```
/* $Header$ */
/*************************************************************
*******
** FILE:  BIN2TXT.C
**
** Copyright(c) 1997-present.  All rights reserved.
** Biocontrol Technology Incorporated, Indiana, PA USA
** This file constitutes intellectual property of the company and
 may
** not be used for other than company purposes.
**
** DESCRIPTION:
**    Utility program to convert binary data to text format.
**
** LIMITATIONS:
**   None.
**
** SOFTWARE HISTORY:
**   29SEP99  5147  STR#0000 K.Pasumarthy
**   Initial creation.
**
**   11Jan00  5147  STR#0000 K.Pasumarthy
**   Added options for matlab (-m), excel (-e), and final test gr
oup (-t)
**
**   08MAR00  5147  STR#0000 K.Pasumarthy
**   Modified the way readings are written to the output file und
er -m option.
**
**   01JUNE00  5147  STR#0000 K.Pasumarthy
**   Added demuxing capability for alert messages.
*************************************************************
******/

/***
** System include files
****/
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <conio.h>
#include <malloc.h>

/***
** Local include files
****/
#include "d1ktype.h"
```

```
/***
** Global variables
****/

DT_SPECTRUM packet[1];
DT_MDM_READING mdm_reading[1];
DT_ALERT alert[1];
char optChar = 0;  /* the option choosed. */
int total_packets = 0;
/*
*****************************************************************
************
* Function: write_data
*****************************************************************
************
*/
void write_data (char *in_file_name, char *out_file_name)
{

    FILE *input_file;      /* pointer to file containing binary re
adings */
    FILE *output_file;     /* resulting text file */

    char replace = 0;      /* Check for key entry */
    int ii = 0;            /* Used to print sensor & spectra infor
mation */
    int count = 0;             /* Used as a counter for demuxing s
pectra */
    int struct_size = 0;   /* holds the size of the data to read f
rom the file */
    int done = 0;
    /* Open the input file to read data */
    if ((input_file = fopen(in_file_name, "rb")) == NULL)
    {
        printf("\a"); /* Beep */
        printf ("\nCannot open input file\n");
        exit(1);
    }
    /* Open the output file to write */
    if( (output_file = fopen( out_file_name, "rb" ) ) != NULL )
    {
        printf( "\n\aWarning -  Output file exists " );
        fclose( output_file );
        printf( "\nWant to overwrite the existing file? ( Y, N ):
" );
        replace = tolower( getche() );
        if( replace == 'n' ) exit(1);
    }
```

```
    output_file = fopen( out_file_name, "w" );

    while(!done)
    {
       packet[0].objHdr.id = getw( input_file );

       if(packet[0].objHdr.id == EOD_DATE)
       {
           struct_size = (sizeof(DT_EOD_DATE) - sizeof(packet[0].
objHdr.id));
           fread(&packet[0].objHdr.size, struct_size, 1, input_fi
le);
       }
       else if(packet[0].objHdr.id == SPECTRUM)
       {
           if(optChar != 't')
           {
               fprintf( output_file, "%u ", packet[0].objHdr.id);
           }
           struct_size = (sizeof(DT_SPECTRUM) - sizeof(packet[0].
objHdr.id));
           fread(&packet[0].objHdr.size, struct_size, 1, input_fi
le);
           if(optChar !='t')
           {
               fprintf( output_file, "%u ", packet[0].objHdr.size
 );
           }
           fprintf( output_file, "%u ", packet[0].spectrum_type )
;
           if(optChar !='t')
           {
               fprintf( output_file, "%u ", packet[0].mode );
               fprintf( output_file, "%u ", packet[0].sit_count )
;
               fprintf( output_file, "%u ", packet[0].ses_count )
;
               fprintf( output_file, "%u ", packet[0].sub_count )
;
           }
           fprintf( output_file, "%lu ", packet[0].time_stamp );
           if(optChar !='t')
           {
               fprintf( output_file, "%u ", packet[0].year );
               fprintf( output_file, "%lu ", packet[0].bico_user_
id );
               if(optChar == 'm')
```

```
                {
                    strcpy(packet[0].serial_num, "00000000000");
                }
                fprintf( output_file, "%s ", &packet[0].serial_num
  );
                fprintf( output_file, "%u ", packet[0].bin_code );
                fprintf( output_file, "%u ", packet[0].invasive );
            }
            for(ii = 0; ii < DT_MAX_SPECTRA_CHANNELS; ii++)

            {
                fprintf( output_file, "% .15f ", packet[0].spectru
m[ii] );
            }
            if(optChar !='t')
            {
                for(ii = 0; ii < (DT_MAX_SENSOR_CHANNELS); ii++)


                {
                    fprintf( output_file, "% .3f ", packet[0].sens
ors[ii] );
                }
            }
            else
            {
                for(ii = 0; ii < (DT_MAX_SENSOR_CHANNELS - 1); ii+
+)

                {
                    fprintf( output_file, "% .3f ", packet[0].sens
ors[ii] );
                }
            }
            fprintf( output_file, "\n" );
        }
        else if(packet[0].objHdr.id == SPECTRUM_STATUS)
        {
            fprintf( output_file, "%u ", packet[0].objHdr.id);
            struct_size = (sizeof(DT_SPECTRUM_STATUS) -
sizeof(packet[0].objHdr.id));
            fread(&packet[0].objHdr.size, struct_size, 1, input_fi
le);
            /***
            * The first 40 bytes of the spectrum and spectrum stat
us are identical
            * except invasive reading is defined as error_codes in
 spectrum_status
```

```
            * structure.
            ***/
            fprintf( output_file, "%u ", packet[0].objHdr.size );
            fprintf( output_file, "%u ", packet[0].spectrum_type )
;
            fprintf( output_file, "%u ", packet[0].mode );
            fprintf( output_file, "%u ", packet[0].sit_count );
            fprintf( output_file, "%u ", packet[0].ses_count );
            fprintf( output_file, "%u ", packet[0].sub_count );
            fprintf( output_file, "%lu ", packet[0].time_stamp );
            fprintf( output_file, "%u ", packet[0].year );
            fprintf( output_file, "%lu ", packet[0].bico_user_id )
;
            if(optChar == 'm')
            {
                strcpy(packet[0].serial_num, "00000000000");
            }
            fprintf( output_file, "%s ", &packet[0].serial_num );
            fprintf( output_file, "%u ", packet[0].bin_code );
            fprintf( output_file, "%u ", packet[0].invasive ); /*
Error codes */
            if(optChar == 'm')
            {
                for(ii = 0; ii < DT_MAX_SPECTRA_CHANNELS; ii++)

                {
                    fprintf( output_file, "0.0 ");
                }
                for(ii = 0; ii < DT_MAX_SENSOR_CHANNELS; ii++)

                {
                    fprintf( output_file, "0.0 ");
                }
            }
            fprintf( output_file, "\n");
        }
        else if(packet[0].objHdr.id == READING)
        {
            fprintf( output_file, "%u ", packet[0].objHdr.id);
            struct_size = (sizeof(DT_MDM_READING) - sizeof(packet[
0].objHdr.id));
            fread(&mdm_reading[0].objHdr.size, struct_size, 1, inp
ut_file);
            fprintf( output_file, "%u ", mdm_reading[0].objHdr.siz
e );
            if(optChar == 'm')
```

```
            {
                strcpy(mdm_reading[0].readingID.serial_num, "00000
000000");
            }
            fprintf( output_file, "%s ", &mdm_reading[0].readingID
.serial_num );
            fprintf( output_file, "%lu ", mdm_reading[0].readingID
.bico_user_id);
            fprintf( output_file, "%u ", mdm_reading[0].readingID.
mode );
            fprintf( output_file, "%u ", mdm_reading[0].reading.em
pty);
            if(optChar == 'm')
            {
                fprintf( output_file, "0");
            }
            else
            {
                fprintf( output_file, "%d/", mdm_reading[0].readin
g.time.month);
                fprintf( output_file, "%d/",
mdm_reading[0].reading.time.day_of_month);
                fprintf( output_file, "%d ", mdm_reading[0].readin
g.time.year);
                fprintf( output_file, "%d:", mdm_reading[0].readin
g.time.hours);
                fprintf( output_file, "%d:", mdm_reading[0].readin
g.time.minutes);
                fprintf( output_file, "%d ", mdm_reading[0].readin
g.time.seconds);
            }
            fprintf( output_file, "%d ", mdm_reading[0].reading.gl
ucose);
            fprintf( output_file, "%d ", mdm_reading[0].reading.co
ntrol);
            fprintf( output_file, "%d ", mdm_reading[0].reading.in
vasive);
            fprintf( output_file, "%d ", mdm_reading[0].reading.bi
n);
            fprintf( output_file, "%d ", mdm_reading[0].reading.qm
_flag);
            fprintf( output_file, "%u ", mdm_reading[0].reading.st
atus);
            fprintf( output_file, "%d ", mdm_reading[0].reading.co
rrection);
            if(optChar == 'm')
            {
                for(ii = 0; ii < DT_MAX_SPECTRA_CHANNELS; ii++)
```

```
                {
                    fprintf( output_file, "0.0 ");
                }
                for(ii = 0; ii < DT_MAX_SENSOR_CHANNELS; ii++)

                {
                    fprintf( output_file, "0 ");
                }
            }
            fprintf( output_file, "\n");
        }
        else if(packet[0].objHdr.id == ALERT)
        {
            fprintf( output_file, "%u ", packet[0].objHdr.id);
            struct_size = (sizeof(DT_ALERT) - sizeof(packet[0].obj
Hdr.id));
            fread(&alert[0].objHdr.size, struct_size, 1, input_fil
e);

            fprintf( output_file, "%u ", alert[0].objHdr.size );
            fprintf( output_file, "%lu ", alert[0].bico_user_id );
            if(optChar == 'm')
            {
                strcpy(alert[0].serial_num, "00000000000");
            }
            fprintf( output_file, "%s ", &alert[0].serial_num );
            if(optChar == 'm')
            {
                fprintf( output_file, "0 ");
            }
            else
            {
                fprintf( output_file, "%d/", alert[0].date.month);
                fprintf( output_file, "%d/", alert[0].date.day_of_
month);
                fprintf( output_file, "%d ", alert[0].date.year);
                fprintf( output_file, "%d:", alert[0].date.hours);
                fprintf( output_file, "%d:", alert[0].date.minutes
);
                fprintf( output_file, "%d ", alert[0].date.seconds
);
                fprintf( output_file, "%u ", alert[0].mode );
                fprintf( output_file, "%u ", alert[0].stat_code);
            }
            if(optChar == 'm')
```

```
            {
                for(ii = 0; ii < DT_MAX_SPECTRA_CHANNELS; ii++)

                {
                    fprintf( output_file, "0.0 ");
                }
                for(ii = 0; ii < DT_MAX_SENSOR_CHANNELS + 8; ii++)

                {
                    fprintf( output_file, "0 ");
                }
            }
            fprintf( output_file, "\n");
        }
        else if(packet[0].objHdr.id == 7)
        {
            printf("\aConversion done");
            printf("\nTotal packets converted %u", count);
            break;
        }
        else
        {
        }
        count ++;
        if(count == total_packets) break;
        if( ( (feof( input_file ) || ferror( input_file ))) )   /*
 End of file */
                                            /* else EOF character in fil
e */
        {
            printf("\aConversion done");
            printf("\nTotal packets converted %u", count - 1);
            break;
        }
    }  /* end of while loop */

    fclose( input_file );
    fclose( output_file );

}
/*
*****************************************************************
************
* Function: main
*****************************************************************
************
```

```
*/
int main(int argc, char *argv[])
{
    if (argc < 3)
    {
        fprintf(stderr, "Usage: bin2txt infile outfile total_spec
tra \n");
        fprintf(stderr, " infile = input file to read data from \
n");
        fprintf(stderr, " outfile = output file to write data to
\n");
        fprintf(stderr, " [optional] -m for matlab \n");
        fprintf(stderr, " [optional] -t for final test group \n")
;
        fprintf(stderr, " [optional] -e for excel, default is -e\
n");
        fprintf(stderr, " [optional] number of spectra to convert
, default converts
all\n");
        exit(3);
    }
    if(argv[3])
    {
        optChar = tolower(argv[3][1]);
    }
    if(argv[4])
    {
        total_packets = atoi(argv[4]);
    }
    write_data(argv[1], argv[2]);
    return(0);
}

/***
* End of File
***/
```