Chrontel

CH7025 TV ENCODER PROGRAMMING GUIDE

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.**

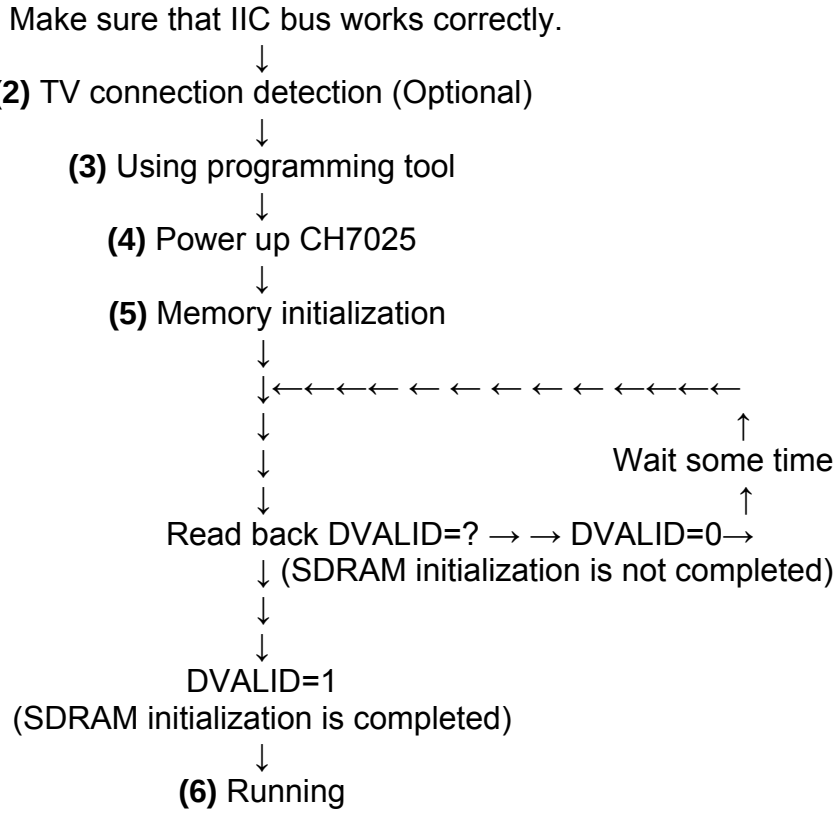
Revision 1.02
September 05, 2007

Prepared By: Junfeng
Reviewed By:

CONTENTS

PROGRAMMING SEQUENCE

REGISTER SETTING SEQUENCE



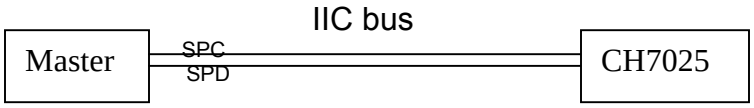
The following chapters show how to finish step 1 to step 6.

Chapter 1:

YOU SHOULD KNOW

1) As CH7025 is controlled through IIC bus as a client, the IIC bus should work correctly before programming CH7025.

The device address of CH7025 is 0xEC(for write) or 0xED(for read).



2) Chrontel will provide CH7025 programming tool (CH7025_RegSet.exe) together with this document, Chapter 3 will introduce how to use this tool.

3) This programming guide introduces 6 basic steps to make CH7025 run. Appendix introduces how to use the features of CH7025 and a whole program example. Also sample code will be provided in this document.

4) It is recommend that you should read the Datasheet of CH7025 before programming CH7025,

Chapter 2:

TV CONNECTION DETECTION

This step is optional: if you confirm that your TV or VGA display has been connected to CH7025, go to Chapter 3: Using Programming Tool.

You can use this feature of CH7205 before or after CH7025 powered up:

To use this feature before CH7025 power up, a crystal is required to connect to CH7205 chip, and set DISPON (bit 1 of 7Dh) to “0”, which is just default value.
To use this feature after CH7025 power up, the crystal is not necessary, but you should set DISPON (bit 1 of 7Dh) to “1”.

Here, we assume that CH7025 isn’t powered up, so DISPON should be set to “0”.

CH7025 can detect connection of TV or VGA display. You could confirm the connection between CH7025 and TV or VGA display in this step.

Address: 7Dh

BIT:	7	6	5	4	3	2	1	0
SYMBOL:	Reserved	Reserved	Reserved	Reserved	Reserved		DISPON	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 5-4) return attach information for DAC3 channel according to the table2-1 below.
DACAT1[1:0] (bits 3-2) return attach information for DAC2 channel according to the table2-1 below.
DACAT0[1:0] (bits 1-0) return attach information for DAC1 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:

Usage:

1) Set DISPON to “0”(Using Crystal) or “1”(Not using Crystal).
2) Set SPPSNS to ‘1’.
3) Read the value of register 7Fh.
4) Check if the connection is correct according table2-1.
5) Set SPPSNS to ‘0’.
6) If the connection is not correct, check the connection of CH7025 and TV or VGA display and repeat from 1).

Sample code:

```
        int i = 0;

       unsigned char val = 0, dac[3] = {0}; // assume “unsigned char” is 8-bit wide.

        val = I2CRead ( 0x7D); // Read value of register 7Dh

#ifdef USE_CRYSTAL
        val &= 0xFD; // Set DISPON to “0”
#else
        val |= 0x02; // Set DISPON to “1”
#endif

        val |= 0x01; // Set SPPSNS to ‘1’

        I2CWrite ( 0x7D, val );

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

        dac[0] = val & 0x03; // Get DAC0 attach information
```

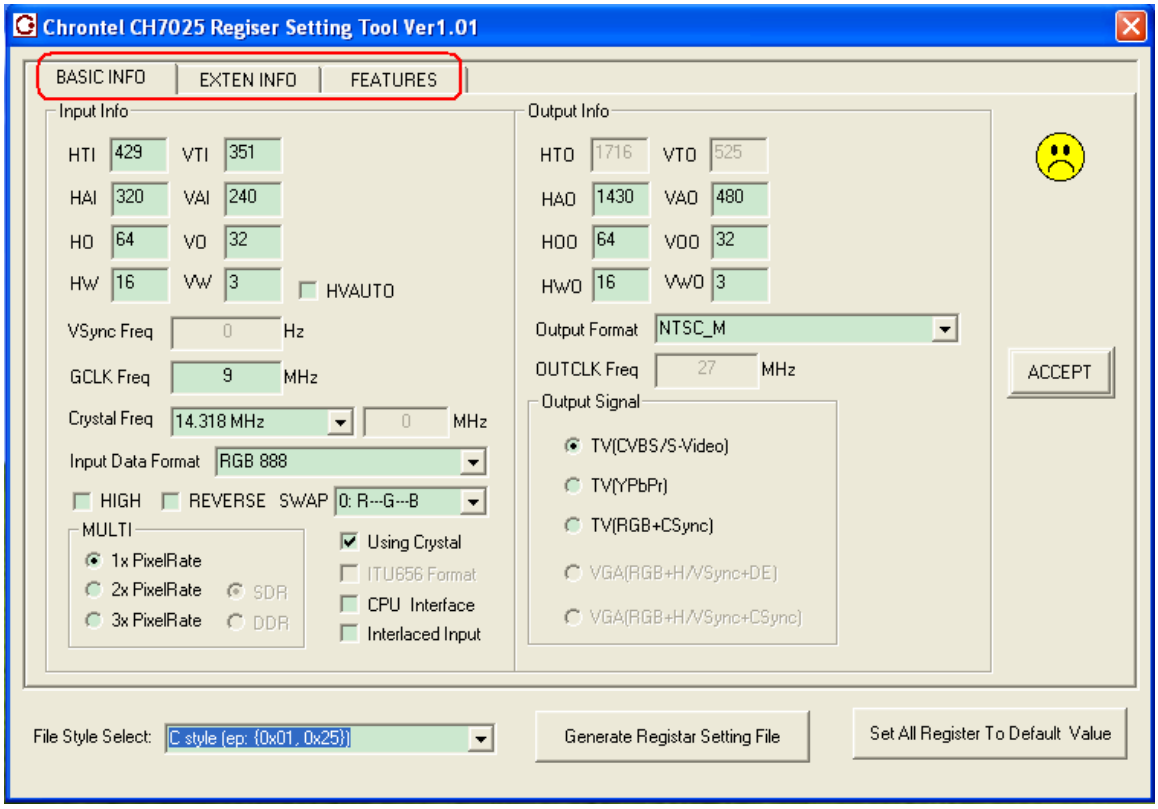
```
dac[1] = val & 0x0C; // Get DAC1 attach information

dac[2] = val & 0x30; // Get DAC2 attach information

for( i=0; i!=3 ; ++i)  // Check if connection of 3 DACs is correct
{
        switch( dac[i] )
        {
        case 0:
                //No attached display
                break;
        case 1:
                //Connect
                break;
        case 2:
                //Short to ground
                break;
        default:
                //Error occur
        }
}
val  = I2CRead ( 0x7D );
val &= 0xFE;
I2CWrite ( 0x7D, val ); // Set SPPSNS to '0'
```

Chapter 3:

USING PROGRAMMING TOOL

Chrontel will provide a tool for programming CH7025. With this tool, it will be very easy to program CH7025. In this chapter, we introduce the usage of this tool, which is named CH7025_RegSet.exe.

The interface of CH7025_RegSet.exe is following: it has three tab-pages, “BASIC INFO”, “EXTEN INFO” and “FEATURES”.



Now we introduce how to use the this tool to program CH7025, including following steps:

1) Set basic information. (“BASIC INFO” tab-page)

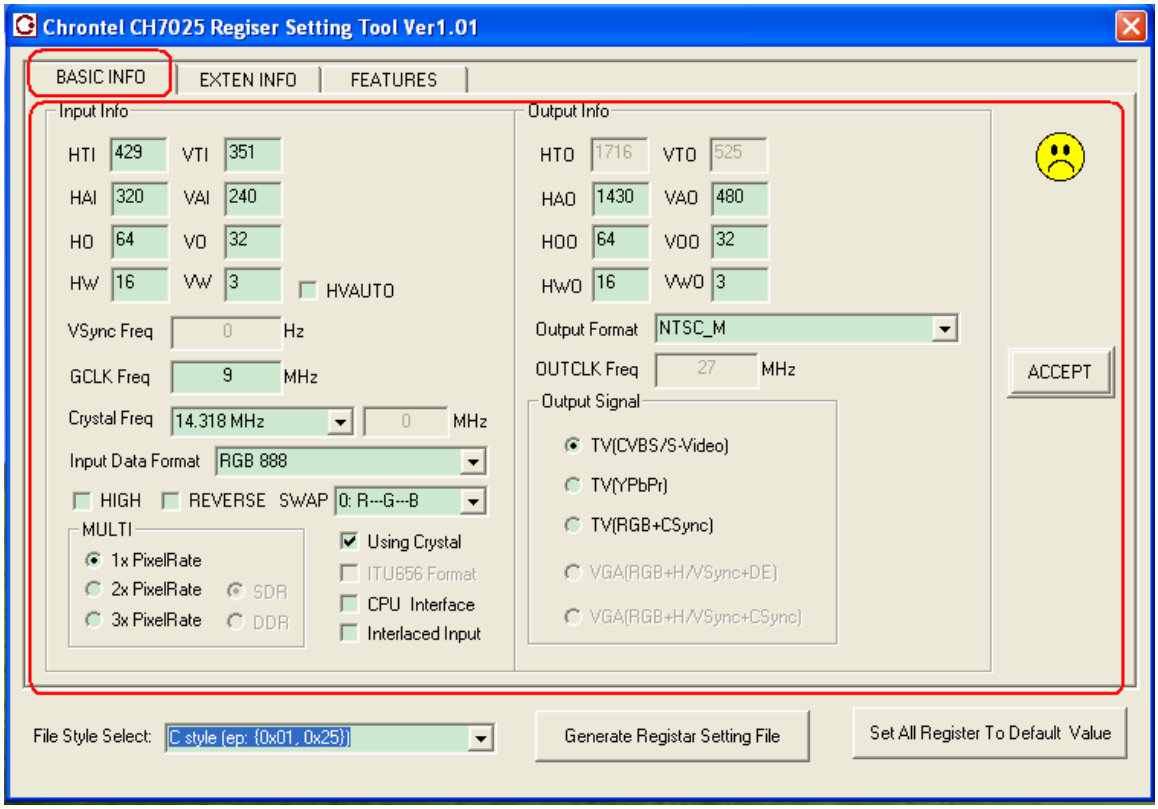
2) Set extended information. (“EXTEN INFO” tab-page)

3) Set features. (“FEATURES” tab-page)

4) Generate register map file.

1) Set basic information:

The interface of basic information is as following:



Basic information tab-page mainly includes basic input information and output information.

Items introduction:

Input Info:

"HTI": horizontal total pixels of input timing. (hint1)

"HAI": horizontal active pixels of input timing.

"HO": horizontal offset pixels of input timing.

"HW": horizontal sync width in pixels of input timing.

"VTI": vertical total pixels of input timing.

"VAI": vertical active pixels of input timing.

"VO": vertical offset pixels of input timing.

"VW": vertical sync width in pixels of input timing.

"HVAUTO": if enable the "HVAUTO" function of CH7025. (Check CH7025 datasheet for details)

"Vsync Freq": input frame rate frequency.

"GCLK Freq": input clock frequency.

"Crystal Freq": frequency of the crystal connected to CH7025. It can be the pre-defined value or any other values.

"Input Data Format": input data format just as its name.

"HIGH": if data align to high bits. (Check CH7025 datasheet for details)

"REVERSE": if data reversed. (Check CH7025 datasheet for details)

"SWAP": if data swapped. (Check CH7025 datasheet for details)

"MULTI": if is multi-input mode, it includes 1x input rate, 2x input rate and 3x input rate. (Check CH7025 datasheet for details)

"SDR and DDR": in 2x input rate mode, single edge or double edge can be selected. (Fig 3.1 below)

"Using Crystal": CH7025 can work with or without a crystal. We strongly recommend that connect a crystal to CH7025 chip.

"ITU656 format": TV format, if input of CH7025 is ITU656 TV format, make this check box selected.

"CPU interface": CH7025 support CPU-interface input data format, which has 3 signals besides data pins: CSB (chip select), VSYNC (frame sync), WEB (write enable).

"Interlaced input": select if the input date is interlaced.

Output Info:

"HTO": horizontal total pixels of output timing.

"HAO": horizontal active pixels of output timing.

"HOO": horizontal offset pixels of output timing.

"HWO": horizontal sync width of output timing in pixels.

"VTO": vertical total pixels of output timing.

"VAO": vertical active pixels of output timing.

"VOO": vertical offset pixels of output timing.

"VWO": vertical sync width of output timing in pixels.

"Output Format": Output video format.

"OUTCLK Freq": output clock frequency.

"Output Signal": output video signal type. CH7025 supports 5 kinds of output video signal types:

TV(CVBS/S-Video),
TV(YPbPr),
TV(RGB+Csync),
VGA(RGB + Hsyns +Vsync + DE)
VGA(RGB + Hsync + Vsync + Csync).

Usage:

Now we introduce how to set basic information. Before operate on the tool, you should decide the input video format of CH7025 first. Generic, CH7025 accepts three kinds of video format:

a) Normal progressive video signal.
b) Interlaced video signal.
c) CPU interface video signal.

If the input is a), make "Interlaced input" and "CPU interface" **un-selected**.
If the input is b), make "Interlaced input" check box selected.
If the input is c), make "CPU interface" check box selected.

CH7025_RegSet.exe will disable some units automatically when you operate on it. The disabled Units mean that they need not be set under your selected mode. For example, if you select "CPU interface" check box, HTI and VTI edit box will be disabled, HAI and VAI are not disabled, which show that under CPU interface, HTI and VTI are not required, but HAI and VAI are necessary.

Based on the three different input video formats, we introduce them respectively.

a) Normal progressive video signal.

Step 1) Make sure that “CPU interface” and “Interlaced input” check boxes un-selected.

Step 2) Decide your input timing information.

Fill the input timing information into the corresponding units: HTI, HAI, HO, HW, VTI, VAI, VO and VW. If you select “HVAUTO”, these values need not to be filled in, but you should fill in your input frame rate frequency.

Fill the input clock frequency (GCLK freq). The range of the frequency is from 2.3MHz to 120MHz, any other values can not be accepted.

Step 3) Decide your crystal frequency.

If you do not want use crystal, un-select the “Using crystal” check box, and skip this step.

We recommend that you connect one crystal to CH7025, here tell CH7025 the crystal frequency: Make the “Using Crystal” check box selected. If your crystal frequency is in the pre-defined frequencies, select it in the “Crystal Freq” combo box, otherwise fill it in the edit box.

Step 4) Decide your input data format.

CH7025 accepts 12 kinds of data format:
(1) RGB 888
(2) RGB 666
(3) RGB 565
(4) RGB 555
(5) DVO
(6) 8-bit YCbCr 4:2:2
(7) 10-bit YCbCr 4:2:2
(8) YCbCr 4:4:4
(9) YCbCr 4:4:4 with embedded sync
(10) Consecutive RGB 666
(11) Consecutive RGB 565
(12) Consecutive RGB 555

You can check CH7025 datasheet for details about input data format. Here select input data format from the “Input Data Format” combo box.

According your input data format, select “HIGH”, “REVERSE” and “SWAP” units. Please check CH7025 datasheet for details about these three items.

Attention: when you select YCbCr 4:2:2 format, Hsync and Vsync signal flow into CH7025 must be high polarity!

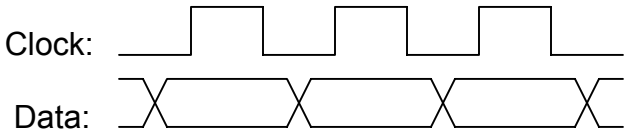
Step 5) Decide if your input data is multiple-rate to pixel rate.

CH7025 accepts three clock rates to pixel rate: 1x, 2x and 3x:

1x means that input clock frequency is equal to pixel rate.
2x means that input clock frequency is double of pixel rate.
3x means that input clock frequency is triple of pixel rate.

In this step, select the correct multiple rate. When you select “2x Pixelrate”, “SDR” or “DDR” can be changed. SDR means that for each clock pulse you send date one time; DDR means that for each clock pulse, you send data two times, one at raising edge of the clock pulse, and one at falling edge of the clock pulse. Following shows wave of SDR and DDR:

SDR:



DDR:

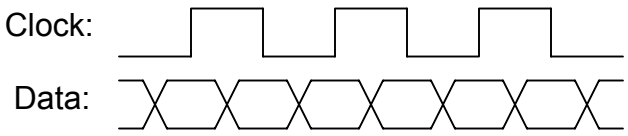


Fig 3.1

Step 6) Decide the output video format of CH7025.

CH7025 support TV out and VGA out formats, for TV out format, it supports NTSC_M, NTSC_J, NTSC_443, PAL_B/D/G/H/I, PAL_M, PAL_N, PAL_Nc and PAL_60; for VGA out format, it supports VGA and VGA (bypass scaler). Please check CH7025 datasheet for details.

From the “Output Format” combo box, select which output format will be the output data format of CH7025 as you want.

Step 7) Decide output timing information.

If the output format you select is TV out, the recommended timing values (HTO, HAO, HOO, HWO, VTO, VAO, VOO, VWO) and output clock frequency will be filled automatic. But you also can modify them except HTO, VTO and output clock frequency.

If the output format you select is VGA out, one set of output timing values will also be filled automatic (640x480 60Hz). Usually, you should modify them to the correct values as you want.

Step 8) Decide output signal.

CH7025 supports 5 kinds of output video signal types:

(1) TV(CVBS/S-Video).
(2) TV(YPbPr).
(3) TV(RGB+Csync).
(4) VGA(RGB + Hsyns +Vsync + DE).
(5) VGA(RGB + Hsync + Vsync + Composite Sync).

In this step, just select the output signal you want CH7025 to send out.

Step9) Accept your select item.

After you confirm your selection, press “ACCEPT” button, which will update register settings.

b) Interlaced video signal.

If the input of CH7025 is interlaced video signal, the timing should be same as TV format, which the CH7025_RegSet.exe will tell you.

Step1) Make “Interlaced Input” check box selected.

Step2) Decide your crystal frequency.

Please refer to Step3) of a) Normal progressive signal.

Step3) Decide your input data format.

Please refer to Step4) of a) Normal progressive signal.

If your input is standard ITU656 TV format, please select “8-bit YCbCr 4:2:2” or “10-bit YCbCr 4:2:2” item from “input data format” combo box, after this, the “ITU656 Format” check box will be enabled, make it selected.

Step4) Decide if your input data is multiple-rate to pixel rate.

If you select “ITU656 Format” in last step, skip this step.

Please refer to Step5) of a) Normal progressive signal.

Step5) Decide your output video format of CH7025.

Please refer to Step6) of a) Normal progressive signal.

In this condition, “VGA out” and “VGA out (bypass scaler)” can not be selected which the tool will tell you.

Step6) Decide output signal.

In this condition, CH7025 supports 3 kinds of output video signal types:

(1) TV(CVBS/S-Video).
(2) TV(YPbPr).
(3) TV(RGB+Csync).

Just select the output signal you want CH7025 to send out.

Step7) Accept your select item.

After you confirm your selection, press “ACCEPT” button, which will update register settings.

c) CPU interface video signal.

CH7025 supports CPU interface video signal, which requires signal: VSync(frame sync), CSB(chip select), WEB(Write enable) and data pins.

Step1) Make “CPU Interface” check box selected.

Step2) Decide your input timing information.

Just as the tool show you, in this step you need just fill HAI and VAI, which describe the video frame size.

Step3) Decide your crystal frequency.

In this condition, crystal **must** be used. If your crystal frequency is in the pre-defined frequencies, select it in the “Crystal Freq” combo box, otherwise fill it in the edit box.

Step4) Decide your input data format.

Please refer to Step4) of a) Normal progressive signal.

Step5) Decide if your input data is multiple-rate to pixel rate.

Please refer to Step5) of a) Normal progressive signal.

Step6) Decide your output format.

Please refer to Step6) of a) Normal progressive signal.

Step7) Decide output timing information.

Please refer to Step7) of a) Normal progressive signal.

Step8) Decide your output signal.

Please refer to Step8) of a) Normal progressive signal.

Step9) Accept your select item.

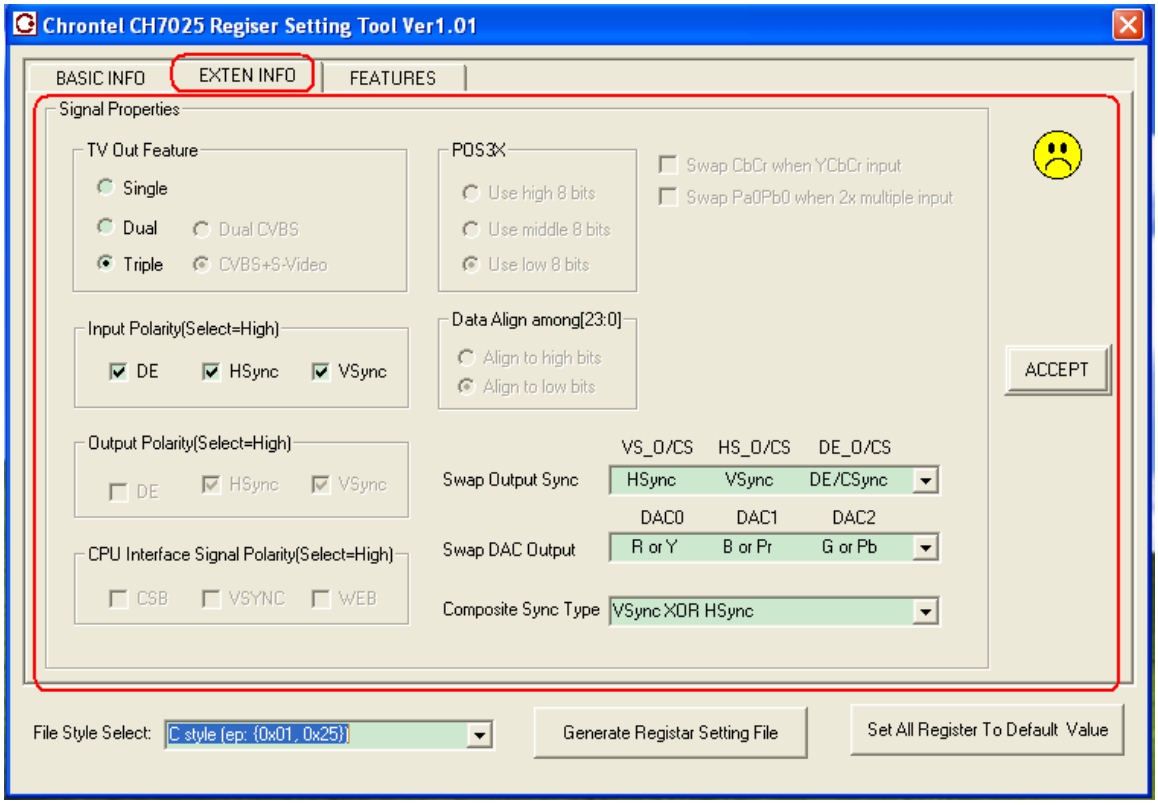
After you confirm your selection, press “ACCEPT” button, which will update register settings.

Now you finish the setting of “BASIC INFO”. Actually, although the sequence of the steps above is recommended, you don’t have to follow it exactly. You can set the information with a different sequence if you like. For some condition, some units will be useless, so these useless units will be disabled automatic.

Press the “ACCEPT” button is to update register settings. Whenever you change your selection, please press this button to update register settings. The change from cry-face to smile-face indicates that register settings have been updated.

2) Set extended information:

The interface of extended information is as following:



Extended information tab-page mainly includes sync and DAC information.

Items introduction:

"TV Out Feature":

Select number of output channels. There are three options: Single, Dual and Triple. Because CH7025 has three DACs, following is the relationship between DAC and TV output signal.

For CVBS, one DAC is needed.
For S-Video, two DACs are needed.
For YPbPr, three DACs are needed.

So, "Single" means that CH7025 can send CVBS or S-Video signal or YPbPr signal; "Dual" means that CH7025 can send two CVBS signals or one CVBS signal and one S-Video signal; "Triple" means that CH7025 can send three CVBS signals.

You could select the corresponding option as you want. When you select “Dual”, you also can choose two CVBS signals or one CVBS signal and one S-Video signal will be the output signals of CH7025, default is CVBS + S-Video.
If the output format of CH7025 is VGA out, “TV Out Feature” will be disabled automatic.

“Input Polarity”:

This means the polarity of input signals including DE, Hsync and Vsync. According the input signal of CH7025, make them checked or un-checked. Checked means polarity is high voltage level.

If the input signal of CH7025 is TV format signal (BT656) or is through CPU interface, “Input Polarity” will be disabled automatic because there is no sync signal flow into CH7025.

“Output Polarity”:

This means the polarity of output signals including DE, Hsync and Vsync. According the output signal of CH7025, make them checked or un-checked. Checked means polarity is high voltage level.

If the output signal of CH7025 is VGA out, “Output Polarity” will be disabled automatic.

“CPU interface signal polarity”:

This means the polarity of input signals when using cpu interface input, including CSB, VSYNC and WEB. According the polarities of these signals, make them checked or un-checked. Checked means polarity is high voltage level.

If the input is not through CPU interface, “CPU interface signal polarity” will be disabled automatic.

“POS3X”:

This is used when input clock frequency is 3x pixel rate. You can select which 8 bits among 24 bits of data will be used.

“Data Align Among [23:0]”:

This is used when input clock frequency is 2x pixel rate. You can select bit 0 – bit 15 or bit 16 – bit 23 among 24 bits of data will be used.

"Composite Sync Type":

This is used when you want CH7025 send composite sync signal. There are 8 kinds of composite sync to be selected. Select the one you want from the combo box.

"Swap Output Sync":

This is used when you want exchange the signification of the three pins: VS_O/CS, HS_O/CS and DE_O/CS. Select the one you want from the combo box. (Check CH7025 datasheet for details)

"Swap DAC Output":

This is used when you want exchange the signification of the three DACs: DAC0, DAC1 and DAC2. Select the one you want from the combo box. (Check CH7025 datasheet for details)

"Swap CbCr when YCbCr input"

This is used when input of CH7025 is YCbCr format. Make this check box selected will swap Cb and Cr.

"Swap Pa0Pb0 when 2x multiple input"

This is used when input of CH7025 is 2x-multipled format. In default, CH7025 assume that first data is the high bits of the pixel, and second is low bits of the pixel, if your data flow into CH7025 is not follow the default sequence, make this check box selected.

Usage:

Step1)

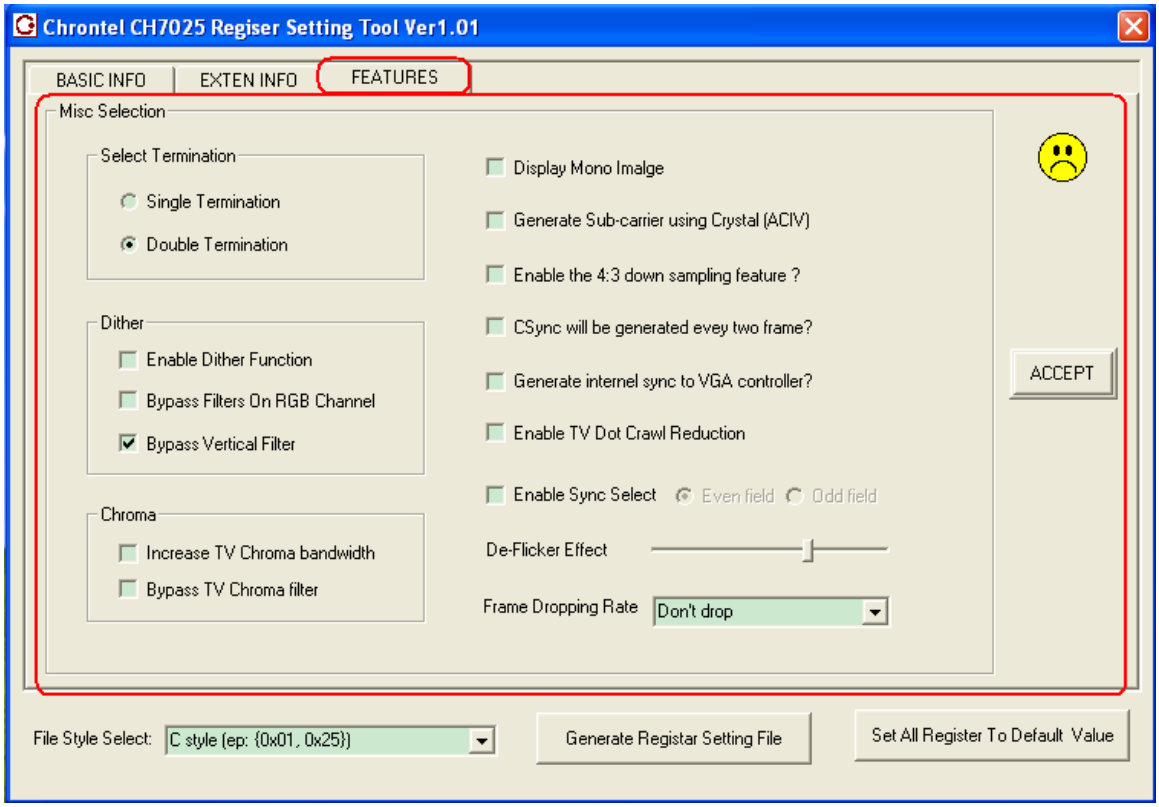
The usage is very simple, according your condition, make exact selection. Please refer to CH7025 data sheet for details.

Step2)

Press the "ACCEPT" button in the right to update register settings. The change from cry-face to smile-face indicates that register settings have been updated.

3) Set features:

The interface of features is as following:



Features tab-page includes information about the features of CH7025.

Items introduce:

"Select Termination":

There are two options: Single termination and double termination. CH7025 will use double termination option default, which bring stable image. You may select one according your condition.

"Dither":

There are three items: "Enable Dither Function" controls if you want to enable dither function; "Bypass filter on RGB channel" controls if you want bypass the adaptive filter on RGB channel; "Bypass Vertical Filter" controls if you want to bypass the vertical adaptive filter.

"Chroma":

There are two items: “Increase TV chroma bandwidth” controls if you want to increase the bandwidth of chroma. Wide bandwidth brings high chroma. “Bypass TV chroma Filter” controls if you want to bypass the chroma filter.

“Display Mono Image”:

This check box is used if you want CH7025 send mono image to display on TV set or VGA display.

“Generate Sub-carrier using Crystal (ACIV)”:

This check box is used when the image CH7025 send out has no color or the color is not correct. Usually this problem is caused by the input clock (GCLK) which has no good quality. In default, CH7025 use GCLK to generate sub-carrier frequency, so a bad GCLK will cause no color or incorrect color. So CH7025 provide an option that you can control CH7025 to generate sub-carrier using the crystal if there is one has been connected to CH7205. If you want use the feature, make this check box checked.

“Enable the 4:3 down sampling feature”:

This check box is used when you want to enable the 4:3 down sampling feature, please check CH7025 datasheet for details.

“Csync will be generated every two frame”:

This check box is used only when you select the “TV (RGB+Csync)” output signal. When checked, Csync will be generated every two frame, otherwise, Csync will be generated every frame.

“Generate internal sync to VGA controller”:

This check box is used when your graphic controller wants to accept sync signal as reference. CH7025 has the function to generate sync signal and send it to graphic controller. Make this check box checked to enable this feature.

“Enable TV Dot Crawl Reduction”:

This check box is used if output signal is TV out. Make it checked to enable this feature.

“Enable Sync Select”:

This check box is used if input format is interlaced. When input of CH7025 is interlaced, sync signals of even field and odd field can all be used. In default, sync signal of even field will be used, you can change it here. First enable this feature by making this check box checked, then select the field you want.

"Frame Drop rate":

This combo box is used to select the frame dropping rate. There are 4 options to be selected. Frame dropping rate means that CH7025 can ignore one frame every two frames (2:1 drop) or every three frames (3:1 drop) or every four frames (4:1 drop). Usually, CH7025 does not ignore any frame (Don't drop). This feature is used when input frame rate is too fast (>100Hz).

"De-flicker Effect":

This slider bar is used to control the de-flicker effect. Scroll to right means stronger de-flicker effect.

Usage:

Step1)

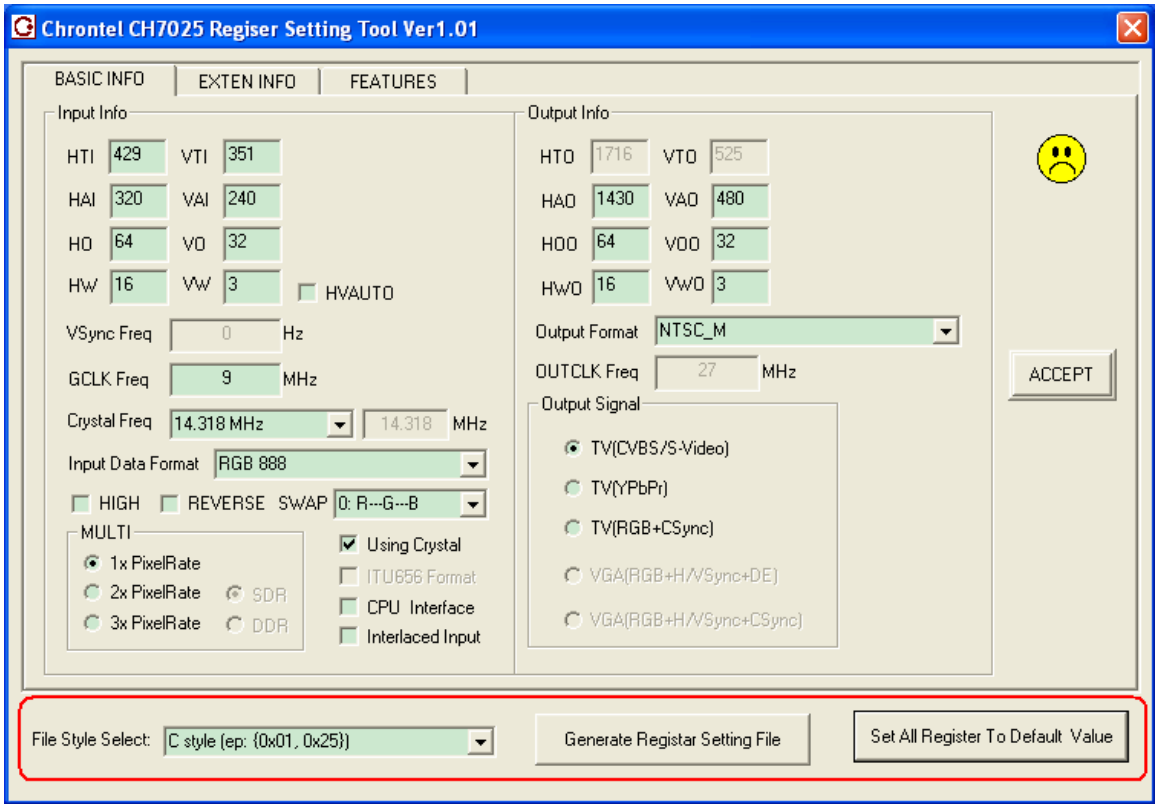
According your condition, make exact selection of the items. Please refer to CH7025 data sheet for details.

Step2)

Press the "ACCEPT" button in the right to update regsiter settings. The change from cry-face to smile-face indicates that register settings have been updated.

4) Generate register map:

Following shows the interface of this section:



Items introduction:

"File Style Select":

This combo box is to select the style of register map file which is to be generated. There are three options: "C style", "Assembly style" and "Chrontel Internal Format". This is used to make it easy to write code for using CH7025.

"Generate Register Setting File":

Press this button to generate a register map file. According "File Style Select", C style file or assembly style file named "CH7025_RegMap.rem" will be generated and opened. You can copy the content of the file into your code. Thus it is so easy to set CH7025 to make it work correctly.

An example is provided at the end of this chapter to show how to using the generated register map file to set CH7025.

"Set All Registers To Default Values":

Press this button to cancel your setting on the tools. This button is used when you want to set CH7025 again. Press this button and go back to the beginning of this chapter.

Usage:

Step1) Decide the format of regmap file to be generated.

CH7025_RegSet.exe provide three items. If you are using C language for your development, select "C Style"; if you are using assembly language, select "Assembly Style"; "Chrontel internal format" is used for Chrontel internal test.

Step2) Press the "Generate Register Setting File" button.

After press button, register setting file has been generated according your selection in the steps above. Please check "Example code" below for how to use this register setting file.

Here, you have finished main setting of programming CH7025. Actually, the target in this chapter is only to get the register setting file. Now, go to next chapter------Chapter 4: Power up CH7025.

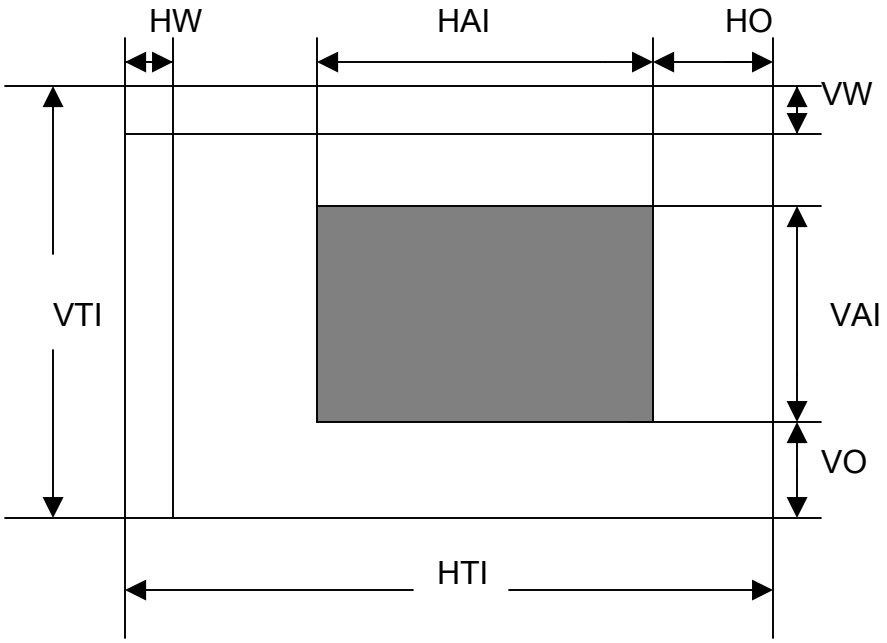
Example code:

```
// Using the register map file:
// Assume that you select "C style" and the red below is copied from the
// register map file:

unsigned char reg_map[][2] = {
        { 0x00, 0x00 }, // red section is copied from CH7025_RegMap.rem.
        { 0x01, 0x01 },
        { 0x02, 0x02 },
        ...
        { 0x7D, 0x7D },
        { 0x7E, 0x7E },
        { 0x7F, 0x7F },
};

// Write registers:
int index = 0;
int size = sizeof ( reg_map ) / ( 2 * sizeof ( unsigned char ) );
for ( index= 0;index< size; ++index)
```

```
{
        I2CWrite ( reg_map[index][0], reg_map[index][1];
}
```

Hint1:

Chrontel timing definition:



Chapter 4:

POWER UP CH7025

Register 04h is the power state register.

Address: 04h

BIT:	7	6	5	4	3	2	1	0
SYMBOL:								FPD
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

FPD (bit0) controls the power on/off state. When FPD is “0”, the CT318 is in power-up state, when FPD is “1”, the CT318 is in power-down state. At power-down state, the CT318 accepts SPP access.

Usage:

Only set FPD to ‘0’.

Sample code:

```
unsigned char val = 0;

val  = I2CRead ( 0x04 ); // Read the Pre-value of register 04h

val &= 0xFE;

I2CWrite ( 0x04, val ); // Set FPD to ‘0’.
```

Chapter 5:

MEMORY INITIALIZATION

Address: 06h

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

MEMINIT (bit 1) is to set SDRAM in initialization state. When this bit is set to “0”, the SDRAM initialization sequence is beginning.

Address: 7Eh

BIT:	7	6	5	4	3	2	1	0
SYMBOL:					DVALID			
TYPE:	R	R	R	R	R	R	R	R
DEFAULT:	0	0	0	0	0	0	0	0

Register 7Eh is the SDRAM status register. It is read-only register.
DVALID (bit 3): when DVALID=1, SDRAM initialization is completed; otherwise SDRAM initialization is not completed.

Usage:

1) Set MEMINT bit to “1”.
2) Set MEMINT bit to “0”.
3) Read the value of DVALID.
4) When DVALID=0, the SDRAM initialization is not completed, you should wait for some time. Then go back to 3) above.
5) Repeat 3) and 4) until DVALID=1.

Sample code:

```
unsigned char val = 0;

val = I2CRead ( 0x06 );

val |= 0x02;
```

```
I2CWrite ( 0x06, val ); // Set MEMINIT to '1'

val &= 0xFD;

I2CWrite ( 0x06, val ); // Set MEMINT to '0'

While (1)  // You could set wait time here
{
        if ( I2CRead( 0x7E ) & 0x08 ) // check DVALID bit
        {
                // SDRAM initialization complete
                break;
        }
        // Wait for a while
}
```

Chapter 10:

START RUNNING

Address: 06h

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

STOP (bit 0) is to stop the scaler and SDRAM control operation. When this bit is "1", these logics are stopped; when this bit is "0", these logics work normally.

Usage:

Only set STOP bit to "0".

Sample code:

```
unsigned char val = 0;

val = I2CRead ( 0x06 );

val &= 0xFE;

I2CWrite ( 0x06, val ); // Set STOP to '0'.
```

When this step is completed, CH7025 begin to work correctly. You could see images from TV set or VGA display. In addition, you can adjust the quality and properties of the image on the TV set or VGA display, such as position on screen, brightness, zoom function, rotation, flipping and so on. Appendix A introduces the details.

Appendix A:

USING CH7025 FEATURES

CH7025 provides interfaces to adjust the quality and properties of the image on TV set 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 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 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 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 2Eh 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 |= 0x80;
        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 |= 0x40;
        else
                val &= 0xFB;
```

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

```
        vst = pzs->vStart;
        vend = pzs->vEnd;

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

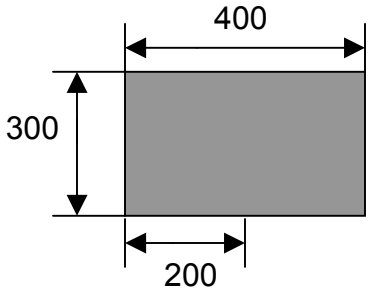
Sample code:

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 };
```

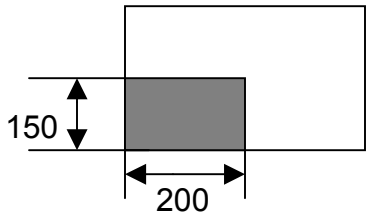
Example 1: zoom 100% image that looks like no zoom:

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



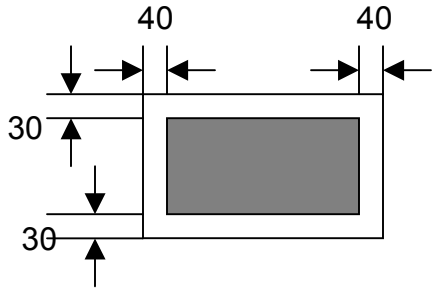
Example 2: zoom down-left quarter of the image:

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



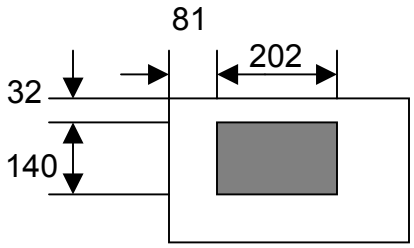
Example 3: zoom center:

```
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 );
```

Appendix B:

WHOLE EXAMPLE

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

Assume:

1) 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]:

Sample code:

```
// BEGIN HERE:

typedef unsigned char       uchar;
typedef unsigned int        uint;

// define variable to store the register map:
unsigned char REG_MAP[ ][2] = {
        { 0x00, 0x00 }, // red section is copied from CH7025_RegMap.rem.
        { 0x01, 0x01 },
        ...
        { 0x44, 0x22 },
        { 0x77, 0x11 },
};
#define REGMAP_LENGTH  ( sizeof( REG_MAP ) / (2 * sizeof ( uchar ) ) )


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

// CH7025 related function declaration:
void Delay ( uint time ); // software delay functon
uint CH7025_CntDtt ( ); // connection detect
uint CH7025_MemInit( ); // wait for memory init complete

#define USING_CONNECT_DETECT
```

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

        // Make sure CH7025 in the system:
        if ( I2CRead ( 0x00 ) != 0x55 ) // DID of CH7025 is 0x55
        {
                return 1; // CH7025 can not be operated through I2C bus
        }

        // CH7025 can operated through I2C bus, go on

        // TV connection detect, this is optional.
#ifdef  USING_CONNECT_DETECT
        if ( ! CH7025_CntDtt( ) )
        {
                return 2; // TV set or VGA display not connected to CH7025.
        }
#endif

        // Write CH7025_RegMap.rem to registers of CH7025:
        for ( ii = 0 ; ii < REGMAP_LENGTH ; ++ii )
        {
                I2CWrite ( REG_MAP[ii][0], REG_MAP[ii][1] );
        }

        // Power up CH7025
        I2CWrite ( 0x04, I2CRead ( 0x04 ) & 0xFE ); // set bit 0 to '0'

        // Memory initializaion:
        if ( !CH7025_MemInit( ) )
        {
                return 3; // Memory init can not be completed.
        }

        // Start running:
        I2CWrite ( 0x06, I2CRead ( 0x06 ) & 0xFE ); // set bit 0 to '0'.

        // Now you can see image on TV set or VGA display.

        return 0; // success!

} // end of main
```

```
// function defination:

// Funtion: software delay.
// Param: time:delay time.
// Return value: none.
void Delay ( uint time )
{
        int ii = 0;
        for ( ii = 0; ii != time; ++ii )
                ;
}

// Function: connection detect
// Param: none.
// Return value: return 1 if success detect, otherwise return 0.
uint CH7025_CntDtt ( )
{
        uint retval = 0;

        int ii = 0;

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

        // as CH7025 NOT power up now, set DISPON to “0”:

        I2CWrite ( 0x7D, I2CRead ( 0x7D) & 0xFD); // Set DISPON to “0”

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

        val  = I2CRead ( 0x7F ); // read detect result

        for ( ii = 0 ; ii !=3; ++ ii )
        {
                dac[ii] =  val & ( 3 <<  ( 2 * ii ) ) ;

                if ( dac[ii] == 1 )

                        retval = 1;
        }

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

        retuen retval;

}
```

```
// Function: wait for memory init complete
// Param: none.
// Return value: return 1 if memory init complete successfully, otherwise return 0.
uint CH7025_MemInit( )
{
        uint retval = 0;

        uint count = 10000; // try times

        I2CWrite ( 0x06, I2CRead ( 0x06 ) | 0x02 ); // Set MEMINT to '1'

        I2CWrite ( 0x06, I2CRead ( 0x06 ) & 0xFD); // Set MEMINT to '0'

        While ( count -- )
        {
                if ( I2CRead ( 0x7E ) & 0x08 )

                        retval = 1;

                Delay ( 1000 ); // any delay function can be accepted here
        }

        return retval;
}


// END HERE
```

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

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.