

Implementing the Color Space Transformation Algorithm Using the TMS320C2xx

APPLICATION REPORT: SPRA364

*Vivian Shao
Field Application Engineer
Customer Application Center
Semiconductor Sales and Marketing
Texas Instruments – Taiwan*

*Digital Signal Processing Solutions
March 1997*



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.

Contents

Abstract.....	7
Introduction	8
Software Description.....	10
Variable Declaration and Initial Values	10
Transforming (R, G, B) to (Y, U, V)	11
Appendix A. RGB2YUV.ASM.....	13
Appendix B. VECTORS.ASM.....	16
Appendix C. RGB2YUV.CMD	17
Appendix D. RGB2YUV.LOG.....	18

Figures

Figure 1. Image Data (R, G, B) Storage Format in Memory	8
Figure 2. Image Data (Y, U, V) Storage Format in Memory	9

Examples

Example 1. Defining Variables in 'C2xx Assembly Language Code	10
Example 2. Initial Values of Coefficients Defined in .data Section	10
Example 3. Initializing Coefficients in the Program	11
Example 4. Pointing to the Input and Output Data Sections	11
Example 5. Calculating (Y, U, V) From the Input Data (R, G, B).....	12

Implementing the Color Space Transformation Algorithm Using the TMS320C2xx

Abstract

The Texas Instruments (TI™) TMS320 family of digital signal processors (DSPs) is widely used for digital image processing. Color space transformation, one of the basic image processing algorithms, is used for digital still camera, color printer, color scanner, and other video product applications.

This application report describes how to implement the algorithm of transforming the image color space from (R, G, B) to (Y, U, V) using TMS320C2xx assembly language. With the information provided, users can implement other color space transformations, such as (Y, U, V) to (R, G, B), (R, G, B) to (Y, Cr, Cb), etc., by modifying the transformation coefficient matrix.



Introduction

Many different color spaces are used to represent a color image, for example, (R, G, B), (Y, U, V), (Y, Cr, Cb), (Y, I, Q), etc. Equation 1 represents the general formula for the color space transformation.

$$\begin{bmatrix} S_1' \\ S_2' \\ S_3' \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} & C_{13} \\ C_{21} & C_{22} & C_{23} \\ C_{31} & C_{32} & C_{33} \end{bmatrix} \begin{bmatrix} S_1 \\ S_2 \\ S_3 \end{bmatrix} \quad (\text{Equation 1})$$

where

$[S_1 \ S_2 \ S_3]^T$ is the original color space.

$[S_1' \ S_2' \ S_3']^T$ is the transformed color space.

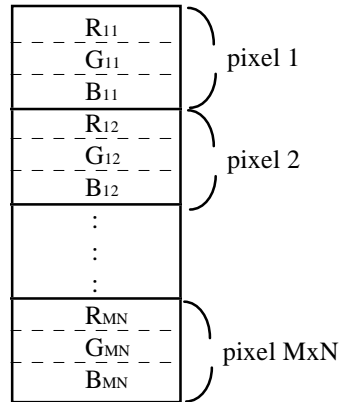
$[C_{i,j}]$ is the coefficient matrix of transformation.

Equation 2 transforms (R, G, B) space to (Y, U, V) space.

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.148 & -0.289 & 0.437 \\ 0.615 & -0.515 & -0.100 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \quad (\text{Equation 2})$$

Assume that the size of an image frame is M x N pixels, where M is the number of pixels/line, N is the number of lines/frame, and each pixel contains three pieces of information: R, G, and B. The data is stored in memory as shown in Figure 1.

Figure 1. Image Data (R, G, B) Storage Format in Memory



After calculating Equation 2, the transformed data (Y, U, V) is stored in another section of memory, as shown in Figure 2.

Figure 2. Image Data (Y, U, V) Storage Format in Memory

Y ₁₁
Y ₁₂
⋮
Y _{MN}
U ₁₁
U ₁₂
⋮
U _{MN}
V ₁₁
V ₁₂
⋮
V _{MN}



Software Description

Variable Declaration and Initial Values

The transformation program includes the following variables:

INPUT DATA:	{R11, G11, B11, R12, G12, B12,, RMN, GMN, BMN}
OUTPUT DATA	{Y11, Y12,, YMN}
	{U11, U12,, UMN}
	{V11, V12,, VMN}
COEFFICIENTS:	{C11, C12, C13
	C21, C22, C23
	C31, C32, C33}

Example 1 shows how to define the variables in 'C2xx assembly language code.

Example 1. Defining Variables in 'C2xx Assembly Language Code

```

;-----
;   Define variables & data blocks
;-----
C11      .usect ".r2y_var", 1           ; coefficients of transformation matrix
C12      .usect ".r2y_var", 1
C13      .usect ".r2y_var", 1
C21      .usect ".r2y_var", 1
C22      .usect ".r2y_var", 1
C23      .usect ".r2y_var", 1
C31      .usect ".r2y_var", 1
C32      .usect ".r2y_var", 1
C33      .usect ".r2y_var", 1

RGB_BLOCK .usect ".rgb_blk", 3*8*8      ; input data array
Y_BLOCK   .usect ".yuv_blk", 8*8        ; output data arrays
U_BLOCK   .usect ".yuv_blk", 8*8
V_BLOCK   .usect ".yuv_blk", 8*8

```

Define the initial values of the coefficients in the .data section as shown in Example 2.

Example 2. Initial Values of Coefficients Defined in .data Section

```

.data
;-----
;   Y   0.299  0.587  0.114           R
;   U   = -0.148 -0.289  0.437       G
;   V   0.615  -0.515 -0.100       B
;-----
R2Y_COFF .word  32768*299/1000, 32768*587/1000, 32768*114/1000
          .word -32768*148/1000, -32768*289/1000, 32768*437/1000
          .word  32768*615/1000, -32768*515/1000, -32768*100/1000

```



Initialize these coefficients in the program, as shown in Example 3.

Example 3. Initializing Coefficients in the Program

```
-----  
; RGB to YUV coefficient initialization  
-----  
RGB2YUV_INIT:  
    SPM    0  
    SETC   SXM  
    CLRC   OVM  
    MAR    *, AR2  
    LAR    AR2, #C11  
    RPT    #9-1  
    BLPD   #R2Y_COFF, *+  
    RET
```

Transforming (R, G, B) to (Y, U, V)

We use four auxiliary registers as pointers to the input data section (RGB_BLOCK) and output data sections (Y_BLOCK, U_BLOCK, V_BLOCK) (see Example 4).

Example 4. Pointing to the Input and Output Data Sections

```
RGB2YUV:  
-----  
; AR assignment:  
; AR2 -> RGB_BLOCK  
; AR3 -> Y_BLOCK  
; AR4 -> U_BLOCK  
; AR5 -> V_BLOCK  
; AR6 -> Counter for frame  
; AR7 -> Counter for line  
-----  
    LAR    AR0, #2  
    LAR    AR2, #RGB_BLOCK  
    LAR    AR3, #Y_BLOCK  
    LAR    AR4, #U_BLOCK  
    LAR    AR5, #V_BLOCK  
    LAR    AR6, #64-1  
    MAR    *, AR2  
    LDP    #C11
```

Then calculate (Y, U, V) from the input data (R, G, B),



Example 5. Calculating (Y, U, V) From the Input Data (R, G, B)

```

;-----
;   RGB to YUV Transformation:
;-----
;   Y      C11  C12  C13      R
;   U  =    C21  C22  C23      G
;   V      C31  C32  C33      B
;-----
R2Y_LOOP:
    LT      *+                                ; Y1 = C11 * R + C12 * G + C13 * B
    MPY     C11
    LTP     *+
    MPY     C12
    LTA     *0-, AR3                        ; AR3 -> Y_BLOCK
    MPY     C13
    APAC
    SACH     *+, 1, AR2

    LT      *+                                ; U = C21 * R + C22 * G + C23 * B
    MPY     C21
    LTP     *+
    MPY     C22
    LTA     *0-, AR4                        ; AR4 -> U_BLOCK
    MPY     C23
    APAC
    SACH     *+, 1, AR2

    LT      *+                                ; V = C31 * R + C32 * G + C33 * B
    MPY     C31
    LTP     *+
    MPY     C32
    LTA     *+, AR5                        ; AR5 -> V_BLOCK
    MPY     C33
    APAC
    SACH     *+, 1, AR6

    BANZ    R2Y_LOOP, *-, AR2

    RET

```

Note that C_{ij} s are in S.Q15 format, and input data are in S16.Q0 format. After multiplying input data with $C_{i,j}$, the result stored in the Product register is in S17.Q15 format. Because $PM=0$, which is set in `RGB2YUV_INIT`, the data in the accumulator will be in S17.Q15 format. Therefore, the data in the accumulator needs to be shifted one bit left before storing back to the memory. The instruction to store with `shifr` is

SACH *+, 1, ARx

The complete assembly file, linker command file, and testing pattern file are attached in the appendixes that follow.



Appendix A. RGB2YUV.ASM

```
-----
;
;   Filename:          RGB2YUV.ASM
;   Description:       Color space transformation {R,G,B} to {Y,U,V}
;   Author:           Vivian Shao
;   Date:             03/20/1997
;
-----
        .def  start
        .def  RGB2YUV_INIT, RGB2YUV

        .data
;-----
;   Y   =  0.299  0.587  0.114      R
;   U   = -0.148 -0.289  0.437      G
;   V   =  0.615 -0.515 -0.100      B
;-----
R2Y_COFF
        .word 32768*299/1000, 32768*587/1000, 32768*114/1000
        .word -32768*148/1000, -32768*289/1000, 32768*437/1000
        .word 32768*615/1000, -32768*515/1000, -32768*100/1000
;-----
;   Define variables & data blocks
;-----
C11     .usect ".r2y_var", 1
C12     .usect ".r2y_var", 1
C13     .usect ".r2y_var", 1
C21     .usect ".r2y_var", 1
C22     .usect ".r2y_var", 1
C23     .usect ".r2y_var", 1
C31     .usect ".r2y_var", 1
C32     .usect ".r2y_var", 1
C33     .usect ".r2y_var", 1

RGB_BLOCK .usect ".rgb_blk", 3*8*8
Y_BLOCK   .usect ".yuv_blk", 8*8
U_BLOCK   .usect ".yuv_blk", 8*8
V_BLOCK   .usect ".yuv_blk", 8*8
;-----
;   RGB2YUV CODE SECTION
;-----
        .text
start:
        CALL  RGB2YUV_INIT
        CALL  RGB2YUV
        B     start
;-----
;   RGB to YUV coefficient initialization
;-----
RGB2YUV_INIT:
        SPM  0
        SETC SXM
```



```
CLRC OVM
MAR *, AR2
LAR AR2, #C11
RPT #9-1
BLPD #R2Y_COFF, *+
RET
```

```
-----
; RGB to YUV Transformation:
;
;
; Y      C11  C12  C13  R
; U  =   C21  C22  C23  G
; V      C31  C32  C33  B
;
-----
```

RGB2YUV:

```
-----
; AR assignment:
; AR2 -> RGB_BLOCK
; AR3 -> Y_BLOCK
; AR4 -> U_BLOCK
; AR5 -> V_BLOCK
; AR6 -> Counter for frame
; AR7 -> Counter for line
;
-----
```

```
LAR AR0, #2
LAR AR2, #RGB_BLOCK
LAR AR3, #Y_BLOCK
LAR AR4, #U_BLOCK
LAR AR5, #V_BLOCK
LAR AR6, #64-1
MAR *, AR2
LDP #C11
```

```
-----
; Multiplication:
; Cij:      S.Q15
; R,G,B:    S16.Q0
; ACC:      S17.Q15 (PM=0)
; Need the integer part (S16) => SACH x,1 ; shift left 1 bit
;
-----
```

R2Y_LOOP:

```
LT *+ ; Y1 = C11 * R + C12 * G + C13 * B
MPY C11
LTP *+
MPY C12
LTA *0-, AR3 ; AR3 -> Y_BLOCK
MPY C13
APAC
SACH *+, 1, AR2

LT *+ ; U = C21 * R + C22 * G + C23 * B
MPY C21
LTP *+
MPY C22
LTA *0-, AR4 ; AR4 -> U_BLOCK
MPY C23
APAC
```



```
SACH    *, 1, AR2

LT      *+                                ; V = C31 * R + C32 * G + C33 * B
MPY     C31
LTP     *+
MPY     C32
LTA     *, AR5                            ; AR5 -> V_BLOCK
MPY     C33
APAC
SACH    *, 1, AR6
BANZ    R2Y_LOOP, *, AR2

RET
```



Appendix B. VECTORS.ASM

```
-----
;
; Filename: VECTORS.ASM
; Description: Define vector table of C2xx
;
; Author: Vivian Shao
; Date: 11/15/1996
;
-----
        .def      reset
        .ref      start

        .sect     "vectors"
reset:  B         start
```




Appendix C. RGB2YUV.CMD

```

/*****
/*
/* Usage:    dsplnk <obj files...> -o <out file> -m <map file> lab.cmd    */
/*
/*****
vectors.obj
rgb2yuv.obj
-v0
-m rgb2yuv.map
-o rgb2yuv.out

MEMORY
{
    /* Program Space */
    PAGE 0 :
        VECS      : origin = 0h , length = 040h    /* Vectors */
        PROG      : origin = 040h , length = 0FC0h  /* 4K ROM */

    /* Data Space */
    PAGE 1 :
        MMREGS     : origin = 0h , length = 60h    /* MMRS */
        B2          : origin = 060h , length = 020h /* On-chip DARAM B2 */
        B0          : origin = 0200h , length = 0100h /* B0 */
        B1          : origin = 0300h , length = 0100h /* B1 */
        VARS        : origin = 1000h , length = 0080h
        SARAM       : origin = 1080h , length = 0F80h /* Internal RAM */
}

/*-----*/
/* SECTIONS ALLOCATION */
/*-----*/
SECTIONS
{
    vectors :    {} > VECS      PAGE 0          /*INTERRUPT VECTOR TABLE */
    .text   :    {} > PROG      PAGE 0          /* CODE */
    .data   :    {} > PROG      PAGE 0          /*INITIALIZATION DATA TABLES */
    .bss    :    {} > B0        PAGE 1          /* UNINITIALIZED DATA */
    .r2y_var:    {} > VARS      PAGE 1          /* variables in one page */
    .rgb_blk:    {} > SARAM      PAGE 1          /* RGB data buffer */
    .yuv_blk:    {} > SARAM      PAGE 1          /* YUV data buffer */
}

```



Appendix D. RGB2YUV.LOG

```
wa st0>>13, ARP=,x
wa st0<<7, DP=,x
mem Y_BLOCK
fill 0x1080, 1, 1, 10
fill 0x1081, 1, 1, 20
fill 0x1082, 1, 1, 30
fill 0x1083, 1, 1, 10
fill 0x1084, 1, 1, 20
fill 0x1085, 1, 1, 30
fill 0x1086, 1, 1, 10
fill 0x1087, 1, 1, 20
fill 0x1088, 1, 1, 30
fill 0x1089, 1, 1, 10
fill 0x108a, 1, 1, 20
fill 0x108b, 1, 1, 30
fill 0x108c, 1, 1, 10
fill 0x108d, 1, 1, 20
fill 0x108e, 1, 1, 30
fill 0x108f, 1, 1, 10
fill 0x1090, 1, 1, 20
fill 0x1091, 1, 1, 30
fill 0x1092, 1, 1, 10
fill 0x1093, 1, 1, 20
fill 0x1094, 1, 1, 30
fill 0x1095, 1, 1, 10
fill 0x1096, 1, 1, 20
fill 0x1097, 1, 1, 30
```

```
fill 0x1098, 1, 1, 10
fill 0x1099, 1, 1, 20
fill 0x109a, 1, 1, 30
fill 0x109b, 1, 1, 10
fill 0x109c, 1, 1, 20
fill 0x109d, 1, 1, 30
fill 0x109e, 1, 1, 10
fill 0x109f, 1, 1, 20
fill 0x10a0, 1, 1, 30
fill 0x10a1, 1, 1, 10
fill 0x10a2, 1, 1, 20
fill 0x10a3, 1, 1, 30
fill 0x10a4, 1, 1, 10
fill 0x10a5, 1, 1, 20
fill 0x10a6, 1, 1, 30
fill 0x10a7, 1, 1, 10
fill 0x10a8, 1, 1, 20
fill 0x10a9, 1, 1, 30
fill 0x10aa, 1, 1, 10
fill 0x10ab, 1, 1, 20
fill 0x10ac, 1, 1, 30
fill 0x10ad, 1, 1, 10
fill 0x10ae, 1, 1, 20
fill 0x10af, 1, 1, 30
```



```
fill 0x10b0, 1, 1, 10
fill 0x10b1, 1, 1, 20
fill 0x10b2, 1, 1, 30
fill 0x10b3, 1, 1, 10
fill 0x10b4, 1, 1, 20
fill 0x10b5, 1, 1, 30
fill 0x10b6, 1, 1, 10
fill 0x10b7, 1, 1, 20
fill 0x10b8, 1, 1, 30
fill 0x10b9, 1, 1, 10
fill 0x10ba, 1, 1, 20
fill 0x10bb, 1, 1, 30
fill 0x10bc, 1, 1, 10
fill 0x10bd, 1, 1, 20
fill 0x10be, 1, 1, 30
fill 0x10bf, 1, 1, 10
fill 0x10c0, 1, 1, 20
fill 0x10c1, 1, 1, 30
fill 0x10c2, 1, 1, 10
fill 0x10c3, 1, 1, 20
fill 0x10c4, 1, 1, 30
fill 0x10c5, 1, 1, 10
fill 0x10c6, 1, 1, 20
fill 0x10c7, 1, 1, 30

fill 0x10c8, 1, 1, 10
fill 0x10c9, 1, 1, 20
fill 0x10ca, 1, 1, 30
fill 0x10cb, 1, 1, 10
fill 0x10cc, 1, 1, 20
fill 0x10cd, 1, 1, 30
fill 0x10ce, 1, 1, 10
fill 0x10cf, 1, 1, 20
fill 0x10d0, 1, 1, 30
fill 0x10d1, 1, 1, 10
fill 0x10d2, 1, 1, 20
fill 0x10d3, 1, 1, 30
fill 0x10d4, 1, 1, 10
fill 0x10d5, 1, 1, 20
fill 0x10d6, 1, 1, 30
fill 0x10d7, 1, 1, 10
fill 0x10d8, 1, 1, 20
fill 0x10d9, 1, 1, 30
fill 0x10da, 1, 1, 10
fill 0x10db, 1, 1, 20
fill 0x10dc, 1, 1, 30
fill 0x10dd, 1, 1, 10
fill 0x10de, 1, 1, 20
fill 0x10df, 1, 1, 30

fill 0x10e0, 1, 1, 10
fill 0x10e1, 1, 1, 20
fill 0x10e2, 1, 1, 30
fill 0x10e3, 1, 1, 10
fill 0x10e4, 1, 1, 20
```



fill 0x10e5, 1, 1, 30
fill 0x10e6, 1, 1, 10
fill 0x10e7, 1, 1, 20
fill 0x10e8, 1, 1, 30
fill 0x10e9, 1, 1, 10
fill 0x10ea, 1, 1, 20
fill 0x10eb, 1, 1, 30
fill 0x10ec, 1, 1, 10
fill 0x10ed, 1, 1, 20
fill 0x10ee, 1, 1, 30
fill 0x10ef, 1, 1, 10
fill 0x10f0, 1, 1, 20
fill 0x10f1, 1, 1, 30
fill 0x10f2, 1, 1, 10
fill 0x10f3, 1, 1, 20
fill 0x10f4, 1, 1, 30
fill 0x10f5, 1, 1, 10
fill 0x10f6, 1, 1, 20
fill 0x10f7, 1, 1, 30

fill 0x10f8, 1, 1, 10
fill 0x10f9, 1, 1, 20
fill 0x10fa, 1, 1, 30
fill 0x10fb, 1, 1, 10
fill 0x10fc, 1, 1, 20
fill 0x10fd, 1, 1, 30
fill 0x10fe, 1, 1, 10
fill 0x10ff, 1, 1, 20
fill 0x1100, 1, 1, 30
fill 0x1101, 1, 1, 10
fill 0x1102, 1, 1, 20
fill 0x1103, 1, 1, 30
fill 0x1104, 1, 1, 10
fill 0x1105, 1, 1, 20
fill 0x1106, 1, 1, 30
fill 0x1107, 1, 1, 10
fill 0x1108, 1, 1, 20
fill 0x1109, 1, 1, 30
fill 0x110a, 1, 1, 10
fill 0x110b, 1, 1, 20
fill 0x110c, 1, 1, 30
fill 0x110d, 1, 1, 10
fill 0x110e, 1, 1, 20
fill 0x110f, 1, 1, 30

fill 0x1110, 1, 1, 10
fill 0x1111, 1, 1, 20
fill 0x1112, 1, 1, 30
fill 0x1113, 1, 1, 10
fill 0x1114, 1, 1, 20
fill 0x1115, 1, 1, 30
fill 0x1116, 1, 1, 10
fill 0x1117, 1, 1, 20
fill 0x1118, 1, 1, 30
fill 0x1119, 1, 1, 10



fill 0x111a, 1, 1, 20
fill 0x111b, 1, 1, 30
fill 0x111c, 1, 1, 10
fill 0x111d, 1, 1, 20
fill 0x111e, 1, 1, 30
fill 0x111f, 1, 1, 10
fill 0x1120, 1, 1, 20
fill 0x1121, 1, 1, 30
fill 0x1122, 1, 1, 10
fill 0x1123, 1, 1, 20
fill 0x1124, 1, 1, 30
fill 0x1125, 1, 1, 10
fill 0x1126, 1, 1, 20
fill 0x1127, 1, 1, 30

fill 0x1128, 1, 1, 10
fill 0x1129, 1, 1, 20
fill 0x112a, 1, 1, 30
fill 0x112b, 1, 1, 10
fill 0x112c, 1, 1, 20
fill 0x112d, 1, 1, 30
fill 0x112e, 1, 1, 10
fill 0x112f, 1, 1, 20
fill 0x1130, 1, 1, 30
fill 0x1131, 1, 1, 10
fill 0x1132, 1, 1, 20
fill 0x1133, 1, 1, 30
fill 0x1134, 1, 1, 10
fill 0x1135, 1, 1, 20
fill 0x1136, 1, 1, 30
fill 0x1137, 1, 1, 10
fill 0x1138, 1, 1, 20
fill 0x1139, 1, 1, 30
fill 0x113a, 1, 1, 10
fill 0x113b, 1, 1, 20
fill 0x113c, 1, 1, 30
fill 0x113d, 1, 1, 10
fill 0x113e, 1, 1, 20
fill 0x113f, 1, 1, 30