Chrontel

CH7025/26 SDTV VGA HDTV ENCODER PROGRAMMING GUIDE

Revision 2.02
February 25, 2008
Prepared By: Junfeng

CONTENTS

PREFACE

There are two chapters and three appendixes in this document:

Chapter1 introduces how to use DAC connection detect function of CH7025/26.

Chapter 2 introduces how to use the programming tool of CH7025/26.

Appendix A introduces how to use the features of CH7025/26.

Appendix B provides an example of how to programming CH7025/26.

Appendix C introduces how to use internal test pattern for debugging.

Actually, it is so easy to programming CH7025/26 using the programming tool. You could just refer to Chapter 2 and Appendix B to finish the main work of programming CH7025/26 quickly, which will make image showed on TV or VGA display.

This document could be used together with CH7025/26 datasheet Ver1.0.

Chapter 1:

DAC CONNECTION DETECTION

CH7025/26 can detect the connection of TV or VGA display. **A crystal is required** when you want to confirm the connection between CH7025/26 and TV or VGA display using this feature. If you not use this function of CH7025/26, skip this chapter.

Address: 7Dh

BIT:	7	6	5	4	3	2	1	0
SYMBOL:	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	SPPSNS
TYPE:	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
DEFAULT	0	0	0	0	0	0	0	0

SPPSNS(bit 0) of register 7Dh is the DAC sense signal for connection detect.

Address: 7Fh

BIT:	7	6	5	4	3	2	1	0
SYMBOL:	Reserved	Reserved	DACAT2[1]	DACAT2[0]	DACAT1[1]	DACAT1[0]	DACAT0[1]	DACAT0[0]
TYPE:	R	R	R	R	R	R	R	R
DEFAULT:	0	0	0	0	0	0	0	0

Register 7Fh is read-only register:

DACAT2[1:0] (bits 3-2) return attach information for DAC2 channel according to the table2-1 below.
DACAT1[1:0] (bits 3-2) return attach information for DAC1 channel according to the table2-1 below.
DACAT0[1:0] (bits 1-0) return attach information for DAC0 channel according to the table2-1 below.

Table2-1: Attached Display Mapping

DACATn[1:0]	Attached Display
00	No Attached Display
01	Connected
11	Short to ground
10	Reserved

Following is the usage and sample code of connection detection:

Steps:

1) Power up CH7025/26.
2) Set bit 3, bit 4 and bit 5 of Register 04h to “1”.
3) Set SPPSNS to ‘1’.
4) Delay some time (>=100ms).
5) Read the value of register 7Fh.
6) Set SPPSNS to ‘0’.
7) Set bit 3, bit 4 and bit 5of Register 04h to “0”.
8) Power down CH7025/26.

According the return value of register 7Fh, you could see which DAC is connected to TV or VGA display based on Table 2-1.

Sample code:

Refer to Appendix B.

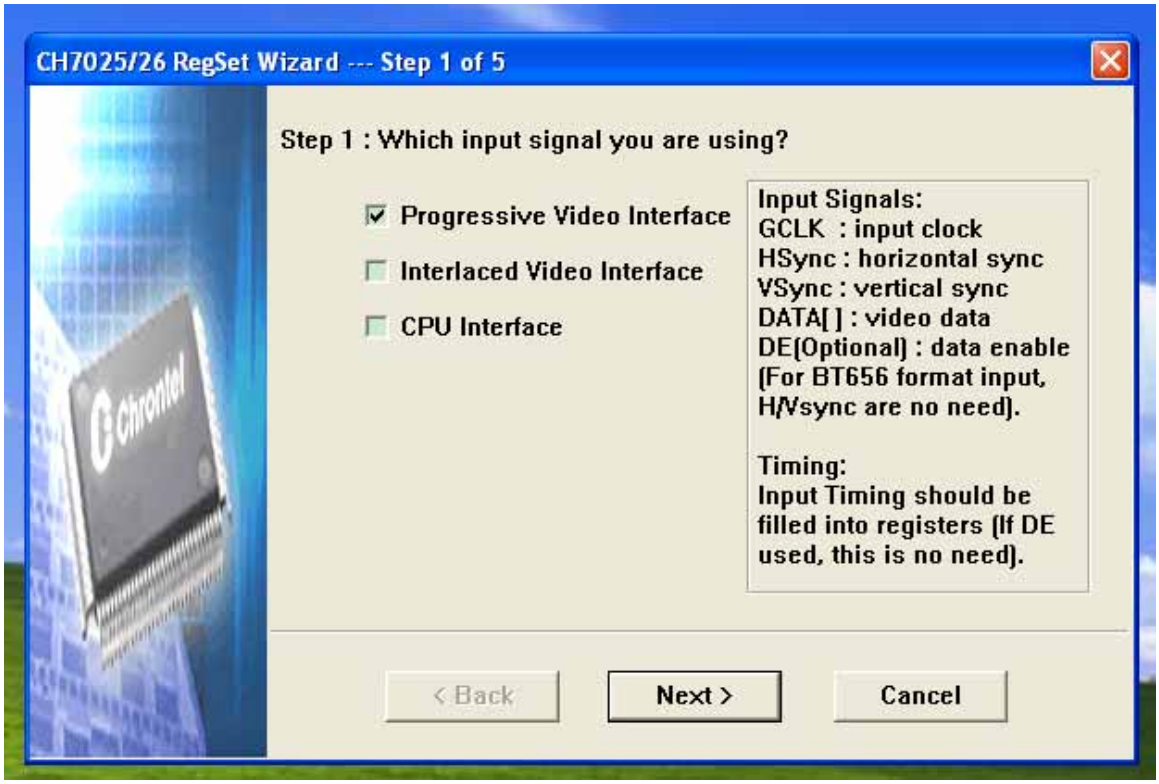
Chapter 2:
USING PROGRAMMING TOOL

The Register programming tool (CH7025(26) RegSet.exe) is created to help software engineer program workable register map of CH7025/26. The following sections explain how to use Register programming tool step by step.

2.1 Complete Wizard

CH7025/26 RegSet.exe provides wizard of 5 steps for you to complete the basic setting information when it starts to run.

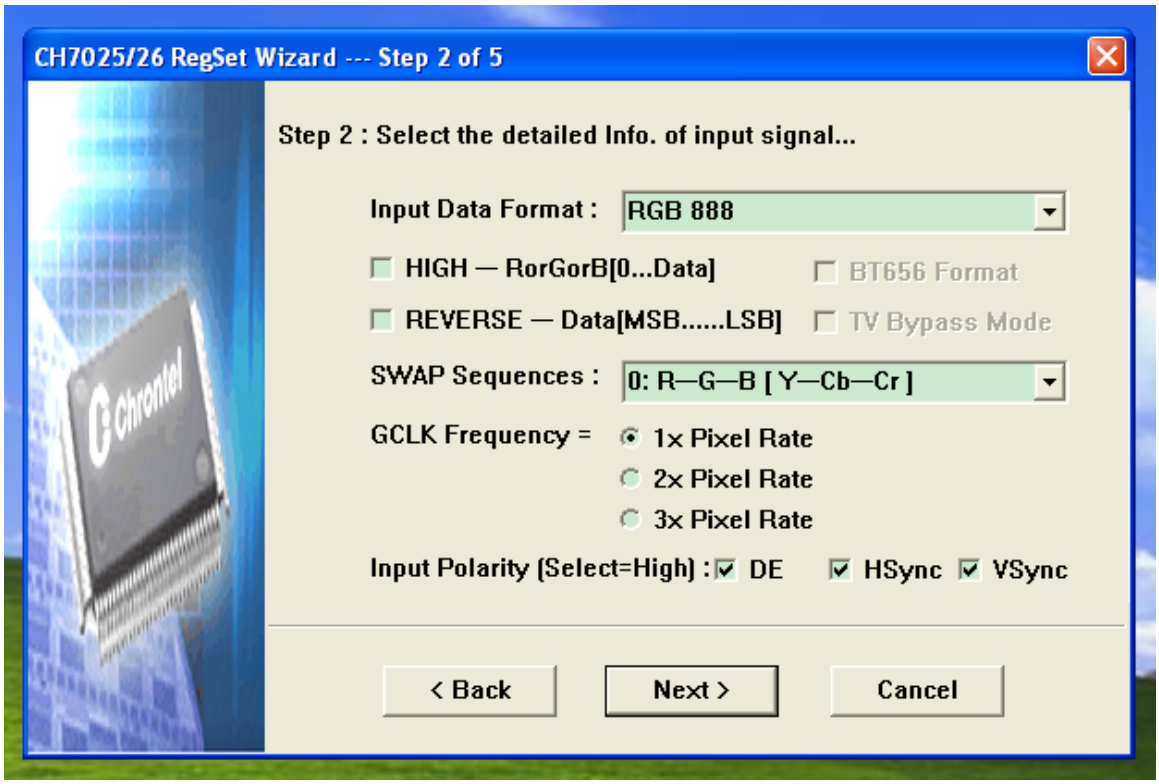
2.1.1 Wizard step 1:



In step1, you should decide the input signal format, in the right of the dialog shows the detailed information of these three input formats.

Click “Next” to complete this step.

2.1.2 Wizard step 2:



In step 2, you should decide the detailed information of input signal.

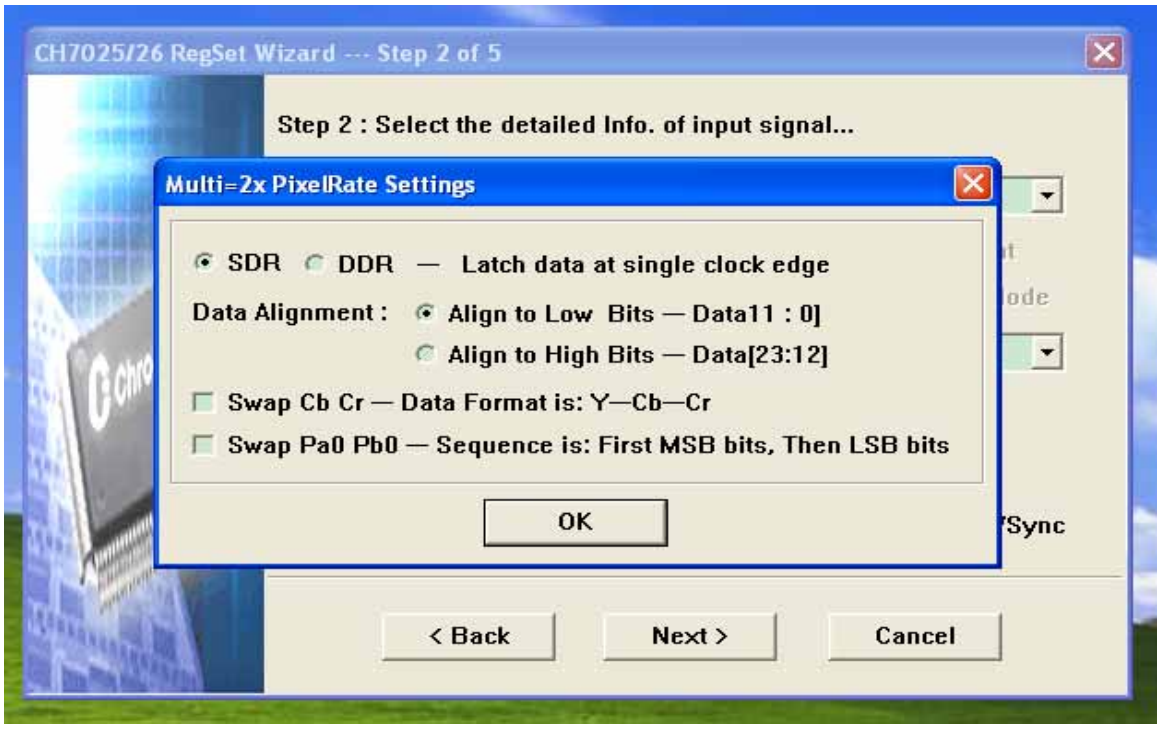
Please pay attention to the check button “BT656 Format”, when the input data format is YCbCr4:2:2, this check button will be enabled, if input data format (progressive or interlaced) follow the BT656 standard, make it checked.

If TV bypass mode is used (only when input is interlaced), make “TV_BP” checked.

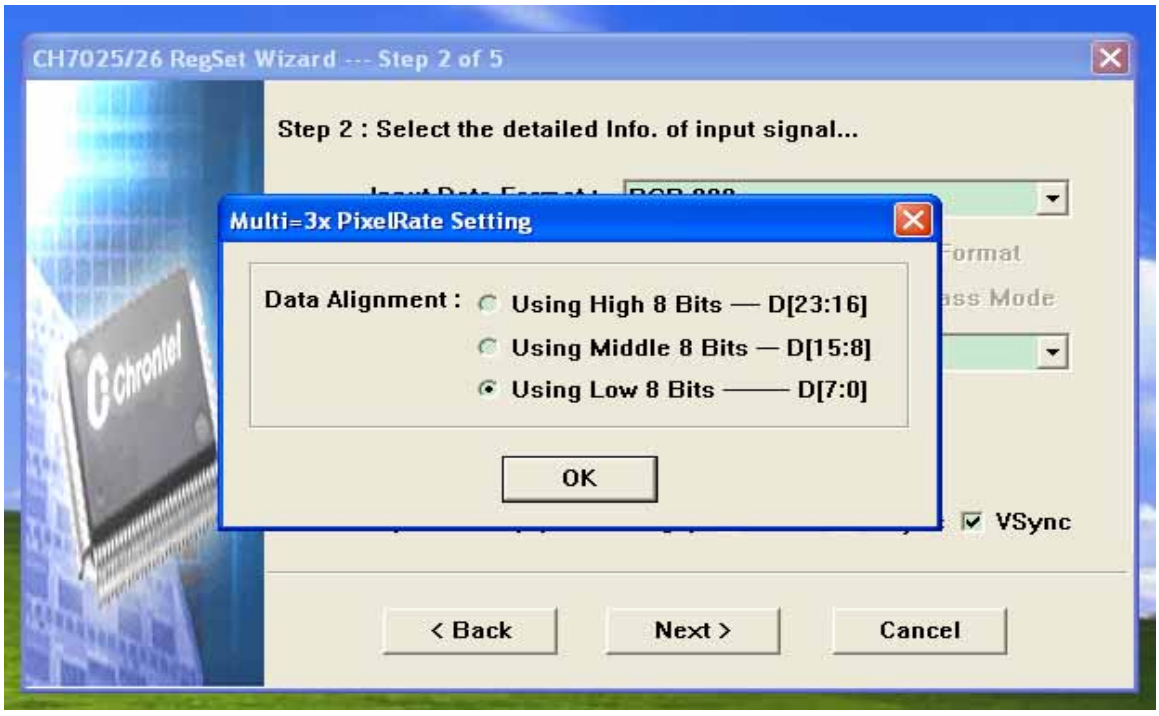
“GCLK Frequency” is used to select whether the input data is multiplexed.

When you select “2x Pixel Rate” check button, a dialog will pop up to let you to select the detailed information under 2x pixel rate input, as following, choose the correct items, and click “OK”.

For SDR and DDR, please refer to hint1.



When you select “3x Pixel Rate” check button, a dialog will pop up to let you select the detailed information under 3x pixel rate input, as following, choose the correct items, and click “OK”.



Click “Next” to complete this step.

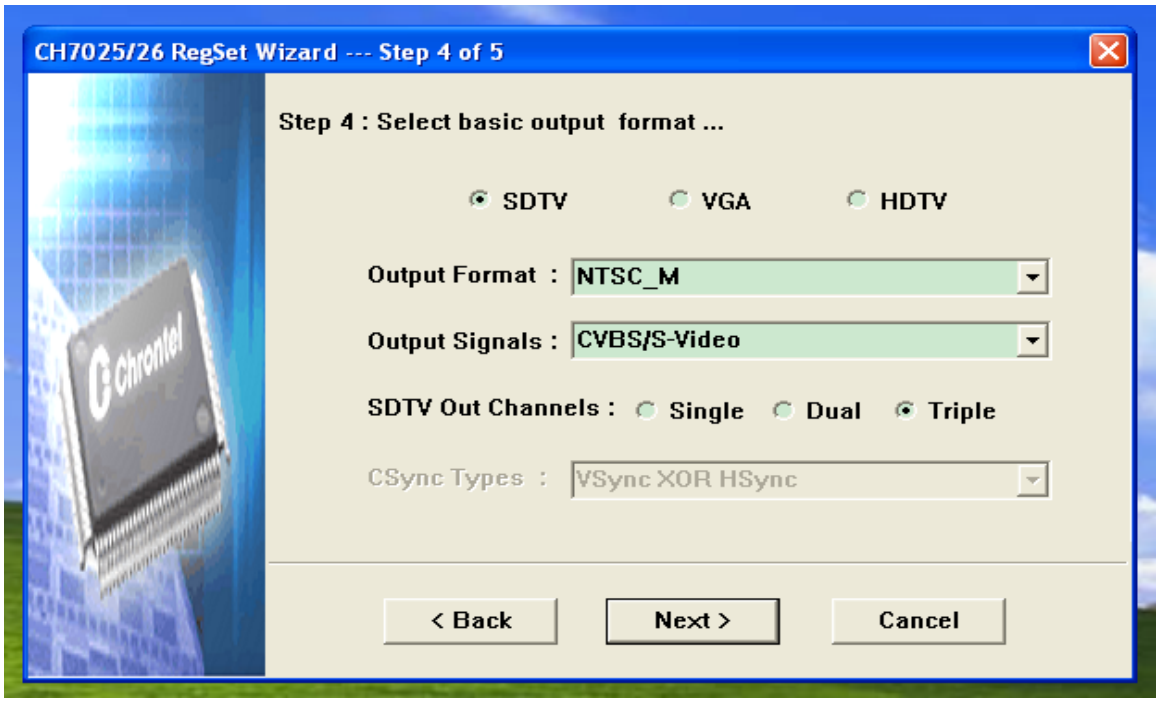
2.1.3 Wizard step 3:

In step 3, you should fill the input timing, such as HTI, HAI, VTI, VAI and so on (hint2). When the edit box is disabled, it needs not to be filled.

Please pay attention to the check button “HVAUTO”. If you make this button checked, the input timings need not to be filled, which means CH7025/26 will count them according Hsync, Vsync and DE(data enable) signal. So, when the “HVAUTO” check button is checked, DE signal is necessary, also, you should fill the input frame rate.

Click “Next” to complete this step.

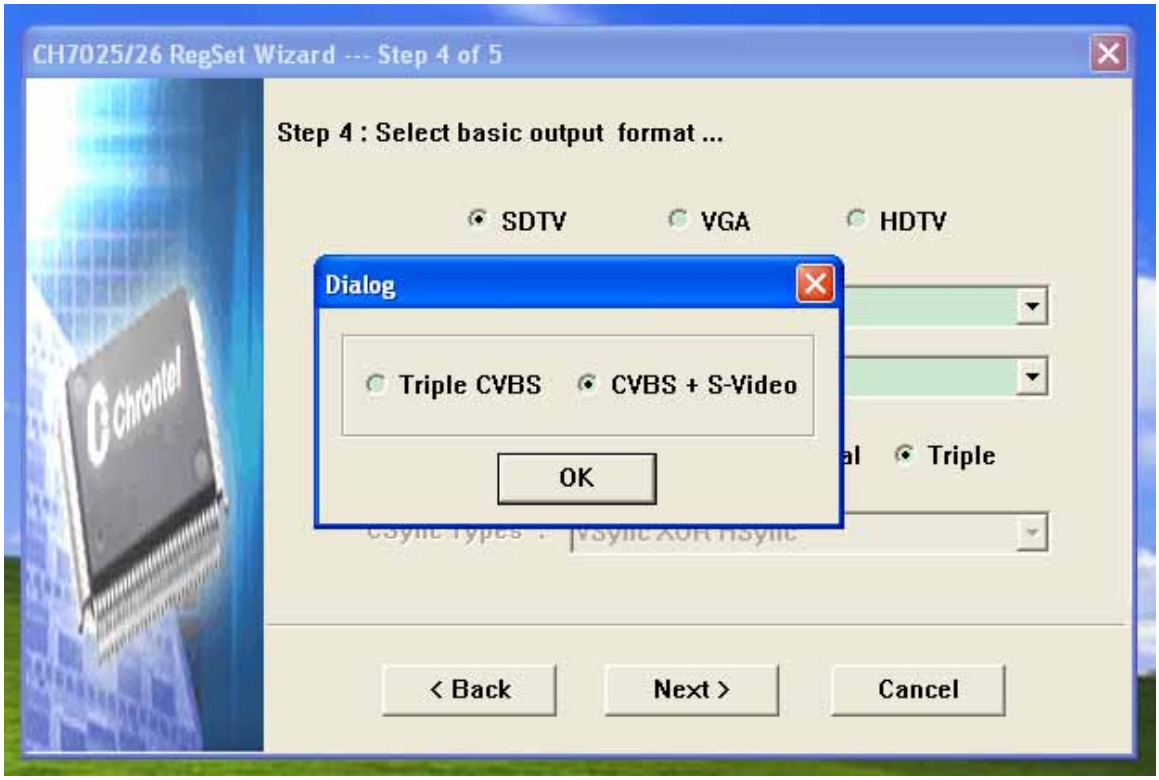
2.1.4 Wizard step 4:



In step 4, you should select the basic output format information.

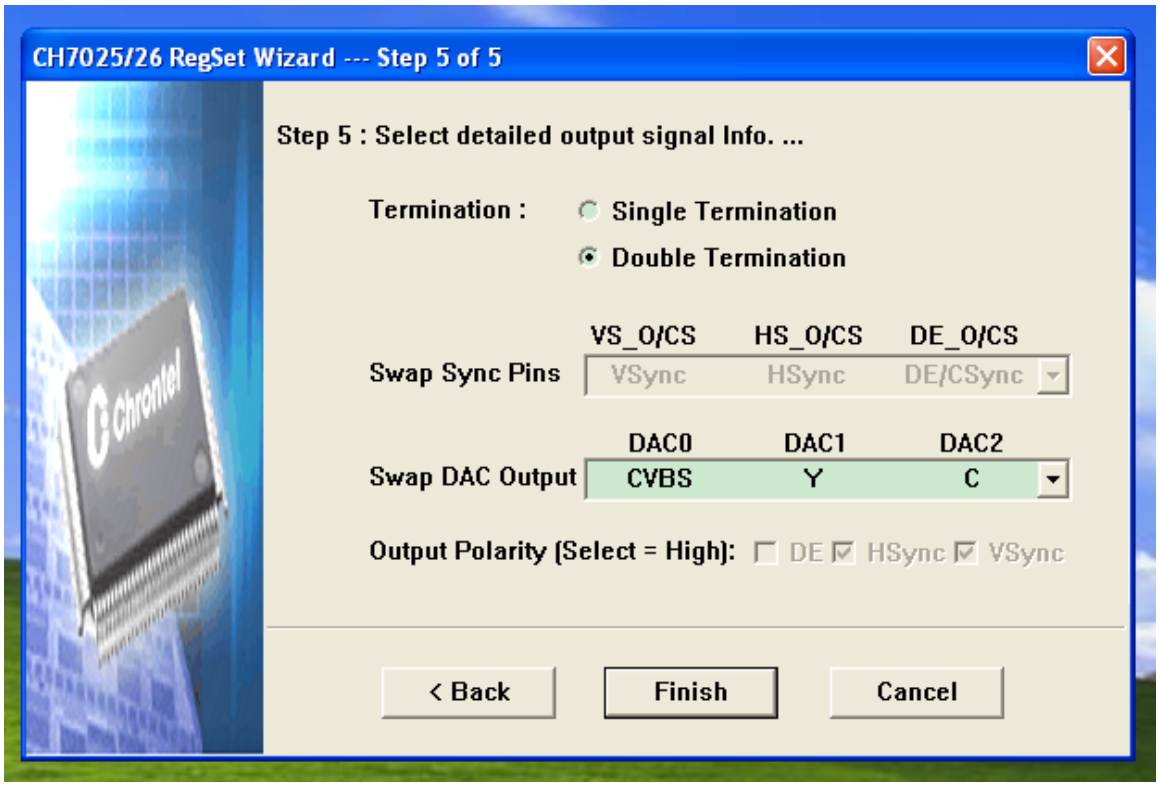
When select “Dual” radio button, a dialog will pop up to ask you the detailed information as following, choose the correct item, and clock “OK”.

When select “Triple” radio button, a dialog will pop up to ask you the detailed information as following, choose the correct item, and clock “OK”.



Click “Next” to complete this step.

2.1.5 Wizard step 5:

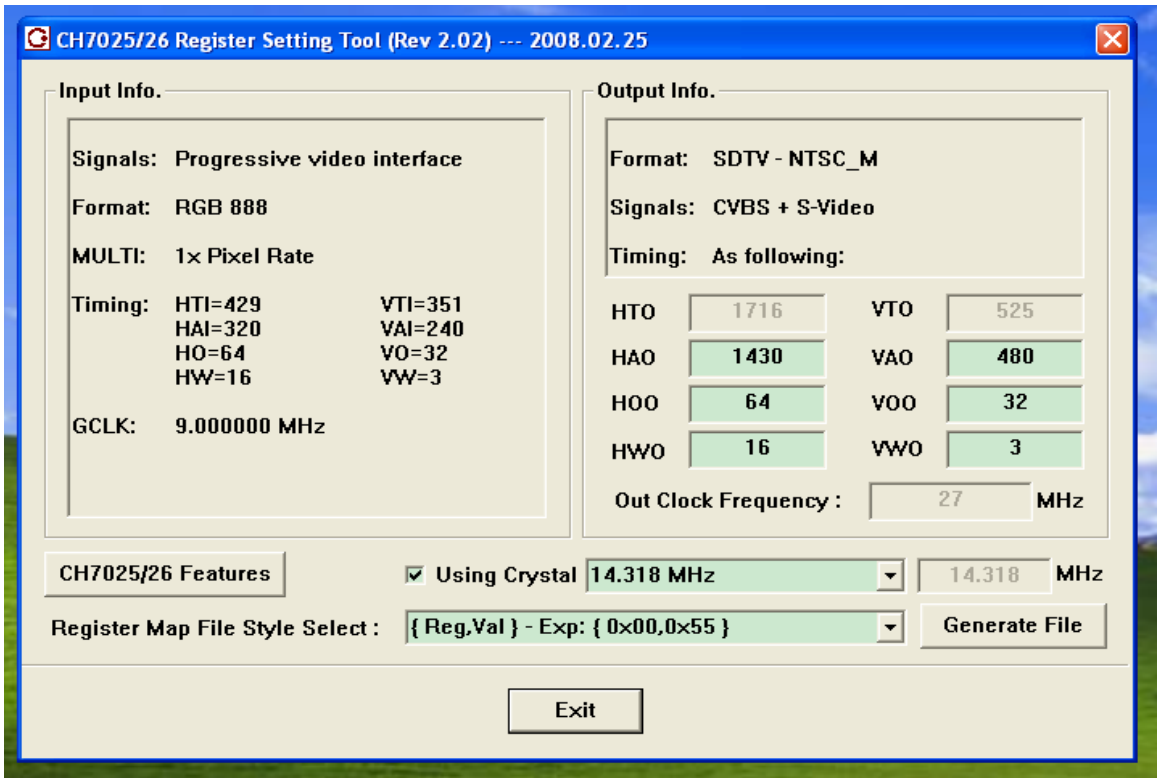


In step 5, you should decide the detailed output signal information.

Choose the correct items, and click “Finish”. Here, you have finish the wizard, basic information have been recorded.

2.2 Complete last operation

After click “Finish” button in last section, the last operation dialog will pop up, as following:



This dialog will show the main information you decide in the wizard:

“Input Info.” shows basic input information.

“Output Info.” shows basic output information. Please pay attention that when TV output format was selected, you could modify the “HAO”, “VAO”, “HOO”, “VOO”, “HWO” and “VWO” output timings, but “HTI”, “VTI” and the output clock frequency are fixed as show you; when VGA output format was selected, you could modify all of these parameter as you want.

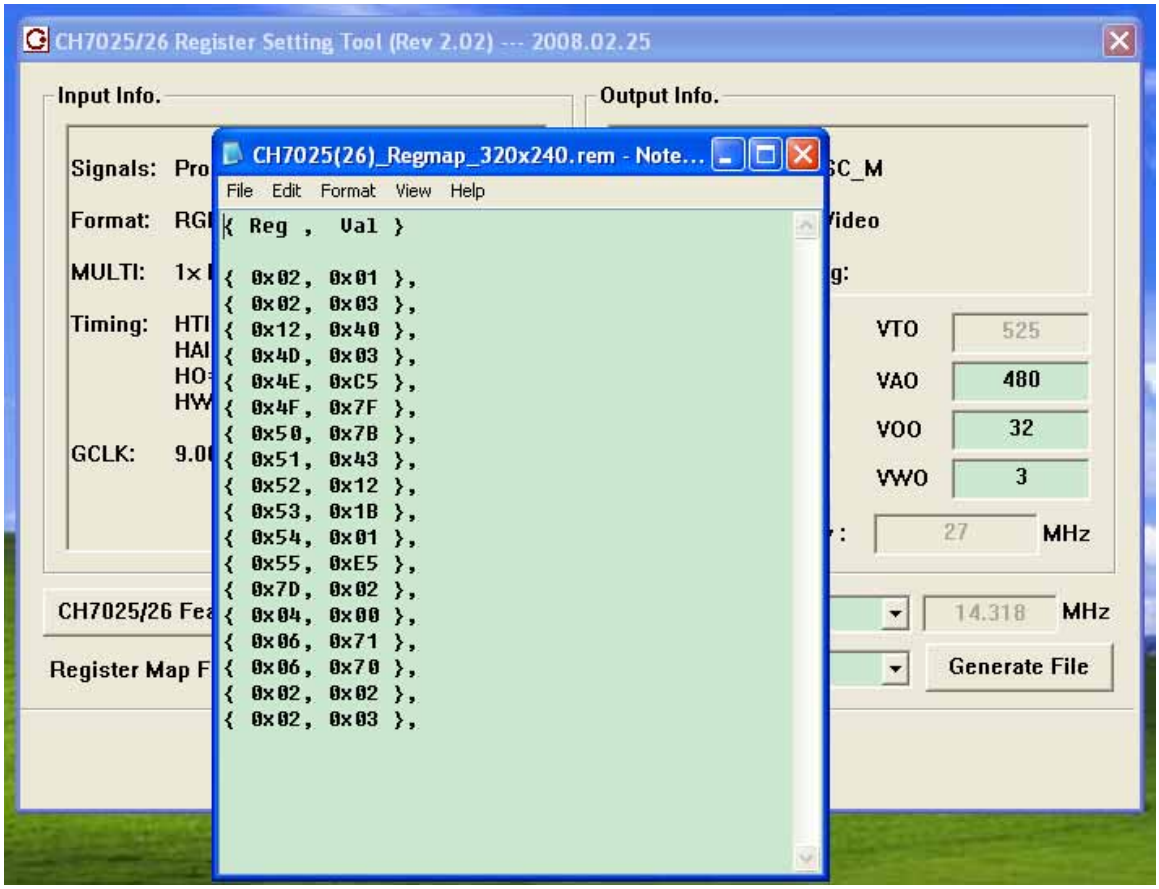
“CH7025/26 Features” let you can modify some features of CH7025/26.

“Using Crystal” check button let you can choose if a crystal will be used with CH7025/26 when it enabled (For example, when CPU interface was selected in the wizard, this check button will be disable, which said that a crystal must be used).

The combo box and edit box right to the “Using Crystal” check button are used to decide the frequency of the crystal.

2.3 Generate register setting file

Now to generate the register setting file based on the selection you just made. First choose the file format from the "Register Map File Styles" combo box, then click "Generate Register Setting file", one ".txt" file with correct register values will be created and opened.



Here, you have finished main setting of programming CH7025/26. Actually, after setting the registers of CH7025/26 as the file show you **(NOTE)**, you could see image on TV display. Please refer to Appendix B for example.

In addition, you could adjust the quality and properties of the image on TV VGA display, such as position on screen, brightness, zoom function, rotation, flipping and so on. Please refer to Appendix A for details.

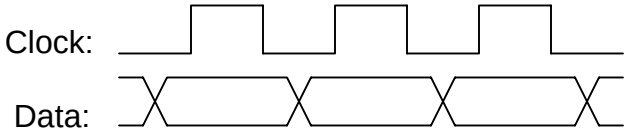
NOTE:

We recommend that to set registers of CH7025/26 following the sequence of registers in the register setting file.

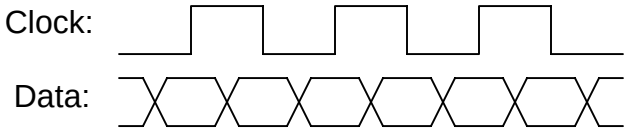
Hint1:

Definition of SDR and DDR:

SDR:

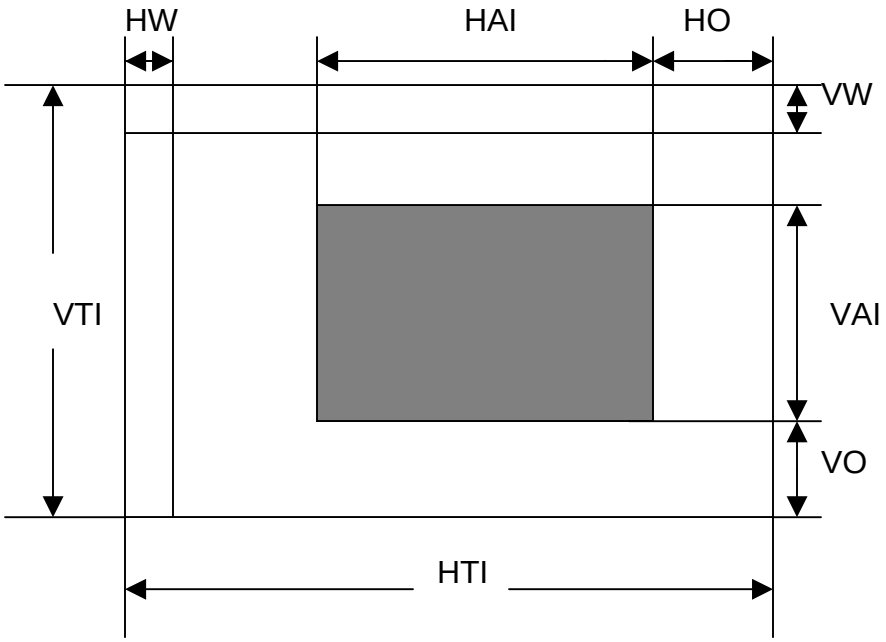


DDR:



Hint2:

Chrontel input timing definition:



Appendix A:

USING CH7025/26 FEATURES

CH7025/26 provides interfaces to adjust the quality and properties of the image on TV or VGA display. The features includes:

1) Adjust hue.
2) Adjust saturation.
3) Adjust contrast.
4) Adjust brightness.
5) Adjust sharpness (text enhancement).
6) Adjust display position on the screen.
7) Using flip function.
8) Using rotation function.
9) Using zoom function.

Following introduce how to use these features of CH7025/26 respectively.

1) Adjust hue:

Address 2Eh

BIT:	7	6	5	4	3	2	1	0
SYMBOL:	Reserved	HUE[6]	HUE[5]	HUE[4]	HUE[3]	HUE[2]	HUE[1]	HUE[0]
TYPE:	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
DEFAULT:	0	1	0	0	0	0	0	0

HUE[6:0] (bits 6 - bit 0) define the TV Hue control HUE[6:0]. The adjusted angle in the color space is (HUE[6:0]-64)/2 degrees, positive angle is toward magenta color, negative angle is toward green color.

Usage:

Usually, the default hue value (64) could be reasonable, but you can write a new hue value to the register 2Eh to adjust hue of the image, the value should be in the range 0 ~ 127.

Sample function:

```
// Function:
        Adjust image hue based on the pre-value of hue.
// Name:
        AdjustHue.
// Param:
         dif: difference with current value
void AdjustHue ( int  dif )
{
        int new_val = I2CRead ( 0x2E ) + dif;
        if ( new_val > 127 )
                new_val = 127;
        if ( new_val < 0 )
                new_val = 0;
        I2CWrite ( 0x2E, new_val );
}

// Function:
        Set a new hue value.
// Name:
        SetHue
// Param:
        hue: the new value of hue.
void SetHue ( unsigned char hue)
{
        if ( hue > 127 )
                hue = 127;
        I2CWrite ( 0x2E, hue );
}
```

2) Adjust saturation:

Address: 2Fh

BIT:	7	6	5	4	3	2	1	0
SYMBOL:	Reserved	SAT[6]	SAT[5]	SAT[4]	SAT[3]	SAT[2]	SAT[1]	SAT[0]
TYPE:	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
DEFAULT:	0	1	0	0	0	0	0	0

SAT[6:0](bits 6-0) define the TV Color Saturation control SAT[6:0].The Color Saturation is multiplied by SAT[6:0]/64.

Usage:

Usually, the default hue value (64) could be reasonable, but you can write a new saturation value to the register 2Fh to adjust hue of the image, the value should be in the range 0 ~ 127.

Sample function:

```
// Function:
        Adjust image saturation based on the pre-value of hue.
// Name:
        AdjustSat.
// Param:
         dif: difference with current value
void AdjustSat ( int  dif )
{
        int new_val = I2CRead ( 0x2F ) + dif;
        if ( new_val > 127 )
                new_val = 127;
        if ( new_val < 0 )
                new_val = 0;
        I2CWrite ( 0x2F, new_val );
}

// Function:
        Set a new saturation value.
// Name:
        SetSat.
// Param:
        hue: the new value of saturation.
void SetSat ( unsigned char sat)
{
        if ( sat > 127 )
                sat = 127;
        I2CWrite ( 0x2F, sat );
}
```

3) Adjust contrast:

Address 30h

BIT:	7	6	5	4	3	2	1	0
SYMBOL:	Reserved	CTA[6]	CTA[5]	CTA[4]	CTA[3]	CTA[2]	CTA[1]	CTA[0]
TYPE:	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
DEFAULT:	0	1	0	0	0	0	0	0

CTA[6:0] (bits 6-0) define the TV contrast control CTA[6:0]. The luma is multiplied by CTA[6:0]/64.

Usage:

Usually, the default contrast value (64) could be reasonable, but you can write a new contrast value to the register 30h to adjust contrast of the image, the value should be in the range 0 ~ 127.

Sample function:

```
// Function:
        Adjust image contrast based on the pre-value of hue.
// Name:
        AdjustContrast.
// Param:
         dif: difference with current value
void AdjustConTrast ( int  dif )
{
        int new_val = I2CRead ( 0x30 ) + dif;
        if ( new_val > 127 )
                new_val = 127;
        if ( new_val < 0 )
                new_val = 0;
        I2CWrite ( 0x30, new_val );
}

// Function:
        Set a new contrast value.
// Name:
        SetContrast
// Param:
        contrast: the new value of contrast.
void SetContrast ( unsigned char contrast)
{
        if ( contrast > 127 )
                contrast = 127;
        I2CWrite ( 0x30, contrast );
}
```

4) Adjust brightness:

Address 31h

BIT:	7	6	5	4	3	2	1	0
SYMBOL:	BRI[7]	BRI[6]	BRI[5]	BRI[4]	BRI[3]	BRI[2]	BRI[1]	BRI[0]
TYPE:	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
DEFAULT:	1	0	0	0	0	0	0	0

BRI[7:0] (bits 7-0) define the TV brightness control BRI[7:0]. The Brightness will be adjusted by (BRI[7:0]-128).

Usage:

Usually, the default brightness value (128) could be reasonable, but you can write a new brightness value to the register 31h to adjust brightness of the image, the value should be in the range 0 ~ 255.

Sample function:

```
// Function:
        Adjust image brightness based on the pre-value of hue.
// Name:
        AdjustBrightness.
// Param:
         dif: difference with current value
void AdjustBrightness ( int  dif )
{
        int new_val = I2CRead ( 0x31 ) + dif;
        if ( new_val > 255 )
                new_val = 255;
        if ( new_val < 0 )
                new_val = 0;
        I2CWrite ( 0x31, new_val );
}

// Function:
        Set a new brightness value.
// Name:
        SetBrightness.
// Param:
        brightness: the new value of brightness.
void SetBrightness ( unsigned char brightness)
{
        if ( brightness > 255 )
                brightness = 255;
        I2CWrite ( 0x30, brightness );
}
```

5) Adjust sharpness:

Address 32h

BIT:	7	6	5	4	3	2	1	0
SYMBOL:						TE[2]	TE[1]	TE[0]
TYPE:	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

DEFAULT:	0	0	0	0	0	1	0	0

TE[2:0] (bits 2-0) define TV Sharpness control (Text Enhancement) TE[2:0].

Usage:

Usually, default sharpness value (4) could be reasonable, TE=4 means no enhancement. Larger setting than 4 boost the high frequency band of the picture; Smaller setting than 4 smoothes the image. You can write a new TE value to the register 32h[2:0] to adjust sharpness of the image, the value should be in the range 0 ~ 7.

Sample function:

```
// Function:
        Adjust image sharpness based on the pre-value of hue.
// Name:
        AdjustTE.
// Param:
         dif: difference with current value
void AdjustTE ( int  dif )
{
        int new_val = ( I2CRead ( 0x32 ) & 0x07 ) + dif;
        if ( new_val > 7 )
                new_val = 7;
        if ( new_val < 0 )
                new_val = 0;
        new_val |= ( I2CRead ( 0x32 ) & 0xF8 );
        I2CWrite ( 0x32, new_val );
}

// Function:
        Set a new sharpness value.
// Name:
        SetTE.
// Param:
        te: the new value of sharpness.
void SetTE ( unsigned char te)
{
        unsigned char new_val;
        if ( te > 7 )
                te = 7;
        new_val = ( I2CRead ( 0x32 ) & 0xF8 ) | te;
        I2CWrite ( 0x30, new_val );
}
```

6) Adjust display position on the screen:

Address 33h

BIT:	7	6	5	4	3	2	1	0
SYMBOL:	Reserved	Reserved	Reserved	Reserved	VP[11]	VP[10]	VP[9]	VP[8]
TYPE:	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
DEFAULT:	0	0	0	0	1	0	0	0

Address 34h

BIT:	7	6	5	4	3	2	1	0
SYMBOL:	VP[7]	VP[6]	VP[5]	VP[4]	VP[3]	VP[2]	VP[1]	VP[0]
TYPE:	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
DEFAULT:	0	0	0	0	0	0	0	0

VP[7:0] combine with VP[11:8] to define the TV vertical position adjustment VP[11:0]. The number of lines that is adjusted is determined by VP[11:0]-2048. If the value is positive, the picture is moved upward; if the value is negative, the picture is moved downward.

Address 35h

BIT:	7	6	5	4	3	2	1	0
SYMBOL:	Reserved	Reserved	Reserved	Reserved	HP[11]	HP[10]	HP[9]	HP[8]
TYPE:	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
DEFAULT:	0	0	0	0	1	0	0	0

Address: 36h

BIT:	7	6	5	4	3	2	1	0
SYMBOL:	HP[7]	HP[6]	HP[5]	HP[4]	HP[3]	HP[2]	HP[1]	HP[0]
TYPE:	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
DEFAULT:	0	0	0	0	0	0	0	0

HP[7:0] (bits 7-0) combine with HP[11:8] to define TV horizontal position adjustment HP[11:0]. The number of pixels that is adjusted is determined by HP[11:0]-2048. If the value is positive, the picture is moved to the right; if the value is negative, the picture is moved to the left.

Usage:

Usually, the default horizontal and vertical values are 2048, which mean no position adjust. You can write a new brightness value to the register above to

adjust the position of the image on the screen, the value should be in the range 0 ~ 4095.

Sample function:

```
// Function:
        Adjust the position of the image on the screen.
// Name:
        AdjustPos ( int h_dif, int v_dif );
// Param:
        h_dif: horizontal difference( positive: move right; negative: move left)
        v_dif: vertical difference( positive: move up; negative: move down)

void AdjustPos( int h_dif, int v_dif )
{
        int hp = ( I2CRead ( 0x35 ) << 8 ) | I2CRead ( 0x36 );
        int vp = ( I2CRead ( 0x33 ) << 8 ) | I2CRead ( 0x34 );
        int new_hp = hp + dif;
        int new_vp = vp + dif;

        if ( new_hp > 4095 )
                new_hp = 4095;
        if ( new_hp < 0 )
                new_hp = 0;
        if ( new_vp > 4095 )
                new_vp = 4095;
        if ( new_vp < 0 )
                new_vp = 0;

        // We recommend that when set postion( HP or VP): first set high
        // bits and second set low bits, just as following:
        I2CWrite ( 0x35, (new_hp >> 8 ) & 0x0F ); // Write high bits of hp
        I2CWrite ( 0x36, (new_hp & 0xFF ) ); // Write low bits of hp

        I2CWrite ( 0x33, (new_vp >> 8 ) & 0x0F ); // Write high bits of vp
        I2CWrite ( 0x34, (new_vp & 0xFF ) ); // Write low bits of vp

}
```

7) Using flip function:

Address: 2Dh

BIT:	7	6	5	4	3	2	1	0
SYMBOL:					VFLIP	HFLIP		

TYPE:	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
DEFAULT:	0	0	0	0	0	0	0	0

VFIP (bit 3) is the vertical flip bit. When this bit set to “1”, the image will flip in vertical direction.

HFIP (bit 2) is the horizontal flip bit. When this bit set to “1”, the image will flip in horizontal direction.

Usage:

When using flip function, you just need to set VFLIP or HFLIP to ‘1’.

Sample function:

```
// Function:
        Make or Unmake image flip in vertical direction.
// Name:
        void VerticalFlip ( BOOL IsFlip ).
// Param:
        IsFlip: When using flip function, IsFlip is TRUE, otherwise FALSE.

void VerticalFlip ( BOOL IsFlip )
{
        unsigned char val = I2CRead ( 0x2D );
        if ( IsFlip )
                val |= 0x08;
        else
                val &= 0xF7;
        I2CWrite ( 0x2D, val );
}

// Function:
        Make or Unmake image flip in horizontal direction.
// Name:
        void HorizontalFlip ( BOOL IsFlip ).
// Param:
        IsFlip: When using flip function, IsFlip is TRUE, otherwise FALSE.

void HorizontalFlip ( BOOL IsFlip )
{
        unsigned char val = I2CRead ( 0x2D );
        if ( IsFlip )
                val |= 0x04;
        else
                val &= 0xFC;
```

```
        I2CWrite ( 0x2D, val );
    }
```

8) Using rotation function:

Address: 2Dh

BIT:	7	6	5	4	3	2	1	0
SYMBOL:							ROTATE[1]	ROTATE[0]
TYPE:	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
DEFAULT:	0	0	0	0	0	0	0	0

ROTATE[1:0] (bit 1 bit 0) controls image rotation.

ROTATE[1:0]=”00”: no rotation
ROTATE[1:0]=”01”: 90 degree rotation
ROTATE[1:0]=”10”: 180 degree rotation
ROTATE[1:0]=”11”: 270 degree rotation

Usage:

When using rotation function, you just need to set RATATE[1:0] to 0 (No rotation), 1 (90 degree rotation), 2 (180 degree rotation) or 3 (270 degree rotation) .

Sample function:

```
    // Function:
        Make image on the screen rotate.
    // Name:
        void Rotate ( unsigned char dgr ).
    // Param:
        dgr:    dgr = 0 means no rotate.
                dgr = 1 means 90 degree rotate.
                dgr = 2 means 180 degree rotate.
                dgr = 3 means 270 degree rotate.

    Void Rotate ( unsigned char dgr )
    {
        unsigned char val = I2CRead ( 0x2D );
        if ( dgr > 3 )
            dgr = 0; // if dgr get out of the range, make it to 0.
        val &= 0xFC;
        val |= dgr;
        I2CWrite ( 0x2D, val );
    }
```

9) Using zoom function:

Address: 27h

BIT:	7	6	5	4	3	2	1	0
SYMBOL:	IMGZOOM	Reserved	HEND[10]	HEND[9]	HST[8]	HST[10]	HST[9]	HST[8]
TYPE:	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
DEFAULT:	0	0	1	0	1	0	0	0

IMGZOOM (bit 7) is to enable the image zoom feature. When this bit set is to “1”, the zoom feature is enabled.

HEND[10:8] (bit 5 – bit 3) combine with HEND[7:0] of Register 29h to define HEND[10:0], the Horizontal End position.

HST[10:8] (bit 2 – 0) combine with HST[7:0] of Register 28h to define HST[10:0], the Horizontal Start position.

Address: 28h

BIT:	7	6	5	4	3	2	1	0
SYMBOL:	HST[7]	HST[6]	HST[5]	HST[4]	HST[3]	HST[2]	HST[1]	HST[07]
TYPE:	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
DEFAULT:	0	0	0	0	0	0	0	1

Address: 29h

BIT:	7	6	5	4	3	2	1	0
SYMBOL:	HEND[7]	HEND[6]	HEND[5]	HEND[4]	HEND[3]	HEND[2]	HEND[1]	VHEND[0]
TYPE:	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
DEFAULT:	1	0	0	1	0	1	1	0

Address: 2Ah

BIT:	7	6	5	4	3	2	1	0
SYMBOL:	Reserved	Reserved	VEND[10]	VEND[9]	VEND[8]	VST[10]	VST[9]	VST[8]
TYPE:	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
DEFAULT:	0	0	0	0	1	0	0	0

VEND[10:8] (bit 5 – bit 3) combine with VEND[7:0] of Register 2Ch to define VEND[10:0], the Vertical End position.

VST[10:8] (bit 2 – bit 0) combine with VST[7:0] of Register 2Bh to define VST[10:0], the Vertical Start position.

Address: 2Bh

BIT:	7	6	5	4	3	2	1	0
SYMBOL:	VST[7]	VST[6]	VST[5]	VST[4]	VST[3]	VST[2]	VST[1]	VST[0]
TYPE:	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
DEFAULT:	0	0	0	0	0	0	0	1

Address: 2Ch

BIT:	7	6	5	4	3	2	1	0
SYMBOL:	VEND[7]	VEND[6]	VEND[5]	VEND[4]	VEND[3]	VEND[2]	VEND[1]	VEND[0]
TYPE:	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
DEFAULT:	1	1	1	0	0	0	0	0

Usage:

Zoom function is based on the input timing. For example, if you want to make up-left quarter of the image zoomed, you should calculate the HST, HEND, VST and VEND from: HTI, HAI, HO, HW, VTI, VAI, VO, VW (NOT OUTPUT TIMING)!

HST, HEND, VST and VEND should be in the range following:

HST and HEND: 1 ~ HAI
VST and VEND: 1 ~ VAI

The sequence of using zoom function:

1) Make CH7025/26 stop running.
2) Write HST, HEND, VST and VEND with your desired values.
3) Set IMGZOOM bit to '1'.
4) Make CH7025/26 running.

Sample function:

```
// define two structures here:

// record timing information:
typedef struct {
        unsigned int ht,      // horizontal total pixels
        unsigned int ha,      // horizontal active pixels
        unsigned int ho,      // horizontal offset pixels
        unsigned int hw,      // horizontal sync width in pixels
        unsigned int vt,      // vertical total pixels
        unsigned int va,      // vertical active pixels
        unsigned int vo,      // vertical offset pixels
```

```
        unsigned int vw,      // vertical sync width in pixels
} TIMING, *PTIMING;

// typedef struct {
        // horizontal start point:
        unsigned int hStart, // (1 ~ HAI)
        // hortizontal end point:
        unsigned int hEnd,   // (1 ~ HAI)
        // vertical start point:
        unsigned int vStart, // (1 ~ VAI)
        // vertical end point:
        unsigned int vSize,  // (1 ~ VAI)
} ZOOMSIZE, *PZOOMSIZE;


// Function:
        Zoom up the image on the screen.
// Name:
        void ZoomEnable (PZOOMSIZE pzs, PTIMING ptm, BOOL IsZoom)
// Param:
        pzs: ZOOMSIZE pointer which describes the zoom information.
        Ptm: TIMING pointer which describes input timing information.
        IsZoom: indicate to enable zoom function or not.

Void ZoomEnable ( PZOOMSIZE pzs, PTIMING ptm, BOOL IsZoom )
{
        unsigned char val = 0;
        double temp = 0.0;
        unsigned int hst = 0, hend = 0, vst = 0, vend = 0;

        //Stop running:
        val = I2CRead ( 0x06 );
        val |= 0x01;
        I2CWrite ( 0x06, val );

        if ( !IsZoom ) // disable zoom function
        {
                val = I2CRead ( 0x27 );
                val &= 0x7F;
                I2CWrite ( 0x27, val );
        }
        else // enable zoom function
        {
                // Just for simplify code:
                hst = pzs->hStart ;
                hend = pzs->hEnd;
```

```
        if (hend > ptm -> ha )
                hend = ptm -> ha;
        vst = pzs->vStart;
        vend = pzs->   end;
        if ( vend > ptm -> va )
                vend = ptm -> va;

        //Write values to registers:
        val = I2CRead ( 0x27 );
        val = (val & 0xC0) | ((hend>>5 ) & 0x38) | (hst >> 8) & 0x07);
        I2CWrite ( 0x27, val );
        I2CWrite ( 0x28, ( hst & 0xFF ) );
        I2CWrite ( 0x29, ( hend & 0xFF ) );

        val = I2CRead ( 0x2A );
        val = (val & 0xC0) | ((vend >>5) &0x38) | ((vst >> 8) & 0x07);
        I2CWrite ( 0x2A, val );
        I2CWrite ( 0x2B, ( vst & 0xFF ) );
        I2CWrite ( 0x2C, ( vend & 0xFF ) );

        //IMGZOOM to '1'
        val = I2CRead ( 0x27 );
        val |= 0x80;
        I2CWrite ( 0x27, val );
}

// Make CH7025/26 running:
val = I2CRead ( 0x06 );
val &= 0xFE;
I2CWrite ( 0x06, val );
```

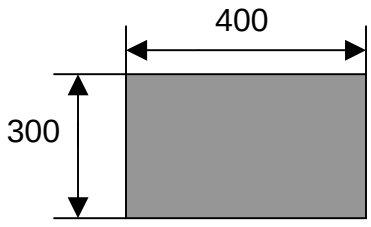
Sample code:

Assume that input timing is initialized as following:

```
                //      HT    HA    HO    HW    VT    VA    VO    VW
TIMING tm = {   500,    400,  50,   10,   400,  300,  50,   10 };
```

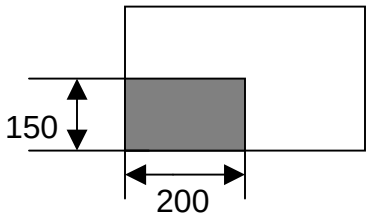
Example 1: zoom 100% image that looks like no zoom:

```
ZOOMSIZE sz;
sz.hStart = 1;
sz.hEnd  = 400;
sz.vStart = 1;
sz.vend  = 300;
ZoomEnable ( &sz, &tm, TRUE );
```



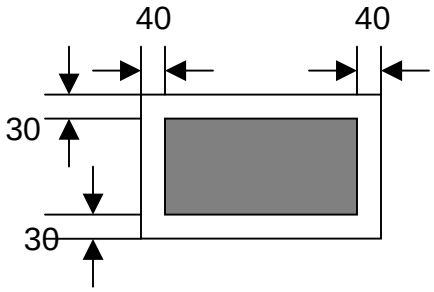
Example 2: zoom down-left quarter of the image:

```
ZOOMSIZE zs;
sz.hStart = 1;
sz.hEnd  = 200;
sz.vStart = 151;
sz.  end  = 300;
ZoomEnable ( &sz, &tm, TRUE );
```



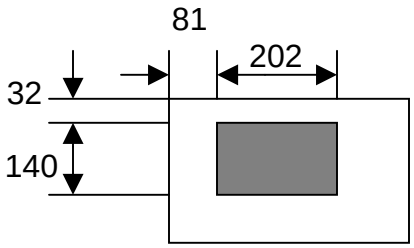
Example 3: zoom center:

```
ZOOMSIZE zs;
zs.hStart = 41;
zs.hEnd  = 360;
zs.vStart = 31
zs.  end  = 270;
ZoomEnable ( &sz, &tm, TRUE );
```



Example 4: generic zoom:

```
ZOOMSIZE zs;
zs.hStart = 82;
zs.hEnd  = 283;
zs.vStart = 33;
zs.  end   = 172;
ZoomEnable ( &zs, &tm, TRUE );
```



Example 5: Cancel zoom function:

```
ZOOMSIZE zs = {0}; // To cancel zoom, this not used in ZoomEnable()
ZoomEnable ( &zs, &tm, FALSE );
```

Appendix B:

SAMPLE CODE

Here we provide a whole example to show how to program CH7025/26.

Assume:

5) Using C code.

2) I2C functions have been defined in other place:
unsigned char I2CRead (unsigned char index);
void I2CWrite (unsigned char index, unsigned char value);

3) You have the register map file (See chapter 3), and you have copied the content to variable REG_MAP[ ][2]:

6) The LCD controller that sends video data to CH7025/26 has been set correctly, and the video data flowing into CH7025/26 is stable.

NOTE:

We recommended that first to configure the LCD controller who will send out video data to CH7025/26, and then configure CH7025/26 after the video data flowing into CH7025/26 become stable.

Sample code:

```
// BEGIN HERE:

typedef unsigned char     uchar;
typedef unsigned int      uint;

#define CH7025_DID        0x55
#define CH7026_DID        0x54

// define variable to store the register map:
unsigned char REG_MAP[ ][2] = {
        { 0x04, 0x31 }, // red section is copied from register setting file.
        { 0x0B, 0x04 },
        ...
        { 0x44, 0x22 },
        { 0x77, 0x11 },
        ...
};
#define REGMAP_LENGTH  ( sizeof( REG_MAP ) / (2 * sizeof ( uchar ) ) )
```

```
// I2C function, defined in other place:
extern uchar I2CRead ( uchar index );
extern void I2CWrite ( uchar index, uchar value);

// CH7025/26 related function declaration:
uint Dacs_CntDtt ( ); // connection detect

#define USING_CONNECT_DETECT


int main( )
{
        uchar ii = 0;
        uchar id = 0;

        // Make sure CH7025/26 in the system:
        id = I2CRead ( 0x00 );
        if ( (id != CH7025_DID ) && (id != CH7026_DID))
        {
                return 1; // CH7025/26 was not found
        }

        // CH7025/26 was found, go on

        // DAC connection detect, this is optional.

#ifdef  USING_CONNECT_DETECT

        if ( ! Dacs_CntDtt( ) )
        {
                return 2; // TV or VGA display not connected to CH7025/26.
        }

#endif

        // Write CH7025/26 RegMap to registers of CH7025/26:
        for ( ii = 0 ; ii < REGMAP_LENGTH ; ++ii )
        {
                I2CWrite ( REG_MAP[ii][0], REG_MAP[ii][1] );
        }


        // Now you can see image on TV or VGA display. According Appendix A to
        // modify the features of CH7025/26.

        Return 0; // success!
```

```
} // end of main

// function defination:

// Function: connection detect
// Param: none.
// Return value: bit0 == 1: DAC0 is connected to TV or VGA display.
                  Bit1 == 1: DAC1 is connected to TV or VGA display.
                  Bit2 == 1: DAC2 is connected to TV or VGA display.
                  Return 0---TV or VGA display not connected

uint Dacs_CntDtt ( )
{
        uint retval = 0;

        int ii = 0;

        uchar val = 0, dac[3] = {0};

        // Power up CH7025/26

        I2CWrite ( 0x04, I2CRead ( 0x04) & 0xFE );

        // Set bit 3,4,5 of register 04h to “1”

        I2CWrite ( 0x04, I2CRead ( 0x04) | 0x38 );

        // Set SPPSNS to “1”

        I2CWrite ( 0x7D, I2CRead ( 0x7D ) | 0x01); // Set SPPSNS to ‘1’

        //Delay some time (>=100ms)

        Sleep (200);

        // Read 0x7F to see the result

        val = I2CRead ( 0x7F ); // Read value of register 7Fh

        // Set SPPSNS to “0”

        I2CWrite ( 0x7D, I2CRead ( 0x7D) & 0xFE );

        // Set bit 3,4,5 of Register 04h to “0”
```

```
        I2CWrite ( 0x04, I2CRead ( 0x04) & 0xC7 );

        // Power down  CH7025/26

        I2CWrite ( 0x04, I2CRead ( 0x04) | 0x01 );

        // See the result …

        dac[0] = ( val & 0x03 ) >> 0; // Get DAC0 attach information

        dac[1] = ( val & 0x0C ) >> 2; // Get DAC1 attach information

        dac[2] = ( val & 0x30 ) >> 4; // Get DAC2 attach information

        if ( dac[0] == 0x01 )
        {
                retval |= ( 0x01 << 0 );
        }

        if ( dac[1] == 0x01 )
        {
                retval |= ( 0x01 << 1 );
        }

        if ( dac[2] == 0x01 )
        {
                retval |= ( 0x01 << 2 );
        }

        retuen retval;

}

// END HERE
```

Appendix C:

TEST PATTERN

CH7025/26 provides internal test pattern for debugging. In this appendix introduce how to use internal pattern to debug CH7025/26.

Address: 03h

BIT:	7	6	5	4	3	2	1	0
SYMBOL:	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	PG
TYPE:	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
DEFAULT	0	0	0	0	0	0	0	0

PG (bit 0): Register page selection bit. Set to 0 to select page 1, set to 1 to select page 2. Normally, it only need to set registers in page 1(PG=0) to make CH7025/26 work correctly.

Address: 04h(Page2)

BIT:	7	6	5	4	3	2	1	0
SYMBOL:	Reserved	Reserved	TEST	TSYNC	TSTP[3]	TSTP[2]	TSTP[1]	TSTP[0]
TYPE:	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
DEFAULT	0	0	0	0	0	1	0	0

TEST (bit 5): the test pattern enable bit.

TSYNC (bit 4): when set to 1, the internal generated sync will be used.

TSTP [3:0]: select the kind of the test patterns, as following table:

Test pattern selection

TSTP[3:0]	Test pattern
0	Black
1	White
2	Vertical ramp
3	Horizontal ramp
4	Color bar
5	One pixel wide color bar
6	Zigzag

Steps:

1) Finish Setting CH7025/26 as previous chapters in this document (Hint).

2) Set register 03h to “0x01”.

3) Set register 04h to “0x3**X**”, here **X** is the TSTP value based on the table above. For example, 0x34 is to select color bar.

4) Set register 03h to “0x00”.

After the steps above, the selected test pattern will showed on TV or VGA display. If not, or the image is not correct, it is indicated there is problem in your hardware or software or the register setting, please debug it until the test pattern showed on TV or VGA display.

To close the test pattern, following steps below:

1) Set register 03h to “0x01”.

2) Set register 04h to “0x04”.

3) Set register 03h to “0x00”.

Hint:

Generate test pattern requires that some registers to be set correctly, so you should first generate a set of correct register values using the programming tool (CH7025(26)RegSet.exe) and fill them into CH7025/26.

VERSION HISTORY

Rev.	Date	Page	Description
0.1	04/08/2007	All	First English Version (Beta)
1.01	08/10/2007	All	Second English Version (Released)
1.02	09/05/2007	All	Third English Version
1.03	10/10/2007	All	Forth English Version
2.00	01/05/2008	All	Fifth English Version
2.01	01/10/2008	All	First Chinese Version
2.02	25/02/2008	All	Sixth English Version

DESCRIPTION:

Version 0.1:

Used for progressive video input format which is RGB + H/V Sync or RGB + H/V Sync + DE, and output video format is TV (CVBS, S-Video, YPbPr) or VGA.

Version 1.01:

A full-function tool named CH7025_RegSet.exe is provided together with this document:

1) Add support to interlaced video input format.
2) Add support to CPU interface video input format.
3) Add support to TV(RGB+Csync) output format.
4) Add support to more features.
5) Add sample function and code to show how to program CH7025.

Version 1.02:

Modify CH7205_RegSet.exe to adjust the image quality when VGA out.

Version 1.03

Perfect the quality of big resolution input, such as 800x600, 800x480 and so on.

Version 2.00

The programming tool has been updated, and this version of programming guide document is produced based on the updated programming tool.

Add test pattern operation for debugging.

Version 2.01

The first Chinese version, translated from version 2.00.

Version 2.02

Full – function programming guide for CH7025/26, which add support for HDTV output format, and add some features.