TEXAS INSTRUMENTS

Color-Space Format Conversion

Using the TMS320C80 Transfer Controller

Application Report

1998

Digital Signal Processing Solutions

Texas Instruments

Printed in U.S.A., March 1998

SPRA174

Color-Space Format Conversion

Using the TMS320C80 Transfer Controller

Henry Yiu

SPRA174
March 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 applications 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.

Contents

List of Figures

Color-Space Format Conversion

ABSTRACT

This application report describes the conversion of color-picture data from the macroblock (MB) format used by ITU-T Recommendation H.261 to the format used by the Texas Instruments (TI™) TMS320C80 family of digital signal processors (DSPs). Conversion is necessary in some instances because of the difference between the MB format and the TI format and the necessity of the ’C80 to maintain compatibility between the formats in use. This document presents an overview of the methods used to convert the ’C80 for compatibility with existing formats.

1 Introduction

The TMS320C80 transfer controller can be used to process various formats of color-picture video data. The format used by ITU-T Recommendation H.261 (that is, the macroblock, or MB, format) and the format used by TI differ, as shown in Figure 1. The H.261 codec requires all color-picture video data to be in the MB format, while the ’C80 implementation handles the H.261 codec by using the TI format. Conversions to the TI format from the MB format are made without adding excessive overhead to the codec.

Figure 1 shows that the H.261 codec places the MB-format data in memory using the Y-bytes, the 8 x 8 C_r bytes, and the 8 x 8 C_b bytes of MB 1 of group of blocks (GOB) 1. The H.261 codec repeats the process for MB 2 through MB 33. The process is then repeated for GOB 2 through 12.

Figure 1 also shows how picture data (352 pixels x 288 lines) is stored in memory by the TI implementation of the H.261 codec. The TI format puts the 101376 Y bytes (352 x 288), the 25 344 C_b bytes (176 x 144), and the 25344 C_r bytes (176 x 144) in memory, but in a different order from that of the MB format. This implementation is part of the H.320 utility library that is part of the H.320 codec implementation. Both the MB format and the TI format eventually divide the picture into small 8-bit-wide pixels.

The TI-format implementation, with only minor modification, can interface directly with external video capture and display devices. Since the Y, C_b, and C_r data are directly output as data, only slight modification is necessary to drive the VRAM (video random-access memory) buffers. Other manufacturer's implementations, such as GEC Plessey's VP261 chipset, use data in the MB format. For the 'C80 to be compatible with H.261, the TI format must be converted. The transfer controller (TC) must read and implement the program code stored in on-chip memory before locating any programs in off-chip memory. Conversion-code data is placed in off-chip memory to minimize the amount of on-chip memory used for the conversion process.

The optimum method is to use a guide table to offset-guide each packet-table transfer for each 8 x 8 MB. Guide tables must be located in the 'C80 on-chip memory and must have a minimum of 2376 (6 x 33 x 12) entries. One way to accomplish this is to use the TC to perform a variable-patch offset-guided packet-table transfer. The 'C80 must be able to implement the conversion process with a minimum of TC overhead.

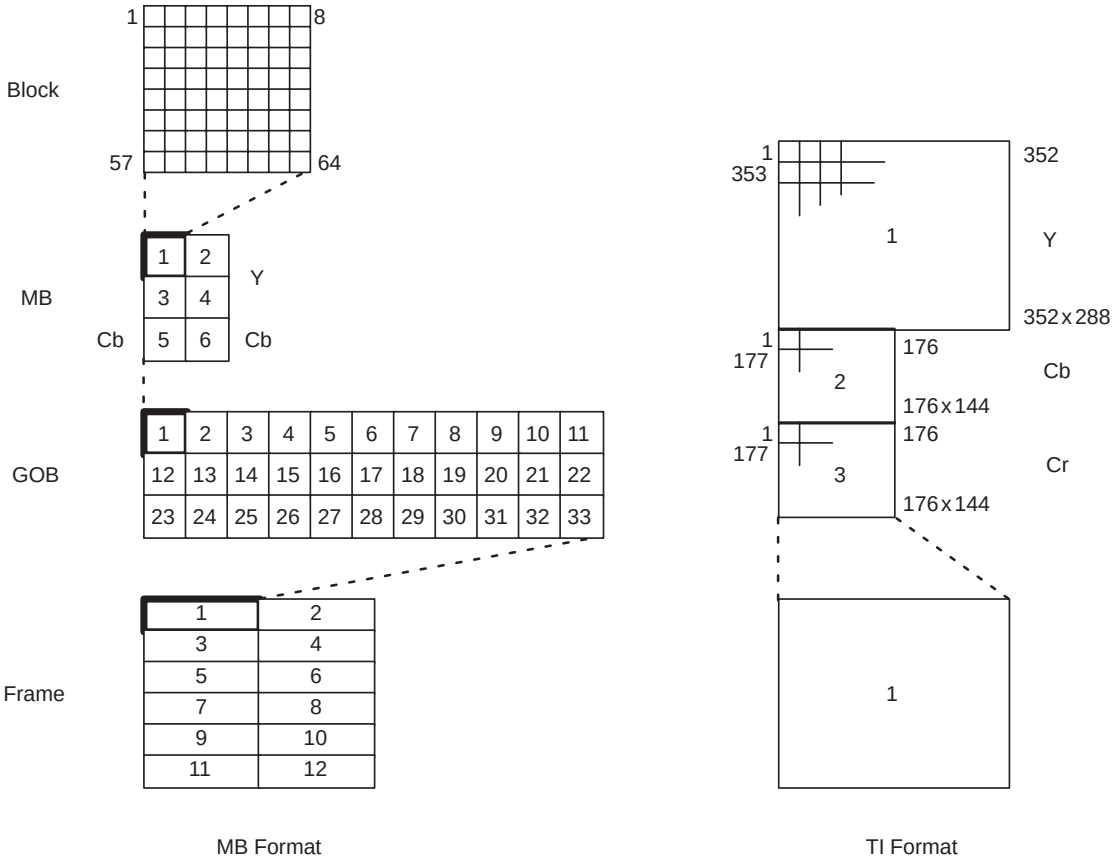


Figure 1. MB and TI Formats

2 Build Your Own Packet-Transfer (PT) Parameter-Update Table

To avoid having large guide tables in on-chip memory, use packet transfers (PTs) to update the parameters in the PT table that are originally used for data transfers. It is possible to set up a PT parameter-update table using off-chip memory. The PT parameter-update table can include different pitch and count values as table entries. Before going to off-chip memory to get the PT parameter-update table, the TC must read on-chip memory to get the PT parameter-update-table address for the PT table that performs the parameter updates. This method increases TC overhead and requires that three conditions be considered.

- The total number of PT parameter-update-table reads must be minimized to avoid increasing the TC overhead required to read a PT table into on-chip memory.
- The total number of PT parameter-update tables must be minimized to reduce on-chip memory usage.
- The size of PT parameter-update tables must be kept small to minimize off-chip memory usage.

Packet-transfer (PT) parameter-update tables can be built several ways using the ’C80 processor. Because it is necessary to minimize PT parameter-update-table read activity, an arrangement that maximizes length and data storage and that is common to both the TI and MB formats should be selected. For H.261 video codecs, the optimal structure is eleven MBs arranged in three lines in a GOB in the MB format. The dimensions of this arrangement match those of an 8 x 176 section of the Y-data signal and an 8 x 88 section of the C_b data and C_r data signals in the TI format. A single dimension-to-dimension PT match is made without the need to update parameters.

Video signals require the transfer of three data items: Y data, C_b data, and C_r data. Use of three separate PT parameter-update tables is wasteful. C_b and C_r data share count and pitch values in both formats. They differ only in the memory start addresses of C_b and C_r. Use of a PT parameter-update table is made to update Y-data and use of the same table can be made to update both C_b and C_r data. Similarly, Y block 1, Y block 2, Y block 3, and Y block 4 share count and pitch values in both formats.

3 Format-Conversion Implementation

Figure 2 shows the implementation described in the previous sections. For purposes of minimizing overhead, this is an optimal system for format conversions. It is not necessary to determine the source and destination sides of the PT parameter-update-table data, because both sides work identically. The transfer sequence is as follows: Y block 1 and Y block 2 of MB 1 through MB 11; C_b of MB 1 through MB 11; Y block 3 and Y block 4 of MB 1 through MB 11; and C_r of MB 1 through MB 11. Only the following three PT tables are needed:

- A PT table for the Y block data
- A PT table for C_r data and C_b data
- A PT table to update the previous PT-table data

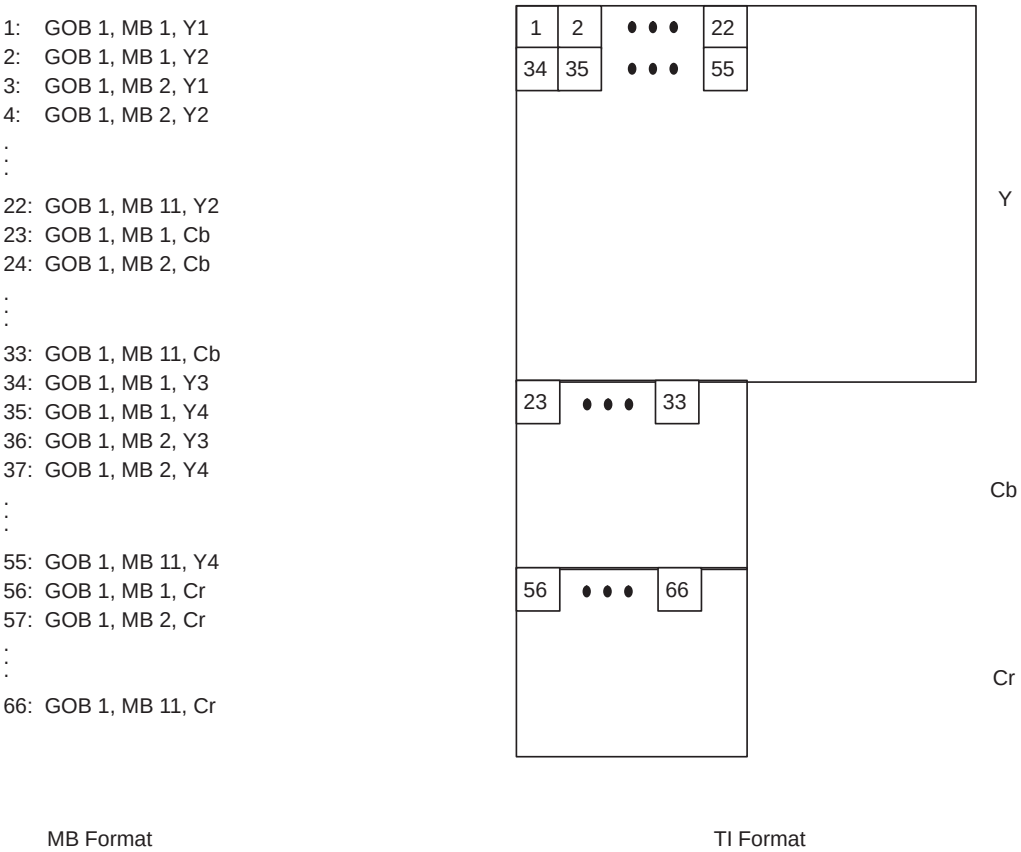


Figure 2. Format-Conversion Sequence for GOB 1 and MB 1 to MB 11

Three linked PT parameter-update tables are used, and a big-endian format is assumed as the transfer converts video data from MB format to TI format. Reverse conversion is possible by setting the X bit (the exchange src/dst) of the PT parameter-update-table options. The most basic of the dimension-to-dimension transfers is used for the MB-to-TI format conversion.

PT-source and PT-destination addresses are placed in the PT parameter-update table. For the TI-format addresses, the PT parameter-update table contains:

- 72 addresses for Y data 288 lines/8 lines per block x 2 sides left and right
- 36 addresses for C_b data 144 lines/8 lines per block x 2 sides left and right
- 36 addresses for C_r data 144 lines/8 lines per block x 2 sides left and right

For the MB format, the PT parameter-update table contains:

- 72 addresses for Y data 12 GOBs x 3 MB lines x 2 (Y1 and Y2, Y3 and Y4)
- 36 addresses for C_b data (12 GOBs x 3 MB lines)
- 36 addresses for C_r data (12 GOBs x 3 MB lines)

The number of addresses used by the TI format and the MB format should be the same. Each address requires four bytes. The number of bytes required by the PT parameter-update table value is:

(72 + 72 + 36 + 36 + 36 + 36) x 4 = 1152.

The PT parameter-update table uses a PT to transfer 16 bytes (the four addresses) from the PT parameter-update table to the PT table for Y and to the PT table for C_bC_r. For transfers to be correct, the address differences between the PT table value for Y and the PT table value for C_bC_r must be known. The address difference is assumed to be 64. Consequently, the PT table for C_bC_r is placed immediately after the PT table for Y. Therefore, the B-pitch value is set to 64.

PT Table for PT Updates†

Next Entry Address = PT Table for Y	PT Options = 0x00000000
Src Start Address = PT Parameter-Update Table	Dst Start Address = PT Table for Y + 8
Src A = 16	Dst A = 8
Src B = 0	Dst B = 2 – 1
Src C = 0	Dst C = 0
Src B Pitch = 0	Dst B Pitch = 64
Src C Pitch = 0	Dst C Pitch = 0

PT Table for Y

Next Entry Address = PT Table for C_bC_r	PT Options = 0x00000000
Src Start Address = To Be Updated	Dst Start Address = To Be Updated
Src A = 8	Dst A = 8
Src B = 16 – 1	Dst B = 8 – 1
Src C = 11 – 1	Dst C = 22 – 1
Src B Pitch = 8	Dst B Pitch = 352
Src C Pitch = 64 x 6	Dst C Pitch = 8

PT Table for C_bC_r

Next Entry Address = PT Table for PT Update	PT Options = 0x00000000
Src Start Address = To Be Updated	Dst Start Address = To Be Updated
Src A = 8	Dst A = 8
Src B = 8 – 1	Dst B = 8 – 1
Src C = 11 – 1	Dst C = 11 – 1
Src B Pitch = 8	Dst B Pitch = 176
Src C Pitch = 64 x 6	Dst C Pitch = 8

PT Parameter-Update Table‡

MB Format Image + 0	TI Format Image + 0
MB Format Image + 64 x 4	TI Format Image + 352 x 288
MB Format Image + 64 x 2	TI Format Image + 352 x 8
MB Format Image + 64 x 5	TI Format Image + 352 x 288 + 176 x 144
MB Format Image + 11 x 64 x 6 + 0	TI Format Image + 352 x 16
MB Format Image + 11 x 64 x 6 + 64 x 4	TI Format Image + 352 x 288 + 176 x 8
MB Format Image + 11 x 64 x 6 + 64 x 2	TI Format Image + 352 x 24
MB Format Image + 11 x 64 x 6 + 64 x 5	TI Format Image + 352 x 288 + 176 x (144 + 8)
.	.
.	.
.	.
etc. ...	etc. ...
MP Instruction Cache Address	MP Instruction Cache Address
MP Instruction Cache Address	MP Instruction Cache Address

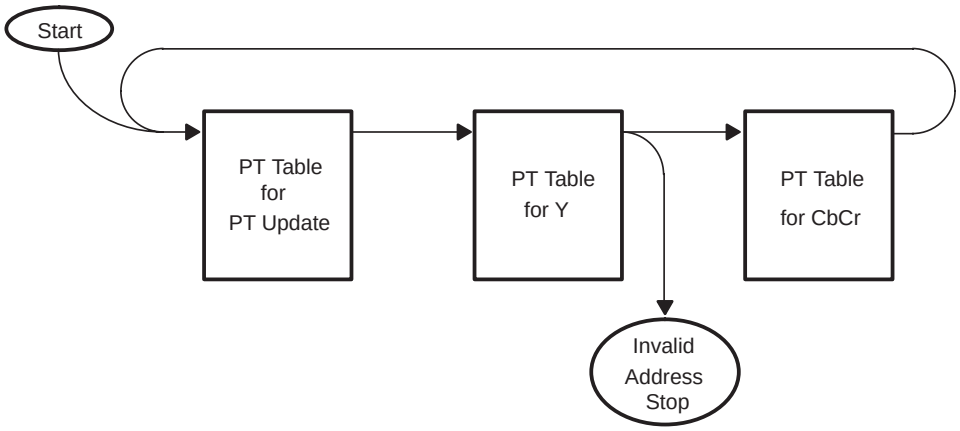
† Start from this
‡ Located in off-chip memory

Figure 3. PT Tables and PT Parameter-Update Table

Figure 4 shows table read cycles and displays the logic flow. Once initiated, the PT does not stop because no stop bit is set in the options field of the PT. For the PT table to stop at the last transfer, the S_{rc} and D_{st} addresses in the PT parameter-update table must be updated so that the last transfer transfers another table to update the PT tables rather than transferring image data. Alternatively, the PT options can be placed in the PT parameter-update table, and a stop bit can be set as the last PT option in the table.

Instead of using complex and wasteful memory methods, the PT parameter-update table can have, as its last address location, an invalid PT location that is the address of the MP instruction cache. This creates a memory-fault interrupt that is serviced by the processor interrupt-service routine. The interrupt-service routine causes the processor to halt, reload the memory addresses, and reset the stop bit. Then, it clears the fault-status (FLTSTS) register bits to enable the PT to continue and to stop normally.

The total number of PT-table reads is 216 [12 GOBs x 3 MB lines x 6 (Update, Y block 1, Y block 2, C_b, Update, Y block 3, Y block 4, and C_r)]. Two additional PT table reads are required to handle the invalid address stops, for a total of 218. The amount of on-chip memory used is 192 bytes (3 PT tables x 64 bytes/table). The amount of off-chip memory used is 1168 bytes (1152 bytes + 16 bytes for invalid addresses). Figure 4 shows how the PT tables are linked and how the format is converted for the entire frame.



PT Table Read			
1–3	Update PT Tables	GOB 1, MB 1–11, Y1 and 2	GOB 1, MB 1–11, C_b
4–6	Update PT Tables	GOB 1, MB 1–11, Y3 and 4	GOB 1, MB 1–11, C_r
7–9	Update PT Tables	GOB 1, MB 12–22, Y1 and 2	GOB 1, MB 12–22, C_b
10–12	Update PT Tables	GOB 1, MB 12–22, Y3 and 4	GOB 1, MB 12–22, C_r
13–15	Update PT Tables	GOB 1, MB 23–33, Y1 and 2	GOB 1, MB 23–33, C_b
16–18	Update PT Tables	GOB 1, MB 23–33, Y3 and 4	GOB 1, MB 23–33, C_r
19–21	Update PT Tables	GOB 2, MB 1–11, Y1 and 2	GOB 2, MB 1–11, C_b
22–24	Update PT Tables	GOB 2, MB 1–11, Y3 and 4	GOB 2, MB 1–11, C_r
.	.	.	.
.	.	.	.
.	.	.	.
211–213	Update PT Tables	GOB 12, MB 23–33, Y1 and 2	GOB 12, MB 23–33, C_b
214–216	Update PT Tables	GOB 12, MB 23–33, Y3 and 4	GOB 12, MB 23–33, C_r
217–218	Specify Invalid Address	Invalid Address Stop	

Figure 4. PT Table Read Cycles

4 Conclusion

The method described in the preceding sections requires:

- 218 PT table reads
- 192 bytes of on-chip memory for use by the PT tables
- 1168 bytes of off-chip memory for use by the PT parameter-update table

Because the frame size is more than 152KB and reading the PT table is equivalent to reading approximately 14KB, the total overhead associated with the PT-table reads is quite small. The overhead associated with the reading and writing of the PT parameter-update table is about 2.3KB.

The color-space conversions in this application report illustrate the capabilities of the transfer controller in handling guided-packet transfers. In particular, if a guided-packet transfer does not meet a specific requirement, the engineer can write a PT parameter-update table to update the parameters in the PT table originally intended for the data transfer. However, the engineer must be aware of the overhead requirements of the transfer controller when using this method.

```
#include <stdlib.h>
#include <task.h>
#include <mp_ptreq.h>
#include <mvp_hw.h>
#define  IMMB  0x80000000     /* MB Format Image address */
#define  IMTI  0x80252000     /* TI Format Image address */
#define  BY1  0
#define  BY2  64*2
#define  BCb  64*4
#define  BCr  64*5
#define  M1   0
#define  M2   11*6*64
#define  M3   22*6*64
#define  G    33*6*64
#define  TY    0
#define  TCb   352*288
#define  TCr   352*288+176*144
#define  TRY   352*8
#define  TCY   176
#define  TRC   176*8
```

```
#define  TCC   88
#pragma DATA_ALIGN(PT_UPDATE_TABLE,64)
unsigned long PT_UPDATE_TABLE[292] = {
  IMMB+M1+BY1,      IMTI+TY,            IMMB+M1+BCb,      IMTI+TCb,
  IMMB+M1+BY2,      IMTI+TY+TRY,        IMMB+M1+BCr,      IMTI+TCr,
  IMMB+M2+BY1,      IMTI+TY+TRY*2,      IMMB+M2+BCb,      IMTI+TCb+TRC,
  IMMB+M2+BY2,      IMTI+TY+TRY*3,      IMMB+M2+BCr,      IMTI+TCr+TRC,
  IMMB+M3+BY1,      IMTI+TY+TRY*4,      IMMB+M3+BCb,      IMTI+TCb+TRC*2,
  IMMB+M3+BY2,      IMTI+TY+TRY*5,      IMMB+M3+BCr,      IMTI+TCr+TRC*2,
  IMMB+M1+BY1+G,    IMTI+TCY+TY,        IMMB+M1+BCb+G,    IMTI+TCC+TCb,
  IMMB+M1+BY2+G,    IMTI+TCY+TY+TRY,    IMMB+M1+BCr+G,    IMTI+TCC+TCr,
  IMMB+M2+BY1+G,    IMTI+TCY+TY+TRY*2,  IMMB+M2+BCb+G,    IMTI+TCC+TCb+TRC,
  IMMB+M2+BY2+G,    IMTI+TCY+TY+TRY*3,  IMMB+M2+BCr+G,    IMTI+TCC+TCr+TRC,
  IMMB+M3+BY1+G,    IMTI+TCY+TY+TRY*4,  IMMB+M3+BCb+G,    IMTI+TCC+TCb+TRC*2,
  IMMB+M3+BY2+G,    IMTI+TCY+TY+TRY*5,  IMMB+M3+BCr+G,    IMTI+TCC+TCr+TRC*2,
  IMMB+M1+BY1+G*2,  IMTI+TY+TRY*6,      IMMB+M1+BCb+G*2,  IMTI+TCb+TRC*3,
  IMMB+M1+BY2+G*2,  IMTI+TY+TRY*7,      IMMB+M1+BCr+G*2,  IMTI+TCr+TRC*3,
  IMMB+M2+BY1+G*2,  IMTI+TY+TRY*8,      IMMB+M2+BCb+G*2,  IMTI+TCb+TRC*4,
  IMMB+M2+BY2+G*2,  IMTI+TY+TRY*9,      IMMB+M2+BCr+G*2,  IMTI+TCr+TRC*4,
  IMMB+M3+BY1+G*2,  IMTI+TY+TRY*10,     IMMB+M3+BCb+G*2,  IMTI+TCb+TRC*5,
  IMMB+M3+BY2+G*2,  IMTI+TY+TRY*11,     IMMB+M3+BCr+G*2,  IMTI+TCr+TRC*5,
  IMMB+M1+BY1+G*3,  IMTI+TCY+TY+TRY*6,  IMMB+M1+BCb+G*3,  IMTI+TCC+TCb+TRC*3,
  IMMB+M1+BY2+G*3,  IMTI+TCY+TY+TRY*7,  IMMB+M1+BCr+G*3,  IMTI+TCC+TCr+TRC*3,
  IMMB+M2+BY1+G*3,  IMTI+TCY+TY+TRY*8,  IMMB+M2+BCb+G*3,  IMTI+TCC+TCb+TRC*4,
  IMMB+M2+BY2+G*3,  IMTI+TCY+TY+TRY*9,  IMMB+M2+BCr+G*3,  IMTI+TCC+TCr+TRC*4,
  IMMB+M3+BY1+G*3,  IMTI+TCY+TY+TRY*10, IMMB+M3+BCb+G*3,  IMTI+TCC+TCb+TRC*5,
  IMMB+M3+BY2+G*3,  IMTI+TCY+TY+TRY*11, IMMB+M3+BCr+G*3,  IMTI+TCC+TCr+TRC*5,
  IMMB+M1+BY1+G*4,  IMTI+TY+TRY*12,     IMMB+M1+BCb+G*4,  IMTI+TCb+TRC*6,
  IMMB+M1+BY2+G*4,  IMTI+TY+TRY*13,     IMMB+M1+BCr+G*4,  IMTI+TCr+TRC*6,
  IMMB+M2+BY1+G*4,  IMTI+TY+TRY*14,     IMMB+M2+BCb+G*4,  IMTI+TCb+TRC*7,
  IMMB+M2+BY2+G*4,  IMTI+TY+TRY*15,     IMMB+M2+BCr+G*4,  IMTI+TCr+TRC*7,
  IMMB+M3+BY1+G*4,  IMTI+TY+TRY*16,     IMMB+M3+BCb+G*4,  IMTI+TCb+TRC*8,
  IMMB+M3+BY2+G*4,  IMTI+TY+TRY*17,     IMMB+M3+BCr+G*4,  IMTI+TCr+TRC*8,
  IMMB+M1+BY1+G*5,  IMTI+TCY+TY+TRY*12, IMMB+M1+BCb+G*5,  IMTI+TCC+TCb+TRC*6,
  IMMB+M1+BY2+G*5,  IMTI+TCY+TY+TRY*13, IMMB+M1+BCr+G*5,  IMTI+TCC+TCr+TRC*6,
  IMMB+M2+BY1+G*5,  IMTI+TCY+TY+TRY*14, IMMB+M2+BCb+G*5,  IMTI+TCC+TCb+TRC*7,
  IMMB+M2+BY2+G*5,  IMTI+TCY+TY+TRY*15, IMMB+M2+BCr+G*5,  IMTI+TCC+TCr+TRC*7,
  IMMB+M3+BY1+G*5,  IMTI+TCY+TY+TRY*16, IMMB+M3+BCb+G*5,  IMTI+TCC+TCb+TRC*8,
  IMMB+M3+BY2+G*5,  IMTI+TCY+TY+TRY*17, IMMB+M3+BCr+G*5,  IMTI+TCC+TCr+TRC*8,
```

```
IMMB+M1+BY1+G*6,  IMTI+TY+TRY*18,     IMMB+M1+BCb+G*6,  IMTI+TCb+TRC*9,
IMMB+M1+BY2+G*6,  IMTI+TY+TRY*19,     IMMB+M1+BCr+G*6,  IMTI+TCr+TRC*9,
IMMB+M2+BY1+G*6,  IMTI+TY+TRY*20,     IMMB+M2+BCb+G*6,  IMTI+TCb+TRC*10,
IMMB+M2+BY2+G*6,  IMTI+TY+TRY*21,     IMMB+M2+BCr+G*6,  IMTI+TCr+TRC*10,
IMMB+M3+BY1+G*6,  IMTI+TY+TRY*22,     IMMB+M3+BCb+G*6,  IMTI+TCb+TRC*11,
IMMB+M3+BY2+G*6,  IMTI+TY+TRY*23,     IMMB+M3+BCr+G*6,  IMTI+TCr+TRC*11,
IMMB+M1+BY1+G*7,  IMTI+TCY+TY+TRY*18, IMMB+M1+BCb+G*7,  IMTI+TCC+TCb+TRC*9,
IMMB+M1+BY2+G*7,  IMTI+TCY+TY+TRY*19, IMMB+M1+BCr+G*7,  IMTI+TCC+TCr+TRC*9,
IMMB+M2+BY1+G*7,  IMTI+TCY+TY+TRY*20, IMMB+M2+BCb+G*7,  IMTI+TCC+TCb+TRC*10,
IMMB+M2+BY2+G*7,  IMTI+TCY+TY+TRY*21, IMMB+M2+BCr+G*7,  IMTI+TCC+TCr+TRC*10,
IMMB+M3+BY1+G*7,  IMTI+TCY+TY+TRY*22, IMMB+M3+BCb+G*7,  IMTI+TCC+TCb+TRC*11,
IMMB+M3+BY2+G*7,  IMTI+TCY+TY+TRY*23, IMMB+M3+BCr+G*7,  IMTI+TCC+TCr+TRC*11,
IMMB+M1+BY1+G*8,  IMTI+TY+TRY*24,     IMMB+M1+BCb+G*8,  IMTI+TCb+TRC*12,
IMMB+M1+BY2+G*8,  IMTI+TY+TRY*25,     IMMB+M1+BCr+G*8,  IMTI+TCr+TRC*12,
IMMB+M2+BY1+G*8,  IMTI+TY+TRY*26,     IMMB+M2+BCb+G*8,  IMTI+TCb+TRC*13,
IMMB+M2+BY2+G*8,  IMTI+TY+TRY*27,     IMMB+M2+BCr+G*8,  IMTI+TCr+TRC*13,
IMMB+M3+BY1+G*8,  IMTI+TY+TRY*28,     IMMB+M3+BCb+G*8,  IMTI+TCb+TRC*14,
IMMB+M3+BY2+G*8,  IMTI+TY+TRY*29,     IMMB+M3+BCr+G*8,  IMTI+TCr+TRC*14,
IMMB+M1+BY1+G*9,  IMTI+TCY+TY+TRY*24, IMMB+M1+BCb+G*9,  IMTI+TCC+TCb+TRC*12,
IMMB+M1+BY2+G*9,  IMTI+TCY+TY+TRY*25, IMMB+M1+BCr+G*9,  IMTI+TCC+TCr+TRC*12,
IMMB+M2+BY1+G*9,  IMTI+TCY+TY+TRY*26, IMMB+M2+BCb+G*9,  IMTI+TCC+TCb+TRC*13,
IMMB+M2+BY2+G*9,  IMTI+TCY+TY+TRY*27, IMMB+M2+BCr+G*9,  IMTI+TCC+TCr+TRC*13,
IMMB+M3+BY1+G*9,  IMTI+TCY+TY+TRY*28, IMMB+M3+BCb+G*9,  IMTI+TCC+TCb+TRC*14,
IMMB+M3+BY2+G*9,  IMTI+TCY+TY+TRY*29, IMMB+M3+BCr+G*9,  IMTI+TCC+TCr+TRC*14,
IMMB+M1+BY1+G*10, IMTI+TY+TRY*30,     IMMB+M1+BCb+G*10, IMTI+TCb+TRC*15,
IMMB+M1+BY2+G*10, IMTI+TY+TRY*31,     IMMB+M1+BCr+G*10, IMTI+TCr+TRC*15,
IMMB+M2+BY1+G*10, IMTI+TY+TRY*32,     IMMB+M2+BCb+G*10, IMTI+TCb+TRC*16,
IMMB+M2+BY2+G*10, IMTI+TY+TRY*33,     IMMB+M2+BCr+G*10, IMTI+TCr+TRC*16,
IMMB+M3+BY1+G*10, IMTI+TY+TRY*34,     IMMB+M3+BCb+G*10, IMTI+TCb+TRC*17,
IMMB+M3+BY2+G*10, IMTI+TY+TRY*35,     IMMB+M3+BCr+G*10, IMTI+TCr+TRC*17,
IMMB+M1+BY1+G*11, IMTI+TCY+TY+TRY*30, IMMB+M1+BCb+G*11, IMTI+TCC+TCb+TRC*15,
IMMB+M1+BY2+G*11, IMTI+TCY+TY+TRY*31, IMMB+M1+BCr+G*11, IMTI+TCC+TCr+TRC*15,
IMMB+M2+BY1+G*11, IMTI+TCY+TY+TRY*32, IMMB+M2+BCb+G*11, IMTI+TCC+TCb+TRC*16,
IMMB+M2+BY2+G*11, IMTI+TCY+TY+TRY*33, IMMB+M2+BCr+G*11, IMTI+TCC+TCr+TRC*16,
IMMB+M3+BY1+G*11, IMTI+TCY+TY+TRY*34, IMMB+M3+BCb+G*11, IMTI+TCC+TCb+TRC*17,
IMMB+M3+BY2+G*11, IMTI+TCY+TY+TRY*35, IMMB+M3+BCr+G*11, IMTI+TCC+TCr+TRC*17,
0x01818000,       0x01818000,         0x01818000,       0x01818000      };
```

```
unsigned int  PT_flag;
void MBtoTIFormat ()
{
        long  sema;
        PTREQ *ptu, *pty, *ptc;
        ptu = (PTREQ*) (0x01010300);            /* Empty MP para RAM   */
        pty = (PTREQ*) (0x01010340);            /* PT for Y table      */
        ptc = (PTREQ*) (0x01010380);            /* Must be next to Y   */
        sema = TaskOpenSema (–1, 0);
        ptu–>link = pty;                        /* point to next PT    */
        ptu–>word[0] = 0x00000000;              /* PT option           */
        ptu–>word[1] = (long)PT_UPDATE_TABLE;   /* Src address         */
        ptu–>word[2] = ((long)pty) + 8;         /* Dst address         */
        ptu–>word[3] = (0 << 16) | 16;          /* Src B << 16 | Src A */
        ptu–>word[4] = (1 << 16) | 8;           /* Dst B << 16 | Dst A */
        ptu–>word[5] = 0;                       /* Src C count         */
        ptu–>word[6] = 0;                       /* Dst C count         */
        ptu–>word[7] = 0;                       /* Src B pitch         */
        ptu–>word[8] = 64;                      /* Dst B pitch         */
        ptu–>word[9] = 0;                       /* Src C pitch         */
        ptu–>word[10] = 0;                      /* Dst C pitch         */
        ptu–>word[11] = 0;                      /* Color/Transparency  */
        ptu–>word[12] = 0;                      /* Color/Transparency  */
        pty–>link = ptc;                        /* point to next PT    */
        pty–>word[0] = 0x00000000;              /* PT option           */
        pty–>word[1] = 0x0;                     /* Src address         */
        pty–>word[2] = 0x0;                     /* Dst address         */
        pty–>word[3] = (15 << 16) | 8;          /* Src B << 16 | Src A */
        pty–>word[4] = (7 << 16) | 8;           /* Dst B << 16 | Dst A */
        pty–>word[5] = 10;                      /* Src C count         */
        pty–>word[6] = 21;                      /* Dst C count         */
        pty–>word[7] = 8;                       /* Src B pitch         */
        pty–>word[8] = 352;                     /* Dst B pitch         */
        pty–>word[9] = 384;                     /* Src C pitch         */
        pty–>word[10] = 8;                      /* Dst C pitch         */
        pty–>word[11] = 0;                      /* Color/Transparency  */
        pty–>word[12] = 0;                      /* Color/Transparency  */
        ptc–>link = ptu;                        /* point to next PT    */
        ptc–>word[0] = 0x00000000;              /* PT option           */
```

```
    ptc->word[1] = 0x0;                    /* Src address         */
    ptc->word[2] = 0x0;                    /* Dst address         */
    ptc->word[3] = (7 << 16) | 8;          /* Src B << 16 | Src A */
    ptc->word[4] = (7 << 16) | 8;          /* Dst B << 16 | Dst A */
    ptc->word[5] = 10;                     /* Src C count         */
    ptc->word[6] = 10;                     /* Dst C count         */
    ptc->word[7] = 8;                      /* Src B pitch         */
    ptc->word[8] = 176;                    /* Dst B pitch         */
    ptc->word[9] = 384;                    /* Src C pitch         */
    ptc->word[10] = 8;                     /* Dst C pitch         */
    ptc->word[11] = 0;                     /* Color/Transparency  */
    ptc->word[12] = 0;                     /* Color/Transparency  */
    PT_flag = 1;
    PtReqIssue (ptu, sema);                /* Issue it.      */
    TaskWaitSema (sema);                   /* Wait for done. */
    TaskCloseSema (sema);
    PT_flag = 0;
}
void isrMemoryFault ()
{
if (PT_flag == 1 && (FLTSTS & 0x1))          /* If this PT caused the fault */
  {
*((long*)(0x01010008)) = (long)0x01010380;  /* Write correct addr to suspended
                                               area */
*((long*)(0x0101000C)) = (long)0x01010380;  /* Write correct addr to suspended
                                                area */
*((long*)(0x01010004)) |= (long)0x80000000; /* Put stop bit */
FLTSTS |= 0x1;                              /* Clear MP PT fault */
INTPEN |= 0x4000;                           /* Clear memory fault */
  }
  else
    _isrMemoryFault ();                     /* Real memory fault */
}
```

Appendix A Glossary

address: Program-code or data-storage location

bit: One-eighth of a byte

buffer: An intermediate storage register

byte: Eight bits; the smallest data character size

'C80: TMS320C80; a TI digital-signal-processor chip

Cb: Color-difference signal

chipset: One or more VLSIs used to accomplish a specific task

CIF: Common intermediate format

CODEC or codec: enCOder/DECoder or COmpression/DECompression

Cr: Color-difference signal

DSP: Digital signal processor

Dst: The three lowest bits of register code for field moves

GBSC: Group-of-blocks start code

GN: Group number

GOB: Group of blocks

IB: Image block (8 pixels x 8 lines)

ITU: International Telecommunications Union

k or K: 1000 (one thousand)

M: 1000000 (one million)

MB: Macroblock

PT: Packet transfer

PUT: Parameter-update table

QCIF: Quarter common intermediate format; 3176 pixels by 144 lines

Src: The three lowest bits of register code for nonfield moves, register-to-register moves, or D-register source-code moves

TI: Texas Instruments Incorporated

Y: Brilliance-intensity signal