Appendix D. C Language Program to Read an Archive Record

```
#include <stdio.h>
#include <string.h>

void open_archive(char *);
void close_archive(void);
void unpackSem2(short *);

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

    FILE *fout;
    char arcfile[80],outfile[80];
    short iend = 0;
    short i,j,rnum;
    extern struct archive_rec {
        long cSum;
        long ihd[4][6];
        short cSumFlag;
        short major;
        short status[10];
        short qual[16];
        short minor[16];
        short mdf[16][40];
        float analog[17];
        float head[4][27];
        float ssLoc[16][2];
        float mep0[16][9];
        float mep90[16][9];
        float mepOmni[16][4];
        float ted0[16][8];
        float ted30[16][8];
        float ted0s[4][8];
        float ted30s[4][8];
        float tedback[4][2];
        float tedfx[16][7];
    } rec;

    if (argc < 3) {
        printf("Arguments: 1) Input archive file name\n");
        printf("           2) Output file name\n");
        exit(0);
    }

    strcpy(arcfile,argv[1]);
    strcpy(outfile,argv[2]);

    /* printf("Archive file = %s\n",arcfile); */

    /* Open the output file */

    if (( fout = fopen(outfile,"w"))==NULL) {
        printf("Cannot open %s -- aborting.\n",outfile);
        exit(0);
    }
```

```
    /* Open the archive file */

    open_archive(arcfile);
    rnum = 0;

 /* Daily POES archive data file has 2700 32-second records if
    it has a complete day */

 while((rnum < 3000) && (iend == 0)){

    unpackSem2(&iend);

    if(iend == 0) {

    fprintf(fout,"ihd:\n");
    for(i=0; i<4; i++) {
        fprintf(fout,"%d %d %4d %3d %8ld %4d %5d\n",
           i,rec.ihd[i][0],rec.ihd[i][1],rec.ihd[i][2],
           rec.ihd[i][3],rec.ihd[i][4],rec.ihd[i][5]);

    }
    fprintf(fout,"ssLoc:\n");
    for(i=0; i<16; i++) {
        fprintf(fout,"%d %6.2f
%6.2f\n",i,rec.ssLoc[i][0],rec.ssLoc[i][1]);
    }

    fprintf(fout,"status:\n");
    for(i=0; i<10; i++) fprintf(fout,"%4d ",rec.status[i]);
    fprintf(fout,"\n");
    fprintf(fout,"qual:\n");
    for(i=0; i<16; i++) fprintf(fout,"%3d ",rec.qual[i]);
    fprintf(fout,"\n");
    fprintf(fout,"minor:\n");
    for(i=0; i<16; i++) fprintf(fout,"%4d ",rec.minor[i]);
    fprintf(fout,"\n");
    fprintf(fout,"mdf:\n");
    for(i=0; i<40; i++) {
        fprintf(fout,"%3d ",i);
        for(j=0; j<16; j++)
            fprintf(fout,"%2d ",rec.mdf[j][i]);
        fprintf(fout,"\n");
    }
    fprintf(fout,"analog:\n");
    for(i=0; i<8; i++) fprintf(fout,"%8.2f ",rec.analog[i]);
    fprintf(fout,"\n");
    for(i=8; i<16; i++) fprintf(fout,"%8.2f ",rec.analog[i]);
    fprintf(fout,"\n");
    fprintf(fout,"head:\n");
    for(j=0; j<27; j++)
        fprintf(fout,"%3d %10.2f %10.2f %10.2f %10.2f\n",
            j,rec.head[0][j],rec.head[1][j],rec.head[2][j],
            rec.head[3][j]);

    fprintf(fout,"mep0:\n");
    for(j=0; j<16; j++) {
```

```
    fprintf(fout,"%2d ",j);
    for(i=0; i<9; i++) fprintf(fout,"%8.2f ",rec.mep0[j][i]);
    fprintf(fout,"\n");
}
fprintf(fout,"mep90:\n");
for(j=0; j<16; j++) {
    fprintf(fout,"%2d ",j);
    for(i=0; i<9; i++) fprintf(fout,"%8.2f ",rec.mep90[j][i]);
    fprintf(fout,"\n");
}
fprintf(fout,"mepOmni:\n");
for(j=0; j<16; j++) {
    fprintf(fout,"%2d %10.2f %10.2f %10.2f %10.2f\n",
        j,rec.mepOmni[j][0],rec.mepOmni[j][1],
        rec.mepOmni[j][2],rec.mepOmni[j][3]);
}

fprintf(fout,"ted0:\n");
for(j=0; j<16; j++) {
    fprintf(fout,"%2d ",j);
    for(i=0; i<8; i++) fprintf(fout,"%8.2f ",rec.ted0[j][i]);
    fprintf(fout,"\n");
}
fprintf(fout,"ted30:\n");
for(j=0; j<16; j++) {
    fprintf(fout,"%2d ",j);
    for(i=0; i<8; i++) fprintf(fout,"%8.2f ",rec.ted30[j][i]);
    fprintf(fout,"\n");
}

fprintf(fout,"ted0s:\n");
for(j=0; j<4; j++) {
    fprintf(fout,"%2d ",j);
    for(i=0; i<8; i++) fprintf(fout,"%8.2f ",rec.ted0s[j][i]);
    fprintf(fout,"\n");
}
fprintf(fout,"ted30s:\n");
for(j=0; j<4; j++) {
    fprintf(fout,"%2d ",j);
    for(i=0; i<8; i++) fprintf(fout,"%8.2f ",rec.ted30s[j][i]);
    fprintf(fout,"\n");
}
fprintf(fout,"tedback:\n");
for(j=0; j<4; j++) {
    fprintf(fout,"%2d ",j);
    for(i=0; i<2; i++) fprintf(fout,"%8.2f ",rec.tedback[j][i]);
    fprintf(fout,"\n");
}
 fprintf(fout,"tedfx:\n");
for(j=0; j<16; j++) {
    fprintf(fout,"%2d ",j);
    for(i=0; i<7; i++) fprintf(fout,"%8.2f ",rec.tedfx[j][i]);
    fprintf(fout,"\n");
}

fprintf(fout,"========================================\n");
rnum = rnum+1;
```

```
    } /* End if */

} /* End do */
    printf("Records read: %d\n",rnum--);
    close_archive();
    close(fout);
    printf("Done\n");

}



/* Function: unpackSem2.c

    unpackSem2.c returns a NOAA POES SEM2 data record from an
                already opened file in packed binary format.

    Calling from Fortran:  It is recommended that Fortran users use the
                           Fortran 77 subroutine: readArcSub.f instead.

    Compilation: c89 -c unpackSem2.c

    Programmer:  Sue Greer

    Note: variables of type long are 4-byte integers, short are 2-byte
          integers.

*/

#include <stdio.h>
#include <string.h>
#include <sys/types.h>
#include <netinet/in.h>

static FILE *fp;

struct archive_rec {
    long cSum;
    long ihd[4][6];
    short cSumFlag;
    short major;
    short status[10];
    short qual[16];
    short minor[16];
    short mdf[16][40];
    float analog[17];
    float head[4][27];
    float ssLoc[16][2];
    float mep0[16][9];
    float mep90[16][9];
    float mepOmni[16][4];
    float ted0[16][8];
    float ted30[16][8];
    float ted0s[4][8];
    float ted30s[4][8];
```

```
    float tedback[4][2];
    float tedfx[16][7];
} rec;

void unpackSem2 (short *IEND) {

    extern FILE *fp;
    unsigned char arc[2544];
    long i,j,jj,k,m,n,mf,mo,mp0,mp90,qf,tf;
    long b1,b2,b3;
    float cf = .0001;
    float cnvrt[] = {
  0.0,1.0,2.0,3.0,4.0,5.0,6.0,7.0,8.0,9.0,10.0,11.0,
12.0,13.0,14.0,15.0,16.0,17.0,18.0,19.0,20.0,21.0,22.0,23.0,24.0,25.0,
26.0,27.0,28.0,29.0,30.0,31.0,32.0,34.5,36.5,38.5,40.5,42.5,44.5,
46.5,48.5,50.5,53.0,56.0,59.0,62.0,65.5,69.5,73.5,77.5,81.5,85.5,
89.5,93.5,97.5,101.5,106.5,112.5,118.5,124.5,131.5,139.5,147.5,155.5,
163.5,171.5,179.5,187.5,195.5,203.5,213.5,225.5,237.5,249.5,263.5,
279.5,295.5,311.5,327.5,343.5,359.5,375.5,391.5,407.5,427.5,451.5,
475.5,499.5,527.5,559.5,591.5,623.5,655.5,687.5,719.5,751.5,783.5,
815.5,855.5,903.5,951.5,999.5,1055.5,1119.5,1183.5,1247.5,1311.5,
1375.5,1439.5,1503.5,1567.5,1631.5,1711.5,1807.5,1903.5,1999.5,2111.5,
2239.5,2367.5,2495.5,2623.5,2751.5,2879.5,3007.5,3135.5,3263.5,3423.5,
3615.5,3807.5,3999.5,4223.5,4479.5,4735.5,4991.5,5247.5,5503.5,5759.5,
6015.5,6271.5,6527.5,6847.5,7231.5,7615.5,7999.5,8447.5,8959.5,9471.5,
9983.5,10495.5,11007.5,11519.5,12031.5,12543.5,13055.5,13695.5,14463.5,
15231.5,15999.5,16895.5,17919.5,18943.5,19967.5,20991.5,22015.5,23039.5,
24063.5,25087.5,26111.5,27391.5,28927.5,30463.5,31999.5,33791.5,35839.5,
37887.5,39935.5,41983.5,44031.5,46079.5,48127.5,50175.5,52223.5,54783.5,
57855.5,60927.5,63999.5,67583.5,71679.5,75775.5,79871.5,83967.5,88063.5,
92159.5,96255.5,100351.5,104447.5,109567.5,115711.5,121855.5,127999.5,
135167.5,143359.5,151551.5,159743.5,167935.5,176127.5,184319.5,192511.5,
200703.5,208895.5,219135.5,231423.5,243711.5,255999.5,270335.5,286719.5,
303103.5,319487.5,335871.5,352255.5,368639.5,385023.5,401407.5,417791.5,
438271.5,462847.5,487423.5,511999.5,540671.5,573439.5,606207.5,638975.5,
671743.5,704511.5,737279.5,770047.5,802815.5,835583.5,876543.5,925695.5,
974847.5,1023999.5,1081343.5,1146879.5,1212415.5,1277951.5,1343487.5,
1409023.5,1474559.5,1540095.5,1605631.5,1671167.5,1753087.5,1851391.5,
    1949695.5,1998848.0
    };

    b1 = 256*256*256;
    b2 = 256*256;
    b3 = 256;

    fread(arc, sizeof(unsigned char),2544,fp);
    if(feof(fp)) {
        printf("End of file\n");
        *IEND = 1;
    }
    if(ferror(fp)) {
        printf("File read error\n");
        *IEND = 2;
    }

    /* A 32-second record was read, now unpack it*/
```

```
/*for(i=0; i<215; i++)
    printf ("%d = %d\n",i,arc[i]); */

/* Checksum flag */
rec.cSumFlag = arc[0]*b1 + arc[1]*b2 + arc[2]*b3 + arc[3];
/* Checksum */
rec.cSum = arc[4]*b1 + arc[5]*b2 + arc[6]*b3 + arc[7];
/* Major Frame Number */
rec.major = arc[8]*b3 + arc[9];
/* Status: instrument on/off, etc. */
for(i=0; i<10; i++) rec.status[i] = arc[i+10];
/* Analog data, etc. */
for(i=0; i<68; i+=4) {
    k = (i+1)/4;
    rec.analog[k] = (arc[i+20]*b1 + arc[i+21]*b2 + arc[i+22]*b3
        + arc[i+23])*cf;
}

/* Sub-satellite location */

rec.ssLoc[0][0] = (arc[88]*b1 + arc[89]*b2 + arc[90]*b3 +
                  arc[91]) * cf;
rec.ssLoc[1][0] = (arc[92]*b1 + arc[93]*b2 + arc[94]*b3 +
                  arc[95]) * cf;
rec.ssLoc[2][0] = (arc[96]*b1 + arc[97]*b2 + arc[98]*b3 +
                  arc[99]) * cf;
rec.ssLoc[3][0] = (arc[100]*b1 + arc[101]*b2 + arc[102]*b3 +
                  arc[103]) * cf;
rec.ssLoc[0][1] = (arc[104]*b1 + arc[105]*b2 + arc[106]*b3 +
                  arc[107]) * cf;
rec.ssLoc[1][1] = (arc[108]*b1 + arc[109]*b2 + arc[110]*b3 +
                  arc[111]) * cf;
rec.ssLoc[2][1] = (arc[112]*b1 + arc[113]*b2 + arc[114]*b3 +
                  arc[115]) * cf;
rec.ssLoc[3][1] = (arc[116]*b1 + arc[117]*b2 + arc[118]*b3 +
                  arc[119]) * cf;
rec.ssLoc[4][0] = (arc[120]*b1 + arc[121]*b2 + arc[122]*b3 +
                  arc[123]) * cf;
rec.ssLoc[5][0] = (arc[124]*b1 + arc[125]*b2 + arc[126]*b3 +
                  arc[127]) * cf;
rec.ssLoc[6][0] = (arc[128]*b1 + arc[129]*b2 + arc[130]*b3 +
                  arc[131]) * cf;
rec.ssLoc[7][0] = (arc[132]*b1 + arc[133]*b2 + arc[134]*b3 +
                  arc[135]) * cf;
rec.ssLoc[4][1] = (arc[136]*b1 + arc[137]*b2 + arc[138]*b3 +
                  arc[139]) * cf;
rec.ssLoc[5][1] = (arc[140]*b1 + arc[141]*b2 + arc[142]*b3 +
                  arc[143]) * cf;
rec.ssLoc[6][1] = (arc[144]*b1 + arc[145]*b2 + arc[146]*b3 +
                  arc[147]) * cf;
rec.ssLoc[7][1] = (arc[148]*b1 + arc[149]*b2 + arc[150]*b3 +
                  arc[151]) * cf;
rec.ssLoc[8][0] = (arc[152]*b1 + arc[153]*b2 + arc[154]*b3 +
                  arc[155]) * cf;
rec.ssLoc[9][0] = (arc[156]*b1 + arc[157]*b2 + arc[158]*b3 +
                  arc[159]) * cf;
rec.ssLoc[10][0] = (arc[160]*b1 + arc[161]*b2 + arc[162]*b3 +
```

```
                    arc[163]) * cf;
rec.ssLoc[11][0] = (arc[164]*b1 + arc[165]*b2 + arc[166]*b3 +
                    arc[167]) * cf;
rec.ssLoc[8][1] = (arc[168]*b1 + arc[169]*b2 + arc[170]*b3 +
                    arc[171]) * cf;
rec.ssLoc[9][1] = (arc[172]*b1 + arc[173]*b2 + arc[174]*b3 +
                    arc[175]) * cf;
rec.ssLoc[10][1] = (arc[176]*b1 + arc[177]*b2 + arc[178]*b3 +
                    arc[179]) * cf;
rec.ssLoc[11][1] = (arc[180]*b1 + arc[181]*b2 + arc[182]*b3 +
                    arc[183]) * cf;
rec.ssLoc[12][0] = (arc[184]*b1 + arc[185]*b2 + arc[186]*b3 +
                    arc[187]) * cf;
rec.ssLoc[13][0] = (arc[188]*b1 + arc[189]*b2 + arc[190]*b3 +
                    arc[191]) * cf;
rec.ssLoc[14][0] = (arc[192]*b1 + arc[193]*b2 + arc[194]*b3 +
                    arc[195]) * cf;
rec.ssLoc[15][0] = (arc[196]*b1 + arc[197]*b2 + arc[198]*b3 +
                    arc[199]) * cf;
rec.ssLoc[12][1] = (arc[200]*b1 + arc[201]*b2 + arc[202]*b3 +
                    arc[203]) * cf;
rec.ssLoc[13][1] = (arc[204]*b1 + arc[205]*b2 + arc[206]*b3 +
                    arc[207]) * cf;
rec.ssLoc[14][1] = (arc[208]*b1 + arc[209]*b2 + arc[210]*b3 +
                    arc[211]) * cf;
rec.ssLoc[15][1] = (arc[212]*b1 + arc[213]*b2 + arc[214]*b3 +
                    arc[215]) * cf;

/* Header information */

rec.head[0][1] = rec.ssLoc[0][0];
rec.head[1][1] = rec.ssLoc[4][0];
rec.head[2][1] = rec.ssLoc[8][0];
rec.head[3][1] = rec.ssLoc[12][0];
rec.head[0][2] = rec.ssLoc[0][1];
rec.head[1][2] = rec.ssLoc[4][1];
rec.head[2][2] = rec.ssLoc[8][1];
rec.head[3][2] = rec.ssLoc[12][1];

for(j=0; j<4; j++) {
    m = 2;
    i = 1640+j*96;
    for(k = 3;k<96; k+=4){
        m++;
        rec.head[j][m] = (arc[i+1+k-4]*b1 + arc[i+2+k-4]*b2
            + arc[i+3+k-4]*b3 + arc[i+4+k-4]) * cf;
    }
}

/*=====================================================*/
/* Each 32-second record contains four 8-second records */
/*=====================================================*/

j = 0;    /* Index for 8-sec records */
/* --- ihd info, 4-byte integers, ihd[j][0..4] --- */
for(i=0; i<20; i+=4) {
    k = (i+1)/4;
```

```
    rec.ihd[j][k] = arc[i+216]*b1 + arc[i+217]*b2
            + arc[i+218]*b3 + arc[i+219];
}
/* --- Satellite orbital inclination, head[j][0] --- */
rec.head[j][0] = (arc[236]*b1 + arc[237]*b2 + arc[238]*b3 +
    arc[239])*cf/10.;
/* --- Orbit number, ihd[j][5] --- */
rec.ihd[j][5] = arc[240]*b1 + arc[241]*b2 + arc[242]*b3 +
    arc[243];
/* --- Quality flag, 2-byte integer, qual[0..3] --- */
qf = 0;
for(i=0; i<7; i+=2){
    rec.qual[qf] = arc[i+244]*b3 + arc[i+245];
    qf++;
}
/* --- Minor frame numbers, 2-byte integer, minor[0..3] --- */
mf = 0;
for(i=0; i<7; i+=2){
    rec.minor[mf] = arc[i+252]*b3 + arc[i+253];
    mf++;
}
/* --- Missing data flags, 1-byte integer, mdf[0..3][0..39] --- */
for(i=0; i<4; i++) {
    for(n=0; n<39; n++) {
        rec.mdf[i][n] = arc[n+40*i+260];
    }
}

j = 1;    /* Index for 8-sec records */
/* --- ihd info, 4-byte integers, ihd[j][0..4] --- */
for(i=0; i<20; i+=4) {
    k = (i+1)/4;
    rec.ihd[j][k] = arc[i+420]*b1 + arc[i+421]*b2
            + arc[i+422]*b3 + arc[i+423];
}
/* --- Satellite orbital inclination, head[j][0] --- */
rec.head[j][0] = (arc[440]*b1 + arc[441]*b2 + arc[442]*b3 +
    arc[443])*cf/10.;
/* --- Orbit number, ihd[j][5] --- */
rec.ihd[j][5] = arc[444]*b1 + arc[445]*b2 + arc[446]*b3 +
    arc[447];
/* --- Quality flag, 2-byte integer, qual[4..7] --- */
for(i=0; i<7; i+=2){
    rec.qual[qf] = arc[i+448]*b3 + arc[i+449];
    qf++;
}
/* --- Minor frame numbers, 2-byte integer, minor[4..7] --- */
for(i=0; i<7; i+=2){
    rec.minor[mf] = arc[i+456]*b3 + arc[i+457];
    mf++;
}
/* --- Missing data flags, 1-byte integer, mdf[4..7][0..39] --- */
for(i=4; i<8; i++) {
    for(n=0; n<39; n++) {
        rec.mdf[i][n] = arc[n+40*(i-4)+464];
    }
}
```

```
j = 2;    /* Index for 8-sec records */
/* --- ihd info, 4-byte integers, ihd[j][0..4] --- */
for(i=0; i<20; i+=4) {
    k = (i+1)/4;
    rec.ihd[j][k] = arc[i+624]*b1 + arc[i+625]*b2
            + arc[i+626]*b3 + arc[i+627];
}
/* --- Satellite orbital inclination, head[j][0] --- */
rec.head[j][0] = (arc[644]*b1 + arc[645]*b2 + arc[646]*b3 +
    arc[647])*cf/10.;
/* --- Orbit number, ihd[j][5] --- */
rec.ihd[j][5] = arc[648]*b1 + arc[649]*b2 + arc[650]*b3 +
    arc[651];
/* --- Quality flag, 2-byte integer, qual[8..11] --- */
for(i=0; i<7; i+=2){
    rec.qual[qf] = arc[i+652]*b3 + arc[i+652];
    qf++;
}
/* --- Minor frame numbers, 2-byte integer, minor[8..11] --- */
for(i=0; i<7; i+=2){
    rec.minor[mf] = arc[i+660]*b3 + arc[i+661];
    mf++;
}
/* --- Missing data flags, 1-byte integer, mdf[8..11][0..39] --- */
for(i=8; i<12; i++) {
    for(n=0; n<39; n++) {
        rec.mdf[i][n] = arc[n+40*(i-8)+668];
    }
}

j = 3;    /* Index for 8-sec records */
/* --- ihd info, 4-byte integers, ihd[j][0..4] --- */
for(i=0; i<20; i+=4) {
    k = (i+1)/4;
    rec.ihd[j][k] = arc[i+828]*b1 + arc[i+829]*b2
            + arc[i+830]*b3 + arc[i+831];
}
/* --- Satellite orbital inclination, head[j][0] --- */
rec.head[j][0] = (arc[848]*b1 + arc[849]*b2 + arc[850]*b3 +
    arc[851])*cf/10.;
/* --- Orbit number, ihd[j][5] --- */
rec.ihd[j][5] = arc[852]*b1 + arc[853]*b2 + arc[854]*b3 +
    arc[855];
/* --- Quality flag, 2-byte integer, qual[12..15] --- */
for(i=0; i<7; i+=2){
    rec.qual[qf] = arc[i+856]*b3 + arc[i+857];
    qf++;
}
/* --- Minor frame numbers, 2-byte integer, minor[12..15] --- */
for(i=0; i<7; i+=2){
    rec.minor[mf] = arc[i+864]*b3 + arc[i+865];
    mf++;
}
/* --- Missing data flags, 1-byte integer, mdf[12..15][0..39] --- */
for(i=12; i<16; i++) {
    for(n=0; n<39; n++) {
```

```
        rec.mdf[i][n] = arc[n+40*(i-12)+872];
    }
}

/*-----------------------------------------------------*/
/*                 MEPED Sensor Data                   */
/*-----------------------------------------------------*/

j = mp0 = mp90 = mo = 0;

/* MEPED 0-deg telescope mep0[0..3][0..8]  */
for(i=0; i<4; i++) {
    jj = j*4+i;
    for(k=0; k<9; k++) {
        if(rec.mdf[jj][k+1] == 1) {
            rec.mep0[jj][k] = -999.;
        } else {
            rec.mep0[jj][k] = cnvrt[arc[mp0+1032]];
        }
        mp0++;
    }
}
/* MEPED 90-deg telescope mep0[0..3][0..8]  */
for(i=0; i<4; i++) {
    jj = j*4+i;
    for(k=0; k<9; k++) {
        if(rec.mdf[jj][k+10] == 1) {
            rec.mep90[jj][k] = -999.;
        } else {
            rec.mep90[jj][k] = cnvrt[arc[mp90+1068]];
        }
        mp90++;
    }
}
/* MEPED Omnidirectional [0..3][0..3]  */
for(i=0; i<4; i++) {
    jj = j*4+i;
    for(k=0; k<4; k++) {
        rec.mepOmni[jj][k] = cnvrt[arc[mo+1104]];
        mo++;
    }
    if(rec.mdf[jj][19] == 1) rec.mepOmni[jj][0] = -999.;
    if(rec.mdf[jj][20] == 1) rec.mepOmni[jj][1] = -999.;
    if((jj % 2 == 1) && (rec.mdf[jj][21] == 1))
        rec.mepOmni[jj][3] = -999.;
    if((jj % 2 == 0) && (rec.mdf[jj][21] == 1))
        rec.mepOmni[jj][2] = -999.;
}

j = 1;
mp0 = mp90 = mo = 0;
/* MEPED 0-deg telescope mep0[4..7][0..8]  */
for(i=0; i<4; i++) {
    jj = j*4+i;
    for(k=0; k<9; k++) {
        if(rec.mdf[jj][k+1] == 1) {
            rec.mep0[jj][k] = -999.;
```

```
        } else {
            rec.mep0[jj][k] = cnvrt[arc[mp0+1184]];
        }
        mp0++;
    }
}
/* MEPED 90-deg telescope mep0[4..7][0..8]  */
for(i=0; i<4; i++) {
    jj = j*4+i;
    for(k=0; k<9; k++) {
        if(rec.mdf[jj][k+10] == 1) {
            rec.mep90[jj][k] = -999.;
        } else {
            rec.mep90[jj][k] = cnvrt[arc[mp90+1220]];
        }
        mp90++;
    }
}
/* MEPED Omnidirectional [4..7][0..3]  */
for(i=0; i<4; i++) {
    jj = j*4+i;
    for(k=0; k<4; k++) {
        rec.mepOmni[jj][k] = cnvrt[arc[mo+1256]];
        mo++;
    }
    if(rec.mdf[jj][19] == 1) rec.mepOmni[jj][0] = -999.;
    if(rec.mdf[jj][20] == 1) rec.mepOmni[jj][1] = -999.;
    if((jj % 2 == 1) && (rec.mdf[jj][21] == 1))
        rec.mepOmni[jj][3] = -999.;
    if((jj % 2 == 0) && (rec.mdf[jj][21] == 1))
        rec.mepOmni[jj][2] = -999.;
}

j = 2;
mp0 = mp90 = mo = 0;
/* MEPED 0-deg telescope mep0[8..11][0..8]  */
for(i=0; i<4; i++) {
    jj = j*4+i;
    for(k=0; k<9; k++) {
        if(rec.mdf[jj][k+1] == 1) {
            rec.mep0[jj][k] = -999.;
        } else {
            rec.mep0[jj][k] = cnvrt[arc[mp0+1336]];
        }
        mp0++;
    }
}
/* MEPED 90-deg telescope mep0[8..11][0..8]  */
for(i=0; i<4; i++) {
    jj = j*4+i;
    for(k=0; k<9; k++) {
        if(rec.mdf[jj][k+10] == 1) {
            rec.mep90[jj][k] = -999.;
        } else {
            rec.mep90[jj][k] = cnvrt[arc[mp90+1372]];
        }
        mp90++;
```

```
    }
}
/* MEPED Omnidirectional [8..11][0..3]  */
for(i=0; i<4; i++) {
    jj = j*4+i;
    for(k=0; k<4; k++) {
        rec.mepOmni[jj][k] = cnvrt[arc[mo+1408]];
        mo++;
    }
    if(rec.mdf[jj][19] == 1) rec.mepOmni[jj][0] = -999.;
    if(rec.mdf[jj][20] == 1) rec.mepOmni[jj][1] = -999.;
    if((jj % 2 == 1) && (rec.mdf[jj][21] == 1))
        rec.mepOmni[jj][3] = -999.;
    if((jj % 2 == 0) && (rec.mdf[jj][21] == 1))
        rec.mepOmni[jj][2] = -999.;
}

j = 3;
mp0 = mp90 = mo = 0;
/* MEPED 0-deg telescope mep0[12..15][0..8]  */
for(i=0; i<4; i++) {
    jj = j*4+i;
    for(k=0; k<9; k++) {
        if(rec.mdf[jj][k+1] == 1) {
            rec.mep0[jj][k] = -999.;
        } else {
            rec.mep0[jj][k] = cnvrt[arc[mp0+1488]];
        }
        mp0++;
    }
}
/* MEPED 90-deg telescope mep0[12..15][0..8]  */
for(i=0; i<4; i++) {
    jj = j*4+i;
    for(k=0; k<9; k++) {
        if(rec.mdf[jj][k+10] == 1) {
            rec.mep90[jj][k] = -999.;
        } else {
            rec.mep90[jj][k] = cnvrt[arc[mp90+1524]];
        }
        mp90++;
    }
}
/* MEPED Omnidirectional [12..15][0..3]  */
for(i=0; i<4; i++) {
    jj = j*4+i;
    for(k=0; k<4; k++) {
        rec.mepOmni[jj][k] = cnvrt[arc[mo+1560]];
        mo++;
    }
    if(rec.mdf[jj][19] == 1) rec.mepOmni[jj][0] = -999.;
    if(rec.mdf[jj][20] == 1) rec.mepOmni[jj][1] = -999.;
    if((jj % 2 == 1) && (rec.mdf[jj][21] == 1))
        rec.mepOmni[jj][3] = -999.;
    if((jj % 2 == 0) && (rec.mdf[jj][21] == 1))
        rec.mepOmni[jj][2] = -999.;
}
```

```
/*-----------------------------------------------------*/
/*     TED uncalibrated energy flux, max channel       */
/*          response, and max channel                  */
/*-----------------------------------------------------*/

/* TED 0-deg and 30-deg uncalibrated energy flux */
for(k=0; k<4; k++) {
    for(i=0; i<4; i++) {
        j = k*4 + i;    /* 0..15 */
        rec.ted0[j][0] = cnvrt[arc[1120+i*4+k*152]];
        if(rec.mdf[j][26] == 1) rec.ted0[j][0] = -999.;
        rec.ted0[j][1] = cnvrt[arc[1121+i*4+k*152]];
        if(rec.mdf[j][28] == 1) rec.ted0[j][1] = -999.;
        rec.ted0[j][2] = cnvrt[arc[1122+i*4+k*152]];
        if(rec.mdf[j][30] == 1) rec.ted0[j][2] = -999.;
        rec.ted0[j][3] = cnvrt[arc[1123+i*4+k*152]];
        if(rec.mdf[j][32] == 1) rec.ted0[j][3] = -999.;
        rec.ted0[j][4] = arc[1136+i*4+k*152];
        rec.ted0[j][5] = arc[1137+i*4+k*152];
        if(rec.mdf[j][34] == 1) {
            rec.ted0[j][4] = -999.;
            rec.ted0[j][5] = -999.;
        }
        rec.ted0[j][6] = cnvrt[arc[1138+i*4+k*152]];
        if(rec.mdf[j][35] == 1) rec.ted0[j][6] = -999.;
        rec.ted0[j][7] = cnvrt[arc[1139+i*4+k*152]];
        if(rec.mdf[j][36] == 1) rec.ted0[j][7] = -999.;

        rec.ted30[j][0] = cnvrt[arc[1152+i*4+k*152]];
        if(rec.mdf[j][27] == 1) rec.ted30[j][0] = -999.;
        rec.ted30[j][1] = cnvrt[arc[1153+i*4+k*152]];
        if(rec.mdf[j][29] == 1) rec.ted30[j][1] = -999.;
        rec.ted30[j][2] = cnvrt[arc[1154+i*4+k*152]];
        if(rec.mdf[j][31] == 1) rec.ted30[j][2] = -999.;
        rec.ted30[j][3] = cnvrt[arc[1155+i*4+k*152]];
        if(rec.mdf[j][33] == 1) rec.ted30[j][3] = -999.;
        rec.ted30[j][4] = arc[1168+i*4+k*152];
        rec.ted30[j][5] = arc[1169+i*4+k*152];
        if(rec.mdf[j][34] == 1) {
            rec.ted30[j][4] = -999.;
            rec.ted30[j][5] = -999.;
        }
        rec.ted30[j][6] = cnvrt[arc[1170+i*4+k*152]];
        if(rec.mdf[j][35] == 1) rec.ted30[j][6] = -999.;
        rec.ted30[j][7] = cnvrt[arc[1171+i*4+k*152]];
        if(rec.mdf[j][36] == 1) rec.ted30[j][7] = -999.;
    }
}

/*-----------------------------------------------------*/
/*          TED spectra and background                 */
/*-----------------------------------------------------*/

for(j=0; j<4; j++) {
    for(i=0; i<8; i++) {
        rec.ted0s[j][i] = cnvrt[arc[i+j*8+2024]];
```

```
            rec.ted30s[j][i] = cnvrt[arc[i+j*8+2060]];
        }
        rec.tedback[j][0] = cnvrt[arc[j+2056]];
        rec.tedback[j][1] = cnvrt[arc[j+2092]];
    }
    for(j=0; j<16; j++) {
        if(j==0 || j==4 || j==8 || j==12) {
            for(i=0; i<4; i++) {
                if(rec.mdf[j][i+22] == 1) {
                    for(k=0; k<4; k++) {
                        rec.ted0s[k][i] = -999.;
                    }
                }
            }
        }
        if(j==2 || j==6 || j==10) {
            for(i=4; i<8; i++) {
                if(rec.mdf[j][i+18] == 1) {
                    for(k=0; k<4; k++) {
                        rec.ted0s[k][i] = -999.;
                    }
                }
            }
        }
        if(j==1 || j==5 || j==9 || j==13) {
            for(i=0; i<4; i++) {
                if(rec.mdf[j][i+22] == 1) {
                    for(k=0; k<4; k++) {
                        rec.ted30s[k][i] = -999.;
                    }
                }
            }
        }
        if(j==3 || j==7 || j==11) {
            for(i=4; i<8; i++) {
                if(rec.mdf[j][i+18] == 1) {
                    for(k=0; k<4; k++) {
                        rec.ted30s[k][i] = -999.;
                    }
                }
            }
        }
        if(j == 14) {
            if(rec.mdf[j][0] == 1) rec.tedback[0][0] = -999.;
            if(rec.mdf[j][22] == 1) rec.tedback[1][0] = -999.;
            if(rec.mdf[j][25] == 1) rec.tedback[2][0] = -999.;
            if(rec.mdf[j][24] == 1) rec.tedback[3][0] = -999.;
            if(rec.mdf[j][23] == 1) rec.tedback[1][1] = -999.;
        }
        if(j == 15) {
            if(rec.mdf[j][0] == 1) rec.tedback[0][1] = -999.;
            if(rec.mdf[j][25] == 1) rec.tedback[2][1] = -999.;
            if(rec.mdf[j][23] == 1) rec.tedback[3][1] = -999.;
        }
    }

    /*-------------------------------------------------*/
```

```
    /*                   TED flux                          */
    /*-----------------------------------------------------*/
    tf = 0;
    for(i=0; i<4; i++) {
        for(n=0; n<7; n++) {
            for(j=0; j<4; j++) {
                k = i*4+j;
                rec.tedfx[k][n] = (arc[tf+2096]*b1 +
                    arc[tf+2097]*b2 + arc[tf+2098]*b3 +
                    arc[tf+2099]) * cf;
                tf+=4;
            }
        }
    }

} /* End of unpackSem2 */

void open_archive(char *filename) {
    extern FILE *fp;
    short i;
    size_t len;
    len = strlen(filename);

     /* printf("In open: archive file = *%s*\n",filename); */
    for (i=0; i<=len; i++) {
        if (filename[i] == ' ') {
            filename[i] = '\0';
            break;
        }
    }

    if((fp = fopen(filename, "rb")) == NULL) {
        printf("Cannot open packed archive file %s.\n",filename);
        fclose(fp);
        exit(1);
    } else {
        printf("Opened packed archive file %s.\n",filename);
    }

}

void close_archive(void) {
    extern FILE *fp;
    fclose(fp);
}
```