

```

/* normalscan.c
   =====
   Author: R.J.Barnes
*/

#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <string.h>
#include <time.h>
#include "rtypes.h"
#include "option.h"
#include "rtime.h"
#include "limit.h"
#include "radar.h"
#include "rprm.h"
#include "iqdata.h"
#include "rawdata.h"
#include "fitblk.h"
#include "fitdata.h"
#include "fitacf.h"

#include "taskid.h"
#include "errlog.h"
#include "rmsg.h"
#include "radarshell.h"
#include "rmsgsnd.h"
#include "tsg.h"
#include "tsgtable.h"
#include "maketsg.h"
#include "global.h"
#include "setup.h"
#include "reopen.h"
#include "tmseq.h"
#include "build.h"
#include "sync.h"
#include "interface.h"
#include "hdw.h"

#define SCHEDULER "schedule"
#define ERRLOG "errlog"

#define CONTROL_NAME "control_program"
#define TASK_NAMES
"echo_data", "iqwrite", "rawacfwrite", "fitacfwrite"

```

```

char cmdlne[1024];
char progid[80]="normalscan.c";
char progame[256];
struct TaskID *errlog;

char *tasklist[]=
{ TASK_NAMES,
  0};

int arg=0;
struct OptionData opt;

int main(int argc,char *argv[]) {

  int ptab[8] = {0,14,22,24,27,31,42,43};

  int lags[LAG_SIZE][2] = {
    { 0, 0},          /* 0 */
    {42,43},         /* 1 */
    {22,24},         /* 2 */
    {24,27},         /* 3 */
    {27,31},         /* 4 */
    {22,27},         /* 5 */

    {24,31},         /* 7 */
    {14,22},         /* 8 */
    {22,31},         /* 9 */
    {14,24},         /* 10 */
    {31,42},         /* 11 */
    {31,43},         /* 12 */
    {14,27},         /* 13 */
    { 0,14},         /* 14 */
    {27,42},         /* 15 */
    {27,43},         /* 16 */
    {14,31},         /* 17 */
    {24,42},         /* 18 */
    {24,43},         /* 19 */
    {22,42},         /* 20 */
    {22,43},         /* 21 */
    { 0,22},         /* 22 */

    { 0,24},         /* 24 */

    {43,43}};       /* alternate lag-0 */

```

```

char *sname=NULL;
char *ename=NULL;
char *sdname={SCHEDULER};
char *edname={ERRLOG};
char logtxt[1024];

int n;
pid_t sid;
int exitpoll=0;

int scnsc=120;
int scnus=0;
int skip;
int cnt=0;

unsigned char fast=0;
unsigned char discretion=0;

strcpy(cmdlne,argv[0]);
for (n=1;n<argc;n++) {
    strcat(cmdlne," ");
    strcat(cmdlne,argv[n]);
}

strncpy(combf,progid,80);
OpsSetupCommand(argc,argv);
OpsSetupRadar();
OpsSetupShell();

RadarShellParse(&rstable,"sbm l ebm l dfrq l nfrq l dfrang l
nfrang l dmpinc l nmpinc l frqrng l xcnt l",
                &sbm,&ebm,
                &dfrq,&nfrq,
                &dfrang,&nfrang,
                &dmpinc,&nmpinc,
                &frqrng,&xcnt);

cp=150;
intsc=7;
intus=0;
mppul=8;
mplgs=23;
mpinc=1500;
dmpinc=1500;
nrang=75;

```

```

rsep=45;
txpl=300;

SiteStart();

/* ===== PROCESS COMMAND LINE ARGUMENTS ===== */

OptionAdd(&opt,"di",'x',&discretion);

OptionAdd(&opt,"el",'t',&ename);
OptionAdd(&opt,"sc",'t',&sname);

OptionAdd(&opt,"frang",'i',&frang);
OptionAdd(&opt,"rsep",'i',&rsep);

OptionAdd( &opt, "dt", 'i', &day);
OptionAdd( &opt, "nt", 'i', &night);
OptionAdd( &opt, "df", 'i', &dfrq);
OptionAdd( &opt, "nf", 'i', &nfrq);
OptionAdd( &opt, "xcf", 'i', &xcnt);

OptionAdd(&opt,"fast",'x',&fast);

arg=OptionProcess(1,argc,argv,&opt,NULL);

if (sname==NULL) sname=sdname;
if (ename==NULL) ename=edname;

sid=RShellRegister(sname,CONTROL_NAME);

errlog=TaskIDMake(ename);
OpsLogStart(errlog,progname,argc,argv);

SiteSetupHardware();

if (fast) {
    cp=151;
    scnsc=60;
    scnus=0;
    intsc=3;
    intus=0;
}

```

```

}

if ( discretion) cp= -cp;

txpl=(rsep*20)/3;

if (fast) sprintf(progname,"normalscan (fast)");
else sprintf(progname,"normalscan");

OpsFitACFStart();

OpsSetupTask(tasklist);
for (n=0;n<tnum;n++) {
    RMsgSndReset(tlist[n]);
    RMsgSndOpen(tlist[n],strlen(cmdlne),cmdlne);
}

do {

    if (SiteStartScan()==0) continue;

    if (OpsReOpen(2,0,0) !=0) {
        ErrLog(errlog,progname,"Opening new files.");
        for (n=0;n<tnum;n++) {
            RMsgSndClose(tlist[n]);
            RMsgSndOpen(tlist[n],strlen(cmdlne),cmdlne);
        }
    }

    scan=1;

    ErrLog(errlog,progname,"Starting scan.");

    if (xcnt>0) {
        cnt++;
        if (cnt==xcnt) {
            xcf=1;
            cnt=0;
        } else xcf=0;
    } else xcf=0;

    skip=OpsFindSkip(scnsc,scnus);

    if (backward) {
        bmnum=sbm-skip;
        if (bmnum<ebm) bmnum=sbm;
    }
}

```

```

} else {
    bmnum=sbm+skip;
    if (bmnum>ebm) bmnum=sbm;
}

do {

    TimeReadClock(&yr, &mo, &dy, &hr, &mt, &sc, &us);

    if (OpsDayNight()==1) {
        stfrq=dfrq;
        mpinc=dmpinc;
        frang=dfrang;
    } else {
        stfrq=nfrq;
        mpinc=nmpinc;
        frang=nfrang;
    }

    sprintf(logtxt, "Integrating beam:%d intt:%ds.%dus
(%d:%d:%d:%d)", bmnum,
                intsc, intus, hr, mt, sc, us);
    ErrLog(errlog, progname, logtxt);

    ErrLog(errlog, progname, "Setting beam.");

    SiteSetIntt(intsc, intus);
    SiteSetBeam(bmnum);

    ErrLog(errlog, progname, "Doing clear frequency search.");

    sprintf(logtxt, "FRQ: %d %d", stfrq, frqrng);
    ErrLog(errlog, progname, logtxt);

    if (SiteFCLR(stfrq, stfrq+frqrng)==FREQ_LOCAL)
        ErrLog(errlog, progname, "Frequency Synthesizer in local
mode.");

    SiteSetFreq(tfreq);

    sprintf(logtxt, "Transmitting on: %d (Noise=%g)", -
tfreq, noise);

    ErrLog(errlog, progname, logtxt);

    tsgid=SiteTimeSeq(ptab);

```

```

nave=SiteIntegrate(lags);
if (nave<0) {
    sprintf(logtxt,"Integration error:%d",nave);
    ErrLog(errlog,programe,logtxt);
    continue;
}
sprintf(logtxt,"Number of sequences: %d",nave);
ErrLog(errlog,programe,logtxt);

OpsBuildPrm(&prm,ptab,lags);
OpsBuildIQ(&iq);
OpsBuildRaw(&raw);

FitACF(&prm,&raw,&fblk,&fit);

ErrLog(errlog,programe,"Sending messages.");

msg.num=0;
msg.tsize=0;

RMsgSndAdd(&msg,sizeof(struct RadarParm),(unsigned char *)
&prm,
    PRM_TYPE,0);

RMsgSndAdd(&msg,sizeof(struct IQData),(unsigned char *)
&iq,
    IQ_TYPE,0);

RMsgSndAdd(&msg,strlen(sharedmemory)+1,sharedmemory,
    IQS_TYPE,0);

RMsgSndAdd(&msg,sizeof(struct RawData),(unsigned char *)
&raw,
    RAW_TYPE,0);

RMsgSndAdd(&msg,sizeof(struct FitData),(unsigned char *)
&fit,
    FIT_TYPE,0);
RMsgSndAdd(&msg,strlen(programe)+1,programe,
    NME_TYPE,0);

for (n=0;n<tnum;n++) RMsgSndSend(tlist[n],&msg);

```

```
    ErrLog(errlog,programe,"Polling for exit.");

    exitpoll=RadarsShell(sid,&rstable);
    if (exitpoll !=0) break;
    scan=0;
    if (bmnum==ebm) break;
    if (backward) bmnum--;
    else bmnum++;

} while (1);
ErrLog(errlog,programe,"Waiting for scan boundary.");

if (exitpoll==0) OpsWaitBoundary(scnsc,scnus);

} while (exitpoll==0);
SiteEnd();
for (n=0;n<tnum;n++) RMsgSndClose(tlist[n]);
ErrLog(errlog,programe,"Ending program.");
RShellTerminate(sid);
return 0;
}
```