# Converting Data Formats Between BMP and COFF Files for TMS320 DSP Simulation

*APPLICATION REPORT: SPRA435*

*Vivian Shao*
*Semiconductor Sales & Marketing*
*Texas Instruments Taiwan Limited Technical Staff*

*Digital Signal Processing Solutions*
*April 1998*

## IMPORTANT NOTICE

Texas Instruments (TI) reserves the right to make changes to its products or to discontinue any semiconductor product or service without notice, and advises its customers to obtain the latest version of relevant information to verify, before placing orders, that the information being relied on is current.

TI warrants performance of its semiconductor products and related software to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are utilized to the extent TI deems necessary to support this warranty. Specific testing of all parameters of each device is not necessarily performed, except those mandated by government requirements.

Certain application using semiconductor products may involve potential risks of death, personal injury, or severe property or environmental damage ("Critical Applications").

TI SEMICONDUCTOR PRODUCTS ARE NOT DESIGNED, INTENDED, AUTHORIZED, OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT APPLICATIONS, DEVICES OR SYSTEMS OR OTHER CRITICAL APPLICATIONS.

Inclusion of TI products in such applications is understood to be fully at the risk of the customer. Use of TI products in such applications requires the written approval of an appropriate TI officer. Questions concerning potential risk applications should be directed to TI through a local SC sales office.

In order to minimize risks associated with the customer's applications, adequate design and operating safeguards should be provided by the customer to minimize inherent or procedural hazards.

TI assumes no liability for applications assistance, customer product design, software performance, or infringement of patents or services described herein. Nor does TI warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such semiconductor products or services might be or are used.

## TRADEMARKS

TI is a trademark of Texas Instruments Incorporated.

Other brands and names are the property of their respective owners.

**CONTACT INFORMATION**

| | |
|---|---|
| US TMS320 HOTLINE | (281) 274-2320 |
| US TMS320 FAX | (281) 274-2324 |
| US TMS320 BBS | (281) 274-2323 |
| US TMS320 email | dsph@ti.com |

# Contents

# Converting Data Formats Between BMP and COFF Files for TMS320 DSP Simulation

## Abstract

This application report is aimed at digital signal processor (DSP) software development engineers working with image processing algorithms. The report focuses on the conversion process using data format converting utilities supporting the Texas Instruments (TI™) TMS320 (C1x/C2x/C2xx/C5x/C54x) fixed-point digital signal processor (DSP) in COFF format.

Basically, two main software tools are required to verify image processing algorithms: a DSP simulator and Windows application software. The TI TMS320 DSP simulator uses the COFF (Common Object File Format) data format. The most popular data format supported by Windows application software is BMP (Bit Map). (Displaying the image using Windows application software is the most effective way to verify image processing algorithms.)

The utilities used to *convert the data format between BMP and COFF files* make the algorithm verification procedure easier – converting BMP to COFF, processing the data using the DSP simulator, then converting the resulting COFF to BMP and finally displaying the processed image on screen.


.

# Product Support

## Related Documentation

The following list specifies product names, part numbers, and literature numbers of corresponding TI documentation.

- *TMS320C5x C Source Debugger User's Guide*, February 1994, Literature number SPRU055B

## World Wide Web

TI's World Wide Web site at www.ti.com contains the most up-to-date product information, revisions, and additions. New users must register with TI&ME before they can access the data sheet archive. TI&ME allows users to build custom information pages and receive new product updates automatically via email.

# Introduction

The most effective way to verify image-processing algorithms is to use Windows application software to display the image. BMP is one of the image file formats used by Windows 3.X. However, it cannot be used directly as the input to the TMS320 simulator.As a result, the BMP file must be converted to either a TMS320 assembly file, which includes the image data as initialized data section and can be linked to main program, or a TMS320 I/O data file, which is in heximal and connected to the input/output pin defined by siminit.cmd.

This application note includes two main topics. The first introduces the file formats for the BMP file, TMS320 assembly file, and TMS320 I/O data file. The second topic explains how to use the conversion utilities. Appendixes A, B, and C contain the C source codes.

Because BMP cannot be used directly as input to a TMS320 simulator, the BMP file must be converted to either of the following files:

- TMS320 Assembly File

  Includes the image data as an initialized data section and can be linked to the main program

- TMS320 I/O Data File

  In heximal and connected to an input/output pin defined by siminit.cmd

# File Formats Description

## Bit Map File (.BMP)

BMP is an image file format used by Windows 3.X. Almost all Windows application software supports BMP format.

The BMP data structure includes three main parts:

- BITMAPFILEHEADER

  BMP file information

- BITMAPINFOHEADER

  BMP image information

- PIXEL DATA

  BMP image data information

### BMP File Information

The BITMAPFILEHEADER data structure defined in WINDOWS.H is:

```
typedef struct tagBITMAPFILEHEADER
        {
                WORD        bfType;
                DWORD       bfSize;
                WORD        bfReserved1;
                WORD        bfReserved2;
                DWORD       bfOffBits;
        }       BITMAPFILEHEADER;
typedef BITMAPFILEHEADER FAR *LPBITMAPFILEHEADER;
typedef BITMAPFILEHEADER      *PBITMAPFILEHEADER;
```

where

| | |
|---|---|
| **bfType** | file type, which is "BM" |
| **bfSize** | file size, including file header |
| **bfReserved1** | not used, set to 0 |
| **bfReserved2** | not used, set to 0 |
| **bfOffbits** | total bytes from the beginning of the file to the beginning of the image data, i.e., the length of the header |

## BMP Image Information

The BITMAPINFOHEADER data structure defined in WINDOWS.H is:

```
typedef struct tagBITMAPINFOHEADER
        {
                DWORD       biSize;
                DWORD       biWidth;
                DWORD       biHeight;
                WORD        biPlanes;
                WORD        biBitCount;
                DWORD       biCompression;
                DWORD       biSizeImage;
                DWORD       biXPelsPerMeter;
                DWORD       biYPelsPerMeter;
                DWORD       biClrUsed;
                DWORD       biClrImportant;
        }       BITMAPINFOHEADER;
typedef BITMAPINFOHEADER FAR *LPBITMAPINFOHEADER;
typedef BITMAPINFOHEADER      *PBITMAPINFOHEADER;
```

where

**bfSize** 40 for Windows BMP and 12 for OS/2 BMP

**biWidth** width of image in pixels

**biHight** hight of image in pixels

**biPlanes** number of color planes; should be 1 in BMP

**biBitCount** bits needed per pixel

| biBitCount | Colors |
|---|---|
| 1 | 2 |
| 4 | 16 |
| 8 | 256 |
| 24 | full color |

**biCompression** 0 as no compression

**biSizeImage** size of image in bytes

**biXPelsPerMeter** pixels/meter in horizontal

**biYPelsPerMeter** pixels/meter in vertical

**biClrUsed** number of colors that image uses;
0 means number of colors = biBitCount

**biClrImportant** number of important colors;
0 means all the colors are important

## BMP Image Data Information

Image colored data is follows the file headers "BITMAPFILEHEADER" and "BITMAPINFOHEADER." The sequence is BGRBGRBGR (where B represents blue, G represents green, and R represents red).

### *Values of Header Parameters and Image data*

The following example defines the BMP header of a full colored 160 pixels (width) x 120 pixels (length) image:

```
char    bmpheader[54]=          /* BMP FILE HEADER 54 BYTES */
        {       0x42, 0x4d,             /* bfType = 'BM'        */
                0x36, 0xe1, 0x00, 0x00,/* bfSize = 57654       */
                0x00, 0x00,             /* bfReserved1 = 0      */
                0x00, 0x00,             /* bfReserved2 = 0      */
                0x36, 0x00, 0x00, 0x00,/* bf0ffBits = 54       */
                0x28, 0x00, 0x00, 0x00,/* biSize = 40          */
                0xa0, 0x00, 0x00, 0x00,/* biWidth = 160        */
                0x78, 0x00, 0x00, 0x00,/* biHeight = 120       */
                0x01, 0x00,             /* biPlanes = 1         */
                0x18, 0x00,             /* biBitCount = 24      */
                0x00, 0x00, 0x00, 0x00,/* biCompression = 0    */
                0x00, 0xe1, 0x00, 0x00,/* biSizeImageSize = 57600
        */
                0xc4, 0x0e, 0x00, 0x00, /* biXpelsPerMeter = 3780
        */
                0xc4, 0x0e, 0x00, 0x00, /* biYpelsPerMeter = 3780
        */
                0x00, 0x00, 0x00, 0x00,/* biClrUsed =  0
        */
                0x00, 0x00, 0x00, 0x00 /* biClrImportant = 0
        */
        }
```

And the image data {B,G,R} can be declared as

```
char    bmpImageData[57600];        /* 160x120x3 bytes      */
```

The data unit in the BMP file is byte and the data format is ASCII. The content of a BMP file can look like:

```
BM6á  6 (  x   _ _á Ä_ Ä_
×åµ×ÆÐÖÓÂ×ÉÒÑÏÓÉÖÕÕÕÖ¹ÒÛÌØÐÑÕØÅÛÔÙÝàÝÔàÒÚÚá×ßý
ÙÜØÙ×ßÓÝÔÛÉÀÓÔÐÍÕ×ÚÉÏÓÕ×ÙÄÔËØÉÕ×Ñ×ÔÁÐèÑÒÝßÑäÒÕ
ÛÃÔÜÏÒÙÈßÐËÕ×ËÕÒÚÚÒÚÎèßÜÝÍÛääâÞØàÝÈÙæààÞÝâÛÄÜêÏÝ
æÛàçßÞéáíßàå ......(continued).........
```

## TMS320 I/O File (.DAT)

The data in both an input data file and an output data file is displayed as heximal. For example,

```
00e5
00c6
00d3
00c9
00cf
 :
```

The data structure declared in C is:

```
typedef struct byte5    /* DSP OUTPUT DATA FILE FORMAT */
{
char    byte0;          /* example : 0023              */
char    byte1;          /* byte0='0', byte1='0'        */
char    byte2;          /* byte2='2', byte3='3'        */
char    byte3;          /* enter takes one byte */
char    enter;
} byte5;
byte5   b[19200], g[19200], r[19200];   /* Image Size = 160*120 */
```

Note that the image data formats in BMP and DAT are different. For example, "A" in BMP means "0041" in DAT.

## TMS320 Assembly File (.ASM)

There are two ways to define data sections in COFF. The first method uses ".sect" for initialized data. For example,

```
.sect   "graph_b"
.word   00d7h
.word   00d7h
   :       :
.sect   "graph_g"
.word   00e5h
.word   00c6h
   :       :
.sect   "graph_r"
.word   00b5h
.word   00d0h
   :       :
```

The second method uses ".usect" for uninitialized data. For example,

```
r_in:   .usect  "graph_r", 19200
g_in:   .usect  "graph_g", 19200
b_in:   .usect  "graph_b", 19200
```

For simulating an image, the image data can be stored to some location in memory space. Therefore, we can create initialized data sections, for example, graph_r, graph_g, graph_b, with color information of the image, then link with the program.

The data structure of the heximal number followed by ".word" defined in C is:

```
typedef struct byte6    /* DSP OUTPUT DATA FILE FORMAT */
        {
        char    byte0;          /* example : 0023              */
        char    byte1;          /* byte0='0', byte1='0'        */
        char    byte2;          /* byte2='2', byte3='3'        */
```

```
        char    byte3;          /* enter occupy one byte        */
        char    hex;            /* ='h'                         */
        char    enter;
        } byte6;
byte6   b[19200], g[19200], r[19200];   /* Image Size = 160*120 */
```

# Converting Data Formats Between BMP and COFF

Three data converting files are included in this application report.

- BMP to DAT

  Converts bit map file to COFF data file

- BMP to ASM

  Converts bit map file to assembly file

- DAT to BMP

  Converts COFF data file to bit map file

## BMP to DAT

The utility "bmp2dat.exe" should be run in MS-DOS using the following steps (the italicized instructions are generated by utility; the bold is entered by the user.):

```
C:\..\bmp2dat
Enter BMP filename(*.bmp):  bug16.bmp
```

Three files will be generated: B.DAT, G.DAT, and R.DAT.

## BMP to ASM

The utility "bmp2asm.exe" should be run in MS-DOS using the following steps (the italicized instructions are generated by utility; the bold is entered by the user.):

```
C:\..\bmp2asm
Enter BMP filename(*.bmp): bug16.bmp
```

Three files will be generated: GRAPH_B.ASM, GRAPH_G.ASM, and GRAPH_R.ASM.

## DAT to BMP

The utility "dat2bmp.exe" should be run in MS-DOS using the following steps (the italicized instructions are generated by utility; the bold are entered by the user.):

```
C:\..\dat2bmp
Enter R data filename:    r_out.dat
Enter G data filename:    g_out.dat
Enter B data filename:    b_out.dat
Enter BMP filename(*.bmp): test.bmp
```

The data files "r_out.dat," "g_out.dat," and "b_out.dat" can be any user's filename. The files can also be the results from the simulator by connecting output to the files. The file names can be defined in the "siminit.cmd" file. Please refer to the TI *TMS320C5x C Source Debugger User's Guide*.

The BMP file, test.bmp, will be generated.

# Appendix A. BMP2DAT.CPP

```
/*********************************************
*     BMP2DAT.CPP                            *
*********************************************/
#include    <stdlib.h>
#include    <stdio.h>

typedef struct byte5               /* DSP OUTPUT DATA FILE FORMAT   */
      {
      char  byte0;                 /* example : 0023             */
      char  byte1;                 /* byte0='0', byte1='0' */
      char  byte2;                 /* byte2='2', byte3='3' */
      char  byte3;                 /* enter occupy one byte */
      char  enter;
      }     byte5;

main()
{
      FILE  *ifp, *ofp;
      char  bmpfile[20];
      int   i=0;
      char  byte2, byte3;
      byte5 b[19200], g[19200], r[19200]; /* Image Size = 160*120 */
      char  bmphead[54];           /* bit map file header */
      char  itemp[57600];          /* Image Size*3 = 160*120*3 bytes */
//-------------------------------------------
//    GET DATA FROM BMP FILE
//-------------------------------------------
   printf("\nEnter BMP filename(*.bmp): ");
      scanf("%s", &bmpfile);

      if ((ifp=fopen(bmpfile,"r"))==NULL)
      {     printf("Cannot open this bmp file \n");
            exit(1);
      };
   fread(bmphead, 1, 54, ifp);
      fread(itemp, 1, 57600, ifp);
      fclose(ifp);
//-----------------------------------------------------
//    DATA TRANSFORMATION : From string to hex
//    example: '0023' -> 23h
//-----------------------------------------------------
      for (i=0;i<19200;i++)   /* The order of ouput data is B G R */
   {
//--B----------------------------------------------
      b[i].byte0 = '0';
      b[i].byte1 = '0';
      byte2 = (itemp[3*i] >> 4) & 0x0f;
      byte3 = itemp[3*i] & 0x0f;
      if ((byte2 >= 0) && (byte2 <= 9))
            b[i].byte2 = byte2 + 0x30;
      if (byte2 >= 10)
            b[i].byte2 = byte2 - 9 + 0x60;
      if ((byte3 >= 0) && (byte3 <= 9))
            b[i].byte3 = byte3 + 0x30;
      if (byte3 >= 10)
            b[i].byte3 = byte3 - 9 + 0x60;
```

```
      b[i].enter = 0x0a;
//--G---------------------------------------------
      g[i].byte0 = '0';
      g[i].byte1 = '0';
      byte2 = (itemp[3*i+1] >> 4) & 0x0f;
      byte3 = itemp[3*i+1] & 0x0f;
      if ((byte2 >= 0) && (byte2 <= 9))
            g[i].byte2 = byte2 + 0x30;
      if (byte2 >= 10)
            g[i].byte2 = byte2 - 9 + 0x60;
      if ((byte3 >= 0) && (byte3 <= 9))
            g[i].byte3 = byte3 + 0x30;
      if (byte3 >= 10)
            g[i].byte3 = byte3 - 9 + 0x60;
      g[i].enter = 0x0a;
//--R---------------------------------------------
      r[i].byte0 = '0';
      r[i].byte1 = '0';
      byte2 = (itemp[3*i+2] >> 4) & 0x0f;
      byte3 = itemp[3*i+2] & 0x0f;
      if ((byte2 >= 0) && (byte2 <= 9))
            r[i].byte2 = byte2 + 0x30;
      if (byte2 >= 10)
            r[i].byte2 = byte2 - 9 + 0x60;
      if ((byte3 >= 0) && (byte3 <= 9))
            r[i].byte3 = byte3 + 0x30;
      if (byte3 >= 10)
            r[i].byte3 = byte3 - 9 + 0x60;
      r[i].enter = 0x0a;
//-------------------------------------------
//    RESTORE TRANSFERED DATA TO AN BMP FILE
//-------------------------------------------
      if ((ofp=fopen("B.dat","w"))==NULL)
      {     printf("Cannot open file test.in\n");
            exit(1);
      };
      fwrite(b, 5, 19200, ofp);
      fclose(ofp);

      if ((ofp=fopen("G.dat","w"))==NULL)
      {     printf("Cannot open file test.in\n");
            exit(1);
      };
      fwrite(g, 5, 19200, ofp);
      fclose(ofp);

      if ((ofp=fopen("R.dat","w"))==NULL)
      {     printf("Cannot open file test.in\n");
            exit(1);
      };
      fwrite(r, 5, 19200, ofp);
      fclose(ofp);
}
```

# Appendix B. BMP2ASM.CPP

```
/*********************************************
*     BMP2ASM.CPP                            *
*********************************************/
#include    <stdlib.h>
#include    <stdio.h>

typedef struct byte5          /* DSP OUTPUT DATA FILE FORMAT */
      {
      char  byte0;            /* example : 0023             */
      char  byte1;            /* byte0='0', byte1='0' */
      char  byte2;            /* byte2='2', byte3='3' */
      char  byte3;            /* enter occupy one byte */
      char  hex;
      char  enter;
      }     byte6;

main()
{
      FILE  *ifp, *ofp;
      char  bmpfile[20];
      int   i=0;
      char  byte2, byte3;
      byte6 b[19200], g[19200], r[19200]; /* Image Size = 160*120 */
      byte6 outtemp[1];
      char  bmphead[54];      /* bit map file header */
      char  itemp[57600];     /* Image Size*3 = 160*120*3 bytes   */
      char  graph_r[17]={' ','.','s','e','c','t',
                         ' ','"','g','r','a','p','h','_','r','"',0x0a};
      char  graph_g[17]={' ','.','s','e','c','t',
                         ' ','"','g','r','a','p','h','_','g','"',0x0a};
      char  graph_b[17]={' ','.','s','e','c','t',
                         ' ','"','g','r','a','p','h','_','b','"',0x0a};
      char  word16[7]=  {' ','.','w','o','r','d',' '};
//-------------------------------------------
//    GET DATA FROM BMP FILE
//-------------------------------------------
   printf("\nEnter BMP filename(*.bmp): ");
      scanf("%s", &bmpfile);

      if ((ifp=fopen(bmpfile,"r"))==NULL)
      {     printf("Cannot open this bmp file \n");
            exit(1);
      };
   fread(bmphead, 1, 54, ifp);
      fread(itemp, 1, 57600, ifp);
      fclose(ifp);

//-------------------------------------------
//    DATA TRANSFORMATION : From string to hex
//    example: '0023' -> 23h
//-------------------------------------------
      for (i=0;i<19200;i++)   /* The order of ouput data is B G R */
   {
//--B-----------------------------------------------
      b[i].byte0 = '0';
      b[i].byte1 = '0';
```

```
      byte2 = (itemp[3*i] >> 4) & 0x0f;
      byte3 = itemp[3*i] & 0x0f;
      if ((byte2 >= 0) && (byte2 <= 9))
            b[i].byte2 = byte2 + 0x30;
      if (byte2 >= 10)
            b[i].byte2 = byte2 - 9 + 0x60;
      if ((byte3 >= 0) && (byte3 <= 9))
            b[i].byte3 = byte3 + 0x30;
      if (byte3 >= 10)
            b[i].byte3 = byte3 - 9 + 0x60;
      b[i].hex = 'h';
      b[i].enter = 0x0a;
//--G-------------------------------------------------
      g[i].byte0 = '0';
      g[i].byte1 = '0';
      byte2 = (itemp[3*i+1] >> 4) & 0x0f;
      byte3 = itemp[3*i+1] & 0x0f;
      if ((byte2 >= 0) && (byte2 <= 9))
            g[i].byte2 = byte2 + 0x30;
      if (byte2 >= 10)
            g[i].byte2 = byte2 - 9 + 0x60;
      if ((byte3 >= 0) && (byte3 <= 9))
            g[i].byte3 = byte3 + 0x30;
      if (byte3 >= 10)
            g[i].byte3 = byte3 - 9 + 0x60;
      g[i].hex = 'h';
      g[i].enter = 0x0a;
//--R-------------------------------------------------
      r[i].byte0 = '0';
      r[i].byte1 = '0';
      byte2 = (itemp[3*i+2] >> 4) & 0x0f;
      byte3 = itemp[3*i+2] & 0x0f;
      if ((byte2 >= 0) && (byte2 <= 9))
            r[i].byte2 = byte2 + 0x30;
      if (byte2 >= 10)
            r[i].byte2 = byte2 - 9 + 0x60;
      if ((byte3 >= 0) && (byte3 <= 9))
            r[i].byte3 = byte3 + 0x30;
      if (byte3 >= 10)
            r[i].byte3 = byte3 - 9 + 0x60;
      r[i].hex = 'h';
      r[i].enter = 0x0a;
      }
//-----------------------------------------
//    RESTORE TRANSFERED DATA TO AN BMP FILE
//-----------------------------------------
      if ((ofp=fopen("graph_b.asm","w"))==NULL)
      {     printf("Cannot open file test.in\n");
            exit(1);
      };
   fwrite(graph_b, 1, 17, ofp);
   for (i=0; i<19200; i++)
   {
      fwrite(word16, 1, 7, ofp);
      outtemp[0] = b[i];
            fwrite(outtemp, 6, 1, ofp);
      }
      fclose(ofp);
```

```
      if ((ofp=fopen("graph_g.asm","w"))==NULL)
      {     printf("Cannot open file test.in\n");
            exit(1);
      };
   fwrite(graph_g, 1, 17, ofp);
   for (i=0; i<19200; i++)
   {
      fwrite(word16, 1, 7, ofp);
      outtemp[0] = g[i];
            fwrite(outtemp, 6, 1, ofp);
   }
      fclose(ofp);

      if ((ofp=fopen("graph_r.asm","w"))==NULL)
      {     printf("Cannot open file test.in\n");
            exit(1);
      };
   fwrite(graph_r, 1, 17, ofp);
   for (i=0; i<19200; i++)
   {
      fwrite(word16, 1, 7, ofp);
      outtemp[0] = r[i];
            fwrite(outtemp, 6, 1, ofp);
   }
      fclose(ofp);
}
```

# Appendix C. DAT2BMP.CPP

```
/*********************************************
*     DAT2BMP.CPP                            *
*********************************************/
#include          <stdlib.h>
#include    <stdio.h>

typedef struct byte5          /* DSP OUTPUT DATA FILE FORMAT */
      {
      char  byte0;            /* example : 0023             */
      char  byte1;            /* byte0='0', byte1='0' */
      char  byte2;            /* byte2='2', byte3='3' */
      char  byte3;            /* enter occupy one byte */
      char  enter;
      }     byte5;

main()
{
      FILE  *ifp, *ofp;
      char  rfile[20],gfile[20],bfile[20],outfile[20];
      int   i=0;
      byte5 b[19200], g[19200], r[19200]; /* Image Size = 160*120 */
      char  otemp[57600];     /* Image Size*3 = 160*120*3 bytes   */
      char  bmpheader[54]=    /* BMP FILE HEADER 54 BYTES         */
            {0x42, 0x4d,            /* bfType =       'BM'   */
            0x36, 0xe1, 0x00, 0x00, /*bfSize =  57654        */
            0x00, 0x00,             /* bfReserved1 =  0      */
            0x00, 0x00,             /* bfReserved2 =  0      */
            0x36, 0x00, 0x00, 0x00, /* bf0ffBits =54         */
            0x28, 0x00, 0x00, 0x00, /* biSize = 40           */
            0xa0, 0x00, 0x00, 0x00, /* biWidth = 160         */
            0x78, 0x00, 0x00, 0x00, /* biHeight = 120        */
            0x01, 0x00,             /* biPlanes = 1          */
            0x18, 0x00,             /* biBitCount = 24       */
            0x00, 0x00, 0x00, 0x00, /* biCompression = 0     */
            0x00, 0xe1, 0x00, 0x00, /* biSizeImageSize = 57600*/
            0xc4, 0x0e, 0x00, 0x00, /* biXpelsPerMeter = 3780*/
            0xc4, 0x0e, 0x00, 0x00, /* biYpelsPerMeter = 3780*/
            0x00, 0x00, 0x00, 0x00, /* biClrUsed =    0      */
            0x00, 0x00, 0x00, 0x00  /* biClrImportant = 0    */
            };
//-------------------------------------------
//    GET DATA FROM 3 DSP OUTPUT DAT FILES
//-------------------------------------------
   printf("\nEnter R data filename: ");
      scanf("%s", &rfile);
   printf("\nEnter G data filename: ");
      scanf("%s", &gfile);
   printf("\nEnter B data filename: ");
      scanf("%s", &bfile);
   printf("\nEnter BMP filename(*.bmp): ");
      scanf("%s", &outfile);

      if ((ifp=fopen(bfile,"r"))==NULL)
      {     printf("Cannot open file test.in\n");
            exit(1);
      };
```

```
      fread(b, 5, 19200, ifp);
      fclose(ifp);

      if ((ifp=fopen(gfile,"r"))==NULL)
      {     printf("Cannot open file test.in\n");
            exit(1);
      };
      fread(g, 5, 19200, ifp);
      fclose(ifp);

      if ((ifp=fopen(rfile,"r"))==NULL)
      {     printf("Cannot open file test.in\n");
            exit(1);
      };
      fread(r, 5, 19200, ifp);
      fclose(ifp);

//-------------------------------------------
//    DATA TRANSFORMATION : From string to hex
//    example: '0023' -> 23h
//-------------------------------------------
      for (i=0;i<19200;i++)   /* The order of ouput data is B G R */
   {
//-----------------------------------------------------------
   if ((b[i].byte2 >= 48) && (b[i].byte2 <= 57))
      otemp[3*i] = 16*(b[i].byte2-48);
   if ((b[i].byte2 >= 65) && (b[i].byte2 <= 70))
      otemp[3*i] = 16*(b[i].byte2-65+10);
   if ((b[i].byte2 >= 97) && (b[i].byte2 <= 102))
      otemp[3*i] = 16*(b[i].byte2-97+10);

   if ((b[i].byte3 >= 48) && (b[i].byte3 <= 57))
      otemp[3*i] = otemp[3*i] + (b[i].byte3-48);
   if ((b[i].byte3 >= 65) && (b[i].byte3 <= 70))
      otemp[3*i] = otemp[3*i] + (b[i].byte3-65+10);
   if ((b[i].byte3 >= 97) && (b[i].byte3 <= 102))
      otemp[3*i] = otemp[3*i] + (b[i].byte3-97+10);
//-----------------------------------------------------------
   if ((g[i].byte2 >= 48) && (g[i].byte2 <= 57))
      otemp[3*i+1] = 16*(g[i].byte2-48);
   if ((g[i].byte2 >= 65) && (g[i].byte2 <= 70))
      otemp[3*i+1] = 16*(g[i].byte2-65+10);
   if ((g[i].byte2 >= 97) && (g[i].byte2 <= 102))
      otemp[3*i+1] = 16*(g[i].byte2-97+10);

   if ((g[i].byte3 >= 48) && (g[i].byte3 <= 57))
      otemp[3*i+1] = otemp[3*i+1] + (g[i].byte3-48);
   if ((g[i].byte3 >= 65) && (g[i].byte3 <= 70))
      otemp[3*i+1] = otemp[3*i+1] + (g[i].byte3-65+10);
   if ((g[i].byte3 >= 97) && (g[i].byte3 <= 102))
      otemp[3*i+1] = otemp[3*i+1] + (g[i].byte3-97+10);
//-----------------------------------------------------------
   if ((r[i].byte2 >= 48) && (r[i].byte2 <= 57))
      otemp[3*i+2] = 16*(r[i].byte2-48);
   if ((r[i].byte2 >= 65) && (r[i].byte2 <= 70))
      otemp[3*i+2] = 16*(r[i].byte2-65+10);
   if ((r[i].byte2 >= 97) && (r[i].byte2 <= 102))
      otemp[3*i+2] = 16*(r[i].byte2-97+10);
```

```
   if ((r[i].byte3 >= 48) && (r[i].byte3 <= 57))
      otemp[3*i+2] = otemp[3*i+2] + (r[i].byte3-48);
   if ((r[i].byte3 >= 65) && (r[i].byte3 <= 70))
      otemp[3*i+2] = otemp[3*i+2] + (r[i].byte3-65+10);
   if ((r[i].byte3 >= 97) && (r[i].byte3 <= 102))
      otemp[3*i+2] = otemp[3*i+2] + (r[i].byte3-97+10);
   }
//-------------------------------------------
//    RESTORE TRANSFERED DATA TO AN BMP FILE
//-------------------------------------------
      if ((ofp=fopen(outfile,"w"))==NULL)
      {     printf("Cannot open output file \n");
            exit(1);
      }

   fwrite(bmpheader, 1, 54, ofp);   /* bit map file header */
   fwrite(otemp, 1, 57600, ofp);          /* bit map file BGR data */

   fclose(ofp);
}
```