ((unsigned long) length);
/* printf ("sta_shift = %u\n", t->sta_shift);   */
}

/*-------------------------------------------------------------------------*
 *                          t_set_Trigger                                  *
 *-------------------------------------------------------------------------*/
/* Set trigger status for given channel.                                   */

PUBLIC
void t_set_Trigger (channel, trigger_status)
unsigned int channel;
FLAG trigger_status;
{
   Q_CHANNEL far * qch;
   Q_TYPE type;

   if (trigger_status) {
      qch = (Q_CHANNEL far *) _fmalloc (sizeof(Q_CHANNEL));
      if (qch == NULL) er_abort (T_NO_STORAGE);

      suds_initialize_tag (TRIGGERS, &qch->structtag);
      suds_initialize (TRIGGERS, &qch->info);

      qch->channel = channel;
      qch->info.tr_name = st_get_station(channel)->info.sc_name;
      type.channel = qch;
      q_enqueue (&channel_queue, type);
   }
}

/*-------------------------------------------------------------------------*
 *                          t_set_TriggerConfirmCount                      *
 *-------------------------------------------------------------------------*/
/* Set TriggerConfirmationCount constant.                                  */

PUBLIC
void t_set_TriggerConfirmCount (count)
int count;
{
   t->trigger_confirmation_count = count;
   t->trigsetting.beginttime = u_timestamp ();
}

/*-------------------------------------------------------------------------*
 *                          t_set_TriggerTimeLimit                         *
 *-------------------------------------------------------------------------*/
/* Set TriggerTimeLimit constant.                                          */

PUBLIC
void t_set_TriggerTimeLimit (timelimit)
double timelimit;
{
   t->trigger_time_limit = timelimit;
   t->trigsetting.beginttime = u_timestamp ();
}

/*-------------------------------------------------------------------------*
 *                          t_set_trigger_status                           *
 *-------------------------------------------------------------------------*/
/* Set trigger_enabled flag.                                               */

PUBLIC
void t_set_trigger_status (trigger_status)
FLAG trigger_status;
{
   trigger_enabled = trigger_status;
}

/*-------------------------------------------------------------------------*
 *                          t_toggle_trigger_status                        *
 *-------------------------------------------------------------------------*/
/* Toggle the trigger_enabled flag.                                        */

PUBLIC
void t_toggle_trigger_status ()
{
   trigger_enabled = (trigger_enabled) ? FALSE : TRUE;
}

/*-------------------------------------------------------------------------*
 *                          t_trigger_check                                *
 *-------------------------------------------------------------------------*/
/* Look for new triggers on all participating channels.                    */

PUBLIC
FLAG t_trigger_check ()
{
   Q_BUFFER far * b_head;
   Q_LINK *c_head, *t_head, *previous;
```

```
Q_TRIGGER * tch;
Q_TYPE type;
int far * channel_buffer;
unsigned int new_trigger_ctr, trigger_ctr;
FLAG done;
long offset;

if (! trigger_enabled)
   return FALSE;

new_trigger_ctr = trigger_ctr = 0;

for (b_head = dm_get_first_buffer(); b_head != NULL;
     b_head = dm_get_next_buffer()) {

   /* Clear out the ghosts */
   trigger_window_check (b_head->info.begintime);

   /* Don't trigger on bank switched buffers */
   if (b_head->bank_switched) {
      continue;
   }

   /* Scan buffer for any new triggers */
   for (c_head = channel_queue.head; c_head != NULL; c_head = c_head->next) {
      offset = b_head->info.blocksize * c_head->type.channel->channel;
      channel_buffer = b_head->data + b_head->info.blocksize * c_head->type.channel->
      if (triggered_on_channel (c_head->type.channel, channel_buffer, b_head->info.do
                                b_head->info.blocksize)) {
         tch = (Q_TRIGGER *) calloc (1, sizeof(Q_TRIGGER));
         if (tch == NULL) er_abort (T_NO_STORAGE);

         tch->trigger_flag = FALSE;
         tch->trigger_time = b_head->info.begintime +
            (((double) trigger_location) / b_head->info.dig_rate);
         tch->qch = c_head->type.channel;
         type.trigger = tch;

         /* Put trigger in trigger list according to trigger time */
         previous = NULL;
         for (done = FALSE, t_head = trigger_queue.head;
              !done && t_head != NULL;
              t_head = t_head->next)
            if (tch->trigger_time < t_head->type.trigger->trigger_time) {
               q_insert_link (&trigger_queue, previous, type);
               done = TRUE;
            } else
               previous = t_head;

         if (!done) q_enqueue (&trigger_queue, type);
         new_trigger_ctr++;

      } else if (c_head->type.channel->info.trig_value == FALSE) {
         c_head->type.channel->info.lta = c_head->type.channel->info.abs_lta;
         c_head->type.channel->info.sta = c_head->type.channel->info.abs_sta;
      }
   }
}

/* Are there any new triggers? */
event_flag = FALSE;
if (new_trigger_ctr) {
   for (t_head = trigger_queue.head; t_head != NULL; t_head = t_head->next) {
      if (t_head->type.trigger->qch->info.trig_value == OFF) {
         t_head->type.trigger->trigger_flag = TRUE;
         t_head->type.trigger->qch->info.trig_value = ON;
         t_head->type.trigger->qch->info.trig_time = t_head->type.trigger->trigger_t
      }
      if (t_head->type.trigger->trigger_flag) {
         if (!event_flag && last_trigger < t_head->type.trigger->trigger_time) {
            last_trigger = t_head->type.trigger->trigger_time;
            lasttrig = t_head->type.trigger;
         }
         trigger_ctr++;
      }
      if (trigger_ctr >= t->critical_nu)
         event_flag = TRUE;        /* we have triggered an event */
   }
}
if (event_flag && lasttrig) lasttrig->qch->info.trig_value = EVENT_TRIGGER;
s_display_triggers ();
return (event_flag);
}
```

```
/* FILE: mtrigger.h                                (D. Tottingham  04/26/92)

This is an include file of defines, data structure definitions and
external declarations that are common in the mtrigger module.

*/

#ifndef _MTRIGGER_
#define _MTRIGGER_

/*************************************************************************
                                INCLUDES
*************************************************************************/
#include "mconst.h"
#include "mqueue.h"
#include "suds.h"


/*************************************************************************
                                DEFINES
*************************************************************************/
#define TRIGGER_ENABLED                 TRUE

#define EVENT_TRIGGER                   2
#define NOISE_INIT                      1

/* STA/LTA Constants */
#define LTA_LOWER_BOUND                 30
#define STA_SHIFT                       4
#define LTA_SHIFT                       8

/* Trigger Constants */
#define CRITICAL_ALPHA                  15
#define CRITICAL_BETA                   5
#define TRIGGER_CONFIRMATION_COUNT      30
#define CRITICAL_NU                     3
#define TRIGGER_TIME_LIMIT              15
#define FIRST_DIFFERENCE                ON

/* Trigger states */
#define CHANNEL_TRIGGER_DISABLED        0
#define CHANNEL_TRIGGER_ENABLED         1
#define CHANNEL_TRIGGERED               2
#define CHANNEL_TRIGGERED_EVENT         3

/* Event Constants */
#define MIN_EVENT_TIME                  30
#define MAX_EVENT_TIME                  60

/* Event Energy Constants */
#define CRITICAL_GAMMA                  2
#define EVENT_CONTINUATION_COUNT        30
#define CRITICAL_MU                     3

/* Structure defines */
#define critical_alpha                  trigsetting.const1
#define critical_beta                   trigsetting.const2
#define trigger_confirmation_count      trigsetting.threshold
#define critical_nu                     trigsetting.const3
#define sta_window                      trigsetting.const4
#define lta_window                      trigsetting.wav_inc
#define trigger_time_limit              trigsetting.aperture

#define min_event_time                  eventsetting.minduration
#define max_event_time                  eventsetting.maxduration
#define critical_gamma                  eventsetting.const1
#define critical_mu                     eventsetting.const2
#define event_continuation_count        eventsetting.threshold


/*************************************************************************
                            STRUCTURE DEFINITIONS
*************************************************************************/
typedef struct {
        FLAG first_difference;
        unsigned int lta_lower_bound;
        unsigned int sta_shift;
        unsigned int lta_shift;
        SUDS_STRUCTTAG structtag;
        SUDS_TRIGSETTING trigsetting;
} T_TRIGGER;

typedef struct {
        SUDS_STRUCTTAG structtag;
        SUDS_EVENTSETTING eventsetting;
} T_EVENT;


/*************************************************************************
                            EXTERNAL DECLARATIONS

These functions can be called from all modules that include this file.

*************************************************************************/
PUBLIC void t_display_event_params (FILE *);
PUBLIC void t_display_triggers (FILE *);
PUBLIC void t_display_trigger_params (FILE *);
PUBLIC FLAG t_event_done ();
PUBLIC FLAG t_event_detected();
PUBLIC unsigned int t_get_channel_trigger_state (unsigned int);
PUBLIC T_EVENT far * t_get_eventsetting ();
PUBLIC Q_CHANNEL far * t_get_head_trigger ();
PUBLIC double t_get_last_trigger ();
PUBLIC Q_CHANNEL far * t_get_next_trigger ();
PUBLIC FLAG t_get_trigger_status ();
PUBLIC double t_get_trigger_time (unsigned int);
PUBLIC T_TRIGGER far * t_get_trigsetting ();
PUBLIC void t_initialize ();
PUBLIC void t_initialize_params ();
PUBLIC void t_reset_event ();
PUBLIC void t_reset_trigger ();
PUBLIC void t_set_CriticalAlpha (int);
PUBLIC void t_set_CriticalBeta (int);
PUBLIC void t_set_CriticalGamma (int);
PUBLIC void t_set_CriticalMu (int);
PUBLIC void t_set_CriticalNu (int);
PUBLIC void t_set_EventContinueCount (int);
PUBLIC void t_set_event ();
PUBLIC void t_set_first_difference (FLAG);
PUBLIC void t_set_lta_lower_bound (unsigned int);
PUBLIC void t_set_LTAWindow (unsigned int);
PUBLIC void t_set_MaxEventTime (double);
PUBLIC void t_set_MinEventTime (double);
PUBLIC void t_set_netwname (char *);
PUBLIC void t_set_STAWindow (unsigned int);
PUBLIC void t_set_Trigger (unsigned int, FLAG);
PUBLIC void t_set_TriggerConfirmCount (int);
PUBLIC void t_set_TriggerTimeLimit (double);
PUBLIC void t_set_trigger_status (FLAG);
PUBLIC void t_toggle_trigger_status ();
PUBLIC FLAG t_trigger_check ();

#endif
```

```
/* FILE: mutils.c                                   (D. Tottingham  04/26/92)

This is a collection of C utility functions that are used by xdetect and
supporting modules.  All functions have been written and compiled medium
model.  The following functions are included:

u_build_date ()                 build a date: YRMNDY
u_build_date_fn ()              build a date filename: pathname\YRMNDY
u_build_timeline ()             convert abs. time to an ascii string
u_convert_time ()               convert a struct timeb into abs. time
u_get_diskspace ()              get the percentage of free disk space
u_gmtime ()                     convert time value into structure.
u_ispow2 ()                     determine whether number is a power of two
u_log2 ()                       compute the base-2 logarithm of given number
u_strcpy ()                     copies far string2 to far string1
u_strncmp ()                    compare first n character of strings
u_strncpy ()                    copies n characters of far string2 to far string1
u_strnlen ()                    get length of far string
u_timestamp ()                  get timestamp
u_tzset_GMT ()                  set timezone to GMT

EXTERNAL FUNCTIONS CALLED:

er_abort ()                     display an error message then quit

HISTORY:

V1.95   (09/29/90)  Added functionality to u_ispow2() and u_log2() to make sure
                    that they only operate on positive, non-zero numbers.
                                                                              */

/*****************************************************************************
                                INCLUDE FILES
*****************************************************************************/
#include <dos.h>
#include <math.h>
#include <malloc.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys\types.h>
#include <sys\timeb.h>
#include <time.h>

#include "mconst.h"
#include "merror.h"
#include "mutils.h"


/*****************************************************************************
                                GLOBALS
*****************************************************************************/
int daylight;
long timezone;
char *tzname[];


/*---------------------------------------------------------------------------*
 *                              u_build_date                                 *
 *---------------------------------------------------------------------------*/
/* Build a date: YRMNDY                                                      */

PUBLIC
long u_build_date (abs_time)
double abs_time;
{
   struct tm * local_tm;
   long date, ltime;

   ltime = (long) abs_time;
   local_tm = localtime (&ltime);

   date = (((long)local_tm->tm_year) * 10000) + ((local_tm->tm_mon + 1) * 100) +
          local_tm->tm_mday;

   return (date);
}

/*---------------------------------------------------------------------------*
 *                            u_build_date_fn                                *
 *---------------------------------------------------------------------------*/
/* Build a date filename: pathname\YRMNDY.                                   */

PUBLIC
int u_build_date_fn ( abs_time, path_name, file_name )
double abs_time;
char path_name[];
char file_name[];
{
   struct tm * local_tm;
   long ltime;
   int j;

   ltime = (long) abs_time;
   local_tm = localtime (&ltime);

   file_name[0] = '\0';

   j = sprintf (file_name, "%s%02d%02d%02d", path_name,
                local_tm->tm_year,
                local_tm->tm_mon + 1,
                local_tm->tm_mday);

   return (j);
}

/*---------------------------------------------------------------------------*
 *                            u_build_timeline                               *
 *---------------------------------------------------------------------------*/
/* Convert the absolute time given in time_buf to an ascii string.           */

PUBLIC
char * u_build_timeline (abs_time, style)
double abs_time;
unsigned int style;
{
   char * timeline, * time_str;
   long time;
   unsigned int millitm;

   time = (long) abs_time;
   millitm = (unsigned int) ((abs_time - ((double) time)) * 1000);

   time_str = ctime (&time);
   timeline = calloc (( strlen(time_str) + MAX_NUMSTR_LENGTH), sizeof(char));

   if (timeline) {
      switch (style) {
         case 0:
            sprintf (timeline, "%.19s.%03d %s", time_str, millitm, &time_str[20]);
            break;
         case 1:
            sprintf (timeline, "%.19s.%03d %3s", time_str, millitm, tzname[0]);
            break;
         case 2:
            sprintf (timeline, "%.7s%.4s %.8s.%03d", &time_str[4], &time_str[20],
                     &time_str[11], millitm);
            break;
      }
      return (timeline);
   } else er_abort (U_NO_STORAGE);
}
```

```
/*-------------------------------------------------------------------------*
 *                          u_convert_time                                 *
 *-------------------------------------------------------------------------*/
/* Convert struct timeb into an absolute time.                             */

PUBLIC
double u_convert_time (abs_timeb)
struct timeb abs_timeb;
{
   double abs_time;

   abs_time = (double) abs_timeb.time;
   abs_time += ((double) abs_timeb.millitm) / 1000.0;

   return (abs_time);
}

/*-------------------------------------------------------------------------*
 *                          u_get_diskspace                                *
 *-------------------------------------------------------------------------*/
/* Get the percentage of free disk space for this path.                    */

PUBLIC
long u_get_diskspace (path_name, drive_ptr)
char path_name[];
int * drive_ptr;
{
   static char * device_id[] = {NULL,"A:","B:","C:","D:","E:","F:","G:",NULL};
   struct diskfree_t diskspace;
   long percent_avail;
   int i;

   /* Look for drive letter in pathname */
   for (i = 1; device_id[i]; i++) {
      if (!strnicmp (device_id[i], path_name, 2))
         break;
   }

   /* If drive letter not found, get drive letter of cwd */
   if (!device_id[i]) _dos_getdrive (&i);

   /* Call _dos_getdiskfree to determine free disk space */
   if (!_dos_getdiskfree (i, &diskspace)) {
      percent_avail = ((long) diskspace.avail_clusters) * 100 /
                      diskspace.total_clusters;
      *drive_ptr = i - 1;
      return (percent_avail);
   }
   else return ( INVALID_DRIVE);
}

/*-------------------------------------------------------------------------*
 *                          u_gmtime                                       *
 *-------------------------------------------------------------------------*/
/* Convert time value into structure.                                      */

PUBLIC
void u_gmtime (time, tm_struct)
double time;
U_ELAPSED_TIME * tm_struct;
{
   unsigned long elapsed_time;

   elapsed_time = (unsigned long) time;

   tm_struct->millisec = (unsigned int) ((time - ((double) elapsed_time)) *
                         1000.0);

   tm_struct->days = elapsed_time / 86400;
   elapsed_time %= 86400;

   tm_struct->hours = elapsed_time / 3600;
   elapsed_time %= 3600;

   tm_struct->minutes = elapsed_time / 60;
   tm_struct->seconds = elapsed_time % 60;
}

/*-------------------------------------------------------------------------*
 *                          u_ispow2                                       *
 *-------------------------------------------------------------------------*/
/* Checks to see whether number is a power of two.                         */

PUBLIC
FLAG u_ispow2 (number)
unsigned long number;
{
   double powr;

   if (number == 0) return (FALSE);

   powr = u_log2 (number);
   if ( ceil(powr) > powr )
      return (FALSE);
   else
      return ( TRUE);
}

/*-------------------------------------------------------------------------*
 *                          u_log2                                         *
 *-------------------------------------------------------------------------*/
/* Compute the base-2 logarithm of given number.                           */

PUBLIC
double u_log2 (number)
unsigned long number;
{
   double ln_2, ln_d;

   if (number == 0) return (0.0);

   ln_d = log(((double) number));
   ln_2 = log(2.0);

   return (ln_d / ln_2);
}

/*-------------------------------------------------------------------------*
 *                          u_strcpy                                       *
 *-------------------------------------------------------------------------*/
/* Copies far string2 to far string1.                                      */

PUBLIC
char far * u_strcpy (string1, string2)
char far * string1;
char far * string2;
{
   int i;

   for (i = 0; i == 0 || string2[i-1] != '\0'; i++)
      string1[i] = string2[i];

   return (string2);
}

/*-------------------------------------------------------------------------*
 *                          u_strncmp                                      *
 *-------------------------------------------------------------------------*/
/* Compare first n character of strings.                                   */

PUBLIC
int u_strncmp (string1, string2, n)
char far * string1;
char * string2;
unsigned int n;
{
```

```
   int i, length;

   if (u_strnlen (string1, n) != (length = strlen (string2))) return (0);
   for (i = 0; i < n && i <= length; i++)
      if (string1[i] > string2[i]) return (1);
      else if (string1[i] < string2[i]) return (-1);

   return (0);
}

/*-----------------------------------------------------------------------*
 *                              u_strncpy                                *
 *-----------------------------------------------------------------------*/
/* Copies n characters of far string2 to far string1.                    */

PUBLIC
char far * u_strncpy (string1, string2, n)
char far * string1;
char far * string2;
unsigned int n;
{
   int i;

   for (i = 0; i < n && (!i || string2[i-1] != '\0'); i++)
      string1[i] = string2[i];

   return (string1);
}

/*-----------------------------------------------------------------------*
 *                              u_strnlen                                *
 *-----------------------------------------------------------------------*/
/* Get length of far string.                                             */

PUBLIC
unsigned int u_strnlen (string1, n)
char far * string1;
unsigned int n;
{
   int i;

   for (i = 0; i < n && string1[i] != '\0'; i++);

   return (i);
}

/*-----------------------------------------------------------------------*
 *                              u_timestamp                              *
 *-----------------------------------------------------------------------*/
/* Get timestamp.                                                        */

PUBLIC
double u_timestamp ()
{
   struct timeb current_timeb;

   ftime (&current_timeb);
   return ( u_convert_time (current_timeb));
}

/*-----------------------------------------------------------------------*
 *                              u_tzset_GMT                              *
 *-----------------------------------------------------------------------*/
/* Set timezone to GMT.                                                  */

PUBLIC
void u_tzset_GMT ()
{
   putenv ("TZ=GMT");
   tzset();
}
```

```
/* FILE: mutils.h                                  (D. Tottingham  04/26/92)

This is an include file of defines, data structure definitions and
external declarations that are common in the mutils module.

*/

#ifndef _MUTILS_
#define _MUTILS_

/*************************************************************************
                                INCLUDES
*************************************************************************/
#include "mconst.h"


/*************************************************************************
                                DEFINES
*************************************************************************/
#define INVALID_DRIVE           -1


/*************************************************************************
                        STRUCTURE DEFINITIONS


*************************************************************************/
typedef struct {
        unsigned int days;
        unsigned int hours;
        unsigned int minutes;
        unsigned int seconds;
        unsigned int millisec;
) U_ELAPSED_TIME;


/*************************************************************************
                            EXTERNAL DECLARATIONS

These functions can be called from all modules that include this file.

*************************************************************************/
PUBLIC long u_build_date (double);
PUBLIC int u_build_date_fn (double, char *, char *);
PUBLIC char * u_build_timeline (double, unsigned int);
PUBLIC double u_convert_time (struct timeb);
PUBLIC long u_get_diskspace (char *, int *);
PUBLIC void u_gmtime (double, U_ELAPSED_TIME *);
PUBLIC FLAG u_ispow2 (unsigned long);
PUBLIC double u_log2 (unsigned long);
PUBLIC int u_strncmp (char far *, char *, unsigned int);
PUBLIC char far * u_strcpy (char far *, char far *);
PUBLIC char far * u_strncpy (char far *, char far *, unsigned int);
PUBLIC unsigned int u_strnlen (char far *, unsigned int);
PUBLIC double u_timestamp ();
PUBLIC void u_tzset_GMT ();

#endif
```

```
/* FILE: mvideo.h                                (D. Tottingham 04/26/92)

This is an include file of the defines, data structure definitions and
external data declarations for the mscreen module.

*/


#ifndef _MVIDEO_
#define _MVIDEO_

/*************************************************************************
                                INCLUDES
*************************************************************************/
#include "mconst.h"

#if defined (EGA)
#include "ega.h"

#elif defined (VGA)
#include "vga.h"

#elif defined (MONO)
#include "mono.h"
#endif

#endif
```

```
/* FILE: powrstor.h                                (D. Tottingham  04/26/92)   */

/* ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ */
/*        PowerSTOR Header                                                    */
/*                                                                            */
/*          Copyright (C) 1989 by Acme Software.                              */
/*                             P.O. Box 40734                                 */
/*                             Portland, OR  97240                            */
/*                             Ph (503) 235-2816                              */
/*                                                                            */
/*          This software is intended for your use in                         */
/*          constructing programs and software systems.  You                  */
/*          may use the provided source and object files for                  */
/*          that purpose.  No royalties or fees of any sort                   */
/*          are charged for such use.  You may sell or other-                 */
/*          wise provide this software to others, only in                     */
/*          linked form, as part of a larger program or                       */
/*          software system.                                                  */
/*                                                                            */
/*          You may not sell or otherwise provide this                        */
/*          software to others, in any form which may be                      */
/*          used to construct programs or software systems.                   */
/*          This includes the original and any copies of                      */
/*          the source code as well as the object modules.                    */
/*          Distribution of any libraries which contain                       */
/*          PowerSTOR code is similarly prohibited.                           */
/*                                                                            */
/*          You may make copies of this software as you see                   */
/*          fit, provided such copies are for your own use                    */
/*          and will not be provided to any other person or                   */
/*          persons.  Any other distribution of this software                 */
/*          is forbidden as outlined above.                                   */
/*                                                                            */
/* -------------------------------------------------------------------------- */
/****************************************************************************/

#ifndef _POWRSTOR_
#define _POWRSTOR_

     /* ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ */
     /* STORstatus                                                             */
     /*                                                                        */
     /* A STORstatus is returned by all the PowerSTOR functions except         */
     /* QuitPwr and Sstatchk.  The STOstatus enumeration defines all the       */
     /* valid values that may be returned as a STORstatus                      */
     /*                                                                        */

enum STOstatus{ Okay, InvalInit, BadInitData, NotInitialized, NotEnoughMem,
                FileERROR, EXPerror, EXTerror, InvalidADR, NoMoreMem,
                NoSuchHeap, NoMoreHeaps
              };

typedef int STORstatus;

     /* ---------------------------------------------------------------------- */

/****************************************************************************/

     /* ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ */
     /* INITelement                                                            */
     /*                                                                        */
     /* The address of an array of INITelement is passed to the InitPwr        */
     /* function.  The contents of this array controls what memory devices,    */
     /* and how much of each kind can be available as PowerSTOR memory.        */
     /*                                                                        */
     /* The dvcType enumeration lists all the valid memory devices.  The       */
     /* memType variable in each array element must be assigned one of         */
     /* these dvcType values to tell PowerSTOR the device type.                */
     /*                                                                        */
     /* The NulMem type must be assigned to the last element in the            */
     /* initialization array.  This terminates the array, which can            */
     /* be any size up to 16 elements.                                         */
     /*                                                                        */

#define POWRSTOR_USAGE          2
#define SMALLEST_CHUNK          16384


enum dvcType( NulMem, ExpMem, ExtMem, DskFile );

union INITelement
{
    struct
    {
        int       memType;              /* Type identifier (DskFile)      */
        unsigned  MaxSize;              /* Maximum file size (in blocks)  */
        char      namestring[64];       /* Path and File name string      */

    } diskF;

    struct
    {
        int       memType;              /* Type identifier (ExpMem)       */
        unsigned  Rsrvd;                /* Reserved blocks                */

    } EXPmem;

    struct
    {
        int       memType;              /* Type identifier (ExtMem)       */
        int       LoLimit;              /* Low block number               */
        int       HiLimit;              /* High block number+1            */

    } EXTmem;

    struct
    {
        int       memType;              /* Type identifier (NulMem)       */
                                        /* This type of structure is the  */
                                        /* init array terminator          */
    } Nulmem;

};

     /* ---------------------------------------------------------------------- */

/****************************************************************************/

     /* ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ */
     /* STORheap                                                               */
     /*                                                                        */
     /* A STORheap structure identifies a particular STORheap.  It is          */
     /* assigned by the newSheap function when the STORheap is created.        */
     /* All functions that access a STORheap require this identifier.          */
     /*                                                                        */

struct STORheap
{
    unsigned BATptr;
    unsigned ID;
};

     /* ---------------------------------------------------------------------- */

/****************************************************************************/

     /* ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ */
     /* PowerSTOR functions                                                    */
     /*                                                                        */

   /* Initialize and activate PowerSTOR */
STORstatus InitPwr( unsigned          NeededSize,  /* in 16K byte blocks   */
                    union INITelement *initArray   /* initialization array */
                  );

   /* De-activate PowerSTOR */
```

```
void QuitPwr();

   /* Show the number of bytes of PowerSTOR memory currently in use */
STORstatus STORused( unsigned long  *bytesUsed );

   /* Show the number of bytes of PowerSTOR memory currently */
   /* available for allocation to STORheaps                  */
STORstatus STORavai( unsigned long  *bytesAvailable );

   /* Create a new STORheap */
/** dtotting: changed **/
STORstatus newSheap( struct STORheap far *heapID );

   /* Delete a STORheap */
STORstatus  delSheap( struct STORheap  heapID );

   /* Set a STORheap's size */
STORstatus  setSheap( struct STORheap  heapID,
                      unsigned long    newSize
                    );

   /* Allocate more PowerSTOR memory to a STORheap */
STORstatus  aloSheap( struct STORheap  heapID,
                      unsigned long    size,
                      unsigned long    *newSegBase
                    );

   /* Show the current size of a STORheap */
STORstatus  sizSheap( struct STORheap  heapID,
                      unsigned long  *size
                    );

   /* Move data from a STORheap into main memory */
/** dtotting: changed **/
STORstatus  getSTOR( struct STORheap  heapID,
                     unsigned long    fromSTOR,
                     void        far *toMemory,
                     unsigned         num_bytes
                   );

   /* Move data from main memory to a STORheap */
/** dtotting: changed **/
STORstatus  putSTOR( struct STORheap  heapID,
                     void         far *fromMemory,
                     unsigned long    toSTOR,
                     unsigned         num_bytes
                   );

   /* Print a message to the DOS standard error output if a STORstatus */
   /* is not Okay                                                      */
STORstatus  Sstatchk( STORstatus  stat );

    /* ------------------------------------------------------------------- */
/***************************************************************************/

#endif
```

```
/* FILE: suds.h                                (P. Ward & D. Tottingham 04/26/92)

SUDS Version 1.2

Some care has been taken to keep bytes aligned on 4 byte boundaries for
portabilty.  Individual fields within all structures have unique names for
compatability with the database system.  Expandability is accomplished by
adding new structures and by adding new fields to the end of these structures.
Such additions need to be coordinated among all users and should not be taken
lightly.

HISTORY:
   none
                                                                          */

#ifndef _SUDS_
#define _SUDS_

/************************************************************************
                              DEFINES
************************************************************************/
/* Define code for type of machine suds is being compiled on.
   "6" is 80x86 machine                                                   */
#define MACHINE         "6"

/* Define symbols for missing data. */
#define NODATA          "-32767."
#define NOTIME          "-2147472000" /* Dec 14, 1901 00:00 00 GMT, near largest long */
#define NOCHAR          " "
#define NOSTRG          "#"
#define NOLIST          "0"

/* Integer defines for standard variable types  */
#define CHR             1       /* char 1 byte                          */
#define MIN             2       /* char number 1 byte                   */
#define STR             3       /* string                               */
#define BTS             4       /* char 1 byte                          */
#define SHT             5       /* short 2 bytes                        */
#define LNG             6       /* long 4 bytes                         */
#define FLT             7       /* float 4 bytes                        */
#define DBL             8       /* double 8 bytes                       */
#define STI             9       /* struct stat_ident 8 bytes            */
#define STT             10      /* st_time 4 bytes                      */
#define MST             11      /* ms_time 8 bytes                      */
#define LLT             12      /* lon_lat 8 bytes                      */
#define CAL             13      /* struct calib 500 bytes               */
#define CPX             14      /* struct complex 16 bytes              */
#define BTW             15      /* unsigned short 2 bytes               */

/* Structure types or identifiers */
#define NO_STRUCT       0
#define ATODINFO        29
#define CALIB           23
#define CALIBRATION     9
#define COMMENT         20
#define COMPLEX         24
#define DESCRIPTRACE    7
#define DETECTOR        28
#define EQUIPMENT       4
#define ERROR           15
#define EVENT           12
#define EVENTSETTING    27
#define EV_DESCRIPT     13
#define FEATURE         10
#define FOCALMECH       16
#define LAYERS          19
#define LOCTRACE        8
#define MOMENT          17
#define MUXDATA         6
#define ORIGIN          14
#define PROFILE         21
#define RESIDUAL        11
#define SHOTGATHER      22
#define STAT_IDENT      1
#define STATIONCOMP     5
#define STRUCTTAG       2
#define TERMINATOR      3
#define TIMECORRECTION  30
#define TRIGGERS        25
#define TRIGSETTING     26
#define VELMODEL        18

#define TOTAL_STRUCTS   30


/* Standard variable types redefined for portability and clarity  */
typedef char            CHAR;   /* A single ascii character                 */
typedef char            MINI;   /* A 1 byte integer (0 to 255)              */
typedef char            STRING; /* A character string, null byte terminated */
typedef unsigned char   BITS8;  /* An 8 bit field                           */
typedef unsigned short  BITS16; /* A 16 bit field                           */
typedef short           SH_INT; /* A 16 bit signed integer                  */
typedef long            LG_INT; /* A 32 bit signed integer                  */
typedef float           FLOAT;  /* A 32 bit floating point number, IEEE     */
typedef double          DOUBLE; /* A 64 bit double precision number, IEEE   */
typedef struct {
   FLOAT   fx;
   FLOAT   fy;
} VECTOR;

typedef struct {
   FLOAT   cr;
   FLOAT   ci;
} COMPLEXX;

typedef struct {
   DOUBLE  dr;
   DOUBLE  di;
} D_COMPLEX;

typedef struct {
   FLOAT   xx;
   FLOAT   yy;
   FLOAT   xy;
} TENSOR;


typedef LG_INT  ST_TIME;     /*  stamp GMT time in seconds before or after Jan 1,
                                 1970, resolution is one second          */

typedef DOUBLE  MS_TIME;     /* GMT time in seconds before or after Jan 1,
                                1970, resolution finer than microseconds*/

typedef DOUBLE  LONLAT;      /* latitude or longitude in degrees, N and E positive*/

/************************************************************************
                          STRUCTURE DEFINITIONS

************************************************************************/

/* SUDS_FORM:  Information on each suds structure used for input from ascii,
               output to ascii, loading or unloading the database, etc.    */

typedef struct {
   LG_INT num;
   STRING *meaning;
} SUDS_CODES;

typedef struct {
   LG_INT  fstype;          /* Structure types or identifiers              */
   STRING  *fname;          /* Name used to identify structure item        */
   LG_INT  ftype;           /* Type of variable in field                   */
   LG_INT  flength;         /* Length of variable in the field             */
```

```
   LG_INT  offset;          /* Offset of variable pointer from beginning of
                               structure measured in bytes, where first byte
                               of structure=0.  This field is potential
                               machine dependent and ideally should be
                               determined by a program like the ddlp
                               (Data Description Language Processor) of
                               db_vista.  As long as structures are carefully
                               defined to align 4 bytes boundaries, we can get
                               away without writing such a complex program for
                               most machines.                              */
   SUDS_CODES *codelist;    /* If value indexes a codes list, this is list */
   STRING  *initval;        /* Value to initialize structure               */
   STRING  *fformat;        /* Printf type format to read + write field    */
   LG_INT  nextfstype;      /* If structure, this is fstype                */
   LG_INT  frecord;         /* database record for this field              */
   LG_INT  ffield;          /* dstabase field for this field               */
   LG_INT  form_row;        /* row of field in forms editor                */
   LG_INT  form_col;        /* beginning column of field in forms editor   */
   STRING  *allowchar;      /* allowed characters on input                 */
} SUDS_FORM;


/* SUDS_STRUCTTAG:  Structure to identify structures when archived together  */

#define ST_MAGIC   'S'      /* magic character for sync in structtag        */

typedef struct {
   CHAR sync;               /* The letter S. If not present, error exists.
                               Use to unscramble damaged files or tapes.    */
   CHAR machine;            /* code for machine writing binary file for use
                               in identifying byte order and encoding.      */
   SH_INT id_struct;        /* structure identifier: numbers defined above  */
   LG_INT len_struct;       /* structure length in bytes for fast reading
                               and to identify new versions of the structure */
   LG_INT len_data;         /* length of data following structure in bytes  */
} SUDS_STRUCTTAG;


/* SUDS_STATIDENT:  Station identification.                                  */

typedef struct {            /* station component identifier                 */
   STRING  network[4];      /* network name                                 */
   STRING  st_name[5];      /* name of station where equipment is located   */
   CHAR    component;       /* component v,n,e                              */
   SH_INT  inst_type;       /* instrument type                              */
} SUDS_STATIDENT;


/* SUDS_ATODINFO: Information on the A to D converter                        */

typedef struct {
   SH_INT  base_address;    /* base I/O address of this device              */
   SH_INT  device_id;       /* device identifier                            */
   BITS16  device_flags;    /* device flags                                 */
   SH_INT  extended_bufs;   /* number of extended buffers used              */
   SH_INT  external_mux;    /* AtoD external mux control word               */
   CHAR    timing_source;   /* AtoD timing source: i=internal, e=external   */
   CHAR    trigger_source;  /* AtoD trigger source: i=internal, e=external  */
} SUDS_ATODINFO;


/* SUDS_CALIBRATION: Calibration information for a station component         */

#define NOCALPTS 30

typedef struct {
   COMPLEXX   pole;         /* pole                                         */
   COMPLEXX   zero;         /* zero                                         */
} SUDS_CALIBR;

typedef struct {
   SUDS_STATIDENT ca_name;  /* station component identification             */
   FLOAT msxgain;           /* maximum gain of calibration curve            */
   FLOAT normaliz;          /* factor to multiply standard calib by to make
                               peak at given frequency=1                    */
   SUDS_CALIBR cal[NOCALPTS]; /* calibration info                           */
   ST_TIME   begint;        /* time this calibration becomes effective      */
   ST_TIME   endt;          /* time this calibration is no longer effective */
} SUDS_CALIBRATION;


/* SUDS_COMMENT:  Comment tag to be followed by the bytes of comment         */

typedef struct {
   SH_INT   refer;          /* structure identifier comment refers to       */
   SH_INT   item;           /* item in structure comment refers to          */
   SH_INT   length;         /* number of bytes in comment                   */
   SH_INT   unused;
} SUDS_COMMENT;


/* SUDS_DESCRIPTRACE:  Descriptive information about a seismic trace.
                       Normally followed by waveform.                      */

typedef struct {
   SUDS_STATIDENT dt_name;  /* station component identification             */
   SH_INT  localtime;       /* minutes to add to GMT to get local time      */
   CHAR    datatype;        /* s=short(16 bit), r=12 bit data, 4 lsb time,
                               l=long(32 bit), f=float,
                               d=double, c=complex, v=vector, t=tensor      */
   CHAR    descriptor;      /* g=good, t=telemetry noise, c=calibration, etc*/
   MS_TIME begintime;       /* time of first data sample                    */
   SH_INT  digi_by;         /* agency code who digitized record; 0=original */
   SH_INT  processed;       /* processing done on this waveform             */
   LG_INT  length;          /* number of samples in trace                   */
   FLOAT   rate;            /* samples per second                           */
   FLOAT   mindata;         /* minimum value of data (type s,l,f only)      */
   FLOAT   maxdata;         /* maximum value of data (type s,l,f only)      */
   FLOAT   avenoise;        /* average value of first 200 samples (type s,l,f only)
   LG_INT  numclip;         /* number of clipped datapoints                 */
   FLOAT   rate_correct;    /* rate correction to be added to rate          */
   MS_TIME time_correct;    /* time correction to be added to begintime     */
} SUDS_DESCRIPTRACE;


/* SUDS_DETECTOR:  Information on detector program being used                */

typedef struct {
   CHAR    dalgorithm;      /* triggering algorithm: x=xdetect, m=mdetect
                               e=eqdetect                                   */
   CHAR    event_type;      /* c=calibration, e=earthquake, E=explosion,
                               f=free run, n=noise, etc.                    */
   CHAR    net_node_id[10]; /* network node identification                  */
   FLOAT   versionnum;      /* software version number                      */
   LG_INT  event_number;    /* unique event number assigned locally.        */
   LG_INT  spareL;          /* spare                                        */
} SUDS_DETECTOR;


/* SUDS_EQUIPMENT:  Equipment making up a station/component. Primarily used for
                    maintenance but may be referenced by researcher. One or more
                    structures exist for each piece of equipment making up a
                    station/component.                                       */

typedef struct {
   SUDS_STATIDENT this;     /* identifier of this piece of equipment        */
   SUDS_STATIDENT previous; /* next piece of equipment toward sensor        */
   SUDS_STATIDENT next;     /* next piece of equipment toward recorder      */
   STRING  serial[8];       /* serial number                                */
   SH_INT  model;           /* model such as L4, HS10, etc.                 */
   SH_INT  knob1;           /* knob setting or series resistor value of Lpad*/
   SH_INT  knob2;           /* knob setting or shunt  resistor value of Lpad*/
   SH_INT  reason;          /* reason change was made                       */
   FLOAT   frequency;       /* sensor corner frequency, vco freq, transmitter
                               frequency, etc.                              */
   ST_TIME effective;       /* date/time these values became effective      */
```

```
} SUDS_EQUIPMENT;


/* SUDS_ERROR:  Error matrix                                          */

typedef struct {
   FLOAT   covarr[10];       /* covariance matrix                     */
} SUDS_ERROR;


/* SUDS_EVENT:  General information about an event.                   */

typedef struct {
   SH_INT  authority;        /* organization processing the data      */
   LG_INT  number;           /* unique event number assigned by organization */
   SH_INT  felt;             /* number of felt reports                */
   CHAR    mintensity;       /* maximum Modified Mercali Intensity    */
   CHAR    ev_type;          /* e=earthquake, E=explosion, n=nuclear,
                                i=icequake, b=b_type, n=net, r=regional,
                                t=teleseism, c=calibration, n=noise   */
   CHAR    tectonism;        /* observed u=uplift, s=subsidence, S=strikeslip
                                faulting, N=normal faulting, T=thrust */
   CHAR    waterwave;        /* seiche, tsunami, etc.                 */
   CHAR    mechanism;        /* t=thrust, s=strike-slip, n=normal, e=explosive*/
   CHAR    medium;           /* medium containing explosion or event  */
   FLOAT   size;             /* magnitude or pounds TNT for explosions */
} SUDS_EVENT;


/* SUDS_EVENTSETTING:  Settings for earthquake trigger system         */

typedef struct {
   STRING  netwname[4];      /* network name                          */
   MS_TIME beginttime;       /* time these values in effect           */
   SH_INT  const1;           /* trigger constant 1                    */
   SH_INT  const2;           /* trigger constant 2                    */
   SH_INT  threshold;        /* trigger threshold                     */
   SH_INT  const3;           /* trigger constant 3                    */
   FLOAT   minduration;      /* minimum duration for event            */
   FLOAT   maxduration;      /* maximum duration for event            */
   CHAR    algorithm;        /* triggering algorithm: x=xdetect, m=mdetect
                                e=eqdetect                            */
   CHAR    spareK;           /* spare                                 */
   SH_INT  spareI;           /* spare                                 */
} SUDS_EVENTSETTING;


/* SUDS_EVDESCR:  Descriptive information about an event typically used for
                  major, destructive earthquakes. This structure is typically
                  associated with EVENT structure.                    */

typedef struct {
   STRING  eqname[20];  /* Popular name used to refer to this earthquake*/
   STRING  country[16]; /* country of earthquake                       */
   STRING  state[16];   /* state, province or other political subdivision*/
   SH_INT  localtime;   /* hours to add to GMT to get local time       */
   SH_INT  spareB;
} SUDS_EVDESCR;


/* SUDS_FEATURE:  Observed phase arrival time, amplitude, and period. */

typedef struct {
   SUDS_STATIDENT fe_name;   /* station component identification      */
   SH_INT  obs_phase;        /* observed phase code                   */
   CHAR    onset;            /* wave onset descriptor, i or e         */
   CHAR    direction;        /* first motion: U,D,+,-                 */
   SH_INT  sig_noise;        /* ratio ampl. of first peak or trough to noise */
   CHAR    data_source;      /* i=interactive,a=automatic,r=rtp, or user code*/
   CHAR    tim_qual;         /* timing quality given by analyst: 0-4, etc.
                                n=ignore timing                       */
   CHAR    amp_qual;         /* amplitude quality given by analyst: 0-4, etc.
                                n=ignor amplitude information         */
   CHAR    ampunits;         /* units amplitude measured in: d=digital counts
                                m=mm on develocorder, etc.            */
   SH_INT  gain_range;       /* 1 or gain multiplier if gain range in effect */
   MS_TIME time;             /* phase time, x value where pick was made */
   FLOAT   amplitude;        /* peak-to-peak amplitude of phase       */
   FLOAT   period;           /* period of waveform measured           */
   ST_TIME time_of_pick;     /* time this pick was made               */
   SH_INT  pick_authority;   /* organization processing the data      */
   SH_INT  pick_reader;      /* person making this pick               */
} SUDS_FEATURE;


/* SUDS_FOCALMECH:  General information about a focal mechanism.      */

typedef struct {
   FLOAT   astrike;          /* strike of plane a                     */
   FLOAT   adip;             /* dip of plane a                        */
   FLOAT   arake;            /* rake of plane a                       */
   FLOAT   bstrike;          /* strike of plane b                     */
   FLOAT   bdip;             /* dip of plane b                        */
   FLOAT   brake;            /* rake of plane b                       */
   CHAR    prefplane;        /* preferred plane a or b or blank       */
   CHAR    spareC[3];
} SUDS_FOCALMECH;


/* SUDS_LAYERS:  Velocity layers.                                     */

typedef struct {
   FLOAT   thickness;        /* thickness in kilometers               */
   FLOAT   pveltop;          /* p velocity at top of layer            */
   FLOAT   pvelbase;         /* p velocity at base of layer           */
   FLOAT   sveltop;          /* s velocity at top of layer            */
   FLOAT   svelbase;         /* s velocity at base of layer           */
   SH_INT  function;         /* velocity function in layer: 0=constant,
                                1=linear, 2=exponential, etc.         */
   SH_INT  spareF;
} SUDS_LAYERS;


/* SUDS_LOCTRACE:  Location of trace.                                 */

typedef struct {
   SUDS_STATIDENT lt_name;   /* station component identification      */
   STRING  *fileloc;         /* pointer to pathname in file system    */
   STRING  *tapeloc;         /* pointer to name of tape or offline storage */
   LG_INT  beginloc;         /* bytes from begining of file to trace  */
} SUDS_LOCTRACE;


/* SUDS_MOMENT:  Moment tensor information.                           */

typedef struct {
   BITS8   datatypes;        /* sum of: 1=polarities, 2=amplitudes,
                                4=waveforms, etc.                     */
   CHAR    constraints;      /* solution constrained: d=deviatoric,
                                c=double couple                       */
   CHAR    spareD[2];
   FLOAT   sc_moment;        /* scalar moment                         */
   FLOAT   norm_ten[6];      /* normalized moment tensor              */
} SUDS_MOMENT;


/* SUDS_MUXDATA:  Header for multiplexed data                         */

typedef struct {
   STRING  netname[4];       /* network name                          */
   MS_TIME begintime;        /* time of first data sample             */
   SH_INT  loctime;          /* minutes to add to GMT to get local time */
   SH_INT  numchans;         /* number of channels: if !=1 then multiplexed */
   FLOAT   dig_rate;         /* samples per second                    */
   CHAR    typedata;         /* s=short(16 bit), r=12 bit data, 4 lsb time,
                                l=long(32 bit), f=float,
```

```
                                d=double, c=complex, v=vector, t=tensor      */
   CHAR     descript;          /* g=good, t=telemetry noise, c=calibration, etc*/
   SH_INT   dc_offset;         /* dc offset                                    */
   LG_INT   numsamps;          /* number of sample sweeps. Typically not known
                                  when header is written, but can be added later*/
   LG_INT   blocksize;         /* number of demultiplexed samples per channel if
                                  data is partially demultiplexed, otherwise=0 */
} SUDS_MUXDATA;


/* SUDS_ORIGIN: Information about a specific solution for a given event      */

typedef struct {
   LG_INT   number;            /* unique event number assigned by organization */
   SH_INT   authority;         /* organization processing the data             */
   CHAR     version;           /* version of solution within organization      */
   CHAR     or_status;         /* processing status: f=final, a=automatic, etc */
   CHAR     preferred;         /* p=preferred location                         */
   CHAR     program;           /* name of processing program  h=hypo71,
                                  l=hypolayer, i=isc, c=centroid, etc.         */
   CHAR     depcontrl;         /* depth control: f=fixed, etc.                 */
   CHAR     convergence;       /* hypocentral convergence character            */
   LG_INT   region;            /* geographic region code assigned locally      */
   MS_TIME  orgtime;           /* origin time                                  */
   LONLAT   or_lat;            /* latitude, north is plus                      */
   LONLAT   or_long;           /* longitude, east is plus                      */
   FLOAT    depth;             /* depth in kilometers, + down                  */
   FLOAT    err_horiz;         /* horizontal error in km                       */
   FLOAT    err_depth;         /* vertical error in km                         */
   FLOAT    res_rms;           /* rms of residuals                             */
   STRING   crustmodel[6];     /* code for model used in this location         */
   SH_INT   gap;               /* azimuthal gap in degrees                     */
   FLOAT    nearstat;          /* distance in km to nearest station            */
   SH_INT   num_stats;         /* number of stations reporting phases          */
   SH_INT   rep_p;             /* number of p phases reported                  */
   SH_INT   used_p;            /* number of p times used in the solution       */
   SH_INT   rep_s;             /* number of s phases reported                  */
   SH_INT   used_s;            /* number of s times used in the solution       */
   SH_INT   mag_type;          /* magnitude type: coda,tau,xmag ml,mb,ms,mw    */
   SH_INT   rep_m;             /* number of magnitude readings reported        */
   SH_INT   used_m;            /* number of magnitude readings used            */
   FLOAT    magnitude;         /* magnitude value                              */
   FLOAT    weight;            /* average magnitude weight                     */
   FLOAT    mag_rms;           /* rms of magnitude                             */
   ST_TIME  effective;         /* time this solution was calculated            */
} SUDS_ORIGIN;


/* SUDS_PROFILE:  Grouping of shotgathers by profile.                        */

typedef struct {
   int junk1;
   /* What is your suggestion? */
} SUDS_PROFILE;


/* SUDS_RESIDUAL:  Calculated residuals for arrival times, magnitudes, etc.  */

typedef struct {
   LG_INT   event_num;         /* unique event number                          */
   SUDS_STATIDENT re_name;     /* station component identification             */
   SH_INT   set_phase;         /* phase code set for this solution             */
   CHAR     set_tim_qual;      /* timing quality assigned for this soln: 0-4   */
   CHAR     set_amp_qual;      /* amplitude quality assigned for this soln: 0-4*/
   FLOAT    residual;          /* traveltime residual or phase magnitude       */
   FLOAT    weight_used;       /* weight used in this solution                 */
   FLOAT    delay;             /* delay time or station correction used        */
   FLOAT    azimuth;           /* azimuth event to station, 0 north            */
   FLOAT    distance;          /* distance in km event to ststion              */
   FLOAT    emergence;         /* angle of emergence from source, 0=down,180=up*/
} SUDS_RESIDUAL;


/* SUDS_SHOTGATHER:  Grouping of waveforms by source event                   */

typedef struct {
   int junk2;
   /* What is your suggestion? */
} SUDS_SHOTGATHER;


/* SUDS_STATIONCOMP:  Generic station component information                  */

typedef struct {
   SUDS_STATIDENT sc_name;     /* station component identification             */
   SH_INT   azim;              /* component azimuth clockwise from north,
                                     0 for vertical                           */
   SH_INT   incid;             /* component angle of incidence from vertical
                                     0 is vertical, 90 is horizontal          */
   LONLAT   st_lat;            /* latitude, north is plus                      */
   LONLAT   st_long;           /* longitude, east is plus                      */
   FLOAT    elev;              /* elevation in meters                          */
   CHAR     enclosure;         /* d=dam, n=nuclear power plant, v=underground
                                     vault, b=buried, s=on surface, etc.      */
   CHAR     annotation;        /* annotated comment code                       */
   CHAR     recorder;          /* type device data recorded on                 */
   CHAR     rockclass;         /* i=igneous, m=metamorphic, s=sedimentary      */
   SH_INT   rocktype;          /* code for type of rock                        */
   CHAR     sitecondition;     /* p=permafrost, etc.                           */
   CHAR     sensor_type;       /* sensor type: d=displacement, v=velocity,
                                  a=acceleration, t=time code                 */
   CHAR     data_type;         /* s=short(16 bit), r=12 bit data, 4 lsb time,
                                  l=long(32 bit), f=float,
                                  d=double, c=complex, v=vector, t=tensor      */
   CHAR     data_units;        /* data units: d=digital counts, v=millivolts,
                                  n=nanometers (/sec or /sec/sec)              */
   CHAR     polarity;          /* n=normal, r=reversed                         */
   CHAR     st_status;         /* d=dead, g=good                               */
   FLOAT    max_gain;          /* maximum gain of the amplifier                */
   FLOAT    clip_value;        /* +-value of data where clipping begins        */
   FLOAT    con_mvolts;        /* conversion factor to millivolts: mv per counts
                                     0 means not defined or not appropriate
                                  max_ground_motion=digital_sample*con_mvolts*
                                  max_gain                                     */
   SH_INT   channel;           /* a2d channel number                           */
   SH_INT   atod_gain;         /* gain of analog to digital converter          */
   ST_TIME  effective;         /* date/time these values became effective      */
   FLOAT    clock_correct;     /* clock correction in seconds.                 */
   FLOAT    station_delay;     /* seismological station delay.                 */
} SUDS_STATIONCOMP;


/* SUDS_TERMINATOR:  Structure to end a sequence of related structures when
                     loaded in a serial file or on a serial device.          */

typedef struct {
   SH_INT   structid;          /* id for structure at beginning of this sequence*/
   SH_INT   spareA;
} SUDS_TERMINATOR;


/* SUDS_TIMECORRECTION:  Time correction information.                        */

typedef struct {
   SUDS_STATIDENT tm_name;     /* time trace station id used to determine
                                  correction.                                  */
   MS_TIME  time_correct;      /* time correction to be added to begintime     */
   FLOAT    rate_correct;      /* rate correction to be added to rate          */
   CHAR     sync_code;         /* synchronization code as follows:
                                     0 = total failure,   1 = 1 second synch,
                                     2 = 10 second synch, 3 = minute synch,
                                  4, 5 = successful decode.                    */
   CHAR     program;           /* program used to decode time:
                                     e = irige, c = irigc                      */
   ST_TIME  effective_time;    /* time this correction was calculated          */
   SH_INT   spareM;
```

```
} SUDS_TIMECORRECTION;


/* SUDS_TRIGGERS:  Earthquake detector trigger statistics                    */

typedef struct {
   SUDS_STATIDENT tr_name;   /* station component identification            */
   SH_INT  sta;              /* short term average                          */
   SH_INT  lta;              /* long term average; pre_lta for xdetect      */
   SH_INT  abs_sta;          /* short term absolute average                 */
   SH_INT  abs_lta;          /* long term absolute average                  */
   SH_INT  trig_value;       /* value of trigger level (eta)                */
   SH_INT  num_triggers;     /* number of times triggered during this event */
   MS_TIME trig_time;        /* time of first trigger                       */
} SUDS_TRIGGERS;


/* SUDS_TRIGSETTING:  Settings for earthquake trigger system                 */

typedef struct {
   STRING  netwname[4];      /* network name                                */
   MS_TIME beginttime;       /* time these values in effect                 */
   SH_INT  const1;           /* trigger constant 1                          */
   SH_INT  const2;           /* trigger constant 2                          */
   SH_INT  threshold;        /* trigger threshold                           */
   SH_INT  const3;           /* trigger constant 3                          */
   SH_INT  const4;           /* trigger constant 4                          */
   SH_INT  wav_inc;          /* weighted average increment                  */
   FLOAT   sweep;            /* trigger sweep time in seconds               */
   FLOAT   aperture;         /* seconds for coincident station triggers     */
   CHAR    algorithm;        /* triggering algorithm: x=xdetect, m=mdetect
                                e=eqdetect                                  */
   CHAR    spareJ;           /* spare                                       */
   SH_INT  spareI;           /* spare                                       */
} SUDS_TRIGSETTING;


/* SUDS_VELMODEL:  Velocity model                                            */

typedef struct {
   STRING  netname[4];       /* network name                                */
   STRING  modelname[6];     /* model name                                  */
   CHAR    spareE;
   CHAR    modeltype;        /* p=profile A to B, a=area within corners A B */
   LONLAT  latA;             /* latitude of point A, north is plus          */
   LONLAT  longA;            /* longitude of point A, east is plus          */
   LONLAT  latB;             /* latitude of point B, north is plus          */
   LONLAT  longB;            /* longitude of point B, east is plus          */
   ST_TIME time_effective;   /* time this model was created                 */
} SUDS_VELMODEL;

#endif
```

```
/* FILE: timer.c                                    (D. Tottingham  04/26/92)

This is a collection of C functions that manage a stopwatch.  All functions
have been written and compiled medium model.  The following functions are
included:

diff_time ()                    compute difference between start and stop times
set_starttime ()                set start time
set_stoptime ()                 set stop time

EXTERNAL FUNCTIONS CALLED:
   none

HISTORY:
   none

*/


/*************************************************************************
                            INCLUDE FILES

*************************************************************************/
#include <sys\types.h>
#include <sys\timeb.h>

#include "mconst.h"
#include "timer.h"


/*************************************************************************
                                GLOBALS

*************************************************************************/
PRIVATE struct timeb start[N_TIMERS], stop[N_TIMERS];


/*-----------------------------------------------------------------------*
 *                            set_starttime                              *
 *-----------------------------------------------------------------------*/
/* Set start time.                                                       */

PUBLIC
void set_starttime (timer_id)
unsigned int timer_id;
{
   ftime (&(start[timer_id]));
}

/*-----------------------------------------------------------------------*
 *                            set_stoptime                               *
 *-----------------------------------------------------------------------*/
/* Set stop time.                                                        */

PUBLIC
void set_stoptime (timer_id)
unsigned int timer_id;
{
   ftime (&(stop[timer_id]));
}

/*-----------------------------------------------------------------------*
 *                            diff_time                                  *
 *-----------------------------------------------------------------------*/
/* Compute the delta time between the start and stop times.              */

PUBLIC
double diff_time (timer_id)
unsigned int timer_id;
{
   double time1, time2;

   time2 = stop[timer_id].time + ((double) stop[timer_id].millitm)/1000.0;
   time1 = start[timer_id].time + ((double) start[timer_id].millitm)/1000.0;

   return (time2 - time1);
}
```

```
/* FILE: timer.h                                    (D. Tottingham  04/26/92)

This is an include file of the defines, data structure definitions and
external data declarations for using the timer module.

*/

#ifndef _TIMER_
#define _TIMER_

/*************************************************************************
                                INCLUDES

*************************************************************************/
#include "mconst.h"


/*************************************************************************
                                DEFINES

*************************************************************************/
#define TIMER_0         0
#define TIMER_1         1
#define TIMER_2         2
#define TIMER_3         3

#define N_TIMERS        4


/*************************************************************************
                          EXTERNAL DECLARATIONS

These functions can be called from all modules that include this file.

*************************************************************************/
PUBLIC void set_starttime (unsigned int);
PUBLIC void set_stoptime (unsigned int);
PUBLIC double diff_time (unsigned int);

#endif
```

```
/* FILE: xdetect.c                                    (D. Tottingham  04/26/92)

This is the driver for xdetect, the multi-channel event detection and data
collection program.  The program has been written and compiled medium model.

EXTERNAL FUNCTIONS CALLED:

dm_initialize ()                    initialize the queues
dm_initialize_parsms ()             initialize parameters
dm_initialize_time ()               initialize the time
dm_reset ()                         reset this module and release all storage
dsp_spectral_detection ()           return flag for spectral_detection
dsp_spectral_detection_done ()      return flag for spectral recording done
dsp_get_dsp_status ()               get dsp_enabled flag
dsp_domain_check ()                 analyze the current buffer for spectral lines
dsp_initialize ()                   initialize DSP package
dsp_initialize_params ()            initialize key parameters
dt_dump_configuration ()            dump the DT2821 configuration
dt_get_channel_size ()              get channel block size
dt_get_number_of_ext_buffers ()     get number of external buffers
dt_get_scan_count ()                get scan count
dt_initialize_ADC ()                initialize the DT2821
dt_initialize_mux ()                initialize multiplexer
dt_initialize_params()              initialize parameters
dt_start_ADC ()                     start continuous data acquisition
dt_stop_ADC ()                      stop continuous data acquisition
e_initialize ()                     initialize event parameters
e_get_bell_status ()                get bell_enabled
e_ring_bell ()                      ring event alert bell if enabled
e_toggle_bell_status ()             toggle bell_enabled flag
er_abort ()                         display an error message then quit
f_check_pathname ()                 check pathname for validity
f_initialize_index ()               initialize the index file
f_initialize_params ()              initialize key parameters
fr_initialize ()                    initialize free run parameters
fr_toggle_freerun ()                toggle free run event
h_initialize_params ()              initialize a screen header
h_set_uptime ()                     update uptime.
h_toggle_status ()                  toggle a status bit
l_display_location ()               write location to stream
l_get_location_status ()            get location_enabled
l_initialize ()                     initialize location module
l_locate_event ()                   locate the event
l_reset_location ()                 reset the location flag
l_toggle_location_status ()         toggle location_enabled flag
o_check_pathname ()                 check log pathname for validity
o_initialize_params ()              initialize log pathname
o_open_logfile ()                   open a log file for output
o_write_logfile ()                  write string to log file stream
p_initialize ()                     initialize parser
p_parse_commands ()                 parse commands in input stream
pk_initialize ()                    initialize pick module
pk_pick_arrivals ()                 pick arrivals using trigger information
pk_pick_magnitudes ()               pick magnitudes
pk_reset_picks ()                   reset pick queues
r_check_time ()                     compare time with reboot time
r_get_reboot_status ()              get reboot_enabled
r_get_reboot_time ()                get reboot time
r_initialize_reboot ()              initialize reboot structure
r_toggle_reboot_status ()           toggle reboot_enabled flag
s_initialize ()                     initialize the screen module
s_initialize_screen ()              initialize the graphics screen
s_display_ids ()                    display station id, channel, and trigger status
s_display_traces ()                 display normalized traces on the screen
s_display_triggers ()               display triggers on the screen
s_reset_screen ()                   reset the monochrome screen to text
s_select_display ()                 select the display mode
s_set_channelrange ()               set the channel range in the block display
st_initialize_params ()             initialize station queue
st_initialize_stations ()           initialize station list
t_display_triggers ()               write event information to stream
t_get_trigger_status ()             get trigger_enabled
t_initialize ()                     initialize the trigger algorithm
t_initialize_params ()              initialize the trigger parameters
t_reset_trigger ()                  reset trigger variables
t_toggle_trigger_status ()          toggle trigger_enabled flag
t_trigger_check ()                  look for new triggers
u_build_timeline ()                 convert abs. time to an ascii string
u_ispow2 ()                         determine whether i_num is a power of two
u_tzset_GMT ()                      set timezone to GMT
u_timestamp ()                      get timestamp

HISTORY:

(04/26/92) o Verified support for PC-NFS and Lantastic NOS.
V2.04
           o Fixed bug in f_write_buffers().  Events collected right after
             midnight would clobber the 00 event from the previous day.  The
             reason for this was that the time used to create wvm filename
             was not consistent with time used to update the index file.
             WVM filenames are now based on the most recent buffer's timestamp.


/*************************************************************************
                              INCLUDE FILES
*************************************************************************/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <atldefs.h>

#include "mconst.h"
#include "mdemux.h"
#include "mdsp.h"
#include "mdt28xx.h"
#include "merror.h"
#include "mevent.h"
#include "mfile.h"
#include "mfreerun.h"
#include "mlocate.h"
#include "mlog.h"
#include "mparse.h"
#include "mpick.h"
#include "mreboot.h"
#include "mscreen.h"
#include "mscrnhdr.h"
#include "mstation.h"
#include "mtrigger.h"
#include "mutils.h"
#include "xdetect.h"
#include "timer.h"


/*************************************************************************
                              GLOBAL DATA
*************************************************************************/
FLAG Debug_enabled;                             /* debug on/off */
PRIVATE FLAG quit_enabled = FALSE;              /* quit flag */
PRIVATE unsigned int display_type;

PRIVATE char out_str[85];
double min_time, max_time, avg_E;


/*************************************************************************
                          FUNCTION DECLARATIONS
*************************************************************************/
PRIVATE void quit ();



/*-----------------------------------------------------------------------
```

```
 *                      initialize_command_line                          *
 *-------------------------------------------------------------------------*/
/* Initialize command line parameters.                                    */

PRIVATE
void initialize_command_line ()
{
   Debug_enabled = DEBUG_ENABLED;
   dt_initialize_mux ();
}

/*-------------------------------------------------------------------------*
 *                        initialize_defaults                              *
 *-------------------------------------------------------------------------*/
/* Initialize default parameters.                                         */

PRIVATE
void initialize_defaults ()
{
   u_tzset_GMT ();
   st_initialize_params();
   dm_initialize_params();
   dsp_initialize_params();
   dt_initialize_params();
   e_initialize ();
   f_initialize_params();
   fr_initialize();
   h_initialize_params();
   l_initialize();
   o_initialize_params();
   p_initialize();
   pk_initialize();
   r_initialize_reboot();
   s_initialize();
   t_initialize_params ();
}


/*-------------------------------------------------------------------------*
 *                       process_command_line                              *
 *-------------------------------------------------------------------------*/
/* Process the command line parameters.                                   */

PRIVATE
void process_command_line (filename, argc, argv)
char filename[];
int argc;
char * argv[];
{
   FLAG found_filename;

   /* Search for command line options and the filename */
   found_filename = FALSE;
   while (--argc) {
      switch (argv[argc][0]) {
         case '/':
            switch (toupper(argv[argc][1])) {
               case 'D':
                  Debug_enabled = TRUE;
                  break;

               /* room for other slash options */
            }
            break;
         default:
            strcpy (filename, argv[argc]);
            found_filename = TRUE;
      }
   }

   if (! found_filename) {
      printf ("Enter control file [<CR> = %s]  >> ", INPUT_FILENAME);
      gets (filename);
      printf ("\n");
      if (filename[0] == '\0')
         strcpy (filename, INPUT_FILENAME);
   }

}

/*-------------------------------------------------------------------------*
 *                        process_input_file                               *
 *-------------------------------------------------------------------------*/
/* Process the input file.                                                */

PRIVATE
void process_input_file (filename)
char filename[];
{
   FILE * inp_stream;

   o_write_logfile ("<<<<<<<<<<<<<<<<<<<<<<   N E W   S E S S I O N   >>>>>>>>>>>>>>>>
   sprintf (out_str, "XDETECT Version %5.2f, Last Revision %s\n", VERSIONNUM, REVISI
   o_write_logfile (out_str);
   sprintf (out_str, "INPUT FILE:  %s\n\n", filename);
   o_write_logfile (out_str);

   /* If the input file exists, process the contents */
   if (inp_stream = fopen (filename, "r")) {
      p_parse_commands (inp_stream);
      fclose (inp_stream);
   } else er_abort (X_INVALID_INPUTFILE);
}

/*-------------------------------------------------------------------------*
 *                           display_setup                                 *
 *-------------------------------------------------------------------------*/
/* Display setup information.                                             */

PRIVATE
void display_setup ()
{
   /* Print out some key flags and values */
   printf ("\nBell      DSP       Location  Trigger   Reboot    Reboot time   Clk so
   o_write_logfile ("Bell      DSP       Location  Trigger   Reboot    Reboot time
   sprintf (out_str, "%-10s%-10s%-10s%-10s%-10s%-8s      %-14s\n\n",
            ((e_get_bell_status())       ? "ENABLED"  : "DISABLED"),
            ((dsp_get_dsp_status())      ? "ENABLED"  : "DISABLED"),
            ((l_get_location_status())   ? "ENABLED"  : "DISABLED"),
            ((t_get_trigger_status())    ? "ENABLED"  : "DISABLED"),
            ((r_get_reboot_status())     ? "ENABLED"  : "DISABLED"),
            r_get_reboot_time(),
            ((dt_get_clock_source() == 'i') ? "INTERNAL" : "EXTERNAL"));

   printf (out_str);
   o_write_logfile (out_str);
   sprintf (out_str, "Number of external buffers = %d\n\n", dt_get_number_of_ext_buf
   printf (out_str);
   o_write_logfile (out_str);
}

/*-------------------------------------------------------------------------*
 *                          check_pathnames                                *
 *-------------------------------------------------------------------------*/
/* Check pathnames.                                                       */

PRIVATE
void check_pathnames ()
{
   o_check_pathname();
   f_check_pathname();
}

/*-------------------------------------------------------------------------*
 *                          initialize_system                              *
 *-------------------------------------------------------------------------*/
```

```
/* Initialize system parameters.                                        */

PRIVATE
void initialize_system ()
{
   /* Prime the trigger algorithm */
   t_initialize ();

   /* Initialize the station list */
   st_initialize_stations ();

   /* Initialize the DSP */
   dsp_initialize();

   /* Initialize the ADC */
   dt_initialize_ADC ();

   /* Allocate internal and external multiplexed buffers */
   dt_initialize_buffers();

   /* Initialize the demux queue.  Setup the PowerSTOR */
   dm_initialize ();

   /* Start data acquisition */
   dt_start_ADC ();

   /* Display the current setup */
   display_setup ();

   /* Initialize the screen and screen header */
   s_initialize_screen ();
   dm_initialize_time ();

   /* Initialize the index file */
   f_initialize_index ();

   /* Initialize the ids */
   display_type = HOME;
   s_select_display (HOME);
   s_display_ids ();
}

/*----------------------------------------------------------------------*
 *                      post_trigger_analysis                           *
 *----------------------------------------------------------------------*/

PRIVATE
void post_trigger_analysis ()
{
   FILE * log_stream;

   /* Display triggers on screen. */
   if (Debug_enabled)
      t_display_triggers (stdout);
   if (LOG_FILE) {
      log_stream = o_open_logfile ();
      t_display_triggers (log_stream);
      fclose (log_stream);
   }
}

/*----------------------------------------------------------------------*
 *                            process_buffer                            *
 *----------------------------------------------------------------------*/
/* Main processing loop.                                                */

PRIVATE
void process_buffer ()
{
   FILE * log_stream;
   double dt;

   /* Do we have a buffer to process */
   if (! dt_buffer_full())
      return;

   /* Main processing loop */
   set_starttime(TIMER_0);
   dm_get_new_buffers();
   s_display_traces (0, 0);
   h_set_uptime ();

   if (dsp_spectral_detection()) {
      if (dsp_spectral_detection_done()) {
        f_write_buffers (END_FILE, F_CALIBRATION);
        h_toggle_status (H_RECORD_CALIBRATION);
      }
      else f_write_buffers (CONTINUE_FILE, F_CALIBRATION);

   } else if (t_event_detected()) {
      if (t_event_done ()) {
         /* Event is over.  Locate the event */
         if (l_locate_event()) {
            if (LOG_FILE) {
               log_stream = o_open_logfile ();
               l_display_location (log_stream);
               fclose (log_stream);
            }
            if (Debug_enabled) l_display_location (stdout);
         }
         f_write_buffers (END_FILE, F_EARTHQUAKE);
         l_reset_location ();
         pk_reset_picks ();
         t_reset_trigger ();
         h_toggle_status (H_RECORD_EVENT);
         if (Debug_enabled)
            printf ("Data Acquisition continued...\n");

      } else {
         f_write_buffers (CONTINUE_FILE, F_EARTHQUAKE);
         pk_pick_magnitudes ();
      }

   } else if (r_check_time()) {

   } else if (fr_continue_freerun()) {

   } else if (quit_enabled) {
       quit();

   } else if (dsp_domain_check()) {
      if (Debug_enabled)
         printf ("Calibration found and being saved...\n");
      h_toggle_status (H_RECORD_CALIBRATION);
      f_write_buffers (START_FILE, F_CALIBRATION);

   } else if (t_trigger_check ()) {
      e_ring_bell ();

      /* Setup to collect eq data */
      h_toggle_status (H_RECORD_EVENT);

      /* Pick the p-arrivals and magnitudes */
      pk_pick_arrivals ();
      pk_pick_magnitudes ();

      /* Do some post-trigger processing at this point */
      post_trigger_analysis ();

      /* Write pre-trigger data to disk */
      f_write_buffers (START_FILE, F_EARTHQUAKE);
   }

   set_stoptime(TIMER_0);
   dt = diff_time(TIMER_0);
   if (dt < min_time) min_time = dt;
```

```
   if (dt > max_time) max_time = dt;
   if (t_event_detected() /* || free_run_flag */)
      avg_E += (dt - avg_E) / dt_get_number_of_ext_buffers();
}

/*-------------------------------------------------------------------------*
 *                      display_copyright_notice                           *
 *-------------------------------------------------------------------------*/
/* Display a copyright notice.                                             */

PRIVATE
void display_copyright_notice()
{
   printf ("*****************************************************************************\n\
   printf ("*                              XDETECT V%4.2f                               *\n\
   printf ("*                                                                           *\n\
   printf ("*  This binary contains proprietary and copyrighted object code as          *\n\
   printf ("*  follows:                                                                 *\n\
   printf ("*                                                                           *\n\
   printf ("*               MONOCHROME, EGA, and VGA Graphics Libraries                 *\n\
   printf ("*                Copyright (C) 1988, Symmetric Research                     *\n\
   printf ("*                                                                           *\n\
   printf ("*                        PowerSTOR For C Library                            *\n\
   printf ("*                   Copyright (C) 1989, Acme Software                       *\n\
   printf ("*                                                                           *\n\
   printf ("*                              ATLAB                                        *\n\
   printf ("*                   Copyright (C) 1989, Data Translation                    *\n\
   printf ("*                                                                           *\n\
   printf ("*  Duplication and/or distribution of this binary outside of                *\n\
   printf ("*  the U.S. Geological Survey is prohibited without expressly written       *\n\
   printf ("*  permission from the aforementioned parties.                              *\n\
   printf ("*                                                                           *\n\
   printf ("*****************************************************************************\n");
}

/*-------------------------------------------------------------------------*
 *                              quit                                       *
 *-------------------------------------------------------------------------*/
/* Reset screen to text.  Quit xdetect.                                    */

PRIVATE
void quit ()
{
   double avg_time, max_sps;

   dt_stop_ADC ();
   s_reset_screen ();
   dm_reset();

   printf ("\n\nMinimum time                 = %lf sec.\n", min_time);
   printf ("Maximum time                 = %lf sec.\n", max_time);
   if (avg_E) {
      max_sps = dt_get_channel_size() / avg_E;
      printf ("Average event time           = %lf sec.\n", avg_E);
      printf ("Maximum digitization rate    = %lf sps.\n", max_sps);
   }

   o_write_logfile ("..... End of session .....\n");

   exit ( 1);
}

/*-------------------------------------------------------------------------*
 *                              m a i n                                    *
 *-------------------------------------------------------------------------*/
main (argc, argv)
int argc;
char * argv[];
{
   FILE * log_stream;
   char * timeline;
   char filename[MAX_FILENAME_LENGTH];
   int c;

s_reset_screen ();

display_copyright_notice ();

/* Opening banner */
printf ("\n%s (%s)    Tottingham\n", REVISION_NAME, REVISION_DATE);
printf ("Multi-Channel Event Detection and Collection Program\n\n");

initialize_command_line ();

process_command_line (filename, argc, argv);

initialize_defaults ();

process_input_file (filename);

check_pathnames ();

initialize_system ();

/* Display the current unit configuration */
if (Debug_enabled)
   dt_dump_configuration (stdout);
if (LOG_FILE) {
   log_stream = o_open_logfile();
   dt_dump_configuration (log_stream);
   fclose (log_stream);
}

min_time = 10000;  /* just a big number */
max_time = avg_E = 0;

while (1) {
   if (c = keystat ()) {
      switch (c) {
         case CNTRL_B:
            e_toggle_bell_status ();

            if (Debug_enabled)
               printf ("Bell %s \n",
                     ((e_get_bell_status()) ? "ENABLED" : "DISABLED"));
            break;
         case CNTRL_F:
            fr_toggle_freerun();
            break;
         case CNTRL_L:
            l_toggle_location_status ();
            if (Debug_enabled)
               printf ("Location %s \n",
                     ((l_get_location_status()) ? "ENABLED" : "DISABLED"));
            break;
         case CNTRL_HOME:
            if (display_type != HOME) {
               s_select_display (HOME);
               s_display_ids ();
               s_display_traces (0, 0);
               s_display_triggers ();
               display_type = HOME;
            }
            break;
         case CNTRL_PGDOWN:
            if (display_type == BLOCK &&
                !s_set_channelrange (MAX_BLOCK, MAX_BLOCK))
               continue;
            else if (display_type != BLOCK) {
               s_select_display (BLOCK);
               display_type = BLOCK;
               s_set_channelrange (0, MAX_BLOCK);
            }
            s_display_ids ();
            s_display_traces (0, 0);
            s_display_triggers ();
```

```
            break;
         case CNTRL_PGUP:
            if (display_type == BLOCK &&
                !s_set_channelrange (-MAX_BLOCK, MAX_BLOCK))
               continue;
            else if (display_type != BLOCK) {
               s_select_display (BLOCK);
               display_type = BLOCK;
               s_set_channelrange (0, MAX_BLOCK);
            }
            s_display_ids ();
            s_display_traces (0, 0);
            s_display_triggers ();
            break;
         case CNTRL_Q:
            quit_enabled = TRUE;
            break;
         case CNTRL_R:
            r_toggle_reboot_status ();

            if (Debug_enabled)
               printf ("Reboot %s \n",
                       ((r_get_reboot_status()) ? "ENABLED" : "DISABLED"));
            break;
         case CNTRL_T:
            t_toggle_trigger_status ();

            h_toggle_status (H_AUTOTRIGGER_ENABLED);
            if (Debug_enabled)
               printf ("Autotriggering %s \n",
                       ((t_get_trigger_status()) ? "ENABLED" : "DISABLED"));
            break;
         case DOWN_ARROW:
            s_display_traces (0, -1);
            break;
         case F1:
            s_view_help ();
            while (! keystat())
               process_buffer ();
            s_clear ();
            s_select_screen (SCREEN_1);
            h_update ();
            s_display_ids ();
            s_display_traces (0, 0);
            s_display_triggers ();
            if (display_type == BLOCK)
               s_display_bar ();
            break;
         case UP_ARROW:
            s_display_traces (0, 1);
            break;
         default:
            break;
      }
   }
   process_buffer ();
  }
}
```

```
/* FILE: xdetect.h                                (D. Tottingham  04/26/92)

This is an include file of the defines, data structure definitions and
external data declarations used by the xdetect program.

*/

#ifndef _XDETECT_
#define _XDETECT_

/*************************************************************************
                                INCLUDES

*************************************************************************/
#include "mconst.h"


/*************************************************************************
                                DEFINES

*************************************************************************/
#define REVISION_DATE           "04/26/92"
#define REVISION_NAME           "XDETECT204"
#define VERSIONNUM              2.04

/* Input file */
#define INPUT_FILENAME          "XDETECT.INP"

/* Enable Flags */
#define DEBUG_ENABLED           FALSE

/* Keyboard Constants */
#define CNTRL_B                 0x3002
#define CNTRL_F                 0x2106
#define CNTRL_HOME              0x7700
#define CNTRL_L                 0x260c
#define CNTRL_PGDOWN            0x7600
#define CNTRL_PGUP              0x8400
#define CNTRL_Q                 0x1011
#define CNTRL_R                 0x1312
#define CNTRL_T                 0x1414
#define F1                      0x3b00
#define DOWN_ARROW              0x5000
#define UP_ARROW                0x4800


/*************************************************************************
                                GLOBALS

*************************************************************************/
PUBLIC FLAG Debug_enabled;

#endif
```

File: readme (8/16/92)

I. Title Page.

U. S. DEPARTMENT OF THE INTERIOR

U. S. GEOLOGICAL SURVEY

DESCRIPTIONS OF SEISMIC ARRAY COMPONENTS:

PART 2. SOFTWARE MODULES FOR DATA ACQUISITION/PROCESSING

Compiled by

W. H. K. Lee

MS 977, 345 Middlefield Road

Menlo Park, CA 94025

Open-File Report 92-597-B

August, 1992

II. Disclaimer.

This report is preliminary and has not been reviewed for conformity with U. S. Geological Survey editorial standards. Any use of trade, firm, or product names is for descriptive purposes only and does not imply endorsement by the U. S. Government.

Although these programs have been used by the U.S. Geological Survey, no warranty, expressed or implied, is made by the USGS as to the accuracy and functioning of the programs and related program material, nor shall the fact of distribution constitute any such warranty, and no responsibility is assumed by the USGS in connection therewith.

III. Date of Latest Revision.

August 16, 1992.

IV. Text.

INTRODUCTION

In the summer of 1990, funding was available to design and implement two portable seismic arrays for the volcano program. The approach was based on Lee et al. (1989). Several contracts were awarded to commercial companies to design and implement various components needed to build the portable arrays. The purpose of this report to present the software modules for data acquisition and processing -- SUDSPICK, SUDSPLOT, SUDSPROC/SUDSMAN, SUDSSQZ, PC-QMAP, AND XDETECT -- in detail as submitted by the contractors. Source code on PC-DOS/MS-DOS diskette for this report is presented in U. S. Geological Survey Open-File Report 92-597-B.

SUDSPICK

SUDSPICK is a computer program for automatic picking of the first P-arrival from waveform files created by the XDETECT realtime data acquisition program.

SUDSPLOT

SUDSPLOT is a computer program for plotting seismic waveform files created by the XDETECT realtime data acquisition program.

SUDSPROC/SUDSMAN

SUDSPROC/SUDSMAN are two computer programs for managing the data processing tasks of the waveform files created by the XDETECT realtime data acquisition program.

SUDSSQZ

SUDSSQZ is a computer program to "squeeze" out insignificant data in a waveform file created by the XDETECT realtime data acquisition program.

PC-QMAP

PC-QMAP consists of two computer programs to plot earthquake hypocenter data on a map. QMAPHP is for using the HP LaserJet printers, and QMAPPS is for using a Postscript laser printer.

XDETECT

XDETECT (version 2.04) is a computer program for realtime data acquisition and processing. It is an updated version of the XDETECT program released previously in Tottingham and Lee (1989).

REFERENCES

Lee, W. H. K., D. M. Tottingham, and J. O. Ellis (1989). Design and implementation of a PC-based seismic data acquisition, processing, and analysis system, IASPEI Software Library, 1, 21-46.

Tottingham, D. M., and W. H. K. Lee (1989). XDETECT: A fast seismic data acquisition and processing program, U.S. Geol. Surv. Open-File Report 89-205.

V. Diskette Contents.

This diskette contains eight directories:

(1) SUDSPICK -- source code for the SUDSPICK program written by Small Systems Support.

(2) SUDSPLOT -- source code for the SUDSPLOT program written by Small Systems Support.

(3) SUDSPROC -- source code for the SUDSPROC program written by Small Systems Support.

(4) SUDSMAN -- source code for the SUDSMAN program written by Small Systems Support.

(5) SUDSSQZ -- source code for the SUDSSQZ program written by Small Systems Support.

(6) QMAPHP -- source code for the QMAP program for HP LaserJet printer by Small Systems Support.

(7) QMAPPS -- source code for the QMAP program for a Postscript laser printer by Small Systems Support.

(8) XDETECT -- source code for the XDETECT program (version 2.04) by Tottco Consulting Group.

Open-File Report 92-597A and -B. Description of Seismic Array Components: Part 2. Software Modules for Data Acquisition/Processing. 1992. 313 p. and one 3.5-in. diskette.

This report describes software modules for data acquisition and processing – – SUDSPICK, SUDSPLOT, SUDSPROC/SUDSMAN, SUDSSQZ, PC-QMAP, AND XDETECT – – as submitted by the contractors. Open-File Report 92-597A (313 p.) contains the software documentation (including listings of source code). Open-File Report 92-597B is a 3.5-in. diskette containing the source code for the above mentioned software modules.

Requirements for part B: IBM 386 or 486 PC or compatible; minimum 1 MB RAM; math coprocessor; VGA graphics board and monitor; minimum 40 MB hard disk; 3.5-inch floppy disk drive; PC or MS DOS 4.01 or later; Microsoft C, Macro Assembler, and Fortran Compilers; and plotting library modules by Small Systems Support and Symmetric Research.