Chrontel

CH7025/26 电视 (VGA) 编码器
编程指南

Information presented in this document is relevant to Chrontel driver module software, and Chrontel TV-Out chipset products. It may be only used for Chrontel software development aid. It may not be used for any other purpose. Chrontel and author of this document are not liable for misuse of information contained herein. Chrontel and author of document are not liable for errors found herein. Information contained herein may not be used within life support systems or nuclear facility applications without the specific written consent of Chrontel. **Do not distribute this document to non-designated unauthorized parties in any form without the specific written consent of Chrontel. Do not read and destroy this document if you are not designated recipient. Information contained herein is subject to change with or without notice. Communicate with Chrontel Software and Systems Engineering Department for up-to-date changes.**

Revision 2.00
January 02, 2008

Prepared By: Junfeng
Reviewed By:

目录

前言

本文档包括两章和两个附录：

第一章介绍怎样使用 CH7025/26 的 DAC 连接侦测功能。

第二章介绍怎样使用编程工具 CH7025(26) RegSet.exe。

附录 A 介绍怎样使用 CH7025/26 的特性功能。

附录 B 提供了对 CH7025/26 编程的完整的示例代码。

事实上，使用编程工具 CH7025(26) RegSet.exe 对 CH7025/26 进行编程是非常容易的，您可以只参考第二章和附录 B 就可以快速的完成主要的编程，这样电视或者显示器就可以看到 CH7025/26 输出的图像了。

第一章：连接侦测

CH7025/26 能够侦测与电视或者显示器的连接状况。在使用这个功能的时候必须连接一个晶振到 CH7025/26。如果不想使用这个功能，跳过这一章。

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

寄存器 7Dh 的 SPPSNS 位是连接侦测的触发位。

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

寄存器 7Fh 是只读寄存器：

DACAT2[1:0]：返回 DAC2 的连接信息。
DACAT1[1:0]：返回 DAC1 的连接信息。
DACAT0[1:0]：返回 DAC0 的连接信息。

关于各个 DAC 返回值的含义参照下面的表格 2-1：

表 2-1: 连接信息含义

DACATn[1:0]	含义
00	电视或显示器没有连接
01	电视或显示器已连接
11	接地
10	保留

接下来介绍使用 CH7025/26 连接侦测功能的步骤：

1) CH7025/26 上电.

2) 将寄存器 04h 的位 3，位 4 和位 5 设置为 1。

3) 将寄存器 7Dh 的 SPPSNS 设置成 1。

4) 延迟一些时间(>=100ms)。

5) 读取寄存器 7Fh 的值。

6) 将寄存器 7Dh 的 SPPSNS 设置成 0。

7) 将寄存器 04h 的位 3，位 4 和位 5 设置为 0。

8) CH7025/26 断电。

根据寄存器 7Fh 返回的值，结合表 2-1，可以判断哪个 DAC 已经连接到电视或显示器，以及哪个 DAC 没有连接。

示例代码： 请参考附录 B。

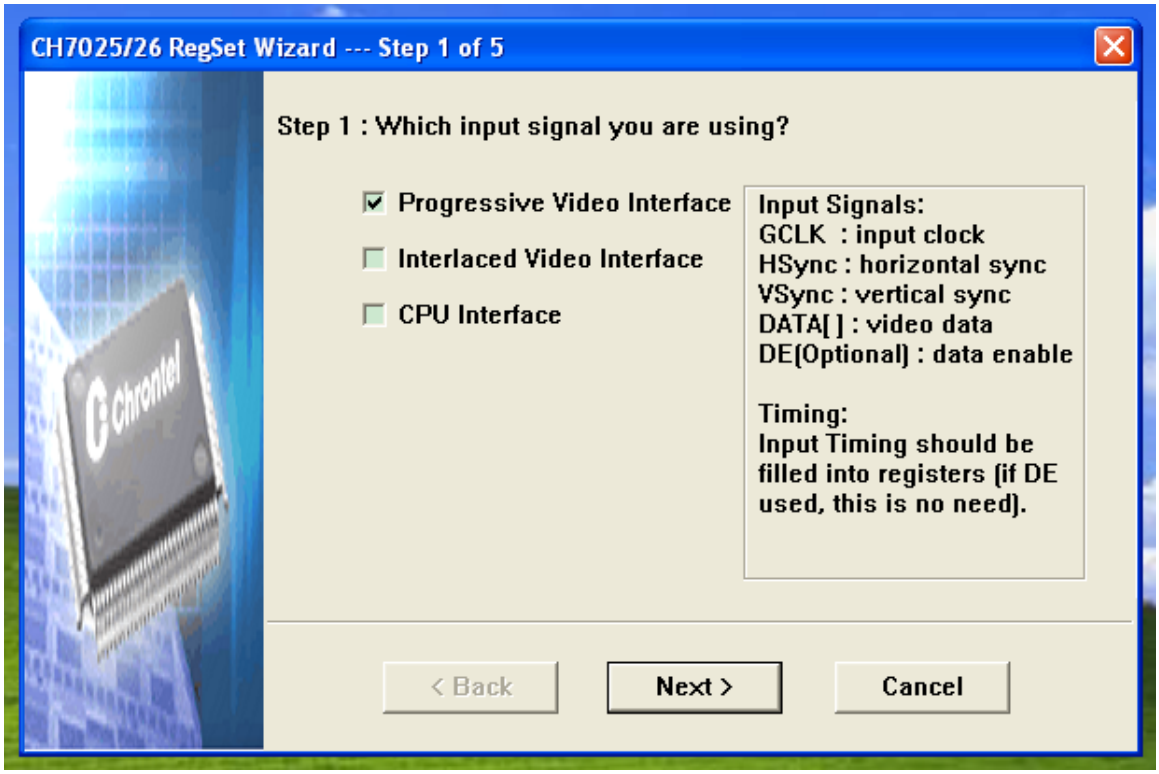
第二章：使用工具

编程工具 CH7025(26) RegSet.exe 能够帮助使用 CH7025/26 的工程师快速的生成可工作的 CH7025/26 寄存器列表。接下来的各部分介绍怎样使用这个编程工具。

2.1 完成向导

当 CH7025/26 RegSet.exe 开始运行的时候，它提供一个包括五个步骤的向导来帮助你完成基本信息的设定。下面介绍向导程序的使用：

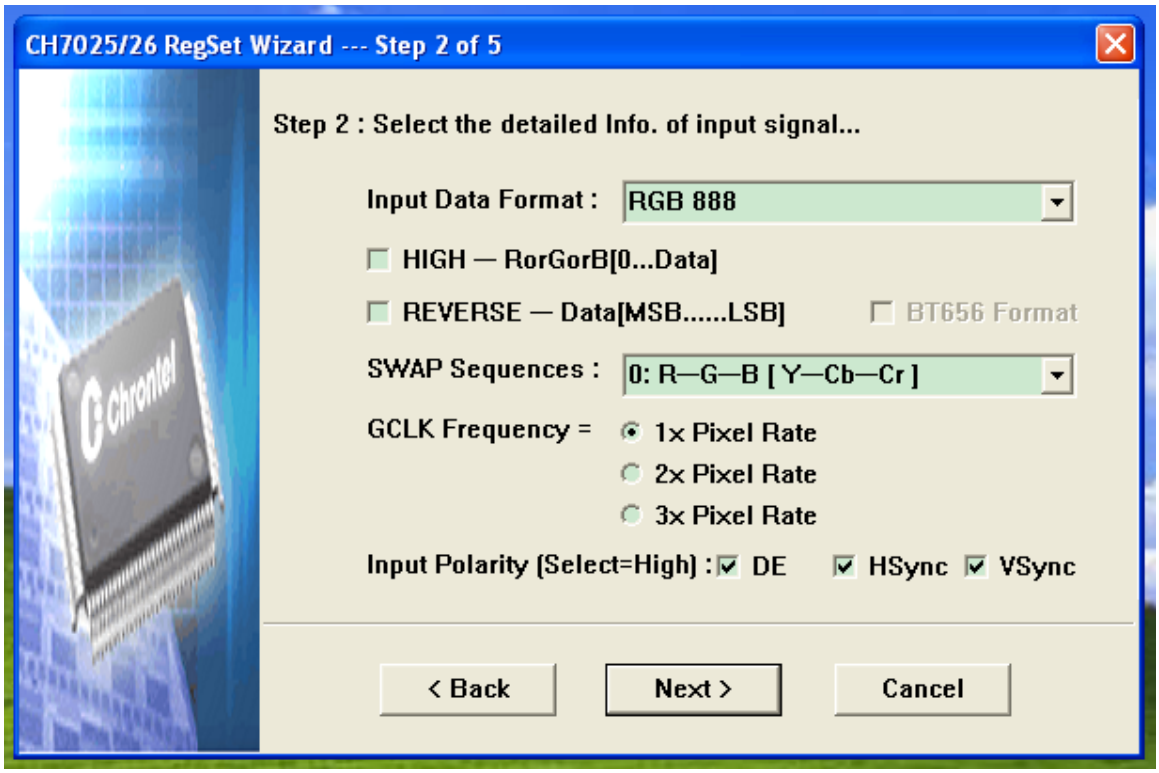
2.1.1 向导步骤一：



在步骤一中，确定 CH7025/26 输入信号的格式，在选择格式的同时，程序右边部分会显示所选格式的基本特征和要求。I

完成此步骤，按”Next>”进入下一步。

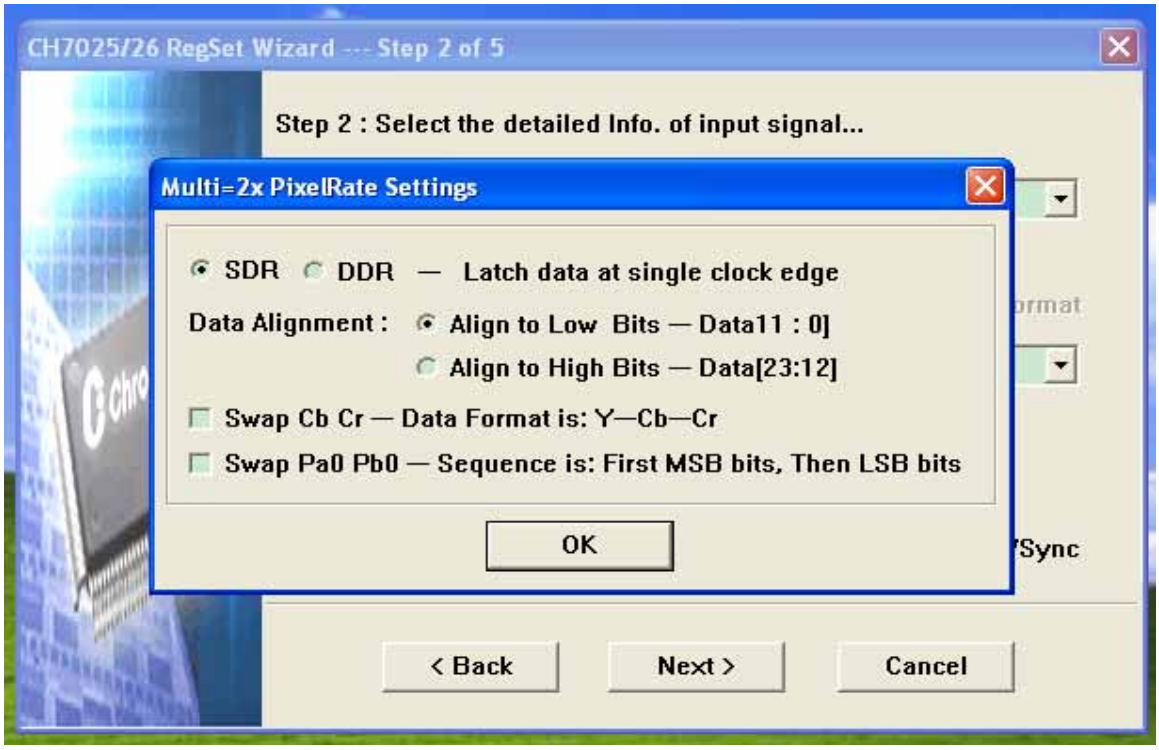
2.1.2 向导步骤二：



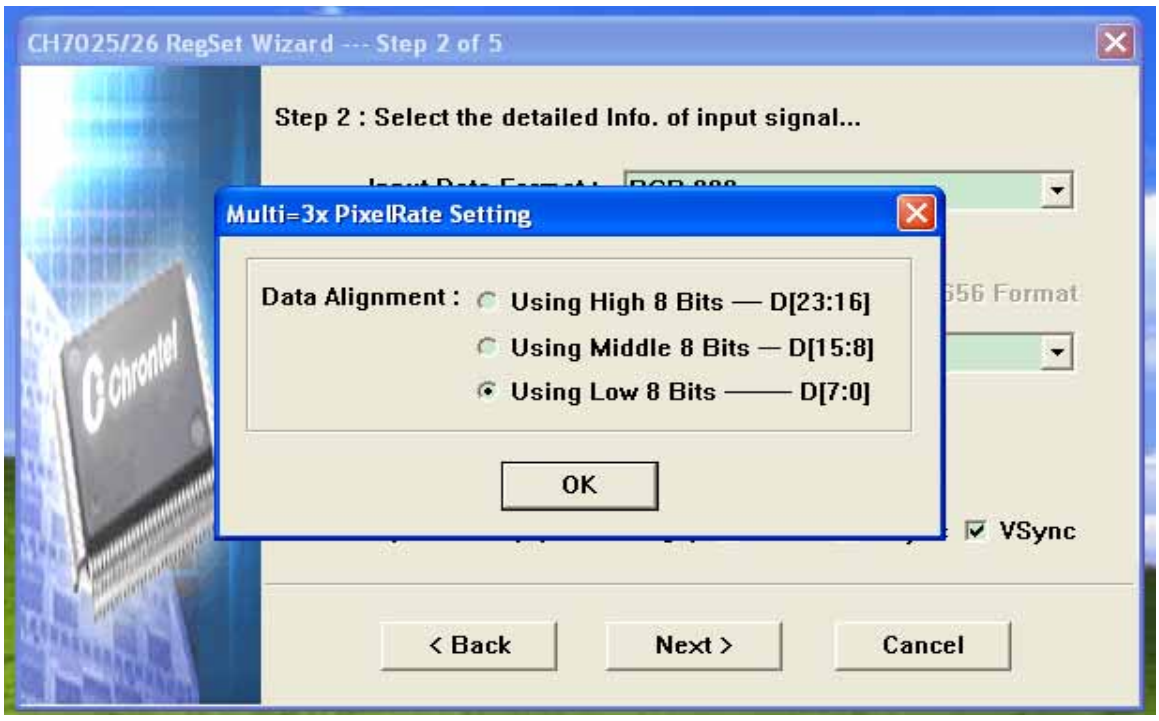
在步骤二中，设定关于输入信号的详细信息。

请注意 “BT656 Format”复选框，当选择”Input Data Format”是 YCbCr4:2:2 的时候，这个复选框将会被使能，如果输入的信号格式遵循 BT656 标准，那么选中这个复选框。

当选择”GCLK Frequency”等于”2x Pixel Rate”的时候，会弹出一个对话框帮助设定在此种情况下的详细信息，如下图，选择相应的选项，然后按”OK”。关于 SDR 和 DDR 的具体含义，请参考提示一。



当选择”GCLK Frequency”等于”3x Pixel Rate”的时候，会弹出一个对话框帮助设定在此种情况下的详细信息，如下图，选择相应的选项，然后按”OK”。



完成此步骤，按”Next>”进入下一步。

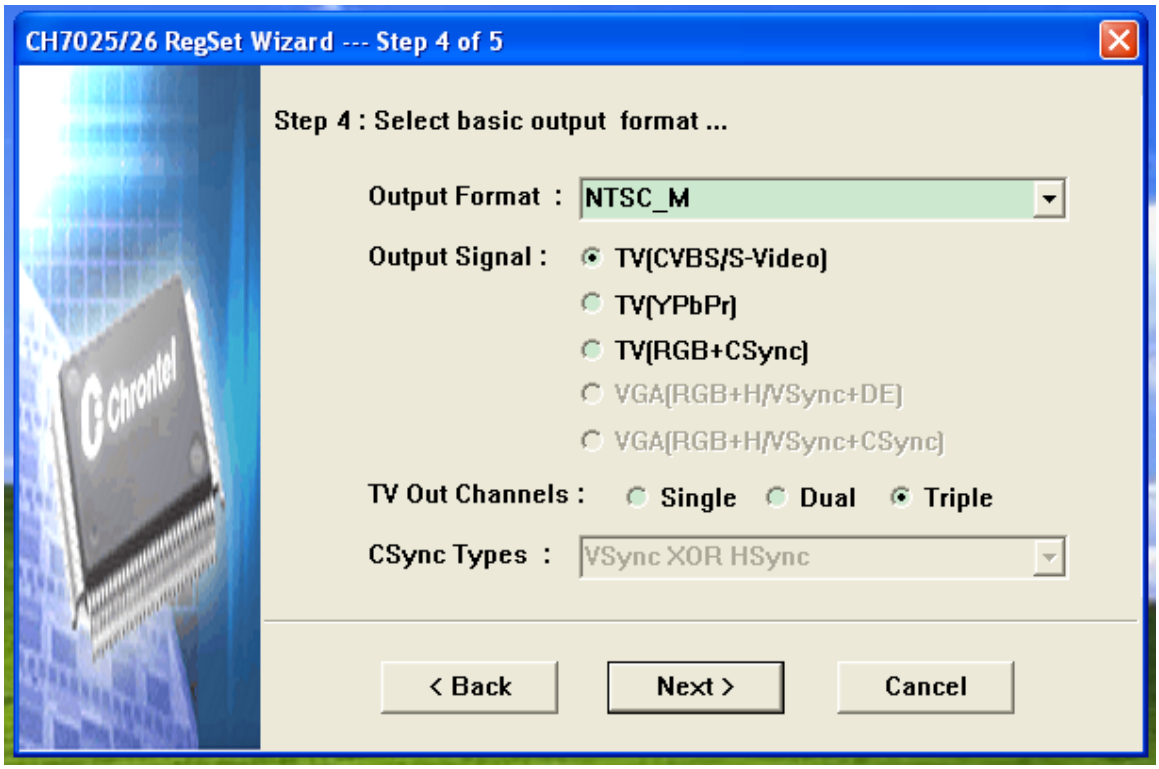
2.1.3 向导步骤三：

在步骤三中，设定输入信号的 timing 信息：HTI，HAI，VTI，VAI 等等（提示二），如果选项没有使能，表明此项信息不需要设定。

请注意“HVAUTO”复选框，如果选中这个复选框，输入信号的 timing 信息不需要设定，在这种情况下，CH7025/26 会根据 Hsync， Vsync 和 DE 信号自动计算输入的 timing 信息。因此，当“HVAUTO”复选框被选中的情况下，DE 信号是必需的，同时，必须设定输入视频信号的刷新率。

完成此步骤，按 “Next>” 进入下一步。

2.1.4 向导步骤四：

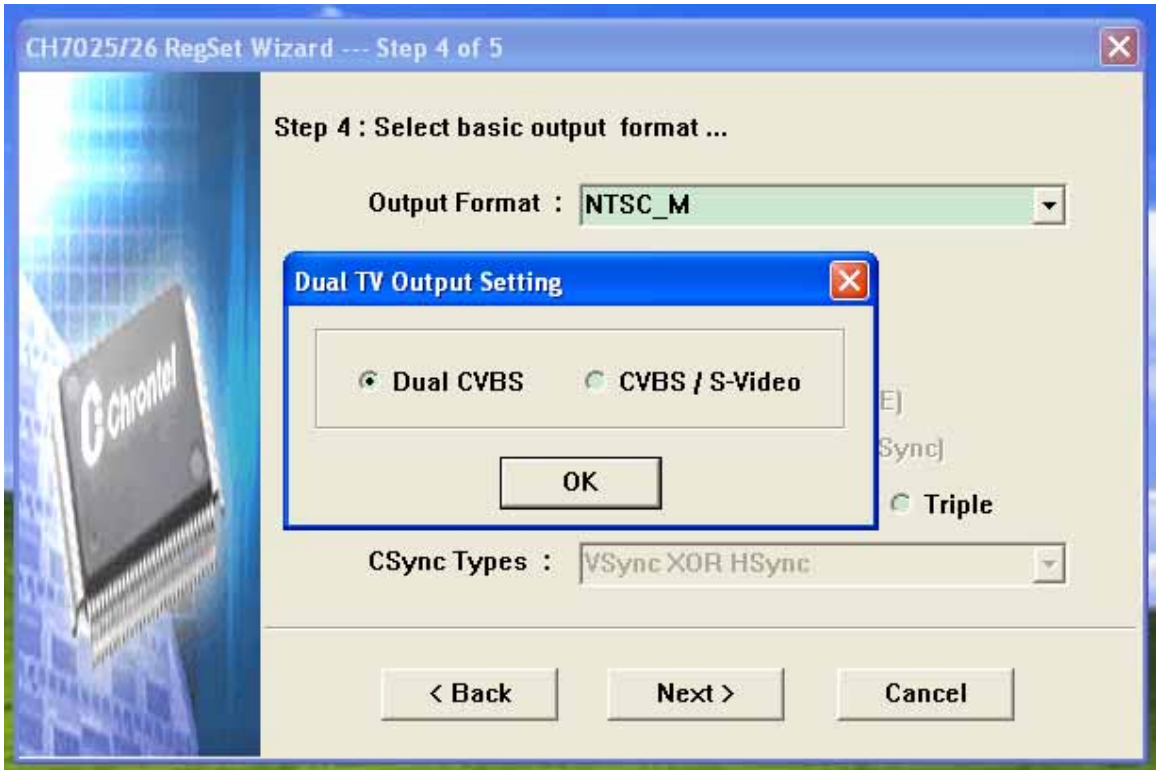


在步骤四中，设定输出信号的格式。

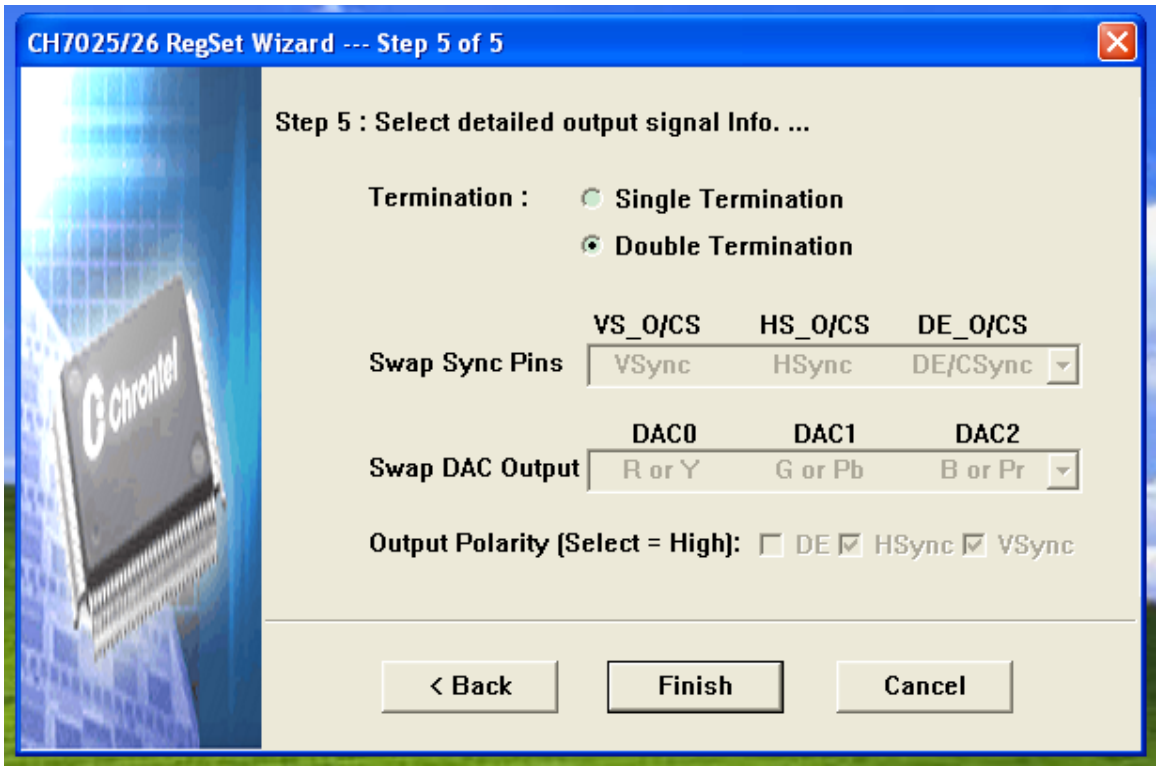
请注意 “Output Signal”的五个单选框，CH7025/26 支持五种输出信号接口，其中三个针对电视格式，两个针对 VGA 格式。

当选择了 “Dual”单选框，会弹出一个对话框帮助设定两路输出情况下的详细信息，如下图，选择相应的选项，按 “OK”。

完成此步骤，按”Next>”进入下一步。


CH7025/26 RegSet Wizard --- Step 4 of 5
Step 4 : Select basic output format ...
Output Format :
NTSC_M
Dual TV Output Setting
Dual CVBS
CVBS / S-Video
OK
Triple
CSync Types :
VSync XOR HSync
< Back
Next >
Cancel

2.1.5 向导步骤五：

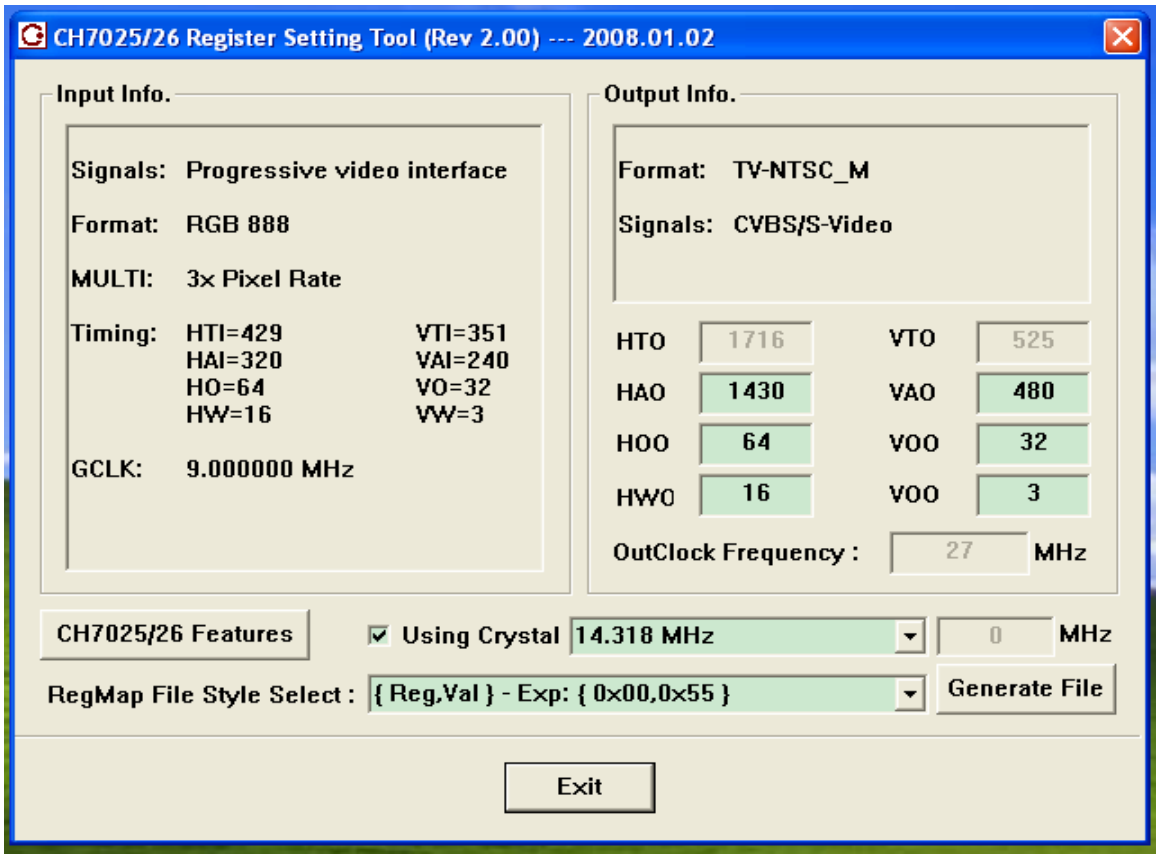


在步骤五中，设定输出信号的详细信息。

完成此步骤后，按”Finish”按钮来完成整个向导程序。此时，基本的信息已经设定完成。

2.2 完成最后部分的设定

完成向导程序后，会弹出最后一步设定的对话框，如下图：



这个对话框会显示在向导程序中设定的基本信息：

“Input Info.”：显示输入信号的基本信息。

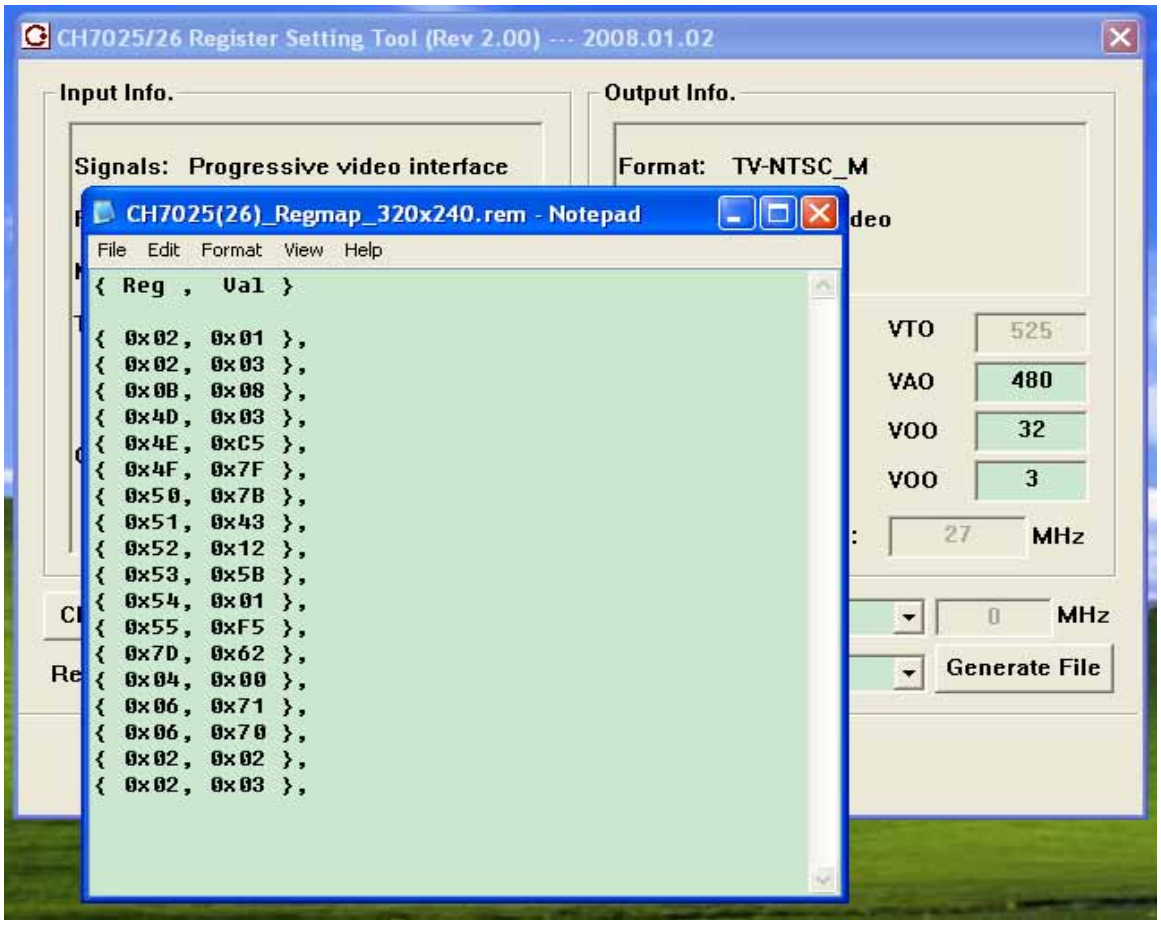
“Output Info.”： 显示输出信号的基本信息。需要特别注意的是，当输出信号是电视格式时，”HAO”，”VAO”，”HOO”，”VOO”，”HWO”，”VWO”这几个设定是可以改变的，但是”HTO”和”VTO”以及”OutClock Frequency”是不能改变的，因为对于电视格式来说这三个参数是固定不变的；当输出信号是 VGA 格式时，上述这些参数都是可以改变的。

“CH7025/26 Features” ：按此按钮可以设定 CH7025/26 的一些属性。

“Using Crystal” ：设定是否使用晶振。当这个复选框没有被使能时，表示必须使用晶振。（例如，当在向导中选择了 CPU 接口的输入信号时，晶振是必须使用的，这个时候这个复选框就不会被使能）。右面的下拉框和编辑框用来设定晶振的频率。

2.3 生成寄存器设定文件

完成上面所有的设定之后，接下来生成一个寄存器设定文件。首先，选择文件的格式，共有三种格式供选择：第一种数组形式的寄存器列表，第二种是带有 I2C 函数的程序列表，第三种是 Chrontel 内部使用的文件格式；然后按"Generate File"按钮，寄存器设定文件将会被生成并会以记事本的程序打开。



这样，对 CH7025/26 的编程工作已经基本上完成，当你按照寄存器设定文件设定完 CH7025/26 后，你已经可以在电视或 VGA 显示器上看到图像（注）。请参考附录 B 中的示例代码。

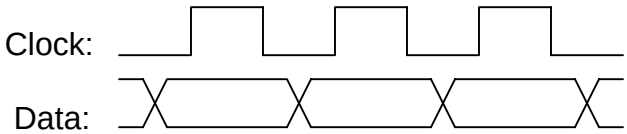
接下来，你可以修改和改善在电视或 VGA 显示器上图像的质量和属性，象图像在屏幕上的位置，图像的亮度，图像的放大，旋转，反转等等。请参考附录 A 中的详细信息。

注: 我们强烈建议设定 CH7025/26 的时候，完全按照寄存器设定文件中的寄存器顺序来设定 CH7025/26.

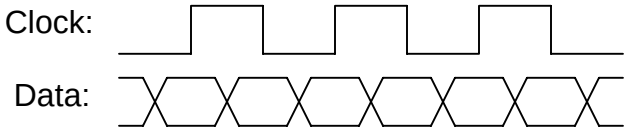
提示一：

SDR 和 DDR 的定义：

SDR:

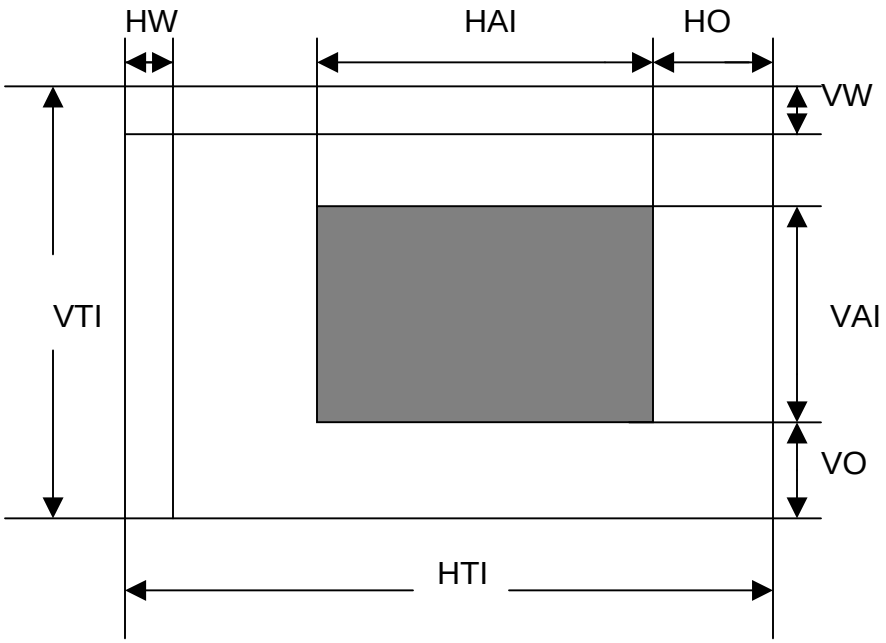


DDR:



提示二：

Chrontel 输入 timing 设定定义：



附录 A：特性调节

CH7025/26 提供了调整图像质量和属性的功能，可以调整的功能和属性如下：

1) 灰度调整
2) 饱和度调整
3) 对比度调整
4) 亮度调整
5) 锐度调整
6) 显示位置调整
7) 图像反转功能
8) 图像旋转功能
9) 图像放大功能

接下来分别介绍怎样使用这些属性和功能：

1) 灰度调整

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) 可以调整图像颜色的灰度，图像颜色的灰度调整的公式是 (HUE[6:0]-64)/2 度。上电时默认值为 0。

使用说明：

通常情况下，灰度的默认值是一个比较合理的值。可以写新的灰度值到寄存器 2Eh，灰度值的范围为 0 ~ 127。

示例函数：

```
// 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) 饱和度调整

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) 调整图像颜色的饱和度，增大会提供饱和度，反之亦然。

使用说明：

通常情况下，饱和度的默认值是比较合理的，可以写新的饱和度值到寄存器 2Fh，饱和度的值的范围为 0 ~ 127。

示例函数：

```
// 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) 对比度调整

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) 调整图像的对比度，增大会提高对比度，反之亦然。

使用说明：

通常情况下，对比度的默认值是比较合理的，可以写新的对比度值到寄存器30h，对比度的值的范围为 0 ~ 127。

示例函数：

```
// 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) 亮度调整

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)调整图像的亮度，增大会提高亮度，反之亦然。

使用说明：

通常情况下，亮度的默认值是比较合理的，可以写新的亮度值到寄存器 31h，亮度的值的范围为 0 ~ 255。

示例函数：

```
// 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) 锐度调整

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) 控制锐度。

使用说明：

锐度的默认值 4，可以根据应用的不同适当的调整图像文字的锐度以达到需要的效果。锐度的范围为 0~6。

示例函数：

```
// 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) 显示位置调整

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]和 VP[11:8]合并成 VP[11:0] 定义了图像在垂直方向的显示位置。VP[11:0]-2048 决定了调整的行数，如果这个值是正的，图像就往上移，反之亦然。

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]和 HP[11:8]合并成 HP[11:0]定义了图像在水平方向的显示位置。HP[11:0]-2048 决定了调整的点数，如果这个值是正的，图像会往右移，反之亦然。

使用说明：

通常情况下，HP 和 VP 的默认值使图像在屏幕的中心位置显示，可以根据实际情况做适当的调整。HP 和 VP 的范围为 0~4095。

示例函数：

```
// 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) 图像反转功能

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

VFLIP (bit 3) 控制垂直方向的反转功能。

HFLIP (bit 2) 控制水平方向的反转功能。

使用说明：

如果要使图像在垂直方向反转，将 VFLIP 设置成 1；如果想使图像在水平方向反转，将 HFLIP 设置成。

示例代码：

```
// 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) 图像旋转功能

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) 控制图像的旋转功能。

使用说明:

ROTATE[1:0]等于”00”时，图像不旋转；
ROTATE[1:0]等于”01”时，图像旋转 90 度；
ROTATE[1:0]等于”10”时，图像旋转 180 度；
ROTATE[1:0]等于”11”时，图像旋转 270 度。

示例代码：

```
// 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) 图像放大功能

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) 控制图像的放大功能。

HEND[10:8]和寄存器 29h 的 HEND[7:0]合并成 HEND[10:0]定义水平方向的终止位置。

HST[10:8]和寄存器 28h 的 HST[7:0]合并成 HST[10:0]定义水平方向的起始位置。

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]和寄存器 2Ch 的 VEND[7:0]合并成 VEND[10:0]定义垂直方向的终止位置。

VST[10:8]和寄存器 2Bh 的 VST[7:0]合并成 VST[10:0]定义垂直方向的起始位置。

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

使用说明：

图像的放大程度是在输入 timing 的基础上进行的，例如，如果想将左上角四分之一的图像放大到整个屏幕，要根据输入 timing 的参数（HTI，HAI，HO，HW，VTI，VAI，VO，VW）来计算 HST，HEND，VST 和 VEND，而不是根据输出的 timing 计算！

HST，HEND， VST 和 VEND 的范围如下：

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

使用图像放大功能的步骤如下：

1) CH7025/26 暂停工作（设置寄存器 06h 的 STOP 为 1）。
2) 计算并写入 HST， HEND， VST 和 VEND 的值到 CH7025/26。
3) 设置 IMGZOOM 为 1。
4) 恢复 CH7025/26 正常工作。

示例函数：

```
// 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->vEnd;
                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 );
```

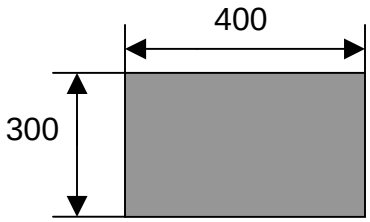
示例代码：

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 };
```

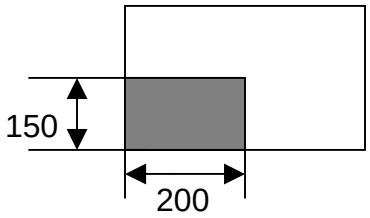
Example 1: zoom 100% image that looks like no zoom:

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



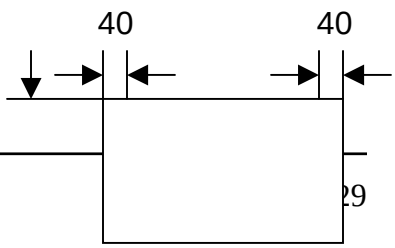
Example 2: zoom down-left quarter of the image:

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

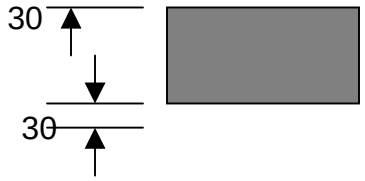


Example 3: zoom center:

```
ZOOMSIZE zs;
```

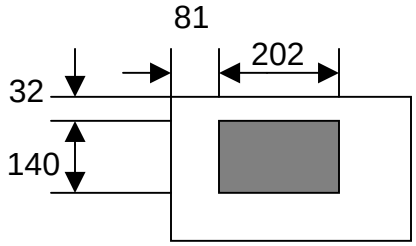


```
zs.hStart = 41;
zs.hEnd  = 360;
zs.vStart = 31
zs.vEnd  = 270;
ZoomEnable ( &sz, &tm, TRUE );
```



Example 4: generic zoom:

```
ZOOMSIZE zs;
zs.hStart = 82;
zs.hEnd  = 283;
zs.vStart = 33;
zs.vEnd   = 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 );
```

附录 B：示例代码

这里提供完整的例子来介绍怎样编程 CH7025/26.。

假定：

1) 使用 C 代码。

2) I2C 函数已经在其他地方定义：
unsigned char I2CRead (unsigned char index);
void I2CWrite (unsigned char index, unsigned char value);

3) 寄存器设定文件已经生成，并且已经复制到数组 REG_MAP[ ][2]。

4) 作为 CH7025/26 输入的视频信号已经稳定。

注：强烈建议在实际的配制过程中，首先配制送出视频信号的 LCD 控制器，在视频信号已经稳定之后再配制 CH7025/26。

示例代码：

```
// 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; // Get DAC0 attach information

        dac[1] = val & 0x0C; // Get DAC1 attach information

        dac[2] = val & 0x30; // 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 );
        }

        I2CWrite ( 0x7D, I2CRead ( 0x7D ) & 0xFE ); // Set SPPSNS to ‘0’

        retuen retval;

}

// END HERE
```

修订历史

Rev.	Date	Page	Description
0.1	04/08/2007	全部	第一个英文版本(测试)
1.01	08/10/2007	全部	第二个英文版本
1.02	09/05/2007	全部	第三个英文版本
1.03	10/10/2007	全部	第四个英文版本
2.00	01/05/2008	全部	第五个英文版本
2.01	01/10/2008	全部	第一个中文版本

描述：

版本 0.1：

只能用于 RGB + H/V Sync 或 RGB + H/V Sync + DE 的视频输入信号， 输出信号是 TV (CVBS， S-Video， YPbPr)或者 VGA。

版本 1.01：

全功能的编程工具 CH7025_RegSet.exe 将会和此文档一起使用：

1) 支持 interlaced 视频输入信号
2) 支持 CPU 接口的视频输入信号
3) 支持 TV(RGB+Csync) 格式的输出信号
4) 支持更多的特性
5) 加入示例代码

版本 1.02：

修改编程工具 CH7205_RegSet.exe 来改善了当 VGA 输出时的图像质量。

版本 1.03：

改善对大分辨率的输入信号（800x600，800x480）的支持。

版本 2.00：

更新编程工具 CH7025(26) RegSet.exe，此工具采用更人性化的方式帮助对 CH7025/26 进行编程。

版本 2.01：

第一个中文版本，其功能和版本 2.00 相同。