目录

1. 课程设计题目：

YUV 彩色图像处理之汉字叠加

2. 课程设计基本要求

2.1. 课程设计原理

汉字叠加就是在原始图像相应的行列中加入汉字码，首先利用 PCtoLCD 字模软件取出相应的字库，然后根据汉字码中的值，判断每个字节的每一位是否为1,如为1则将图像中的值设定为255,改变成颜色，为 0 则不变，在这里要注意的是原始图像的大小。彩色图像的产生是由 matlab 进行取点，生成相应的.dat，有 CCS 软件仿真图形。

2.2. 课程设计工具

Code Composer Studio V3.3 白金版软件，计算机，DSP 硬件仿真器；SZ-DSPF 开发教学平台

2.3. 课程设计硬件设置

先将实验箱上的电源开关“MS2”“MS3”、“MS4”和“MS5”按下，再将机箱右侧的船型开关往“I”方向打开电源；SZ-5416D 主控模块上的 J2、J4、J7，J9，J16 短接；在“设置模块”中将“A”和“C”设置为“1”。然后开始做实验，注意在做 DSP 实验时开始按了 SZ-5416D 主控模块上的 K1 硬件复位后，程序运行中不要再按复位键，以免实验由于 DSP 复位而失败。

3. 课程设计内容

3.1 Code Composer Studio V3.3 软件的安装

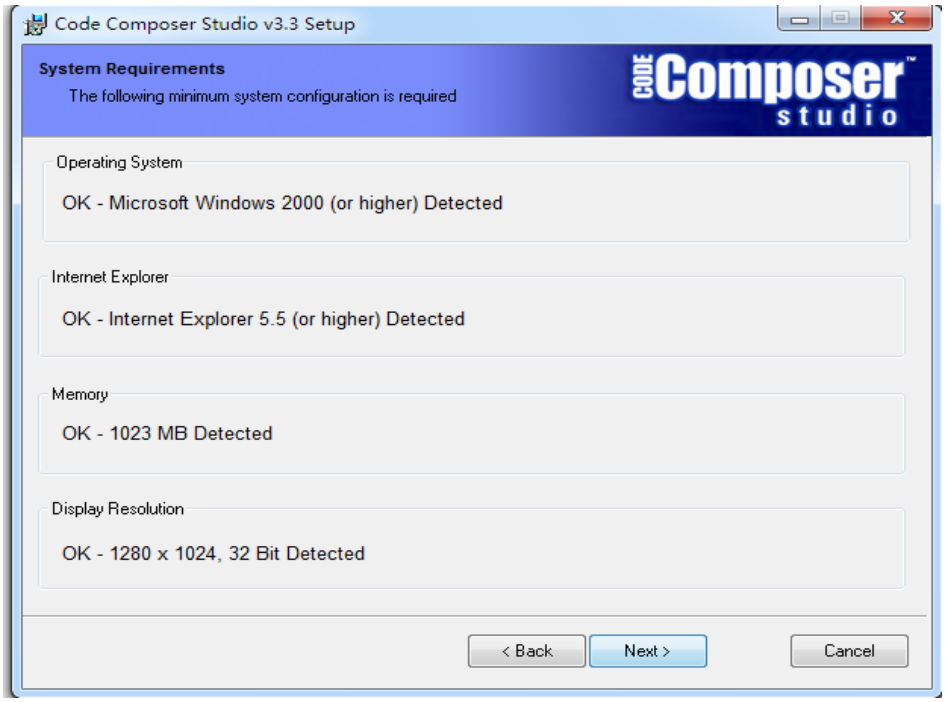
3.1.1 Code Composer Studio V3.3 简介

Code Composer Studio V3.3 白金版支持多处理器运行将分析特性提高到新的水平，可不断满足高级嵌入式系统开发发展的需求。统一的新型断点管理器、缓存状态可视化工具，完全集成的分析系统和代码覆盖功能，CCS V3.3 为 DSP 开发人员提供了强大的工具，能更高效地分析系统运行状态，减少开发工作，从而加速新产品上市场进程。CCSV3.3 支持 TI 的 TMS320C6000、TMS320C5000 与 TMS320C2000 DSP 平台外，CCS v3.3 还能更好地显示 ARM 处理器的存储器使用情况，这对采用基于达芬奇技术的多处理器系统的开发人员来说尤其有用。ARM 存储器管理单元(MMU)的表格化显示功能可反映物理与虚拟地址情况，并提供了保护信息显示完整的地址映射。过滤与排序功能则令编程人员能有重点的检查域、过程或存储域，以进行深入具体的分析。

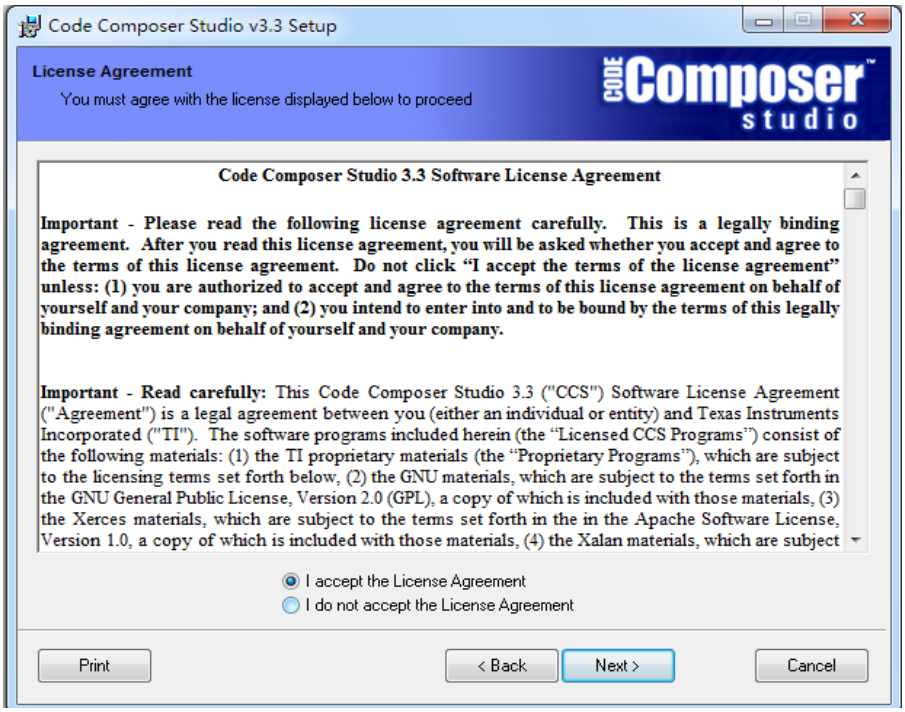
3.1.2 Code Composer Studio V3.3 在 Win7 下的安装

- 打开 CCS3.3 的安装文件夹，选中 Setup.exe，右击选择“以管理员身份运行”，出现下图

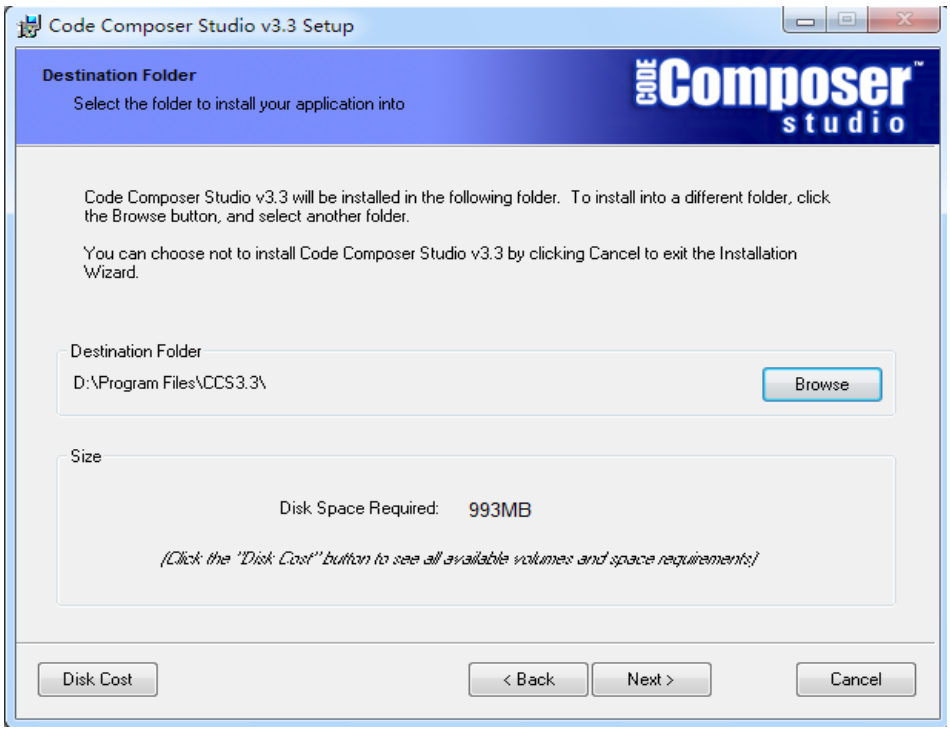
➢ 点击“Next”，出现下图界面



- 点击“Next”，出现下图，选择“I Accept the License Agreement”，点击“Next

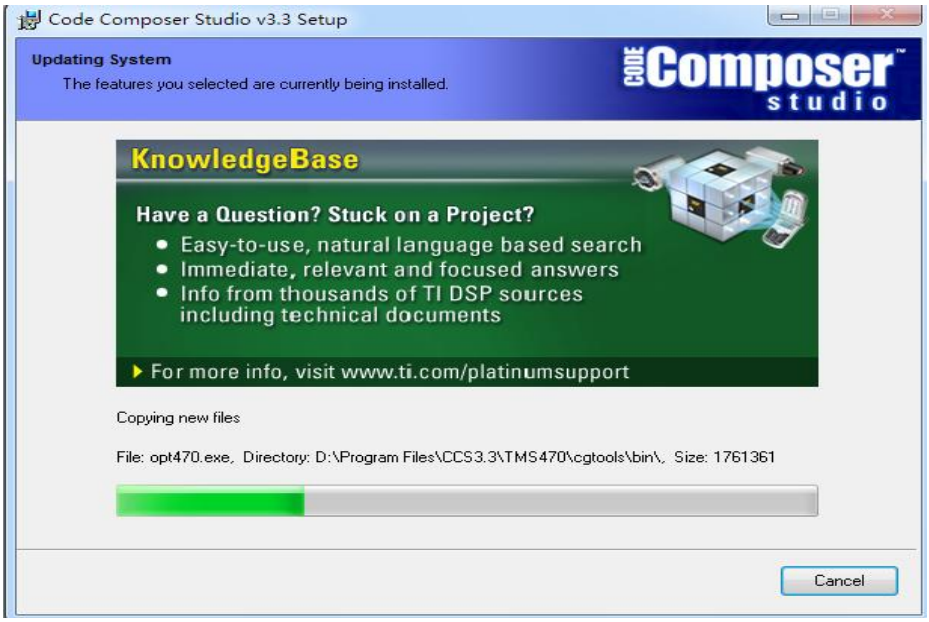


- 点击“Browse”选择安装路径，点击“Next”

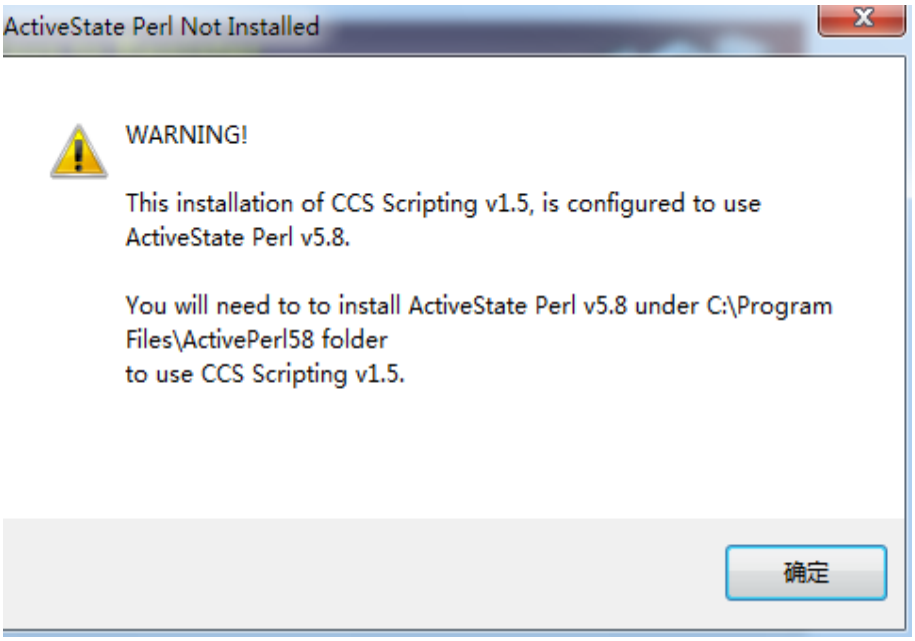


➢ 点击“Install Now”

➢ 出现下图安装进程界面

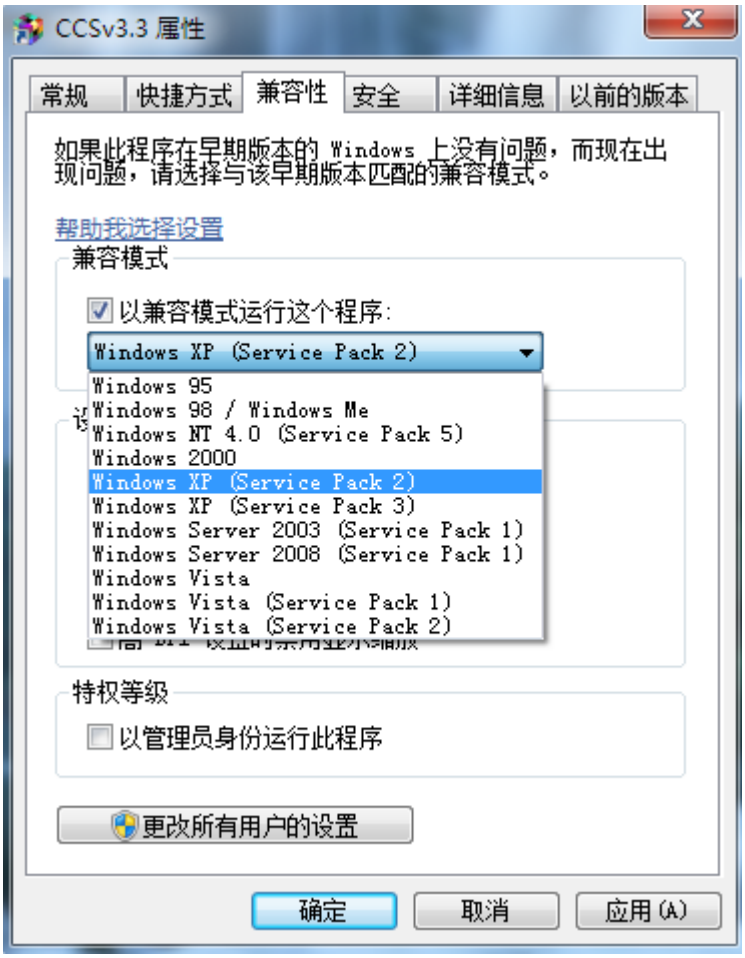


- 若安装过程出现 Warning，点击“确定”



- 安装完成，点击“Finish”

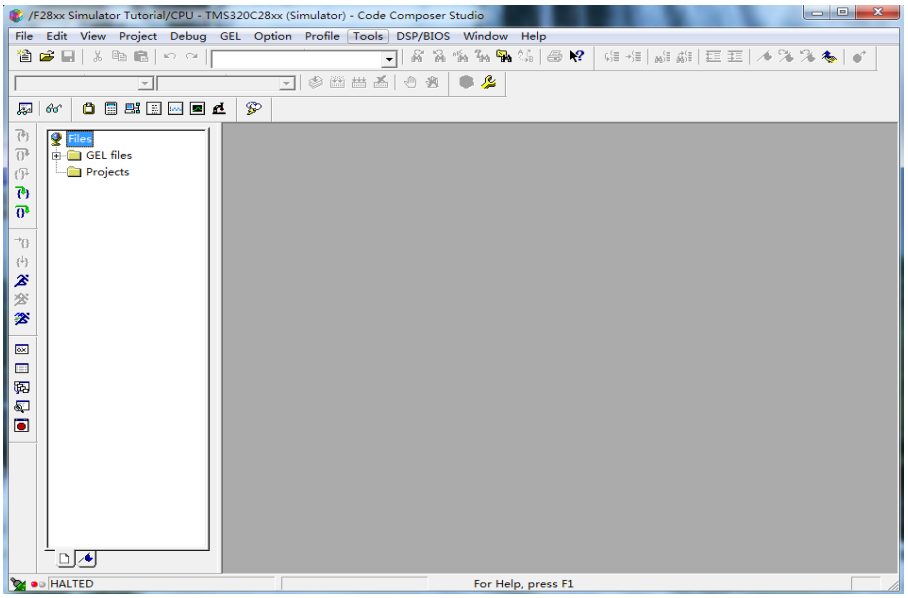
- 由于 Win7 不兼容 CCS3.3，因此要修改一下兼容性，点击“确定”就 OK 了



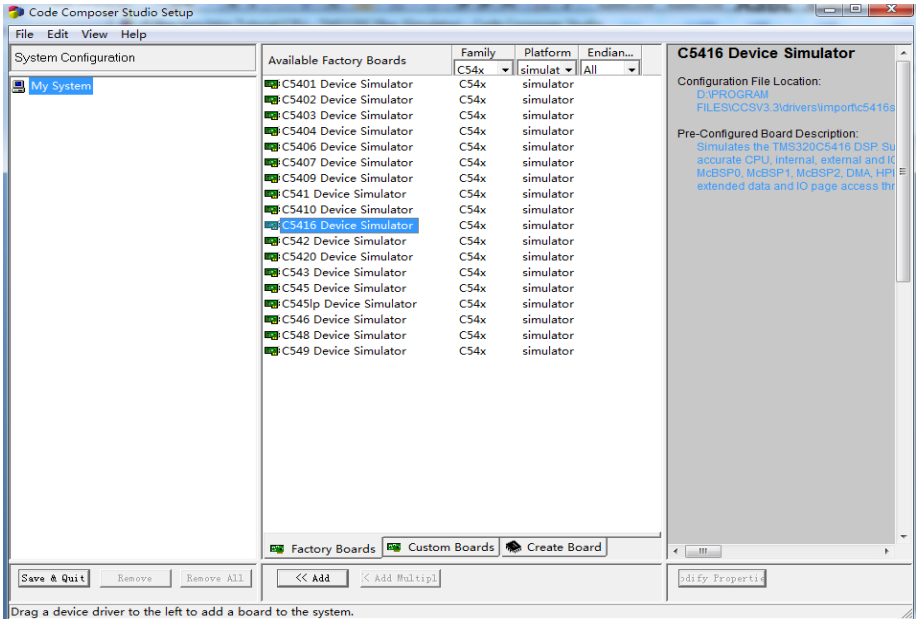
3.2 Code Composer Studio V3.3 的使用

3.2.1 Code Composer Studio V3.3 的启动和设置

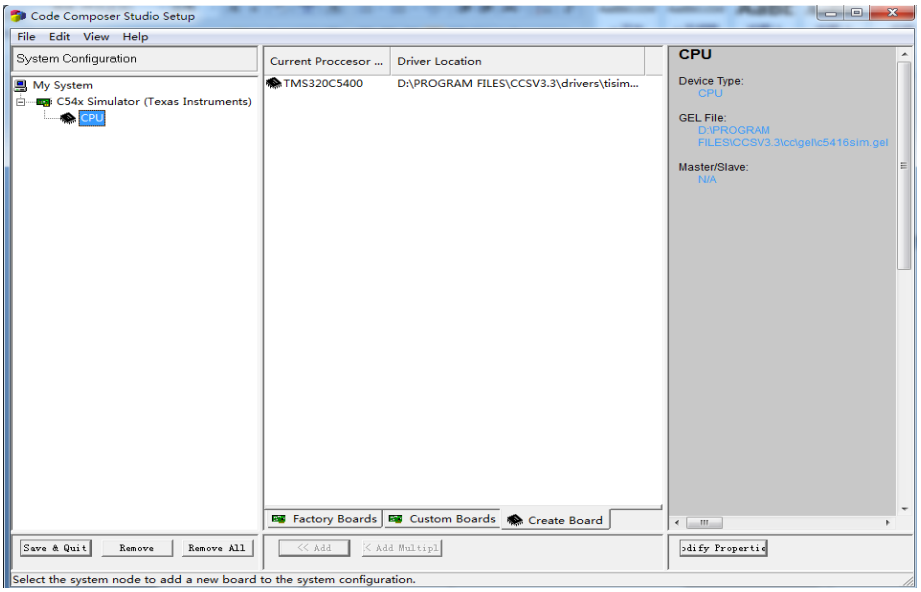
- 双击桌面的 CCSV3.3. ico  ，进入 CCS 环境



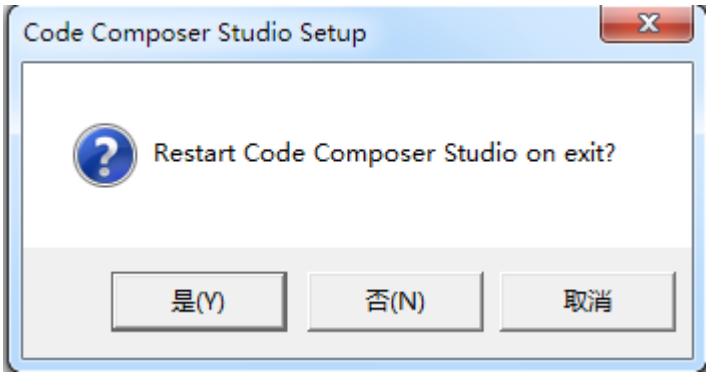
➢ 选择 File->Launch Setup,进入下图



➢ 双击“C5416 Device Simulator”,进行适当的 System Configuration，点击”Save&Quit”

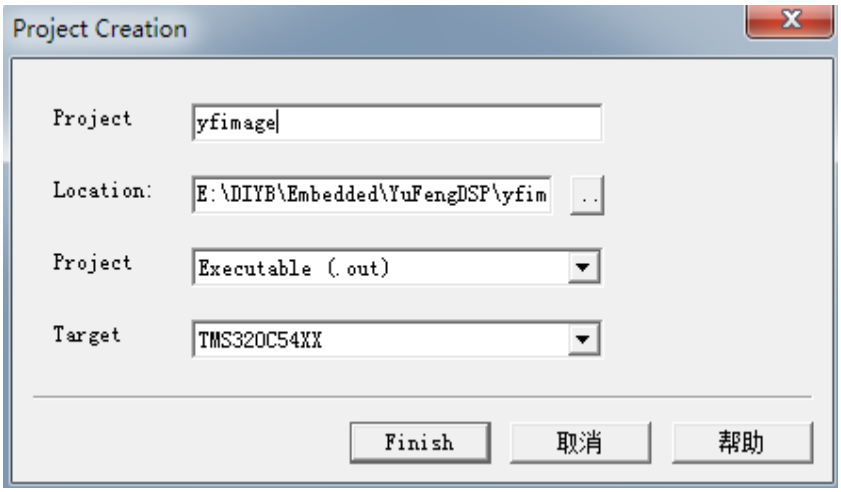


- 出现下图，选择“是”

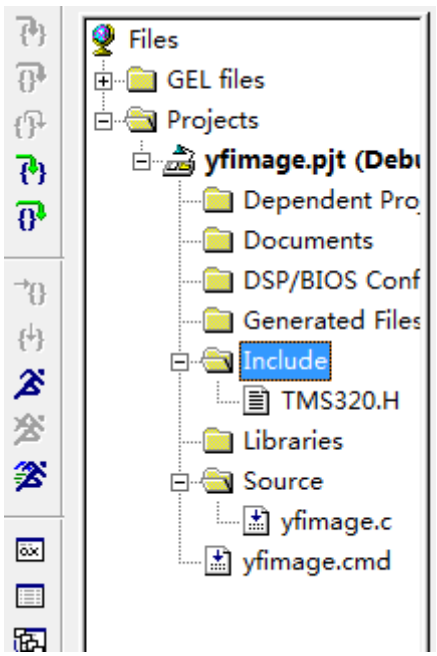


3.2.2 Code Composer Studio V3.3 新建一个工程

- 选择“Project ->New”,进行如下设置（Note：Location 不能有中文），点击 Finish

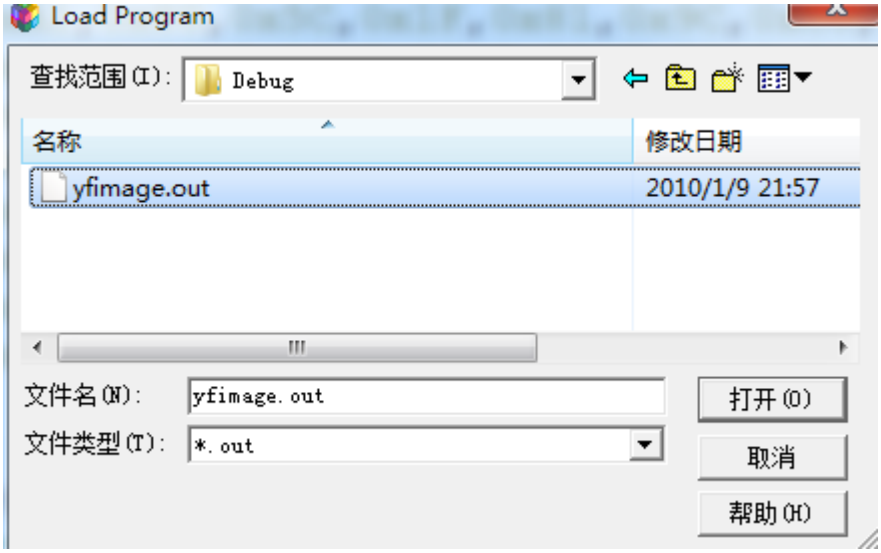


- 在“Project Window”里进行添加文件

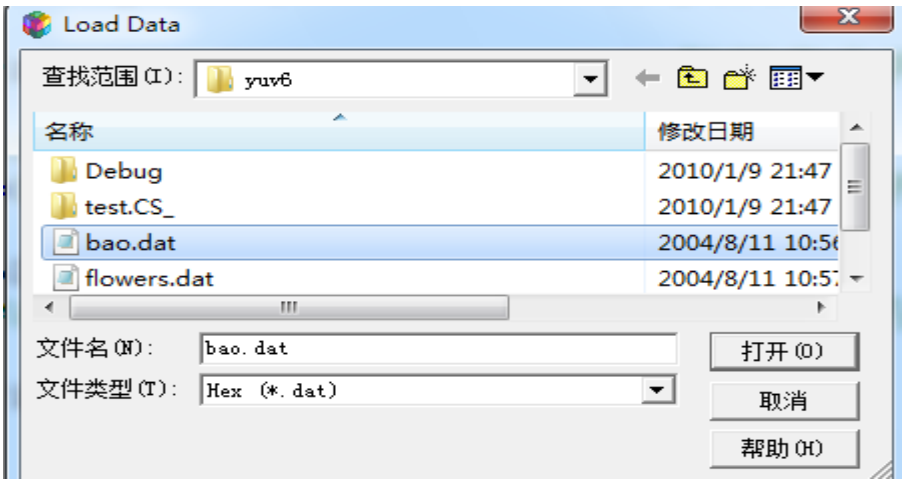


- 进行编译，选择”Project->Built All”或者选择红色按钮 

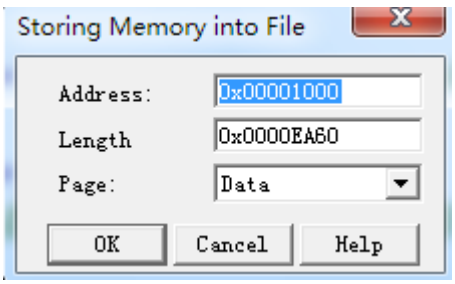
- 下载程序，“File->Load Program”，选择“yfimage.out”，点击“打开”



➢ 下载数据“File->Data->Load Data”,选择“Bao.dat”，并打开

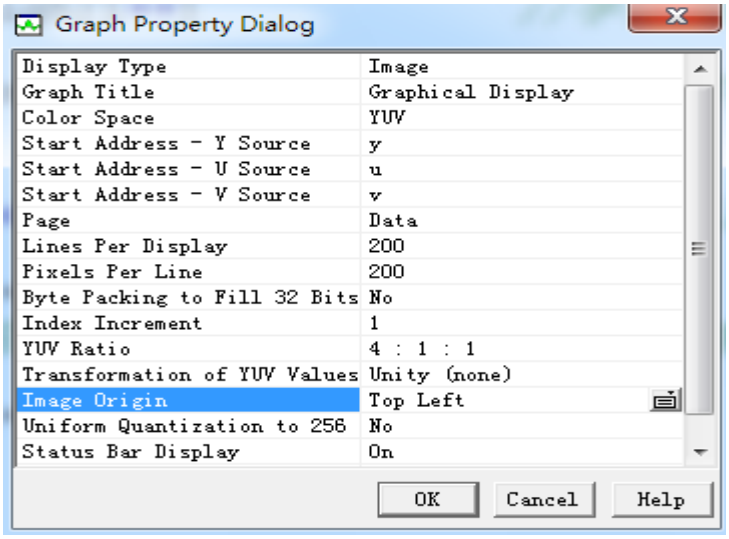


➢ 显示加载数据的首地址，数据长度，存放区域点击 OK



➢ 点击“View->Graph->Image”进行如下设置，并点击“OK”，出现豹

子



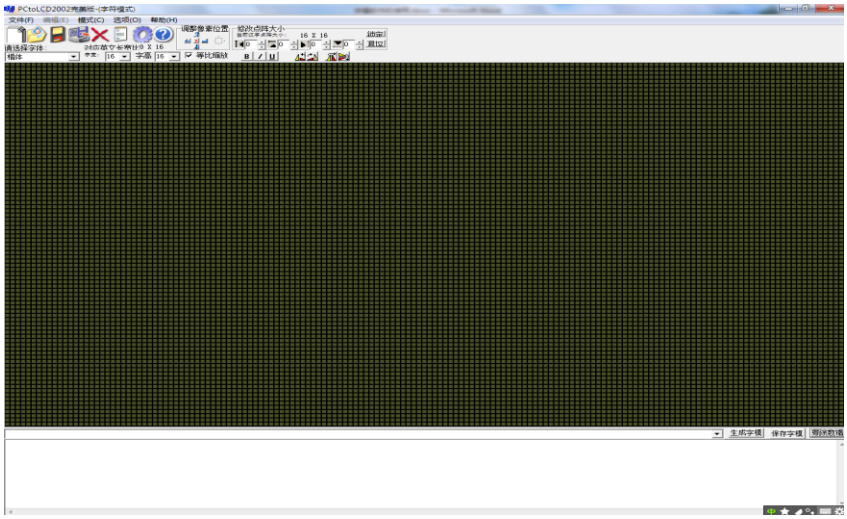
- 调试“Debug->Go Main”

```
main()
{
  hzplay(16,8,40,0);
  for(;;)
  {}
}
```

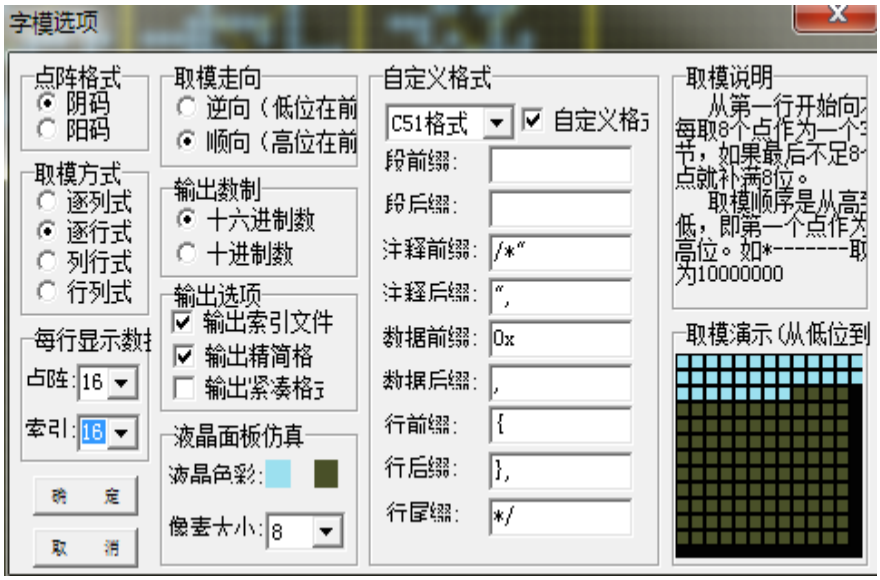
➢ 选择“Debug->Run”或者点击

3.3 字模软件 PCtoLCD 的使用

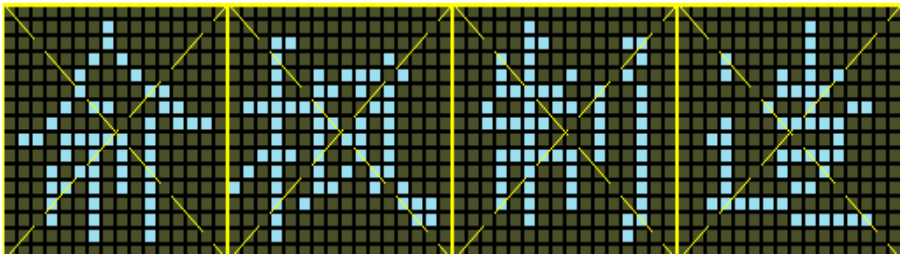
➢ 双击字模软件 PCtoLCD2002.exe，进入



➢ 单击选项，出现设置，进行如下设置（注意设置选项与程序代码对应）



- 输入**“俞枫制造”**四个字，生成相应字模代码，并保存



代码如下：

0x00,0x00, 0x00,0x00, 0x00,0x00, 0x00,0x00,
0x01,0x00, 0x00,0x00, 0x04,0x00, 0x00,0x40,
0x01,0x00, 0x18,0x00, 0x06,0x0C, 0x00,0x40,
0x02,0x80, 0x10,0x10, 0x14,0x04, 0x10,0x40,
0x04,0x40, 0x12,0xE8, 0x14,0x88, 0x11,0x50,
0x04,0x30, 0x13,0x50, 0x1F,0x24, 0x01,0xE0,
0x0B,0x18, 0x7E,0x30, 0x25,0xA4, 0x02,0x4C,
0x10,0x2E, 0x13,0x50, 0x3E,0x24, 0x11,0xF0,
0x6F,0xA0, 0x12,0xD0, 0x04,0x24, 0x30,0x10,
0x0A,0xA0, 0x3A,0x50, 0x1F,0xA4, 0x11,0xF0,
0x16,0xA0,0x52,0xB0, 0x14,0x84, 0x11,0x20,
0x1A,0xA0, 0x93,0x28, 0x14,0x84, 0x10,0xC0,
0x12,0x20, 0x14,0x0E, 0x14,0x84, 0x3F,0x00,
0x12,0x20, 0x10,0x06, 0x04,0x0C, 0x00,0xFC,

```
0x02,0x20, 0x10,0x00, 0x00,0x08, 0x00,0x00,
0x00,0x00, 0x00,0x00, 0x00,0x00, 0x00,0x00,
```

3.4 程序设计

- 头文件设计：TMS320.h

```
#ifndef _TMS320
#define _TMS320
typedef unsigned int   uint;
typedef unsigned short ushort;
typedef short DATA;
typedef long LDATA;
#define PASS -1
#define ABSVAL abs
#define SHIFT15 >>15
#define SHIFT1  /2
#define ROUND 0x400
#define DIV2   >>1
#endif
```

- Coff 目标文件设计：yfimage.cmd

```
-c
-l rts.lib
-stack 0x200
-heap 0x200
MEMORY
{
 PAGE 0:
```

```
INT_PM_DRAM:    origin = 0080h, length = 1380h
EXT_PM_RAM:     origin = 1400h, length = 01000h
 PAGE 1:
INT_DM_SCRATCH_PAD_DRAM:     origin = 0060h, length = 20h
INT_DM_1:                    origin = 0080h, length = 0f80h
EXT_DM_RAM:                  origin = 1000h, length = 0ef00h
}
SECTIONS
{
        .text    :     {} > INT_PM_DRAM  PAGE 0
      // 可执行代码和浮点常数
        .cinit   :     {} > INT_PM_DRAM   PAGE 0
      //已初始化的全局和静态变量表
        .switch  :     {} > INT_PM_DRAM   PAGE 0
      //Switch 语句的跳转表
        .data    :     {} > INT_DM_1      PAGE 1//数据区
        .stack   :     {} > INT_DM_1      PAGE 1//软件堆栈
        .bss     :     {} > INT_DM_1      PAGE 1//全局和静态变量
        .sysmem  :     {} > INT_DM_1      PAGE 1
      //malloc 函数的动态存储区
        .const   :     {} > INT_DM_1      PAGE 1//字符串
        .dout    :     {} > INT_DM_1      PAGE 1//
        .x       :     {} > EXT_DM_RAM    PAGE 1//自定义
 }
```

- 源文件的设计：yfimage.c

```
#include "TMS320.H"
#include  "math.h"
#pragma DATA_SECTION (y,".x")
DATA y[40000];
#pragma DATA_SECTION (u,".x")
DATA u[10000];
#pragma DATA_SECTION (v,".x")
DATA v[10000];

DATA tab1[]={
//;--  俞枫制造  --  **  楷体_GB2312, 16  **
//; 当前所选字体下一个汉字对应的点阵为:  宽度 x 高度=64x16,  调整后为: 64*16
0x00,0x00,  0x00,0x00,  0x00,0x00,  0x00,0x00,
0x01,0x00,  0x00,0x00,  0x04,0x00,  0x00,0x40,
0x01,0x00,  0x18,0x00,  0x06,0x0C,  0x00,0x40,
0x02,0x80,  0x10,0x10,  0x14,0x04,  0x10,0x40,
0x04,0x40,  0x12,0xE8,  0x14,0x88,  0x11,0x50,
0x04,0x30,  0x13,0x50,  0x1F,0x24,  0x01,0xE0,
0x0B,0x18,  0x7E,0x30,  0x25,0xA4,  0x02,0x4C,
0x10,0x2E,  0x13,0x50,  0x3E,0x24,  0x11,0xF0,
0x6F,0xA0,  0x12,0xD0,  0x04,0x24,  0x30,0x10,
0x0A,0xA0,  0x3A,0x50,  0x1F,0xA4,  0x11,0xF0,
0x16,0xA0,  0x52,0xB0,  0x14,0x84,  0x11,0x20,
0x1A,0xA0,  0x93,0x28,  0x14,0x84,  0x10,0xC0,
```

```
0x12,0x20,  0x14,0x0E,  0x14,0x84,  0x3F,0x00,
0x12,0x20,  0x10,0x06,  0x04,0x0C,  0x00,0xFC,
0x02,0x20,  0x10,0x00,  0x00,0x08,  0x00,0x00,
0x00,0x00,  0x00,0x00,  0x00,0x00,  0x00,0x00,

};
//;--  俞枫制造  --  **  楷体_GB2312, 16  **
//; 当前所选字体下一个汉字对应的点阵为:  宽度 x 高度=64x16,  调整后为: 64*16

DATA tab1[]=
{
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x08,0x00,
0x00,0x00,0x04,0x00,0x02,0x10,0x04,0x20,
0x00,0xF0,0x04,0x00,0x04,0x10,0x04,0x20,
0x07,0x80,0x08,0x00,0x1C,0x50,0x08,0x38,
0x00,0x00,0x0B,0x00,0x04,0x10,0x06,0xE0,
0x00,0x00,0x14,0x0C,0x0F,0xD0,0x38,0x40,
0x00,0x60,0x24,0x76,0x7C,0x50,0x0A,0x50,
0x07,0x80,0x07,0xC4,0x0E,0x1E,0x05,0xF0,
0x00,0x00,0x7C,0x44,0x15,0xF0,0x19,0x10,
0x00,0x00,0x0A,0x4C,0x24,0x10,0x68,0xE0,
0x00,0x1C,0x09,0x70,0x24,0x10,0x08,0x20,
0x3F,0xE6,0x11,0x00,0x04,0x10,0x08,0x50,
```

```
0x00,0x00,0x20,0x00,0x04,0x10,0x0B,0x8E,
0x00,0x00,0x00,0x00,0x00,0x10,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x10,0x00,0x00
};
void hzplay(char high,char size, char ypos,char xpos)
//本程序是一个通用程序，只要用户输入字体高度、尺寸、列地址、行地址就可
{
   char i,j,k,s1,a;
   s1=0x80;
   for(i=0;i<high;i++)              //字体高度
   {//3
    for(j=0;j<size;j++)               //每行多少字节 per line 8 byte
    {//2
    s1=0x80;
    for(k=0;k<8;k++)                 //每行八位不变per byte 8 bit
    {//1
    a=(tab1[i*size+j]&(s1>>k));
    //取出相应的点数据＝行数*位数+每行的字节数
    if(a)
    y[200*i+j*8+k+ypos+200*xpos]=255;
    }//1
    }//2
```

```
    }//3
}
main()
{
   hzplay(16,8,40,0);
   for(;;)
   {}
}
```

4. 课程设计总结

心得体会

课程设计的任务一般是设计并调试一个简单的 YUV 彩色图像处理之汉字叠加，需要我们综合运用《DSP 技术及应用》课程的知识，通过查阅资料、方案论证与选定，设计分析指标及讨论，完成设计任务。

在这次课程设计中，我学会了怎样去根据课题的要求去设计调试程序，动手能力得到很大的提高。从中我发现自己并不能很好的去使用我所学到的知识，在以后学习中我要加强对程序的设计能力的培养。通过这次课程设计，我能掌握工程设计的步骤和方法，了解科学实验的程序和实施方法，这对今后从事技术工作无疑是个启蒙训练。通过这种综合训练，我提高动手组织实验的基本技能，培养分析解决问题的实际本领，为以后毕业设计和从事实际工作打下基础。同时也让我充分认识到理论与实践的差别，认识到只有从本质上理解了原理，才能更好的实现设计。

最后，感谢张老师的支持和教导，期待新的征途。