9. Appendix C: Script *cutpaste* and Examples of Its Use

The following listing is the cut-and-paste event-extraction script "*cutpaste*", used to generate event files from continuous-record files. A copy of the *cutpaste* script is in anonymous *ftp* on computer "andreas.wr.usgs.gov", in file "~ftp/pub/outgoing/evans/event_extraction/cutpaste/cutpaste". This Open-File Report serves to release them to the public, subject to the limitations cited on the title page. Following the *cutpaste* listing is a short example of a script that calls *cutpaste* for the desired events. Such calling scripts form a sufficient record of extraction activities and comments (e.g., Table 3 is derived from our own *cutpaste*-calling scripts).

Listing of *cutpaste* Script

```
#!/bin/sh
#
# Merge file for STS-2, SV/H-1, or CMG-3ESP data.
#
# Usage:
# cutpaste ( 1 2 3 | 4 5 6 ) dddd ( sts2 | svh1 | 3esp ) ssss YY JJJ hh mm ss rrr "blah blah"
# argc:          1 2 3         4              5              6   7  8   9 10 11  12     13
#
# where "1 2 3" or "4 5 6" are the triplet of channels to convert,
# "dddd" is the 4-digit serial number of the DAS, "sts2", "svh1", or
# "3esp" are the respective instrument type, "ssss" is the station name,
# "YY JJJ hh mm ss" is the start time of the file, and "rrr" is the
# length of record to produce (seconds), and "blah blah" is a comment.
#
# Produces cut and pasted SEG-Y files (one per channel), named
# "YYMMDD.hhmmss.<sts2|svh1|3esp>.ssss.<1|2|3|4|5|6>" and a single
# cut and pasted ah file (containing three channels), named
# "YYMMDD.hhmmss.<sts2|svh1|3esp>.ssss.ah".
#
# Bugs:  does no checking, just hammers away.  Assumes you are asking for
# something reasonable and are in the right directory.
#
# SEG-Y filename format:  R<Julian day>.<stream number>/hh.mm.ss.dddd.c
#
# Changed output filename format (reversed station and instrument fields)
# on 01 Sept 93 to make segy2ah pick up the correct station name (which
# it takes from the second-to-last "."-separated field).  No change to
# input format (still jug then site).
#
dest=/passcal10/data/SFT/extracts     # Destination directory
#
prog=$0
V=$1
shift ; N=$1
shift ; E=$1
shift ; DAS=$1
shift ; jug=$1
shift ; site=$1
shift ; YY=$1
shift ; JJJ=$1
shift ; hh=$1
shift ; mm=$1
shift ; ss=$1
shift ; len=$1
shift ; comment=$1
```

```
#
echo "-------------'pwd'-------------"
echo "$prog   $V $N $E   $DAS $jug $site   $YY:$JJJ:$hh:$mm:$ss $len
#
if [ -f 'ls *.$DAS.[$V$N$E] | head -1' ]; then
  echo "Processing..."
  segymerge -c *.$DAS.$V -s $YY:$JJJ:$hh:$mm:$ss -l $len -t 1 > $YY$JJJ.$hh$mm$ss.$jug.$site.$V
  segymerge -c *.$DAS.$N -s $YY:$JJJ:$hh:$mm:$ss -l $len -t 1 > $YY$JJJ.$hh$mm$ss.$jug.$site.$N
  segymerge -c *.$DAS.$E -s $YY:$JJJ:$hh:$mm:$ss -l $len -t 1 > $YY$JJJ.$hh$mm$ss.$jug.$site.$E
#
  if [ -f 'ls $YY$JJJ.$hh$mm$ss.$jug.$site.[$V$N$E] | head -1' ]; then
    segy2ah $YY$JJJ.$hh$mm$ss.$jug.$site.[$V$N$E]
    cat $YY$JJJ.$hh$mm$ss.$jug.$site.[$V$N$E].ah > $YY$JJJ.$hh$mm$ss.$jug.$site.ah
    /bin/rm $YY$JJJ.$hh$mm$ss.$jug.$site.[$V$N$E].ah
#
#   Should discard 0-length files here...
    echo "Moving to $dest;"
    ls -la $YY$JJJ.$hh$mm$ss.$jug.$site.[$V$N$E] $YY$JJJ.$hh$mm$ss.$jug.$site.ah
    mv -i $YY$JJJ.$hh$mm$ss.$jug.$site.[$V$N$E] $YY$JJJ.$hh$mm$ss.$jug.$site.ah $dest
  fi
fi
```

Example of *cutpaste* Usage

```
#!/bin/csh
# Run a series of the cutpaste script, starting from directory above
# those (e.g., R216.01) created by ref2segy.
#
#########################################################################
# L=local, R=regional, T=teleseism, (L/R at about S-P = 10 s;
# R/T at about distance 3-6 degrees, based on frequency content).
#
# L/R generated from CalNet listings via shell "seltran", which
# calls "qfetch" and "sum2cut" (Appendix D).
# R/T generated from USGS QED (NEIS) system lists, via program "dz"
# (Appendix E).
#########################################################################
#
# Start 216:20:13:47
#
# 04 August 93
cd R216.01
#
cutpaste    1 2 3    7064 sts2 sft5    93 216 20 20 07    099  "v sm CalNet L/R, m2.7 d1.57 h4.9"
#
# 05 August 93
cd ../R217.01
#
cutpaste    1 2 3    7064 sts2 sft5    93 217 10 19 31    120  "sm CalNet L/R, m2.8 d2.00 h9.1"
cutpaste    1 2 3    7064 sts2 sft5    93 217 18 43 58    094  "sm CalNet L/R, m1.9 d0.75 h0.9"
#
# 06 August 93
cd ../R218.01
#
```

```
cutpaste    1 2 3    7064 sts2 sft5    93 218 00 31 45    150  "CalNet L/R, m3.1 d2.06 h15.4"
# Unidentified surface wave -03:34 (similar to offshore N. CA events of late)
cutpaste    1 2 3    7064 sts2 sft5    93 218 11 01 42    098  "sm CalNet L/R, m2.6 d1.45 h8.4"
cutpaste    1 2 3    7064 sts2 sft5    93 218 23 34 31    093  "CalNet L/R, m2.6 d0.54 h7.2"
#
# SPANS DAY BREAK (002320 070*20):
cutpaste    1 2 3    7064 sts2 sft5    93 219 00 12 21   6100  "nice T, @NORTHEAST OF TAIWAN"
#
# 07 August 93
cd ../R219.01
#
cutpaste    1 2 3    7064 sts2 sft5    93 219 06 16 00   1560  "T, OFF COAST OF JALISCO, MEXICO"
cutpaste    1 2 3    7064 sts2 sft5    93 219 13 43 30    120  "CalNet L/R, m3.5 d1.39 h18.4"
cutpaste    1 2 3    7064 sts2 sft5    93 219 18 03 00  15420  "T, @SOUTH OF FIJI ISLANDS"
#
# 08 August 93
cd ../R220.01
#
# Ignore "T, @MINAHASSA PENINSULA, SULAWESI m5.4/4.9 d112 h62; weak spectral-only surface wave"
cutpaste    1 2 3    7064 sts2 sft5    93 220 08 45 55  23650  "T, @SOUTH OF MARIANA ISLANDS (GUAM)"
cutpaste    1 2 3    7064 sts2 sft5    93 220 20 14 40   4340  "T, @MARIANA ISLANDS"
#
# 09 August 93
cd ../R221.01
#
cutpaste    1 2 3    7064 sts2 sft5    93 221 02 19 15    195  "CalNet L/R, m3.0 d2.07 h15.4"
cutpaste    1 2 3    7064 sts2 sft5    93 221 02 49 00    900  "Unidentified surface wave"
cutpaste    1 2 3    7064 sts2 sft5    93 221 05 32 45    150  "CalNet L/R, m3.0 d2.46 h2.5"
cutpaste    1 2 3    7064 sts2 sft5    93 221 09 26 42   4160  "T, @MARIANA ISLANDS m5.3/5.5 d84 h60"
cutpaste    1 2 3    7064 sts2 sft5    93 221 11 50 00  12300  "T, @HINDU KUSH REGION"
cutpaste    1 2 3    7064 sts2 sft5    93 221 21 40 18    094  "CalNet L/R, m2.1 d0.69 h5.4"
cutpaste    1 2 3    7064 sts2 sft5    93 221 23 08 41    099  "v sm CalNet L/R, m2.1 d1.53 h12.9"
#
# 10 August 93
cd ../R222.01
#
cutpaste    1 2 3    7064 sts2 sft5    93 222 01 04 57  15603  "T, @OFF W. COAST OF S. ISLAND, N.Z"
cutpaste    1 2 3    7064 sts2 sft5    93 222 05 52 56    101  "CalNet L/R, m2.6 d1.98 h0.1"
cutpaste    1 2 3    7064 sts2 sft5    93 222 09 04 50  12970  "T, @MARIANA ISLANDS"
#
# Stop 222:18:41:27
```

10. Appendix D: Script *seltran* and Program *sum2cut* Source Code

The script *seltran* runs USGS program *qfetch*, which finds the most current CalNet local-earthquake information for a specified date range. This result is piped to USGS program *qselect* to extract two distance-magnitude sets. The selected event summary lines are finally translated by program *sum2cut*, listed below, into the form used by shell *cutpaste* to extract events from the continuous data stream.

Both the source code and the script are available *via* anonymous *ftp* from "andreas.wr.usgs.gov" in directory "~ftp/pub/outgoing/evans/event_extraction". This Open-File Report serves to release them to the public, subject to the limitations cited on the title page.

Script *seltran*

```
#!/bin/csh
# Shell to gather events for extraction from SFT continuous data.
#
# Run two annuli about SFT5, taking all events of ML>=2.0 out to
# 200 km, and ML>=3.5 from 200 to 400 km.  Then run sum2cut on
# each file to convert formats and set time windows.
#
# USAGE:  seltran date1 date2 > output_file
#                    $1    $2
# Format of dates is that of qfetch:  [CC]YYMMDDhhmmss[+-offset]
#
# John R. Evans, USGS, Menlo Park, 09/93.
#
rsh -n andreas "qfetch -time $1 $2 | \
   qselect -magnitude 2.0 10.0 -delta 37.505053 -121.328514 0.0 200.0" > \
   seltran.temp1.xxx
#
rsh -n andreas "qfetch -time $1 $2 | \
   qselect -magnitude 3.5 10.0 -delta 37.505053 -121.328514 200.01 400.0" >> \
   seltran.temp1.xxx
#
sort seltran.temp1.xxx > seltran.temp2.xxx
/bin/rm seltran.temp1.xxx
#
sum2cut < seltran.temp2.xxx
/bin/rm seltran.temp2.xxx
```

Source Code of *sum2cut* Follows:

```
/*
 * SUM2CUT (SUMmary card to CUTpaste formatter)
 *
 * USAGE:  seltran < summary_cards > output_file
 *
 * BUGS:
 *
 * Update CENTURY appropriately.
 *
 * Station coordinates list hardwired.  Change STA structure, and
 * NSTA appropriately.
 *
 */

#include <stdio.h>
#include <math.h>
#include <string.h>
#include <local/local.h>
#include <local/stdtyp.h>
#include <local/mathconst.h>
#include <local/date_time.h>

#define MAX(a,b) (((a)>(b)) ? (a) : (b))        /* Larger value */

#define MAXBUF          40      /* decode.h buffer size         */
#include <local/decode.h>

#define STR_LEN         256     /* Maximum scratch-string length */

#define CENTURY         1900
/* Coordinates of possible station(s) from
   which to calculate delta and azimuth */
#define SFT2NAM         "sft2"
#define SFT2LAT         37.153042
#define SFT2LON         -121.812232
#define SFT2ELEV        518

#define SFT4NAM         "sft4"
#define SFT4LAT         37.38883
#define SFT4LON         -121.49450
#define SFT4ELEV        660

#define SFT5NAM         "sft5"
#define SFT5LAT         37.505053
#define SFT5LON         -121.328514
#define SFT5ELEV        312

#define NSTA            3

typedef struct {                /* Station information          */
        char    code[5];        /* Name                         */
        double  lat;            /* Location (+N, degrees)       */
        double  lon;            /* (+E, degrees)                */
        double  elev;           /* meters                       */
} STA;

typedef struct {                /* Station information          */
        TIME    ot;             /* Origin time                  */
        double  lat;            /* Epicenter (+N, degrees)      */
        double  lon;            /* (+E, degrees)                */
} EVENT;

#define strneq !strncmp

        void    datime();       /* -lq (YMDhms <- epochal)      */
        void    delaz();        /* -lq (geocen.c)               */
        void    error();        /* -lq (error.c)                */
        void    exit();         /* UNIX C library               */
        bool    isleap();       /* -lq                          */
        void    read_ev();      /* Defined below                */
        void    refpt();        /* -lq (geocen.c)               */
        TIME    timvar();       /* -lq (YMDhms -> epochal)      */

main(argc, argv)
        int     argc;
        char    **argv;
{
        char    in_str[STR_LEN]; /* Input buffer                */

        int     styr, stmo, stday, sthr, stmn;
        double  stsec;
        double  delta, az0, az1;

        int     yr, mo, day, hr, mn, jday;
        double  sec, lat, lon;
        double  depth, mag;

        double  Ptime;
```

```
        int     seci, dur;
        TIME    start_t;

        int     ii;             /* Dummy                        */

        STA     stations[10];   /* Station list                 */

        /* Build station list the dumb way */

        (void)strncpy(stations[0].code, SFT2NAM, 5);
        stations[0].lat = SFT2LAT;
        stations[0].lon = SFT2LON;
        stations[0].elev = SFT2ELEV;

        (void)strncpy(stations[1].code, SFT4NAM, 5);
        stations[1].lat = SFT4LAT;
        stations[1].lon = SFT4LON;
        stations[1].elev = SFT4ELEV;

        (void)strncpy(stations[2].code, SFT5NAM, 5);
        stations[2].lat = SFT5LAT;
        stations[2].lon = SFT5LON;
        stations[2].elev = SFT5ELEV;


        if (argc > 1)
                error ("Usage: %s < summary_cards > output_file", argv[0]);

        /* Main loop--read summary card, output NSTA cutpaste lines */
        while(fgets(in_str, sizeof(in_str), stdin) != NULL) {
                read_ev(in_str, &yr, &mo, &day, &hr, &mn, &sec,
                        &lat, &lon, &depth, &mag);

                /* Loop over stations for output */
                for (ii = 0 ; ii < NSTA ; ii++) {

#define                         /* Infer record window */
#define                         PAD_BEG 30.0    /* Pad record window at front   */
#define                         PAD_END 60.0    /* Pad record window at rear    */
#define                         DEG2KM  111.2
#define                         P_VEL   5.0     /* Nominal P velocity (km/s)    */
#define                         LR_VEL  28.16   /* Nominal surface-wave velocity
                                                   (s/degree)                   */

                        refpt((stations[ii].lat * RAD), (stations[ii].lon * RAD));
                        delaz(lat * RAD, lon * RAD, &delta, &az0, &az1);
                        delta *= DEG;
                        az0 *= DEG;
                        az1 *= DEG;

                        /* Start at -P time - PAD_BEG */
                        Ptime = delta * DEG2KM / P_VEL;
                        start_t = timvar(CENTURY + yr, mo, day, hr, mn, sec,
                                GREGORIAN);
                        start_t += Ptime - PAD_BEG;
                        (void)datime(start_t,
                                &styr, &stmo, &stday, &sthr, &stmn, &stsec,
                                GREGORIAN);
                        jday = yrday(stmo, stday, isleap(styr, GREGORIAN));
                        seci = stsec;

                        /* End at nominal surface-wave time + PAD_END */
                        dur = (delta * LR_VEL - Ptime) + PAD_BEG + PAD_END;

                        /* Output */
                        (void)printf(
                                "cutpaste    1 2 3    dddd jjjj %s    ",
                                stations[ii].code);
                        (void)printf(
                                "%2d %03d %02d %02d %02d    %03d  \"CalNet L/R, ",
                                styr - CENTURY, jday, sthr, stmn, seci, dur);
                        (void)printf(
                                "m%3.1lf d%.2lf h%.1lf, @%5.2lfN%7.2lfE\"\n",
                                mag, delta, depth, lat, lon);
                }
        }

        return(0);
}

/*
 * Read an event line from QEDs defaulting numbers to zero and
 * strings to null if not present.  Returns TRUE if it seems to
 * be an event line, otherwise FALSE.
 *
 30408 0032 53.41 37 21.36 121 43.04    8.24    1.6  42  58  3.  0.07  0.1  0.4 A2
 30726 0310 18.27 36 50.92 121 35.26    9.16    2.3  60  39  3.  0.16  0.2  0.4 B2
  /
```

```
void
read_ev(in_str, yr, mo, day, hr, mn, sec, lat, lon, depth, mag)
        char    in_str[];               /* Input buffer                 */
        int     *yr, *mo, *day, *hr, *mn; /* Origin time                */
        double  *sec;
        double  *lat, *lon;             /* Epicenter                    */
        double  *depth;                 /* h (km)                       */
        double  *mag;                   /* ML                           */
{
        double  latm, lonm;

        *yr = 0;
        *mo = 0;
        *day = 0;
        *hr = 0;
        *mn = 0;
        *sec = 0.0;
        *lat = 0.0;
        *lon = 0.0;
        *depth = 0.0;
        *mag = 0.0;

        *yr = DECODE(in_str, 2, atoi);
        *mo = DECODE((in_str + 2), 2, atoi);
        *day = DECODE((in_str + 4), 2, atoi);
        *hr = DECODE((in_str + 7), 2, atoi);
        *mn = DECODE((in_str + 9), 2, atoi);
        *sec = (double)DECODE((in_str + 12), 5, atof);
        *lat = (double)DECODE((in_str + 18), 2, atof);
        latm = (double)DECODE((in_str + 21), 5, atof);
        *lat = *lat + latm / 60.;
        *lon = (double)DECODE((in_str + 27), 3, atof);
        lonm = (double)DECODE((in_str + 31), 5, atof);
        *lon = -(*lon + lonm / 60.);            /* Make E+ */
        *depth = (double)DECODE((in_str + 38), 5, atof);
        *mag = (double)DECODE((in_str + 46), 3, atof);
}
```

```
#
#-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
#
# Generic Makefile for programs.
# Works for any mix of sun3, sun4, Fortran, and C (I hope!).
#
# "make install" fails the first time unless the executable has
# already been copied to $(BINDIR).
#
# The following is required in the user's .cshrc:
#
# setenv ARCH `/bin/arch`
# if( $ARCH == "sun3") then
#         setenv FLOAT -f68881
# else
#         setenv FLOAT ""
# endif
# setenv DIR_USR /usr/local
# setenv DIR_SRC /usr/src/local

FC      = f77
CC      = cc
FFLAGS  = -u $(FLOAT) -D$(ARCH)
CFLAGS  = $(FLOAT) -D$(ARCH) -g
BINDIR  = $(DIR_USR)

PROG    = sum2cut
SRCARC  = $(PROG).src.a

FSRCS   =
CSRCS   = sum2cut.c
SCRIPT  = seltran

OBJECTS = $(FSRCS:%.f=$(ARCH)/%.o) $(CSRCS:%.c=$(ARCH)/%.o)

LIBS    = -lg -lm
LLIBS   = $(LIBS)

INCLS   =
OTHERS  = Makefile  $(SCRIPT).sh $(INCLS)
OTHERBIN=

# P     = enscript -r2 -p-

$(ARCH)/$(PROG): $(OBJECTS)
        $(CC) $(CFLAGS) $(OBJECTS) $(LIBS) -o $@

$(OBJECTS): $(INCLS)

$(ARCH)/%.o: %.c
        $(CC) $(CFLAGS) -c $(@F:.o=.c) -o $@

$(ARCH)/%.o: %.f
        $(FC) $(FFLAGS) -c $(@F:.o=.f) -o $@

install: $(ARCH)/install $(ARCH)/$(PROG) $(SCRIPT).sh
#       /bin/rm install
#       ln -s $(ARCH)/install install
        /bin/mv $(BINDIR)/$(PROG) $(BINDIR)/$(PROG).old
        cp $(ARCH)/$(PROG) $(BINDIR)
        chmod 755 $(BINDIR)/$(PROG)
        cp $(SCRIPT).sh $(BINDIR)/$(SCRIPT)
        chmod 755 $(BINDIR)/$(SCRIPT)
        @touch install
        @touch $(ARCH)/install

# CAUTION:  do not run "make clean" on more than one machine at a time;
# the archive commands ("ar") might conflict.
clean:
        /bin/rm $(ARCH)/$(PROG) $(OBJECTS) $(BINDIR)/$(PROG).old
        ar ruv $(SRCARC) $(FSRCS) $(CSRCS) $(OTHERS) $(OTHERBIN)

lint: $(FSRCS) $(CSRCS) $(INCLS)
        /usr/bin/lint -xh -u $(FSRCS) $(CSRCS) $(LLIBS) > lint

print:  $(FSRCS) $(CSRCS) $(OTHERS)
#       pr -l66 $? | $P
        enscript $?
        @touch print

printall:
#       pr -l66 $(FSRCS) $(CSRCS) $(OTHERS) | $P
        enscript $(FSRCS) $(CSRCS) $(OTHERS)
        @touch print

printexport:
        pr -t $(FSRCS) $(CSRCS) $(OTHERS)
```

11. Appendix E: Program *dz* Example and Source Code

This program reads listings produced from the USGS/NEIS on-line hypocenter reporting service called *QED*, and produces input appropriate for the *cutpaste* script. The *QED* service can be reached *via* the Internet by "telnet neis.cr.usgs.gov", and logging in as user "QED". Select option "q", give your name and institution, select option "d" to specify a date window, and finally select option "g" ("go") to obtain the needed event listing. Use text-selection options on your workstation to copy the listing to a file that is then passed to *dz* (edit out all but date and event lines). An example of such an (edited) listing (accessed 23 December 1993) and the resulting output from *dz* follow:

Example of Input to *dz*

```
DEC 18
032212.9   38.824N 122.757W    5G          1.1  35 NORTHERN CALIFORNIA. ML 3.1
205356.0   63.361N 149.817W 100  4.3       0.8  41 CENTRAL ALASKA
211028.7   62.918N 150.663W 136?           0.8  15 CENTRAL ALASKA
224419.9   20.571S 173.881W  33N 5.6 5.7  1.1  82 TONGA ISLANDS

DEC 19
072144.1*  15.353N  94.277W  33N 4.1       0.7  15 NEAR COAST OF OAXACA, MEXICO
080314.1   63.348N 151.241W  10G 4.1       0.7  21 CENTRAL ALASKA
083855.7   37.636N 118.833W   5G          1.0  41 CALIFORNIA-NEVADA BORDER REGION.
103134.1?   7.29 N 126.69 E  33N 5.3 4.9  1.3  25 MINDANAO, PHILIPPINE ISLANDS
114532.1   25.213N  62.614E  33N 5.2 5.1  1.0  46 SOUTHWESTERN PAKISTAN

DEC 20
003659.8*  51.750N 174.986W  33N 4.3       1.2  12 ANDREANOF ISLANDS, ALEUTIAN IS.
030352.3   42.118N 122.028W   5G          0.7  12 OREGON. ML 2.9 (GS).
083406.8   61.882N 150.570W  60G          0.5  13 SOUTHERN ALASKA
135619.3    6.752S 131.314E  33N 6.0 5.8  1.2  82 TANIMBAR ISLANDS REG.,
155657.7   62.206N 150.473W  10G          0.9  38 CENTRAL ALASKA. ML 3.4 (AEIC).
173659.7   32.333N 115.404W   5G          0.6  14 CALIF.-BAJA CALIF. BORDER
```

Resulting Output from *dz*

```
YYYY/MM/DD hh:mm:ss.ss    Lat,N  Long,W  h,km  mb  Ms   delta  azimuth  F-E region

**SKIP**
**SKIP**
1993/12/18 03:22:12.90    38.824 -122.757    5  0.0 0.0    1.732 320.024  NORTHERN CALIFORNIA
1993/12/18 20:53:56.00    63.361 -149.817  100  4.3 0.0   31.038 335.495  CENTRAL ALASKA
1993/12/18 21:10:28.70    62.918 -150.663  136  0.0 0.0   31.027 334.360  CENTRAL ALASKA


1993/12/18 22:44:19.90   -20.571 -173.881   33  5.6 5.7   76.251 229.926  TONGA ISLANDS
(Delta/back-azm:  sft2=075.730/229.638, sft4=076.075/229.826, sft5=076.251/229.926)

  delta    # code       time(s)     (min s)     dT/dD      hh:mm:ss.ss

  76.25    1  P          705.26   11  45.26     5.6799     22:56:05.16
           2  pP         715.61   11  55.61     5.6945     22:56:15.51
           3  PcP        716.88   11  56.88     4.3197     22:56:16.78
           4  sP         719.70   11  59.70     5.6913     22:56:19.60
           5  PP         876.40   14  36.40     8.4227     22:58:56.30
```

6	PKiKP	1050.12	17	30.12	1.4975	23:01:50.02
7	pPKiKP	1060.98	17	40.98	1.4968	23:02:00.88
8	sPKiKP	1064.95	17	44.95	1.4969	23:02:04.85
9	SKiKP	1261.14	21	1.14	1.5536	23:05:21.04
10	S	1287.91	21	27.91	10.9685	23:05:47.81
11	pS	1301.09	21	41.09	11.0094	23:06:00.99
12	sS	1305.57	21	45.57	10.9954	23:06:05.47
13	SKSac	1312.90	21	52.90	7.2666	23:06:12.80
14	SKKSac	1313.90	21	53.90	7.5423	23:06:13.80
15	ScS	1317.63	21	57.63	8.0658	23:06:17.53
16	SPn	1323.76	22	3.76	12.8483	23:06:23.66
17	pSKSac	1327.06	22	7.06	7.2797	23:06:26.96
18	PnS	1328.49	22	8.49	12.8330	23:06:28.39
19	sSKSac	1331.23	22	11.23	7.2759	23:06:31.13
20	SS	1582.91	26	22.91	15.1157	23:10:42.81
21	PKKPdf	1847.02	30	47.02	-1.6060	23:15:06.92
22	PKKPbc	1859.03	30	59.03	-2.1875	23:15:18.93
23	SKKPdf	2058.25	34	18.25	-1.5475	23:18:38.15
24	PKKSdf	2062.23	34	22.23	-1.5473	23:18:42.13
25	SKKSdf	2273.22	37	53.22	-1.4894	23:22:13.12
26	P'P'df	2342.58	39	2.58	-1.8160	23:23:22.48
27	S'S'df	3201.94	53	21.94	-1.6197	23:37:41.84
28	S'S'ac	3213.43	53	33.43	-2.2053	23:37:53.33

(est from JB) LR 2147.22 35 47.22 28.1600 23:20:07.12

cutpaste 1 2 3 dddd jjjj sft5 93 352 22 55 05 2201 "T, @TONGA ISLANDS m5.6/5.7 d76 h33"

SKIP
1993/12/19 07:21:44.10 15.353 -94.277 33 4.1 0.0 32.595 125.501 NEAR COAST OF OAXACA, MEXICO
1993/12/19 08:03:14.10 63.348 -151.241 10 4.1 0.0 31.506 334.656 CENTRAL ALASKA
1993/12/19 08:38:55.70 37.636 -118.833 5 0.0 0.0 1.982 85.452 CALIFORNIA-NEVADA BORDER REGIO

1993/12/19 10:31:34.10 7.290 126.690 33 5.3 4.9 102.549 289.555 MINDANAO, PHILIPPINE ISLANDS
(Delta/back-azm: sft2=102.303/289.160, sft4=102.464/289.420, sft5=102.549/289.555)

delta	#	code	time(s)	(min s)	dT/dD	hh:mm:ss.ss
102.55	1	Pdiff	832.80	13 52.80	4.4389	10:45:26.90
	2	pPdiff	843.37	14 3.37	4.4389	10:45:37.47
	3	sPdiff	847.41	14 7.41	4.4389	10:45:41.51
	4	PP	1086.07	18 6.07	7.5051	10:49:40.17
	5	PKiKP	1094.00	18 14.00	1.8188	10:49:48.10
	6	pPKiKP	1104.85	18 24.85	1.8182	10:49:58.95
	7	sPKiKP	1108.83	18 28.83	1.8183	10:50:02.93
	8	SKiKP	1306.39	21 46.39	1.8639	10:53:20.49
	9	SKSac	1469.75	24 29.75	4.7029	10:56:03.85
	10	pSKSac	1484.32	24 44.32	4.7131	10:56:18.42
	11	sSKSac	1488.37	24 48.37	4.7103	10:56:22.47
	12	SKKSac	1507.57	25 7.57	7.1031	10:56:41.67
	13	Sdiff	1534.65	25 34.65	8.3233	10:57:08.75
	14	pSdiff	1548.59	25 48.59	8.3233	10:57:22.69
	15	sSdiff	1552.83	25 52.83	8.3233	10:57:26.93
	16	SP	1629.76	27 9.76	10.5677	10:58:43.86
	17	PS	1634.19	27 14.19	10.5609	10:58:48.29
	18	PKKPbc	1791.62	29 51.62	-2.9638	11:01:25.72

19	PKKPdf	1800.98	30	0.98	-1.8550	11:01:35.08
20	SS	1964.01	32	44.01	13.8199	11:04:18.11
21	SKKPbc	2010.49	33	30.49	-2.7106	11:05:04.59
22	SKKPdf	2013.36	33	33.36	-1.8275	11:05:07.46
23	PKKSbc	2014.49	33	34.49	-2.7101	11:05:08.59
24	PKKSdf	2017.35	33	37.35	-1.8274	11:05:11.45
25	SKKSdf	2229.59	37	9.59	-1.7946	11:08:43.69
26	SKKSac	2231.59	37	11.59	-2.5204	11:08:45.69
27	P'P'df	2293.12	38	13.12	-1.9200	11:09:47.22
28	S'S'ac	3146.04	52	26.04	-2.9314	11:24:00.14
29	S'S'df	3155.66	52	35.66	-1.8597	11:24:09.76

(est from JB) LR 2887.79 48 7.79 28.1600 11:19:41.89

cutpaste 1 2 3 dddd jjjj sft5 93 353 10 44 27 2814 "T, @MINDANAO, PHILIPPINE ISLANDS m5.3/4.9 d1

SKIP

1993/12/19 11:45:32.10	25.213	62.614	33	5.2 5.1	117.173 355.990	SOUTHWESTERN PAKISTAN
1993/12/20 00:36:59.80	51.750	-174.986	33	4.3 0.0	39.720 308.708	ANDREANOF ISLANDS, ALEUTIAN IS
1993/12/20 03:03:52.30	42.118	-122.028	5	0.0 0.0	4.644 353.578	OREGON
1993/12/20 08:34:06.80	61.882	-150.570	60	0.0 0.0	30.321 332.870	SOUTHERN ALASKA

1993/12/20 13:56:19.30 -6.752 131.314 33 6.0 5.8 107.855 275.251 TANIMBAR ISLANDS REG
(Delta/back-azm: sft2=107.503/274.835, sft4=107.734/275.109, sft5=107.855/275.251)

delta	#	code	time(s)	(min s)		dT/dD	hh:mm:ss.ss
107.86	1	Pdiff	856.35	14	16.35	4.4389	14:10:35.65
	2	pPdiff	866.92	14	26.92	4.4389	14:10:46.22
	3	sPdiff	870.96	14	30.96	4.4389	14:10:50.26
	4	PKiKP	1103.79	18	23.79	1.8678	14:14:43.09
	5	pPKiKP	1114.63	18	34.63	1.8673	14:14:53.93
	6	sPKiKP	1118.61	18	38.61	1.8674	14:14:57.91
	7	PP	1125.38	18	45.38	7.3138	14:15:04.68
	8	SKiKP	1316.40	21	56.40	1.9091	14:18:15.70
	9	SKSac	1493.58	24	53.58	4.2885	14:21:12.88
	10	pSKSac	1508.21	25	8.21	4.2969	14:21:27.51
	11	sSKSac	1512.25	25	12.25	4.2946	14:21:31.55
	12	SKSdf	1533.19	25	33.19	1.9360	14:21:52.49
	13	SKKSac	1544.88	25	44.88	6.9572	14:22:04.18
	14	pSKSdf	1548.01	25	48.01	1.9360	14:22:07.31
	15	sSKSdf	1552.00	25	52.00	1.9360	14:22:11.30
	16	Sdiff	1578.82	26	18.82	8.3233	14:22:38.12
	17	pSdiff	1592.75	26	32.75	8.3233	14:22:52.05
	18	sSdiff	1596.99	26	36.99	8.3233	14:22:56.29
	19	SP	1684.89	28	4.89	10.2077	14:24:24.19
	20	SP	1686.52	28	6.52	9.2372	14:24:25.82
	21	PS	1689.29	28	9.29	10.2008	14:24:28.59
	22	PS	1690.83	28	10.83	9.2370	14:24:30.13
	23	PKKPbc	1775.42	29	35.42	-3.1450	14:25:54.72
	24	PKKPab	1790.37	29	50.37	-4.4368	14:26:09.67
	25	PKKPdf	1791.07	29	51.07	-1.8790	14:26:10.37
	26	SKKPbc	1995.73	33	15.73	-2.8543	14:29:35.03
	27	PKKSbc	1999.73	33	19.73	-2.8537	14:29:39.03
	28	SKKPdf	2003.58	33	23.58	-1.8574	14:29:42.88
	29	PKKSdf	2007.57	33	27.57	-1.8573	14:29:46.87

```
          30  SS        2036.57  33  56.57    13.5324    14:30:15.87
          31  SKKSac    2217.88  36  57.88    -2.6457    14:33:17.18
          32  SKKSdf    2219.97  36  59.97    -1.8308    14:33:19.27
          33  P'P'df    2282.91  38   2.91    -1.9271    14:34:22.21
          34  S'S'ac    3130.07  52  10.07    -3.0873    14:48:29.37
          35  S'S'df    3145.73  52  25.73    -1.8824    14:48:45.03

(est from JB) LR        3037.19  50  37.19    28.1600    14:46:56.49

cutpaste    1 2 3    dddd jjjj sft5    93 354 14 09 36   2940  "T, @TANIMBAR ISLANDS REG m6.0/5.8 d108 h33"

**SKIP**
1993/12/20 15:56:57.70   62.206 -150.473  10  0.0 0.0   30.496 333.417  CENTRAL ALASKA
1993/12/20 17:36:59.70   32.333 -115.404   5  0.0 0.0    7.092 135.061  CALIF
```

Source Code of *dz* Follows:

The source code is available *via* anonymous *ftp* from "andreas.wr.usgs.gov" in directory "~ftp/pub/outgoing/evans/event_extraction". This Open-File Report serves to release them to the public, subject to the limitations cited on the title page.

```
/*
 * DZ -- simple distance, azimuth, and traveltime calculator,
 * input from QED, calls ttimes with distance value, outputs
 * all relevant data.  Positive North and East.
 *
 * 27 Sept 93, add output of "epick" "pick" files for each event.
 * Only outputs for channel code "v", i.e., for rotated traces.
 * Requires change in logic so all stations get computed delta/azm
 * and one pick structure is made for each.  Also added output of
 * all backazimuths to facilitate correct rotations later.
 *
 * USAGE:  dz QED_file_name > output_file
 *
 * BUGS:
 *
 * WARNING:  epick "pick" files have times for SFT5 exactly, but
 * estimates values for other sites from ttimes-output slownesses.
 * Interpret with care.
 *
 * Must have the following links in place:
    ln -s /we/itch/evans/src/resids/iasp91/iasp91.hed .
    ln -s /we/itch/evans/src/resids/iasp91/iasp91.tbl .
 *
 * Works only on Sun3 "evanss"--fragile.  Fixable, but need ttimes
 * available and need to change path names appropriately.
 *
 * Creates and destroys files "ttimes.in.junk" and "ttimes.out.junk".
 *
 * Remember to update DATA_YEAR, CENTURY, the definitions of
 * central_lat and central_lon, and the format of "cutpaste" output
 * (the station name) as needed.  Not neat, but functional.
 *
 * Also, change (0, MIN_MAG_0), (MID_DELTA, MIN_MAG_1) and
 * (180, MIN_MAG_2) which define a bi-linear distance-magnitude
 * rule determining which epicenters from the imput file are
 * processed and which are skipped.  (The minimum magnitude so
 * determined is compared to the maximum of mb and Ms.)
 *
 */

#include <stdio.h>
#include <math.h>
#include <string.h>
#include <local/local.h>
#include <local/stdtyp.h>
#include <local/mathconst.h>
#include <local/date_time.h>
#include "pick_struct.h"

#define MAX(a,b) (((a)>(b)) ? (a) : (b))        /* Larger value */

#define MAXBUF          40      /* decode.h buffer size         */
#include <local/decode.h>

/* Maximum size of QED event-list */
#define MXQED           2000
#define FE_LEN          31      /* Flinn-Engdahl array length   */
#define STR_LEN         256     /* Maximum scratch-string length */

#define DATA_YEAR       1993
#define CENTURY         1900
#define MIN_MAG_0       2.5     /* minimum mb at zero distance  */
#define MID_DELTA       5.      /* slope-bread distance (deg)   */
#define MIN_MAG_1       4.9     /* minimum mb at MID_DELTA      */
#define MIN_MAG_2       5.6     /* minimum mb at the antipode   */

/* Coordinates of possible station(s) from
   which to calculate delta and azimuth */
#define SFT2NAM         "sft2"
#define SFT2LAT         37.153042
#define SFT2LON         -121.812232
#define SFT2ELEV        518

#define SFT4NAM         "sft4"
#define SFT4LAT         37.38883
#define SFT4LON         -121.49450
#define SFT4ELEV        660

#define SFT5NAM         "sft5"
#define SFT5LAT         37.505053
#define SFT5LON         -121.328514
#define SFT5ELEV        312

#define NSTA            3

typedef struct {                /* Station information          */
        char    code[5];        /* Name                         */
        double  lat;            /* Location (+N, degrees)       */
```

```
        double  lon;            /* (+E, degrees)                */
        double  elev;           /* meters                       */
        double  delta;          /* Distance to event, degrees   */
} STA;

/* Surface wave nominal velocity (s/degree) */
#define LR_VEL          28.16

typedef struct {                /* Known-event information      */
        TIME    ot;             /* Origin time                  */
        double  lat;            /* Hypocenter (+N, degrees)     */
        double  lon;            /* +E, degrees                  */
        float   depth;          /* kilometers                   */
        float   mag;            /* mb                           */
        float   ms;             /* Ms                           */
        char    fe[FE_LEN];     /* Flinn-Engdahl region name    */
        double  delta;          /* Distance (degrees)           */
        double  back;           /* Back azimuth (degrees)       */
        int     hit;            /* Number of times triggered on */
} QEDMAP;

#define strneq !strncmp

        void    ep_out();       /* Defined below                */
        int     build_qed();    /* Defined below                */
        void    ph_sav();       /* Defined below                */
        bool    read_ev();      /* Defined below                */
        char    *tm_str();      /* Defined below                */
        char    *tm_cut();      /* Defined below                */

        void    datime();       /* -lq (YMDhms <- epochal)      */
        void    delaz();        /* -lq (geocen.c)               */
        FILE    *fopen();       /* Unix C library               */
        FILE    *efopen();      /* -lq (efopen.c)               */
        void    error();        /* -lq (error.c)                */
        bool    isleap();       /* -lq                          */
        void    refpt();        /* -lq (geocen.c)               */
        void    report();       /* -lq (error.c)                */
        TIME    timvar();       /* -lq (YMDhms -> epochal)      */

#define MAXPHASE        100     /* Maximum number of phases     */
        int     npcks;          /* Number of phases for event   */
        TIME    ph_t[MAXPHASE]; /* Phase travel time            */
        double  ph_s1[MAXPHASE]; /* Phase slowness, s/degree    */
        double  ph_s2[MAXPHASE]; /* Phase slowness derivative, s/d^2 ??? */
        char    ph_n[MAXPHASE][7]; /* Phase name                */

main(argc, argv)
        int     argc;
        char    **argv;
{
        int     yr, mo, day, hr, mn;
        double  sec;

        int     nqed;           /* Number of events read        */

        int     ii;             /* Dummy                        */
        int     jj;             /* Dummy                        */
        int     skiped_last = FALSE;    /* Output flag          */

        float   min_mag;        /* Minimum magnitude to pass    */

        char    qed_name[40];   /* Name of "QED" file           */
        FILE    *qed_file;
        QEDMAP  qedm[MXQED];    /* QED-event map                */

        FILE    *tt_file;       /* ttimes I/O files             */
        char    in_str[STR_LEN]; /* Input buffer                */
        char    ttout[16];      /* Output buffer for time of day */

        char    sys_str[256];   /* ttimes command buffer        */

        bool    first_ph = TRUE; /* Take note of first-arriving phase */
        TIME    phltime;        /* Its traveltime               */

        double  central_lat = SFT5LAT;  /* Latitude of station  */
        double  central_lon = SFT5LON;  /* Longitude of station */

        STA     stations[10];   /* Station list                 */

/* Build station list the dumb way */

(void)strncpy(stations[0].code, SFT2NAM, 5);
stations[0].lat = SFT2LAT;
stations[0].lon = SFT2LON;
stations[0].elev = SFT2ELEV;

(void)strncpy(stations[1].code, SFT4NAM, 5);
```

```
        stations[1].lat = SFT4LAT;
        stations[1].lon = SFT4LON;
        stations[1].elev = SFT4ELEV;

        (void)strncpy(stations[2].code, SFT5NAM, 5);
        stations[2].lat = SFT5LAT;
        stations[2].lon = SFT5LON;
        stations[2].elev = SFT5ELEV;

        /* Check usage */
        if (argc < 2)
                error ("Usage: %s QED_file_name > output_file", argv[0]);

        qed_name[0] = '\0';
        --argc, argv++;
        (void)strcpy(qed_name, *argv);  /* "QED" file name      */

        /* Use middle station as reference point for build_qed */
        refpt((central_lat * RAD), (central_lon * RAD));

        /* Read in QED list */
        if((qed_file = fopen(qed_name, "r")) == NULL)
                error("Error opening \"QED\" file \"%s\".", qed_name);
        if ((nqed = build_qed(qedm, qed_file)) > 0) {
                (void)printf("YYYY/MM/DD hh:mm:ss.ss    Lat,N  Long,W");
                (void)printf("  h,km  mb  Ms   delta  azimuth, F-E region\n");

        /* Run ttimes for each event, extracting the useful fraction */
        sys_str[0] = '\0';
        (void)strcat(sys_str,
                "/we/geotele/evans/usr/ttimes");
        (void)strcat(sys_str, " < ttimes.in.junk");
        (void)strcat(sys_str, " > ttimes.out.junk");

        for (ii = 0 ; ii < nqed ; ii++) {

                /* Bother only with qualifying magnitudes */

                if (qedm[ii].delta < MID_DELTA) {
                        min_mag = MIN_MAG_0 +
                                (MIN_MAG_1 - MIN_MAG_0) *
                                qedm[ii].delta / MID_DELTA;
                }

                else {
                        min_mag = MIN_MAG_1 +
                                (MIN_MAG_2 - MIN_MAG_1) *
                                (qedm[ii].delta - MID_DELTA) /
                                (180.0 - MID_DELTA);
                }

                if (MAX(qedm[ii].mag, qedm[ii].ms) >= min_mag) {

                        if (skiped_last)
                                (void)printf("\n");
                        skiped_last = FALSE;

                        /* Create input to ttimes, and run ttimes */

                        tt_file = fopen("ttimes.in.junk", "w");
                        (void)fprintf(tt_file,
                                "all\n\n%.3f\n%.3lf\n-1\n-1\n",
                                qedm[ii].depth, qedm[ii].delta);
                        (void)fclose(tt_file);

                        if ((system(sys_str) & 0377) == 127)
                                error("ttimes call failed!");

                        /* Output relevent event information */

                        datime(qedm[ii].ot,
                                &yr, &mo, &day, &hr, &mn, &sec,
                                GREGORIAN);
                        (void)printf("\n%4i/%02i/%02i %02i:%02i:%05.2f  ",
                                yr, mo, day, hr, mn, sec);
                        (void)printf("%7.3lf %8.3lf %3.0f  ",
                                qedm[ii].lat, qedm[ii].lon,
                                qedm[ii].depth);
                        (void)printf("%3.1f %3.1f  %7.3lf %7.3lf  ",
                                qedm[ii].mag, qedm[ii].ms,
                                qedm[ii].delta,
                                qedm[ii].back);
                        (void)printf("%s\n",
                                qedm[ii].fe);

                        /* Calculate and output all deltas
                           and backazimuths */
                        (void)printf("(Delta/back-azm:  ");
```

```
        for (jj = 0 ; jj < NSTA ; jj++) {
                double  delta, az0, az1;

                refpt((stations[jj].lat * RAD),
                        (stations[jj].lon * RAD));
                delaz(qedm[ii].lat * RAD,
                        qedm[ii].lon * RAD,
                        &delta, &az0, &az1);
                az0 *= DEG;
                delta *= DEG;
                stations[jj].delta = delta;

                (void)printf("%s=%07.3lf/%07.3lf",
                        stations[jj].code,
                        delta, az0);
                if(jj == NSTA - 1)
                        (void)printf(")\n");
                else
                        (void)printf(", ");
        }
        (void)printf("\n");

        /* copy through the useful parts of ttimes
           output */
        npcks = 0;      /* Start epick phase list */
        tt_file = fopen("ttimes.out.junk", "r");
        while(fgets(in_str, sizeof(in_str), tt_file) !=
                NULL)
                if (strneq("Source depth (km):",
                        in_str, 18))
                        break;
        while(fgets(in_str, sizeof(in_str), tt_file) !=
                NULL) {
                if (strneq(
                        "Enter delta:  Source depth (km):",
                        in_str, 32))
                        break;
                else {
                        if (strneq("Enter delta:  ",
                           in_str, 14)) {
                           in_str[53 + 14] = '\0';
                           (void)printf(
                              "%s    hh:mm:ss.ss\n",
                              in_str + 14);
                        }
                        else {
                           TIME tt = 0;
                           tt = (double)DECODE(
                                 (in_str + 22), 9, atof);
                           if (tt > 0.0) {
                              if(first_ph) {
                                 phltime = tt;
                                 first_ph = FALSE;
                              }
                              in_str[53] = '\0';
                              (void)printf("%s    %s\n",
                                 in_str, tm_str(
                                 qedm[ii].ot, tt,
                                 ttout));

                              /* Save phase for
                                 epick "pick" file */
                              ph_sav(in_str, tt);
                           }
                           else
                              (void)printf("%s", in_str);
                        }
                }

                /* Protect DECODE from leftovers */
                for (jj = 0 ; jj < STR_LEN ; jj++)
                        in_str[jj] = ' ';
        }


        /* Estimate surface waves */
        sec = qedm[ii].delta * LR_VEL;
        mn = sec / 60;
        (void)printf("(est from JB) LR");
        (void)printf("     %10.2f%4i%7.2f%11.4f    %s\n\n",
                sec, mn, (sec - (mn * 60)), LR_VEL,
                tm_str(qedm[ii].ot, sec, ttout));

        (void)fclose(tt_file);
        first_ph = TRUE;
        if ((system("/bin/rm ttimes.out.junk") &
                0377) == 127)
```

```
                        report("rm call failed...");

                    /* Output for "cutpaste" script
                       (from (first phase - PAD_FRONT)
                       to (LR + PAD_REAR)) */

#define             PAD_FRONT       60.0    /* s */
#define             PAD_REAR        700.0   /* s */

                    (void)printf("cutpaste    1 2 3    ");
                    (void)printf("dddd jjjj sft5    ");
                    (void)printf("%s   %4d  \"T, @%s ",
                            tm_cut((TIME)(qedm[ii].ot + phltime -
                                    PAD_FRONT + 0.5), ttout),
                            (int)(sec - phltime +
                                    PAD_FRONT + PAD_REAR),
                            qedm[ii].fe);
                    (void)printf("m%3.1f/%3.1f d%.01f h%.0f\"\n",
                            qedm[ii].mag, qedm[ii].ms,
                            qedm[ii].delta, qedm[ii].depth);

                    /* Output phase list to epick "pick" file */
                    ep_out(qedm[ii].ot, ttout,
                            qedm[ii].delta, stations);
                }

                /* Report skipped events FYI */
                else {
                        datime(qedm[ii].ot,
                                &yr, &mo, &day, &hr, &mn, &sec,
                                GREGORIAN);

                        if (!skiped_last)
                                (void)printf("\n**SKIP**\n");
                        skiped_last = TRUE;

                        (void)printf("%4i/%02i/%02i %02i:%02i:%05.2f  ",
                                yr, mo, day, hr, mn, sec);
                        (void)printf("%7.3lf %8.3lf %3.0f  ",
                                qedm[ii].lat, qedm[ii].lon,
                                qedm[ii].depth);
                        (void)printf("%3.1f %3.1f  %7.3lf %7.3lf  ",
                                qedm[ii].mag, qedm[ii].ms,
                                qedm[ii].delta,
                                qedm[ii].back);
                        (void)printf("%s\n",
                                qedm[ii].fe);
                }
            }

            if ((system("/bin/rm ttimes.in.junk") & 0377) == 127)
                    report("rm call failed...");
        }

        (void)fclose(qed_file);
}

/*
 * Given two TIME variables, add, and return time string "hh:mm:ss.ss".
 */

char *
tm_str(t1, t2, str)
        TIME    t1;
        TIME    t2;
        char    *str;
{
        int     yr, mo, day, hr, mn;
        double  sec;

        datime((t1 +  t2), &yr, &mo, &day, &hr, &mn, &sec, GREGORIAN);
        (void)sprintf(str, "%02i:%02i:%05.2f", hr, mn, sec);

        return (str);
}

/*
 * Given one TIME variable, returns time string "yy jjj hh mm ss".
 */

char *
tm_cut(tt, str)
        TIME    tt;
        char    *str;
{
        int     yr, mo, day, hr, mn, jday;
        double  sec;
```

```
        datime(tt, &yr, &mo, &day, &hr, &mn, &sec, GREGORIAN);
        jday = yrday(mo, day, isleap(yr, GREGORIAN));
        (void)sprintf(str, "%02i %03i %02i %02i %02i",
                yr - CENTURY, jday, hr, mn, (int)sec);

        return (str);
}

/*
 * Read an event line from QEDs defaulting numbers to zero and
 * strings to null if not present.  Returns TRUE if it seems to
 * be an event line, otherwise FALSE.
 */

bool
read_ev(in_str, hr, mn, sec, lat, lath, lon, lonh, depth, mag, ms, fe)
        char    in_str[];               /* Input buffer                 */
        int     *hr, *mn;
        double  *sec;
        double  *lat, *lon;
        float   *depth, *mag, *ms;
        char    *lath, *lonh;
        char    fe[];                   /* Flinn-Engdahl buffer         */
{
        int     ii;                     /* Dummy                        */
        bool    rd;

rd = FALSE;

*hr = 0;
*mn = 0;
*sec = 0.0;
*lat = 0;
*lath = '\0';
*lon = 0;
*lonh = '\0';
*depth = 0;
*mag = 0;
*ms = 0;
for (ii = 0 ; ii < FE_LEN ; ii++)
        fe[ii] = '\0';

*hr = DECODE(in_str, 2, atoi);
*mn = DECODE((in_str + 2), 2, atoi);
*sec = (double)DECODE((in_str + 4), 4, atof);
*lat = (double)DECODE((in_str + 10), 6, atof);
*lath = in_str[16];
*lon = (double)DECODE((in_str + 18), 7, atof);
*lonh = in_str[25];
*depth = (float)DECODE((in_str + 27), 3, atof);
*mag = (float)DECODE((in_str + 32), 3, atof);
*ms = (float)DECODE((in_str + 36), 3, atof);

(void)strcpy(fe, (in_str + 48));
fe[FE_LEN - 1] = '\0';

/* Strip trailing comments (avoiding 1-character abreviations) */
for (ii = 2 ; ii < FE_LEN ; ii++) {
        if ((fe[ii] == '.' && fe[ii - 2] != ' ') || fe[ii] == '\n') {
                fe[ii] = '\0';
                break;
        }
}

        /* Kludge, since there seems no other guarantee in QED listings */
        if ((*lath == 'N' || *lath == 'n' || *lath == 'S' || *lath == 's') &&
                (*lonh == 'E' || *lonh == 'e' || *lonh == 'W' || *lonh == 'w'))
                rd = TRUE;

        return (rd);
}

/*
 * Read "QED" file and build data base of events.
 */

int
build_qed(qedm, qed_file)
        QEDMAP  qedm[];                 /* Known-event map              */
        FILE    *qed_file;
{
        int     nqed;
        char    in_str[STR_LEN];        /* Input buffer                 */
        int     yr = DATA_YEAR, mo = 0, day, hr, mn;
        double  sec;
        double  lat, lon;
        float   depth, mag, ms;
        char    lath, lonh;
```

```
        char    fe[FE_LEN];             /* Flinn-Engdahl buffer         */
        double  delta, az0, az1;
        int     ii;                     /* Dummy                        */

        /* refpt called from here in other versions.  In "dz", called from main. */

        nqed = 0;
        while (fgets(in_str, sizeof(in_str), qed_file) != NULL) {

                ii = intmo(in_str);     /* Always present above events  */
                if (ii != 0) {
                        mo = ii;
                        if (sscanf(in_str + 3, "%d", &day) != 1)
                                error("Could not read day in QEDs");
                }

                if (read_ev(in_str, &hr, &mn, &sec,
                        &lat, &lath, &lon, &lonh, &depth, &mag, &ms, fe)) {

                        /* Clear input string to avoid problems
                         * with sugsequent event and non-event lines */
                        for (ii = 0 ; ii < STR_LEN ; ii++)
                                in_str[ii] = '\0';

                        if (lath != 'N')
                                lat = -lat;

                if (lonh != 'E')
                        lon = -lon;


                /* Calculate distance to event */
                delaz(lat * RAD, lon * RAD, &delta, &az0, &az1);
                delta *= DEG;
                az0 *= DEG;

                if (mo == 0)
                        error("Programming error.");

                if (nqed >= MXQED)
                        error("Too many QED entries.");

                /* Save event information */
                qedm[nqed].ot =
                        timvar(yr, mo, day, hr, mn, sec, GREGORIAN);

                qedm[nqed].lat = lat;
                qedm[nqed].lon = lon;

                qedm[nqed].depth = depth;
                qedm[nqed].mag = mag;
                qedm[nqed].ms = ms;

                (void)strcpy(qedm[nqed].fe, fe);

                qedm[nqed].delta = delta;
                qedm[nqed].back = az0;
                qedm[nqed].hit = 0;

                nqed++;
                        nqed++;
                }
        }
        return (nqed);
}


/*
 * Add picks for one phase to the epick "pick" file
 */

void
ep_out(ot, str, delta, stations)
        TIME    ot;                     /* Event origin time            */
        char    str[];                  /* "cuspaste" string passed in  */
        double  delta;                  /* "center" (SFT5) delta, deg   */
        STA     stations[];             /* Station list                 */
{
        FILE    *ep_file;               /* Output stream                */
        PMAIN   pck;                    /* An epick "pick"              */
        int     iph, ist;
        double  ddelta;

        /* Open file (construct name from "cuspaste" string) */
        str[2] = str[3];
        str[3] = str[4];
        str[4] = str[5];
        str[5] = '.';
        str[6] = str[7];
```

```
        str[7] = str[8];
        str[8] = str[10];
        str[9] = str[11];
        str[10] = str[13];
        str[11] = str[14];
        str[12] = '.';
        str[13] = 'e';
        str[14] = 'p';
        str[15] = '\0';

        ep_file = efopen(str, "w");

        /* Create and write out epick "pick" structures */
        for (iph = 0 ; iph < npcks ; iph++) {
                for (ist = 0 ; ist < NSTA ; ist++) {

                        (void)strcpy(pck.stcode, stations[ist].code);
                        (void)strcpy(pck.chan, "v");

                        (void)strcpy(pck.phcode, ph_n[iph]);
                        ddelta = stations[ist].delta - delta;
                        pck.t = ot + ph_t[iph] + ddelta * ph_s1[iph];
/*                              (ph_s1[iph] + ph_s2[iph] * ddelta / 2); */
                        pck.setime = 999.999;

                        pck.onset = ' ';
                        pck.pol = ' ';
                        pck.qual = 'i';         /* Meaning "iaspi91"    */

                        pck.amp = 0.0;
                        pck.freq = 0.0;

                        if (fwrite((char *)&pck, sizeof(pck), 1, ep_file) != 1)
                                report("Can't write picks");
                }
        }

        /* Clean up */
        (void)fclose(ep_file);
}

/*
 * Save phase traveltimes and names for epick "pick" file.
 */

void
ph_sav(in_str, tt)
        char    *in_str;
        TIME    tt;
{
        int     jj;

        if (npcks == MAXPHASE)
                report("Too many phase names--ignored\n%s", in_str);

        else {
                ph_t[npcks] = tt;
                ph_s1[npcks] = (double)DECODE((in_str + 46), 10, atof);
                ph_s2[npcks] = (double)DECODE((in_str + 65), 10, atof);

                (void)strncpy(ph_n[npcks], in_str + 14, 6);
                ph_n[npcks][6] = '\0';
                for (jj = 5 ; jj >= 0 ; jj--)
                        if(ph_n[npcks][jj] == ' ')
                                ph_n[npcks][jj] = '\0';

                npcks++;
        }
}
```

```
/* Character month -> integer month */

#include <string.h>

#define strneq  !strncmp
#define streq   !strcmp

int
intmo(chmo)
        char    *chmo;
{
        if (strneq(chmo, "Jan", 3) || strneq(chmo, "JAN", 3))
                return(1);
        else if (strneq(chmo, "Feb", 3) || strneq(chmo, "FEB", 3))
                return(2);
        else if (strneq(chmo, "Mar", 3) || strneq(chmo, "MAR", 3))
                return(3);
        else if (strneq(chmo, "Apr", 3) || strneq(chmo, "APR", 3))
                return(4);
        else if (strneq(chmo, "May", 3) || strneq(chmo, "MAY", 3))
                return(5);
        else if (strneq(chmo, "Jun", 3) || strneq(chmo, "JUN", 3))
                return(6);
        else if (strneq(chmo, "Jul", 3) || strneq(chmo, "JUL", 3))
                return(7);
        else if (strneq(chmo, "Aug", 3) || strneq(chmo, "AUG", 3))
                return(8);
        else if (strneq(chmo, "Sep", 3) || strneq(chmo, "SEP", 3))
                return(9);
        else if (strneq(chmo, "Oct", 3) || strneq(chmo, "OCT", 3))
                return(10);
        else if (strneq(chmo, "Nov", 3) || strneq(chmo, "NOV", 3))
                return(11);
        else if (strneq(chmo, "Dec", 3) || strneq(chmo, "DEC", 3))
                return(12);
        else
                return(0);
}
```

```
/* STCSIZE should be abandoned in favor of using CODESIZE from ahhead.h */

/*
 * STCSIZE and CHANSIZE define the maximum number of PRINTABLE characters
 * in the respective strings.  These strings, therefore, MAY NOT BE NULL
 * TERMINATED.  Use 'strncmp(..., STCSIZE)' and '... "%.*s", STCSIZE,
 * stcode ...' and such.
 */

#define STCSIZE         4               /* Length of station code       */
#define CHANSIZE        6               /* Length of channel code       */
#define PHCSIZE         8               /* Length of phase code         */
#define NULL_TIME       1.0e10

/* Indespensable part of a PICK */
typedef struct {
        char    stcode[STCSIZE];/* Station code                         */
        char    chan[CHANSIZE]; /* Channel code                         */
        char    phcode[PHCSIZE];/* Phase code                           */
        TIME    t;              /* Arrival time                         */
        float   setime;         /* Standard error of arrival time       */
        char    onset;          /* 'e' or 'i'                           */
        char    pol;            /* Polarity:  'u' or 'd'                */
        char    qual;           /* Pick quality                         */
        float   amp;            /* Amplitude                            */
                                /* Note: convention (e.g. peak-to-peak  */
                                /* vs. zero-to-peak) is not enforced by */
                                /* program (but is important).          */
        float   freq;           /* Frequency                            */
} PMAIN;

/* Time from clock or radio time channel */
typedef struct {
        TIME    t;
} CODE_T;

/* Related things measured from a trace */
typedef struct {
        PMAIN   pm;             /* Required                             */
        CODE_T  c_t;            /* Optional                             */
} PICK;
```

```
#
#-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
#
# Generic Makefile for programs.
# Works for any mix of sun3, sun4, Fortran, and C (I hope!).
#
# "make install" fails the first time unless the executable has
# already been copied to $(BINDIR).
#
# The following is required in the user's .cshrc:
#
# setenv ARCH `/bin/arch`
# if( $ARCH == "sun3") then
#         setenv FLOAT -f68881
# else
#         setenv FLOAT ""
# endif
# setenv DIR_USR /usr/local
# setenv DIR_SRC /usr/src/local

FC      = f77
CC      = cc
FFLAGS  = -u $(FLOAT) -D$(ARCH)
CFLAGS  = $(FLOAT) -D$(ARCH) -g
BINDIR  = $(DIR_USR)

PROG    = dz
SRCARC  = $(PROG).src.a

FSRCS   =
CSRCS   = dz.c \
        intmo.c
OBJECTS = $(FSRCS:%.f=$(ARCH)/%.o) $(CSRCS:%.c=$(ARCH)/%.o)

LIBS    = -lq -lm
LLIBS   = $(LIBS)

INCLS   =
OTHERS  = Makefile $(INCLS)
OTHERBIN=

# P     = enscript -r2 -p-

$(ARCH)/$(PROG): $(OBJECTS)
        $(CC) $(CFLAGS) $(OBJECTS) $(LIBS) -o $@

$(OBJECTS): $(INCLS)

$(ARCH)/%.o: %.c
        $(CC) $(CFLAGS) -c $(@F:.o=.c) -o $@

$(ARCH)/%.o: %.f
        $(FC) $(FFLAGS) -c $(@F:.o=.f) -o $@

install: $(ARCH)/install $(ARCH)/$(PROG)
#       /bin/rm install
#       ln -s $(ARCH)/install install
        /bin/mv $(BINDIR)/$(PROG) $(BINDIR)/$(PROG).old
        cp $(ARCH)/$(PROG) $(BINDIR)
        chmod 755 $(BINDIR)/$(PROG)
        @touch install
        @touch $(ARCH)/install

# CAUTION:  do not run "make clean" on more than one machine at a time;
# the archive commands ("ar") might conflict.
clean:
        /bin/rm $(ARCH)/$(PROG) $(OBJECTS) $(BINDIR)/$(PROG).old
        ar ruv $(SRCARC) $(FSRCS) $(CSRCS) $(OTHERS) $(OTHERBIN)

lint: $(FSRCS) $(CSRCS) $(INCLS)
        /usr/bin/lint -xh -u $(FSRCS) $(CSRCS) $(LLIBS) > lint

print:  $(FSRCS) $(CSRCS) $(OTHERS)
#       pr -166 $? | $P
        enscript $?
        @touch print

printall:
#       pr -166 $(FSRCS) $(CSRCS) $(OTHERS) | $P
        enscript $(FSRCS) $(CSRCS) $(OTHERS)
        @touch print

printexport:
        pr -t $(FSRCS) $(CSRCS) $(OTHERS)
```

12. Appendix F: Manual Page and Source Code for *ahwwvb*

The algorithm used in the *ahwwvb* time-code reader is an amalgam of robust processes coded in C and FORTRAN—several running-median filters, an edge-finding algorithm, and an L1 fit to the second ticks. The latter, and the BCD deciphering code that follows, are taken from Carl Johnson's earlier efforts, while the signal preprocessors are creations of the first author of this Report. The signal processors were designed from general principals (*Evans*, 1992) and optimized by tests on real data, mostly of 100 sps. It also works well at 250 sps, but is only lightly tested at other sample rates. It is the most reliable automated WWVB reader known to us.

Both the source code and the manual page are available *via* anonymous *ftp* from "andreas.wr.usgs.gov" in directory "~ftp/pub/outgoing/evans/wwvb". This Open-File Report serves to release them to the public, subject to the limitations cited on the title page.

Source code for the running-median routine (from the first author's *librmf*), and relevant routines from Bruce Julian's *libq* are listed in Appendix H. The I/O library *libah* is available from Lamont-Doherty ("ftp lamont.ldeo.columbia.edu" in compressed file "/pub/ah.tar.Z").

SYNOPSIS

ahwwvb -m *min_rate ah_file* [**-vb** *code channel*] [**-v**] [**-la** | **-l** *list_file*] [**-y** *year*] [**-r** | **-s**] [**-d** *delay*] [**-nc**]

DESCRIPTION

Reads a WWVB radio time-code trace in (*xdr*) AH format, infers sampling rate and the NTIS/UTC time of the first sample. Accordingly, changes the WWVB trace starting time and digitizing interval, as well as those of any other traces indicated. Affects all traces listed in *list_file* (in the same format as *epick*'s *list_file*), or else changes *all* traces in *ah_file* (**-la**, the default). If **-l** is used, copies through, unchanged, all other traces encountered. Performs simple relatedness checks on traces to be changed, *but only with the* **-la** *option*. Simply asks whether they all have the same length, original sampling interval, and original starting time.

ahwwvb is intended primarily for correcting start time and digitizing interval of data from "five-day recorders" and other instruments that record WWVB radio time code in parallel with seismic data, but which may give inaccurate start time or digitizing interval due to inaccurate internal clocks or analog-instrumentation foibles. Hence, *ahwwvb* assumes that all traces to be changed are *exactly the same* time interval and that *all* traces from that parallel record are targeted for the change. Since AH has no formal way of associating a group of traces, this game is somewhat dangerous. *You may get bizarre results if you change too few or too many traces.* Logs the changes it makes in the AH header.

ahwwvb does significant signal massaging to extract robustly the most accurate time. Uses a combination of several running-median filters and an edge-finding algorithm to interpret the WWVB trace into a binary time series. This binary series is decoded by Carl Johnson's routine, the operation of which is partly mysterious. The translation to binary includes a number of empirically set parameters for determining what a sharp corner really is and how extreme the signal massaging may be. These may need adjustment in some cases (see file *ahwwvb.h*). Also biases the result to account for a filtered *versus* unfiltered WWVB trace. Uses an L1 fit (in C. Johnson's code) to calculate fractional seconds of the start time, and the sampling interval.

It is often the case that part of the starting time can be inferred, but not all of it. Fractional seconds and sample interval are found most often, then units and tens of seconds, and lastly (as one group) the Julian day, hour, and minute. Any one of these not found by interpreting WWVB is simply set to respective values taken from the original AH header, with appropriate warnings sent to *stdout*. WWVB does not tell one the year, so that is taken from the original AH header or overridden by the -y option.

One of the sad discoveries made in writing *ahwwvb* is that current five-day playback procedures are very poor. Many files have varying digitizing rates, witnessed by varying intervals between WWVB "ticks", and some files have drastic fluctuations in rate. Based on my small test sample, these all appear to be increases in sample interval (WWVB appearing compressed) lasting on the order of 10 s and often occurring several times per minute. Presumably, the tape-speed compensation is not working properly and the recorder slows episodically (this mechanical dragging is known to be common near the terminal end of tapes, due to imperfections in the recorder design and, to a lesser degree, maintenance). Even this veteran observer failed to notice many of these fluctuations until looking at the tick intervals rather closely, so I have added some diagnostics to *ahwwvb*, *to wit* a statistical evaluation of tick intervals. This problem often is accompanied by a computed digitizing rate different from that requested. The work-around (other than *fixing* the tape-speed compensation!) is to check the time of a WWVB tick mark near each your picks, using *epick* to repick WWVB where necessary. Data digitized in real time from the networks are, as one expects, rock constant in sampling rate.

ahwwvb reports the value of variable *ires* if **-v** flag is set. *ires* means:

0 = total failure to decode WWVB,
1 = sample interval and fractional "sec" decoded successfully,
2 = units of "sec" decoded too,
3 = all of "sec" decoded,
(4 is not used)
5 = WWVB was fully decoded (Julian day, hour, and minute too)

ARGUMENTS

-m *min_rate* is the *lowest* sampling rate (samples per second) that this trace could possibly be. This figure is used to design a long running-median filter to clip off noise spikes shorter than 60% (**-r**) or 80% (**-s**) of the shortest valid signal pulse (0.2 s, in the case of WWVB). You should give the highest reasonable number but not overestimate, or valid data could be clipped and unreadable as well. You may disable filtering by giving any *min_rate* < 15 sps. I know of no good reason, aside from bad judgement in sampling rates, to override the filtering. If filtering is done, *ahwwvb* prefilters with a series of shorter RMF windows to improve edge fidelity. In any case, *ahwwvb* tells you what it did or did not do.

ah_file The (*xdr*) AH file containing the WWVB trace and possibly other traces.

OPTIONS

-vb *code* and *channel* identify the WWVB trace in *ah_file*. Defaults are "WWVB" and either "TIME", "time", or "T", respectively.

-v Cause verbose output that provides a trace of the processing done. The default is to print out only the summary message and any error messages. (Users with scripts designed to digest the output of early versions of *ahwwvb* should invoke this option.)

-la (Default) *All* traces in *ah_file* are changed.

-l *list_file* must contain one or more AH file names and trace sequence-numbers; only these traces are changed. The WWVB trace must be listed here (a check is made). *list_file* is in the same format as for *epick*.

-y *year* is the year of the first sample in the file. It overrides the year found in the original AH header.

-d *delay* is the total lag of the recorded WWVB signal behind UTC. It is typically 18±5 ms radio lag + the flight time. *delay* is given in ms (default 0). The start time is set earlier than the recorded WWVB signal by this amount.

-r

-s Is the WWVB trace "rounded" (**-r**, the default) or "sharp" (**-s**)? That is, does the trace appear to have passed through a low-pass filter, as is the case for the five-day recorder (the discriminators apply low pass filters to all traces). WWVB digitized by CUSP, for example, is more nearly raw, with apparently sharp transitions. However, WWVB received on a TrueTime® radio and digitized at 1000 sps on a PASSCAL recorder shows significantly rounded transitions, with about 5 ms from the start of the transition to the steepest slope.

The issue is where to infer that the original WWVB signal transitioned. For **-r**, it is assumed to be one sample interval before the steepest first difference; for **-s**, it is taken to be at the same sample interval as the steepest first difference. This distinction makes only a one-sample-interval difference. In all cases, the true transition time is assumed to be midway in the identified sample interval. It is preferable to use **-s** with **-d** to compensate explicitly when you know the time lag caused by filtering ("**-d** (*time_of_steepest_slope* − *time_of_actual_transition* + *any_other_lags*)").

-nc By default, *ahwwvb* updates the traces themselves, as described. This action can be suppressed by the **-nc** ("no change") flag, in which case, the result is still reported but nothing is done with it. The **-nc** option us useful for tests and to those who record clock corrections elsewhere (and apply them to results rather than to the original data).

SEE ALSO

epick—X11 *ah* picking routine derived and extended from Lamont's *sunpick*. *tcpick* invokes *epick* in a mode allowing one to manually read (and optionally correct to) a recorded time code like WWVB.

ahclk—called by *ahwwvb* to make the actual changes to AH headers.

Other *ah* filters.

BUGS

list_file must be sorted with all the traces for any particular file together and in ascending numerical order within each file. *mkahlist* does that for you. Someday, *ahclk* could be made smarter about this, but it hasn't been yet.

ahwwvb should be smart enough to add the WWVB trace ID to any *list_file*, but it isn't.

Uses inflagrevious FORTRAN spaghetti code of C. Johnson, so it is largely a mystery how the interpreter works (or doesn't). Watch out for errors (particularly 1-, 10-, and 60-s errors?).

ahwwvb cannot be used as a UNIX filter.

Either *ahwwvb* or *ahclk* should check relatedness for the -l option too.

Would be more accurate with a rawer WWVB. TrueTime® receivers try make the trace pretty by thresholding the signal to a running mean of the signal, thereby damaging the information contained (adding non-constant small delays to the transitions).

The -r transition bias is utterly empirical, set from years of experience almost exclusively with 100 sps five-day recorder data. It should at least be user-controllable or, better yet, knowledgeable of the filtering.

Should offer a way of holding the digitizing rate fixed, for the case of digital recorders.

ahwwvb is new and tested on only ten examples—**keep your eyes open!**

The filtering and L1 fit are noticeably slow at high sample rates. 2-minute 1000-sps records decode in about 7 minutes on a Sun 3/60.

ahwwvb has been used at 100, 200, and 1000 sps. It probably will work out to 2000 sps, but there are some poorly understood magic numbers in C. Johnson's code, and these may cause it to fail at some high rate. Of course, it will certainly fail below 5 sps because WWVB has meaningful pulses as short as 0.2 s.

AUTHOR

John R. Evans, USGS, Menlo Park, CA. Original written March 16, 1992. -nc flag added June 17, 1992. Made to work for high sampling rates (at least to 1000 sps) June 23, 1992.

```
/*
 * AH filter AHWWVB (AH WWVB reader):
 *    Read WWVB and change AH headers accordingly.  See "ahwwvb.man".
 *
 * dbxtool -I . -I /we/itch/evans/src/lib/rmf /we/itch/evans/usr.sun3/ahwwvb &
 * stop in median
 * when at "/we/itch/evans/src/lib/rmf/cmedian.c":143 { print iout,arrout[iout] ; }
 * when ( ( iout % 100 ) == 0) { print iout,arrout[iout] ; }
 * stop if ( iout == 15000 )
 * run -m 990 06.29.0192.4.ah -vb 0192 4 -nc
 */

#include <stdio.h>
#include <math.h>
#include <rpc/rpc.h>
#include <local/ahhead.h>
#include <local/stdtyp.h>
#include <local/date_time.h>

#define MIN(a,b)        (((a) < (b)) ? (a) : (b))       /* Smaller value */
#define MAX(a,b)        (((a) > (b)) ? (a) : (b))       /* Larger value */

#include "ahwwvb.h"

#define strneq  !strncmp
#define streq   !strcmp

        void            error();        /* -lq (error.c)                */
        void            report();       /* -lq (error.c)                */
        FILE            *efopen();      /* -lq (efopen.c)               */
        char            *emalloc();     /* -lq                          */
        char            *erealloc();    /* -lq                          */
        void            mnday();        /* -lq                          */
        bool            isleap();       /* -lq                          */
        TIME            timvar();       /* -lq                          */
        void            datime();       /* -lq                          */

        /*              xdr_gethead()       -lah                        */
        /*              mkdatspace()        -lah                        */
        /*              xdr_getdata()       -lah                        */

        int             read_wwvb();    /* Defined below                */
        TIME            ah2epoch();     /* Defined below                */
        int             seq_ok();       /* Defined below                */
        void            ch_files();     /* Defined below                */
        void            decode_WWVB();  /* Defined below                */
        void            tick_size();    /* Defined below                */
        static int      icomp();        /* Defined below                */
        int             is_wwvb();      /* Defined below                */
        void            to_bin();       /* Defined below                */
        static int      fcomp();        /* Defined below                */
        bool            is_max();       /* Defined below                */
        bool            is_min();       /* Defined below                */
        void            wwvb_();        /* wwvb.f                       */

        char    *progname;

main(argc, argv)
        int     argc;
        char    **argv;
{
        FILE            *ah_file;
        char            ah_name[NM_LEN]; /* Name of AH file with WWVB trace */
        XDR             xdr_in;

        int             n_traces;       /* Total traces in ah_file      */
        int             itrace;

        int             wwvb_seq;       /* Position of WWVB trace       */
        ahhed           wwvb_head;      /* Space for WWVB AH header     */
        float           *wwvb_trace;    /* Pointer to WWVB AH trace     */


        FILE            *list_file;     /* listfile a la "epick"        */
        char            list_name[NM_LEN];
        int             all_traces;     /* Over-ride listfile and change
                                           all traces */
        int             change_ah;      /* Actually change ah traces?   */
        int     yr, mo, day, hr, mn;
        double  sec;

        bool            verbose = FALSE; /* Write out gory details?     */

        FILE            *temp_file;     /* listfile a la "epick"        */
        char            temp_name[NM_LEN]; /* List file name actually
                                           passed to AHCLK */

        float           min_rate;       /* Minimum digitizing rate possible */
```

```
        int             RMF_win;        /* Resulting maximal RMF window length */

        char            wwvb_code[CODESIZE]; /* WWVB trace code         */
        char            wwvb_chan[CHANSIZE]; /* WWVB trace channel      */

        int             year = 0;       /* Year of trace; overrides ahheader */

        TIME            new_st;         /* New starting time            */
        double          new_int;        /* New sampling interval        */
        TIME            old_st;         /* Original starting time       */
        double          old_int;        /* Original sampling interval   */

        char            sys_str[STR_LEN]; /* Scratch string             */
        int             sysret;

        bool            rounded;        /* Is the trace "rounded" (filtered)? */
        TIME            lagg;           /* Set clock back by this ammount */

        /* Initialize */
        new_st = NULL;                  /* No-change flag               */
        new_int = -1.;                  /* No-change flag               */

        min_rate = -1.0;                /* Flag no-value-given          */
        RMF_win = -1;                   /* Flag no-value-given          */
        ah_name[0] = '\0';              /* Flag no-value-given          */

        all_traces = TRUE;              /* Default:  change all traces  */
        change_ah = TRUE;

        list_name[0] = '\0';
        list_file = (FILE *) NULL;

        (void)strcpy(wwvb_code, "WWVB");
        (void)strcpy(wwvb_chan, DEFAULT_CHAN);  /* == "TIME", "time", or "T" */

        rounded = TRUE;
        lagg = 0.0;

        /* Check usage */
        progname = argv[0];
        if (argc < 4)
                error("Usage:  %s -m min_rate ah_file [-vb code chan] [ -v ] [-la | -l listfile] [-y year] [-r | -s] [-d delay] [
-nc]",
                        argv[0]);

        /* Interpret command line arguments */
        for (--argc, argv++; argc; --argc, argv++) {

                /* Minimum sampling rate implies RMF window length */
                if (strneq(*argv, "-m", 2)) {
                        --argc, argv++;
                        if (sscanf((*argv), "%f", &min_rate) != 1)
                                error("Unreadable mimimum digitization rate.");
                }

                /* listfile name, or do all traces */
                else if (strneq(*argv, "-l", 2)) {
                        if (strneq(*argv, "-la", 3)) {
                                all_traces = TRUE;
                        }
                        else {
                                all_traces = FALSE;
                                --argc, argv++;
                                (void)strcpy(list_name, *argv);
                        }
                }

                /* Verbose output? */
                else if (strneq(*argv, "-v", 2) && !strneq(*argv, "-vb", 3)) {
                        printf("AHWWVB verbose output:\n");
                        verbose = TRUE;
                }

                /* WWVB code and channel names */
                else if (strneq(*argv, "-vb", 3)) {
                        --argc; argv++;
                        if (argc < 2)
                                error("Bad WWVB code or channel.");
                        (void)strcpy(wwvb_code, *argv);
                        --argc; argv++;
                        (void)strcpy(wwvb_chan, *argv);
                }

                /* new sampling interval */
                else if (strneq(*argv, "-y", 2)) {
                        --argc; argv++;
                        if ((sscanf((*argv), "%d", &year) != 1))
                                error("Bad year in -y option.");
```

```
		}

		/* "Rounded" or "sharp" WWVB trace? */
		else if (strneq(*argv, "-r", 2))
			rounded = TRUE;
		else if (strneq(*argv, "-s", 2))
			rounded = FALSE;

		/* Radio + flight time lag */
		else if (strneq(*argv, "-d", 2)) {
			--argc; argv++;
			if ((sscanf((*argv), "%lf", &lagg) != 1))
				error("Bad delay in -d option.");
			if (lagg < 0.0)
				error("NEGATIVE delay given--non-causal!");
		}

		/* Do not change traces */
		else if (strneq(*argv, "-nc", 3))
			change_ah = FALSE;

		/* WWVB-containing AH-file name */
		else
			(void)strcpy(ah_name, *argv);
	}

	/* Check for required arguments */
	if (ah_name[0] == '\0')
		error("No WWVB-containing AH file given (\"ah_file\").");
	if (min_rate < 0.0)
		error("No valid minimum sampling rate (\"min_rate\") given.");

	/* Modify RMF window length */
	if (rounded)
		RMF_win = 1 + 2 *
			((int)(0.2 * min_rate * RMF_FUDGE_r) - 2);
	else
		RMF_win = 1 + 2 *
			((int)(0.2 * min_rate * RMF_FUDGE_s) - 2);

	if (RMF_win < 3) {
		RMF_win = 0;
		if (verbose) {
			printf("Sampling interval too low;");
			printf(" no RMF filtering done.\n");
		}
	}
	else if (verbose)
		printf("Longest RMF window is of %d points.\n", RMF_win);

	/* Inform user */
	if (verbose)
		if (rounded)
			printf("Compensating for low-pass filtered WWVB.\n");
		else
			printf("Assuming unfiltered WWVB.\n");

	if (verbose && (lagg > 0.0))
		printf("Correcting for %f ms flight time + radio lag.\n", lagg);

	/* Get WWVB trace and header */
	ah_file = efopen(ah_name, "r");
	xdrstdio_create(&xdr_in, ah_file, XDR_DECODE);

	n_traces = read_wwvb(&xdr_in, wwvb_code, wwvb_chan,
		&wwvb_head, &wwvb_trace, &wwvb_seq, ah_name,
		all_traces, &old_st, &old_int);

	xdr_destroy(&xdr_in);
	(void)fclose(ah_file);

	/* Interpret WWVB */
	(void)decode_WWVB(rounded, year, min_rate, RMF_win,
		wwvb_head, wwvb_trace, &new_st, &new_int, old_st, old_int,
		verbose);
	new_st += (lagg / 1000.0);		/* Remove any delay	*/

	/* By default, change ah traces accordingly. */
	if (change_ah) {

		/* (-la) Create temporary list_file,
		   a complete list of traces in ah_file */

		if (all_traces) {
			(void)sprintf(temp_name, "%s.XXXXXX", ah_name);
			(void)mktemp(temp_name);

			temp_file = efopen(temp_name, "w");
```

```
                for (itrace = 1 ; itrace <= n_traces ; itrace++)
                        fprintf(temp_file, "%s %d\n", ah_name, itrace);

                (void)fclose(temp_file);
        }

        /* (-1) check that WWVB trace is in list_file
           and check sequence in listfile */

        else {
                if (list_name[0] == '\0')
                        error("Requires listfile name.");

                list_file = efopen(list_name, "r");

                rewind (list_file);
                if (!seq_ok(list_file, ah_name, wwvb_seq))
                        error("listfile out of sequence or WWVB trace not listed there.");

                (void)fclose(list_file);

                (void)strcpy(temp_name, list_name);
        }


                /* Spawn AHCLK to make the necessary changes in AH headers */

                ch_files(temp_name, TRUE, new_st, TRUE, new_int, verbose);

                /* Clean up and exit */

               if (all_traces) {
                        (void)sprintf(sys_str, "/bin/rm -f %s", temp_name);
                        if ((sysret = ((system(sys_str) >> 8) & 0377)) == 127)
                                report("Could not delete temporary list file.");
#if DEBUG
                        printf("/bin/rm returns %d.\n", sysret);
#endif DEBUG
                }
        }

        /* In all cases, report the result. */
        (void)datime(new_st, &yr, &mo, &day, &hr, &mn, &sec, GREGORIAN);

        printf("\n----------------------ahwwvb----------------------\n");
        printf("File \"%s\", code \"%s\", channel \"%s\":\n",
                ah_name, wwvb_code, wwvb_chan);
        printf("Start time = %04d/%02d/%02d  %02d:%02d:%07.4lf UTC\n",
                yr, mo, day, hr, mn, sec);
        printf("Sample interval = %.10lf s (%lf sps)\n",
                new_int, 1. / new_int);
        printf("--------------------------------------------------\n\n");

        /* Clean up and quit */
        free((char *)wwvb_trace);
        exit(0);
}

/*
 * Check sequence in listfile and presence of WWVB.
 * If either is awry, return FALSE.
 */

typedef struct {
        char            fname_in[NM_LEN];
        int             rec_num;
        int             ugid;
} TLIST;

int
seq_ok(list_file, ah_name, wwvb_seq)
        FILE            *list_file;
        char            *ah_name;
        int             wwvb_seq;
{
        char            in_str[STR_LEN];        /* One line from list file */

        int             i;
        int             same_nm;
        int             seq;
        int             rv;
        int             vb;

        int             nlines;
        TLIST           *file_list[MX_LST];     /* known AH-file names  */

        rv = TRUE;
```

```
        vb = FALSE;
        nlines = 0;
        file_list[0] = (TLIST *)emalloc(sizeof(TLIST));

        while (fgets(in_str, sizeof(in_str), list_file) != NULL) {
                if(sscanf(in_str, "%s%d%d",
                        file_list[nlines]->fname_in,
                        &(file_list[nlines]->rec_num),
                        &(file_list[nlines]->ugid))
                        >= 2) {

                        if (streq(file_list[nlines]->fname_in, ah_name)
                                && (file_list[nlines]->rec_num == wwvb_seq))
                                vb = TRUE;

                        same_nm = TRUE;
                        seq = file_list[nlines]->rec_num;
                        for(i = nlines ; i >= 0 ; --i) {
                                if (streq(file_list[nlines]->fname_in,
                                        file_list[i]->fname_in)) {
                                        if (!same_nm)
                                                rv = FALSE;
                                        if (file_list[i]->rec_num >= seq
                                                && i < nlines)
                                                rv = FALSE;
                                        else
                                                seq = file_list[i]->rec_num;
                                }

                                else {
                                        same_nm = FALSE;
                                }
                        }

                        nlines++;
                        if (nlines >= MX_LST)
                                error("listfile too long.");
                        file_list[nlines] = (TLIST *)emalloc(sizeof(TLIST));
                }
                else
                        error("Bad listfile format.");
        }

        for (i = 0 ; i <= nlines ; i++)
                free((char *)file_list[i]);

        return (rv && vb);
}

/*
 * Call AHCLK to change the AH file headers.
 */

void
ch_files(temp_name, start, new_st, interval, new_int, verbose)
        char    *temp_name;     /* List file name actually passed to AHCLK */
        int     start;          /* Change start time if TRUE            */
        TIME    new_st;         /* Recalculated start time              */
        int     interval;       /* Change sampling interval if TRUE     */
        double  new_int;        /* Recalculated sampling interval       */
        bool    verbose;        /* Write out gory details?      */
{
        int     yr, mo, day, hr, mn;
        double  sec;
        char    sys_str[STR_LEN];
        char    add_str[STR_LEN];
        int     sysret;

        (void)sprintf(sys_str, "ahclk -l %s", temp_name);
        if (start) {
                (void)datime(new_st, &yr, &mo, &day, &hr, &mn, &sec, GREGORIAN);
                (void)sprintf(add_str, " -s %04d %02d %02d %02d %02d %1f",
                        yr, mo, day, hr, mn, sec);
                (void)strcat(sys_str, add_str);
        }
        if (interval) {
                (void)sprintf(add_str, " -i %.10lf", new_int);
                (void)strcat(sys_str, add_str);
        }

        /* Report change */
        if (verbose) {
                printf("\nChanging start time and interval ...\n", sys_str);
                printf("\t%s\n\t", sys_str);
        }

        if ((sysret = ((system(sys_str) >> 8) & 0377)) == 127)
                report("Trouble changing AH files.  CONDITION OF FILES UNKNOWN!");
```

```
        else if (verbose)
                printf("... successful.\n\n", sys_str);

#if DEBUG
        printf("AHCLK returns %d.\n", sysret);
#endif DEBUG

}

/*
 * Read all trace and header pairs.  Return the WWVB trace and header,
 * its sequence (position) within the AH file, and the total number of
 * traces in the AH file.  Allocates space for wwvb_trace, which must
 * be released elsewhere.  If "-la" option is in force, make relatedness
 * check.
 */

int
read_wwvb(xdr_in, wwvb_code, wwvb_chan, wwvb_head, wwvb_trace,
        wwvb_seq, ah_name, all_traces, old_st, old_int)
        XDR             *xdr_in;
        char            *wwvb_code;     /* WWVB trace code              */
        char            *wwvb_chan;     /* WWVB trace channel           */
        ahhed           *wwvb_head;     /* Space for WWVB AH header     */
        float           **wwvb_trace;   /* Pointer to WWVB AH trace     */
        int             *wwvb_seq;      /* Position of WWVB trace       */
        char            *ah_name;       /* Name of AH file with WWVB trace */
        int             all_traces;     /* TRUE if "-la"                */
        TIME            *old_st;        /* Return original starting time */
        double          *old_int;       /* and sampling interval        */
{
        ahhed           temp_head;      /* Space for arbitrary AH header */
        float           *temp_trace;    /* Pointer to WWVB AH trace     */
        int             n_traces = 0;   /* Total traces in ah_file      */

        bool            first = TRUE;   /* Flag first trace             */
        bool            GOT_IT = FALSE; /* Flag whether WWVB found at all */

        long            t_length;       /* Trace length                 */
        TIME            t_start;        /* Original trace start time    */
        TIME            t_temp;         /* Original trace start time    */
        float           t_int;          /* Original trace sampling interval */

        while (TRUE) {

                /* Read header */
                if (xdr_gethead(&temp_head, xdr_in) != 1)
                        break;

                /* Relatedness check */
                if (all_traces) {
                        if (first) {
                                first = FALSE;
                                t_length = temp_head.record.ndata;
                                t_start = ah2epoch(temp_head.record.abstime);
                                t_int = temp_head.record.delta;
                        }

                        else {
                                t_temp = ah2epoch(temp_head.record.abstime);

                                if (temp_head.record.ndata != t_length
                                        || t_temp != t_start
                                        || temp_head.record.delta != t_int)

                                        error("Traces not all related!");
                        }
}

/* Successful */
n_traces++;

/* Allocate trace storage */
temp_trace = (float *)mkdatspace(&temp_head);

/* Read trace */
if (xdr_getdata(&temp_head, (char *)temp_trace, xdr_in) == -1)
        error("Error reading data record.");

/* Save WWVB when found */
if (is_wwvb(temp_head, wwvb_code, wwvb_chan)) {
        *wwvb_head = temp_head;
        *wwvb_trace = temp_trace;
        *wwvb_seq = n_traces;

        /* Pass back originals */
        *old_st = ah2epoch(temp_head.record.abstime);
        *old_int = temp_head.record.delta;
```

```
                        GOT_IT = TRUE;
                }

                else
                        free((char *)temp_trace);
        }

        if (GOT_IT)
                return(n_traces);
        else
                error("No WWVB trace found in %d trace(s) in \"%s\".",
                        n_traces, ah_name);
}

TIME
ah2epoch(ah_t)
        struct ah_time  ah_t;           /* Given an AH-header time ...  */
                                        /* ... return epochal time      */
{
        return(timvar((int)ah_t.yr, (int)ah_t.mo, (int)ah_t.day,
                (int)ah_t.hr, (int)ah_t.mn, (double)ah_t.sec,
                GREGORIAN));
}

/*
 * Determine whether this trace is a WWVB trace.
 */

int
is_wwvb(temp_head, wwvb_code, wwvb_chan)
        ahhed           temp_head;      /* Space for arbitrary AH header */
        char            *wwvb_code;     /* WWVB trace code               */
        char            *wwvb_chan;     /* WWVB trace channel if ==
                                           DEFAULT_CHAN, recognizes both
                                           "TIME" and "T" (note caps)    */
{
        if (streq(temp_head.station.code, wwvb_code)) {
                if (streq(wwvb_chan, DEFAULT_CHAN)) {
                        if (streq(temp_head.station.chan, "TIME")
                                || streq(temp_head.station.chan, "time")
                                || streq(temp_head.station.chan, "T"))
                                return (TRUE);
                        else
                                return (FALSE);
                }

                else {
                        if (streq(temp_head.station.chan, wwvb_chan))
                                return (TRUE);
                        else
                                return (FALSE);
                }
        }

        else
                return (FALSE);
}

/*
 * Massage and decode WWVB trace, the latter by calling C. Johnson's
 * routines.  Report the signal processing and any fudges needed.  Compile
 * and report statistics on lengths of WWVB "ticks" to reveal inconstant
 * digitizing rates.
 */

void
decode_WWVB(rounded, year, min_rate, RMF_win, wwvb_head, wwvb_trace,
        new_st, new_int, old_st, old_int, verbose)

        bool            rounded;        /* Is the trace "rounded" (filtered)? */
        int             year;           /* Year; overrides ahheader if != 0 */
        ahhed           wwvb_head;      /* WWVB AH header               */
        float           *wwvb_trace;    /* Pointer to WWVB AH trace     */
        TIME            *new_st;        /* New starting time            */
        double          *new_int;       /* New sampling interval        */
        float           min_rate;       /* Minimum digitizing rate possible */
        int             RMF_win;        /* Resulting maximal RMF window length */
        TIME            old_st;         /* Original starting time       */
        double          old_int;        /* Original sampling interval   */
        bool            verbose;        /* Write out gory details?      */
{
        int             ns;             /* Number of samples in time code */
        int             strt;           /* Starting point in time code  */
        int             *ks;            /* Record to decode     */
        float           step;           /* Sampling rate        */
        int             ires;           /* Decoding "score" (synch. level) */
```

```
        /* Time of first sample as returned by WWVB routine of C. Johnson */
        long            kda;            /* ("Julian") day of year       */
        long            khr;            /* Hour of day                  */
        long            kmn;            /* Minute of hour               */
        float           sec;            /* Seconds past minute          */

        double          threshold;      /* Threshold (mean of trace) for
                                           reducing float to binary (int) */
        int             ii, jj;         /* Loop dummies                 */
        int             RMF_half;       /* Half-width RMF window length */

        /* Variables for epochal time translation */
        TIME            tpt;
        int             tpyr, tpmo, tpdy, tphr, tpmn;
        double          tpsec;

#if DEBUG
        for (ii = 0 ; ii < wwvb_head.record.ndata ; ii++)
                printf("%f\n", wwvb_trace[ii]);
#endif DEBUG

        /* Despike with long-window running-median filter (maximal
           filter sequence would be 3, 5, 7, 9, ..., RMF_win,
           which is slow but not prohibative.) */

        if (RMF_win >= 3) {

                /* First kill single-point spikes, which are common
                   and can draw edges toward them by one sample */
                if (RMF_win > 3) {
                        if (verbose)
                                printf("Running RMF prefilter of 3 points ...\n");
                        (void)median(wwvb_trace, wwvb_trace,
                                (int)wwvb_head.record.ndata, 3, FALSE);
                }

                /* And kill two-point spikes for the same reason */
                if (RMF_win > 5) {
                        if (verbose)
                                printf("Running RMF prefilter of 5 points ...\n");
                        (void)median(wwvb_trace, wwvb_trace,
                                (int)wwvb_head.record.ndata, 5, FALSE);
                }

                /* And do a half window for really long final windows */
                RMF_half = 1 + 2 * (RMF_win / 4);
                if (RMF_half > 9) {
                        if (verbose)
                                printf("Running RMF prefilter of %d points ...\n",
                                        RMF_half);
                        (void)median(wwvb_trace, wwvb_trace,
                                (int)wwvb_head.record.ndata, RMF_half, FALSE);
                }

                /* Run the near-maximal RMF filter */
                if (verbose)
                        printf("Running main RMF filter of %d points ...\n",
                                RMF_win);
                (void)median(wwvb_trace, wwvb_trace,
                        (int)wwvb_head.record.ndata, RMF_win, FALSE);

#if DEBUG
                for (ii = 0 ; ii < wwvb_head.record.ndata ; ii++)
                        printf("%f\n", wwvb_trace[ii]);
#endif DEBUG

                /* Trim unfiltered ends */
                ns = wwvb_head.record.ndata - RMF_win + 1;
                strt = (RMF_win - 1) / 2;
        }

        else {
                /* Use entire (unfiltered) trace */
                ns = wwvb_head.record.ndata;
                strt = 0;
        }

        /* Interpret the trace into binary */

        ks = (int *)emalloc(sizeof(int) * ns);
        (void)to_bin(rounded, min_rate, wwvb_head.record.ndata,
                strt, ns, wwvb_trace, ks);

#if DEBUG
        for (ii = 0 ; ii < strt ; ii++)
                printf("-1\n");
        for (jj = 0 ; jj < ns ; jj++)
                printf("%d\n", ks[jj]);
```

```
        for (ii = 0 ; ii < strt ; ii++)
                printf("-1\n");
#endif DEBUG

        /* Decode WWVB with C. Johnson's FORTRAN spaghetti */
        if (verbose)
                printf("Interpreting time series ...\n");
        wwvb_ (&ns, ks, &step, &kda, &khr, &kmn, &sec, &ires);
        if (ires <= 0)
                if (kda == -1)
                        error("ELSKEW apparantly in infinite loop.  Giving up.");
                else
                        error("WWVB was completely unreadable (ires=%d).",
                                ires);

        /* Since ires > 0, new sampling interval OK.  Verify and report it. */
        if ((min_rate > 0.0) && (1. / step) < min_rate) {
                fprintf(stderr, "New sampling interval is lower");
                fprintf(stderr, " than -m argument claimed.\n");
                error("Filtering may have failed; files NOT changed.");
        }
        else
                *new_int = step;

        if (verbose && (*new_int != old_int))
                printf("Sampling rate changed from %.3f to %.3f sps.\n",
                        1. / old_int, 1. / (*new_int));

/* Compile and report statistics on the duration of WWVB "ticks" */
tick_size(ns, ks, step, verbose);

/* If partly decoded, fill in any unknowns with old values */
/* ires:                                                   */
/* 0 = total failure to decode                             */
/* 1 = fractional "sec" OK                                 */
/* 2 = units of "sec" OK too                               */
/* 3 = all of "sec" OK                                     */
/* 5 = fully decoded (Jday, hour, and minute OK too)       */

(void)datime(old_st,
        &tpyr, &tpmo, &tpdy, &tphr, &tpmn, &tpsec, GREGORIAN);

if (ires < 5) {
        printf("WWVB was only partly readable (ires=%d):\n", ires);

        printf("\tDefaulting to original month (%d), day (%d),",
                tpmo, tpdy);
        printf(" hour (%d), and minute (%d).\n", tphr, tpmn);

        /* If part of seconds unknown, default them to old values */
        if (ires < 3) {
                double  tens, units, fracs;

                if (sec < 0.0 || tpsec < 0.0)
                        error("Programming bug 1.");

                if (ires == 1) {
                        units = aint(tpsec);
                        fracs = sec - aint((double)sec);
                        tpsec = units + fracs;

                        printf("\tUsing WWVB fractional seconds");
                        printf(" (%.3f) with original units", fracs);
                        printf(" of seconds (%.0f s).\n", units);
                }

                else if (ires == 2) {
                        tens = 10. * aint(tpsec / 10.);
                        fracs = (sec / 10.);
                        fracs = 10. * (fracs - aint(fracs));
                        tpsec = tens + fracs;

                        printf("\tUsing WWVB units and fractional");
                        printf(" seconds (%.3f) with original", fracs);
                        printf(" tens of seconds (%.0f s).\n", tens);
                }

                else
                        error("Programming bug 2.");
        }

        else if (ires == 3)
                tpsec = sec;

        else
                error("Programming bug 3.");
}
```

```
        /* Decide what year to use.  If 0 (the default),
           use the original trace starting year) */
        if (year > 0) {
                if (verbose)
                        if (year != tpyr)
                                printf("Changing year to %d.\n", year);
                        else
                                printf("Keeping year at %d.\n", year);
                tpyr = year;
        }
        else if (verbose)
                printf("Defaulting year to %d.\n", tpyr);

        /* If fully decoded, use WWVB date and time */
        if (ires == 5) {
                /* Translate J-day to month and day */
                mnday(kda, isleap(tpyr, GREGORIAN), &tpmo, &tpdy);

                tphr = khr;
                tpmn = kmn;
                tpsec = sec;
        }

        /* Set start time back to beginning of wwvb_trace */
        tpsec -= (float)strt * step;

        /* Take second ticks to be half way between the 0 and 1
           samples of the binary trace */
        tpsec -= 0.5 * step;

        /* Resolve and store new starting time via epochal time. */
        *new_st = timvar(tpyr, tpmo, tpdy, tphr, tpmn, tpsec, GREGORIAN);

        /* Clean up */
        free((char *)ks);
}

/*
 * Compile and report statistics on WWVB "tick" lengths.
 */

void
tick_size(ns, ks, step, verbose)
        int             ns;             /* Number of samples in time code */
        int             *ks;            /* Record to decode               */
        float           step;           /* Sampling rate                  */
        bool            verbose;        /* Write out gory details?        */
{
        int             wdth[MAXTRANS]; /* Tick widths                    */
        int             n_wdth;         /* How many there are             */
        int             ii;             /* Loop dummy                     */
        bool            first;          /* Flag first transition          */
        int             last;           /* Previous tick location         */
        float           span;           /* Range of intervals             */

        /* Gather tick widths */
        n_wdth = 0;
        first = TRUE;
        for (ii = 0 ; ii < ns - 1 ; ii++) {
                if ((ks[ii + 1] - ks[ii]) == 1) {       /* Up tick      */
                        if (first) {
                                last = ii;
                                first = FALSE;
                        }

                        else {
                                wdth[n_wdth] = ii - last;
                                last = ii;
                                n_wdth++;
                        }
                }
        }

        /* Sort widths */
        (void)qsort((char *)wdth, n_wdth, sizeof(int), icomp);

        /* Report various ranges and deviations */

        span = (step * wdth[nint(0.95 * (double)(n_wdth - 1))])
                - (step * wdth[nint(0.05 * (double)(n_wdth - 1))]);

        if (verbose || (span > SPAN_WARN * step)) {
                printf("\nSecond intervals range from %.3f to %.3f s.\n",
                        step * wdth[0], step * wdth[n_wdth - 1]);

                if (span > SPAN_WARN * step)
                        printf("WARNING:  ");
                printf("5th and 95th percentiles differ by %.3f s.\n\n", span);
```

```
        }
}

/*
 * Comparison rule for qsort
 */

static int
icomp(a, b)
        int     *a, *b;
{
        if (*a > *b)
                return (1);
        else if (*a < *b)
                return (-1);
        else
                return (0);
}

/*
 * Interpret WWVB code into binary by finding transition points.
 *
 * If "rounded==TRUE", the transition is taken to happen 1 sample before
 * the interval with the largest slope.  This empiricle artifice accounts
 * for low-pass filtering of WWVB by the playback discriminator, and is
 * about right near 100 sps.  If "rounded==FALSE", the interval with the
 * largest slope is taken to be the interval with the transition.  This
 * assumption is appropriate for CUSP data, and probably much other.
 *
 * If the difference peak has a flat top, binary output transitions at
 * the earliest part of peak.
 *
 * This routine CHANGES the input trace (wwvb_trace).
 */

void
to_bin(rounded, min_rate, npts, strt, ns, wwvb_trace, ks)
        bool            rounded;        /* Is the trace "rounded" (filtered)? */
        float           min_rate;       /* Minimum digitizing rate possible */
        long            npts;           /* Number of samples in input code */
        int             ns;             /* Number of samples in output code */
        int             strt;           /* Starting point of output within
                                           the input array              */
        float           *wwvb_trace;    /* Pointer to WWVB AH trace     */
        int             *ks;            /* Record to decode     */
{
        float           *mins, *maxs;   /* Arrays of mins and maxs      */
        int             nmins, nmaxs;   /* Number of mins and maxs      */
        float           min_th, max_th; /* Thresholds for mins and maxs */

        int             scan_win;       /* Window length for "global" extreema */
        int             ii, jj, kk;     /* Loop dummies                 */
        bool            first;          /* Flag first transition        */
        int             new_val, old_val; /* Used in binary translator  */

        int             bias;           /* Put transitions of the binary
                                           trace this many sample intervals
                                           from the largest-first-difference
                                           interval                     */

        /* Initialize */
        mins = (float *)emalloc(sizeof(float));
        nmins = 0;
        maxs = (float *)emalloc(sizeof(float));
        nmaxs = 0;

        if (rounded)
                bias = -1;
        else
                bias = 0;

        /* Take first difference of input
           (points contain the following-interval first difference) */
        for (ii = 0 ; ii < npts - 1 ; ii++)
                wwvb_trace[ii] = wwvb_trace[ii + 1] - wwvb_trace[ii];
        wwvb_trace[ii] = wwvb_trace[ii - 1];    /* kludge */


        /* To determine appropriate thresholds in the first difference,
           first collect all local mins and max's */
        for (ii = strt + 1, jj = 1 ; jj < ns - 1 ; ii++, jj++) {

                if (wwvb_trace[ii] > wwvb_trace[ii - 1]
                        && wwvb_trace[ii] > wwvb_trace[ii + 1]) {

                        maxs[nmaxs] = wwvb_trace[ii];
                        nmaxs++;
                        maxs = (float *)erealloc((char *)maxs,
```

```
                        sizeof(float) * (nmaxs + 1));
                }

                if (wwvb_trace[ii] < wwvb_trace[ii - 1]
                        && wwvb_trace[ii] < wwvb_trace[ii + 1]) {

                        mins[nmins] = wwvb_trace[ii];
                        nmins++;
                        mins = (float *)erealloc((char *)mins,
                                sizeof(float) * (nmins + 1));
                }
        }

        if (nmaxs <= 0)
                error("No local maxima found in first difference.");
        if (nmins <= 0)
                error("No local minima found in first difference.");

        /* Use TH_FUDGE * the TH_PER'th percentile as the threshold */

        (void)qsort((char *)maxs, nmaxs, sizeof(float), fcomp);
        max_th = TH_FUDGE * maxs[(int)((float)nmaxs * TH_PER / 100.)];

        (void)qsort((char *)mins, nmins, sizeof(float), fcomp);
        min_th = TH_FUDGE * mins[(int)((float)nmins * (1.0 - TH_PER / 100.))];


        /* Find transition points and translate to binary */

        /* Find the "global" peaks by scanning for the min/max a window */
        scan_win = MAX(1, 0.1 * min_rate); /* May fail at low sampling rates */

        /* Value prior to first transition not known at beginning */
        first = TRUE;
        for (ii = strt, jj = 0 ; jj < ns - (1 + bias) ; ii++, jj++) {
                if (is_max(wwvb_trace, ii, npts, scan_win, max_th)) {
                        new_val = 1;
                        old_val = 0;
                        ks[jj + 1 + bias] = new_val;

                        if (first) {
                                for (kk = 0 ; kk < jj + 1 + bias ; kk++)
                                        ks[kk] = old_val;
                                first = FALSE;
                        }
                }

                else if (is_min(wwvb_trace, ii, npts, scan_win, min_th)) {
                        new_val = 0;
                        old_val = 1;
                        ks[jj + 1 + bias] = new_val;

                        if (first) {
                                for (kk = 0 ; kk < jj + 1 + bias ; kk++)
                                        ks[kk] = old_val;
                                first = FALSE;
                        }
                }

                else if (!first)
                        ks[jj + 1 + bias] = new_val;
                else if (!first)
                        ks[jj + 1 + bias] = new_val;
                }
        }

        /*
         * Comparison rule for qsort
         */

        static int
        fcomp(a, b)
                float   *a, *b;
        {
                if (*a > *b)
                        return (1);
                else if (*a < *b)
                        return (-1);
                else
                        return (0);
        }

        /*
         * Find "global" maximum.
         */

        bool
        is_max(wwvb_trace, ii, npts, scan_win, max_th)
                float           *wwvb_trace;    /* Pointer to WWVB AH trace     */
                int             ii;             /* Index to wwvb_trace          */
```

```
        long            npts;           /* Number of samples in input code */
        int             scan_win;       /* Window length for "global" extreema */
        float           max_th;         /* Threshold                       */
{
        int             left, right;    /* Window indicies                 */
        int             jj;             /* Loop dummy                      */

        /* If not a local maximum (or flat), it cannot be a global maximum */
        if (wwvb_trace[ii] < wwvb_trace[MAX(0, ii - 1)]
                || wwvb_trace[ii] < wwvb_trace[MIN(npts, ii + 1)])
                return (FALSE);

        /* Is it the largest point in window? */
        left = MAX(0, ii - (int)(scan_win / 2));
        right = MIN(npts, ii + (int)(scan_win / 2));
        for (jj = left ; jj < right ; jj++)
                if (wwvb_trace[jj] > wwvb_trace[ii])
                        return (FALSE);

        /* If the point at ii IS a local maximum (or flat),
           IS (one of) the largest point(s) in the window, and
           IS greater than the threshold,
           it is a "global maximum" */
        if (wwvb_trace[ii] >= max_th)
                return (TRUE);
        else
                return (FALSE);
        else
                return (FALSE);
}

/*
 * Find "global" minimum.
 */

bool
is_min(wwvb_trace, ii, npts, scan_win, min_th)
        float           *wwvb_trace;    /* Pointer to WWVB AH trace     */
        int             ii;             /* Index to wwvb_trace          */
        long            npts;           /* Number of samples in input code */
        int             scan_win;       /* Window length for "global" extreema */
        float           min_th;         /* Threshold                       */
{
        int             left, right;    /* Window indicies                 */
        int             jj;             /* Loop dummy                      */

        /* If not a local minimum (or flat), it cannot be a global minimum */
        if (wwvb_trace[ii] > wwvb_trace[MAX(0, ii - 1)]
                || wwvb_trace[ii] > wwvb_trace[MIN(npts, ii + 1)])
                return (FALSE);

        /* Is it the smallest point in window? */
        left = MAX(0, ii - (int)(scan_win / 2));
        right = MIN(npts, ii + (int)(scan_win / 2));
        for (jj = left ; jj < right ; jj++)
                if (wwvb_trace[jj] < wwvb_trace[ii])
                        return (FALSE);

        /* If the point at ii IS a local minimum (or flat),
           IS (one of) the smallest point(s) in the window, and
           IS less than the threshold,
           it is a "global minimum" */
        if (wwvb_trace[ii] <= min_th)
                return (TRUE);
        else
                return (FALSE);
}
```

```
/*
 * AH filter AHWWVB (AH WWVB reader) include file.
 */

#define NM_LEN  128     /* Maximum file-name length             */
#define STR_LEN 256     /* Maximum scratch-string length        */
#define MX_LST  1024    /* Maximum line count of listfile       */

/* Recognize WWVB channel names "TIME", "time", or "T" */

#define DEFAULT_CHAN    "DFult"

/* Reduce RMF window by this factor
   to account for slow transitions in the reciever */

#define RMF_FUDGE_r     0.60
#define RMF_FUDGE_s     0.80

/* Use TH_FUDGE * the TH_PER'th percentile as the threshold
   determining what a "sharp corner" is (for interpreting trace
   into binary form). */

#define TH_FUDGE        0.5
#define TH_PER          98.

/* How many WWVB "ticks" to compile statistics on */
#define MAXTRANS        2000

/* A warning is issued if WWVB "ticks" (95th - 5th percentile)
   vary by more than this many sample intervals */
#define SPAN_WARN       5.0
```

```
/*      structure for data file header  --      witte, 11 June 85       */

#define AHHEADSIZE 1024
#define CODESIZE 6
#define CHANSIZE 6
#define STYPESIZE 8
#define COMSIZE 80
#define TYPEMIN 1
#define TYPEMAX 6
#define LOGSIZE 202
#define LOGENT 10
#define NEXTRAS 21
#define NOCALPTS 30

typedef struct  {
                float   x;
                float   y;
                } vector;

typedef struct  {
                float   r;
                float   i;
                } complex;

typedef struct  {
                double r;
                double i;
                } d_complex;

typedef struct  {
                float   xx;
                float   yy;
                float   xy;
                } tensor;

struct  ah_time {
                short           yr;     /* year         */
                short           mo;     /* month        */
                short           day;    /* day          */
                short           hr;     /* hour         */
                short           mn;     /* minute       */
                float           sec;    /* second       */
                };

struct  calib   {
                complex         pole;   /* pole         */
                complex         zero;   /* zero         */
                };

struct  station_info    {
                char            code[CODESIZE]; /* station code         */
                char            chan[CHANSIZE]; /* lpz,spn, etc.        */
                char            stype[STYPESIZE];/* wwssn,hglp,etc.     */
                float           slat;           /* station latitude     */
                float           slon;           /*    "    longitude    */
                float           elev;           /*    "    elevation    */
                float           DS;     /* gain */
                float           A0;     /* normalization */
                struct  calib   cal[NOCALPTS];  /* calibration info     */
                };

struct  event_info      {
                float           lat;            /* event latitude       */
                float           lon;            /*   "   longitude      */
                float           dep;            /*   "   depth          */
                struct  ah_time ot;             /*   "   origin time    */
                char            ecomment[COMSIZE];      /*      comment line    */
                };

struct  record_info     {
                short           type;   /* data type (int,float,...)    */
                long            ndata;  /* number of samples            */
                float           delta;  /* sampling interval            */
                float           maxamp; /* maximum amplitude of record  */
                struct  ah_time abstime;/* start time of record section */
                float           rmin;   /* minimum value of abscissa    */
                char            rcomment[COMSIZE];      /* comment line         */
                char            log[LOGSIZE]; /* log of data manipulations */
                };

typedef struct {
                struct  station_info    station;        /* station info */
                struct  event_info      event;          /* event info   */
                struct  record_info     record;         /* record info  */
                float           extra[NEXTRAS]; /* freebies */
                } ahhed;
```

```
#define FLOAT   1
#define COMPLEX 2
#define VECTOR  3
#define TENSOR  4
#define DOUBLE  6
#define INTEGER 7
```

```
c
c-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
c
c-----elskew : l1 solution to n equations in m unknowns
c
      subroutine elskew (m,n,at,d,gu,gd,small,x)
c
c     Note : transpose form
c
c     find x(i) to minimize:
c
c              n                 m
c     esum = sum(skewnorm(k, sum (d(k)-at(i,k)*x(i)) ))
c             k=1               i=1
c
c                              ( gu(k), if (er.gt.+small) )
c     skewnorm(k,er) = er * (  0.0  , if error is small  )
c                              ( gd(k), if (er.lt.-small) )
c
c-----Note:  This routine has been modified to accept the transpose of
c            the 'a' matrix.  This was done because it is more convenient
c            to set up a packed transpose when one does not know
c            the number of equations a priori.
c                                 -Carl
c
      real at(m,n),x(m),d(n),gu(n),gd(n)
      real w(2000),f(2000),b(4,4),col(4),row(4)
      real sc(4),g(4),gp(4),gm(4)
      real hit,small,wt,oldk,gr,tk,t,esum
c
      integer k(2000)
      integer i,j,k,l,m,n,kick,loop,ml,mh,new
c
c-----Also change value in format statement 160
      integer*4 looped,MXLOOP
      parameter (MXLOOP=100000)
c
      if (m.lt.5.and.n.lt.501) goto 5
      return
c
 5    looped = 0
c
      do 30 j=1,m
         x(j)=0.
         do 10 i=1,m
 10      b(i,j)=0.
         sc(j)=0.
         do 20 i=1,n
            sc(j)=sc(j)+abs(at(j,i))
 20      continue
         b(j,j)=n/sc(j)
 30   continue
c
      do 40 i=1,n
         f(i)=d(i)
         k(i)=i
 40   continue
      loop=0
 50   loop=loop+1
      kick=1+mod(loop-1,m)
      if (loop.le.m) goto 190
c
c-----find best equation to -kick- out of basis
      do 110 i=1,m
         gp(i)=0.
         gm(i)=0.
         g(i)=0.
 110  continue
c
      do 135 l=1,n
         if (abs(f(l)).lt.small) goto 120
         if (f(l).gt.small) hit=gu(l)
         if (f(l).lt.small) hit=gd(l)
         do 115 j=1,m
            g(j)=g(j)-at(j,l)*hit
 115     continue
         goto 135
c
 120     do 130 i=1,m
            wt=0.
            do 125 j=1,m
               wt=wt+at(j,l)*b(j,i)
 125        continue
            if (wt.lt.0.) gp(i)=gp(i)-gu(l)*wt
            if (wt.gt.0.) gp(i)=gp(i)-gd(l)*wt
            if (wt.gt.0.) gm(i)=gm(i)-gu(l)*wt
            if (wt.lt.0.) gm(i)=gm(i)-gd(l)*wt
c
```

```
 130     continue
 135  continue
c
      oldk=0.
      do 150 i=1,m
         gr=0.
         do 140 j=1,m
            gr=gr+g(j)*b(j,i)
 140     continue
         if ((gr+gp(i))*(gr+gm(i)).lt.0.) goto 150
         tk=min(abs(gr+gp(i)),abs(gr+gm(i)))
         if (tk.gt.oldk) kick=i
         if (tk.gt.oldk) oldk=tk
 150  continue
c
      if (oldk.eq.0.) return
c
c-----Give up if seem to be in infinite loop.
c     Message this with zero vector (not adequate in general!)
c
      looped = looped + 1
      if (looped.gt.MXLOOP) then
         do 160 j=1,m
            x(j) = 0.0
  160    continue
         return
      endif
c
 190  continue
c
c-----find scalar t where x=x0+(col of b)*t
      do 60 i=1,n
      w(i)=0.
      do 60 j=1,m
         w(i)=w(i)+at(j,i)*b(j,kick)
 60   continue
      call skewer(2000,n,w,f,gu,gd,small,k,t,ml,mh)
c
c-----pick out new basis
      new=k(ml)
      do 70 l=ml,mh
         if (abs(w(new)).lt.abs(w(k(l)))) new=k(l)
 70   continue
      t=f(new)/w(new)
      esum=0.
      do 80 i=1,n
         f(i)=f(i)-w(i)*t
         if (f(i).gt. 0.) esum=esum+gu(i)*f(i)
         if (f(i).lt. 0.) esum=esum+gd(i)*f(i)
 80   continue
c
c-----update x and basis matrix as descrived(transposed) in hadley p.
      do 90 j=1,m
         col(j)=b(j,kick)
         x(j)=x(j)+col(j)*t
         row(j)=0.
         do 85 i=1,m
            row(j)=row(j)-at(i,new)*b(i,j)/w(new)
 85      continue
 90   continue
      row(kick)=1./w(new)-1.
      do 100 i=1,m
         do 95 j=1,m
            b(i,j)=b(i,j)+col(i)*row(j)
 95      continue
 100  continue
      goto 50
c
      end
c
c-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
c
      subroutine skewer (nd,n,w,f,gu,gd,small,k,t,ml,mh)
c
c     solve rank 1 overdetermined equations with skew norm
c
c     inputs- n,w,f,u,d,small,k.   outputs- k,t,ml,mh.
c
c     find  t  to minimize
c
c                n
c     ls  =     sum skewnorm(k,f(k)-w(k)*t)
c               k=1
c
c     where:
c                        ( gu(k)*(er-small) if er.gt.+small      gu.gt.0
c     skewnorm(k,er)  =  ( gd(k)*(er+small) if er.lt.-small      gd.lt.0
c                        (  0.              ) if abs(er).le.small.ge.0.
```

```
c
c     ggu,ggd,w,and f are referenced indirectly as w(k(i)),i=1,n  etc.
c     minima will be at equations k(ml),k(ml+1),...k(mh)
c
      real w(nd),f(nd),gu(nd),gd(nd)
      real g(2001)
      real gn,gp,small,er,t,gnt,gplx,gmix,grad,gpt
c
      integer k(nd)
      integer low,large,ml,mh,itry,n,kix,i,l,k,mlt,mht,nd,ixg
c
      low=1
      large=n
      ml=n
      mh=1
      gn=0.
      gp=0.
c
      do 50 itry=1,n
         kix=low+mod((large-low)/3+itry,large-low+1)
         l=k(kix)
         if (abs(w(l)).eq.0.) goto 50
         t=f(l)/w(l)
         f(l)=w(l)*t
         do 10 i=low,large
            l=k(i)
            er=f(l)-w(l)*t
            g(l)=0.
            if (er.gt.small) g(l)=-w(l)*gu(l)
            if (er.lt.-small) g(l)=-w(l)*gd(l)
10       continue
         call split(low,large,k,g,mlt,mht)
         gnt=gn
         do 20 i=low,mlt
            ixg=k(i)
            gnt=gnt+g(ixg)
20       continue
         gpt=gp
         do 30 i=mht,large
            ixg=k(i)
            gpt=gpt+g(ixg)
30       continue
         gplx=0.
         gmix=0.
         do 40 i=mlt,mht
            l=k(i)
            if (w(l).lt.0.) gplx=gplx-w(l)*gu(l)
            if (w(l).gt.0.) gplx=gplx-w(l)*gd(l)
            if (w(l).gt.0.) gmix=gmix-w(l)*gu(l)
            if (w(l).lt.0.) gmix=gmix-w(l)*gd(l)
40       continue
         grad=gnt+gpt
         if ((grad+gplx)*(grad+gmix).lt.0.) goto 60
         if (grad.ge.0.) low=mht+1
         if (grad.le.0.) large=mlt-1
         if (low.gt.large) goto 60
         if (grad.ge.0.) gn=gnt+gmix
         if (grad.le.0.) gp=gpt+gplx
         if ((grad+gplx).eq.0.) ml=mlt
         if ((grad+gmix).eq.0.) mh=mht
 50   continue
c
 60   ml=min0(ml,mlt)
      mh=max0(mh,mht)
c
      return
      end
c
c-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
c
      subroutine split (low,large,k,g,mlpass,mhpass)
c
c  given g(k(i)),i=low,large
c  then rearrage k(i),i=low,large and find ml,mh so that
c  (g(k(i)),i=low,(ml-1)) .lt. 0 and
c  (g(k(i)),i=ml,mh)=0. and
c  (g(k(i)),i=(mh+1),large) .gt.0.
c
      real g(5000)
c
      integer k(large)
      integer ml,mh,ixg,keep,k,ii,i,ixgml,ixgmh,
     *   mlpass,mhpass,low,large
c
      ml=low
      mh=large
 10   ml=ml-1
 20   ml=ml+1
```

```
      ixg=k(ml)
      if (g(ixg)) 20,30,30
 30   mh=mh+1
 40   mh=mh-1
      ixg=k(mh)
      if (g(ixg)) 50,50,40
 50   keep=k(mh)
      k(mh)=k(ml)
      k(ml)=keep
      ixgml=k(ml)
      ixgmh=k(mh)
      if (g(ixgml).ne.g(ixgmh)) goto 10
c
      do 60 i=ml,mh
         ii=i
         ixg=k(i)
         if (g(ixg).ne.0.0) goto 70
 60   continue
c
      mlpass=ml
      mhpass=mh
      return
c
 70   keep=k(mh)
      k(mh)=k(ii)
      k(ii)=keep
      goto 30
c
      end
```

```
c
c-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
c
c-----wwvb  : wwvb decodor (determines absolute time of first sample)
c
c in                            **  **
c     usage : call wwvb(minrat, ns, ks, step, kda, khr, kmn, sec, ires)
c out                                **  ****  ***  ***  ***  ***  ****
c
c            ns : number of sample in time code
c            ks : record to decode (standard integers,
c                 already massaged into unspikey binary)
c          step : sampling rate
c           kda : ("Julian") day of year        (ires = 5)
c                 (-1 indicates infinite loop exit from elskew)
c           khr : hour of day                   (ires = 5)
c           kmn : minute of hour                (ires = 5)
c           sec : seconds past minute           (ires = 3, 5)
c                 seconds past 10 sec mark      (ires = 2)
c                 seconds past  1 sec mark      (ires = 1)
c
c          ires : synchronization level obtained
c                 0 - total failure to decode
c                 1 - 1 second synch. (fractional "sec" now OK)
c                 2 - 10 second synch. (units of "sec" OK too)
c                 3 - minute synch. (all of "sec" now OK)
c                 5 - fully decoded (day, hour, and minute OK too)
c
c     blame : carl johnson (9 apr 83)
c
c     Modified for hard typing and external signal processing by
c          J. R. Evans, USGS, Menlo Park, March, 1992.
c          Also demystified the use of "t0" and "td" in "rdecod"
c          common block--they had been assumed to form a
c          sequential pair in memory due to their places in
c          the common block (dangerous and obscure programming).
c          These values now are buffered through "tt".
c
      subroutine wwvb (ns, ks, step, kda, khr, kmn, sec, ires)
c
      real s,slast,wt,rate,sdel,step,small,xj10,xj60,
     *   secs,tt(2)
c*    real sum,sex
c
      integer ks(1)
      integer i,i1,i2,i3,i4,i5,j,j1,j2,js,ns,is,ismax,isyn,nsyn,
     *   ires,krate,i10,max10,mknt,iat,itick,ntick,n,
     *   kode,j10,n10,j60,imn,imn10,jmin,ibase,ihr,ihr10,
     *   ida,ida10,ida100
c*    integer kbias,ksis
c
      integer*4 kda
      integer*4 khr
      integer*4 kmn
      real sec
c
      include 'rdecod.inc'
c
      ismax = ns - 1
c
      kda = 0
c
c*----signal conditioning
c*----(replaced by robust processing in the calling routine)
c*
c*----bias calculation
c*    sum = 0.0
c*    do 120 is=1, ns
c*       sum = sum + ks(is)
c*120 continue
c*    kbias = sum / ns
c*
c*----convert to binary representation
c*    do 130 is=1, ns
c*       ksis = 0
c*       if (ks(is) .gt. kbias) ksis = 1
c*       ks(is) = ksis
c*130 continue
c*
c*----first order deglitching
c*    do 150 is = 2, ismax
c*       if (ks(is) .ne. ks(is-1) .and. ks(is) .ne. ks(is+1))
c*   *        ks(is) = ks(is+1)
c*150 continue
c
c-----Calculate approximate digitization interval
c
      do 220 isyn = 1, MAXSYN
```

```
          ksyn(isyn) = 0
  220 continue
c
c     Rising-edge detector
      isyn = 0
      do 230 is = 1, ismax
         if (isyn .lt. MAXSYN) isyn = isyn + 1
         if (ks(is) .eq. 1) goto 230
         if (ks(is+1) .eq. 0) goto 230
         ksyn(isyn) = ksyn(isyn) + 1
         isyn = 0
  230 continue
c
c     Calculate approximate sampling period ("step")
c     kper = 0
      mknt = 0
      do 250 isyn = 1, MAXSYN
         if (ksyn(isyn) .le. mknt) goto 250
         mknt = ksyn(isyn)
         krate = isyn
  250 continue
c
      rate = 0.0
      wt = 0.0
      do 260 i=1, 3
         j = krate + i - 2
         rate = rate + j * ksyn(j)
         wt = wt + ksyn(j)
  260 continue
      rate = rate / wt
      step = 1.0 / rate
c     write(*, *) 'rate, step :', rate, step
c
      ires = 0
      if (krate .le. 1 .or. krate .ge. MAXSYN) return
c
c-----Calculate time code phase
      slast = -10.0
c
      do 315 i=1, 10
         k10(i) = 0
  315 continue
      nsyn = 0
c
      do 320 is = 1, ismax
         if (ks(is) .eq. 1) goto 320
         if (ks(is+1) .eq. 0) goto 320
         s = is * step
         sdel = s - slast
         if (sdel .lt. 1.0 - 2.0 * step) goto 320
         slast = s
         if (sdel .gt. 1.0 + 2.0 * step) goto 320
         js = s
         s = s - js
c--------Initial crude phase estimate
         i10 = 10.0 * s
         i10 = i10 + 1
         if (i10 .gt. 10) i10 = 1
         k10(i10) = k10(i10) + 1
c
         nsyn = nsyn + 1
         ksyn(nsyn) = is
c        type *, nsyn, is, s
         if (nsyn .ge. MAXSYN) goto 340
  320 continue
      if (nsyn .lt. 10) goto 680
c
c-----First crude phase (fractional seconds) estimate
  340 max10 = 0
      do 345 i=1, 10
         if (k10(i) .lt. max10) goto 345
         max10 = k10(i)
         t0 = (i-1) * 0.10
  345 continue
c     type *, 'crude t0 :', t0
c
c-----Setup normal equations (to find start-time fractional
c     seconds and sample interval):  t = t0 + step * is
c
      do 350 i=1, nsyn
      gu(i) = 1.0
      gd(i) = -1.0
      iat = 2 * (i - 1)
      at(iat + 1) = 1.0
      at(iat + 2) = ksyn(i)
c     xx = step * ksyn(i) - t0
      t(i) = anint(step * ksyn(i) - t0)
c     type *, i, ksyn(i), t(i), xx
```

```
  350 continue
c
c-----Solve for final "t0" and "step"
      small = step * 0.01
      call elskew (2, nsyn, at, t, gu, gd, small, tt)
c
c-----If exited apparant infinite loop, then bug out, indicating garbage
      if (tt(1).eq.0.0 .and. tt(2).eq.0.0) then
         ires = 0
         kda = -1
         return
      endif

      t0 = tt(1)
      td = tt(2)
      if (t0 .lt. 0.0) t0 = t0 + 1.0
      if (t0 .ge. 1.0) t0 = t0 - 1.0
      step = td
      sec = t0 + td
c
c-----Valid sampling interval ("step") and
c     fractional seconds of start time ("sec")
      ires = 1

c     write(*, *) 'sync.  1 :', sec
c
c     do 380 i=1, nsyn
c        ti = t0 + step * ksyn(i)
c        type *, ti
c 380 continue
c
c-----Translate time code elements (0-fubar, 1-zero, 2-one, 3-mark)
      ntick = ismax * step
      if (ntick .gt. MAXSYN) ntick = MAXSYN
c
      rate = 1.0 / step
      krate = rate
      do 490 itick = 1, ntick
         i1 = anint((itick - t0) / step) + 0.5
         if (i1 .lt. 1) goto 470
         i2 = i1 + 0.2 * rate
         i3 = i1 + 0.5 * rate
         i4 = i1 + 0.8 * rate
         i5 = i1 +       rate
c
c-----Test first segment
         n = i2 - i1
         j = 0
         do 425 i=1, n
            j = j + ks(i1 + i)
  425    continue
         if (j .lt. 0.7 * n) goto 470 ! fubar
         kode = 1
c
c-----Test second segment
         n = i3 - i2
         j = 0
         do 435 i=1, n
            j = j + ks(i2 + i)
  435    continue
         if (j .lt. 0.3 * n) goto 450 ! tick = zero
         if (j .lt. 0.7 * n) goto 470 ! fubar
         kode = 2
c
c-----Test third part
         n = i4 - i3
         j = 0
         do 445 i=1, n
            j = j + ks(i3 + i)
  445    continue
         if (j .lt. 0.3 * n) goto 460 ! tick = one
         if (j .lt. 0.7 * n) goto 470 ! fubar
         kode = 3
         goto 460
c
c-----Test third segment low
  450    n = i4 - i3
         j = 0
         do 455 i=1, n
            j = j + ks(i3 + i)
  455    continue
         if (j .gt. 0.3 * n) goto 470 ! fubar
c
c-----Test fourth segment low
  460    n = i5 - i4
         j = 0
         do 465 i=1, n
            j = j + ks(i4 + i)
```

```
  465    continue
         if (j .gt. 0.3 * n) goto 470 ! fubar
         goto 480
c
c-----Fubar
  470    kode = 0
c
  480    ksyn(itick) = kode
         j1 = i1 + 1
         j2 = i5
c        write(*, 5480) itick, kode, (ks(j), j=j1, j2)
 5480    format(' ', 2i5, ' :', 60i1)
c
  490 continue
c
c     write(*, 5490) (ksyn(i), i=1, ntick)
 5490 format(' ', 60i1)
c
c-----10-second synchronization
      i10 = 0
c
      do 520 i=1, 10
         k10(i) = 0
  520 continue
c
      do 530 i=1, ntick
         i10 = i10 + 1
         if (i10 .gt. 10) i10 = 1
         if (ksyn(i) .eq. 3) k10(i10) = k10(i10) + 1
  530 continue
c
      j10 = 0
      n10 = 0
      do 540 i=1, 10
         if (k10(i) .lt. n10) goto 540
         j10 = i
         n10 = k10(j10)
  540 continue
c
      if (j10 .lt. 1) return
      xj10 = (j10 - t0) / step
      secs = 9.0 - step * (xj10 - 1.0)
c
c     if (secs .ge. 10.0) secs = secs - 10.0
c     if (secs .lt. 0.0 ) secs = secs + 10.0
c
      if (secs .ge. 10.0) then
        secs = secs - 10.0
        j10 = j10 + 10
      endif
c
      if (secs .lt.  0.0) then
        secs = secs + 10.0
        j10 = j10 - 10
      endif
c
c-----Units of seconds also now valid.
      sec = secs
      ires = 2
c
c     write(*, *) 'sync. 10 :', sec
c
c-----Minute synchronization
      j60 = j10 - 10
c
  550 j60 = j60 + 10
      if (j60 .gt. ntick) return
      if (ksyn(j60 + 1) .ne. 3) goto 550
c
c-----Minute mark found
      xj60 = (j60 - t0) / step
      secs = 59.0 - step * (xj60 - 1.0)
      if (secs .ge. 60.0) secs = secs - 60.0
      if (secs .lt.  0.0) secs = secs + 60.0
c
c-----Tens of seconds also now valid
      sec = secs
      ires = 3
c
c     write(*, *) 'sync. 60 :', sec
c
c     do 580 i=1, 300
c        sex = sec + (i-1) * step
c        type *, i, ks(i), sex
c 580 continue
c
c-----Translate BCD time-of-day code
c-----Minutes
```

```
      jmin = -1
  620 jmin = jmin + 1
      ibase = j60 + 60 * (jmin - 1)
      if (ibase + 2 .lt. 1) goto 620
      if (ibase + 8 .gt. ntick) goto 680
c
      call binary (3, ksyn(ibase+2), imn10)
      if (imn10 .lt. 0) goto 620
      if (imn10 .gt. 5) goto 620
c
      call binary (4, ksyn(ibase+6), imn)
      if (imn .lt. 0) goto 620
      if (imn .gt. 9) goto 620
      kmn = 10 * imn10 + imn - jmin
      if (kmn .gt. 59) goto 620
      if (kmn .lt. 0) kmn = kmn + 60
c
c-----Hours
      jmin = -1
  640 jmin = jmin + 1
      ibase = j60 + 60 * (jmin - 1)
      if (ibase + 13 .lt. 1) goto 640
      if (ibase + 19 .gt. ntick) goto 680
c
      call binary (2, ksyn(ibase+13), ihr10)
      if (ihr10 .lt. 0) goto 640
      if (ihr10 .gt. 2) goto 640
c
      call binary (4, ksyn(ibase+16), ihr)
      if (ihr .lt. 0) goto 640
      if (ihr .gt. 9) goto 640
c
      khr = 10 * ihr10 + ihr
      if (khr .gt. 23) goto 640
      if (kmn + jmin .gt. 59) khr = khr - 1
      if (khr .eq. -1) khr = 23
c
c-----Day of year
      jmin = -1
  660 jmin = jmin + 1
      ibase = j60 + 60 * (jmin - 1)
      if (ibase + 23 .lt. 1) goto 660
      if (ibase + 34 .gt. ntick) goto 680
c
      call binary (2, ksyn(ibase+23), ida100)
      if (ida100 .lt. 0) goto 660
      if (ida100 .gt. 3) goto 660
c
      call binary (4, ksyn(ibase+26), ida10)
      if (ida10 .lt. 0) goto 660
      if (ida10 .gt. 9) goto 660
c
      call binary (4, ksyn(ibase+31), ida)
      if (ida .lt. 0) goto 660
      if (ida .gt. 9) goto 660
c
      kda = 100 * ida100 + 10 * ida10 + ida
      if (ida .gt. 366) goto 660
      if (kmn + jmin .gt. 59 .and. khr .eq. 23) kda = kda - 1
c
c-----Translation successfull (day, hour, and minute also now valid)
      ires = 5
      return
c
c-----Translation failure, return with seconds past minute
  680 return
      end
c
c-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
c
c-----binary : translate binary time code fields
      subroutine binary (n, kd, ires)
c
      integer kd(1)
      integer i,n,ires,ival
c
      ival = 0
      ires = -1
      do 10 i=1, n
         if (kd(i) .lt. 1) return
         if (kd(i) .gt. 2) return
         ival = ival + ival + kd(i) - 1
   10 continue
      ires = ival
c
      return
      end
```

```
c
c-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
c
c-----rdecod.inc : variables needed for decoding
c
      real*4 sec
      integer*4 ksyn
      real*4 gu
      real*4 gd
      real*4 at
      real*4 t
      integer*4 k10
      real*4 t0
      real*4 td
      integer*4 MAXSYN
      parameter (MAXSYN = 2500)

c     ksyn(MAXSYN)      binary blips
c     gu(MAXSYN)        upper constraint
c     gd(MAXSYN)        lower constraint
c     at(2*MAXSYN)      coef. norm. eqs.
c     t(MAXSYN)         right hand side
c     k10(100)
c     t0, td            calculated by ELSKEW called by IRIGE and WWVB
c                       CAUTION:  Due to overly clever programming, it
c                       was essential that t0, td remained a sequential
c                       pair in memory.  This has been demystified in
c                       WWVB.

      common /rdecod/ ksyn(MAXSYN), gu(MAXSYN), gd(MAXSYN),
     *   at(2*MAXSYN), t(MAXSYN), k10(100), t0, td
```

```
#
#-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
#
# Generic Makefile for programs.
# Works for any mix of sun3, sun4, Fortran, and C (I hope!).
#
# "make install" fails the first time unless the executable has
# already been copied to $(BINDIR).
#
# The following is required in the user's .cshrc:
#
# setenv ARCH `/bin/arch`
# if( $ARCH == "sun3") then
#         setenv FLOAT -f68881
# else
#         setenv FLOAT ""
# endif
# setenv DIR_USR /usr/local
# setenv DIR_SRC /usr/src/local

FC      = f77
CC      = cc
FFLAGS  = -u $(FLOAT) -D$(ARCH) -Bstatic
CFLAGS  = $(FLOAT) -D$(ARCH) -Bstatic
BINDIR  = $(DIR_USR)

PROG    = ahwwvb
SRCARC  = $(PROG).src.a

FSRCS   = elskew.f \
        wwvb.f
CSRCS   = ahwwvb.c
OBJECTS = $(FSRCS:%.f=$(ARCH)/%.o) $(CSRCS:%.c=$(ARCH)/%.o)

LIBS    = $(DIR_USR)/librmf.a -lq -lm -lah
LLIBS   = $(LIBS)

INCLS   = rdecod.inc ahhead.h
OTHERS  = Makefile ahwwvb.man announce announce.rev \
        announce.rev2 $(INCLS)
OTHERBIN=

# P     = enscript -r2 -p-

$(ARCH)/$(PROG): $(OBJECTS)
        $(CC) $(CFLAGS) $(OBJECTS) $(LIBS) -o $@

$(OBJECTS): $(INCLS)

$(ARCH)/%.o: %.c
        $(CC) $(CFLAGS) -c $(@F:.o=.c) -o $@

$(ARCH)/%.o: %.f
        $(FC) $(FFLAGS) -c $(@F:.o=.f) -o $@

install: $(ARCH)/install $(ARCH)/$(PROG)
#       /bin/rm install
#       ln -s $(ARCH)/install install
        /bin/mv $(BINDIR)/$(PROG) $(BINDIR)/$(PROG).old
        cp $(ARCH)/$(PROG) $(BINDIR)
        chmod 755 $(BINDIR)/$(PROG)
        @touch install
        @touch $(ARCH)/install

# CAUTION:  do not run "make clean" on more than one machine at a time;
# the archive commands ("ar") might conflict.
clean:
        /bin/rm $(ARCH)/$(PROG) $(OBJECTS) $(BINDIR)/$(PROG).old
        ar ruv $(SRCARC) $(FSRCS) $(CSRCS) $(OTHERS) $(OTHERBIN)

lint: $(FSRCS) $(CSRCS) $(INCLS)
        /usr/bin/lint -xh -u $(FSRCS) $(CSRCS) $(LLIBS) > lint

print:  $(FSRCS) $(CSRCS) $(OTHERS)
#       pr -l66 $? | $P
        enscript $?
        @touch print

printall:
#       pr -l66 $(FSRCS) $(CSRCS) $(OTHERS) | $P
        enscript $(FSRCS) $(CSRCS) $(OTHERS)
        @touch print

printexport:
        pr -t $(FSRCS) $(CSRCS) $(OTHERS)
```

To: dawson@iyers croker@iyers moses@andreas michael@andreas julian@andreas oneill@andreas AWalter lindh@gsvax0 iyer@gsvax0
Cc: evans@evanss
Subject: AHWWVB

My new WWVB interpreter for xdr-AH files is now ready. It has been tested at 100 sps on eight five-day-recorder examples from Arizona and one CUSP example from Pasadena. It should work well, but could require a little tweeking for your own cases. It probably has the most robust signal massaging of any WWVB reader--I spent the better part of a week looking at time codes and making that processing smarter. It uses a series of running-median filters and an edge finder. After massaging, it uses Carl Johnson's inflagrevious FORTRAN spaghetti to interpret the time code. Carl's routine determines digitizing interval and fractional seconds by an L1 fit, so the results are pretty good, though variations of 0.02 s along the trace are routine with five-day recorders.

The result is good enough to have shown just how bad current five-day recorder digitizing is--tape speed, and therefore digitizing rate, vary by as much as 40% in the worst cases (2 of the 8 tested!) producing a visible telescoping of the WWVB signal. AHWWVB tries to warn you about these problems by reporting any apparent fluctuations in WWVB second-intervals in the massaged trace.

Source code and documentation are in:
iyers:/we/itch/evans/src/proc/ahwwvb

Sun3 executable is:
evanss:/we/itch/evans/usr.sun3/ahwwvb

Sun4 executable is:
iyers:/we/itch/evans/usr.sun4/ahwwvb

AHWWVB spawns an AHCLK job to make the actual file changes, so you need that routine too. AHCLK also is used by EPICK to change file start time and digitizing rate, and is a sometimes-useful filter by itself.

Source code and documentation are in:
iyers:/we/itch/evans/src/proc/ahclk

Sun3 executable is:
evanss:/we/itch/evans/usr.sun3/ahclk

Sun4 executable is:
iyers:/we/itch/evans/usr.sun4/ahclk

To compile, you also need my RMF library (which also contains a rather good general despiker):

Source code is in:
iyers:/we/itch/evans/src/lib/rmf

Sun3 binary is:
evanss:/we/itch/evans/usr.sun3/librmf.a

Sun4 binary is:
iyers:/we/itch/evans/usr.sun4/librmf.a

I'll put copies of the manual pages for AHWWVB in your boxs.

Good luck!
John Evans

To: dawson@iyers croker@iyers moses@andreas michael@andreas julian@andreas oneill@andreas AWalter lindh@gsvax0 iyer@gsvax0
Cc: evans@evanss
Subject: AHWWVB

A "-nc" (no change) option has been added to AHWWVB to supress changes
to the ah traces themselves. The default is to change the trace data.

AHWWVB now also prints (stdout) a summary of the results, as follows:

```
--------------------ahwwvb--------------------
File "tbnh1320.xdr", code "LTB6", channel "spT":
Start time = 1990/11/08  13:20:01.3316 UTC
Sample interval = 0.0101375813 s (98.642859 sps)
----------------------------------------------
```

Some folks prefer to collect a list of clock corrections rather than
applying them to the traces. Hence the ability to get this message
without changing the trace data.

Please let me know if you discover any bugs.

--John Evans

To: dawson@iyers croker@iyers michael@andreas julian@andreas oneill@andreas AWalter
Cc: evans@evanss
Subject: AHWWVB

More diddling with AHWWVB (hopefully this will be all for awhile):

There is now a "-v" flag to make it verbose (formerly the only choice). If missing, one only receives warnings and the final answer.

With -v, it will act as before but with addition of two lines of output ("AHWWVB verbose output:" and "Interpreting time series ..."), the former at the beginning and the latter when it finishes filtering and starts trying to interpret the cleaned up trace. There are also a number of minor punctuation changes for the sake of consistancy. Two typical examples are appended here. Script writers take note.

Bruce ran into another bug for high sample rates. The L1 fitter sometimes went into an infinite loop. This should now be fixed and a trap installed just in case.

Please let me know if you discover any other bugs.

--John Evans

```
iyers(evans):111==> ahwwvb -m 80 tbnh1320.xdr -vb LTB6 spT -v
AHWWVB verbose output:
Longest RMF window is of 15 points.
Compensating for low-pass filtered WWVB.
Running RMF prefilter of 3 points ...
Running RMF prefilter of 5 points ...
Running main RMF filter of 15 points ...
Interpreting time series ...

Second intervals range from 0.993 to 1.004 s.
5th and 95th percentiles differ by 0.010 s.

Defaulting year to 1990.

Changing start time and interval ...
        ahclk -l tbnh1320.xdr.a28305 -s 1990 11 08 13 20 1.331641 -i 0.0101375813
        ... successful.


----------------------ahwwvb----------------------
File "tbnh1320.xdr", code "LTB6", channel "spT":
Start time = 1990/11/08  13:20:01.3316 UTC
Sample interval = 0.0101375813 s (98.642859 sps)
--------------------------------------------------

iyers(evans):112==> ahwwvb -m 990 06.29.0192.4.ah -vb 0192 4 -nc

----------------------ahwwvb----------------------
File "06.29.0192.4.ah", code "0192", channel "4":
Start time = 1991/04/10  14:06:29.2087 UTC
Sample interval = 0.0009999890 s (1000.011012 sps)
--------------------------------------------------

iyers(evans):113==>
```

BUG REPORT for AHWWVB.

The lags given with option "-d" were applied with reversed sign
in versions prior to 23 Oct 92. If you have source code, change
one line of ahwwvb.c as follows:

```
from:   new_st -= lagg / 1000.0;                /* Remove any delay     */
to:     new_st += (lagg / 1000.0);              /* Remove any delay     */
```

Otherwise, get binaries from:
Sun4 iyers:/we/itch/evans/usr.sun4
Sun3 evanss:/we/itch/evans/usr.sun3

Sorry about this, folks.

--John

13. Appendix G: Manual Page and Source Code for *ahclk*

The program *ahclk* is a C program called by *ahwwvb* to actually make the changes to start time and sample rate in an *xdr-ah* seismogram's header. Program *ahclk* is a creation of the first author. Both the source code and the manual page are available *via* anonymous *ftp* from "andreas.wr.usgs.gov" in directory "~ftp/pub/outgoing/evans/wwvb". This Open-File Report serves to release them to the public, subject to the limitations cited on the title page.

SYNOPSIS

ahclk -l *list_file* [**-s** *year mo dy hr mn sec*] [**-i** *new_interval*]

DESCRIPTION

Changes starting time and/or digitizing interval in the header of one or more traces listed in *list_file* (same format as *list_file* used by *epick*). Copies through, unchanged, all traces encountered that are NOT listed in *list_file*.

ahclk is intended primarily for correcting start time and digitizing interval of data from "five-day recorders" and other instruments that record a time code in parallel with seismic data, but which may give inaccurate start time or digitizing interval due to inaccurate internal clocks or analog-instrumentation foibles. Hence, it assumes that all traces in *list_file* are *exactly the same* time interval as all the other traces listed. You will get bizarre results if this is not so.

ahclk can be used to correct for clock drift and similar errors that apply individually or to a subset of traces in the AH file by listing just the appropriate trace(s) in *list_file*.

ARGUMENTS

-l *list_file* must contain one or more AH file names and trace sequence-numbers.

OPTIONS

-s *year mo dy hr mn sec* is the time of the first sample in the file (e.g. "-s 1990 10 20 11 04 20.071"). Replaces "ahhead->record->abstime" in the *ah* header. These six values *must* all be positive.

-i ***new_interval* is the new sampling interval (seconds). Replaces "ahhead->record->delta" in the *ah* header. *new_interval must* be positive.**

SEE ALSO

epick—X11 *ah* picking routine derived and extended from Lamont's *sunpick*. *epick* calls *ahclk* to invoke changes in trace timing derived from "alternate time" picks (i.e., of time codes recorded in parallel with seismograms). *ahclk* may be used as a stand-alone filter in the normal way as well.

Other *ah* filters.

BUGS

list_file must be sorted with all the traces for any particular file together and in ascending numerical order within each file. *mkahlist* does that for you. Someday, *ahclk* could be made smarter about this, but it hasn't been yet. *ahclk* does verify this ordering, however.

Should get smart enough to check relatedness of traces being changed.

AUTHOR

John R. Evans, USGS, Menlo Park, CA. Original written September 25, 1990.

```
/*
 * AH filter AHCLK (AH CLocK change):
 */

#include <stdio.h>
#include <rpc/rpc.h>
#include <local/ahhead.h>

#define NM_LEN  128     /* Maximum file-name length             */
#define STR_LEN 256     /* Maximum scratch-string length        */
#define MX_LST  256     /* Maximum line count of listfile       */

#define strneq  !strncmp
#define streq   !strcmp

        struct ah_time  NULL_AH_TIME;   /* NULL starting time           */
        void            error();        /* -lq (error.c)                */
        void            report();       /* -lq (error.c)                */
        FILE            *efopen();      /* -lq (efopen.c)               */

char *progname;

main(argc, argv)
        int     argc;
        char    **argv;
{
        FILE            *list_file;     /* listfile a la "epick"        */
        char            list_name[NM_LEN];
        void            filter();       /* Defined below                */
        struct ah_time  new_st;         /* New starting time            */
        double          new_int;        /* New sampling interval        */
        void            mod_files();    /* Defined below                */
        int             seq_ok();       /* Defined below                */

        /* Initialize */
        (void)bzero((char *)(&NULL_AH_TIME), sizeof(NULL_AH_TIME));

        new_st = NULL_AH_TIME;          /* No-change flag               */
        new_int = -1.;                  /* No-change flag               */

        list_name[0] = '\0';
        list_file = (FILE *) NULL;

        /* Check usage */
        if (argc < 5)
                error("Usage: %s -l listfile [-s year mo dy hr mn sec] [-i new_interval]",
                        argv[0]);

        progname = argv[0];

        /* Interpret command line arguments */
        for (--argc, argv++; argc; --argc, argv++) {

                /* listfile name */
                if (strneq(*argv, "-l", 2)) {
                        --argc, argv++;
                        (void)strcpy(list_name, *argv);
                }

                /* new start time */
                else if (strneq(*argv, "-s", 2)) {
                        --argc; argv++;
                        if (argc < 6)
                                report ("Bad start time");
                        new_st.yr = atoi(*argv); --argc; argv++;
                        new_st.mo = atoi(*argv); --argc; argv++;
                        new_st.day = atoi(*argv); --argc; argv++;
                        new_st.hr = atoi(*argv); --argc; argv++;
                        new_st.mn = atoi(*argv); --argc; argv++;
                        if (sscanf((*argv), "%f", &(new_st.sec)) != 1)
                                report ("Bad start time");
                }

                /* new sampling interval */
                else if (strneq(*argv, "-i", 2)) {
                        --argc; argv++;
                        if ((sscanf((*argv), "%lf", &new_int) != 1) ||
                                (new_int <= 0.))
                                report ("Bad sampling interval");
                }
                else
                        error("Illegal option \"%s\"", *argv);
}

/* Open seismogram listfile */
if(list_name[0] == '\0')
        error("Need listfile name");
list_file = efopen(list_name, "r");
```

```
        /* Check sequence in listfile */
        rewind (list_file);
        if (!seq_ok(list_file))
                error("listfile out of sequence");

        /* Process the named files */
        rewind (list_file);
        (void)mod_files(list_file, new_st, new_int);

        /* Exit */
        (void)fclose(list_file);
        exit(0);
}

/*
 * Modify all files in list_file; copy all others unchanged.
 */

void
mod_files(list_file, new_st, new_int)
        FILE            *list_file;     /* listfile a la "epick"        */
        struct ah_time  new_st;         /* New starting time            */
        double          new_int;        /* New sampling interval        */
{
        void            close_out();    /* Defined below                */
        int             copy_head();    /* Defined below                */
        ahhed           tmp_head;       /* Space to return ah header    */
        void            copy_trace();   /* Defined below                */

        char            fname_in[NM_LEN];       /* Input AH-file name   */
        FILE            *ah_fp_in;
        XDR             xdr_in;

        char            fname_out[NM_LEN];      /* Temporary output file name */
        FILE            *ah_fp_out;
        XDR             xdr_out;

        int             ugid;
        int             rec_num;
        int             cur_rec;
        char            cur_file[NM_LEN];
        char            in_str[STR_LEN];        /* One line from list file */

/* Read list file, processing AH files as it goes */
ah_fp_in = NULL;
ah_fp_out = NULL;
cur_file[0] = '\0';
cur_rec = 0;

while (fgets(in_str, sizeof(in_str), list_file) != NULL) {
        if (sscanf(in_str, "%s%d%d", fname_in, &rec_num, &ugid) < 2)
                error("Bad listfile format.");

        /* Open AH file if not already open */
        if (!streq(cur_file, fname_in)) {

                /* If input file is already open, copy through
                   any remaining traces, then close the output
                   file and overwrite fname_in with it */
                if (ah_fp_in != NULL)
                        close_out(fname_in, ah_fp_in, &xdr_in,
                                fname_out, ah_fp_out, &xdr_out);

                /* Open AH file, attach to XDR stream */

                (void)sprintf(fname_out, "%s.XXXXXX", fname_in);
                (void)mktemp(fname_out);

                ah_fp_in = efopen(fname_in, "r");
                ah_fp_out = efopen(fname_out, "w");
                xdrstdio_create(&xdr_in, ah_fp_in, XDR_DECODE);
                xdrstdio_create(&xdr_out, ah_fp_out, XDR_ENCODE);

                cur_rec = 1;
                (void)strcpy(cur_file, fname_in);
}

/* Copy through any unaffected traces prior to target trace */
while (cur_rec < rec_num) {
        (void)copy_head(&tmp_head, &xdr_in, &xdr_out,
                NULL_AH_TIME, -1.);
        (void)copy_trace(&xdr_in, &xdr_out, tmp_head);
        cur_rec++;
}

/* Modify and copy affected traces */
if (cur_rec == rec_num) {
```

```
                        (void)copy_head(&tmp_head, &xdr_in, &xdr_out,
                                new_st, new_int);
                        (void)copy_trace(&xdr_in, &xdr_out, tmp_head);
                        cur_rec++;
                }
                else
                        error("This should never happen--see Evans");
        }

        /* Close out last file pair */
        close_out(fname_in, ah_fp_in, &xdr_in, fname_out, ah_fp_out, &xdr_out);
}

/*
 * Close out a file pair, overwriting input file with output file.
 */

void
close_out(fname_in, ah_fp_in, xdr_in, fname_out, ah_fp_out, xdr_out)
        char            fname_in[];     /* Input AH-file name   */
        FILE            *ah_fp_in;
        XDR             *xdr_in;

        char            fname_out[];    /* Temporary output file name */
        FILE            *ah_fp_out;
        XDR             *xdr_out;
{
        int             copy_head();    /* Defined below                */
        ahhed           tmp_head;       /* Space to return ah header    */
        char            sys_str[STR_LEN]; /* "system" call argument     */

        /* Copy through any unaffected traces remaining in file */
        while (copy_head(&tmp_head, xdr_in, xdr_out, NULL_AH_TIME, -1.))
                (void)copy_trace(xdr_in, xdr_out, tmp_head);

        /* Close files */
        xdr_destroy(xdr_in);
        xdr_destroy(xdr_out);
        (void)fclose(ah_fp_in);
        (void)fclose(ah_fp_out);

        /* Overwrite input file with output file */
        (void)sprintf(sys_str, "/bin/mv -f %s %s",
                fname_out, fname_in);
        if ((system(sys_str) & 0377) == 127)
                report("File overwrite (mv) failed!");
}

/*
 * Copy through an AH trace, changing start time and sampling interval
 * as requested.  (new_st == NULL_AH_TIME) prevents change of starting
 * time.  (new_int <= 0.) prevents change of sampling interval.
 * Hence "copy_head(hd, xdr_in, xdr_out, NULL_AH_TIME, -1.)" copies
 * header unchanged.
 *
 * Returns copied header in "hd".  Returns TRUE after successful copy,
 * or FALSE if header could not be read properly (e.g., at end-of-file).
 */

int
copy_head(hd, xdr_in, xdr_out, new_st, new_int)
        ahhed           *hd;            /* Space to return ah header    */
        XDR             *xdr_in;
        XDR             *xdr_out;
        struct ah_time  new_st;         /* New starting time            */
        double          new_int;        /* New sampling interval        */
{
        void            log_ahclk();    /* Defined below                */

        /* Read header */
        if (xdr_gethead(hd, xdr_in) != 1)
                return (FALSE);

        /* Log any changes that will be made */
        (void)log_ahclk(hd, new_st, new_int);

        /* Change start time */
        if (!strneq((char *)(&new_st), (char *)(&NULL_AH_TIME),
                sizeof(new_st)))
                hd->record.abstime = new_st;

        /* Change sampling interval */
        if (new_int > 0.)
                hd->record.delta = new_int;

        /* Write header */
        if (xdr_puthead(hd, xdr_out) != 1)
                error("Error writing header.");
```

```
        return (TRUE);
}

/*
 * Log any changes made by ahclk.
 */

void
log_ahclk(hd, new_st, new_int)
        ahhed           *hd;
        struct ah_time  new_st;         /* New starting time            */
        double          new_int;        /* New sampling interval        */
{
        char            log_str[STR_LEN];/* Log message written to ahhead */
        void            log_app();      /* Defined below                */

        if (!strneq((char *)(&new_st), (char *)(&NULL_AH_TIME),
                sizeof(new_st))) {
                (void)sprintf(log_str,
                        "%s:  abstime was %d %d %d %d %d %f;", progname,
                        hd->record.abstime.yr, hd->record.abstime.mo,
                        hd->record.abstime.day, hd->record.abstime.hr,
                        hd->record.abstime.mn, hd->record.abstime.sec);
                (void)log_app(hd, log_str);
        }

        if (new_int > 0.) {
                (void)sprintf(log_str,
                        "%s:  delta was %f;", progname, hd->record.delta);
                (void)log_app(hd, log_str);
        }
}

/*
 * Append log_str to log message in ahhead "hd".
 */

#define MIN(a,b) (((a)<(b)) ? (a) : (b))/* Smaller value                */

void
log_app(hd, log_str)
        ahhed           *hd;
        char            log_str[];
{
        int             strl;
        void            ch_app();       /* Defined below                */

        /* Find end of non-blank part of log */
        for (strl = MIN(LOGSIZE - 1, strlen(hd->record.log)) ;
                (strl >= 0) && (hd->record.log[strl] == ' ') ; --strl);

        /* If find "null" at start of string, overwrite it */
        if (strl == 4 && strneq(hd->record.log, "null", 4))
                strl = 0;

        /* Append new log message */
        if (strl != 0)
                (void)ch_app(&strl, hd, " ");
        (void)ch_app(&strl, hd, log_str);
}

/*
 * Append characters to log until full.
 */

void
ch_app(strl, hd, log_str)
        int             *strl;
        ahhed           *hd;
        char            log_str[];
{
        int             i;
        int             l_len;

        i = 0;
        l_len = strlen(log_str);
        while((*strl < (LOGSIZE - 1)) && (i < l_len))
                hd->record.log[(*strl)++] = log_str[i++];
        hd->record.log[*strl] = '\0';
}

/*
 * Copy through an AH trace, unchanged.
 */

void
copy_trace(xdr_in, xdr_out, hd)
```

```
        XDR             *xdr_in;
        XDR             *xdr_out;
        ahhed           hd;
{
        float           *darray;

        /* Allocate trace storage */
        darray = (float *)mkdatspace(&hd);

        /* Read trace */
        if (xdr_getdata(&hd, (char *)darray, xdr_in) == -1)
                error("Error reading data record");

        /* Write trace */
        if (xdr_putdata(&hd, (char *)darray, xdr_out) == -1)
                error("Error reading data record");

        /* Deallocate trace storage */
        free ((char *)darray);
}

/*
 * Check sequence in listfile
 */

typedef struct {
        char            fname_in[NM_LEN];
        int             rec_num;
        int             ugid;
} TLIST;

int
seq_ok(list_file)
        FILE    *list_file;
{
        char            in_str[STR_LEN];                /* One line from list file */

        int             i;
        int             same_nm;
        int             seq;
        int             rv;

        int             nlines;
        TLIST           *file_list[MX_LST];     /* known AH-file names  */

        rv = TRUE;
        nlines = 0;
        file_list[0] = (TLIST *)emalloc(sizeof(TLIST));

        while (fgets(in_str, sizeof(in_str), list_file) != NULL) {
                if(sscanf(in_str, "%s%d%d",
                        file_list[nlines]->fname_in,
                        &(file_list[nlines]->rec_num),
                        &(file_list[nlines]->ugid))
                        >= 2) {

                        same_nm = TRUE;
                        seq = file_list[nlines]->rec_num;
                        for(i = nlines ; i >= 0 ; --i) {
                                if (streq(file_list[nlines]->fname_in,
                                        file_list[i]->fname_in)) {
                                        if (!same_nm)
                                                rv = FALSE;
                                        if (file_list[i]->rec_num >= seq
                                                && i < nlines)
                                                rv = FALSE;
                                        else
                                                seq = file_list[i]->rec_num;
                                }

                                else {
                                        same_nm = FALSE;
                                }
                        }

                        nlines++;
                        if (nlines >= MX_LST)
                                error("listfile too long");
                        file_list[nlines] = (TLIST *)emalloc(sizeof(TLIST));
                }
                else
                        error("Bad listfile format.");
        }
        for (i = 0 ; i <= nlines ; i++)
                free((char *)file_list[i]);
        return rv;
}
```

```
/*      structure for data file header  --      witte, 11 June 85       */

#define AHHEADSIZE 1024
#define CODESIZE 6
#define CHANSIZE 6
#define STYPESIZE 8
#define COMSIZE 80
#define TYPEMIN 1
#define TYPEMAX 6
#define LOGSIZE 202
#define LOGENT 10
#define NEXTRAS 21
#define NOCALPTS 30

typedef struct  {
                float   x;
                float   y;
                } vector;

typedef struct  {
                float   r;
                float   i;
                } complex;

typedef struct  {
                double r;
                double i;
                } d_complex;

typedef struct  {
                float   xx;
                float   yy;
                float   xy;
                } tensor;

struct  ah_time {
                short           yr;     /* year         */
                short           mo;     /* month        */
                short           day;    /* day          */
                short           hr;     /* hour         */
                short           mn;     /* minute       */
                float           sec;    /* second       */
                };

struct  calib   {
                complex         pole;   /* pole         */
                complex         zero;   /* zero         */
                };

struct  station_info    {
                char            code[CODESIZE]; /* station code         */
                char            chan[CHANSIZE]; /* lpz,spn, etc.        */
                char            stype[STYPESIZE];/* wwssn,hglp,etc.     */
                float           slat;           /* station latitude     */
                float           slon;           /*    "    longitude    */
                float           elev;           /*    "    elevation    */
                float           DS;     /* gain */
                float           A0;     /* normalization */
                struct  calib   cal[NOCALPTS];  /* calibration info     */
                };

struct  event_info      {
                float           lat;            /* event latitude       */
                float           lon;            /*   "   longitude      */
                float           dep;            /*   "   depth          */
                struct  ah_time ot;             /*   "   origin time    */
                char            ecomment[COMSIZE];      /*      comment line    */
                };

struct  record_info     {
                short           type;   /* data type (int,float,...)    */
                long            ndata;  /* number of samples            */
                float           delta;  /* sampling interval            */
                float           maxamp; /* maximum amplitude of record  */
                struct  ah_time abstime;/* start time of record section */
                float           rmin;   /* minimum value of abscissa    */
                char            rcomment[COMSIZE];      /* comment line         */
                char            log[LOGSIZE]; /* log of data manipulations */
                };

typedef struct {
                struct  station_info    station;        /* station info */
                struct  event_info      event;          /* event info   */
                struct  record_info     record;         /* record info  */
                float           extra[NEXTRAS]; /* freebies */
                } ahhed;
```

```
#define FLOAT   1
#define COMPLEX 2
#define VECTOR  3
#define TENSOR  4
#define DOUBLE  6
#define INTEGER 7
```

```
#
#-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
#
# Make file for AHCLK.

CFLAGS= $(FLOAT) -g -DS(ARCH)

SRCS=   ahclk.c

OBJS=   $(SRCS:%.c=$(ARCH)/%.o)

LIBS=   -lah -lq

BIN = $(DIR_USR)

prog:   $(ARCH)/ahclk

$(ARCH)/ahclk: $(OBJS)
        cc -o $@ $(CFLAGS) $(OBJS) $(LIBS)

$(ARCH)/%.o: %.c
        cc $(CFLAGS) -c $(@F:.o=.c) -o $@

install: $(ARCH)/ahclk
        cp $(ARCH)/ahclk $(BIN)

lint: $(SRCS)
        /usr/bin/lint -xh -u $(SRCS) $(LIBS) > lint

clean:
        -@rm $(ARCH)/*.o $(ARCH)/ahclk
        ci -l -sExp -f \
           -m"USGS Computer Program ahclk; Version 1.0; John R. Evans" \
           $(SRCS) Makefile test_ah.in test_ah.out
```

In order to use the makefile in this
directory, the following MUST be
in your .cshrc file

```
#start here

setenv ARCH `/bin/arch`
if( $ARCH == "sun3") then
        setenv FLOAT -f68881
else
        setenv FLOAT ""
endif

#end here
```

14. Appendix H: Manual Pages and Source Code for Portions of *librmf* and *libq*

The library *librmf* contains C and FORTRAN programs for running-median filters and automated despikers based upon them. The routine *median* from this library is used by *ahwwvb* to filter the WWVB trace, and is listed here. Several routines from Bruce Julian's library *libq* are used by *ahwwvb* and *ahclk*, and are listed here. The source code for these routines is available *via* anonymous *ftp* from "andreas.wr.usgs.gov" in directory "~ftp/pub/outgoing/evans/library_stuff". This Open-File Report serves to release them to the public, subject to the limitations cited on the title page.

A descriptive writeup for the median filter is in file "median.writeup" of that *ftp* directory. Manual pages for the date and time routines of *libq* are included, and listed here. The routines *efopen*, *emalloc*, and *erealloc* act just like their standard C counterparts, *fopen*, *malloc*, and *realloc*, but performing error reporting. Manual pages are unavailable for the other routines.

NAME

jdn, date, yrday, mnday, isleap, ckdate – calendar routines

SYNTAX

```
#include <local/date_time.h>
#include <local/qerr.h>

long jdn(yr, mo, da, caltyp)
        int yr, mo, da, caltyp;

void date(n, pyr, pmo, pda, caltyp)
        long n;
        int *pyr, *pmo, *pda, caltyp;

int yrday(mo, da, leap)
        int mo, da, leap;

void mnday(d, leap, pmo, pda)
        int d, leap, *pmo, *pda;

int isleap(yr, caltyp)
        int yr, caltyp;

int ckdate(yr, mo, da, caltyp)
        int yr, mo, da, caltyp;
```

DESCRIPTION

These routines perform computations about the calendar.

Jdn returns the Julian Day Number of the day beginning at noon of the calendar date specified by the arguments *yr*, *mo*, and *da*. **Date** performs the inverse computation, decoding the Julian Day Number *n* and storing the results indirectly, through its pointer arguments *pyr*, *pmo*, and *pda*. In both routines, the argument *caltyp* specifies which calendar is to be used: **GREGORIAN** (modern) or **JULIAN** (old-style). These calendar-type codes are defined in the include file **date_time.h**.

Yrday returns the day of the year corresponding to day number *da* of month number *mo*. **Mnday** performs the inverse computation, determining the month and day of the month corresponding to the *n*th day of the year, and storing them through the pointer arguments *pmo* and *pda*. In both functions the boolean argument *leap* specifies whether the year in question is a leap year

Isleap returns a boolean value telling whether year *yr* is a leap year in the calendar of type *caltyp*. It is suitable for use as the argument *leap* to routines **yrday** and **mnday**.

Ckdate returns the value **0** if the date specified by arguments *yr*, *mo*, and *da* is legal in the calendar of type *caltyp*, and otherwise one of the values **EBADDAY**, **EBADMONTH**, or **EBADYEAR**, defined in the file **qerr.h**.

NOTES

Do not confuse the Julian Day Number, which counts days since 1 January -4713 of the old-style calendar, with the day of the year, which is sometimes incorrectly called the "Julian day".

SEE ALSO

date_time(3Q)

A list of the dates when various countries adopted the Gregorian calendar is given in the "Explanatory Supplement to The Astronomical Ephemeris and The American Ephemeris and Nautical Almanac", published periodically by Her Majesty's Stationary Office.

EXAMPLES

The following fragment of code...

#include <local/date_time.h>

AUTHOR

Bruce R. Julian, USGS Menlo Park, Calif.

NAME

timvar, datime – time routines

SYNTAX

```
#include <local/date_time.h>

TIME timvar(yr, mo, da, hr, mn, sec, caltyp)
        int yr, mo, da, hr, mn;
        double sec;
        int caltyp;

datime(t, pyr, pmo, pda, phr, pmn, psec, caltyp)
        TIME t;
        int *pyr, *pmo, *pda, *phr, *pmn;
        double *psec;
        int caltyp;
```

DESCRIPTION

These routines convert conventional dates and times-of-day to **TIME** (double precision) variables, a form that is computationally efficient and has enormous dynamic range (2.9×10^{-39} seconds to 10^{6} years) and resolution (one part in 10^{17}). This form is more than adequate for all purposes, ranging from geologic to picosecond scales. Using ordinary language-defined arithmetic operators, **TIME** variables may be subtracted from one another to produce time intervals, time intervals may be added and subtracted from each other and multiplied or divided by dimensionless values, and time intervals may be added to or subtracted from times to produce new times.

Timvar converts a calendar date and time of day into a **TIME** variable. **Datime** performs the inverse transformation, decoding the TIME variable *t* and storing the results indirectly, through its pointer arguments. *Caltyp* specifies which calendar is to be used: **GREGORIAN** (modern) or **JULIAN** (old-style).

The C-language include file **date_time.h** defines the type **TIME**, the calendar-type codes **GREGORIAN** and **JULIAN**, and the constant time intervals **YEAR, LEAPYEAR, WEEK, DAY, HOUR, MINUTE, SECOND, MILLISEC, MICROSEC,** and **NANOSEC**, as well as a few things useful with the **calendar**(3Q) routines.

BUGS

No allowance is made for "leap seconds", which are periodically inserted to keep the earth and the clock synchronized for the convenience of navigators.

SEE ALSO

calendar(3Q)

A list of the dates when various countries adopted the Gregorian calendar is given in the "Explanatory Supplement to The Astronomical Ephemeris and The American Ephemeris and Nautical Almanac", published periodically by Her Majesty's Stationary Office.

EXAMPLES

The following fragment of code finds the number of fortnights between two epochs:

```
#include <local/date_time.h>
double fort;
fort = (timvar(1983, 5, 31, 14, 50, 12.34, GREGORIAN) -
        timvar(1950, 1, 1, 0, 0, 0.0, GREGORIAN))/(14*DAY);
```

AUTHOR

Bruce R. Julian, USGS Menlo Park, Calif.

```
#ifndef lint
static char rcsid[]="$Header: calendar.c,v 1.1 90/11/25 12:45:10 julian Exp $";
#endif
/*
 * Calendar computations (old and new-style)
 * Bruce R. Julian, USGS Menlo Park, CA
 */
#include <local/local.h>
#include <local/date_time.h>
#include <local/qerr.h>

/* floor(x/y), where x, y>0 are integers, using integer arithmetic */
#define qfloor(x, y)  (x>0 ? (x)/y : -((y-1-(x))/y))

static int      eom[2][15] = {
        { 0, 31, 59, 90, 120, 151, 181, 212, 243, 273, 304, 334, 365, 396, 424 },
        { 0, 31, 60, 91, 121, 152, 182, 213, 244, 274, 305, 335, 366, 397, 425 },
};

/* Recognize leap years */
bool
isleap(yr, cal)
        int     yr;             /* Year                                 */
        int     cal;            /* Calendar type                        */
{
        bool    l;

        if (yr < 0)
                yr++;
        l = (yr%4 == 0);
        if (cal == GREGORIAN)
                l = l && (yr%100 != 0 || yr%400 == 0);
        return l;
}

/* Check calendar date for legality */
int
ckdate(y, m, d, cal)
        int     y, m ,d;        /* Year, month, date                    */
        int     cal;            /* Calendar type                        */
{
        int     *et = eom[isleap(y, cal)];

        if (y == 0)                         return EBADYEAR;
        if (m < 1 || m > 12)                return EBADMONTH;
        if (d < 1 || d > et[m]-et[m-1])     return EBADDAY;
        return  0;
}

/* Compute day of year */
int
yrday(mo, day, lp)
        int     mo, day;        /* Month, date                          */
        bool    lp;             /* Leap year?                           */
{
        return eom[lp][mo-1] + day;
}

/* Calculate month, date from day of year */
void
mnday(d, lp, pm, pd)
        int     d;              /* Day of year                          */
        bool    lp;             /* Leap year?                           */
        int     *pm, *pd;       /* Month, date                          */
{
        int     i;
        int     *et = eom[lp];

        for (i=1; d>et[i]; i++)
                ;
        *pm = i;
        *pd = d - et[i-1];
}

/* Compute Julian Day Number from calendar date */
long
jdn(yr, mo, day, cal)
        int     yr, mo, day;    /* Year, month, date                    */
        int     cal;            /* Calendar type                        */
{
        if (yr < 0)
                yr++;

        /* Move Jan. & Feb. to end of previous year */
        if (mo <= 2) {
                --yr;
                mo += 12;
        }
```

```
        return qfloor((long)(4*365+1)*(yr+4712), 4) + eom[0][mo-1] + day +
                (cal==GREGORIAN ? -qfloor(yr, 100) + qfloor(yr, 400) + 2 : 0);
}

/* Compute calendar date from Julian Day Number */
void
date(n, py, pm, pd, cal)
        long    n;              /* Julian Day Number                    */
        int     *py, *pm, *pd;  /* Year, month, date                    */
        int     cal;            /* Calendar type                        */
{
        long    d, t;
        int     y;

/* Find position within cycles that are nd days long */
#define CYCLE(n, nd) { t=qfloor(d-1,nd); y+=t*n; d-=t*nd; }

/* The same, with bound on cycle number */
#define LCYCLE(n, nd, l) { t=qfloor(d-1,nd); if (t>l) t=l; y+=t*n; d-=t*nd; }

        y = -4799;
        if (cal == GREGORIAN) {
                d = n + 31739;          /* JD -31739 = 31 Dec 4801 B.C. */
                CYCLE(400, 146097)      /* Four-century cycle           */
                LCYCLE(100, 36524, 3)   /* 100-year cycle               */
        }
        else
                d = n + 31777;          /* JD -31777 = 31 Dec 4801 B.C. */

        CYCLE(4, 1461)                  /* Four-year cycle              */
        LCYCLE(1, 365, 3)               /* Yearly cycle                 */

        if (y <= 0)
                --y;
        *py = y;
        mnday((int)d, isleap(y, cal), pm, pd);
}
```

```
#ifndef lint
static char rcsid[]="$Header: efopen.c,v 1.2 88/10/17 11:15:44 julian Exp $";
#endif
/*
 * Open file, die if can't
 * Bruce R. Julian, USGS Menlo Park, CA
 */
#include <stdio.h>

void            error();

FILE*
efopen(file, mode)
        char            *file, *mode;
{
        FILE    *fp;

        if ((fp = fopen(file, mode)) == NULL)
                error("Can't open file \"%s\", mode %s", file, mode);
        return fp;
}
```

```
#ifndef lint
static char rcsid[]="$Header: emalloc.c,v 1.2 88/10/17 11:15:48 julian Exp $";
#endif
/*
 * Allocate memory, die if can't
 * Bruce R. Julian, USGS Menlo Park, CA
 */
#define NULL    0
#define u_int   unsigned

char    *calloc();              /* Unix C library                       */
char    *malloc();              /* Unix C library                       */
void    error();                /* libq.a                               */
char    *realloc();             /* Unix C library                       */

char*
ecalloc(n, s)
        u_int   n, s;
{
        char    *p;

        if ((p = calloc(n, s)) == NULL)
                error("Can't allocate %u X %u bytes");
        return p;
}

char*
emalloc(s)
        u_int   s;
{
        char    *p;

        if ((p = malloc(s)) == NULL)
                error("Can't allocate %u bytes");
        return p;
}

char*
erealloc(p, s)
        char    *p;
        u_int   s;
{
        if ((p = realloc(p, s)) == NULL)
                error("Can't reallocate %u bytes");
        return p;
}
```

```
#ifndef lint
static char rcsid[]="$Header: eopen.c,v 1.2 88/10/17 11:15:50 julian Exp $";
#endif
/*
 * Open file, die if can't
 */

void    error();

int
eopen(file, mode)
        register char   *file;
        register int    mode;
{
        register int    f;

        if ((f = open(file, mode)) < 0)
                error("Can't open file \"%s\", mode %d", file, mode);
        return f;
}
```

```
#ifndef lint
static char rcsid[]="$Header: error.c,v 1.4 90/04/08 12:32:22 julian Exp $";
#endif
/*
 * Report error, optionally die
 * Bruce R. Julian, USGS Menlo Park, Calif. 14 Dec 1983
 */
#include <stdio.h>
#include <local/stdtyp.h>
#include <varargs.h>

extern  int     sys_nerr;
extern  int     errbase, nerr;
extern  char    *sys_errlist[], *q_errlist[];
extern  int     errno;
extern  char    *progname;

void    exit();
char    *sprintf();

/* Return error message */
char
*ermsg(ier)
        int     ier;
{
static  char    s[12];

        if (ier == 0)
                return (NULL);
        if (ier > 0 && ier < sys_nerr)
                return (sys_errlist[ier]);
        ier -= errbase;
        if (ier > 0 && ier < nerr)
                return (q_errlist[ier]);
        else
                return (sprintf(s, "Error %d", errbase+ier));
}

/* Print error message */
/*VARARGS*/
static void
msg(args)
        va_list args;
{
        char    *fmt;                   /* Format string                */
        char    *p;

        if (progname)
                (void)fprintf(stderr, "%s: ", progname);
        fmt = va_arg(args, char*);
        _doprnt(fmt, args, stderr);
        if ((p = ermsg(errno)) != NULL)
                (void)fprintf(stderr, " (%s)", p);
        (void)fprintf(stderr, "\n");
}

/* Print error message but don't die */
/*VARARGS*/
void
report(va_alist)
        va_dcl
{
        va_list args;

        va_start(args);
        msg(args);
        va_end(args);
}

/* Print error message and die */
/*VARARGS*/
void
error(va_alist)
        va_dcl
{
        va_list args;

        va_start(args);
        msg(args);
        va_end(args);
        exit(1);
}
```

```
/*
*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
   C-callable version of routine "median", an odd-length running
   median filter for floating-point data.

   Arrays are zero-offset, standard C arrays:
      arrin[0..arrlen-1]
      arrout[0..arrlen-1]

   This is a recoding, rather than an interfacing routine to the
   FORTRAN version because the latter proved impractical due to I/O
   system interference between C and FORTRAN (and because these
   difficulties are likely to rise anew each time a new compiler is
   introduced even if I do circumvent them now).

   FORTRAN version's "ln" here is called "arrlen".
   FORTRAN version's "m" here is called "winlen".
   FORTRAN version's "n" here is called "hww".
*/

#include <stdio.h>
#define TRUE 1
#define FALSE 0

void
median(arrin, arrout, arrlen, winlen, iprint)

        float arrin[], arrout[];
        int arrlen, winlen, iprint;
{
        float *sort, sorttemp;
        int *subs, substemp;
        int hww, iold, inew, jj, iout, i, key;
        int inok, ijunk;
        void mederrf();

        if(iprint) {
                fprintf(stdout, "\n Starting MEDIAN. Window length=%3i.", winlen);
                fprintf(stdout, "  Array length=%10i.\n", arrlen);
        }

        hww = (winlen - 1) / 2; /* "Half-window width", which in a zero-offset
                                   array also points to middle-point of window */

        /* Test for errors */
        inok = TRUE;

        if (winlen < 3) {
                fprintf(stderr, " Window length too small.  winlen=%5i\n", winlen);
                mederrf(&inok);
        }

        if (winlen > arrlen) {
                fprintf(stderr, " Window length greater than trace length.\n");
                mederrf(&inok);
        }

        /* Allocate window storage */
        sort = (float *)malloc(winlen * sizeof(float));
        subs = (int *)malloc(winlen * sizeof(int));

        if (((2 * hww) + 1) != winlen) {
                fprintf(stderr, " Window must be odd length.  winlen=%5i\n", winlen);
                mederrf(&inok);
        }

        /* Proceed if input parameters are OK */
        if (inok) {

                /* "Copy-on" ends of arrin where RMF is undefined */
                for(jj = 0 ; jj < hww ; jj++) {
                        arrout[jj] = arrin[jj];
                }
                for(jj = arrlen - hww ; jj < arrlen ; jj++) {
                        arrout[jj] = arrin[jj];
                }

                /* Fill sorting array for first time and bubble sort */
                for(i = 0 ; i < winlen ; i++) {
                        subs[i] = i;
                        sort[i] = arrin[i];
                }

                key = TRUE;
                while(key) {    /* Ascending-order bubble sort */
                        key = FALSE;
                        for(i = 1 ; i < winlen ; i++) {
                                if (sort[i - 1] > sort[i]) {
```

```
                        key = TRUE;
                        sorttemp = sort[i - 1];
                        sort[i - 1] = sort[i];
                        sort[i] = sorttemp;
                        substemp = subs[i - 1];
                        subs[i - 1] = subs[i];
                        subs[i] = substemp;
                    }
                }
            }

            /* Put result (middle point of "sort") into "arrout" */
            arrout[hww] = sort[hww];

            /* Thereafter, drop in only the new point from "arrin" to save sorting time */
            for(iout = hww + 1 ; iout < arrlen - hww ; iout++) {
                iold = iout - (hww + 1);        /* points to oldest element still in sort window */
                inew = iout + hww;              /* points to element to be added to sort window */

                for(i = 0 ; subs[i] != iold ; i++);     /* find old point in sort array */

                sort[i] = arrin[inew];          /* replace old point in sort array */
                subs[i] = inew;

                while(i < winlen - 1) {         /* move point up list */
                    if(sort[i] > sort[i + 1]) {
                        sorttemp = sort[i];
                        sort[i] = sort[i + 1];
                        sort[i + 1] = sorttemp;
                        substemp = subs[i];
                        subs[i] = subs[i + 1];
                        subs[i + 1] = substemp;
                        i++;
                    }
                    else {
                        break;
                    }
                }
                while(i > 0) {                  /* move point down list */
                    if(sort[i - 1] > sort[i]) {
                        sorttemp = sort[i - 1];
                        sort[i - 1] = sort[i];
                        sort[i] = sorttemp;
                        substemp = subs[i - 1];
                        subs[i - 1] = subs[i];
                        subs[i] = substemp;
                        i--;
                    }
                    else {
                        break;
                    }
                }

                /* Put result (middle point of "sort") into "arrout" */
                arrout[iout] = sort[hww];
            }
        }

        if(iprint) {
            fprintf(stdout, " Return from MEDIAN.\n\n");
        }

        /* Clean up */
        free((char *)sort);
        free((char *)subs);
}

/* Error message and flag for immediate exit */

void
mederrf(inok)

        int *inok;

{
        fprintf(stderr, " -----Error in subroutine MEDIAN-----\n");
        fprintf(stderr, "      (Time series returned unfiltered.)\n");
        *inok = FALSE;
}
```

```
#ifndef lint
static char rcsid[]="$Header: time.c,v 1.2 88/10/17 13:56:48 julian Exp $";
#endif
/*
 * Computations based on date and time of day
 * Bruce R. Julian, USGS Menlo Park, CA
 */
#include <math.h>
#include <local/local.h>
#include <local/date_time.h>

/* Convert date and time to TIME variable */
TIME
timvar(yr, mo, da, hr, mn, sec, caltyp)
        int     yr, mo, da, hr, mn;
        double  sec;
        int     caltyp;
{
        return (double)DAY*(jdn(yr, mo, da, caltyp)-BASEJDN) +
                HOUR*hr + MINUTE*mn + SECOND*sec;
}

/* Convert TIME variable to date and time */
void
datime(t, pyr, pmo, pda, phr, pmn, psec, caltyp)
        TIME    t;
        int     *pyr, *pmo, *pda, *phr, *pmn;
        double  *psec;
        int     caltyp;
{
        long    d;

        d = floor(t/DAY);
        date(d+BASEJDN, pyr, pmo, pda, caltyp);
        t -= d*DAY;
        *phr = t/HOUR;
        t -= (*phr)*HOUR;
        *pmn = t/MINUTE;
        *psec = t - (*pmn)*MINUTE;
}
```

```
#define LAMLAM 0
#define LAMMU  1
#define MUMU   2
#define LAMRHO 3
#define MURHO  4
#define RHORHO 5

typedef struct {
        float   vp, vs;         /* Average wave speeds                  */
        float   rho;            /* Average density                      */
        float   s2[6];          /* Zero-lag covariances                 */
        float   a[6][3];        /* Correlation distances                */
        double  (*bfunct)();    /* 3-D Fourier transform function       */
} SCATMDL;

double  bexp(), bgauss();
void    born_pp(), born_ps(), born_sp(), born_ss();
```

```
/*                              C M D _ O P T . H
 * Include file for command-line argument processing
 * Bruce R. Julian, USGS Menlo Park, Calif.
 */
#ifndef CMDOPT_H
#define CMDOPT_H
#include <local/stdtyp.h>

#ifndef DIM
#define DIM(x) (sizeof(x)/sizeof((x)[0]))       /* Dimension of array   */
#endif DIM

struct command {
        int   (*c_func)();          /* Function to be called for option     */
        char *c_name;               /* Name of option in argument list      */
        char *c_args;               /* Args (printed for documentation)     */
        char *c_comment;            /* Printed out for documentation        */
};

char            *aarg();
struct command  *findname();
int             isnarg();
double          narg();
char            *peekarg();
void            prt_help();
void            prt_doc();

int     eargc;                      /* Copy of argc                         */
char    **eargv;                    /* Copy of argv                         */
int     iarg;                       /* Argument index                       */
int     ncmd;                       /* No. of elements in cmd[]             */
char    *progname;                  /* For error messages                   */

#define PROCESS_OPTS { \
        progname = argv[0]; \
        eargc    = argc; \
        eargv    = argv; \
        ncmd     = DIM(cmd); \
        for (iarg=1; iarg<eargc && eargv[iarg][0]=='-'; iarg++) \
                XEQCMD(eargv[iarg]); \
}
#define XEQCMD(s)       (*findname(s)->c_func)()
#endif CMDOPT_H
```

```
/*
 *              c u i n t . h
 * Cubic, bicubic, and tricubic interpolation
 */
#ifndef CUINT_H
#define CUINT_H

typedef struct {        /* Two-dimensional result                       */
        float   f,                      /* Value                        */
                fx, fy,                 /* First derivatives            */
                fxx, fxy, fyy;          /* Second derivatives           */
} PT2;
typedef struct {        /* Three-dimensional result                     */
        float   f,                      /* Value                        */
                fx, fy, fz,             /* First derivatives            */
                fxx, fxy, fyy,          /* Second derivatives           */
                fxz, fyz, fzz;
} PT3;

void    cucof();
void    cuint();
void    bcucof();
void    bcuint();
void    tcucof();
void    tcuint();
#endif CUINT_H
```

```
/*            D A T E _ T I M E . H
 * Include file for calendar and time routines
 * Bruce R. Julian, USGS Menlo Park, Calif.
 */
#ifndef DATETIME_H
#define DATETIME_H

#define BASEJDN         2440588                 /* 1 January 1970       */

/* Calendar codes */
#define GREGORIAN       1                       /* Modern calendar      */
#define JULIAN          0                       /* Old-style calendar   */

/* Time intervals */
#define SECOND          1
#define MINUTE          (60*SECOND)
#define HOUR            (60*MINUTE)
#define DAY             (24*HOUR)
#define WEEK            (7*DAY)
#define YEAR            (365*DAY)
#define LEAPYEAR        (366*DAY)
#define MILLISEC        (1e-3*SECOND)
#define MICROSEC        (1e-6*SECOND)
#define NANOSEC         (1e-9*SECOND)

/* Data types */
typedef double  TIME;

/* External functions */
int     ckdate();
void    date();
bool    isleap();
long    jdn();
void    mnday();
TIME    timvar();
int     yrday();
#endif DATETIME_H
```

```
/*
 * /usr/include/local/dec_float.h
 */

typedef long DEC_FLOAT;
typedef struct dec_dble {
        long    h;
        long    l;
} DEC_DOUBLE;

double  cvdecf();
```

```
#ifndef DECODE_H
#ifndef MAXBUF
#define MAXBUF          20
#endif MAXBUF
char    dcbuf[MAXBUF];

char    *strncpy();

/* ASCII-to-number conversion (for Fortran-like formatted reading) */
/* char *s is beginning of field        */
/* int  j  is field length              */
/*      f is conversion function        */
#define DECODE(s, j, f) (dcbuf[j]='\0', f(strncpy(dcbuf, s, j)))
#define HASDECPT        index(dcbuf, '.')
#endif DECODE_H
```

```
/*                E A R T H C O N S T . H
 * Bruce R. Julian, USGS Menlo Park, Calif., 15 Dec 1983
 */
#ifndef EARTHCONST_H
#define EARTHCONST_H

#define REQUAT          6378.163        /* Equatorial radius            */
#define RPOLAR          6356.177        /* Polar radius                 */
#define REARTH          6371            /* Equal-volume sphere          */
#define FLATTENING      (1/298.24)      /* (REQUAT-RPOLAR)/REQUAT       */
#define KM              (5280*12*0.0000254)     /* Kilometers per mile          */
#define MILE            (1.0/KM)        /* Miles per kilometer          */
#endif
```

```
/*                  F 7 7 T Y P E S . H
 * C data types used by f77 compiler (installation-dependent)
 * Version for Integrated Solutions (4.2BSD) Unix
 * Bruce R. Julian, USGS Menlo Park, Calif.  2 Jan 1986
 */
#define F77INT          int             /* Default integer data type    */
#define F77LOG          int             /* Default logical data type    */
#define F77SLARG        long            /* String-length argument type  */
```

```
/*                  L O C A L . H
 * Bruce R. Julian, USGS Menlo Park Calif., Dec 15 1983
 */
#ifndef LOCAL_H
#define LOCAL_H

#include "stdtyp.h"

#ifndef NULL
#define NULL 0
#endif NULL

#define PRIVATE static                          /* Better name for it   */
#define FOREVER for(;;)                         /* Infinite loop        */
#define NOT !
#define YES 1                                   /* Truth value          */
#define NO 0                                    /* Truth value          */
#ifndef TRUE
#define TRUE 1                                  /* Truth value          */
#define FALSE 0                                 /* Truth value          */
#endif TRUE
#define DIM(x) (sizeof(x)/sizeof((x)[0]))       /* Dimension of array   */
#ifndef MAX
#define MAX(a,b) (((a)>(b)) ? (a) : (b))        /* Larger value         */
#endif MAX
#ifndef MIN
#define MIN(a,b) (((a)<(b)) ? (a) : (b))        /* Smaller value        */
#endif MIN
#define ABS(a) ((a) > 0 ? (a) : -(a))           /* Absolute value       */
#define FAIL -1                                 /* Function return      */
#define FAILED == -1
#define SUCCEED 0                               /* Function return      */
#define SUCCEEDED != -1
#define STDIN 0                                 /* read() write() unit  */
#define STDOUT 1                                /* read() write() unit  */
#define STDERR 2                                /* read() write() unit  */
#define READ 0                                  /* Mode for open()      */
#define WRITE 1                                 /* Mode for open()      */
#define UPDATE 2                                /* Mode for open()      */
#define HEAD 0                                  /* Whence for lseek()   */
#define HERE 1                                  /* Whence for lseek()   */
#define TAIL 2                                  /* Whence for lseek()   */
#define reg register
#define repeat do
#define until(c) while(!(c))
#define streq !strcmp                           /* String comparison    */
#define strneq !strncmp                         /* String comparison    */

#define ALLOC(x) (struct x *) malloc(sizeof(struct x))

#define TURN_ON(x, f)    (x |= (f))
#define TURN_OFF(x, f)   (x &= ~(f))
#define IS_ON(x, f)    (x & (f))
#define IS_OFF(x, f)    (!(x & (f)))

#endif LOCAL_H
```

```
/*            M A T H C O N S T . H
 */
#ifndef MATHCONST_H
#define MATHCONST_H

/* PI and its relatives */
#define PI      3.14159265358979324
#define PIINV   0.318309886183790672            /* 1./PI        */
#define HALFPI  1.57079632679489662
#define TWOPI   6.2831853071795864
#define FOURPI  12.5663706143591730
#define SQRTPI  1.77245385090551603
#define PI32    5.56832799683170785             /* PI**1.5      */

/* Irrational square roots */
#define SQRT2   1.41421356237309505
#define SQRT3   1.73205080756887729
#define SQRTHALF        0.707106781186547524

/* Logarithms */
#define LN2     0.693147180559945310
#define LN10    2.30258509299404568
#define LOG2    0.301029995663981195
#define LOGE    0.434294481903251828

/* Degrees per radian and vice-versa */
#define DEG     57.2957795130823209
#define RAD     0.0174532925199432957

/* One degree in radians, a la Mathematica */
#define Degree  0.0174532925199432957

/* Floating-point resolution (machine-dependent) */
#define EPS     5.96046448e-08
#define DEPS    1.38777878e-17

/* Euler's constant */
#define E_GAMMA 0.577215664901532860606

/* Array subscripts */
#define X 0
#define Y 1
#define Z 2
#define XX 0
#define XY 1
#define YX 1
#define YY 2
#define XZ 3
#define ZX 3
#define YZ 4
#define ZY 4
#define ZZ 5
#endif
```

```
/*
 *                          m o d e l 3 d . h
 */
#ifndef MODEL3D_H
#define MODEL3D_H
#ifndef NULL
#define NULL 0
#endif NULL

typedef struct {
        struct lim {
                float   lat0, lon0, rot;
                int     nx, ny, nz;
        } limits;
        float   *xgrid;         /* [0..nx-1]                                */
        float   *ygrid;         /* [0..ny-1]                                */
        float   *zgrid;         /* [0..nz-1]                                */
        float   ***s;           /* [0..nx-1][0..ny-1][0..nz-1]              */
        float   ***sx;          /* [0..nx-1][0..ny-1][0..nz-1]              */
        float   ***sy;          /* [0..nx-1][0..ny-1][0..nz-1]              */
        float   ***sz;          /* [0..nx-1][0..ny-1][0..nz-1]              */
        float   ***sxy;         /* [0..nx-1][0..ny-1][0..nz-1]              */
        float   ***sxz;         /* [0..nx-1][0..ny-1][0..nz-1]              */
        float   ***syz;         /* [0..nx-1][0..ny-1][0..nz-1]              */
        float   ***sxyz;        /* [0..nx-1][0..ny-1][0..nz-1]              */
} MODEL3D;

typedef struct {        /* A point within the earth                         */
        float   s,              /* Wave slowness                            */
                sx, sy, sz,     /* Spatial derivatives                      */
                sxx, syy, szz,  /* Second derivatives                       */
                sxy, sxz, syz;
} EPOINT;

void    free_mdl();
void    sder();
MODEL3D *rdmdl();
#endif MODEL3D_H
```

```
/*                    Q E R R . H
 * Bruce R. Julian, USGS Menlo Park Calif., 15 Dec. 1983
 */
#ifndef QERR_H
#define QERR_H

#define ERRBASE         1000
#define EBADDAY         (ERRBASE +  1)  /* Illegal day                    */
#define EBADMONTH       (ERRBASE +  2)  /* Illegal month                  */
#define EBADYEAR        (ERRBASE +  3)  /* Illegal year                   */
#define EBADVALUE       (ERRBASE +  4)  /* Illegal value                  */
#define EFEWDATA        (ERRBASE +  5)  /* Not enough data                */
#define ENOCONVRG       (ERRBASE +  6)  /* Convergence Failure            */
#define EUNDERFLOW      (ERRBASE +  7)  /* Underflow                      */
#define ENOSTATION      (ERRBASE +  8)  /* No such station                */
#define ESIZE           (ERRBASE +  9)  /* Inappropriate structure size */
#define ESINGULAR       (ERRBASE + 10)  /* Singular matrix                */
#define ETOODEEP        (ERRBASE + 11)  /* Focal depth too great          */
#define EBADPHASE       (ERRBASE + 12)  /* No such phase                  */
#define ENOFEAS         (ERRBASE + 13)  /* Constraints are inconsistent */
#define EINFIN          (ERRBASE + 14)  /* Solution is unbounded          */
#define ENOTFOUND       (ERRBASE + 15)  /* Can't find desired record    */
#define ENOCUROR        (ERRBASE + 16)  /* No current origin              */
#define ENOCURAM        (ERRBASE + 17)  /* No current average magnitude */
#define ECANTBRACKET    (ERRBASE + 18)  /* Root not bracketed             */
#endif QERR_H
```

```
/*                  Q U A K E . H
 * Bruce R. Julian, USGS Menlo Park Calif., Dec 15 1983
 */
#ifndef QUAKE_H
#define QUAKE_H

#include <local/stdtyp.h>
#include <local/date_time.h>

#ifndef NULL
#define NULL 0
#endif

#define repeat do
#define until(c) while(!(c))

#define FOREVER for(;;)                         /* Infinite loop        */
#define NOT !
#define YES 1                                   /* Truth value          */
#define NO 0                                    /* Truth value          */
#define TRUE 1                                  /* Truth value          */
#define FALSE 0                                 /* Truth value          */
#define DIM(x) (sizeof(x)/sizeof((x)[0]))       /* Dimension of array   */
#ifndef MAX
#define MAX(a,b) ((a) > (b) ? (a) : (b))        /* Larger value         */
#endif MAX
#ifndef MIN
#define MIN(a,b) ((a) < (b) ? (a) : (b))        /* Smaller value        */
#endif MIN
#define ABS(a) ((a) > 0 ? (a) : -(a))           /* Absolute value       */
#define FAIL -1                                 /* Function return      */
#define SUCCEED 0                               /* Function return      */
#define STDIN 0                                 /* read() write() unit  */
#define STDOUT 1                                /* read() write() unit  */
#define STDERR 2                                /* read() write() unit  */
#define READ 0                                  /* Mode for open()      */
#define WRITE 1                                 /* Mode for open()      */
#define UPDATE 2                                /* Mode for open()      */

/*---------------------- E R R O R   C O D E S -------------------------*/
#define ERRBASE         1000
#define EBADDAY         (ERRBASE + 1)   /* Illegal calendar day         */
#define EBADMONTH       (ERRBASE + 2)   /* Illegal calendar month       */
#define EBADYEAR        (ERRBASE + 3)   /* Illegal calendar year        */
#define EBADVALUE       (ERRBASE + 4)   /* Illegal value                */
#define EFEWDATA        (ERRBASE + 5)   /* Not enough data              */
#define ENOCONVRG       (ERRBASE + 6)   /* Failed to converge           */
#define EUNDERFLOW      (ERRBASE + 7)   /* Arithmetic underflow         */
#define ENOSTATION      (ERRBASE + 8)   /* Unknown seismograph station  */
#define ESIZE           (ERRBASE + 9)
#define ESINGULAR       (ERRBASE + 10)  /* Matrix is singular           */
#define ETOODEEP        (ERRBASE + 11)  /* Focal depth too great        */
#define EBADPHASE       (ERRBASE + 12)  /* Unknown seismic phase code   */

/*------------- E V E N T   O R I G I N   S T R U C T U R E ------------*/
typedef struct {
        TIME    o_time;         /* Origin time                          */
        float   o_lat;          /* Epicentral latitude (Rad)            */
        float   o_lon;          /* Epicentral longitude (Rad)           */
        float   o_depth;        /* Focal depth                          */
} ORIGIN;

/*------------------ S T A T I O N   S T R U C T U R E -----------------*/
#define MXSTCOD 4
typedef struct {
        char    s_code[MXSTCOD];/* Station code                         */
        float   gdc[3];         /* Geocentric direction cosines         */
        float   s_elev;         /* Station elevation (m)                */
} STATION;

/*------------------ R E A D I N G   S T R U C T U R E -----------------*/
#define CHANSIZE        6
#define MXPHCOD         8
/* Masks for flags: */
#define ASSOCIATED      01      /* Phase comes from current event       */
#define T_EXISTS        02      /* Arrival time exists                  */
#define USE_T           04      /* Should use arrival time              */
#ifdef SLO
#define S_EXISTS        010     /* Slowness-azimuth exists              */
#define USE_S           020     /* Should use slowness-azimuth          */
#endif
typedef struct {
        STATION sta;            /* Station structure                    */
        char    chan[CHANSIZE]; /* Channel code                         */
        char    phcode[MXPHCOD];/* Phase code                           */
        TIME    atime;          /* Arrival time                         */
        float   setime;         /* Std. err. of arrival time (s)        */
        char    onset;          /* 'e' or 'i'                           */
```

```
        char    pol;            /* Polarity: e.g. 'u' or 'd'            */
        char    qual;           /* Quality code                         */
        float   amp;
        float   freq;
#ifdef SLO
        float   slowness;       /* Slowness (s/Rad)                     */
        float   azimuth;        /* Propagation azimuth (Rad)            */
        float   seslow;         /* Std. err. of slowness (s/Rad)        */
#endif
        char    flags;
} READING;

#ifdef SLO
/* Array subscripts for slowness-vector components */
#define EAST    0
#define NORTH   1
#endif

/*-------------- A S S O C I A T I O N  S T R U C T U R E --------------*/
typedef struct {
        float   delta;          /* Epicentral distance (Rad)            */
        float   aze;            /* Epicenter -> station azimuth (Rad)   */
        float   azs;            /* Station -> epicenter azimuth (Rad)   */
        float   s1[3];          /* Slowness vector at hypocenter        */
        float   s2[3];          /* Slowness vector at station           */
        int     brnum;          /* Branch number                        */
        int     ercode;         /* From body-wave subroutine            */
        float   atresid;        /* Arrival-time residual                */
        float   atsigma;        /* Arrival-time std. err.               */
        float   atwt;           /* Arrival-time weight                  */
        float   atimp;          /* Arrival-time importance              */
#ifdef SLO
        float   szresid[2];     /* slowness-azimuth residual (s/Rad)    */
        float   szsigma;        /* Std. err. of slowness-azimuth (s/Rad)*/
        float   szwt;           /* Weight of slowness-azimuth obs.      */
        float   szimp;          /* Importance of slowness-azimuth obs.  */
#endif
} ASSOC;

/*------------ R E L O C  C O N T R O L  S T R U C T U R E ------------*/
typedef struct {
        tbool   hold[4];        /* Cordinate-restraint flags            */
        float   epcvthresh;     /* Convergence threshold for epicenter  */
        float   fdcvthresh;     /* Convergence threshold for focal depth*/
        float   otcvthresh;     /* Convergence threshold for origin time*/
        float   min_depth;      /* Smallest allowed focal depth         */
        float   max_depth;      /* Largest allowed focal depth          */
        int     max_iter;       /* Iteration limit                      */
        int     max_damp;       /* Damping-step limit                   */
        float   lstart;         /* Initial damping parameter            */
        float   lmin;           /* Minimum damping parameter            */
        float   lfact;          /* Change in lambda per step            */
        float   zdamp;          /* Relative depth damping               */
        float   mu;             /* Frequency of erratic data            */
        int     iprint;         /* Print-control flag                   */
} RLCTRL;

/* Subscripts for origin components */
#define ILAT    0
#define ILON    1
#define IDEPTH  2
#define ITIME   3

/* Print control */
#define NEVER   0
#define FINAL   1
#define ALL     2

/*-------- O R I G I N  S T A T I S T I C S  S T R U C T U R E -------*/
typedef struct {
        float   chisq;          /* Chi squared                          */
        float   sumwt;          /* Sum of weights of readings           */
        int     ndf;            /* No. of degrees of freedom            */
        int     iter;           /* No. of iterations performed          */
        int     ntasoc;         /* No. of associated arrival-time obs.  */
        int     ntused;         /* No. of arrival-time obs. used        */
#ifdef SLO
        int     nsasoc;         /* No. of associated slowness obs.      */
        int     nsused;         /* No. of slowness obs. used            */
#endif
        tbool   converged;
        tbool   held[4];
} STATIS;

/*-------- B O D Y - W A V E  R E S U L T  S T R U C T U R E ---------*/
typedef struct {
        int     bw_brnum;       /* Branch number                        */
        float   bw_t;           /* Travel time                          */
```

```
        float   bw_set;         /* Std. err. of t                       */
        float   bw_s1[3];       /* Slowness vector at hypocenter        */
        float   bw_s2[3];       /* Slowness vector at station           */
        float   bw_tdd;         /* dp/ddelta                            */
        float   bw_tdh;         /* dp/dh                                */
} BWRSLT;
#endif
```

```
/*                    S T A B L E . H
 * USGS Calnet seismograph-station tables
 */
#ifndef STABLE_H
#define STABLE_H

#define CSIZE   4

/* Structure describing open station file */
typedef struct stable {
        char            *idsta;         /* List of station codes        */
        unsigned        nsta;           /* No. of codes in idsta        */
        FILE            *fp;            /* Station file                 */
        long            offset;         /* Offset of info. of first sta.*/
} STABLE;

/* Structure containing info. for one station */
typedef struct stinfo {
        char    code[CSIZE];            /* Station code                 */
        float   lat, lon;               /* Coordinates                  */
        int     ielev;                  /* Elevation (meters)           */
        float   p_anom;                 /* P-time anomaly               */
#if 0
        float   s_anom;                 /* S-time anomaly               */
#endif 0
        float   dmag_anom;              /* Duration-magnitude anomaly   */
        float   amag_anom;              /* Amplitude-magnitude anomaly  */
        int     inst;                   /* Instrument code              */
        float   t_dflt;                 /* Default period for amp_mag   */
        float   cal_dflt;               /* Calibration for amp_mag      */
} STINFO;

STABLE  *stopen();
void    stclose();
int     stindex();
int     stget();
#endif STABLE_H
```

```
/*              S T D T Y P . H
 * Version for PDP-11, VAX-11, and ISI V24
 */
#ifndef STDTYP_H
#define STDTYP_H

/* Specify machine characteristics */
#ifdef sun
#define USHORT
#define VOID    /* Compiler supports void type */
#define UTINY   /* Compiler supports unsigned char type */
#endif sun
#ifdef mc68000  /* Integrated Solutions UNIX */
#define USHORT  /* Compiler supports unsigned short type */
#define VOID
#endif mc68000
#ifdef vax
#define USHORT  /* Compiler supports unsigned short type */
#define UTINY   /* Compiler supports unsigned char type */
#define VOID    /* Compiler supports void type */
#endif vax
#ifdef pdp11
#define USHORT  /* Compiler supports unsigned short type */
#ifdef makedev  /* Crock to see if <sys/types.h> has been included */
#define VOID    /* <sys/types.h> has "typedef short void" */
#endif makedev
#endif pdp11

typedef char            tbits, tbool;
typedef int             bool;
typedef unsigned int    sizetype;
typedef long            lbits;
typedef short           bits, metachar;
#ifdef USHORT
#ifndef sun
typedef unsigned short  ushort;
#endif sparc
#undef  USHORT
#else USHORT
typedef unsigned ushort;                /* Assumes 16-bit machine */
#endif USHORT
#ifdef TINY
typedef char    tiny;
#undef  TINY
#define TINY(n) (char)(n)
#else TINY
typedef char    tiny;
#define TINY(n) (char)(((n) & 0x80) ? (~0xF7 | (n)) : (n))
#endif TINY
#ifdef  UTINY
typedef unsigned char   utiny;
#undef  UTINY
#define UTINY(n) (unsigned char)(n)
#else UTINY
typedef char    utiny;
#define UTINY(n) (unsigned)((n) & 0xFF)
#endif UTINY
#ifndef VOID
#define void int
#endif VOID
#define LURSHIFT(n,b)   (((long)(n) >> (b)) & (0x7FFFFFFF >> (b-1)))
#endif STDTYP_H
```